PHISHING DETECTION:
METHODS BASED ON NATURAL LANGUAGE PROCESSING

ÉDER SOUZA GUALBERTO

TESE DE DOUTORADO EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DETECÇÃO DE PHISHING: MÉTODOS BASEADOS EM PROCESSAMENTO DE LINGUAGEM NATURAL

PHISHING DETECTION: METHODS BASED ON NATURAL LANGUAGE PROCESSING

ÉDER SOUZA GUALBERTO

ORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR, DR.

TESE DE DOUTORADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGEE.TD-170/20

BRASÍLIA/DF: DEZEMBRO - 2020

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PHISHING DETECTION: METHODS BASED ON NATURAL LANGUAGE PROCESSING

ÉDER SOUZA GUALBERTO

TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR.

APROVADA POR:

BRASÍLIA, 15 DE DEZEMBRO DE 2020.

FICHA CATALOGRÁFICA

GUALBERTO, ÉDER SOUZA
Phishing Detection: Methods Based on Natural Language Processing [Distrito Federal] 2020.
xxiv, 130p., 210 x 297 mm (ENE/FT/UnB, Doutor, Engenharia Elétrica, 2020).
Tese de doutorado – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Phishing Detection	2. Word Embedding
3. Feature Extraction	4. Feature Selection
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

GUALBERTO, É. S. (2020). Phishing Detection: Methods Based on Natural Language Processing. Tese de doutorado em Engenharia Elétrica, Publicação PPGEE.TD-170/20, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 130p.

Éder Souza Gualberto

Éder Souza Gualberto

Departamento de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

A Deus, pelo dom da vida
e por todas as maravilhosas
oportunidades que me foram dadas.

A meus pais, Paulo e Jô,
por todo amor, incentivo e
esforço investido na minha educação.

A meu irmão Ádison, pelo
amor fraterno, apoio e amizade.

À minha esposa Fernanda, por todo amor, carinho,
companherismo, e pela família que formamos, que ficou
ainda melhor com a chegada da nossa tão esperada Melissa. Amo-te.

À minha filha Melissa, por transbordar o meu
coração com tanto amor, felicidade e alegria. Saiba filha,
a sua energia contagia a todos nós. É lindo te ver crescer.

A meus amigos e colegas, em especial ao Jurami,
Emílio, Fernando, Thiago, Leandro, André Gustavo,
Adriano, Cléoben, Gabriel, Fuji, Cesar e Diego, que também
acompanharam esta caminhada e fizeram dela algo mais prazeroso.

Agradecimentos

Agradeço a todos do PPGEE, que colaboraram direta e indiretamente para a realização desse trabalho.

Em especial:

Agradeço ao professor Rafael Timóteo por ter aceitado ser meu orientador nesse grande desafio, por suas relevantes orientações e ensinamentos ao longo desse período.

Agradeço ao professor João Paulo Lustosa por suas importantes orientações e correções nos artigos que publicamos.

Agradeço ao professor Daniel Guerreiro por seus ensinamentos nas disciplinas de processos estocásticos e de aprendizado de máquina.

Agradeço ao Thiago Vieira, por suas sugestões técnicas, dicas e apoio durante o doutorado de ambos.

Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN e 465741/2014-2 INCT em Cibersegurança), pela CAPES - Coordenação de Aperfeiçoamento do Pessoal de Nível Superior (Outorgas PROAP PPGEE/UnB, 23038.007604/2014-69 FORTE, e 88887.144009/2017-00 PROBRAL), pela FAP-DF - Fundação de Amparo à Pesquisa do Distrito Federal (Outorgas 0193.001366/2016 UIoT e 0193.001365/2016 SSDDC), pelo Ministério da Economia (Outorgas 005/2016 DIPLA e 083/2016 ENAP), pela Secretaria de Segurança Institucional da Presidência da República do Brasil (Outorga ABIN 002/2017), pelo Conselho Administrativo de Defesa Econômica (Outorga CADE 08700.000047/2019-14), pela Advocacia Geral da União (Outorga AGU 697.935/2019) e pelo DPI/DPG/UnB - Decanatos de Pesquisa e Inovação e de Pós-Graduação da Universidade de Brasília.

The important work of moving the world forward does not wait to be done by perfect men.

—GEORGE ELIOT

Resumo

Nas tentativas de phishing, o criminoso finge ser uma pessoa ou entidade confiável e, por meio dessa falsa representação, tenta obter informações confidenciais de um alvo. Um exemplo típico é aquele em que golpistas tentam passar por uma instituição conhecida, alegando a necessidade de atualização de um cadastro ou de uma ação imediata do lado do cliente e, para isso, são solicitados dados pessoais e financeiros. Uma variedade de recursos, como páginas da web falsas, instalação de código malicioso ou preenchimento de formulários, são empregados junto com o próprio e-mail para executar esse tipo de ação. Geralmente uma campanha de phishing começa com um e-mail. Portanto, a detecção desse tipo de e-mail é crítica. Uma vez que o phishing pretende parecer uma mensagem legítima, as técnicas de detecção baseadas apenas em regras de filtragem, como regras de listas e heurística, têm eficácia limitada, além de potencialmente poderem ser forjadas.

Desta forma, através de processamento de texto, atributos podem ser extraídos do corpo e do cabeçalho de e-mails, por meio de técnicas que expliquem as relações de semelhança e significância entre as palavras presentes em um determinado e-mail, bem como em todo o conjunto de amostras de mensagens. A abordagem mais comum para este tipo de engenharia de recursos é baseada em Modelos de Espaço Vetorial (VSM), mas como o VSM derivada da Matriz de Documentos por Termos (DTM) tem tantas dimensões quanto o número de termos utilizado em um corpus, e dado o fato de que nem todos os termos estão presentes em cada um dos emails, a etapa de engenharia de recursos do processo de detecção de e-mails de phishing tem que lidar e resolver questões relacionadas à "Maldição da Dimensionalidade", à esparsidade e às informações que podem ser obtidas do contexto textual.

Esta tese propõe uma abordagem que consiste em quatro métodos para detectar phishing. Eles usam técnicas combinadas para obter recursos mais representativos dos textos de e-mails que são utilizados como atributos de entrada para os algoritmos de classificação para detectar e-mails de phishing corretamente. Eles são baseadas em processamento de linguagem natural (NLP) e aprendizado de máquina (ML), com estratégias de engenharia de features que aumentam a precisão, recall e acurácia das previsões dos algoritmos adotados, e abordam os problemas relacionados à representação VSM/DTM.

O método 1 usa todos os recursos obtidos da DTM nos algoritmos de classificação, enquanto os outros métodos usam diferentes estratégias de redução de dimensionalidade para lidar com as questões apontadas. O método 2 usa a seleção de recursos por meio das medidas de qui-quadrado e informação mútua para tratar esses problemas. O Método 3 implementa a extração de recursos por meio das técnicas de Análise de Componentes Principais (PCA), Análise Semântica Latente (LSA) e Alocação Latente de Dirichlet (LDA). Enquanto o Método 4 é baseado na incorporação de palavras, e suas representações são obtidas a partir das técnicas Word2Vec, Fasttext e Doc2Vec. Foram empregados três conjuntos de dados (Dataset 1 - o conjunto de dados principal, Dataset 2 e Dataset 3).

Usando o Dataset 1, em seus respectivos melhores resultados, uma pontuação F1 de 99,74% foi alcançada pelo Método 1, enquanto os outros três métodos alcançaram uma medida notável de 100% em todas as medidas de utilidade utilizadas, ou seja até onde sabemos, o mais alto resultado em pesquisas de detecção de phishing para um conjunto de dados credenciado com base apenas no corpo dos e-mails.

Os métodos/perspectivas que obtiveram 100% no Dataset 1 (perspectiva Qui-quadrado do Método 2 - usando cem features, perspectiva LSA do Método 3 - usando vinte e cinco features, perspectiva Word2Vec e perspectiva FastText do Método 4) foram avaliados em dois contextos diferentes. Considerando tanto o corpo do e-mail quanto o cabeçalho, utilizando o primeiro dataset adicional proposto (Dataset 2), onde, em sua melhor nota, foi obtido 99,854% F1 Score na perspectiva Word2Vec, superando o melhor resultado atual para este dataset. Utilizando apenas os corpos de e-mail, como feito para o Dataset 1, a avaliação com o Dataset 3 também se mostrou com os melhores resultados para este dataset. Todas as quatro perspectivas superam os resultados do estado da arte, com uma pontuação F1 de 98,43%, através da perspectiva FastText, sendo sua melhor nota. Portanto, para os dois conjuntos de dados adicionais, esses resultados são os mais elevados na pesquisa de detecção de phishing para esses datasets.

Os resultados demonstrados não são apenas devido ao excelente desempenho dos algoritmos de classificação, mas também devido à combinação de técnicas proposta, composta de processos de engenharia de features, de técnicas de aprendizagem aprimoradas para reamostragem e validação cruzada, e da estimativa de configuração de hiperparâmetros. Assim, os métodos propostos, suas perspectivas e toda a sua estratégia demonstraram um desempenho relevante na detecção de phishing. Eles também se mostraram uma contribuição substancial para outras pesquisas de NLP que precisam lidar com os problemas da representação VSM/DTM, pois geram uma representação densa e de baixa dimensão para os textos avaliados.

Palavras-chave: Detecção de Phishing, Processamento de Linguagem Natural, Word Embedding, Feature Engineering, Feature Extraction, Feature Selection

Abstract

In phishing attempts, the attacker pretends to be a trusted person or entity and, through this false impersonation, tries to obtain sensitive information from a target. A typical example is one in which a scammer tries to pass off as a known institution, claiming the need to update a register or take immediate action from the client-side, and for this, personal and financial data are requested. A variety of features, such as fake web pages, the installation of malicious code, or form filling are employed along with the e-mail itself to perform this type of action. A phishing campaign usually starts with an e-mail. Therefore, the detection of this type of e-mail is critical. Since phishing aims to appear being a legitimate message, detection techniques based only on filtering rules, such as blacklisting and heuristics, have limited effectiveness, in addition to being potentially forged.

Therefore, with the use of data-driven techniques, mainly those focused on text processing, features can be extracted from the e-mail body and header that explain the similarity and significance of the words in a specific e-mail, as well as for the entire set of message samples. The most common approach for this type of feature engineering is based on Vector Space Models (VSM). However, since VSMs derived from the Document-Term Matrix (DTM) have as many dimensions as the number of terms in used in a corpus, in addition to the fact that not all terms are present in each of the e-mails, the feature engineering step of the phishing e-mail detection process has to deal with and address issues related to the "Curse of Dimensionality"; the sparsity and the information that can be obtained from the context (how to improve it, and reveal its latent features).

This thesis proposes an approach to detect phishing that consists of four methods. They use combined techniques to obtain more representative features from the e-mail texts that feed ML classification algorithms to correctly detect phishing e-mails. They are based on natural language processing (NLP) and machine learning (ML), with feature engineering strategies that increase the precision, recall, and accuracy of the predictions of the adopted algorithms and that address the VSM/DTM problems.

Method 1 uses all the features obtained from the DTM in the classification algorithms, while the other methods use different dimensionality reduction strategies to deal with the posed issues. Method 2 uses feature selection through the Chi-Square and Mutual Information measures to address these problems. Method 3 implements feature extraction through the Principal Components Analysis (PCA), Latent Semantic Analysis (LSA), and Latent Dirichlet Allocation (LDA) techniques. Method 4 is based on word embedding, and its representations are obtained from the Word2Vec, Fasttext, and Doc2Vec techniques.

Our approach was employed on three datasets (Dataset 1 - the main dataset, Dataset 2, and Dataset 3).

All four proposed methods had excellent marks. Using the main proposed dataset (Dataset 1), on the respective best results of the four methods, a F1 Score of 99.74% was achieved by Method 1, whereas the other three methods attained a remarkable measure of 100% in all main utility measures which is, to the best of our knowledge, the highest result obtained in phishing detection research for an accredited dataset based only on the body of the e-mails.

The methods/perspectives that obtained 100% in Dataset 1 (perspective Chi-Square of Method 2 - using one-hundred features, perspective LSA of Method 3 - using twenty-five features, perspectiveWord2Vec and perspective FastText of Method 4) were evaluated in two different contexts. Considering both the e-mail bodies and headers, using the first additional proposed dataset (Dataset 2), a 99.854% F1 Score was obtained using the perspective Word2Vec, which was its best mark, surpassing the current best result. Using just the e-mail bodies, as done for Dataset 1, the evaluation employing Dataset 3 also proved to reach the best marks for this data collection. All four perspectives outperformed the state-of-the-art results, with an F1 Score of 98.43%, through the FastText perspective, being its best mark. Therefore, for both additional datasets, these results, to the best of our knowledge, are the highest in phishing detection research for these accredited datasets.

The results obtained by these measurements are not only due to the excellent performance of the classification algorithms, but also to the combined techniques of feature engineering proposed process such as text processing procedures (for instance, the lemmatization step), improved learning techniques for re-sampling and cross-validation, and hyper-parameter configuration estimation. Thus, the proposed methods, their perspectives, and the complete plan of action demonstrated relevant performance when distinguishing between ham and phishing e-mails. The methods also proved to substantially contribute to this area of research and other natural language processing research that need to address or avoid problems related to VSM/DTM representation, since the methods generate a dense and low-dimension representation of the evaluated texts.

Keywords: Phishing Detection, Natural Language Processing, Word Embedding, Feature Engineering, Feature Extraction, Feature Selection

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BoW	Bag-of-Words
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
COVID-19	CoronaVirus Disease of 2019
CPTV	Cumulative Percentage of Total Variation
CRF	Conditional Random Field
DPI	Deep Packet Inspection
DTM	Document-Term Matrix
FPR	False Positive Rate
KNN	K-Nearest Neighbors
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
MWT	Multi-Word Token
NLP	Natural Language Processing
NMF	Non-Negative Matrix Factorization
PCA	Principal Component Analysis
PLSA	Probabilistic Latent Semantic Analysis

PMI Pointwise Mutual Information

POS Part-of-Speech

PV-DBOW Distributed Bag-of-Words version of Paragraph Vector

PV-DM Distributed Memory version of Paragraph Vector

RCNN Recurrent Convolutional Neural Network

RNN Recurrent Neural Network

SMOTE Synthetic Minority Oversampling Technique

SSL Secure Sockets Layer

SVD Singular Value Decomposition

SVM Support Vector Machines

TF-IDF Term Frequency - Inverse Document Frequency

TDM Term-Document Matrix

TPR True Negative Rate

TPR True Positive Rate

UBE Unsolicited Bulk E-mails

ULMFiT Universal Language Model Fine-tuning

URL Uniform Resource Locators

VSM Vector Space Model

XGBoost Extreme Gradient Boosting

List of Symbols

α	Priori Dirichlet probability distribution parameter, related to term-document distribution
$\mathbf{B_{d,m}}$	Eigenvectors matrix of $\mathbf{D}_{t,t}$\$
β	Priori Dirichlet probability distribution parameter, related to topic-term distribution
$\mathbf{C_{t,m}}$	Eigenvectors matrix of $\mathbf{T}_{d,d}$\$
$\mathbf{D_{t,t}}$	Matrix that expresses relation between the texts of the e-mails, obtained from the operation $\mathbf{M}^T\mathbf{M}$
$\mathbf{df_i}$	Number of documents containing the term i
$\mathbf{e}$	Vector of e-mails texts
$\mathbf{E_k}$	Number of expected observations of class k if there is no relationship between the feature variable and the target variable
$\mathbf{F_{d,t}}$	DTM which expresses the BoW model and relates documents to the terms in Method 1
$\mathbf{f_n}$	False Negative
$\mathbf{f_p}$	False Positive
$\mathbf{G_{d,t}}$	DTM weighted through the TF-IDF measure
$\mathbf{H}$	Matrix with selected features obtained through feature selection in Method 2

$\mathbf{I(X,Y)}$	Mutual Information measure
$\mathbf{J}$	Matrix with new extracted features obtained through feature extraction in Method 3
$\mathbf{K}$	Matrix with generated features obtained through word embedding in Method 4
$\mathbf{l}$	Vector of e-mails labels
$\mathbf{M_{d,t}}$	DTM which expresses the BoW model and relates documents to the terms
$\mathbf{N}$	Total number of documents
$\mathbf{O_k}$	Number of observations of class k
ϕ	Topic distribution over all the terms of vocabulary
$\mathbf{p(w_d)}$	Likelihood of w, given α and β, to a corpus of e-mails D
$\mathbf{p(w_i)}$	Probability of w_i occurs individually
$\mathbf{P(w_i, w_j)}$	Probability of the words w_i and w_j occur in the same word window
$\mathbf{p(w_j)}$	Probability of w_j occurs individually
$\mathbf{p_X(x)}$	Probability density of x
$\mathbf{p_{(X,Y)}(x,y)}$	Probability joint density, with X being the feature variable and Y being the target variable
$\mathbf{p_Y(y)}$	Probability density of y
$p(z,w,\phi,\theta\|\alpha,\beta)$	Probability distribution of hidden variables z, w, ϕ and θ, given the observed variables α and β.

$\mathbf{\Sigma_{m,m}}$	Diagonal matrix of the singular values
$\mathbf{T_{d,d}}$	Matrix that expresses relation between the terms of the e-mails texts, obtained from the operation $\mathbf{MM}^{\mathrm{T}}$
$\mathbf{tf_{i,j}}$	Number of occurrences of the term i in document j
$\mathbf{t_n}$	True Negative
$\mathbf{t_p}$	True Positive
θ	Topic distribution over the documents
$\mathbf{w}$	All the terms of the vocabulary
$\tilde{\chi}^2$	Chi-Square test score
$\mathbf{z}$	topic distribution associated to the terms in the documents

1
Introduction

Any man who afflicts the human race with ideas must be prepared to see them misunderstood.
—HENRY LOUIS MENCKEN

The internet plays a crucial role in industries and societies worldwide by providing a wide variety of services. According to [94, 18], 62% of the world´s population are internet users, and, in 2023, this percentage will increase to 66%.

According to [38], in 2019, the total number of e-mails transacted every day exceeded half-trillion, and about 80% of this e-mail traffic were spam messages. Although some of these spam messages are legitimate marketing e-mails, there are also malicious e-mails through which sensitive information can be exposed or subtracted. A successful malicious e-mail can lead to critical incidents such as financial frauds and hacked or hijacked systems, accounts, or profiles. These malicious messages are known as phishing e-mails.

In this type of fraud attempt, the attacker pretends to be a trusted person or entity, and through this false impersonation, tries to obtain sensitive information from a target [55, 3]. A typical example is one in which a scammer tries to pass off as a known institution, claiming the need to update a register or for immediate action from the client-side and, for this, personal and financial data are requested. A variety of features, such as fake web pages, malicious code installation, or form filling are employed along with the e-mail itself to perform this type of action [16].

Phishing, as well as other cybercrimes, is constantly evolving and becoming more crafty and refined. According to [6], in the first quarter of 2020, 75% of all phishing sites used secure sockets layer (SSL). Also according to [6], many Coronavirus Disease of

2019 (COVID-19)- themed phishing attacks have been launched as of mid-March, 2019. In order to convince their targets, phishing approaches employ all available information, ranging from internet and security technologies to the new coronavirus pandemic.

Much research has aimed to develop applications that can correctly detect phishing cases [16, 92, 3, 4]. Some of this research focuses on phishing site detection [93, 9, 109, 113], whereas other lines focus on phishing e-mail detection [1, 76, 45]. This thesis concentrated on phishing e-mail detection. Most of these studies utilize natural language processing (NLP) techniques combined with machine learning techniques to perform such detection activities based on classification tasks. These are effective mechanisms of defense against this type of threat, since such approaches exploit the morphology and semantics of the text. Information such as the text body and header of e-mail messages, URLs, and tags are processed and used as input data to be employed by the classification algorithms [110].

Traditionally, a phishing campaign starts with an e-mail [16, 92, 3, 4], and therefore the detection of this type of e-mail is critical. Since phishing aims to appear as a legitimate message, detection techniques based only on filtering rules, such as blacklisting and heuristic, have limited effectiveness [4], in addition to being potentially forged.

Thus, with data-driven techniques [74], features can be extracted from the e-mail body and header texts using techniques that explain relations of similarity and meaning between the words present in a specific e-mail, as well as for entire sets of message samples.

The most common approach to this type of feature engineering is based on Vector Space Models (VSMs) [37]. In this type of representation, each message uses numerical values to represent each of its terms (words, for instance) as symbols in a vector space, i.e., each term in each text in the entire corpus[1] is a dimension in which each e-mail is denoted by its term ranking.

As explained in [100], since the VSM derived from the Document-Term Matrix (DTM) has as many dimensions as the number of terms in a used corpus, and the fact that not all terms are present in each of the e-mails, the feature engineering step of the phishing e-mail detection process has to deal with and address questions related to the "Curse of Dimensionality" and sparsity [90], [104], [59]. Additional crucial aspects regard the information that can be obtained from the context to embed in the VSM, how to improve it, and explicit its latent features [28] [100].

The first problem, the "Curse of Dimensionality", refers to the high number of

[1]Corpus is a computer-readable collection of text or speech [49].

dimensions in which a text is represented by its word ranking. The second refers to the fact that as data dimensionality increases, data sparsity also increases. These two problems require the processing and storage of large corpus, besides potentially causing overfitting due to some features being rarely observed. Finally, the third problem regards the few context properties of the text incorporated in this VSM representation type.

In order to overcome the three main problems mentioned above, statistical measures, feature extraction, and distributed models based on word embedding are the three most commonly-used techniques in the literature.

Statistical measures can be used to select fewer features, a subset of original features, that are supposed to be more representative than the other features [102].

It is also possible to use feature extraction techniques [44] [57] [82], which, starting from the initial high-dimensional matrix, obtain more discriminative features from the original features extracted from the text.

Instead of selecting more representative features, or performing mathematical transformations or probabilistic calculations on the VSM representation to extract more distinctive features, representing such texts in a fixed shared low-dimensional space, word embedding [45] [101] [8, 108], is also an approach. In this paradigm, a vector and its pre-fixed dimensions represent a word and its contextual information (such as relations with other words and semantic and syntactic similarities).

This thesis concentrates on phishing e-mail detection, mainly on propositions based on the processing of text of the e-mail bodies or/and headers to obtain the necessary feature attributes for the ML algorithms and improved strategies to provide better results training these classification algorithms.

We present an approach that consists of four methods to detect phishing. The first that does not address these VSM issues, and uses all the features obtained from the words used in the e-mail texts. The second that uses feature selection statistic measures to handle this landscape. The third that extracts new features when projecting the initial attributes derived from the term occurrences. And the fourth that is not based on the rank of the terms, but generates a fixed low-dimensional vector representation for each e-mail.

The remainder of this chapter is organized as follows. The problem statement is stated in Section 1.1, and the proposals are outlined in Section 1.2. In Section 1.3, the main general and specific objectives of this thesis are presented. In Section 1.4, we introduce the adopted methodology and, lastly, the contributions of this thesis are presented in Section 1.5. The organization of this thesis for the subsequent chapters is described in

Section 1.6, and the notation employed throughout this work is described in Section 1.7.

1.1 Problem Statement

Although natural language processing and machine learning have been largely utilized in phishing e-mail detection, methods that, together with these techniques, are also based on semantic and similarity enrichment, and established training techniques for machine learning algorithms have not been appropriately employed in this context. Approaches that produce more distinctive features to improve phishing detection and prediction rates for this problem successively are suitable for this scenario.

The following research question then arises: Can a phishing detection approach that addresses the issues related to a VSM derived from a DTM improve the identification of this type of cybercrime?

1.2 Proposals

Our proposal provides and analyzes methods, through the use of combined techniques, to obtain more representative feature attributes from the e-mail texts to feed ML classification algorithms to correctly detect phishing e-mails.

This thesis presents an approach based on feature engineering and enhanced learning techniques for phishing detection. Through a set of text processing, representation methods, and training phases, the approach provides strategies to supply more distinctive/characteristic features for this classification problem from certain robust representation perspectives.

Differently from previous works, we aim to propose a solution to correctly detect phishing e-mails to prevent them from reaching the target user, through a comprehensive process for this landscape. It is based on natural language processing (NLP) and machine learning (ML), with feature engineering strategies that increase the precision, recall, and accuracy of the predictions of the adopted algorithms, and that address the problems posed in Section 1.1, i.e., the "Curse of Dimensionality", the sparsity, and the information that can be obtained from the context.

1.3 Objectives

The broad objective of this thesis is to present a holistic, structured architecture and approach for phishing detection, with methods, based on NLP and ML, that address the problems related to VSM representation derived from DTM so as to improve threat identification predictions.

Toward this end, the following specific objectives are defined:

- Review of the fundamental theoretical framework and concepts related to phishing detection, natural language processing and machine learning.
- Development of pre-processing procedures to provide suitable and correct data cleansing.
- Design structured strategies to handle issues of VSM representation derived from a DTM.
- Development of methods to provide effective and low-dimensional text representation, more specifically for e-mail texts.
- Gathering of optimized features able to typify structures, similarities, and other relevant information for classification problems, such as phishing detection.
- Performing of classification tasks to sort phishing against legitimate e-mails, using ML algorithms fed with these optimized features and trained using an established plan of action.

1.4 Methodology

The proposed approach consists of four methods, each with a different course of action to obtain the desired features.

The methods start from a common stage and then specialize within each method scheme. This common stage refers to the parsing and the pre-processing steps, where the first refers to the process of extraction from the datasets, and the second to the operations, such as lowercasing the text, and passing it through the tokenization, part-of-speech (POS) tagging, and lemmatization phases, using certain databases as a dictionary structure.

From this step onwards, the methods follow different flows. The first three methods are based on the DTM, which is extracted from these processed texts, and represents each e-mail as a ranking of its words in a VSM. Method 1 does not manipulate this structure to control these VSM issues; Method 2 does this by selecting a subset of the original feature set; and Method 3 extracts new features from those of the original feature set. The fourth method, Method 4, does not use the DTM. It associates a fixed vector to each remaining word in an e-mail, that represents it, from which the features are generated (word embedding). Lastly, each of the different sets of features obtained through these methods feeds the same eight ML classification algorithms, using improved learning techniques, concluding each method.

Figure 1.1 outlines the general architecture and methodology of the proposed approach.

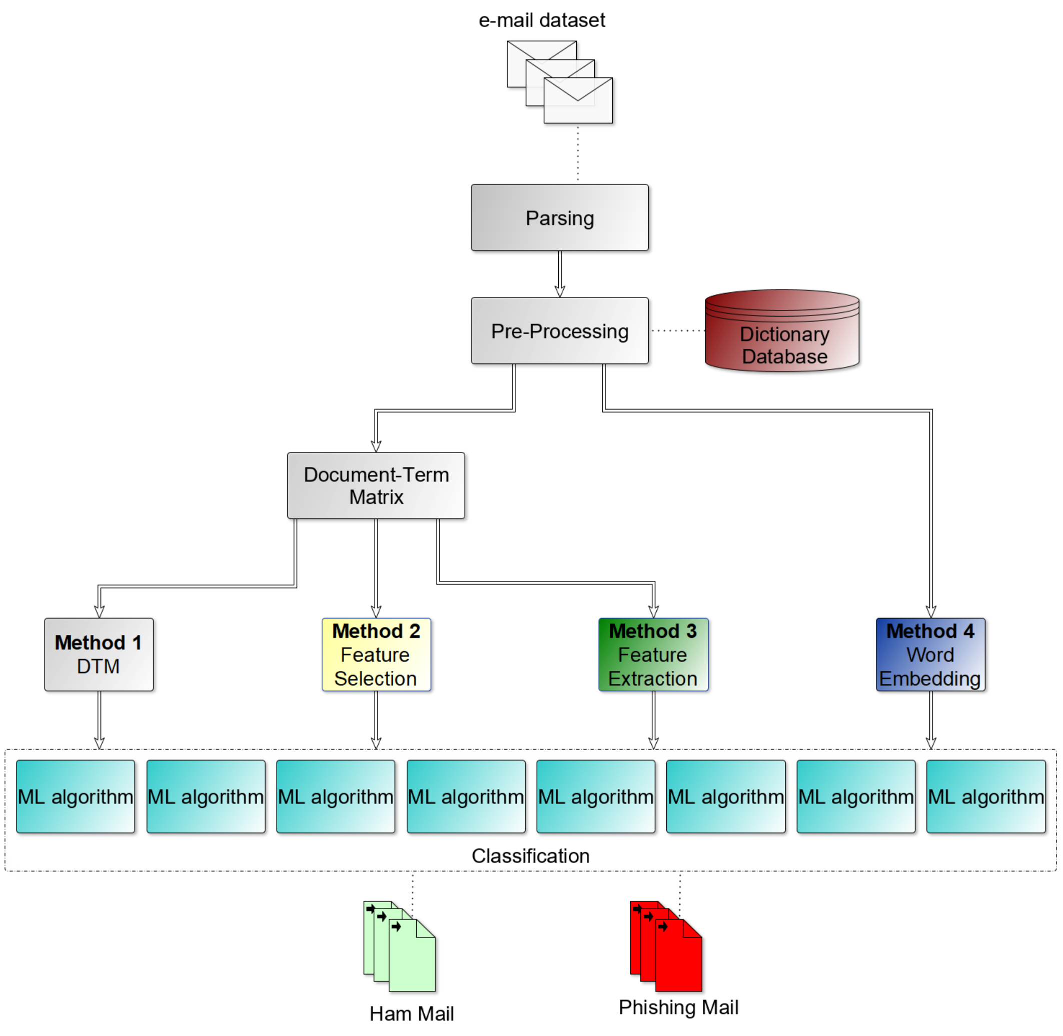


Figure 1.1 The general architecture and methodology of the proposed approach

1.5 Publications

This thesis research covers the phishing detection scenario and proposes methods to improve the identification of this type of threat through NLP and ML techniques. The proposed approach is not only an answer to the above-mentioned problems, but also demonstrate an optimal representation capacity, since it uses a number of features that is much smaller than the original amount of features, but with better performance measures. It shows that these features provide an enhanced distinction of messages from the selected datasets (phishing or legitimate e-mails).

These achieved results also produced the following publications:

- E. S. Gualberto, R. T. De Sousa, T. P. D. Vieira, J. Da Costa, and C. G. Duque, "From Feature Engineering and Topics Models to Enhanced Prediction Rates in Phishing Detection", IEEE Access, vol. 8, pp. 76368-76385, 2020. [40]
    - We proposed an approach based on machine learning to detect phishing e-mail attacks. The methods that compose this approach are performed through a feature engineering process based on natural language processing, lemmatization, topics modeling, improved learning techniques for resampling and cross-validation, and hyperparameters configuration. The first proposed method uses all the features obtained from the Document-Term Matrix (DTM) in the classification algorithms. The second one uses Latent Dirichlet Allocation (LDA) to deal with the already listed VSM problems. The proposed approach reached marks with an F1-measure of 99.95% success rate using the XGBoost algorithm fed with just ten features. It employed an accredited data set and is based only on the body of the e-mails, without using other e-mail features such as its header, IP information, or number of links in the text.
- E. S. Gualberto, R. T. De Sousa, T. P. D. Vieira, J. Da Costa, and C. G. Duque, "The Answer is in the Text: Multi-Stage Methods for Phishing Detection based on Feature Engineering", IEEE Access, vol. 8, pp. 223529-223547, 2020. [39]
    - In this paper, a multi-stage approach is proposed to detect phishing e-mail attacks based on natural language processing and machine learning. It is performed through a feature engineering process based on natural language processing, lemmatization, feature selection, feature extraction, improved learning techniques for resampling and cross-validation, and hyperparameters configuration. We present two methods of the proposed approach: the first one exploiting the Chi-Square statistics and the Mutual Information to improve the dimensionality reduction and the second one based on the Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA). Both

methods handle the problems of the "curse of dimensionality", the sparsity, and the information that must be obtained from the context in the Vector Space Model (VSM) representation. The proposed approach, combined with XGBoost and Random Forest algorithms, achieves marks with an F1-measure of 100% success rate employing reduced feature sets, using the SpamAssassin Public Corpus and the Nazario Phishing Corpus as datasets sources. Even based only on the text of the e-mail bodies, it outperforms state-of-the-art phishing detection research for an accredited dataset, using a much smaller number of features and a lower computational cost.

1.6 Thesis Organization

This thesis is organized as follows:

- In Chapter 2, we describe the main theories and concepts concerning natural language processing and machine learning techniques employed in this research problem. Also, the related works are discussed.
- In Chapter 3, a detailed explanation of the architecture, the methodology, and the procedures employed in each of the approach methods is presented.
- In Chapter 4, the performance measures used to evaluate the proposed work are described, and the obtained results are presented and analyzed.
- In Chapter 5, we conclude the thesis, summarize the main achievements and contributions, and suggest future research.

1.7 Notation

The notations used in this thesis are defined as follows: vectors are denoted by lowercase boldface letters (for instance: **a**, **b** and **c**), and matrices are described by uppercase boldface letters (such as **A**, **B** and **C**). The matrix elements are denoted by this shape: $\mathrm{a}_{i,j}$, i.e., the element of matrix **A** located in line i column j).

2 Theoretical Framework

The creator of the universe works in mysterious ways. But he uses a base ten counting system and likes round numbers.

—SCOTT ADAMS

As already mentioned, this approach proposed in this thesis for phishing detection is mainly focused on natural language processing (NLP) and machine learning (ML) techniques.

Natural Language Processing is an Artificial Intelligence subarea, and its object of study is the acquisition of knowledge by machine agents, using language models from human language (natural language), with the objective of performing activities related to the search for information (text classification, information retrieval, and information extraction) [87].

Machine Learning is also an Artificial Intelligence (AI) subarea concerned with the question of how to build computer programs that automatically improve with experience [71]. The purpose of these algorithms is understanding the example data or the underlying structure of past experiences, modeling the underlying phenomena/processes (the performance criterion is optimized with this experience), and providing an algorithm for this task. A model that is a useful and good enough approximation of the data is expected [5].

We aim to generate more expressive features from the existing terms/words in e-mails (documents) and to subject them to different machine learning algorithms, using enhanced techniques to obtain improved results in classification tasks.

To generate more distinctive attributes, the e-mail texts were submitted to pre-processing in order to: eliminate words/terms that do not add much to the semantics of the documents; strengthen and enrich relationships of synonymy and polysemy; and, in some perspectives of our approach, assign a higher weight to words/terms that better reveal the respective class of document.

Since the classifying of e-mails as phishing or as legitimate messages, as well as other NLP activities in general, must address the problem of sparse and high-dimensionality feature matrices, these pre-processing steps, though they embed more information into the features and make them more meaningful, do not address these problems directly or at least do not do so at a necessary level. It is thus needed to use dimensionality reduction techniques, which can select, extract, or generate more instructive and discriminative features from the pre-processed texts, allowing a low-rank representation of the data.

The features can then be submitted to classification algorithms. The choices of which machine learning algorithms to use, as well as of the dataset, were based on previous works related to phishing detection for comparison purposes, and also on the performance power of the algorithms, in order to obtain the best possible results. This way, it was possible to measure not only the effectiveness of the techniques used in this proposed approach, concerning the results of previous works, but also to introduce more robust algorithms in this research area in order to obtain state-of-the-art marks.

To ensure a good understanding of the techniques used in the research problem proposed in this thesis, this chapter presents the main related concepts and associated theories. The remainder of this chapter is organized as follows: The pre-processing related techniques and concepts are exposed in Section 2.1. The Bag-of-Words and Document-Term Matrix models are introduced in Section 2.2. The techniques employed to promote reduced amount of features are outlined in Section 2.3, and those related to feature selection, feature extraction and word embedding are presented, respectively, in Sections 2.4, 2.5 and 2.6. In Section 2.7, we introduce certain concepts related to supervised learning, and some details about the chosen ML classification algorithms. The related works and the state of the art are discussed in Section 2.8.

2.1 Pre-processing related techniques and concepts

This process starts with all the e-mails of the chosen datasets undergoing a parsing process, in which the text (body and/or header) of all the e-mails is extracted (from which all the necessary features are obtained), keeping their associated labels indicating in each whether it is a phishing e-mail or a legitimate (ham) e-mail.

The e-mail bodies are submitted to pre-processing in order to: eliminate words/terms that do not add much to the semantics of the documents; strengthen and enrich semantics and relationships of synonymy and polysemy; and assign a higher weight to words/terms that better reveal the respective class of document. Some important concepts/discussions related to this step are explored in this Section, such as stopwords, tokenization, part-of-speech tagging, lemmatization, WordNet database, and Stanza database.

2.1.1 Stopwords

Stopwords refer to a class of words that usually have little lexical content or that do not contribute much to the meaning of a sentence. Although there is no universal list representing all stopwords, most cases take prepositions and articles as such.

2.1.2 Tokenization

The tokenization process is performed on the remaining texts, on a term level, i.e., using white spaces (space, tab, and newline) as delimiters, with each text divided into terms (tokens). The tokenization step is critical, not only to create a vocabulary for the corpus under analysis [62] [49], but also to perform other required NLP actions for the proposed approach, such as removing stopwords.

2.1.3 Part-of-Speech (POS) tagging

The Part-of-Speech (POS) tagging process annotates the grammatical class of each token (such as ADJ, for an adjective, ADV for an adverb, NOUN for a noun, PRON for a pronoun, or VERB for a verb [24]).

2.1.4 Lemmatization

To promote the correct understanding of what occurs in a lemmatization process, some concepts need to be explained. They refer to morpheme, stem, affixes, lemma, and lexeme.

Morphemes refer to the smaller meaning-bearing units that comprise a word; **stem** refers to the morpheme that concentrates the word's primary meaning; affixes refer to the morphemes that offer additional meanings of various types to a word [49]; **lemma** is a word or expression, a particular form, that is chosen to represent a lexeme, which in turn is the basic meaning of a stem [62].

The aim of lemmatization is to transform a word into its common base form. To reduce the inflectional forms and the derivationally related forms of a word, lemmatization typically involves the use of a vocabulary and a morphological parsing. A word is analyzed on a morpheme level in order to separate its root morpheme (stem) from its accessory morphemes (affixes), returning it to a lemma shape, i.e., obtaining the stem in dictionary form.

2.1.5 WordNet

WordNet refers to a large lexical database of the English Language [70]. With its synsets (sets of cognitive synonyms), it tries to express the meaning of a concept. These sets include words from the noun, verb, adjective, and adverb grammatical classes. The interlink among the synsets provides a network of meaningfully related words and concepts that can be used to obtain better results in natural language processing.

2.1.6 Stanza

Stanza is an NLP toolkit that supports 66 human languages [80]. From a raw text as input, it delivers useful annotations such as tokenization, Multi-Word Token (MWT) expansion (expanding a raw token into multiple syntactic words), POS and morphological feature tagging (the latter works as an extension of POS tagging, through which it is possible to annotate words with features that distinguish their additional lexical and grammatical properties, not covered by the POS tags [23]), and lemmatization. It also produces annotations related to dependency parsing and named entity recognition.

2.2 The Bag-of-words model and the Document-Term Matrix (DTM)

Bag-of-words (BoW) refers to a model in which a text is represented as a list of words or other n-gram (a continuous sequence of n items of a sample, such as characters, syllables, or words) and their respective multiplicity (how many times each of these occurs in the text under the feature extraction process). It embeds no contextual information, such as grammatical class and order of occurrence of those words [26].

In our approach, from this model, we obtained the Document-Term Matrix that represents each text or document in a row, and each term in a column. Its elements are the ranking of each term and document. This ranking is usually represented by its occurrence count, or by the TF-IDF calculus over the DTM [62]. It is also a type of feature extraction.

2.2.1 Feature weighting through Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that assigns weights to the importance of a term for a document (or for a text sample, such as an e-mail body), which is inserted in a corpus [84]. Expressed by eq. 2.1:

$$w_{i,j} = tf_{i,j} \cdot \log\left(\frac{N}{df_i}\right) \tag{2.1}$$

where: $\mathrm{tf_{i,j}}$ is the number of occurrences of the term i in document j, N is the total number of documents, and $\mathrm{df_i}$ is the number of documents containing i.

Thus, when the term i occurs many times in a small number of $\mathrm{df_i}$ documents, given the equation (2.1), a high weight is assigned to the term i in the $\mathrm{df_i}$ documents in which it occurs. Likewise, a lower weight is assigned to the term i, if this term occurs few times in a document or if it occurs in many documents (an even smaller weight is assigned to a term i if it occurs in all N documents) [62].

2.3 Dimensionality Reduction Methods

In the next sections, the techniques used for dimensionality reduction will be described in detail, in order to elucidate their respective procedures and to allow a clear understanding of the obtained results.

2.4 Feature Selection

The main objective of the feature selection method is to select a subset of original features [21]. For this purpose, the features are ordered by a utility measure, whereby those with the highest values will be selected according to the desired number of features. Unselected features, those with values less than the threshold value, are removed and no longer used in the following activities (classification tasks in this thesis case).

Chi-Square and Mutual Information measures are used. They are both univariate feature selection methods. The first one, Chi-Square, aims to measure the linear dependency between two random variables (an input feature and the target), whereas the second one, Mutual Information, also captures non-linear relationships between the input feature under analysis and the target.

2.4.1 Chi-Square

Chi-square is a statistical test used to calculate the relationship degree between the feature variables and the target variable in a dataset [27]. In this proposed approach, it is used to calculate how dependent the e-mail class (phishing mail or ham mail) is on each of the incoming features.

This test score is given by the equation 2.2:

$$\tilde{\chi}^2 = \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k} \tag{2.2}$$

where O_k is the number of observations of class k (observed frequency) and E_k the number of expected observations of class k if there is no relationship between the feature variable and the target variable (expected frequency). If the chi-square test is 0 (the null hypothesis), there is no association between both variables. They are independent. On the other hand, the higher the chi-square value, the greater the relationship between the two variables (the alternative hypothesis).

2.4.2 Mutual Information

Mutual information is a measure for quantifying the mutual dependence between two variables, based on the entropy (from the information theory) of a random variable. It calculates what amount of information is reached in a random variable from another random variable. That is, in the context of this work proposal, to identify how much information each feature provides to determine if an e-mail is a phishing or a legitimate mail [10].

The mutual information of two jointly continuous random variables is given by the double integral expressed in the equation 2.3:

$$I(X,Y) = \int_Y \int, p_{(X,Y)}(x,y) \log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) dxdy \tag{2.3}$$

where $p_X(x)$ is the probability density of x, $p_Y(y)$ is the probability density of y, and $p_{(X,Y)}(x,y)$ is the joint probability density, with X being the feature variable and Y being the target variable, or vice versa [41].

2.5 Feature Extraction

The feature extraction methods obtain new features (of lower dimensionality) from the original features set. Some transformations do it over the original feature space, that is, the new reduced feature space dimensions are combinations of the original high dimensional data. These new features are intended to be more representative, concentrating relevant information from the underlying data in a non-redundant shape, with less noise [36].

PCA, LSA, and LDA techniques are used. For the first two, the input data is the DTM based on the TF-IDF measure. For LDA, the initial DTM is used.

2.5.1 Principal Component Analysis (PCA)

Principal Component Analysis is a technique that, according to [5], focuses on finding a mapping from the inputs in the original dimensional space to a new smaller dimensional space, always seeking the minimum loss of information.

The principal components can be understood as the underlying structure in the data. They are found by searching for eigenvectors and eigenvalues that maximize the variance of projected data and make them more spread out in the new dimensional space. This technique's basic idea consists of converting variables, potentially correlated, into linearly uncorrelated variables, the principal components, by an orthogonal transformation, as [10] exposes. A pair of eigenvectors and eigenvalues represent each of these. The eigenvector represents a principal component's direction, and the eigenvalue represents how much variance this direction contains. Then, the first principal component contains more variance from the original data than the second; the second principal component contains more variance than the third; and so on. All the principal components are orthogonal to each other.

PCA is based on covariance matrix [19] [107]. Although the code implementation of this technique uses different calculations in order to be more computationally efficient, the PCA steps consist basically of: First, the sample is standardized (centered). To do this, for each feature, the feature column mean is subtracted from each of the values of that particular feature, and is divided by the standard deviation. Next, the covariance matrix is calculated. This is done to detect any relationship between variables, which may mean that these variables contain redundant information. Then, from the covariance matrix, the eigenvectors and the respective eigenvalues, the principal components, are obtained by an eigendecomposition. Thereon, it is decided how many components to work with and finally, the original data matrix is projected onto the axes of the principal component [91]. More details can be found in [48].

PCA can also be calculated through the use of SVD (explained in Subsection 2.5.2).

One of the most commonly used criteria for selecting a quantity of the obtained principal components is to select a Cumulative Percentage of Total Variation (CPTV) [77] [85] [48], which is a percentage of variance, given by the sum of the variances of the first n principal components. It is typically indicated for working with an amount of approximately eighty to ninety percent (depending of the practical details of the dataset under analysis) of the initial variance [48].

2.5.2 Latent Semantic Analysis (LSA)

LSA refers to a mathematical technique in natural language processing, whose purpose is to make explicit (latent) topics/concepts embedded in the input data (the documents), from the analysis of the relationships between these documents and the terms contained therein. Documents and terms are expressed as vectors of elements that correspond to these concepts. Thus, the elements in these vectors indicate the degree of participation of a document or term in the represented topic/concept [54] [22].

This technique is based on a factorization through Singular Value Decomposition (SVD) [62] [53]. The SVD technique and its settings for this study are explained as follows:

Suppose an array $\mathrm{M}_{\mathrm{d,t}}$, which expresses the BoW model and relates documents to the terms found in them (the DTM), where its lines are the documents d, and its columns are the terms t. The SVD stipulates that this matrix M can be factored into the form [63]:

$$M_{d,t} = B_{d,m}\Sigma_{m,m}C^{T}_{t,m} \tag{2.4}$$

where: $\mathbf{B}_{\mathrm{d,m}}$ is the eigenvectors matrix of $\mathbf{D}_{\mathrm{t,t}} = \mathbf{M}^{\mathrm{T}}\mathbf{M}$, $\mathbf{C}_{\mathrm{t,m}}$ is the eigenvectors matrix of $\mathbf{T}_{\mathrm{d,d}} = \mathbf{M}\mathbf{M}^{\mathrm{T}}$. $\Sigma_{\mathrm{m,m}}$ is the diagonal matrix of the singular values σ_{i} of $\mathbf{M}$ (for i = 1, ..., min (d x t)), which are the square roots of the nonzero eigenvalues of **B** and **C**.

In the particular case of the problem discussed in this study, matrix **D** is a matrix that expresses relation between the texts of the e-mail bodies and/or headers (our documents). So, if e-mail j and k have x terms in common, then $\mathrm{d}_{\mathrm{j,k}} = \mathrm{x}$, while in matrix **T**, which expresses relation between the terms, if the terms l and m occur together in y e-mail bodies, then $\mathrm{c}^{\mathrm{T}}_{\mathrm{l,m}} = \mathrm{y}$ [62]. Similarly, as explained in [53], $\mathbf{B}_{\mathrm{d,m}}$ maps terms to topics ($\mathrm{b}_{\mathrm{i,j}}$ is the weight of term i in the topic j) and $\Sigma_{\mathrm{m,m}}\mathbf{C}_{47,107,\mathrm{m}} = \mathrm{S}$ maps topics to documents ($\mathrm{s}_{\mathrm{i,j}}$ is the weight of the topic i in the documents j). Thus, LSA would be used in a similar way as Latent Dirichlet Allocation (LDA), explained in Subsection 2.5.3, is used in [40].

To work with k singular values, this decomposition of **M** is truncated with k elements, as expressed in (2.5).

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \cdots & m_{1,t} \\ m_{2,1} & m_{2,2} & m_{2,3} & \cdots & m_{2,t} \\ m_{3,1} & m_{3,2} & m_{3,3} & \cdots & m_{3,t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{d,1} & m_{d,2} & m_{d,3} & \cdots & m_{d,t} \end{bmatrix} = \begin{bmatrix} b_{1,1} & \cdots & b_{1,k} & \cdots & b_{1,n} \\ b_{2,1} & \cdots & b_{2,k} & \cdots & b_{2,n} \\ b_{3,1} & \cdots & b_{3,k} & \cdots & b_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{d,1} & \cdots & b_{d,k} & \cdots & b_{d,n} \end{bmatrix} \cdot \begin{bmatrix} \sigma_{1,1} & \cdots & \sigma_{1,k} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & \cdots & \sigma_{2,k} & \cdots & \sigma_{2,n} \\ \sigma_{k,1} & \cdots & \sigma_{k,k} & \cdots & \sigma_{k,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{d,1} & \cdots & \sigma_{d,k} & \cdots & \sigma_{d,n} \end{bmatrix} \cdot \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & c_{k,3} & \cdots & c_{k,t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & c_{n,3} & \cdots & c_{n,t} \end{bmatrix} \quad (2.5)$$

It maintains only the first k columns of $\mathbf{B}_{d,m}$, the first k lines and the first k columns of $\Sigma_{m,m}$ and the first k lines of $\mathbf{C}^{T}{}_{t,m}$. That is, the coefficients of these matrices perform a projection onto a k-dimensional Space. This process is portrayed in Figure 2.1.

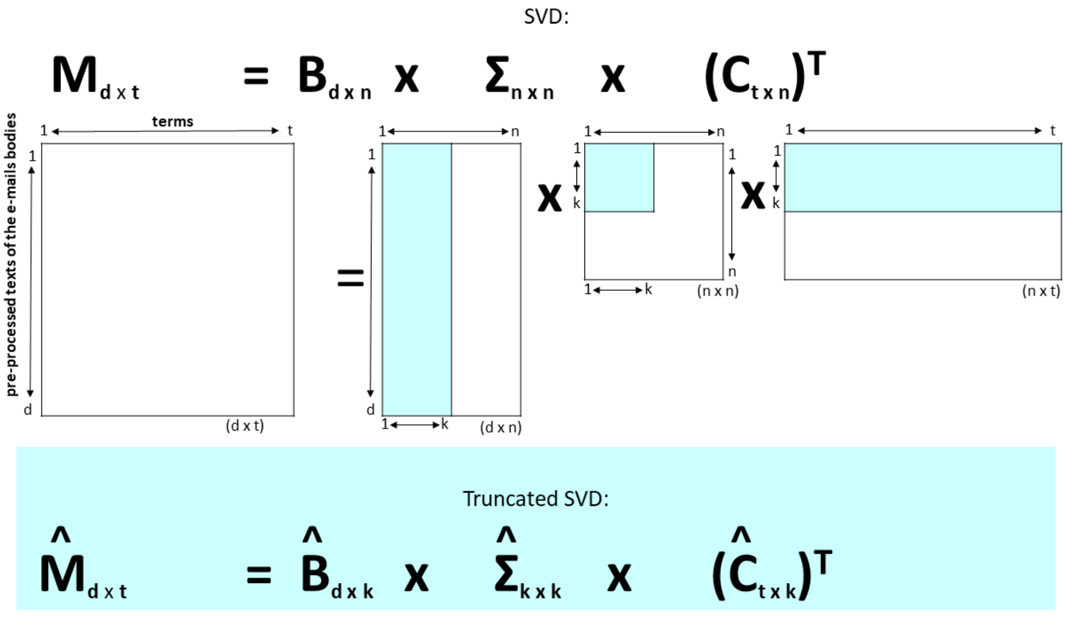

SVD:
M d x t = B d x n x Σ n x n x (C t x n)T
terms
pre-processed texts of the e-mails bodies
1
t
d
(d x t)
n
k
(d x n)
(n x n)
(n x t)
Truncated SVD:
M̂ d x t = B̂ d x k x Σ̂ k x k x (Ĉ t x k)T

Figure 2.1 The SVD truncage process in LSA.

For LSA, the definition of the number of dimensions to select was presented as a decision based on empiricism since the objective is to find an optimal dimensionality that produces similar or better results for the process using it (through the correct induction of underlying similarity relations) [52].

Some details about PCA and LSA are particularly noteworthy. The main difference between PCA, when using SVD, and LSA is the feature-wise normalization. PCA performs it over the DTM (after TF-IDF) before executing SVD, whereas LSA executes SVD directly, without this normalization. Thus while PCA tries to reproduce the highest amount of variance of the original data, LSA tries not to scale up the weight of rarely occurring terms. Both techniques aim to remove some of the noise of the data, provide improved similarity measures among the instances (documents), and reduce the dimensionality [48] [22].

Another relevant point refers to the DTM transformation proposed by the SVD. It considers the underlying process as a process defined by a normal distribution. While the word occurrence count in a text, as well as phishing in a set of incoming e-mails, may be better explained as a process governed by a Poisson distribution this, depending on the paradigm followed, would be an inconsistency [61]. The point here is that although the elements of DTM are derived from the term occurrence count in the texts, they are not used as such, but rather as the weights of its discriminative features in document similarities [100], and this weight can be ranked based on the term occurrence count, its frequency, TF-IDF or other measures. The other perspectives based on VSM (such as the Chi-Square used in Method 2) follow the same conception since their methods are designed for normally-distributed data.

2.5.3 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation is a generative probabilistic model, from the topic model class, in which the documents may be represented as random mixtures of topics, and each topic may be modeled as a distribution over words/terms present in the dataset (vocabulary) [12]. This means that the latent topics probabilities provide an explicit representation of the entire collection of documents (in our case, all the e-mails of the dataset). From the dataset, the probability distributions are estimated, and from them, the latent topics are inferred. Then, the extracted topics may be used as input features, since each e-mail can be represented as a vector that indicates the probability distribution of this document over the selected topics.

The basic idea is that to write a text, there are some pre-defined topics to use in the texts set. Their distributions obey the Dirichlet distribution. It is assumed that during the process of drafting the text, its generation process, the author exchanges these topics, using the words belonging to each. That is, the words from different topics are allocated by the result of the Dirichlet distribution sample result, and, through this process, the document is populated. It is important to note that documents may have the same topics but still be different because they contain different proportions of these topics.

According to [11], to draft a document, first a distribution is chosen, then, for each position of the document terms, a topic assignment is chosen and, finally, the word from the corresponding topic is chosen.

Thus, considering the generative process explained above, and making α the priori Dirichlet probability distribution parameter, related to term-document distribution, β the priori Dirichlet probability distribution parameter, related to topic-term distribution, z the topic distribution associated to the terms in the documents, w all the terms of the vocabulary, ϕ the topic distribution over all the terms of vocabulary, θ the topic distribution over the documents, the probability distribution of all the hidden and observed variables is given by equation 2.6.

$$p(z,w,\phi,\theta|\alpha,\beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{m=1}^{M} p(\theta_m|\alpha) \prod_{v=1}^{V} p(z_{m,v}|\theta_m) p(w_{m,v}|z_{m,v},\phi_{m,k}) \quad (2.6)$$

In equation 2.6, K is the number of topics, ϕ_k is a vector with the vocabulary terms proportion for the topic k, M is the number of documents, θ_m is a vector with the topic proportion for the document $\mathrm{d_m}$, V is the number of words in the vocabulary, $\mathrm{z_{m,v}}$ is the topic distribution associated to term $\mathrm{w_{m,v}}$ in the document $\mathrm{d_m}$, $\mathrm{w_{m,v}}$ is the term $\mathrm{w_v}$ in the document $\mathrm{d_m}$. While K varies from 1 to K, m varies from 1 to M, and V varies from 1 to V.

Given the words observed for the proposed vocabulary, w, and using Bayes' theorem, the hidden structure, that is, the assignments of topics for documents, the document distributions by topics, and the topics distributions by terms can be obtained by the posterior distribution of the latent variables, given the words observed. This relation is expressed in equation 2.7.

$$p(z,\phi,\theta|w,\alpha,\beta) = \frac{p(z,w,\phi,\theta|\alpha,\beta)}{p(w|\alpha,\beta)} \quad (2.7)$$

This equation is intractable to compute, due to its denominator [11] [12]. It is the marginal probability of the observations, and can be expressed as equation 2.8.

$$p(w|\alpha,\beta) = \int_\phi \int_\theta p(w|\alpha,\beta) \quad (2.8)$$

Equation 2.8 is computationally intractable because summing the joint distribution over all the terms found in the collection vocabulary is exponentially large. In this sense, the LDA algorithms provide an approximate inference to this posterior distribution, disclosing its related topics ϕ, its topic proportions θ, and its topic assignments z, that is, the documents latent structure.

The number of desired topics is also necessary. Setting the number of topics to work on can be based on the perplexity [12] or coherence measures [95].

Perplexity refers to a metric that gives the model's average uncertainty for each word in the dataset [79] [97]. In general terms, the idea is that the lower the model's perplexity score, the better its generalization performance. Equation 2.9 gives the perplexity score.

$$Perplexity = exp\left(-\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d}\right) \tag{2.9}$$

In equation 2.9, p (w_d) is the likelihood denoted by equation 2.8 to our corpus of e-mails D, and N_d is the total number of keywords in d-th document of D.

According to [14], perplexity is not aligned with human interpretability. This study showed that these perspectives of the topic models are not correlated. In this context, to obtain a measure that is closer to human judgment, the topic coherence measures are discussed. These measures offer a score that helps to assess how much the obtained topics are semantically interpretable, while perplexity is a score that assesses the topics as artifacts of statistical inference. Two measures of coherence are adopted: C_{UCI} and C_{UMass}. While the first refers to a measure that compares all words, through all possible combinations of pairs, an extrinsic measure, the second refers to an intrinsic measure, a measure that compares a word not with all the other words, but with its preceding and succeeding words [96].

C_{UCI} coherence measure is calculated over all the word pairs of the given top words. It is a measure based on a sliding window, and the Pointwise Mutual Information (PMI) [88].

The Pointwise Mutual Information is a utility measure to assess the associativity between two words. Equation 2.10 gives PMI:

$$PMI(w_i, w_j) = \log\left(\frac{P(w_i, w_j) + \varepsilon}{P(w_i) \cdot P(w_j)}\right) \tag{2.10}$$

where $P(w_i, w_j)$ is the probability of the words w_i and w_j occurring in the same word window, $P(w_i)$ and $P(w_j)$ are, respectively, the probabilities of w_i and w_j occurring individually, and ε is an added value to avoid a logarithm of zero.

The C_{UCI} score is given by equation 2.11:

$$C_{UCI} = \frac{2}{N \cdot (N-1)} \cdot \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} PMI(w_i, w_j) \tag{2.11}$$

In equation 2.11, N refers to the N-top words of a topic, and the remaining terms, as indicated in equation 2.10.

The C_{UMass} coherence measure is based on document co-occurrence counts, which is the document frequencies of the original documents from which the topics are learned. It is an asymmetrical measure based on segmentation, and logarithmic conditional probability [86]. The C_{UMass} score is given by equation 2.12.

$$C_{UMass} = \frac{2}{N \cdot (N-1)} \cdot \sum_{i=2}^{N} \sum_{j=1}^{i-1} \log \left(\frac{P(w_i, w_j) + \varepsilon}{P(w_j)} \right) \tag{2.12}$$

2.6 Word Embedding

Word Embedding refers to distributed representations of words, phrases, or documents in a vector space. It is done by mapping words from a context to a distributed, dense, continuous, fixed vector. This vector, with its dimensions, provides a meaningful representation for words, because it captures and expresses the word context, its relations with other words, as well as its semantic and syntactic similarity[67] [68] [37].

Instead of the words being mapped to discrete dimensions according to their quantity and, as is done in the Bag-of-Words (BoW) and Document-Term Matrix (DTM) approaches, they are mapped to a shared pre-fixed low dimensional space. Each of these dimensions does not necessarily coincide with specific concepts, but a distribution over which they can express a specific aspect, and one dimension may also concur for more than one aspect of word meaning. It also contributes to a lower dimensionality that, coupled to the pre-fixed size, allows words to be projected on the same space to be compared, and also decreases the computational complexity required. These Neural Language Models have presented great results in many natural language processing experiments.

Three word embedding techniques were used: Word2Vec, FastText, and Doc2Vec.

2.6.1 Word2Vec

Word2Vec is a method that, through a two-layer neural network, provides a representation of words from a large corpus of text (such as a large number of e-mails) as vectors [37].

This representation captures the relationships between words, their meanings, as well as their respective uses in similar contexts with other words. The vectors of words used in similar scopes are close to each other in the Word2Vec vector space.

As presented in [67], the Word2Vec method is implemented through two approaches: Continuous Bag-of-Words (CBOW), in which the main idea is to predict a target word using a context, and Skip-gram, that works inversely to CBOW, predicting surrounding context words from the target words.

In the CBOW approach, the input is the surrounding words, its architecture has a projection layer shared by all words in the input (which allows all words to be projected in the same space position), and the output is the target word [67]. The model trains weights for a softmax hidden layer, using stochastic gradient descent and the gradients obtained through back-propagation. The name of this approach comes from the fact that it uses a continuous distributed representation of the context, and the order of words in their source text does not influence this projection. In the Skip-gram approach, the input is the current word, and the output are the nearby words within a certain range before and after the current word [67] [66].

2.6.2 FastText

FastText refers to a technique that, through a neural network, allows the learning of word embeddings. The FastText approach is different from the Word2Vec since it does not associate a distinct vector for each word. It represents each word as a bag of character n-grams (does not treat it as an atomic entity), each of these character n-grams is represented by a vector, and the words are represented as the sum of these representations [13]. That is, it considers the morphology of the words. It also works with the CBOW and Skip-gram methods.

By considering each word as a characters n-gram composition, FastText allows generating better word embeddings for rare words, and also constructing vector representations for words out of the vocabulary (that do not appear in the training corpus), by the sum of the characters n-gram that compose them.

2.6.3 Doc2Vec

Doc2Vec is an unsupervised learning algorithm that implements distributed representations of sentences and documents proposed. It learns a fixed-length dense vector representation for any variable-length pieces of texts (such as a phrase, a sentence, a document, or, in our case, an e-mail body and/or header).

As presented in [56], the Doc2Vec method is implemented through two approaches: Distributed Memory version of Paragraph Vector (PV-DM) and a Distributed Bag-of-Words version of Paragraph Vector (PV-DBOW). Since this method is an extension of the Word2Vec approach towards pieces of texts, it uses the CBOW model (for the PV-DM approach) and the Skip-gram model (for the PV-DBOW approach) with the addition of a new vector for the entire piece of text in the target.

In the PV-DM approach, this new vector (Paragraph ID) is also used to predict the next word (in conjunction with the other input words). At the same time, the word vectors w is trained at each window, the D document vector is also, and, when the training ends, it represents the document. This document vector works as a memory that makes latent what is missing from the current window.

The PV-DBOW approach trains the paragraph vector to predict words in a small window. At each iteration of stochastic gradient descent, it samples a text window, then samples a random word from the text window and forms a classification task given the Paragraph Vector.

2.7 Classification

Classification is a supervised learning activity whose objective is to obtain a discriminating function that separates the samples into different classes. In this type of learning, the goal is to learn a mapping from the input data for a given output. The correctness, the labels, is provided along with the input data (i.e., there is supervision). For the purposes of this study, there are two classes: Phishing e-mail and Ham e-mail (the term used for legitimate e-mails).

For the classification activities, as well as for comparison purposes, eight classification algorithms were used, namely: Support Vector Machines (SVM), Naive Bayes Classifier, Logistic Regression for classification, k-Nearest Neighbor, Decision Trees, Random Forest, Extreme Gradient Boosting (XGBoost), and Multilayer Perceptron (MLP). Their main characteristics are described in the next subsections.

2.7.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) refers to a supervised learning algorithm, in which the objective is to find a hyperplane in the input variable space to best separate the data points into two classes. This choice is based on the hyperplane with the most significant margin, which is the hyperplane that presents the maximum distance between both the data points of both classes. By doing this, new data points can be sorted with more accuracy and precision.

The points that are closer to the hyperplane are named Support Vectors. They influence the position and orientation of the hyperplane, as well as the number of features that influence the dimension of the hyperplane [5].

2.7.2 Naive Bayes Classifier

This classifier assumes that features are independent of each other when applying Bayes' Theorem (conditional probability). The expression 'naive' comes from the fact that it assumes that all the features independently contribute to the given class' probability, which is a strong assumption and unrealistic for real data.

Mathematically, this algorithm assumes the off-diagonal values of the covariance matrix to be 0, that is, they are independent. The joint distribution is the product of individual univariate densities (assuming that they have Gaussian distribution) [10].

2.7.3 Logistic Regression

This classifier, which is similar to linear regression to classification tasks, is based on finding the values for the coefficients (B_0, B_1, B_2, ..., B_n) that weigh each feature (X_0, X_1, X_2, ..., X_n), after that, it performs its predictions transforming the output through the logistic function [5]. Thus, the probability of an e-mail being considered a phishing e-mail (class 1) or a legitimate e-mail (class 0), may be given by:

$$P(Class = 1) = \frac{1}{1 + e^{-g(x)}} \tag{2.13}$$

where:

$$g(x) = B_0 + B_1X_1 + B_2X_2 + ... + B_nX_n \tag{2.14}$$

These weights are estimated from the e-mails dataset, by the Maximum Likelihood method. If $P(Class = 1) > 0.5$, then the e-mail is phishing, and if $P(Class = 1) < 0.5$, the the e-mail is legitimate.

2.7.4 K-Nearest Neighbors - KNN

This algorithm is based on the idea that similar data points are arranged nearby in an n-dimensional space. This similarity is measured by the distance between the points (usually the Euclidean Distance, or the Mahalanobis Distance) [10]. Thus, for a new data point, its classification is predicted by validating the local posterior probability of each existing class by the average of the class membership over its K-nearest neighbors.

This algorithm is susceptible to the curse of dimensionality, since it is based on the distance between data points and its dimensions.

2.7.5 Decision Trees

Decision Tree algorithms are built over the binary tree representation, in which each node corresponds to a single input variable and, given a split point on it, there are branches that bind to new nodes, according to the instance data point values for this feature, that will be below or above of this split point value (for a numeric variable). This procedure is performed until the path arrives at a leaf node, in which the algorithm predicts the output variable class. At any of these nodes, the input set is split into subsets for the next nodes. When analyzing e-mails to check whether they are phishing or ham mail, the amount of times the words present in them, the weights of this amount, or the vectors representing theses e-mails are analyzed according to several split points (to find an optimal one, by information gain or Gini index criteria, for example), and at a leaf node level, it is predicted whether it is a phishing or a legitimate e-mail [5].

2.7.6 Random Forest

Random forest is an ensemble algorithm based on Bootstrap Aggregation (bagging technique). Ensemble is a machine learning technique that combines several base learning algorithms in order to produce a better predictive performance model. Bagging is a technique that uses the bootstrap algorithm to obtain a random sample from a given dataset with a replacement, so it trains the base learners and aggregates their outputs to provide a lower variance model.

This ML classification algorithm creates a set of decision trees on randomly multiple samples of the training set, gets a prediction from each tree, and, utilizing voting of these trees results, gives a better estimation for the test object's final class. In its approach, instead of getting optimal split points for trees, by the randomness of the selected subset of the training set, it selects suboptimal splits. Due to this, different models are created, which will be aggregated by combining their results [5].

2.7.7 XGBoost

Extreme Gradient Boosting (XGBoost) refers to an ensemble that implements (using the boosting technique) a scalable and accurate version of gradient boosting on decision trees. Boosting is a technique that trains models in succession, with each new base learner being trained to correct the errors made by the previous learners. Through the use of weighted versions of the data, more weight is increasingly given to misclassified examples. Learners are included sequentially until no further improvements can be made. The final predictions are obtained by weighted majority voting.

This algorithm offers high speed, performance, portability, and flexibility. It can optimize the loss function through three methods: gradient boosting, stochastic gradient boosting, or regularized gradient boosting. Its default base learners are tree ensembles, in which each is a set of classification trees (CART). As explained for the boosting technique, these trees are added sequentially, and each of them tries to reduce the misclassification of themselves from previous learners [15].

2.7.8 Multilayer Perceptron (MLP)

Refers to an artificial neural network based on perceptrons (a learning algorithm for binary classifiers, which is able to solve only linearly separable problems) as artificial neurons. Artificial Neurons are the essential components of an artificial neural network, in which a process that simulates a biological neuron working occurs. There are input connections that emulate the synapses and their forces, by assigning a weight to each input signal. These input values are summed by a linear combiner, which is also responsible for generating an activation potential (the network internal activation), by comparing this sum with an activation threshold. Thus, a non-linear activation function (a sigmoidal function, in MLP case) provides the neuron output signal.

In this ML classification algorithm, each row of these neurons is a layer. The Multilayer Perceptron (MLP) neural networks have three layers of nodes. The first, the input layer, is responsible for receiving the external stimuli. Then, there is a hidden layer that may be composed of one or more layers. Its function is to extract the environment's behavior, approximating any continuous function. Lastly, in this topology, the output layer provides the answers to the received stimuli.

This structure utilizes the back-propagation technique for training, which, by the gradient descent, fits the network parameters (including with non-linear activation functions) to better express a training set with its labels, iteratively reducing its error. It works in two steps: in the forward pass, the stimuli are passed from layer to layer until the output layer, which provides predictions. These results are compared with the expected output provided by the training set. The prediction errors and its function are then used in the second step called the backward pass, in which the weights and biases of the model are updated by the partial derivatives of the error function, the back-propagation algorithm doing these operations. This process is done until the network converges, or for a predetermined number of epochs (in which case the network will not necessarily converge) [5]. An epoch indicates how many times all the training vectors are used once to update the weights. This measure varies according to the type of learning, i.e, whether it is in online or in batch mode.

2.8 Related works

Machine learning and data-driven approaches have been increasingly employed to solve cybersecurity-related problems [74], [64] [32] [98]. The phishing detection research landscape shows that robust results have been obtained through natural language processing techniques. Most of this research is centered on how to extract, from the text and the metadata of the e-mail, highly distinctive features that allow identifying differences and similarities between these messages, in order to separate them into phishing or legitimate e-mails.

One of the first approaches for phishing e-mail detection based on machine learning was proposed by Fette et al. [31]. It generates features based on e-mail texts and properties, such as whether these e-mails contain javascript code, the number of links in the e-mail, or the number of dots in the present Uniform Resource Locators (URLs). It detected over 96% of the phishing e-mails when submitting the best ten features they found to the Random Forest classification algorithm. Also proposing to select a set of content-based and behavior-based features, Hamid et al. [43] achieved a 94% accuracy rate through the use of the Bayes Net Algorithm, which was fed with eight features.

Similarly in [20], from forty-eight features selected from the specialized literature (related to the e-mail body and header, Javascript and URLs), Daeef et al. proposed a phishing e-mail classification based on two stages, extracting features and submiting them to three ML classification algorithms. They attained an accuracy rate of 99.40%.

Also, using hand-crafted features extracted from the e-mail body and header (twenty-one features), Islam and Abawajy [47] proposed a 3-tier model classification, based on well-known ML algorithms. They obtained an accuracy rate of 97%, when distinguishing phishing and legitimate e-mails.

In [99], with 40 features extracted from the body, subject, and sender e-mail fields, and from the presence/absence of any script and URL, Toolan and Carthy proposed an analysis based on entropy and information gain measures. It utilized the C5.0 decision tree algorithm for classification, reaching an 84.6% success rate when classifying phishing against ham e-mails. Also, based on information gain to select features for phishing detection, Yasin and Abuhasan [110] used text stemming and a WordNet database to preprocess and enrich their e-mails representation. Through this approach, they obtained an accuracy mark of 99.1%, while the proposal PhishNet-NLP, presented by Verma et al. [105], achieved a phishing e-mail detection rate of 97%. The proposal of Verma et al. was based on NLP techniques to check whether e-mails were informative or actionable and other features extracted from the body and header of the e-mails. Developing an improved phishing e-mail classifier with better prediction accuracy and fewer numbers of features was the objective of Akinyelu and Adewumi [2]. They used a set of 15 phishing e-mail features, identified from the literature, and fed the random forest machine learning algorithm. An accuracy of 99.7% was achieved.

An analysis of techniques to promote feature reduction for phishing detection was detailed in [111]. Four techniques (Chi-Square, Information Gain, Latent Semantic Analysis - LSA, and Principal Component Analysis - PCA) were compared. In this approach, the use of these techniques was preceded by stemming, and the features were based on header contents and eventual URLs, besides the body of the e-mails. This proposal reached an accuracy rate of almost 98%. L'Huillier et al. [57] proposed an approach whose features are extracted three ways: structural features extracted directly from the text, features based on keywords, and features obtained through the application of LSA and Latent Dirichlet Allocation (LDA) techniques over the TF-IDF Matrix, which is generated from the corpus texts. It attained an F1-score mark of 99.58%, using 1,017 features to feed the SVM classification algorithm. Likewise, using NLP methods, Ramanathan and Wechsler presented phishGILLNET [82]. It is a 3-layer approach based on a topics model. Through the use of techniques such as Probabilistic Latent Semantic Analysis (PLSA) and Co-training, it obtained an F1-measure of 100% for 200 topics, and 98.3% for 25 and 10 topics (these topics were employed to express the features input to the AdaBoost classification algorithm). In [83], Ramanathan and

Wechsler proposed another phishing detection approach. This method also attempted to discover the entity/organization that the attackers impersonate during phishing attacks. This proposal employed Conditional Random Field (CRF), Latent Dirichlet Allocation (LDA), and the AdaBoost in its best variation, identifying the impersonated entity from messages classified as phishing with a discovery rate of 88.1%.

In [102], using the text as its primary features source, and also incorporating the domain knowledge and lexical features, an approach was presented that reached an F1-measure of 98%. It was based on DTM and TF-IDF, and its best mark was obtained through the use of the Logistic Regression classification algorithm. [44] and [103] proposed methods for phishing detection based on Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), also considering DTM and TF-IDF. The obtained features were submitted as input for several classifications algorithms, achieving, respectively, its best marks of 94.6% (using k-Nearest Neighbor - KNN classification algorithm) and 95.3% (using SVM with 30 features).

Unnithan et al. [76] compared the TF-IDF matrix's employment and the Doc2Vec representation to phishing e-mail detection. They used seven different classification algorithms to assess these two approaches, achieving their best mark (an 88.95% accuracy rate) through the SVM classifier fed by the Doc2Vec representation. Also, a word embedding approach, the proposal presented in [8] was based on the FastText technique. Through the syntactic and semantic similarity of e-mails extracted by the techniques employed, it attained an accuracy rate of 99%. The same authors proposed another approach [108], based on Word2Vec and Neural Bag-of-Ngrams, for phishing e-mails detection. The obtained representations fed some classifiers such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Multi-Layer Perceptron (MLP), reaching in its best variation a 99.1% F1 Score (using Word2Vec and LTSM).

The strategy employed in [72] was based on content and behavior-based features and also on word embedding (Word2Vec) techniques. The obtained features were imputed to a Neural Network classification algorithm, achieving an accuracy of 92.2%. In [33], using 240 features, 200 from Doc2Vec representation (to capture the syntax and semantics of the e-mails) and 40 content- and behavior-based features, Gangavarapu and Jaidhar introduced a hybrid metaheuristic to obtain a discriminative and informative feature subset crucial to Unsolicited Bulk E-mails (UBEs). When classifying e-mails as phishing or legitimate e-mail, this study achieved an accuracy of 99.4% employing the Multi-Layer Perceptron (MLP) for the classification task. Also proposing an approach to select the most discriminative feature set among 40 extracted features, [34] presented a structured procedure to extract and select content and behavior-based features to detect UBEs. Employing 27 of these 40 features (selected through a low variance filter) and the Randon Forest algorithm, the procedure obtained an F1-measure of 99.2 separating phishing and legitimate e-mails.

Chin et al. presented a deep packet inspection (DPI) in [17]. It was based on two components: phishing signature classification and real-time DPI, and its best mark was 98.39% accuracy (using an Artificial Neural Network - ANN) when detecting and mitigating phishing attacks. Based on Recurrent Neural Network (RNN), the approach proposed in [42] took sequences of integer values as inputs for this classification algorithm. These values were obtained abstracting the computer-native copy of an e-mail as a sequence of bytes into the high-level representation (unigrams), represented as unique integers. It attained an F1-measure of 98.63% and an accuracy rate of 98.91%. Through the use of Recurrent Convolutional Neural Networks (RCNN), Fang et al. [30] proposed THEMIS, that employed Word2Vec models e-mails at four levels, simultaneously (header, body, character and word). Its best mark was a F1-Score of 99.31% and an accuracy of 99.84%.

Considering this landscape, this study concentrated on generating more expressive features from the existing terms/words in e-mails (documents), and subjecting them to different machine learning algorithms, using enhanced techniques, in order to obtain improved results in classification tasks. It aimed to assess the performance of features obtained from these robust representation perspectives. Our interest went beyond showing an optimal accuracy (or other isolated metrics) for the models, but rather aimed to present the obtained results in the various utility measures (that are complementary [89]) in order to demonstrate the overall performance of the proposed approach not just its capabilities in one of the classes of this classification problem.

The datasets of most of the works listed in this section were obtained from the PhishingCorpus [75], and from the Spamassassin PublicCorpus [29], which are the main dataset (Dataset 1) adopted to evaluate the proposed approach. As exceptions, some of them, such as [83], [82], [8], [108], [30] and [17], used a clustered dataset in which PhishingCorpus and Spamassassin were part.

3 The Proposed Approach

Those who dream by day are cognizant of many things which escape those who dream only by night.

—EDGAR ALLAN POE

3.1 Detailing the proposed approach

Through the use of natural language processing and machine learning techniques, the proposed approach aimed to achieve refined predictions in phishing e-mail classification using the smallest possible number of features. The proposal is based on a multi-stage methodology, as expressed in Fig. 3.1, where the central purpose orbits around deriving more informative features, and feeds the chosen ML algorithms, training them using established strategies of folding.

The process begins by parsing the text from the e-mail to a vector structure. For each e-mail text, the corresponding label is assigned according to the e-mail collection of the dataset from which it originates (phishing or legitimate e-mail).

After this parsing process, the feature engineering phase starts, and these texts undergo a pre-processing step, where the texts are transformed into lowercase, and the punctuation marks, special characters, possible accents, and stopwords are removed. The terms obtained are then exchanged for their common base form, by reducing their respective inflectional forms and derivationally related forms (tokenization, POS tagging, and lemmatization tasks). This process also allows a moderate feature dimensionality reduction through the semantics of the terms and their synonyms.

These common base forms of the obtained terms were used on two fronts. In the first, based on the term/word count present in e-mails texts after the pre-processing step, a matrix (Document-Term Matrix - DTM), that relates these terms to the pre-processed e-mails texts, is obtained. In the second, fixed-size features are generated from the pre-processed texts through processes based on word embedding, which provides a representation in a fixed shared low-dimensional space for each e-mail.

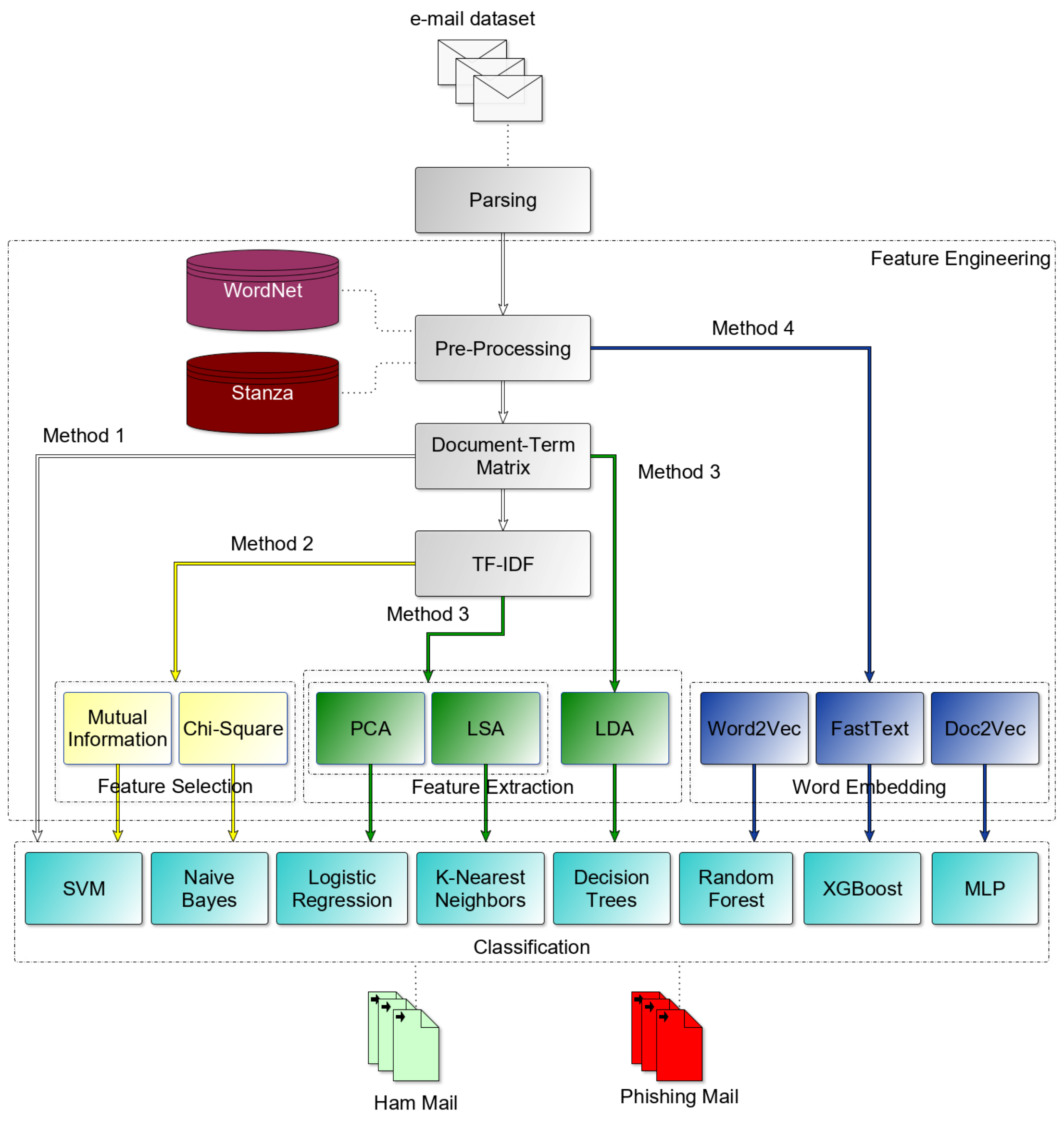


Figure 3.1 The main Architecture of the proposed approach

From the DTM, three methods are followed. All the terms obtained, arranged in DTM as features, that is, without acting directly on the high dimensionality and sparsity problems, are used (Method 1). Some of the input features from two criteria, Chi-Square or Mutual Information, are selected (Method 2). Alternatively, new features from this

Matrix are extracted by Principal Component Analysis, Latent Semantic Analysis, or Latent Dirichlet Allocation (Method 3).

The Method based on word embedding, Method 4, is also implemented in three perspectives. In the first case, the word vector representation is obtained by the interactions and similarities among words (Word2Vec). In the second case (FastText), this representation is also derived from these relations among words, but conceiving the word not as an atomic item, but as a sum of characters (that also has its own vector representation). In these two approaches, the vector representation for each e-mail in the dataset is obtained by the mean of the vectors representing the words present in them. Finally, in the latter case, the vector representation for the e-mail body text is obtained directly from the process that generates the vectors representing its words (Doc2Vec). It works as an auxiliary vector in generating the representative vectors of words and, at the end of the process, it contains a general representation of the submitted text.

The outputs of these four methods to represent the dataset e-mails are then submitted to classification algorithms which, after a learning process with enhanced techniques, provide models that can predict if an e-mail is phishing or ham mail, with excellent results. These global approach settings, stages, steps and details are presented in the next sections.

The remaining of this chapter is organized as follows. The main dataset (Dataset 1) and the data modeling process proposed in this thesis are exposed in Section 3.2, and the parsing and pre-processing phases are outlined in Section 3.3. The proposed methods are introduced in Section 3.4, and, from Section 3.5 to Section 3.8, each of these methods is presented in detail. Finally, the overall strategy for classification is described in Section 3.9.

3.2 Dataset 1 and Data Modeling

Two sources of raw data were considered for the main dataset of this thesis: the SpamAssassin Public Corpus [29] and the Nazario Phishing Corpus [75]. Respectively, they are assumed as the Ham Dataset and as the Phishing Dataset. These two compilations of e-mails are both public datasets and their use in evaluating phishing detection approaches is a widespread practice [42], [73]. The studies presented in [40], [35], [42], [33], [2], [110], [57], [60], [31], [20], [47] and [43] are examples of papers that employed the SpamAssassin Public Corpus and the Nazario Phishing Corpus as dataset sources.

The chosen dataset (Dataset 1) has 6,429 e-mails. From these, 4,150 were labeled as ham e-mails from the SpamAssassin Public Corpus (specifically from easy and hard ham, that represent legitimate e-mails in this collection of e-mail), and 2,279 labeled as phishing e-mails from the Nazario Phishing Corpus (specifically from phishing3.mbox).

Two other datasets were used in this thesis, Dataset 2 and Dataset 3. These are presented in Section 4.3.1 of Chapter 4.

Figure 3.2 presents the data modeling workflow for our approach.

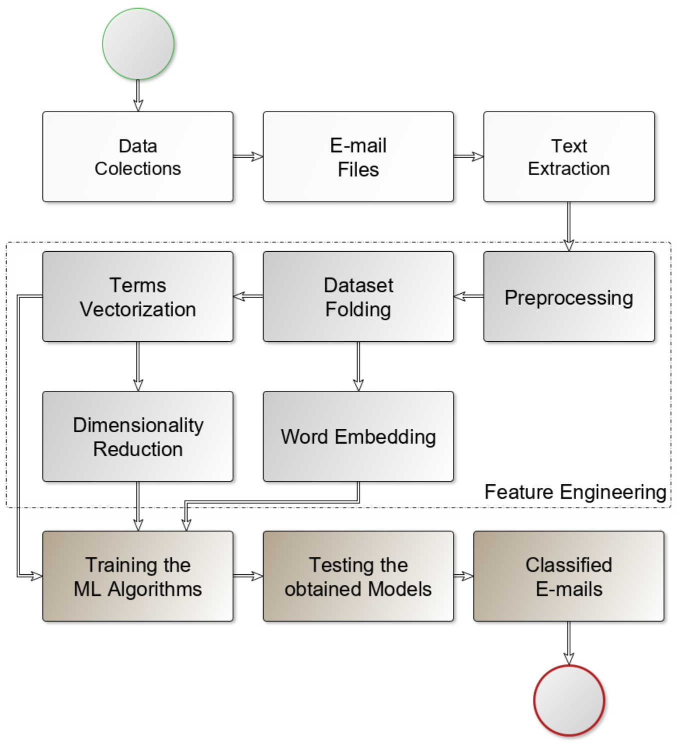


Figure 3.2 The data modeling workflow

As illustrated, in the proposed approach, the data modeling starts from these two collections of e-mails (the raw data), selecting the e-mail files from the subsets of interest. From these files, the texts of the e-mail bodies were extracted.

Then, initiating our feature engineering process, these texts underwent a pre-processing step and then were folded into two sets, training and test sets, with 4,500 and 1,929 samples, respectively. This split was done here to avoid any information leakage from the test set that would lead to biased results. From this stage onwards, the proposed operations were fitted over the training set, and the proposed transformations carried out over the training set and the test set.

Afterwards, according to the strategy of each method of the proposed approach, these texts were treated in different ways. Their words were associated with word embeddings to obtain a representation (in Method 4). Also, from a term vectorization step, a DTM was generated that was used directly in Method 1, where each remaining term stands for a feature of the e-mails. Over this DTM representation, a dimensionality reduction was also performed, where we selected the best features (Method 2) or extracted new ones (Method 3).

For each Method strategy and technique, the training set fed the machine learning algorithms, and the test set fed their resulting models, classifying the e-mails as phishing or ham e-mails.

The modeling data assumed varied vector and matrix shapes along with the proposed approach. These shapes, the methods, and the architecture of the proposed approach are presented in detail in the next Sections, using Dataset 1 as the modeling instance.

3.3 Parsing and Pre-Processing

From all the e-mail files, the text of the e-mail bodies was extracted and arrayed in a vector structure. That is, from 6,429 e-mail files, we generated a vector **e** with the same number of rows:

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{6429} \end{bmatrix}. \tag{3.1}$$

The corresponding labels of these e-mails were also vectorized. They were set according to the collection from which e-mail originated (from the phishing e-mail set or the legitimate e-mail set). This vector is delineated as **l**, thus:

$$\mathbf{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_{6429} \end{bmatrix}. \tag{3.2}$$

Next, our approach underwent a the pre-processing phase. After being arranged in an array structure, the texts of the e-mails bodies were transformed into lowercase and the punctuation marks, special characters, and any accents were removed. After this, the structure underwent a tokenization process, in which the terms/words of the texts were separated with white spaces (space, tab, and newline) used as delimiters. Then, the stopwords were removed. Finally, the terms underwent through a lemmatization process.

The pre-processing stage used **e** as input. All the letters of the texts in **e** were converted into lowercase. Then, punctuation marks (sentence endings; commas, semicolons, and colons; dashes and hyphens; brackets, braces and parentheses; and apostrophes, quotation marks, and ellipsis), special characters, and any accents were excluded.

Then, tokenization was performed over the remaining texts, the stopwords were removed, and POS tagging and lemmatization were executed. To conduct these last steps, we employed the WordNet and the Stanza databases. Some differences between these databases included WordNet performs POS tagging just over nouns, verbs, adjectives, and adverbs classes, while Stanza performed it over all the grammatical classes, besides providing Multi-Word Token (MWT) expansion and morphological feature tagging. Also, WordNet offered the synsets resource, while Stanza did not. However, Stanza offered other features such as annotations related to dependency parsing and named entity recognition (that were not used in this study, since they were tested, but did not improve the prediction results).

Considering these differences, both were tested in all our methods, each once, and the one with better results was established as the default for that method.

When using Wordnet, the remaining number of terms, our potential number of features, was 48,435 (48,435 out of 56,523 in the entire **e** - training and test portions) after the pre-processing stage. This was the initial number of features used in Method 4, and also in the LDA perspective of Method 3 in this thesis. When employing Stanza, this number of potential features was 47,107 (47,107 out of 54,774 in the entire **e** - training and test portions), which was used in Method 1, Method 2, and in the PCA and LSA perspectives of Method 3.

To elucidate the similarity and semantic-based reduction, we now detail some results.

Using WordNet, if POS tagging had not been performed, the number of initial features would have been 51,736 and if, in addition, the synsets and lemmatization had not been performed, the number of initial features would have been 54,413. Besides the feature joins (which correspond to words with the same synonym/lemma) made possible by WordNet synsets, it was also used to select only those features corresponding to words present in it. With the addition of this WordNet utilization, the number of features would increase to around 18,000, but the obtained results were not better than those already available in the related works. Similar behavior was observed when using Stanza for this last purpose.

If Stanza had been used just for tokenization, i.e., if lemmatization, MWT expansion, and POS and morphological feature tagging had not been performed, this number would have been 54,680. Furthermore, if punctuation marks, special characters, and any accents had not been removed, this number would have reached 80,311.

Thus, for both databases, the pre-processing steps also provided a potential dimensionality reduction.

At this point, each element of the vector **e** was pre-processed (two versions, processed with WordNet or with Stanza), and the data in it still had the same shape, namely 6,429 rows.

3.4 The proposed Methods

In general, except for Method 1, which directly submits the DTM (with its ranking based on the occurrence count) to the classification step, the entire discussion of this study up until now has been aimed at detailing the actions/implementations used to prepare and promote robust data to be submitted to the methods that will obtain the features for the classification activity.

The dimensionality reduction carried out by these methods can potentially provide benefits such as reducing computational complexity, processing time, and variance, as well as preventing overfitting, gaining a better understanding of the process that underlies the data, and also allowing better visual analysis of said data [62] [5]. These methods are detailed in the following sections.

3.5 Method 1 - features without dimensionality reduction

As already mentioned, the pre-processed texts present in **e** are divided in two sets: the training and test sets. Through the use of these two sets of **e** in a Bag-of-Words (BoW) model, a Document-Term Matrix (DTM) was constructed, represented by **F**:

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & f_{1,t} \\ f_{2,1} & f_{2,2} & f_{2,3} & \cdots & f_{2,t} \\ f_{3,1} & f_{3,2} & f_{3,3} & \cdots & f_{3,t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{6429,1} & f_{6429,2} & f_{6429,3} & \cdots & f_{6429,t} \end{bmatrix} \quad (3.3)$$

where t can be 48,435, when using WordNet, or 47,107, when using Stanza, in Dataset 1. For Method 4 and the LDA perspective of Method 3, the prediction results using WordNet were better than using Stanza, thus t was 48,435. For Method 1, Method 2 and for PCA and LSA perspectives of Method 3, improved marks are found using Stanza rather than Wordnet, thus t was 47,107.

The architecture of Method 1 is expressed in Figure 3.3.

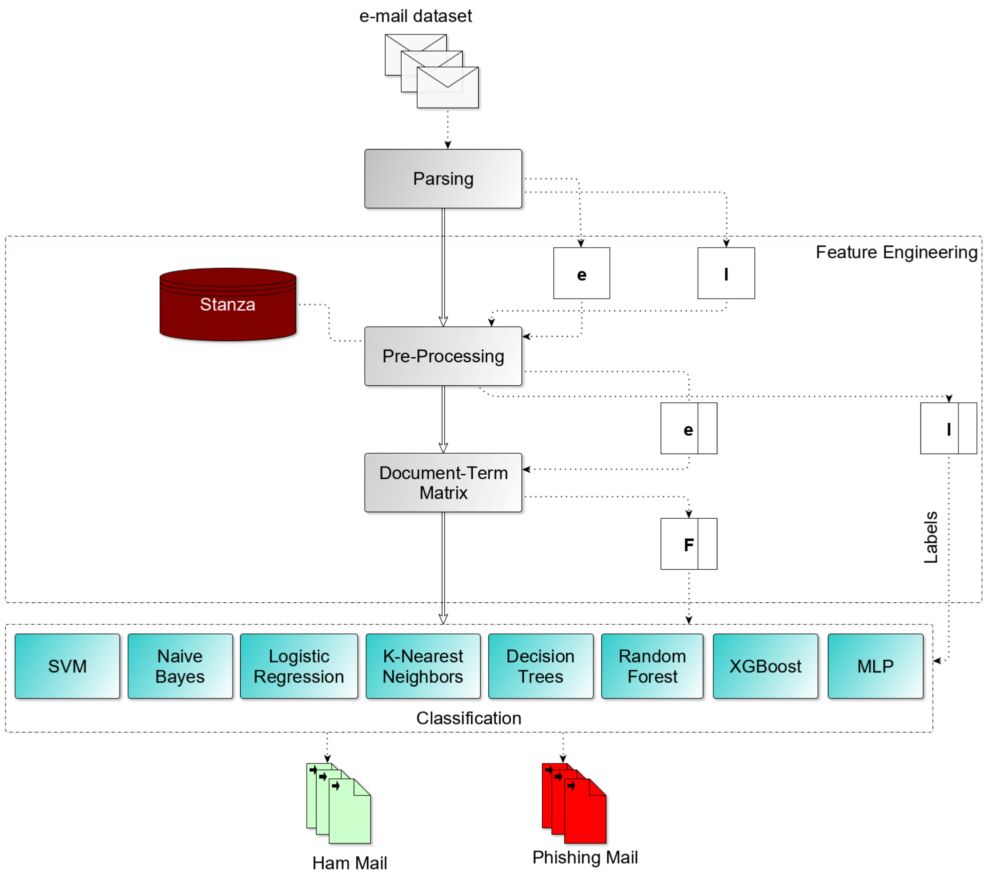


Figure 3.3 The Method 1 Architecture

This method submitted 47,107 features to the classification algorithms, which is the number of features that was provided by Lemmatization and Stanza-based processing.

Matrix **F** was obtained using the terms present in the training portion of **e** (47,107 out of 54,774 in the entire **e** - training and test portions) which were its columns index, through which the e-mail texts from the training and the test sets of **e** are represented (being **F** rows) indicating how many times each of these terms occurred in each e-mail text. **F** was also divided into two portions (training and test sets, with 4,500 and 1,929 samples, respectively).

In Method 1, **F** was used directly in the classification step.

3.6 Method 2 - Feature Selection

The utility measures order the terms obtained from texts according to their contribution in distinguishing one class from the other (those conveying more information regarding the process underlying the data). In this proposed approach, the terms/words obtained from the e-mails were ordered by their importance in the e-mail classification as either ham mail or phishing mail.

For this separation, in Method 2, two tests were used: the chi-square measure and the mutual information measure. Its architecture is presented in Figure 3.4. These two utility measures, respectively, were calculated between each feature and the target class. Then, the desired number of features was selected according to their best scores in these utility measures. In our approach, feature selection through the Chi-Square perspective was used to select between two and a hundred features out of the total number of features. The same number of features was selected by Method 2 through the Mutual Information perspective.

Thus, this method is used in this work with the following mode: after obtaining **F** (training and test sets), the TF-IDF weighting was performed over it. It is expressed by **G**:

$$G = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,47107} \\ g_{2,1} & g_{2,2} & g_{2,3} & \cdots & g_{2,47107} \\ g_{3,1} & g_{3,2} & g_{3,3} & \cdots & g_{3,47107} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{6429,1} & g_{6429,2} & g_{6429,3} & \cdots & g_{6429,47107} \end{bmatrix} \tag{3.4}$$

G was also divided into two portions (training and test sets, with 4,500 and 1,929 samples, respectively). Then, we chose the number of features to work on and performed the feature selection through each cited utility measure. This was done by fitting the chosen measure over the training set, and the proposed selections carried out over the training set and the test set.

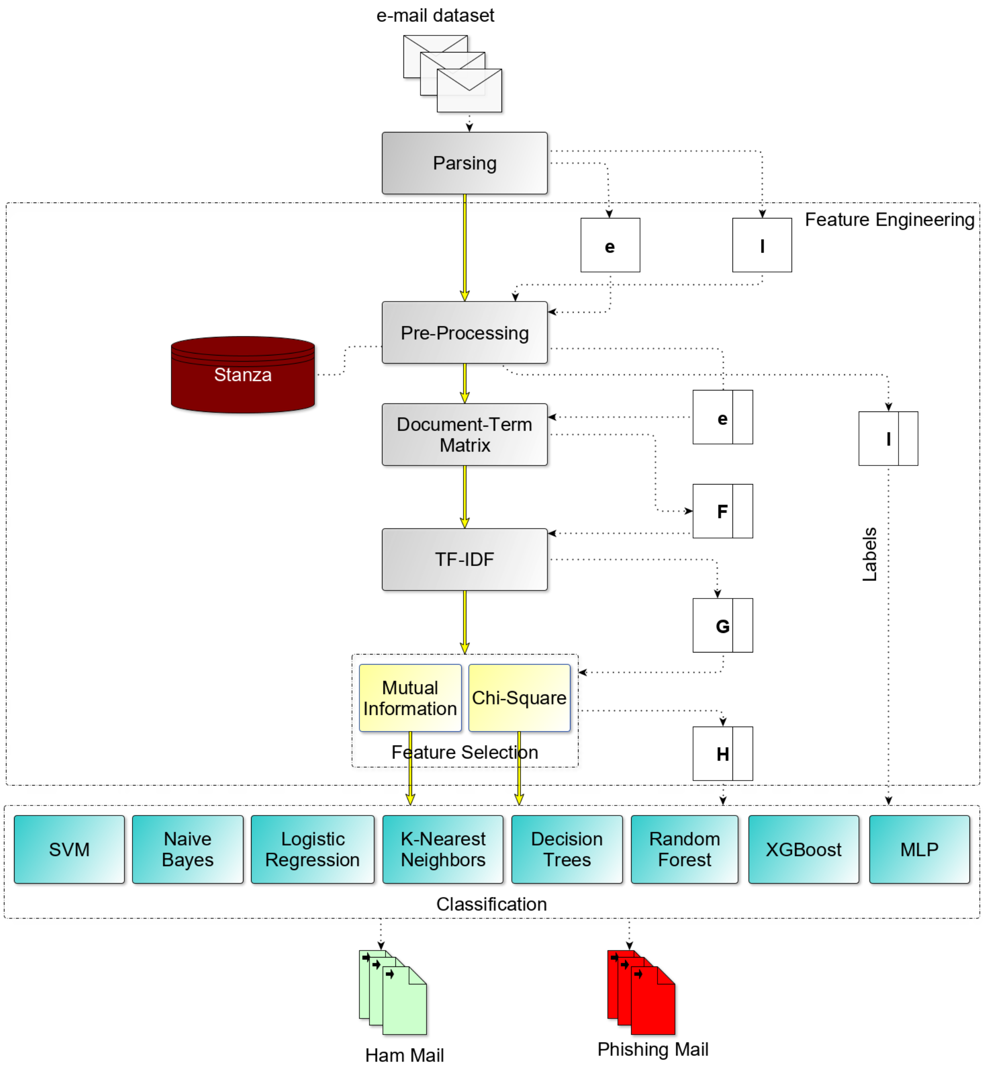


Figure 3.4 The Method 2 Architecture

For instance, our best setting, of the two perspectives of Method 2, was with Chi-Square, which employed one hundred selected features. The matrix **H** represents this setting for 6,429 instances of Dataset 1:

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,100} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,100} \\ h_{3,1} & h_{3,2} & h_{3,3} & \cdots & h_{3,100} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{6429,1} & h_{6429,2} & h_{6429,3} & \cdots & h_{6429,100} \end{bmatrix} \tag{3.5}$$

H was also divided into two portions (training and test sets).

3.7 Method 3 - Feature Extraction

The architecture of Method 3 is presented in Figure 3.5.

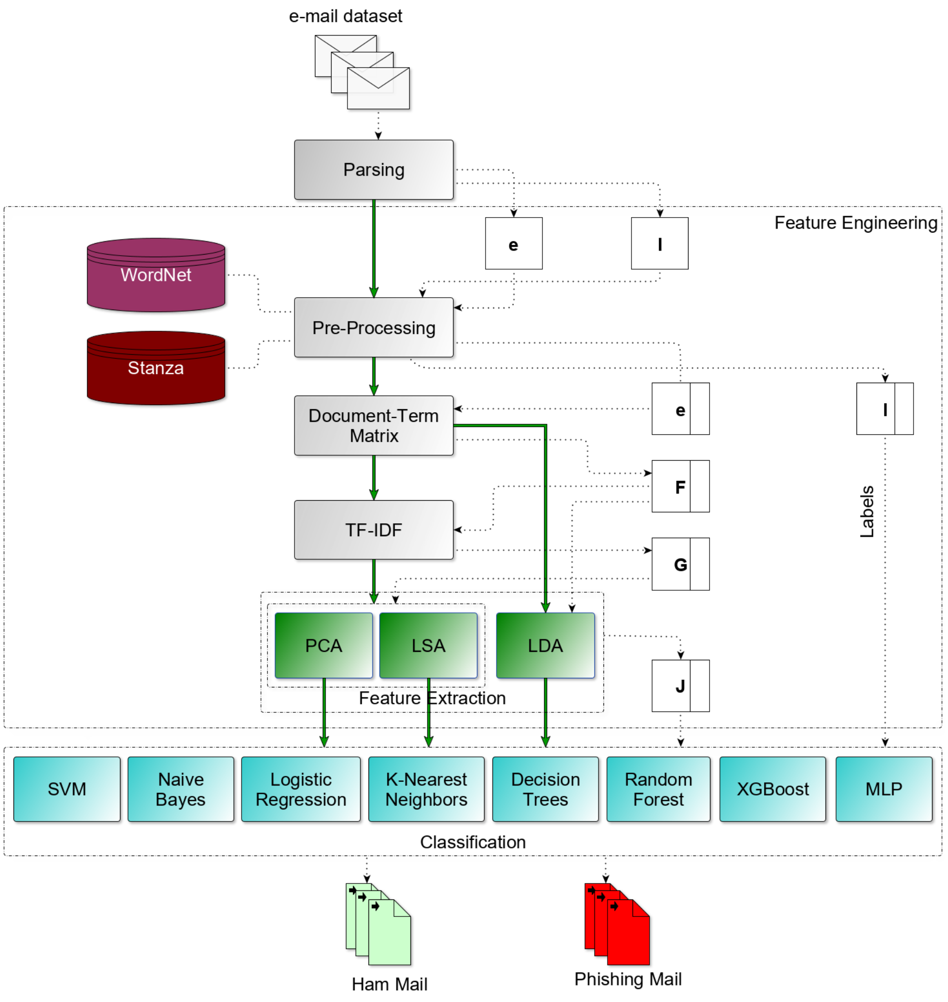


Figure 3.5 The Method 3 Architecture

The feature extraction method obtained new features from the original features set, generally features of lower dimensionality. Some transformations do it over the original feature space, i.e., the new feature space dimensions are combinations of the original high dimensional data. These new features are intended to be more representative, concentrating relevant information from the underlying data in a non-redundant shape.

For the purposes of this study, three feature extraction techniques were used in Method 3: the Principal Component Analysis, the Latent Semantic Analysis, and the Latent Dirichlet Allocation. In our approach, the feature extraction through these perspectives was used to extract between two and a hundred new features from those in the original features set. Thus, Method 3 was used in this work with the following mode: for the PCA and LSA perspectives, we chose the number of new features to work on and performed each proposed operation over **G**. However, for LDA, this was done over **F**.

For instance, our best setting, among the three perspectives of Method 3, was found using LSA, employing twenty-five extracted features. The matrix **J** represents this setting for 6,429 instances of Dataset 1:

$$J = \begin{bmatrix} j_{1,1} & j_{1,2} & j_{1,3} & \cdots & j_{1,25} \\ j_{2,1} & j_{2,2} & j_{2,3} & \cdots & j_{2,25} \\ j_{3,1} & j_{3,2} & j_{3,3} & \cdots & j_{3,25} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ j_{6429,1} & j_{6429,2} & j_{6429,3} & \cdots & j_{6429,25} \end{bmatrix} \tag{3.6}$$

J was also divided into two portions (training and test sets, with 4,500 and 1,929 samples, respectively).

3.8 Method 4 - Word Embedding

In Method 4, we used three techniques to embed words or documents (in our case, e-mails) into a vector space: Word2Vec [67], FastText [13] and Doc2Vec [56]. Its architecture is presented in Figure 3.6.

In the first perspective, the word vector representation was obtained by the interactions and similarities among words (Word2Vec). However, in the second case (FastText), this representation was also derived from these relations among words, but conceiving the word not as an atomic item, but as a sum of characters (which also has its own vector representations). In these two approaches, the vector representation for each e-mail in the dataset was obtained averaging all the vectors representing the words present in them.

Finally, in the latter case, the vector representation for the e-mail body text was obtained directly from the process that generated the vectors representing its words (Doc2Vec). It works as an auxiliary vector in generating the representative vectors of words and, at the end of the process, it contained a general representation of the submitted text.

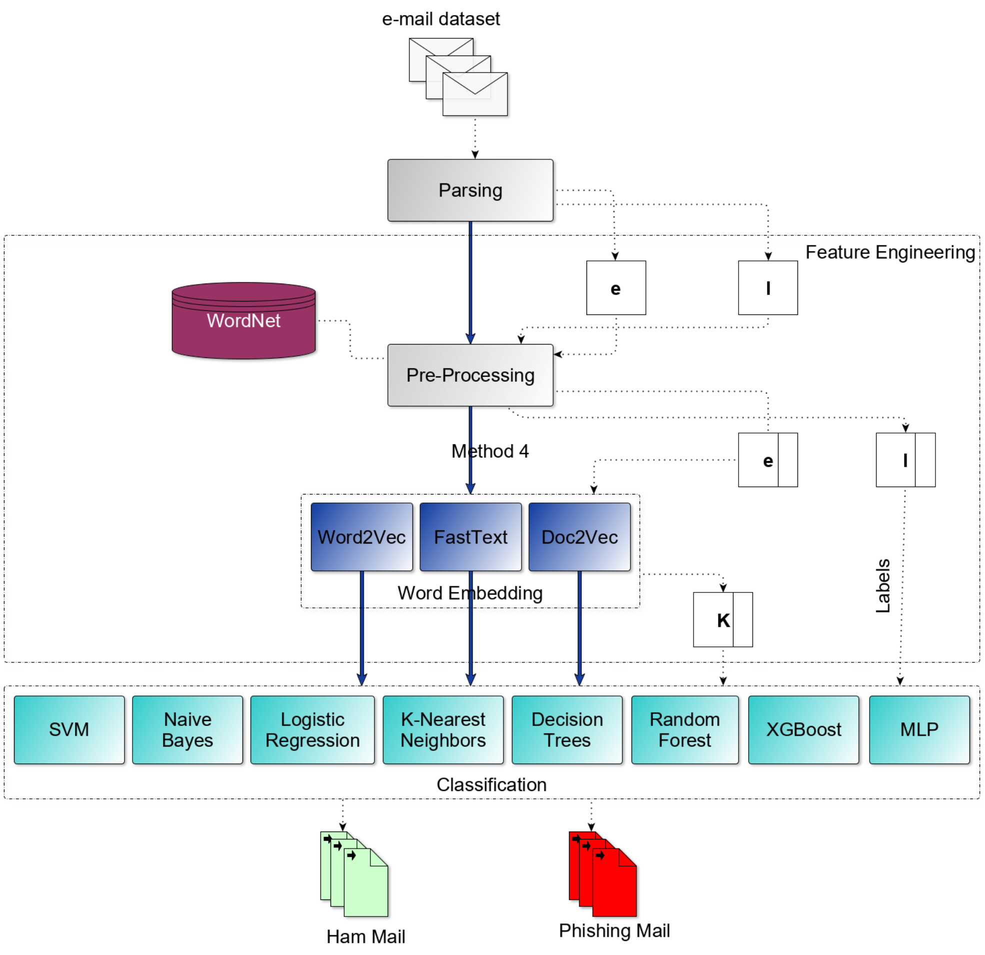


Figure 3.6 The Method 4 Architecture

In our approach, these three word-embedding perspectives were used to obtain a fixed 300-dimension dense vector representation for each e-mail, 300 features, from the terms present in **e** after the pre-processing step.

Thus, Method 4 was used in this work with the following mode: after extracting the texts from the e-mail bodies, passing them through the pre-processing steps, each word from this corpus in **e** was associated to a vector representation. For each of these techniques, these word vector representations were obtained in two ways: using pre-trained models based on external corpora or training a new model based on a vocabulary constructed from the text data of the e-mails in the training set.

These external corpus were:

- **Word2Vec:**
  - Word2Vec pre-trained word vector model, with a fixed-length of 300 dimensions, which contains 3 million 300-dimension English word vectors trained on Google News corpus, of about 100 billion words;
- **FastText:**
  - FastText pre-trained word vector model [69], with a fixed-length of 300 dimensions, which contains 1 million 300-dimension English word vectors trained with subword information (and without it) on Wikipedia 2017, UMBC web-based corpus and statmt.org news dataset, of about 16 billion tokens.
  - FastText pre-trained word vector model [69], with a fixed-length of 300 dimensions, which contains 2 million 300-dimension English word vectors trained with subword information (and without it) on the Common Crawl dataset, of about 600 billion tokens.
- **Doc2Vec:**
  - a Doc2Vec document vector model (using both DBOW and DM), with a fixed-length of 300 dimensions, trained by the authors over a corpus constructed from a dump of all Wikipedia articles, dated January 7, 2020, with a vocabulary of about 1 million words.

Regarding the variations based on new word-embedding models constructed from the e-mail datasets, for the three techniques employed, they were trained over the training set of **e**, where each word/token from this portion of the corpus was associated with a 300-dimension dense vector representation. For instance, our best setting, among the three perspectives of Method 4, was found using Word2Vec or FastText, using pre-trained models based on external corpus, employing a 300-dimension feature representation. The matrix **K** represents this setting for 6,429 e-mail samples of Dataset 1:

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & \cdots & k_{1,300} \\ k_{2,1} & k_{2,2} & k_{2,3} & \cdots & k_{2,300} \\ k_{3,1} & k_{3,2} & k_{3,3} & \cdots & k_{3,300} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{6429,1} & k_{6429,2} & k_{6429,3} & \cdots & k_{6429,300} \end{bmatrix} \tag{3.7}$$

K was also divided into two portions (training and test sets).

3.9 Classification

As explained previously, several techniques were used to obtain the features from the dataset e-mails used in this study. Thus, we have tested:

- Method 1 - that obtained these features, matrix **F**, without a reduction of dimensionality, from the BoW model (the most traditional and straightforward of these, expressed by the DTM with term occurrence count ranking);
- Method 2 - that obtained its features, matrix **H**, using perspectives that reduced dimensionality by feature selection through the Chi-square measure and Mutual Information measures over DTM with the TF-IDF ranking, matrix **G**;
- Method 3 - that obtained its features, matrix **J**, using perspectives that reduced dimensionality through feature extraction: PCA and LSA over DTM with TF-IDF ranking, matrix **G**, and LDA over DTM with term occurrence count ranking, matrix **F**;
- Method 4 - that obtained its features, matrix **K**, using perspectives that generated or associated fixed low dimensional vector representations (through word embedding, using Word2Vec, FastText, and Doc2Vec) of the words/texts in **e**.

Thus, all the classification algorithms used in this study were trained using features from all the methods perspectives and variations, which tried to feed them the most distinctive representation from their respective feature sets.

The proposal also established a course of action for the classification to provide a holistic approach to phishing detection, extracting the most suitable setting from each of the proposed stages. During the classifications tasks, many experiments were performed to find the best configuration of the hyper-parameters of each ML classification algorithm, as well as to implement the proposed strategies for dividing and folding the training/validation dataset before performing the obtained models on the test set.

This stage strategy started by separating the two portions (the training and test sets, already folded since **e** is pre-processed) of the feature set of each Method perspective. For Method 1, the training set referred to 70% of **F**, and the test set the remaining 30%; for the Method 2 perspectives, it referred to matrix **H**; for the Method 3 perspectives, it referred to matrix **J**; and for the Method 4 perspectives, it referred to matrix **K**.

In order to estimate the hyper-parameters of each classification algorithm, the folding plan presented in [25] was adopted. It suggests dividing the training/validation set (which corresponds to 70% of the entire dataset) into two subsets (folds), each with 50% of the samples. They were used as the training and validation sets, respectively, and then the inverse. This process was repeated five times (ten runs in total) for each combination of the various parameters of the running classification algorithm, using the cross-validation technique. At the end of each run, a new random sampling of 2 folds was performed on the samples, with the restriction of maintaining the proportion of the classes observed in the total training/validation set in each of the two subsets, that is, a proper stratification.

Also, in the training step, a wide variety of hyper-parameter settings were tested to estimate the befitting configuration to the proposed e-mail data. The specified folding and cross-validation plan was executed for each of these configuration sets, in each of the chosen ML algorithms, in order to evaluate their respective results.

After this training process, the ML models for the phishing classification problem were obtained from each of the employed ML algorithms. These models were then tested using unseen data, those in the test set (that corresponds to the remaining 30% of the entire dataset).

The training set consisted of 4,500 e-mails (70% of 6,429, as explained in Section 3.2), e-mails represented by their body text, 2,916 ham e-mails, and 1,584 phishing e-mails. The test set consisted of 1,929 e-mails (30% of 6,429), 1,251 ham e-mails, and 678 phishing e-mails, also called support. This split was performed over $\mathbf{e}$, before the DTM construction. In this sense, $\mathbf{e}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, $\mathbf{J}$ and $\mathbf{K}$ were already divided into training and test portions to properly fit the proposed operations to the training set, and to perform these transformations on both sets.

This classification strategy, integrated with the entire proposed architecture, is presented in Fig. 3.7.

Eight ML algorithms were used to perform the proposed classification (phishing detection task): Support Vector Machines (SVM) [78] [5], Naive Bayes Classifier [10], Logistic Regression for classification [5], k-Nearest Neighbor [10], Decision Trees [5], Random Forest [5], Extreme Gradient Boosting (XGBoost) [15] and Multilayer Perceptron (MLP) [5].

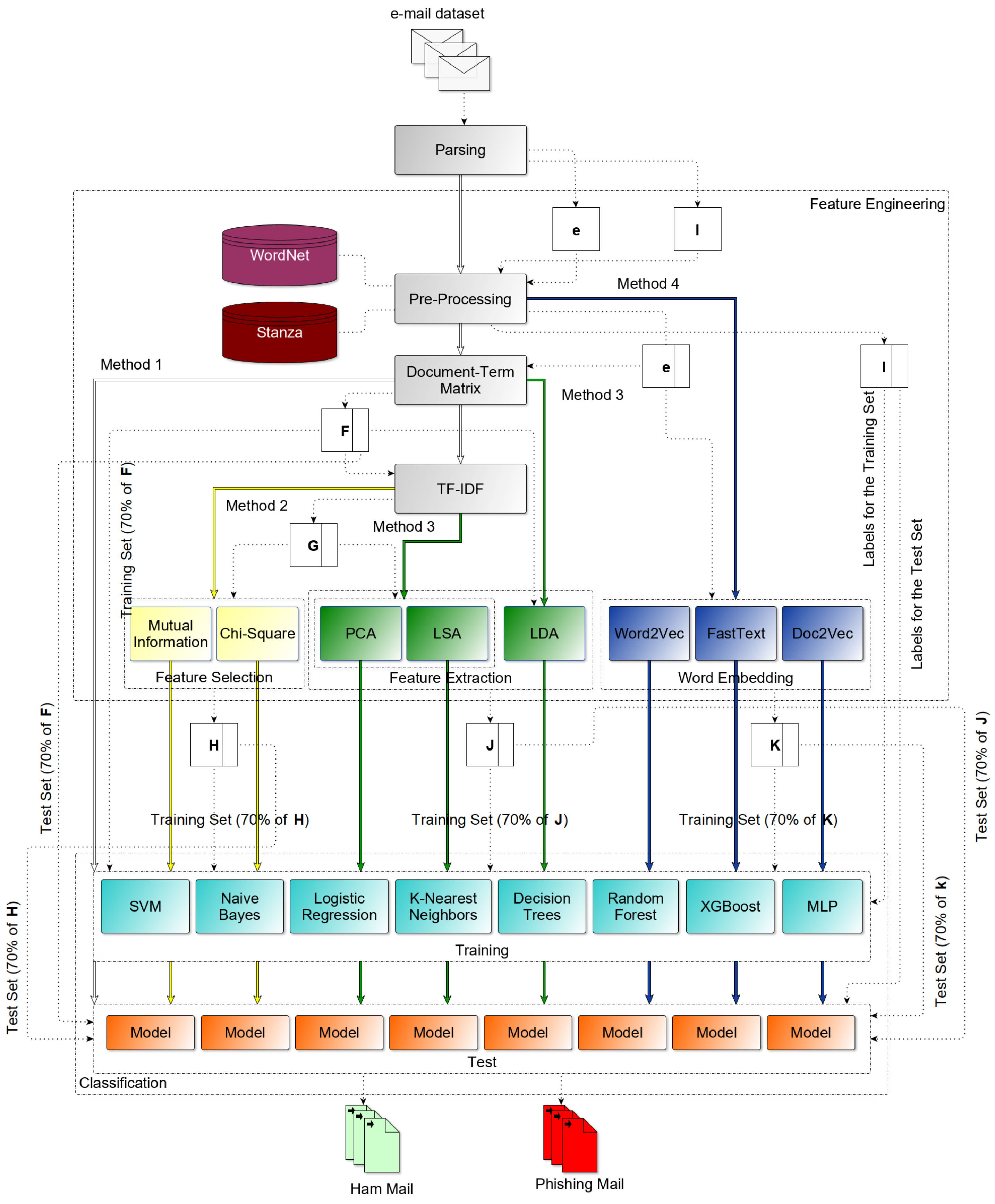


Figure 3.7 The main Architecture and the Dataflow of the proposed Approach

4 Results and Approach Evaluation

I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science, whatever the matter may be.

—SIR WILLIAM THOMSON, LORD KELVIN

In this Chapter, we provided a detailed evaluation of the proposed approach through its prediction results. The utility measures used to assess our methods' results are presented in Subsection 4.1. The results are described in Section 4.2 and Section 4.3. In Section 4.4, pertinent observations are discussed. Certain performance comparisons with related works are discussed in Section 4.5, and, in Section 4.6, the final considerations are presented.

4.1 Measures

To evaluate the performance of the classification algorithms in each perspective of the proposed methods, the following measures were used: accuracy, precision, recall, false positive rate, specificity and F1 score. Their equations are expressed in function of the true positive (t_{p}), false positive (f_{p}), false negative (f_{n}), and true negative (t_{n}) rates.

Accuracy (a):

$$\mathrm{a} = \frac{t_{\mathrm{p}} + t_{\mathrm{n}}}{t_{\mathrm{p}} + f_{\mathrm{p}} + t_{\mathrm{n}} + f_{\mathrm{n}}} \tag{4.1}$$

Precision p:

$$\mathrm{p} = \frac{t_\mathrm{p}}{t_\mathrm{p} + f_\mathrm{p}} \tag{4.2}$$

Recall (r), True Positive Rate (tpr) or Sensitivity:

$$\mathrm{r} = \frac{t_\mathrm{p}}{t_\mathrm{p} + f_\mathrm{n}} \tag{4.3}$$

False Positive Rate (fpr):

$$\mathrm{fpr} = \frac{f_\mathrm{p}}{f_\mathrm{p} + t_\mathrm{n}} \tag{4.4}$$

Specificity or True Negative Rate (tnr):

$$\mathrm{tnr} = 1 - \mathrm{fpr} \tag{4.5}$$

F1 Score (f_1):

$$\mathrm{f}_1 = 2 \cdot \frac{\mathrm{p} \cdot \mathrm{r}}{\mathrm{p} + \mathrm{r}} \tag{4.6}$$

4.2 Results

Initially, the number of features to be worked on with feature selection- and feature extraction-based methods was three hundred in order to establish a basis for comparison with the word embedding-based method. In this thesis, in all its perspectives, Method 4 was constructed over a fixed 300-dimension vector space, the size usually employed to construct this type of model. However, it was observed that the obtained results were slightly lower than the results for one-hundred features or fewer.

The feature amount choices for Method 2 and Method 3 and their perspectives are defined as follows:

For Method 2, the techniques used in its two perspectives were based on a ranking of the highest values that each feature provides according to a certain measure of utility (Chi-Square or Mutual Information), i.e, a measure that captures relationships between variables [21]. In this sense, they also followed the feature quantity settings employed in the Method 3 perspectives.

For Method 3, in the case of the PCA perspective, the Cumulative Percentage of Total Variation (CPTV) [77] [85] [48] was employed to define its feature amount, whereas, for

LSA perspective, this decision was made based on empirism [52], as already mentioned in Subsections 2.5.1 and 2.5.2 of Chapter 2.

Thus, since, in PCA, working with an amount of around eighty and ninety percent (depending on the practical details of the dataset under analysis) of the initial variance [48] is typically indicated, amounts of principal components between 2 and 100 were chosen as the number of features, which respectively represent about 86.29% and 98.70% of the variance. The results for these feature amounts were better than for those shown when 160 features were selected (99.00% of the initial variance), possibly because this last variance percentage also captured, along with the tendency of the underlying process, a certain amount of noise. For LSA, we also tested several quantities of singular values between 2 and 100, trying to achieve the best classification predictions with the least amount of features. Therefore, variations obtaining 2, 3, 5, 10, 25, 50, and 100 features from these perspectives of Method 2 and Method 3 were tested.

For the LDA perspective of Method 3, the number of features to extract through this process was chosen based on two utility measures, perplexity, equation 2.9, and coherence (given by the C_{UCI} and C_{UMass} scores, equations (2.11) and (2.12), respectively), explained in Subsection 2.5.3 of Chapter 2. Given the values obtained, the three best scores of each measure were chosen as the number of topics for the LDA models, and also, for comparison, the worst score of each was also chosen.

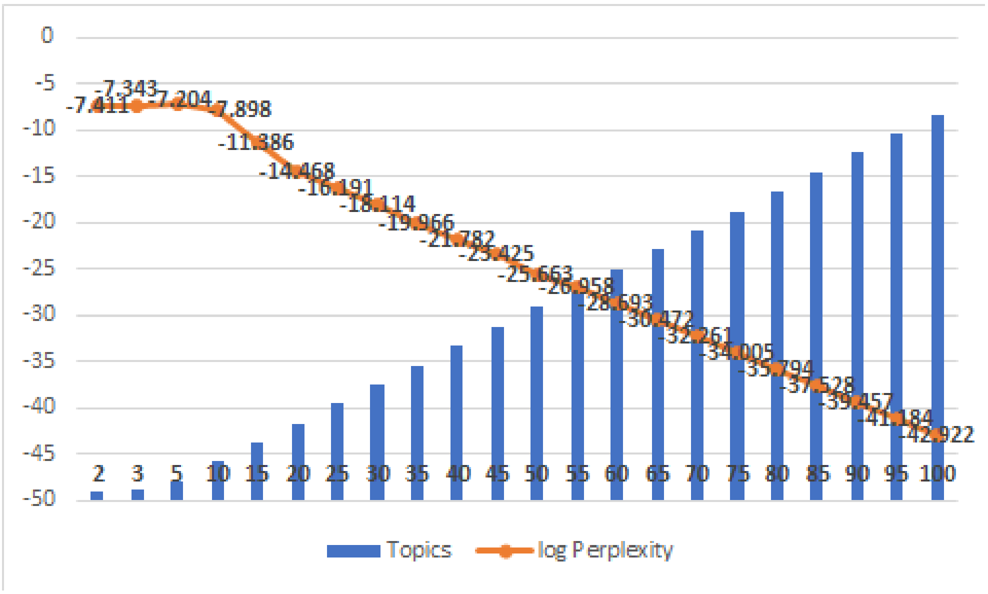


Figure 4.1 The log Perplexity of the proposed LDA models

Fig. 4.1 shows the log perplexity for some amounts of topics between 2 and 100.

Except for the variations with two and with three topics, for all others, the more topics, the lower the value of log perplexity. The three best scores for log perplexity were obtained for five, two, and three topics, while the worst mark was achieved for 100 topics.

Regarding the coherence scores presented in Fig. 4.2, there were many oscillations in these measures when the number of topics in the LDA model was increased, and it was observed that the two highest scores obtained in the C_{UCI} were for 10, 2 and 35 topics, and in the C_{UMass} these were for 2, 35 and 95 topics, while the worst marks of these measures were found for 5 and 100 topics, respectively.

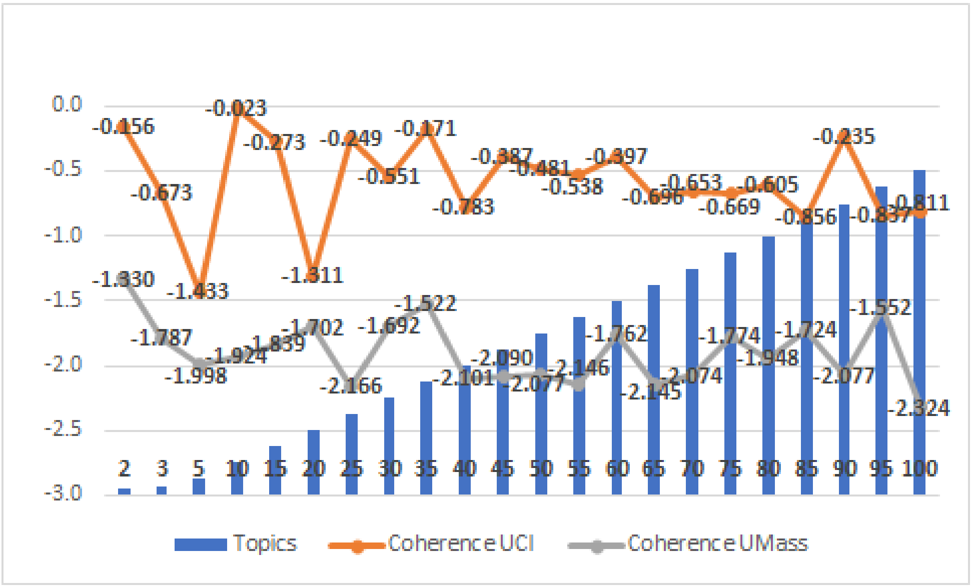


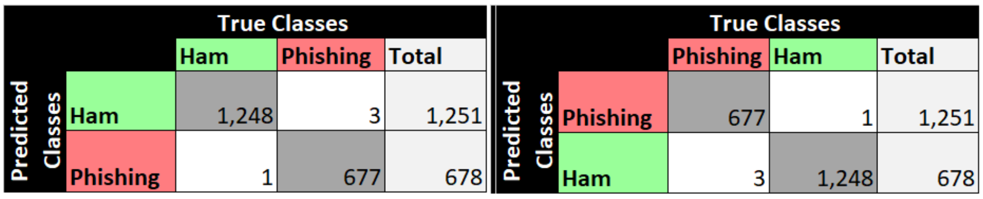
Figure 4.2 The Coherence measures of the LDA perspective models

Therefore, the number of topics chosen for the LDA models to be used in Method 3 were two, three, five, ten, thirty-five, ninety-five, and one-hundred. Thus, the four proposed methods were used in this thesis as follows:

- Method 1 used all features present in **F**, i.e., 47,107 features;
- Method 2 perspectives used the features present in **H**, that selected from two to one-hundred features from **G**;
- Method 3 perspectives used the features present in **J**, that extracted two to one-hundred features from those in **G** for PCA and LSA, and extracted from two to one-hundred features from those in **F** for LDA;
- Method 4 perspectives used the features in **K**, that generated 300-dimension vector representations from the terms in **e**.

The results expressed in this Section and Section 4.3 are the weighted measures obtained from both the phishing detection classes (phishing e-mail or ham e-mail) and their respective samples. The proposed measures were computed for each label, and then an average was computed considering the proportion of the true response for each label in the test set, i.e., weighted by support (the number of true instances for each label). In this sense, this type of average took label imbalance into account.

To illustrate how the weighted averages work for these measures, suppose that for a given ML algorithm, the confusion matrices, presented in Fig. 4.3, were obtained for the test set of Dataset 1.



Predicted Classes	True Classes: Ham	True Classes: Phishing	Total
Ham	1,248	3	1,251
Phishing	1	677	678

Predicted Classes	True Classes: Phishing	True Classes: Ham	Total
Phishing	677	1	1,251
Ham	3	1,248	678

Figure 4.3 Confusion Matrices - Example

It is observed that the first confusion matrix used the ham class as the reference class (positive), and the second confusion matrix used the phishing class as positive. This way, the scores indicated in Table 4.1 were obtained for the chosen evaluation measures.

Table 4.1 Measure Values for both Classes and the Weighted Average

Reference	t_p	t_n	f_p	f_n	total	Accuracy	Precision	Recall	F1 score
Ham Class	1248	677	3	1	1929	0.9979	0.9976	0.9992	0.9984
Phishing Class	677	1248	1	3	1929	0.9979	0.9985	0.9956	0.9971
Weighted Average						0.9979	0.9979	0.9979	0.9979

The weighted average was calculated using the true instances for each reference class (support) as the proposed weight of the scores available for both classes. The scores for these weighted averages are also available in Table 4.1. For the measurement of the F1 Score, for instance, the value of 99.79% was obtained from the calculation expressed in equation 4.7.

$$\frac{1,251 \cdot \mathrm{f}_{1\,(HamClass)} + 678 \cdot \mathrm{f}_{1\,(PhishingClass)}}{1,929} = 0.9979 \tag{4.7}$$

In the following subsections, the best results of the proposed approach are presented. The tables with the complete results of all the variations and perspectives analyzed, as well as additional descriptions about them, are shown in Appendix A.

4.2.1 Method 1 - Approach based on Document-Term Matrix without feature reduction techniques

The values arranged in Table 4.2 refer to the perspective based on the Document-Term Matrix (**F**) that used all the terms obtained from the Bag-of-Words model as features.

Table 4.2 Method 1: Bag-of-Words and Document-Term Matrix - 47,107 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9963	0.9964	0.9963	0.9963
Naive Bayes	0.9562	0.9565	0.9562	0.9563
Logistic Regression	**0.9974**	**0.9974**	**0.9974**	**0.9974**
KNN	0.9948	0.9948	0.9948	0.9948
Decision Trees	0.9917	0.9917	0.9917	0.9916
Random Forest	0.9937	0.9938	0.9937	0.9937
XGBoost	0.9963	0.9964	0.9963	0.9963
MLP	0.9969	0.9969	0.9969	0.9969

As stated earlier, this method did not address the high dimensionality (47,107 dimensions), sparsity (roughly 0.9982, that is, only about 0.18% of the data were non-zero values), and the represented context portion in the vector space model issues of the obtained matrix. However, it was also measured to serve as a baseline/benchmark for the other methods proposed in this thesis. Thus, due to these problems, it demanded more processing capacity and time than a method that addresses these questions (due to its higher complexity), although this perspective reached an F1-score of 99.74% as its best rate. This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 10, penalty as l2, and the rest of its parameters in the default setting.

This method, which only uses the e-mail bodies, performed better than similar approaches, with features derived from the headers, bodies, and links in the e-mails, described in [105], and [110], which obtained respectively 97% and 99.1% as their best phishing detection rate. It also outperformed other classical approaches based on e-mail properties, such as [31], that achieved a measure of 96% in identifying phishing e-mails.

If Stanza had not been employed to perform the lemmatization process, the best result would be a F1-Score of 99.37%, also using the Logistic Regression algorithm. Except for the Naive Bayes algorithm, which was slightly larger in this setting with no lemmatization, all the classification algorithms presented poor performance for their prediction results (Table A.3 in Appendix A).

4.2.2 Method 2 - the perspective based on the Chi-Square measure

For this Method 2 perspective, the Chi-Square measure was used as a dimensionality reduction approach. Its prediction assessment values are available in Tables 4.3 and 4.4. From the original features in the DTM (**F**) columns weighted through the use of TF-IDF (**G**, a desired number of features was selected (**H**) based on this measure.

Table 4.3 Results of the Chi-Square Perspective of Method 2 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9974	0.9974	0.9974	0.9974
Naive Bayes	0.9703	0.9708	0.9703	0.9700
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9984	0.9984	0.9984	0.9984
Decision Trees	0.9922	0.9922	0.9922	0.9922
Random Forest	**1.0000**	**1.0000**	**1.0000**	**1.0000**
XGBoost	0.9995	0.9995	0.9995	0.9995
MLP	0.9984	0.9984	0.9984	0.9984

The results attained through this perspective, with one-hundred features, are presented in Table 4.3. In this setting, we obtained accuracy, precision, recall, F1 score, and Specificity rates of 100%, which is, to the best of our knowledge, the highest result for phishing detection research using only 100 features. It is the best mark using the Chi-Square measure in Method 2. This highly prized measure was achieved using the Random Forest ML classification algorithm, with entropy as a function to measure the quality of a split, $\log_2 100$ as the number of features to consider when looking for the best split, ten as the minimum number of samples required to split an internal node, and the rest of the parameters in the default setting.

The results expressed in Table 4.4 refer to the best marks attained in the remaining proposed variations using this perspective.

Table 4.4 Results of the Chi-Square perspective of Method 2 - all feature set variations

features	Algorithm	Accuracy	F1 Score
2	KNN	0.9708	0.9707
3	KNN	0.9734	0.9733
5	KNN	0.9744	0.9743
10	XGBoost	0.9896	0.9895
25	Random Forest	0.9958	0.9958
50	Random Forest and XGBoost	0.9995	0.9995
100	**Random Forest**	**1.0000**	**1.0000**

4.2.3 Method 2 - Feature Selection: the perspective based on the Mutual Information measure

For this Method 2 perspective, the Mutual Information measure was used as a dimensionality reduction approach. Its prediction assessment values are available in Tables 4.5 and 4.6. From the original features inj the DTM (**F**) columns weighted with the use of TF-IDF (**G**), a desired number of features was selected (**H**) based on this measure.

Table 4.5 Results of the Perspective Mutual Information of Method 2 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9844	0.9844	0.9844	0.9843
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	**0.9990**	**0.9990**	**0.9990**	**0.9990**
XGBoost	0.9984	0.9984	0.9984	0.9984
MLP	0.9969	0.9969	0.9969	0.9969

The results attained through this perspective with twenty-five features are presented in Table 4.5. This setting obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.90%. It was the best mark using the Mutual Information measure in Method 2. This measure was achieved using the Random Forest ML classification algorithm, with entropy as a function to measure the quality of a split, five as the number of features to consider when looking for the best split, two as the minimum number of samples required to split an internal node, and the rest of the parameters in the default setting.

The best marks reached by Method 2 through the Mutual Information measure, in the remaining proposed variations, are presented in Table 4.6.

Table 4.6 Results of the Perspective Mutual Information of Method 2 - all feature set variations

features	Algorithm	Accuracy	F1 Score
2	KNN	0.9797	0.9796
3	Decision Trees	0.9838	0.9838
5	Decision Trees	0.9932	0.9932
10	K-Nearest Neighbors	0.9974	0.9974
25	**Random Forest**	**0.9990**	**0.9990**
50	Random Forest and XGBoost	0.9984	0.9984
100	Random Forest and XGBoost	0.9984	0.9984

4.2.4 Method 3 - Feature Extraction: the perspective based on an Principal Component Analysis

For this Method 3 perspective, Principal Component Analysis was used as a dimensionality reduction approach. Its prediction assessment values are available in Tables 4.7 and 4.8. From the original features in the DTM (**F**) columns weighted through the use of TF-IDF (**G**), a desired number of features was extracted (**J**) based on the principal components, projecting the original feature set in a reduced low-dimension space.

Table 4.7 Results of the PCA perspective of Method 3 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9760	0.9764	0.9760	0.9759
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	0.9984	0.9984	0.9984	0.9984
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9969	0.9969	0.9969	0.9969

The results attained through this perspective, with ten features, are presented in Table 4.7. This setting obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.95%. It is the best mark using PCA in Method 3. This measure was achieved by using the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of the parameters in the default setting.

The results expressed in Table 4.8 refer to the best marks attained in the remaining proposed variations using this perspective.

Table 4.8 Results of the PCA Perspective of Method 3 - all feature set variations

features	Algorithm	Accuracy	F1 Score
2	K-Nearest Neighbors	0.9953	0.9953
3	K-Nearest Neighbors	0.9974	0.9974
5	Decision Trees	0.9984	0.9984
10	**XGBoost**	**0.9995**	**0.9995**
25	XGBoost	0.9995	0.9995
50	Logistic Regression	0.9995	0.9995
100	XGBoost	0.9995	0.9995

4.2.5 Method 3 - Feature Extraction: the perspective based on Latent Semantic Analysis

For this Method 3 perspective, Latent Semantic Analysis was used as a dimensionality reduction approach. Its prediction assessment values are available in Tables 4.9 and 4.10. From the original features in the DTM (**F**) columns weighted through the use of TF-IDF (**G**), a desired number of features was extracted (**J**) based on singular values, projecting the original feature set in a reduced low-dimension space.

The results attained through this perspective with twenty-five features are presented in Table 4.9. In this setting, accuracy, precision, recall, F1 score, and Specificity rates of 100% were obtained, which is, to the best of our knowledge, the highest result in phishing detection research using just 25 features. It is the best mark using the LSA measure in Method 3. This highly prized measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5,

Table 4.9 Results of the LSA perspective of Method 3 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9734	0.9734	0.9734	0.9733
Logistic Regression	0.9995	0.9995	0.9995	0.9995
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9984	0.9984	0.9984	0.9984
Random Forest	0.9995	0.9995	0.9995	0.9995
XGBoost	**1.0000**	**1.0000**	**1.0000**	**1.0000**
MLP	0.9995	0.9995	0.9995	0.9995

the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of the parameters in the default setting.

The best marks attained by Method 3 through the Latent Semantic Analysis, in the remaining proposed variations, are presented in Table 4.10.

Table 4.10 Results of the LSA Perspective of Method 3 - all feature set variations

features	Algorithm	Accuracy	F1 Score
2	XGBoost	0.9953	0.9953
3	KNN and Random Forest	0.9963	0.9963
5	XGBoost and Randon Forest	0.9979	0.9979
10	Random Forest	0.9995	0.9995
25	**XGBoost**	**1.0000**	**1.0000**
50	Random Forest and XGBoost	0.9995	0.9995
100	XGBoost	0.9995	0.9995

4.2.6 Method 3 - Feature Extraction: the perspective based on Latent Dirichlet Allocation

The Method 3 had yet another perspective to extract features, which was based on the use of Latent Dirichlet Allocation (LDA), whose prediction assessment values are in Tables 4.11 and 4.12. From the topics extracted from the e-mail texts available in **F**, representations of the e-mails (**J**) were obtained in terms of the probability distribution of these topics, specific feature vectors for each of these messages.

Table 4.11 Results of the LDA perspective of Method 3 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9943	0.9943	0.9943	0.9943
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	0.9974	0.9974	0.9974	0.9974
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9974	0.9974	0.9974	0.9974

Table 4.11 presents the marks achieved by Method 3 through LDA with ten topics. This approach obtained accuracy, precision, recall, and F1 score measures of 99.95%, FPR of 0%, and a neat specificity of 100%. It was the best mark using LDA in Method 3. This highly prized measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, the rest of the parameters in the default setting. For this variation of LDA perspective, with ten topics, all the classification algorithms used had marks equal to or higher than 99.43%, the best measure presented by Method 1 (F1 score of 99.43% for MLP).

The marks obtained by Method 3 through the LDA with thirty-five topics were also the best for this perspective, as well as those present in Table 4.11, with the same accuracy, precision, recall and F1 score measures of 99.95% with the XGBoost classification algorithm.

The best marks reached by Method 3 through the LDA, in the remaining proposed variations, are presented in Table 4.12.

4.2.7 Method 4 - Feature Generation: the perspective based on Word2Vec

Table 4.13 presents the marks achieved by Method 4 through the Word2Vec perspective with vocabulary sourced from the pre-trained Google News corpus word vector model. As shown in Table 4.13, this approach attained accuracy, precision, specificity, sensitivity, and F1 score measures of 100%, which is, to the best of our knowledge, the highest result obtained in phishing detection research. This highly prized measure was achieved through the K-Nearest Neighbors classification algorithm, with the number of neighbors

Table 4.12 Results of the LDA Perspective of Method 3 - all feature set variations

features	Algorithm	Accuracy	F1 Score
2	Decision Trees	0.9958	0.9958
3	Decision Trees	0.9974	0.9974
5	XGBoost	0.9969	0.9969
10	**XGBoost**	**0.9995**	**0.9995**
35	**XGBoost**	**0.9995**	**0.9995**
95	XGBoost	0.9990	0.9990
100	XGBoost and KNN	0.9984	0.9984

as 1, the weight function as uniform, and the rest of the parameters in the default setting. Except for the Naive Bayes algorithm, all other algorithms achieved marks above 99%, three of which reached a 99.90% precision rate (Logistic Regression, XGBoost, and Multilayer Perceptron).

Table 4.13 Method 4: Feature Generation through the Word2Vec perspective - Vocabulary Source: pre-trained Google News corpus word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9650	0.9654	0.9650	0.9648
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Decision Trees	0.9901	0.9901	0.9901	0.9901
Random Forest	0.9979	0.9979	0.9979	0.9979
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	0.9990	0.9990	0.9990	0.9990

Although it did not reach 100% accuracy, Method 4 through Word2Vec with a vocabulary built from the dataset obtained excellent results in the prediction assessments, Table 4.14. It achieved a F1 score of 99.95% (FPR of 0%, that is 100% of specificity) in 4 out of 8 classification algorithms used, and 99.9% in 3 out of 8 of these. The four best results were obtained from the Logistic Regression, K-Nearest Neighbors, Random Forest, and Multilayer Perceptron algorithms. Due to vocabulary construction, this approach spent more time and consumed more processing power than the one implemented from the pre-trained Google News corpus word vector model.

Table 4.14 Method 4: Feature Generation through the Word2Vec perspective - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9990	0.9990	0.9990	0.9990
Naive Bayes	0.9666	0.9666	0.9666	0.9666
Logistic Regression	**0.9995**	**0.9995**	**0.9995**	**0.9995**
KNN	0.9995	0.9995	0.9995	0.9995
Decision Trees	0.9990	0.9990	0.9990	0.9990
Random Forest	**0.9995**	**0.9995**	**0.9995**	**0.9995**
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	**0.9995**	**0.9995**	**0.9995**	**0.9995**

4.2.8 Method 4 - Feature Generation: the perspective based on FastText

The marks of Method 4 through FastText with vocabulary sourced from the pre-trained Common Crawl word vector model are presented in Table 4.15. This variation achieved a F1 score of 100% (as well as for sensitivity, specificity, accuracy, and precision measures), which was the best measurement attained in Method 4, together with the Word2Vec perspective with vocabulary sourced from the pre-trained Google News corpus word vector model. This highly prized measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of the parameters in the default setting.

Furthermore, concerning the results of Table 4.15, it may be seen that of the seven remaining classification algorithms, two achieved 99.95% precision measurements (K-Nearest Neighbors and Multilayer Perceptron), and the other two reached 99.90% (Logistic Regression and XGBoost) and its worst result, in the Naive Bayes classification algorithm, obtained a 99.32% accuracy rate.

Method 4 through FastText with a vocabulary built from the dataset also obtained significant results in the prediction assessments, Table 4.16. It did not reach an accuracy rate of 100%, but it was very close to this. It attained a F1 score of 99.95% in 5 out of the 8 classification algorithms used (four of them with a specificity of 100%), and its worst mark was 99.37% in the Naive Bayes classification algorithm. Its five best results were obtained from the Support Vector Machine, Logistic Regression, K-Nearest Neighbors, XGBoost, and MultLayer Perceptron algorithms. Due to vocabulary construction, this perspective

Table 4.15 Method: Feature Generation through the FastText perspective - Vocabulary Source: pre-trained Common Crawl word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Naive Bayes	0.9932	0.9932	0.9932	0.9932
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9995	0.9995	0.9995	0.9995
Decision Trees	0.9974	0.9974	0.9974	0.9974
Random Forest	0.9984	0.9984	0.9984	0.9984
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	0.9995	0.9995	0.9995	0.9995

spent more time and consumed more processing power than those implemented from the pre-trained Wikipedia (Table 4.17) and Common Crawl (Table 4.15) word vector models.

Table 4.16 Method: Feature Generation through the FastText perspective - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9995**	**0.9995**	**0.9995**	**0.9995**
Naive Bayes	0.9937	0.9938	0.9937	0.9937
Logistic Regression	**0.9995**	**0.9995**	**0.9995**	**0.9995**
KNN	**0.9995**	**0.9995**	**0.9995**	**0.9995**
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	**0.9995**	**0.9995**	**0.9995**	**0.9995**

The last of these variations for the FastText perspective had its vocabulary sourced from the pre-trained Wikipedia word vector model. Its marks are exhibited in Table 4.17. From these results, it can be observed that all algorithms obtained accuracy values equal to or higher than 99.58%. Its best results (F1 Score of 99.90%) were reached using the Logistic Regression and Random Forest algorithms.

Table 4.17 Method 4: Feature Generation through the FastText perspective - Vocabulary Source: pre-trained Wikipedia word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9974	0.9974	0.9974	0.9974
Logistic Regression	**0.9990**	**0.9990**	**0.9990**	**0.9990**
KNN	0.9984	0.9984	0.9984	0.9984
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	**0.9990**	**0.9990**	**0.9990**	**0.9990**
XGBoost	0.9984	0.9984	0.9984	0.9984
MLP	0.9984	0.9984	0.9984	0.9984

4.2.9 Method 4 - Feature Generation: the perspective based on Doc2Vec

Doc2Vec was the last perspective used in Method 4. It was developed in two ways. The first variation obtained its vocabulary from pre-trained text vectors (Table 4.18).

Table 4.18 Method 4: Feature Generation through the Doc2Vec perpective - Vocabulary Source: pre-trained Wikipedia document vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9979	0.9979	0.9979	0.9979
Naive Bayes	0.9223	0.9229	0.9223	0.9212
Logistic Regression	**0.9984**	**0.9984**	**0.9984**	**0.9984**
KNN	0.9718	0.9720	0.9718	0.9719
Decision Trees	0.9536	0.9535	0.9536	0.9535
Random Forest	0.9917	0.9918	0.9917	0.9916
XGBoost	0.9937	0.9938	0.9937	0.9937
MLP	0.9979	0.9979	0.9979	0.9979

The first variation (shown in Table 4.18) achieved better results, with its best mark being a F1 score of 99.84%. This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 10, penalty as l2, and the rest of its parameters in the default setting. This variation had its vectors trained by the authors over a corpus constructed from a dump of Wikipedia articles.

The second variation was built over the dataset (Table 4.19). Its best result was a F1

Score of 99.48%, with a FPR of 0.24%. This measure was obtained through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table 4.19 Method 4: Feature Generation through the Doc2Vec perspective - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9823	0.9823	0.9823	0.9823
Naive Bayes	0.9619	0.9639	0.9609	0.9611
Logistic Regression	0.9896	0.9896	0.9896	0.9895
KNN	0.9948	0.9948	0.9948	0.9948
Decision Trees	0.9906	0.9906	0.9906	0.9906
Random Forest	0.9927	0.9927	0.9927	0.9927
XGBoost	**0.9948**	**0.9948**	**0.9948**	**0.9948**
MLP	0.9917	0.9917	0.9917	0.9916

4.3 Additional Results

This Section provides an additional analysis of the proposed methods for the perspectives that presented the best results (F1 scores and accuracy of 100%) for Dataset 1, described in Section 3.2 of Chapter 3. They are the Chi-Square Perspective of Method 2, the LSA Perspective of Method 3, the Word2Vec Perspective of Method 4, and the FastText Perspective of Method 4.

The performance of these perspectives was evaluated on two additional datasets (Dataset 2 and Dataset 3). They are described in Subsection 4.3.1.

4.3.1 Dataset 2 and Dataset 3

The IWSPA-AP e-mail dataset was initially built for the Anti-Phishing task of the International Workshop on Security and Privacy Analytics, and since then, it has been maintained and improved. According to [112] and [106], this dataset collects recent and historical e-mails from as many sources as possible, and provides different types of phishing and legitimate e-mails, as well as samples of new and classical attacks.

It is also public and has been used in [102], [103], [44], [76], [8], [108] and [30].

It has two subsets: the full-header subset and the no-header subset which, in this study, were used as Dataset 2 and Dataset 3, respectively. Both subsets have samples from the phishing and ham classes. This thesis employed the IWSPA-AP e-mail dataset version 2, whose full-header subset contains 503 phishing e-mails and 4,082 legitimate e-mails, and its no-header subset contains 629 phishing e-mails and 4,642 legitimate e-mails.

Based on the classification strategy proposed in Section 3.9 of Chapter 3, Dataset 2 and Dataset 3 were folded in two sets (training and test sets) for each of their subsets:

- Dataset 2 - for the full-header subset, the training set contained 3,209 samples, and the test set, 1,376 samples. The training set consisted of 2,859 ham e-mails and 350 phishing e-mails, and the test set comprised 1,223 ham e-mails and 153 phishing e-mails.

- Dataset 3 - for the no-header subset, the training set contained 3,209 samples, and the test set, 1,376 samples. The training set consisted of 2,859 ham e-mails and 350 phishing e-mails, and the test set comprised 1,223 ham e-mails and 153 phishing e-mails.

4.3.2 Prediction Results employing Dataset 2

In Table 4.20, the best marks achieved by the proposed method through its perspectives using Dataset 2 are presented.

Table 4.20 Best Marks Achieved by the Proposed Methods though each of their best perspectives for Dataset 2

Perspective	Embeddings	Algorithm	Accuracy	F1 Score
M2 - Chi-Square	-	D. Tree and R. Forest	0.9971	0.9971
M3 - LSA	-	SVM	0.9964	0.9964
Word2Vec	**Dataset 2**	**SVM, Log Reg and MLP**	**0.9985**	**0.9985**
Word2Vec	Google News	Log Reg and MLP	0.9971	0.9971
FastText	Dataset 2	SVM and Log Reg	0.9978	0.9978
FastText	Common Crawl	MLP	0.9971	0.9971

One of the four perspectives of the proposed Method obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.854%, which are, to the best of our knowledge, the best results in phishing detection research with this dataset, higher than the results presented in [30]. It failed to classify only two phishing e-mails, pointing them out as legitimate e-mails. These highly prized measures were achieved through the SVM, Logistic Regression, or MLP algorithms in the Word2Vec perspective with a vocabulary built from Dataset 2. These marks were obtained using the following configurations:

- SVM - the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of the parameters in the default setting.
- Logistic Regression - the inverse of regularization - C as 1000, penalty as l2, and the rest of the parameters in the default setting.
- MLP - with mini-batch size as 40 and the maximum number of iterations as 100 (hyperparameters), and the rest of the parameters in the default setting.

For Dataset 2, the other perspectives/variations obtained the following marks: the FastText perspective of Method 4 using a vocabulary built over the Dataset 2 achieved an accuracy and F1 score of 99.78%; the FastText and Word2Vec perspectives of Method 4 when using pre-trained models based on external corpus, as well as the Chi-Square perspective of Method 2 (using a hundred features), attained an accuracy and F1 score of 99.71%; and the LSA perspective of Method 3 (using twenty-five features) reached an accuracy and F1 score of 99.64%.

4.3.3 Prediction Results employing Dataset 3

In Table 4.21, the best marks achieved by the proposed method through its perspectives using Dataset 3 are presented.

Table 4.21 Best Marks Achieved by the Proposed Methods through each of their best perspectives for Dataset 3

Perspective	Embeddings	Algorithm	Accuracy	F1 Score
M2 - Chi-Square	-	Random Forest	0.9755	0.9751
M3 - LSA	-	XGBoost	0.9773	0.9771
Word2Vec	Dataset 3	XGBoost	0.9814	0.9812
Word2Vec	Google News	MLP	0.9825	0.9828
FastText	Dataset 3	KNN	0.9790	0.9788
FastText	**Common Crawl**	**SVM**	**0.9843**	**0.9843**

One of the four perspectives of the proposed Method obtained accuracy, precision, recall, F1 score, and Specificity rates of 98.43%, which is, to the best of our knowledge, the best result in phishing detection research with this dataset, higher than the results presented in [8]. These highly prized measures were achieved through the SVM algorithm in the FastText perspective based on pre-trained vectors trained on the Common Crawl dataset. These marks were obtained using the following configurations: the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of the parameters in the default setting.

For Dataset 3, the other perspectives/variations obtained the following marks: the Word2Vec perspective of Method 4 achieved an accuracy of 98.25% and 98.14% in variations using a vocabulary built from the Google News dataset and from Dataset 3, respectively; the FastText perspective of Method 4 using a vocabulary built over Dataset 3 reached an accuracy of 97.88%; the Chi-Square perspective of Method 2 (using a hundred features), attained accuracy and F1 score of 97.75%; and the LSA perspective of Method 3 (using twenty-five features) reached accuracy of 97.73% and F1 score of 97.71%.

It is important to note that, although the performance differences between the use of the two databases in the pre-processing stage were minimal, for Dataset 1, as already mentioned in Chapter 3, Method 4 had better results using WordNet, whereas for Dataset 2 and Dataset 3, the best results were found using Stanza.

4.4 Discussion

Based on the proposed approach's prediction results when using Dataset 1, expressed in Section 4.2, a chart was plotted with the best performance marks in each variation of the proposed Method perspectives. Here, we focus on the F1 Score, displayed in Fig 4.4. A chart that expresses this consolidation for accuracy is available in Appendix A.

In Figure 4.4, Method 1 is represented by a gray color, Method 2 perspectives by shades of yellow, Method 3 perspectives by shades of green, and Method 4 perspectives by shades of blue, in which the perspective variations based on vectors obtained from the dataset are always presented before those obtained from external corpora.

It was observed that Methods 2, 3, and 4, for at least one of the perspectives, achieved 100% in accuracy, precision, recall, and F1 score, i.e., the best performance of the proposed approach. For Method 2, this was attained through the Chi-Square perspective, using one hundred features, for Method 3, through the LSA perspective, using twenty-five features, and for Method 4, through the Word2Vec (employing word vectors trained on the Google News corpus) and through the FastText (employing word vectors trained on Common Crawl dataset), both using 300-dimension English word vectors as features. For Method 1, the best result was an F1 Score of 99.74%.

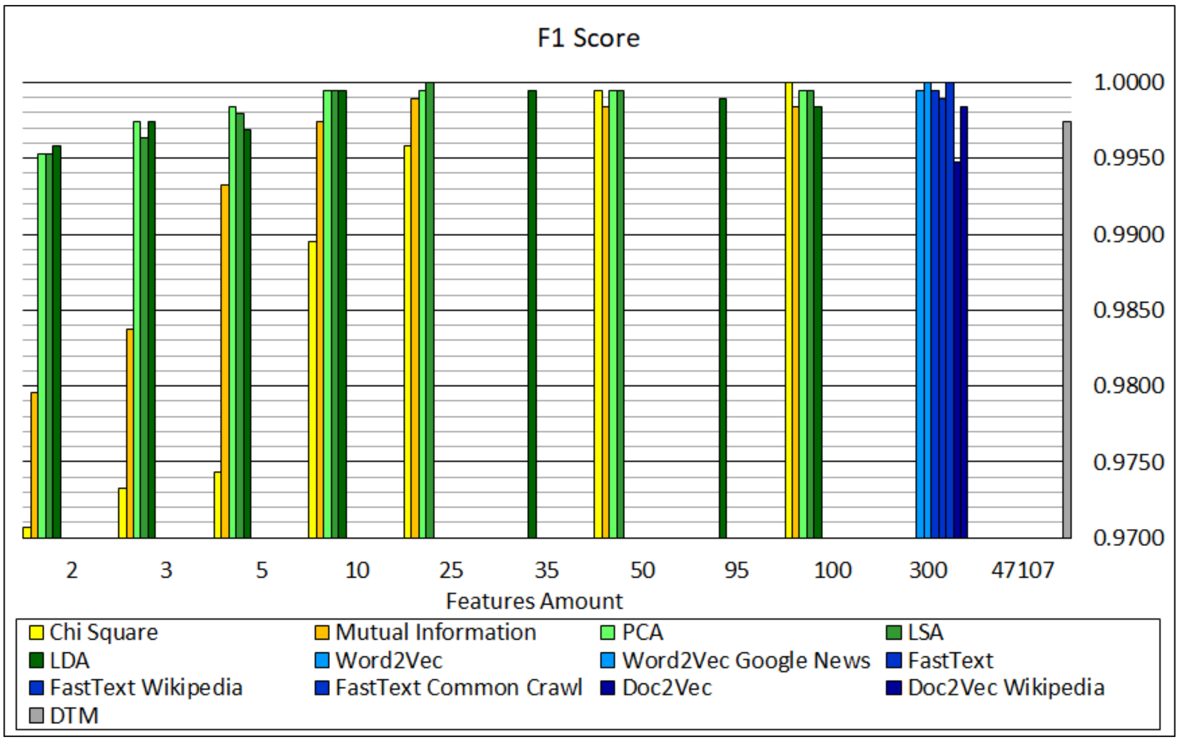


Figure 4.4 F1 Score of the proposed methods in their respective perspectives in each tested feature amount variation.

When analyzing the results of the Method 2 perspectives and their variations, it was observed that the Mutual Information perspective had better results for fewer features than the Chi-Square perspective, which, in turn, was slightly more accurate when using fifty and a hundred features. The worst results were 97.07% (Chi-Square) and 97.96% (Mutual Information) when using two features. The Chi-Square perspective nevertheless attained an F1 score of 99.95%, using fifty features. Mutual information perspective achieved its best mark (F1 score of 99.90%) using twenty-five features.

Method 3 had an outstanding performance overall: with all the variations and all marks above or equal to 99.53%. PCA and LSA had 4 out of 7 feature set variations reaching above or equal 99.95%, LDA presented this performance in two variations. Using two features, LDA reached better results, namely an F1 Score of 99.58%, whereas PCA and LSA had the same result: an F1 accuracy of 99.53%. Subsequently, PCA and LDA had better results than LSA when employing three features, and when five features were used, PCA presented the best marks of these three perspectives. For ten features, they presented the same mark of 99.95% for the F1 Score. PCA maintained these results for twenty-five features, whereas LSA reached a 100% success rate for all marks. LDA repeated its best mark when using thirty-five features. PCA and LSA also attained an F1 score of 99.95% for the variations with fifty and one-hundred features. LDA attained 99.90% and 99.84% for ninety-nine and one-hundred features variations, respectively.

All Method 2 and Method 3 perspectives had their worst marks sets of two, three, and five features were used, respectively.

The representation based on two features was particularly valuable for enabling visualization of the sample scattering and how each perspective established the frontier between the classes. This is portrayed in Fig. 4.5, in a pair plot manner, i.e., creating a grid of axes, where each feature is shared in the y-axis across a single row and in the x-axis across a single column, displaying pairwise relationships in the dataset.

For Method 2, in the Chi-Square perspective, the two selected features were the tokens “be" and “pic", whereas in the Mutual Information, they were the tokens “be" and “http". For Method 3, in the PCA and LSA perspectives, the two features were obtained projecting portions of the original feature set in a 2-dimensional space, whereas in LDA, they were portions of the obtained topics.

From Figures 4.4 and 4.5, and the results presented in Subsection 4.2, it is possible to infer that the arrangement of the samples in the perspectives based on feature extraction techniques better separated the proposed classes when dealing with a reduced feature set, and that the similarities between the samples of the same class became more evident in the perspectives based on the feature selection measures as the amount of features became larger.

Method 4, as well as Method 3, had excellent performance. Except for the Doc2Vec variation based on vectors trained in the dataset of this study (F1 Score of 99.48%), all the variations attained results above or equal to 99.84%. When analyzing only the Method 4 perspectives that obtained the respective text representations averaging all words in each text, Word2Vec, and FastText perspectives, this minimum was 99.90%. For the variations employing vectors constructed from the dataset, the best marks were found for Word2Vec and FastText (99.95%), and the worst for Doc2Vec (99.84%).

As described for Word2Vec and FastText, this vector was obtained by the averages of the word vectors present in each e-mail body, whereas the Doc2Vec used the vector representation generated by the word embedding process. According to [56], because the sense of word order inside the documents is lost, this technique, based on its word embedding averages of all words in a text, did not perform well in recognizing many types of sophisticated linguistic phenomena. In contrast, Kenter et al. 2016, with the model proposed in [51], reported that this technique may produce good results in a wide variety of scenarios, for instance, short text representation issues. Not surprisingly, Method 4

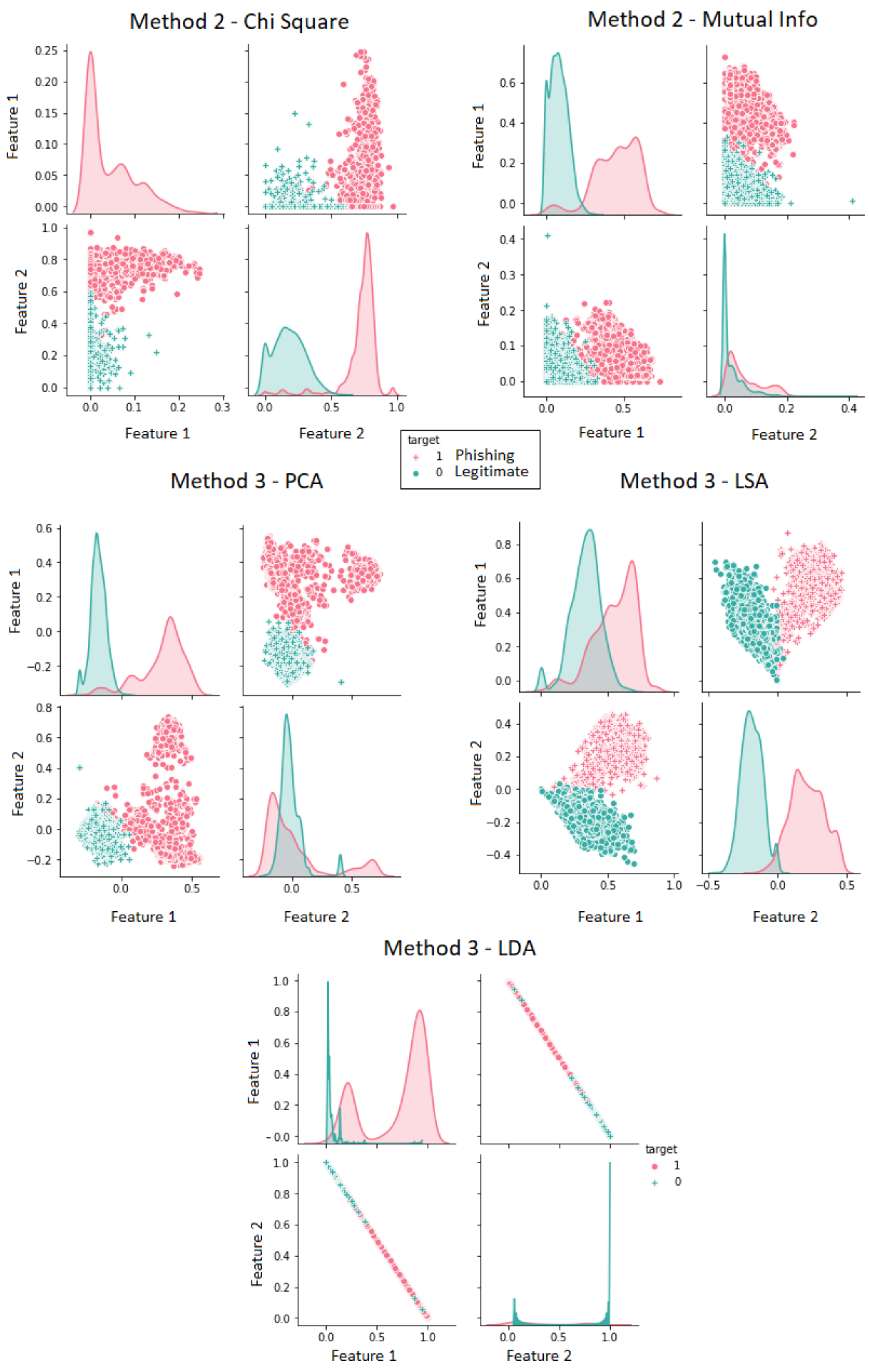


Figure 4.5 Method 2 and Method 3 perspectives using 2 features.

attained marks showing that Word2Vec and FastText perspectives had better results than the Doc2Vec based implementation, even for the approaches based on a vocabulary built over the dataset. However, an implementation that considers pre-trained text embedding on a more substantial basis, as done for Word2Vec and FastText, would be necessary for a more conclusive analysis.

4.5 Performance Comparison

In Table 4.22, this proposal performance, using the marks displayed in Subsection 4.2 for Dataset 1, is confronted with state-of-the-art research aimed at detecting phishing, as described in the baseline study (Section 2.8). They are ordered according to their F1 Scores and the amount of features.

According to these criteria, our best performances were the LSA perspective of Method 3 (with 25 features), followed by the Chi-Square perspective of Method 2 (with 100 features). They were the highest among those compared. In [82], an F1 score of 100% was also attained, but it was done using 200 features, a feature set 8 times greater than ours. As already mentioned, two other variations of our perspectives achieved 100% in F1 Score and accuracy: Word2Vec (employing word vectors trained on the Google News corpus) and the FastText (employing word vectors trained on the Common Crawl dataset), both using three-hundred features.

If this performance comparison were performed considering only the works that used the same datasets, we would then have three comparison contexts, one for each dataset (Dataset 1, Dataset 2, and Dataset 3).

In Table 4.23, this comparison is presented for Dataset 1. It may be observed that our proposed methods obtained the best marks, occupying the first seven positions. For Dataset 1, this thesis proposed methods/perspectives that employ only the bodies of the e-mails.

In Table 4.24, it may be observed that the method proposed here reached an F1 score of 99.85%, which is, to the best of our knowledge, the highest result in phishing detection research using Dataset 2. It is higher than the measurements presented in [30]. It is relevant to mention that the proposed method used version 2.0 of Dataset 2, while in [30], version 1.0 was used. On that occasion, Fang et al. [30] misclassified four samples, one false positive and three false negatives. In our approach for Dataset 2, only two samples were misclassified (two false negatives). For Dataset 2, this thesis proposed methods/perspectives employing the headers and bodies of the e-mails.

Table 4.22 Overall Performance Comparison

Reference	Best Result	Amount of Features	Dataset Sources
M3 - LSA	100%	25	[29] and [75]
M2 - Chi-Square	100%	100	[29] and [75]
Ramanathan and Wechsler [82]	100%	200	cluster
M4 - Word2Vec and FastText - External	100%	300	[29] and [75]
M3 - LDA	99.95%	10 and 35	[29] and [75]
M3 - LSA	99.95%	10, 50 and 100	[29] and [75]
M3 - PCA	99.95%	10, 25, 50 and 100	[29] and [75]
M2 - Chi-Square	99.95%	50	[29] and [75]
M4 - Word2Vec and FastText - dataset	99.95%	300	[29] and [75]
M2 - Mutual Info	99.90%	25	[29] and [75]
M3 - LDA	99.90%	95	[29] and [75]
M4 - FastText - External	99.90%	300	[29] and [75]
Fang et al. [30]	99.85%	256	cluster
M3 - PCA	99.84%	3 and 5	[29] and [75]
M2 - Mutual Info	99.84%	50 and 100	[29] and [75]
M3 - LDA	99.84%	100	[29] and [75]
M4 - Doc2Vec - External	99.84%	300	[29] and [75]
M3 - LDA	99.74%	3	[29] and [75]
M3 - LSA	99.74%	5	[29] and [75]
M2 - Mutual Info	99.74%	10	[29] and [75]
M1	99.74%	47107	[29] and [75]
Akinyelu and Adewumi [2]	99.70%	15	[29] and [75]
M3 - LDA	99.69%	5	[29] and [75]
M3 - LSA	99.63%	3	[29] and [75]
M3 - LDA	99.58%	2 and 10	[29] and [75]
M2 - Chi-Square	99.58%	25	[29] and [75]
L'Huillier et al. [57]	99.58%	1017	[29] and [75]
M3 - LSA and PCA	99.53%	2	[29] and [75]
M4 - Doc2Vec - dataset	99.48%	300	[29] and [75]
Gangavarapu et al. [34]	99.40%	21	[29] and [75]
Daeef et al. [20]	99.40%	48	[29] and [75]
M2 - Mutual Info	99.32%	5	[29] and [75]
Vinayakumar et al. [81]	99.10%	200	cluster
Yasin and Abuhasan [110]	99.10%	16	[29] and [75]
Barathi Ganesh et al. [8]	99%	100	cluster
Gangavarapu and Jaidhar [33]	99%	240	[29] and [75]
M2 - Chi-Square	98.95%	10	[29] and [75]
Halgas et al. [42]	98.63%	5000	[29] and [75]
Chin et al. [17]	98.39%	30	cluster
M2 - Mutual Info	98.38%	3	[29] and [75]
Ramanathan and Wechsler [83]	98.30%	10	cluster
M2 - Mutual Info	97.96%	2	[29] and [75]
Fette et al. [31]	97.64%	10	[29] and [75]
M2 - Chi-Square	97.07% - 97.43%	2, 3 and 5	[29] and [75]
Islam and Abawajy [47]	97.00%	21	[29] and [75]

Table 4.23 Dataset 1 - Performance Comparison

Reference	Best Metric Value	Amount of Features
M3 - LSA	**100%**	**25**
M2 - Chi-Square	**100%**	**100**
M4 - FastText	**100%**	**300**
M4 - Word2Vec	**100%**	**300**
M3 - LDA	99.95%	10
M3 - PCA	99.95%	10
M2 - Mutual Information	99.90%	95
Akinyelu and Adewumi [2]	99.70%	15
L'Huillier et al. [57]	99.58%	1017
Gangavarapu et al. [34]	99.40%	21
Daeef et al. [20]	99.40%	48
Vinayakumar et al. [81]	99.10%	200
Yasin and Abuhasan [110]	99.10%	16
Gangavarapu and Jaidhar [33]	99%	240
Halgas et al. [42]	98.63%	5000
Fette et al. [31]	97.64%	10
Islam and Abawajy [47]	97.00%	21

Table 4.24 Dataset 2 - Performance Comparison

Reference	Best Metric Value	Amount of Features
M4 - Word2Vec	**99.85%**	**300**
Fang et al. [30]	99.84%	830
M4 - FastText	99.78%	300
M2 - Chi-Square	99.71%	100
M3 - LSA	99.64%	25
Barathi Ganesh et al. [8]	99.00%	100
Unnithan et al. [102]	98.00%	40
Vazhayil et al. [103]	95.30%	30
Harikrishnan et al. [44]	94.10%	30
Vinayakumar et al. [108]	92.80%	200

In Table 4.25, it may be observed that the best results of the proposed methods reached an F1 score of 98.43%. It is, to the best of our knowledge, the highest result in phishing detection research using Dataset 3.

It may be observed that the proposed methods obtained the best marks, occupying the first four positions. They are higher than the measurements presented in [8].

For Dataset 3, this thesis proposed methods/perspectives employing only the bodies of the e-mails.

Table 4.25 Dataset 3 - Performance Comparison

Reference	Best Metric Value	Amount of Features
M4 - FastText	**98.43%**	**300**
M4 - Word2Vec	98.28%	300
M2 - Chi-Square	97.75%	100
M3 - LSA	97.71%	25
Barathi Ganesh et al. [8]	97.00%	100
Unnithan et al. [102]	97.00%	40
Harikrishnan et al. [44]	94.60%	30
Vazhayil et al. [103]	93.70%	30
Vinayakumar et al. [108]	92.20%	200

4.6 Final Considerations

It is essential to note that each of the proposed methods in this approach obtained optimal results in their respective categories, as observed when compared with the related works. These marks highlight that the proposed models and their associated techniques made significant contributions to the performance of phishing detection approaches. When the variations of the methods with and without the lemmatization and other resources offered by the dictionary databases (WordNet and Stanza) were compared, it was observed, in all cases, that the results obtained without this step were, in general terms, slightly lower than those that employed it. In addition, two other factors influenced the achievement of these remarkable results. They were the pre-processing steps and the resampling/cross-validation techniques employed in this study. The proposed architecture components, jointly with the dimensionality reduction perspectives, fed the adopted ML algorithms with suitable features. The employment of these algorithms, optimally set, resulted in models that obtained better results than those described in the baseline study.

Regarding the effectiveness of the approach in obtaining the most distinctive features proposed by this thesis, the methods not only presented better results in this feature-based text classification, but also mitigated the posed obstacles related to VSM/DTM representation, providing a dense and low-dimension representation that compressed the data in the proposed texts with reduced noisy information. These problems are related to high dimensionality, sparsity, and contextual information that may be integrated into the proposed representation. For Dataset 1, without the proposed dimensionality reduction and the lemmatization step, for instance, considering all the DTM features in this setting (54,680), the best result would be an F1 score of 99.74%. This is a good result, but at the cost of much greater computational complexity and processing time. Thus, the proposal provides a dense and low-dimension feature set in many size variations that addresses these issues and increases the prediction results of the original feature set. When all the proposed stages of the architecture were implemented before the method perspectives were applied over the DTM weighted by the TF-IDF measure, the sparsity of the **M** matrix was around 99,77%, and after this implementation, the sparsity reached 33% (feature set of 2 attributes) to 50.35% (feature set of 100 attributes) in **S**.

In addition to being among the ML algorithms that achieved an F1 score of 100%, Random Forest and XGBoost frequently achieved the best mark in each variation of Method 2 and Method 3. Random Forest in Method 2, and XGBoost in Method 3. These measures indicate a pattern of great prediction for their use (optimally set) in phishing detection, using features based on the e-mail body texts.

Although the LSA perspective of Method 3 and the Chi-Square perspective of Method 2 also achieved measures of 100% with fewer features using Dataset 1, Method 4, not only for its performance for the three chosen datasets, but also for its paradigm, proved to be a promising line of representation for similarity and classification problems of texts and their respective terms, since it was able to best address and avoid questions such as the curve of dimensionality, high sparsity, and the low amount of contextual information that the classical VSM representation carries. These issues are problematic not only for phishing detection, but also for most natural language processing research. Solutions based on word embedding can still generalize better in unseen events, according to [37]. Once its training occurs on large amounts of unannotated data (as was done with the pre-trained vectors in the textual basis of Google News, Wikipedia, and Common Crawl), words that do not occur in the task training set also have vector representations, and these representations are similar to those of related words that occur in the task training set.

Comparing with other implemented methods as regards the learning mode, Method 4 also presented itself as a better alternative for text representation in phishing detection issues, once it was suitable not only for this learning in batch mode as evidenced in this thesis, but also its approach in online mode appeared to be more promising and flexible. Many alternatives for dealing with the dynamic vocabulary building and adaptive unigram distribution issues have been presented, such as [65], [50], and [58], where good performance and no accuracy drops were observed.

Furthermore, concerning Method 4, another advantage presented was the possibility of further improving the results obtained through the fine-tuning of the word embeddings used, and making the pre-trained models more portable to better fit other uses, avoiding the construction of these word embeddings from scratch. As in [46], a method named Universal Language Model Fine-tuning (ULMFiT) was proposed for transferring learning for any task for NLP, which also introduced some novel technologies to retain previous knowledge and avoid catastrophic forgetting during fine-tuning.

5 Conclusions and Future Work

If I have seen further it is by standing on the shoulders of giants.

—ISAAC NEWTON

Phishing has been a persistent cybercrime problem for all e-mail users, not only in corporate environments where the security measures adopted to deal with this type of incident have been increasingly refined and specialized, but also because this fraudulent practice seems to become even more insightful.

Among the proposed phishing detection techniques, those based on natural language processing and machine learning, in a data-driven approach, demonstrated greater effectiveness and higher accuracy than those based on filtering rules. To perform this category of technique, the required input features for ML classification algorithms, derived from e-mails texts, are represented in VSM. Three main problems are discussed regarding the representation proposed by VSM/DTM: the "Curse of Dimensionality", the sparsity, and the information that must be obtained from the context.

Given this scenario, the primary objective of this thesis was to present a holistic, structured architecture approach for phishing detection, with methods based on NLP and ML, that address the problems related to VSM/DTM representation, thus improving threat identification predictions. Toward this end, the following research question was posed: "Can a phishing detection approach that addresses issues related to vector space models improve the identification of this type of cybercrime?".

By combining established text processing techniques, feature engineering, dimensionality reduction, training, and improved classification algorithms, the proposed approach propounded four Methods.

- The first, used as a baseline result that did not address these VSM issues.

- The second, based on feature selection through statistic measures (Chi-Square and Mutual information).
- The third, based on feature extraction techniques (PCA, LSA, and LDA).
- The fourth, based on word embedding techniques (Word2VEc, FastText, and DOc2Vec).

The remainder of this chapter is organized as follows: The main conclusions are presented in Section 5.1. The contributions of this thesis are outlined in Section 5.2. Finally, future work is suggested in Section 5.3.

5.1 Conclusions

All four proposed methods had excellent marks. Using the main dataset proposed (Dataset 1), its best respective result was an F1 Score of 99.74% with Method 1, whereas the other three methods attained a remarkable measure of 100% for all main utility measures, which is, to the best of our knowledge, the highest result in phishing detection research for an accredited data set based only on the body of e-mails.

The methods/perspectives that obtained 100% in Dataset 1 (perspective Chi-Square of Method 2 - using one-hundred features, perspective LSA of Method 3 - using twenty-five features, perspective Word2Vec, and perspective FastText of Method 4) were evaluated in two different contexts. Using both the bodies and headers of the e-mails of the first additional dataset proposed (Dataset 2), the best mark was a 99.854% F1 Score, which was obtained using the perspective Word2Vec, surpassing the current best result. Also, using just the e-mail bodies, as done for Dataset 1, the evaluation employing Dataset 3 also proved to obtain the best marks for this data collection. All four perspectives outperformed the state-of-the-art results, with an F1 Score of 98.43%, the best mark being through the FastText perspective. Therefore, for both additional datasets, these results were, to the best of our knowledge, the highest in phishing detection research for these accredited datasets.

Method 4, besides having the best performance in all proposed datasets, also demonstrated avoiding "the curve of the dimensionality" and the high sparsity, as well as providing relevant contextual information to the document vector representation through its text embedding.

The results produced by these measurements were not only due to the excellent performance of the classification algorithms, but also to the combined techniques of the feature engineering proposed process such as the text processing procedures (for instance, the lemmatization step), improved learning techniques for resampling and cross-validation, and hyper-parameter configuration estimation.

Thus, the proposed methods, their perspectives, and the entire plan of action demonstrated a relevant performance when distinguishing between ham and phishing e-mails. They also proved to be a substantial contribution to this area of research and to other natural language processing research that need to address or avoid problems related to VSM/DTM representation, once they generate a dense and low-dimension representation for the evaluated texts.

5.2 Contributions

As presented, to properly detect phishing e-mails, we employed natural language processing and machine learning techniques through feature engineering and text representation actions, and strategies to perform division/folding during the classification algorithms training, as well as to promote their performance tests.

Overall, this holistic approach proposed here aims to provide methods that obtain relevant representations for each of the analyzed e-mails and uses them to correctly identify them as legitimate or phishing e-mails. These representations are based on dense vectors of reduced size, and they are used as input features for the chosen classification algorithms which, in turn, train over them and generate models capable of performing such detection. In this sense, from the results detailed in this thesis, the following contributions are highlighted:

- We implemented pre-processing procedures to provide suitable and correct data cleansing for text, together with structured strategies to handle the issues derived from DTM representation.
- We proposed methods to provide effective and low dimensional text representation, employed as optimized features able to typify structures, similarities, and other relevant information for machine learning problems.
- We performed classification tasks to sort phishing against legitimate e-mails, using ML algorithms fed with those features and trained using an established plan of action, from which the models to duly detect phishing e-mails, or other sample classes, are obtained.

- Specifically for the area of phishing detection research, we presented a holistic approach to identify this type of threat, extracting the most suitable settings from each of the stages of our proposed methods.

5.3 Future Works

Future works will focus on approaches that propose the use of enhanced techniques for online learning problems, namely phishing detection, addressing critical issues, such as dynamic vocabulary building and the relationship among the terms in this dynamism, as well as combining word embedding with techniques, such as Latent Dirichlet Allocation, that can provide and structure more information about the text under analysis.

Another research question that is expected to be addressed in the context of phishing detection with regard to word embedding refers to a better fit of pre-trained models for new NLP tasks (database sharing among organizations), as well as actions regarding the maintenance of prior knowledge contained in the representations employed.

We also aim to implement approaches to detect phishing based on deep learning, language models, and transformers, considering that their employment can provide advantages, for instance, using pre-trained models in other languages (different from the initial database language).

A
Methods Results

The cost of a thing is the amount of what I will call life which is required to be exchanged for it, immediately or in the long run.
—HENRY DAVID THOREAU

In this Appendix, a detailed consolidation of the obtained results are described for each of the chosen datasets.

The attained marks for Dataset 1 are presented from Section A.1 to Section A.10. The attained marks for Dataset 2 and Dataset 3 are presented in Section A.11 and Section A.12 respectively.

A.1 Method 1 - Approach based on Document-Term Matrix without feature reduction techniques

This Method did not address the VSM/DTM posed problems. However, it was measured to serve as a baseline/benchmark for the other method proposed in this thesis. Thus, due to these problems, it demanded more processing capacity and time than a method that attend to this questions (due to its higher complexity). Tables A.1 and A.2 express the results achieved when using Stanza and WordNet, respectively, in Method 1. Table A.3 expresses the results attained when neither is used.

Table A.1 Method 1: BoW and DTM - using Stanza - 47,107 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9963	0.9964	0.9963	0.9963
Naive Bayes	0.9562	0.9565	0.9562	0.9563
Logistic Regression	**0.9974**	**0.9974**	**0.9974**	**0.9974**
KNN	0.9948	0.9948	0.9948	0.9948
Decision Trees	0.9917	0.9917	0.9917	0.9916
Random Forest	0.9937	0.9938	0.9937	0.9937
XGBoost	0.9963	0.9964	0.9963	0.9963
MLP	0.9969	0.9969	0.9969	0.9969

Table A.2 Method 1: BoW and DTM - using WordNet - 48,435 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9593	0.9592	0.9593	0.9592
Naive Bayes	0.9635	0.9635	0.9635	0.9635
Logistic Regression	0.9937	0.9937	0.9937	0.9937
KNN	0.8675	0.8986	0.8675	0.8704
Decision Trees	0.9817	0.9818	0.9817	0.9818
Random Forest	0.9901	0.9901	0.9901	0.9901
XGBoost	0.9875	0.9876	0.9875	0.9874
MLP	**0.9943**	**0.9943**	**0.9943**	**0.9943**

Table A.3 Method 1: BoW and DTM without Lemmatization - 63,448 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.8263	0.8614	0.8263	0.8084
Naive Bayes	0.9598	0.9598	0.9598	0.9598
Logistic Regression	0.9937	0.9937	0.9937	0.9937
KNN	0.8565	0.8927	0.8565	0.8598
Decision Trees	0.9750	0.9751	0.9750	0.9750
Random Forest	0.9849	0.9850	0.9849	0.9848
XGBoost	0.9817	0.9821	0.9817	0.9816
MLP	**0.9943**	**0.9943**	**0.9943**	**0.9943**

A.2 Method 2 - the perspective based on Chi-Square measure

For this Method 2 perspective, Chi-Square measure was used as dimensionality reduction approach. Its prediction assessment values are available in Tables from A.4 to A.10. From the original features in DTM columns weighted through the use of TF-IDF, a desired number of features was selected based on this measure. The obtained marks to 2, 3, 5, 10, 25, 50, and 100 features are presented in descending order of their respective best scores. They were obtained using Stanza in their pre-processing step.

The results attained through this perspective with one-hundred features are presented in Table A.4. In this setting, it was obtained accuracy, precision, recall, F1 score, and Specificity rates of 100%, which was, to the best of our knowledge, the highest result in phishing detection researches using just 100 features. It was the best mark using Chi-Square measure in Method 2. This highly prized measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, ten as the number of features to consider when looking for the best split, ten as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

Table A.4 Results of the Chi-Square Perspective of Method 2 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9974	0.9974	0.9974	0.9974
Naive Bayes	0.9703	0.9708	0.9703	0.9700
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9984	0.9984	0.9984	0.9984
Decision Trees	0.9922	0.9922	0.9922	0.9922
Random Forest	**1.0000**	**1.0000**	**1.0000**	**1.0000**
XGBoost	0.9995	0.9995	0.9995	0.9995
MLP	0.9984	0.9984	0.9984	0.9984

The results expressed in Table A.5 refer to those results attained when selecting fifty features through the Chi-Square perspective. It achieved a percentage of 99.95% in accuracy, precision, recall (sensitivity) and F1 Scores measures, using Random Forest[1]

[1]This measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, $\log_2 50$ as the number of features to consider when looking for

and XGBoost[2] algorithms.

Table A.5 Results of the Chi-Square Perspective of Method 2 - feature set with 50 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9963	0.9964	0.9963	0.9963
Naive Bayes	0.9755	0.9760	0.9755	0.9753
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9969	0.9969	0.9969	0.9969
Decision Trees	0.9932	0.9932	0.9932	0.9932
Random Forest	**0.9995**	**0.9995**	**0.9995**	**0.9995**
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9990	0.9990	0.9990	0.9990

This perspective achieved the marks presented in Table A.6, when the classification algorithms were fed with twenty-five features attributes. It attained a percentage of 99.58% in Precision, Recall, and F1 Score measures. These marks were also found using Random Forest algorithm[3].

Table A.6 Results of the Chi-Square Perspective of Method 2 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9927	0.9927	0.9927	0.9927
Naive Bayes	0.9703	0.9702	0.9703	0.9703
Logistic Regression	0.9922	0.9922	0.9922	0.9922
KNN	0.9948	0.9948	0.9948	0.9948
Decision Trees	0.9927	0.9927	0.9927	0.9927
Random Forest	**0.9958**	**0.9958**	**0.9958**	**0.9958**
XGBoost	0.9948	0.9948	0.9948	0.9948
MLP	0.9937	0.9938	0.9937	0.9937

the best split, 2 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[2]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[3]This measure was achieved using Random Forest ML classification algorithm, with the Gini as function to measure the quality of a split, 2 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

Using XGBoost[4] algorithm, the variation with ten features reached an F1 Score of 98.95%. These results are expressed in Table A.7.

Table A.7 Results of the Chi-Square Perspective of Method 2 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9864	0.9865	0.9864	0.9864
Naive Bayes	0.9598	0.9598	0.9598	0.9597
Logistic Regression	0.9875	0.9876	0.9875	0.9874
KNN	0.9875	0.9875	0.9875	0.9875
Decision Trees	0.9875	0.9875	0.9875	0.9875
Random Forest	0.9864	0.9864	0.9864	0.9864
XGBoost	**0.9896**	**0.9896**	**0.9896**	**0.9895**
MLP	0.9854	0.9855	0.9854	0.9853

The tables A.8, A.9, and A.10 express the marks achieved selecting 5, 3 and 2 features respectively. Using five features, the best mark (F1 score of 97.43%) was obtained through the use of the KNN[5] algorithm. Using three features, the best measure (an F1 score of 97.33%) was attained also employing the KNN[6] algorithm. Using two features, the best result (an F1 score of 97.07%) was attained through the use of KNN algorithm[7].

In Table A.11, it is presented the results if the input features did not undergo a lemmatization process. It is done selecting one-hundred features, since this setting of Chi-Square perspective reached the best results for Method 2. It was noted that these results were consistently lower than those in Table A.4.

[4]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[5]This measure was achieved through the use of KNN algorithm, with the number of neighbors as one-hundred, the distance as weight, and the rest of its parameters in the default setting.

[6]This measure was achieved through the use of KNN algorithm, with the number of neighbors as one-hundred, the distance as weight, and the rest of its parameters in the default setting.

[7]This measure was achieved through the use of KNN algorithm, with the number of neighbors as fifty, the distance as weight, and the rest of its parameters in the default setting.

Table A.8 Results of the Chi-Square Perspective of Method 2 - feature set with 5 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9744	0.9750	0.9744	0.9743
Naive Bayes	0.9426	0.9438	0.9426	0.9419
Logistic Regression	0.9724	0.9729	0.9724	0.9721
KNN	**0.9744**	**0.9746**	**0.9744**	**0.9743**
Decision Trees	0.9718	0.9719	0.9718	0.9717
Random Forest	0.9739	0.9743	0.9739	0.9737
XGBoost	0.9739	0.9743	0.9739	0.9738
MLP	0.9718	0.9724	0.9718	0.9716

Table A.9 Results of the Chi-Square Perspective of Method 2 - feature set with 3 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9718	0.9722	0.9718	0.9716
Naive Bayes	0.9619	0.9625	0.9619	0.9616
Logistic Regression	0.9729	0.9734	0.9729	0.9727
KNN	**0.9734**	**0.9736**	**0.9734**	**0.9733**
Decision Trees	0.9692	0.9694	0.9692	0.9691
Random Forest	0.9697	0.9702	0.9697	0.9695
XGBoost	0.9687	0.9691	0.9687	0.9685
MLP	0.9718	0.9725	0.9718	0.9716

Table A.10 Results of the Chi-Square Perspective of Method 2 - feature set with 2 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9687	0.9692	0.9687	0.9685
Naive Bayes	0.9656	0.9667	0.9656	0.9652
Logistic Regression	0.9703	0.9709	0.9703	0.9700
KNN	**0.9708**	**0.9708**	**0.9708**	**0.9707**
Decision Trees	0.9682	0.9684	0.9682	0.9680
Random Forest	0.9687	0.9691	0.9687	0.9685
XGBoost	0.9692	0.9695	0.9692	0.9690
MLP	0.6484	0.5963	0.6484	0.5111

Table A.11 Results of the Chi-Square Perspective of Method 2 - without Lemmatization

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9724	0.9726	0.9724	0.9722
Naive Bayes	0.9426	0.9442	0.9426	0.9418
Logistic Regression	0.9724	0.9726	0.9724	0.9722
KNN	0.9593	0.9613	0.9593	0.9596
Decision Trees	0.9687	0.9687	0.9687	0.9686
Random Forest	**0.9812**	**0.9813**	**0.9812**	**0.9812**
XGBoost	0.9744	0.9749	0.9744	0.9743
MLP	0.9718	0.9719	0.9718	0.9717

A.3 Method 2 - Feature Selection: the perspective based on Mutual Information measure

For this Method 2 perspective, Mutual Information measure was used as dimensionality reduction approach. Its prediction assessment values are available in Tables from A.12 to A.18. From the original features in DTM columns weighted through the use of TF-IDF, a desired number of features was selected based on this measure. The obtained marks to 2, 3, 5, 10, 25, 50, and 100 features are presented in descending order of their respective best scores. They were obtained using Stanza in their pre-processing step.

The results attained through this perspective selecting twenty-five features are presented in Table A.12. In this setting, it was obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.90%. It was the best mark using Mutual Information measure in Method 2. This measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, two as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

The marks reached by Method 2 through the Mutual Information measure selecting fifty features (Table A.13 attained measures of 99.84% for accuracy, precision, recall, and F1 score. They were obtained using Random Forest[8] and XGBoost[9] algorithms.

[8]This measure was achieved through the Random Forest classification algorithm, with the entropy as function to measure the quality of a split, log2 as the number of features to consider when looking for the best split, three as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[9]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6,

Table A.12 Results of the Mutual Information Perspective of Method 2 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9844	0.9844	0.9844	0.9843
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	**0.9990**	**0.9990**	**0.9990**	**0.9990**
XGBoost	0.9984	0.9984	0.9984	0.9984
MLP	0.9969	0.9969	0.9969	0.9969

Table A.13 Results of the Mutual Information Perspective of Method 2 - feature set with 50 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9974	0.9974	0.9974	0.9974
Naive Bayes	0.9838	0.9839	0.9838	0.9838
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	**0.9984**	**0.9984**	**0.9984**	**0.9984**
XGBoost	**0.9984**	**0.9984**	**0.9984**	**0.9984**
MLP	0.9974	0.9974	0.9974	0.9974

Table A.14 Results of the Mutual Information Perspective of Method 2 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9974	0.9974	0.9974	0.9974
Naive Bayes	0.9739	0.9743	0.9739	0.9738
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9969	0.9969	0.9969	0.9969
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	**0.9984**	**0.9984**	**0.9984**	**0.9984**
XGBoost	**0.9984**	**0.9984**	**0.9984**	**0.9984**
MLP	0.9974	0.9974	0.9974	0.9974

For one-hundred features, it was achieved an F1 Score of 99.84% through the use of Random Forest[10] and XGBoost[11] algorithms.

Selecting ten features, the best mark of this perspective was an F1 score of 99.74% through the use of KNN[12] algorithm as exposed in Table A.15.

Table A.15 Results of the Mutual Information Perspective of Method 2 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9943	0.9943	0.9943	0.9943
Naive Bayes	0.9870	0.9871	0.9870	0.9869
Logistic Regression	0.9943	0.9943	0.9943	0.9943
KNN	**0.9974**	**0.9974**	**0.9974**	**0.9974**
Decision Trees	0.9943	0.9943	0.9943	0.9943
Random Forest	0.9963	0.9964	0.9963	0.9963
XGBoost	0.9953	0.9953	0.9953	0.9953
MLP	0.9943	0.9943	0.9943	0.9943

The tables A.16, A.17, and A.18 express the marks achieved through the variations selecting 5, 3 and 2 features respectively. Using five features, the best result (an F1 score of 99.32%) was attained through the use of the Decision Tree[13] algorithm. Using three features, the best result (an F1 score of 98.38%) was achieved through the use of the Decision Tree[14] algorithm. Using two features, the best result (an F1 score of 97.96%) was reached employing the KNN[15] algorithm.

the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[10]This measure was achieved through the Random Forest classification algorithm, with the gini as function to measure the quality of a split, one as the number of features to consider when looking for the best split, two as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[11]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[12]This measure was achieved through the use of KNN algorithm, with the number of neighbors as three, the distance as weight, and the rest of its parameters in the default setting.

[13]This measure was achieved through the use of Decision Tree classification algorithm, with the entropy as function to measure the quality of a split, one-hundred and twenty as the maximum depth, and the rest of its parameters in the default setting.

[14]This measure was achieved through the use of the Decision Tree classification algorithm, with the entropy as function to measure the quality of a split, eleven as the maximum depth, and the rest of its parameters in the default setting.

[15]This measure was achieved through the use of the KNN algorithm, with the number of neighbors as

Table A.16 Results of the Mutual Information Perspective of Method 2 - feature set with 5 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9917	0.9917	0.9917	0.9916
Naive Bayes	0.9849	0.9851	0.9849	0.9848
Logistic Regression	0.9911	0.9912	0.9911	0.9911
KNN	0.9917	0.9917	0.9917	0.9916
Decision Trees	**0.9932**	**0.9932**	**0.9932**	**0.9932**
Random Forest	0.9927	0.9928	0.9927	0.9927
XGBoost	0.9917	0.9917	0.9917	0.9916
MLP	0.9834	0.9834	0.9834	0.9834

Table A.17 Results of the Mutual Information Perspective of Method 2 - feature set with 3 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9760	0.9763	0.9760	0.9759
Naive Bayes	0.9656	0.9655	0.9656	0.9655
Logistic Regression	0.9755	0.9757	0.9755	0.9754
KNN	0.9833	0.9834	0.9833	0.9833
Decision Trees	**0.9838**	**0.9839**	**0.9838**	**0.9838**
Random Forest	0.9823	0.9823	0.9823	0.9822
XGBoost	0.9823	0.9826	0.9823	0.9822
MLP	0.9833	0.9836	0.9833	0.9832

Table A.18 Results of the Mutual Information Perspective of Method 2 - feature set with 2 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9760	0.9764	0.9760	0.9759
Naive Bayes	0.9765	0.9770	0.9765	0.9764
Logistic Regression	0.9739	0.9746	0.9739	0.9737
KNN	**0.9797**	**0.9798**	**0.9797**	**0.9796**
Decision Trees	0.9776	0.9781	0.9776	0.9774
Random Forest	0.9791	0.9792	0.9791	0.9791
XGBoost	0.9770	0.9776	0.9770	0.9769
MLP	0.9760	0.9765	0.9760	0.9758

A.4 Method 3 - Feature Extraction: the perspective based on Principal Component Analysis

For this Method 3 perspective, Principal Component Analysis was used as dimensionality reduction approach. Its prediction assessment values are available in Tables from A.19 to A.25. From the original features in DTM columns weighted through the use of TF-IDF, a desired number of features was extracted based on principal components, projecting the original feature set in a reduced low-dimension space. The obtained marks to 2, 3, 5, 10, 25, 50, and 100 features are presented in descending order of their respective best scores. They were obtained using Stanza in their pre-processing step.

The results attained through this perspective selecting ten features are presented in Table A.19. In this setting, it was obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.95%. It was the best mark using PCA in Method 3. This measure was achieved employing the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.19 Results of the PCA Perspective of Method 3 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9760	0.9764	0.9760	0.9759
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	0.9984	0.9984	0.9984	0.9984
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9969	0.9969	0.9969	0.9969

The mark of 99.95% in accuracy, precision, recall, and F1 score was also the best mark of the variations using the PCA technique to extract 25, 50, and 100 features as displayed in Tables A.20, A.21 and A.22, respectively. Extracting twenty-five features, it was obtained using XGBoost[16] algorithm. Extracting fifty features, it was reached

ten, the distance as weight, and the rest of its parameters in the default setting.

[16]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6,

through the use of the Logistic Regression[17] algorithm. Extracting one-hundred features, it was arrived employing the XGBoost[18] algorithm.

Table A.20 Results of the PCA Perspective of Method 3 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9661	0.9670	0.9661	0.9663
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9990	0.9990	0.9990	0.9990
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9990	0.9990	0.9990	0.9990

Table A.21 Results of the PCA Perspective of Method 3 - feature set with 50 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9677	0.9683	0.9677	0.9678
Logistic Regression	**0.9995**	**0.9995**	**0.9995**	**0.9995**
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	0.9984	0.9984	0.9984	0.9984
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	0.9990	0.9990	0.9990	0.9990

The results presented in Tables A.23, A.24, and A.25 refer to the marks of the variations using the PCA technique to extract 5, 3 and 2 features respectively. Extracting five features, the best value was an F1 score of 99.84%. It was attained employing

the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[17]This measure was achieved through the use of the Logistic Regression algorithm, with the inverse of regularization strength as 1000, the norm used in the penalization as l2, and the rest of its parameters in the default setting.

[18]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.22 Results of the PCA Perspective of Method 3 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9990	0.9990	0.9990	0.9990
Naive Bayes	0.9744	0.9751	0.9744	0.9745
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9990	0.9990	0.9990	0.9990
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9990	0.9990	0.9990	0.9990

Decision Tree[19] algorithm. Extracting three features, the best value was an F1 score of 99.74%. It was attained using KNN[20] algorithm. Extracting two features, the best value was an F1 score of 99.74%. It was attained through the use of the KNN[21] algorithm.

Table A.23 Results of the PCA Perspective of Method 3 - feature set with 5 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9922	0.9922	0.9922	0.9922
Naive Bayes	0.9729	0.9738	0.9729	0.9726
Logistic Regression	0.9885	0.9885	0.9885	0.9885
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	**0.9984**	**0.9984**	**0.9984**	**0.9984**
Random Forest	0.9979	0.9979	0.9979	0.9979
XGBoost	0.9974	0.9974	0.9974	0.9974
MLP	0.9927	0.9927	0.9927	0.9927

[19]This measure was achieved through the use of the Decision Tree classification algorithm, with the entropy as function to measure the quality of a split, fifteen as the maximum depth, and the rest of its parameters in the default setting.

[20]This measure was achieved through the use of the KNN algorithm, with the number of neighbors as ten, the distance as weight, and the rest of its parameters in the default setting.

[21]This measure was achieved through the use of the KNN algorithm, with the number of neighbors as ten, the distance as weight, and the rest of its parameters in the default setting.

Table A.24 Results of the PCA Perspective of Method 3 - feature set with 3 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9828	0.9830	0.9828	0.9827
Naive Bayes	0.9515	0.9514	0.9515	0.9513
Logistic Regression	0.9828	0.9830	0.9828	0.9827
KNN	**0.9974**	**0.9974**	**0.9974**	**0.9974**
Decision Trees	0.9911	0.9911	0.9911	0.9911
Random Forest	0.9937	0.9937	0.9937	0.9937
XGBoost	0.9911	0.9912	0.9911	0.9911
MLP	0.9833	0.9834	0.9833	0.9833

Table A.25 Results of the PCA Perspective of Method 3 - feature set with 2 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9828	0.9830	0.9828	0.9827
Naive Bayes	0.9760	0.9766	0.9760	0.9758
Logistic Regression	0.9828	0.9830	0.9828	0.9827
KNN	**0.9953**	**0.9953**	**0.9953**	**0.9953**
Decision Trees	0.9911	0.9911	0.9911	0.9911
Random Forest	0.9937	0.9937	0.9937	0.9937
XGBoost	0.9901	0.9901	0.9901	0.9901
MLP	0.9880	0.9881	0.9880	0.9880

A.5 Method 3 - Feature Extraction: the perspective based on Latent Semantic Analysis

For this Method 3 perspective, Latent Semantic Analysis was used as dimensionality reduction approach. Its prediction assessment values are available in Tables from A.26 to A.32. From the original features in DTM columns weighted through the use of TF-IDF, a desired number of features was extracted based on singular values, projecting the original feature set in a reduced low-dimension space. The obtained marks to 2, 3, 5, 10, 25, 50, and 100 features are presented in descending order of their respective best scores. They were obtained using Stanza in their pre-processing step.

The results attained through this perspective with twenty-five features are presented in Table A.26. In this setting, it was obtained accuracy, precision, recall, F1 score, and Specificity rates of 100%, which was, to the best of our knowledge, the highest result in phishing detection researches using just 25 features. It was the best mark using LSA measure in Method 3. This highly prized measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.26 Results of the LSA Perspective of Method 3 - feature set with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9734	0.9734	0.9734	0.9733
Logistic Regression	0.9995	0.9995	0.9995	0.9995
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9984	0.9984	0.9984	0.9984
Random Forest	0.9995	0.9995	0.9995	0.9995
XGBoost	**1.0000**	**1.0000**	**1.0000**	**1.0000**
MLP	0.9995	0.9995	0.9995	0.9995

The mark of 99.95% in accuracy, precision, recall and F1 score was the best mark of the variations using LSA technique to extract 10, 50 and 100 features as displayed in Tables A.27, A.28 and A.29, respectively. Extracting ten features, it was attained using the

Random Forest[22] and MLP[23] algorithms. Extracting fifty features, it was reached through the use of the Random Forest[24] and XGBoost[25] algorithms. Extracting one-hundred features, it was achieved employing the XGBoost[26] ML classification algorithm.

Table A.27 Results of the LSA Perspective of Method 3 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9870	0.9870	0.9870	0.9870
Logistic Regression	0.9969	0.9969	0.9969	0.9969
KNN	0.9990	0.9990	0.9990	0.9990
Decision Trees	0.9984	0.9984	0.9984	0.9984
Random Forest	**0.9995**	**0.9995**	**0.9995**	**0.9995**
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	**0.9995**	**0.9995**	**0.9995**	**0.9995**

The results stated in Tables A.30, A.31 and A.32 refer to the marks attained using the PCA technique to extract 5, 3 and 2 features respectively. Extracting five features, the best mark (an F1 score of 99.79%) was obtained using Random Forest[27] and XGBoost[28] algorithms. Extracting three features, the best result (F1 score of 99.63%) was attained

[22]This measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, log2 as the number of features to consider when looking for the best split, 10 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[23]This measure was achieved MLP classification algorithm, with the size of mini-batches as 20 and the maximum number of iterations as 100, and the rest of its parameters in the default setting.

[24]This measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, log2 as the number of features to consider when looking for the best split, 3 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[25]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[26]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[27]This measure was achieved using Random Forest ML classification algorithm, with the entropy as function to measure the quality of a split, 2 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[28]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.28 Results of the LSA Perspective of Method 3 - feature set with 50 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9990	0.9990	0.9990	0.9990
Naive Bayes	0.9760	0.9761	0.9760	0.9760
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9979	0.9979	0.9979	0.9979
Random Forest	**0.9995**	**0.9995**	**0.9995**	**0.9995**
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9990	0.9990	0.9990	0.9990

Table A.29 Results of the LSA Perspective of Method 3 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9990	0.9990	0.9990	0.9990
Naive Bayes	0.9838	0.9839	0.9838	0.9838
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9984	0.9984	0.9984	0.9984
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9990	0.9990	0.9990	0.9990

using KNN[29] and Random Forest[30] algorithms. Extracting two features, the best measure (F1 score of 99.53%) was reached employing the XGBoost[31] algorithm.

Table A.30 Results of the LSA Perspective of Method 3 - feature set with 5 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9943	0.9943	0.9943	0.9942
Naive Bayes	0.9817	0.9821	0.9817	0.9816
Logistic Regression	0.9948	0.9948	0.9948	0.9948
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	**0.9979**	**0.9979**	**0.9979**	**0.9979**
XGBoost	**0.9979**	**0.9979**	**0.9979**	**0.9979**
MLP	0.9948	0.9948	0.9948	0.9948

Table A.31 Results of the LSA Perspective of Method 3 - feature set with 3 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9885	0.9886	0.9885	0.9885
Naive Bayes	0.9776	0.9782	0.9776	0.9774
Logistic Regression	0.9911	0.9912	0.9911	0.9911
KNN	**0.9963**	**0.9963**	**0.9963**	**0.9963**
Decision Trees	0.9937	0.9937	0.9937	0.9937
Random Forest	**0.9963**	**0.9964**	**0.9963**	**0.9963**
XGBoost	0.9953	0.9953	0.9953	0.9953
MLP	0.9901	0.9902	0.9901	0.9901

In Table A.33, it is presented the results if the input features did not undergo a lemmatization process. It is done selecting twenty-five features, since this setting of LSA perspective reached the best results for Method 3. It was noted that these results were consistently lower than those in Table A.26.

[29]This measure was achieved through the use of KNN algorithm, with the number of neighbors as three, the distance as weight, and the rest of its parameters in the default setting.

[30]This measure was achieved using Random Forest ML classification algorithm, with the gini as function to measure the quality of a split, 2 as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[31]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.32 Results of the LSA Perspective of Method 3 - feature set with 2 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9890	0.9892	0.9890	0.9890
Naive Bayes	0.9781	0.9788	0.9781	0.9779
Logistic Regression	0.9911	0.9912	0.9911	0.9911
KNN	0.9943	0.9943	0.9943	0.9943
Decision Trees	0.9906	0.9906	0.9906	0.9906
Random Forest	0.9948	0.9948	0.9948	0.9948
XGBoost	**0.9953**	**0.9953**	**0.9953**	**0.9953**
MLP	0.9917	0.9918	0.9917	0.9916

Table A.33 Results of the LSA Perspective of Method 3 - without Lemmatization

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9729	0.9728	0.9729	0.9728
Naive Bayes	0.9249	0.9287	0.9249	0.9231
Logistic Regression	0.9734	0.9734	0.9734	0.9733
KNN	0.9828	0.9828	0.9828	0.9828
Decision Trees	0.9739	0.9739	0.9739	0.9739
Random Forest	0.9880	0.9881	0.9880	0.9880
XGBoost	**0.9917**	**0.9917**	**0.9917**	**0.9916**
MLP	0.9765	0.9767	0.9765	0.9764

A.6 Method 3 - Feature Extraction: the perspective based on Latent Dirichlet Allocation

For this perspective of Method 3, Latent Dirichlet Allocation (LDA) was used as feature extraction approach. Their prediction assessment values are displayed in Tables from A.34 to A.40. From the topics extracted from the e-mails bodies, representations of the e-mails were obtained in terms of the probability distribution of these topics, specific feature vectors for each of them. The obtained marks to 2, 3, 5, 10, 35, 95, and 100 features are presented in descending order of their respective best scores. They were obtained using WordNet in their pre-processing step.

Table A.34 presents the marks achieved through the LDA perspective extracting ten topics. This approach obtained accuracy, precision, recall, and F1 score measures of 99.95%, FPR of 0%, and a neat specificity of 100%, which was the highest result attained through this perspective. This highly prized measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, the rest of its parameters in the default setting.

Table A.34 Results of the LDA Perspective of Method 3 - feature set with 10 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9969	0.9969	0.9969	0.9969
Naive Bayes	0.9943	0.9943	0.9943	0.9943
Logistic Regression	0.9974	0.9974	0.9974	0.9974
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	0.9974	0.9974	0.9974	0.9974
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9974	0.9974	0.9974	0.9974

The marks reached through this pespective selecting thirty-five topics (Table A.35) were also the best attained by this perspective, as well as those presented in the Table A.34. It obtained the same accuracy, precision, recall and F1 score measures of 99.95% through the XGBoost[32] classification algorithm.

[32]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of

Table A.35 Results of the LDA Perspective of Method 3 - feature set with 35 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9844	0.9847	0.9844	0.9843
Naive Bayes	0.9650	0.9651	0.9650	0.9651
Logistic Regression	0.9896	0.9897	0.9896	0.9895
KNN	0.9974	0.9974	0.9974	0.9974
Decision Trees	0.9984	0.9984	0.9984	0.9984
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	0.9844	0.9846	0.9844	0.9843

The results expressed in Table A.36 refer to the marks obtained when extracting ninety-five input features attributes from the LDA model with 95 topics. It reached a percentage of 99.90% in precision, recall (sensitivity) and F1 score measures, employing the XGBoost [33] algorithm.

Table A.36 Results of the LDA Perspective of Method 3 - feature set with 95 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9958	0.9958	0.9958	0.9958
Naive Bayes	0.9744	0.9746	0.9744	0.9745
Logistic Regression	0.9984	0.9984	0.9984	0.9984
KNN	0.9979	0.9979	0.9979	0.9979
Decision Trees	0.9974	0.9974	0.9974	0.9974
Random Forest	0.9969	0.9969	0.9969	0.9969
XGBoost	**0.9990**	**0.9990**	**0.9990**	**0.9990**
MLP	0.9979	0.9979	0.9979	0.9979

Extracting 100 features from the LDA model with 100 topics, it was achieved an F1 score of 99.84% through the use of the XGBoost[34] and KNN[35] algorithms.

An F1 score of 99.74% was attained when extracting three features from the LDA

instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[33]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[34]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of

Table A.37 Results of the LDA Perspective of Method 3 - feature set with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9948	0.9948	0.9948	0.9948
Naive Bayes	0.9713	0.9718	0.9713	0.9714
Logistic Regression	0.9958	0.9958	0.9958	0.9958
KNN	**0.9984**	**0.9984**	**0.9984**	**0.9984**
Decision Trees	0.9963	0.9964	0.9963	0.9963
Random Forest	0.9963	0.9964	0.9963	0.9963
XGBoost	**0.9984**	**0.9984**	**0.9984**	**0.9984**
MLP	0.9901	0.9901	0.9901	0.9901

model with three topics. It was reached using the Decision Trees[36] and Random Forest[37] algorithms.

Table A.38 Results of the LDA Perspective of Method 3 - feature set with 3 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9922	0.9922	0.9922	0.9922
Naive Bayes	0.9859	0.9859	0.9859	0.9859
Logistic Regression	0.9880	0.9880	0.9880	0.9880
KNN	0.9969	0.9969	0.9969	0.9969
Decision Trees	**0.9974**	**0.9974**	**0.9974**	**0.9974**
Random Forest	**0.9974**	**0.9974**	**0.9974**	**0.9974**
XGBoost	0.9969	0.9969	0.9969	0.9969
MLP	0.9963	0.9964	0.9963	0.9963

For the variation obtained from the LDA model with five topics, it was achieved a 99.69% F1 Score (0.24% of FPR, which was a specificity of almost 99.76%) in

instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[35]This measure was achieved through the K-Nearest Neighbors classification algorithm, with the number of neighbors as 100, the weight function as distance, and the rest of its parameters in the default setting.

[36]This measure was achieved through the Decision Trees classification algorithm, with the entropy as function to measure the quality of a split, four as the depth maximum, and the rest of its parameters in the default setting.

[37]This measure was achieved through the Random Forest classification algorithm, with the entropy as function to measure the quality of a split, $\log_2 2$ as the number of features to consider when looking for the best split, three as the minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

XGBoost[38].

Table A.39 Results of the LDA Perspective of Method 3 - feature set with 5 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9870	0.9870	0.9870	0.9870
Naive Bayes	0.9833	0.9838	0.9833	0.9834
Logistic Regression	0.9870	0.9870	0.9870	0.9869
KNN	0.9958	0.9958	0.9958	0.9958
Decision Trees	0.9963	0.9963	0.9963	0.9963
Random Forest	0.9958	0.9958	0.9958	0.9958
XGBoost	**0.9969**	**0.9969**	**0.9969**	**0.9969**
MLP	0.9906	0.9906	0.9906	0.9906

For the variation obtained from the LDA model with two topics, it had achieved an F1 Score of 99.59% using the Decision Tree algorithm.

Table A.40 Results of the LDA Perspective of Method 3 - feature set with 2 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9812	0.9812	0.9812	0.9812
Naive Bayes	0.9823	0.9823	0.9823	0.9822
Logistic Regression	0.9844	0.9845	0.9844	0.9843
KNN	0.9937	0.9938	0.9937	0.9937
Decision Trees	**0.9958**	**0.9959**	**0.9958**	**0.9958**
Random Forest	0.9937	0.9938	0.9937	0.9937
XGBoost	0.9948	0.9948	0.9948	0.9948
MLP	0.9812	0.9813	0.9812	0.9812

A.7 Method 4 - the perspective based on Word2Vec

For this perspective of Method 4, Word2Vec was used as feature generation approach. Its prediction assessment values are displayed in Tables A.41, A.42 and A.43. From the

[38]This measure was obtained through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

remaining terms resulting after the pre-processing phase, representations of the e-mails were expressed in a fixed vector space of 300 dimensions. They were obtained using WordNet in their pre-processing step.

Table A.41 presents the marks achieved through the Word2Vec perspective with vocabulary sourced from the pre-trained Google News corpus word vector model. This approach attained accuracy, precision, specificity, sensitivity and F1 score measures of 100%, which was, to the best of our knowledge, the highest result in phishing detection researches. This highly prized measure was achieved through the K-Nearest Neighbors classification algorithm, with the number of neighbors as 1, the weight function as uniform, and the rest of its parameters in the default setting. Except for the Naive Bayes algorithm, all other algorithms achieved marks above 99%, of which three reached 99.90% of precision rate (Logistic Regression, XGBoost, and Multilayer Perceptron).

Table A.41 Results of the Word2Vec Perspective of Method 4 - Vocabulary Source: pre-trained Google News corpus word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9650	0.9654	0.9650	0.9648
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Decision Trees	0.9901	0.9901	0.9901	0.9901
Random Forest	0.9979	0.9979	0.9979	0.9979
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	0.9990	0.9990	0.9990	0.9990

Although it had not reached 100% accuracy, Method 4 through Word2Vec with vocabulary built from the Dataset obtained excellent results in prediction assessments, Table A.42. It achieved an F1 score of 99.95% (FPR of 0%, that is 100% of specificity) in four out of eight used classification algorithms and 99.9% in 3 out of 8 of them. The four best results are obtained from: Logistic Regression[39], K-Nearest Neighbors[40], Random

[39]This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 10, penalty as l1, and the rest of its parameters in the default setting.

[40]This measure was achieved through the K-Nearest Neighbors classification algorithm, with the number of neighbors as 3, the weight function as distance, and the rest of its parameters in the default setting.

Forest[41] and MultLayer Perceptron[42] algorithms. Due to vocabulary construction, this approach spent more time and consumed more processing power than the one implemented from the pre-trained Google News corpus word vector model.

Table A.42 Results of the Word2Vec Perspective of Method 4 - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9990	0.9990	0.9990	0.9990
Naive Bayes	0.9666	0.9666	0.9666	0.9666
Logistic Regression	**0.9995**	**0.9995**	**0.9995**	**0.9995**
KNN	**0.9995**	**0.9995**	**0.9995**	**0.9995**
Decision Trees	0.9990	0.9990	0.9990	0.9990
Random Forest	**0.9995**	**0.9995**	**0.9995**	**0.9995**
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	**0.9995**	**0.9995**	**0.9995**	**0.9995**

In Table A.43, it is presented the results if the input features did not undergo a lemmatization process. It is done with vocabulary sourced from the pre-trained Google News corpus word vector model, since this setting of Word2Vec perspective reached the best results for Method 4. Except for Naive Bayes and XGBoost, that had the same results, the measurements obtained without Lemmatization and WordNet-based processing are consistently lower than those in Table A.41.

A.8 Method 4 - the perspective based on FastText

For this perspective of Method 4, FastText was used as feature generation approach. Its prediction assessment values are displayed in Tables 4.17, 4.15 and 4.16. From the remaining terms resulting after the pre-processing phase, representations of the e-mails were expressed in a fixed vector space of 300 dimensions. They were obtained using WordNet in their pre-processing step.

[41]This measure was achieved through the Random Forest classification algorithm, with the entropy as function to measure the quality of a split, $\log_2 2$ as the number of features to consider when looking for the best split, 3 as minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[42]This measure was achieved through the Multilayer Perceptron (MLP) classification algorithm, with the size of mini-batches as 20 and maximum number of iterations as 100 (hyperparameters), and the rest of its parameters in the default setting

Table A.43 Results of the Word2Vec Perspective of Method 4 - Vocabulary Source: built from the Dataset - without Lemmatization

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9979	0.9979	0.9979	0.9979
Naive Bayes	0.9650	0.9655	0.9651	0.9648
Logistic Regression	0.9984	0.9984	0.9984	0.9984
KNN	**0.9990**	**0.9990**	**0.9990**	**0.9990**
Decision Trees	0.9885	0.9885	0.9885	0.9885
Random Forest	**0.9990**	**0.9990**	**0.9990**	**0.9990**
XGBoost	**0.9990**	**0.9990**	**0.9990**	**0.9990**
MLP	0.9974	0.9974	0.9974	0.9974

For Method 4 through the FastText perspective, three variations were used: two of them based on a vocabulary from pre-trained word vector models, and a third, whose vocabulary was built directly from the Dataset 1 and its relations.

The first variation had its vocabulary sourced from the pre-trained Wikipedia word vector model. These marks are exhibited in the table A.44. All algorithms obtained accuracy values equal to or higher than 99.58%. The best results of this variation were attained employing Logistic Regression[43] and Random Forest[44] algorithms.

The second variation had its vocabulary obtained from pre-trained Common Crawl word vector model. Its marks are presented in the table A.45. Through this variation, it was achieved an F1 score of 100% (as well as sensitivity, specificity, accuracy and precision measures), which was the best measurement attained in our entire approach, together with: the Method 4 through Word2Vec with vocabulary sourced from pre-trained Google News corpus word vector model, the LSA perspective of Method 3, using twenty-five features, and the Chi-Square perspective of Method 2, using one-hundred features. This highly prized measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of its parameters in the default setting.

The third variation had its vocabulary built from the Dataset 1. This approach also

[43]This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 1000, penalty as l1, and the rest of its parameters in the default setting.

[44]This measure was achieved through the Random Forest classification algorithm, with the entropy as function to measure the quality of a split, sqrt as the number of features to consider when looking for the best split, 3 as minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

Table A.44 Results of the FastText Perspective of Method 4 - Vocabulary Source: pre-trained Wikipedia word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9984	0.9984	0.9984	0.9984
Naive Bayes	0.9974	0.9974	0.9974	0.9974
Logistic Regression	**0.9990**	**0.9990**	**0.9990**	**0.9990**
KNN	0.9984	0.9984	0.9984	0.9984
Decision Trees	0.9958	0.9958	0.9958	0.9958
Random Forest	**0.9990**	**0.9990**	**0.9990**	**0.9990**
XGBoost	0.9984	0.9984	0.9984	0.9984
MLP	0.9984	0.9984	0.9984	0.9984

Table A.45 Results of the FastText Perspective of Method 4 - Vocabulary Source: pre-trained Common Crawl word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Naive Bayes	0.9932	0.9932	0.9932	0.9932
Logistic Regression	0.9990	0.9990	0.9990	0.9990
KNN	0.9995	0.9995	0.9995	0.9995
Decision Trees	0.9974	0.9974	0.9974	0.9974
Random Forest	0.9984	0.9984	0.9984	0.9984
XGBoost	0.9990	0.9990	0.9990	0.9990
MLP	0.9995	0.9995	0.9995	0.9995

obtained significant results in prediction assessments, expressed in Table A.46. It did not reach an accuracy rate of 100%, but it was very close to this performance. It attained an F1 score of 99.95% in 5 out of 8 classification algorithms (4 of them with a specificity of 100%), and its worst mark was 99.37%, in the Naive Bayes classification algorithm. Its five best results were obtained from: Support Vector Machine[45], Logistic Regression[46], K-Nearest Neighbors[47], XGBoost[48] and MultLayer Perceptron[49] algorithms. Due to vocabulary construction, this approach spent more time and consumed more processing power than those implemented from pre-trained Wikipedia and Common Crawl word vector models.

Table A.46 Results of the FastText Perspective of Method 4 - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9995**	**0.9995**	**0.9995**	**0.9995**
Naive Bayes	0.9937	0.9938	0.9937	0.9937
Logistic Regression	**0.9995**	**0.9995**	**0.9995**	**0.9995**
KNN	**0.9995**	**0.9995**	**0.9995**	**0.9995**
Decision Trees	0.9969	0.9969	0.9969	0.9969
Random Forest	0.9990	0.9990	0.9990	0.9990
XGBoost	**0.9995**	**0.9995**	**0.9995**	**0.9995**
MLP	**0.9995**	**0.9995**	**0.9995**	**0.9995**

[45]This measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 100, the kernel type as rbf, the kernel coefficient - gamma as 0.001, and the rest of its parameters in the default setting.

[46]This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 10, penalty as l2, and the rest of its parameters in the default setting.

[47]This measure was achieved through the K-Nearest Neighbors classification algorithm, with the number of neighbors as 1, the weight function as uniform, and the rest of its parameters in the default setting.

[48]This measure was obtained through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[49]This measure was achieved through the Multilayer Perceptron (MLP) classification algorithm, with the size of mini-batches as 10 and maximum number of iterations as 50 (hyperparameters), and the rest of its parameters in the default setting

A.9 Method 4 - the perspective based on Doc2Vec

For this perspective of Method 4, Doc2Vec was used as feature generation approach. Its prediction assessment values are displayed in Tables A.48 and A.47. From the remaining terms resulting after the pre-processing phase, representations of the e-mails were expressed in a fixed vector space of 300 dimensions. They were obtained using WordNet in their pre-processing step.

Table A.41 presents the marks achieved through the Word2Vec perspective with vocabulary sourced from pre-trained texts vectors, trained by the authors over a corpus constructed from a dump of Wikipedia articles. This approach attained an F1 score of 99.84% in its best mark. This measure was achieved through the Logistic Regression classification algorithm, with the inverse of regularization - C as 10, penalty as l2, and the rest of its parameters in the default setting.

Table A.47 Results of the Doc2Vec Perspective of Method 4 - Vocabulary Source: pre-trained Wikipedia document vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9979	0.9979	0.9979	0.9979
Naive Bayes	0.9223	0.9229	0.9223	0.9212
Logistic Regression	**0.9984**	**0.9984**	**0.9984**	**0.9984**
KNN	0.9718	0.9720	0.9718	0.9719
Decision Trees	0.9536	0.9535	0.9536	0.9535
Random Forest	0.9917	0.9918	0.9917	0.9916
XGBoost	0.9937	0.9938	0.9937	0.9937
MLP	0.9979	0.9979	0.9979	0.9979

The second variation of this perspective had its vocabulary built from the Dataset 1. The results of this variation are expressed in Table A.48. Its best result was an F1 Score of 99.48%, with an FPR of 0.24%. This measure was obtained through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

Table A.48 Results of the Doc2Vec Perspective of Method 4 - Vocabulary Source: built from the Dataset

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9823	0.9823	0.9823	0.9823
Naive Bayes	0.9619	0.9639	0.9609	0.9611
Logistic Regression	0.9896	0.9896	0.9896	0.9895
KNN	**0.9948**	**0.9948**	**0.9948**	**0.9948**
Decision Trees	0.9906	0.9906	0.9906	0.9906
Random Forest	0.9927	0.9927	0.9927	0.9927
XGBoost	**0.9948**	**0.9948**	**0.9948**	**0.9948**
MLP	0.9917	0.9917	0.9917	0.9916

A.10 consolidation

As stated in Chapter 4, based on the proposed approach's prediction results when using Dataset 1, expressed in Section 4.2 and in Appendix A, the charts presented in Fig 4.4 and Fig A.1 were plotted to express the best perfomance marks, respectivelly the F1 Score and the accuracy, in each variation of the proposed Method perspectives.

In these figures, Method 1 is represented by a gray color, Method 2 perspectives by shades of yellow, Method 3 perspectives by shades of green, and Method 4 perspectives by shades of blue, in which the perspective variations based on vectors obtained from the dataset are always presented before those obtained from external corpora.

It was observed that Methods 2, 3, and 4, for at least one of the perspectives, achieved 100% in accuracy, precision, recall, and F1 score, i.e., the best performance of the proposed approach. For Method 2, this was attained through the Chi-Square perspective, using one hundred features, for Method 3, through the LSA perspective, using twenty-five features, and for Method 4, through the Word2Vec (employing word vectors trained on the Google News corpus) and through the FastText (employing word vectors trained on Common Crawl dataset), both using 300-dimension English word vectors as features. For Method 1, the best result was an F1 Score of 99.74%.

A.11 Obtained Results for Dataset 2

As stated in Chapter 4, we propounded an additional analysis of the proposed methods for the perspectives that presented the best results for Dataset 1. It was performed using

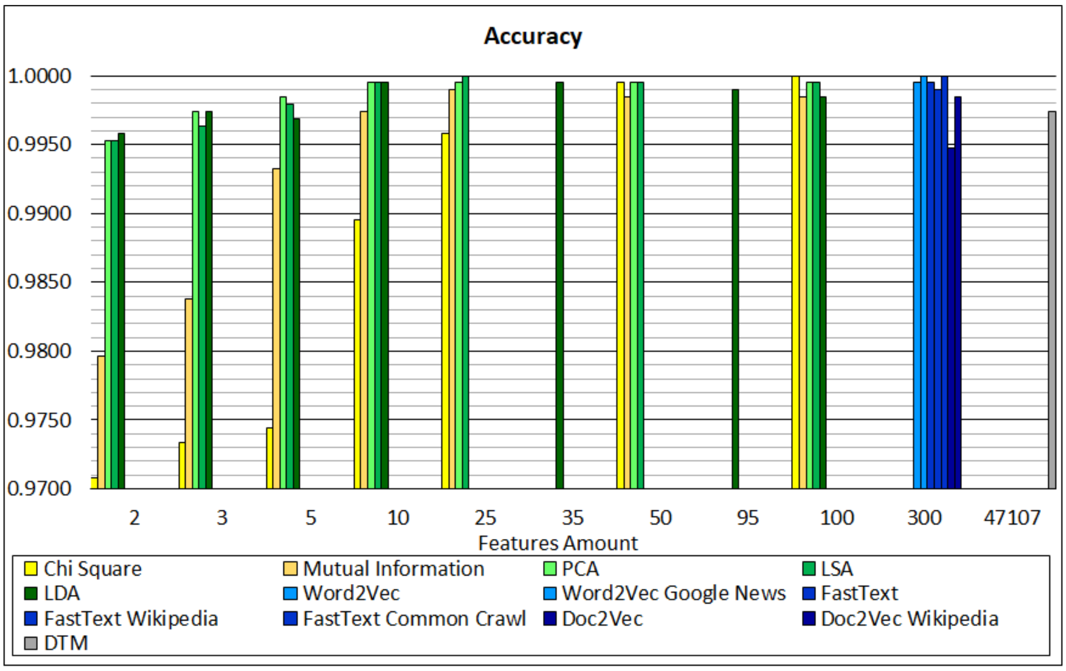


Figure A.1 Accuracy of the proposed methods in their respective perspectives in each tested feature amount variations - Dataset 1.

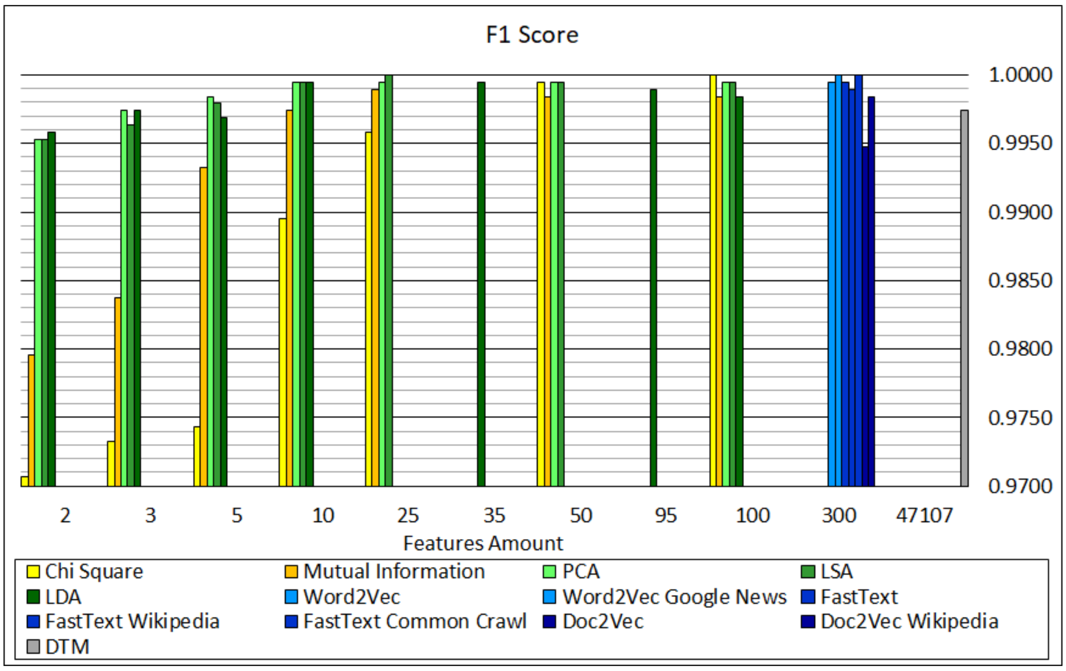


Figure A.2 Accuracy of the proposed methods in their respective perspectives in each tested feature amount variations - Dataset 1.

Dataset 2 and Dataset 3. Section A.11 presents the results for Dataset 2.

The perspectives/methods used in this analysis were Chi-Square Perspective of Method 2, the LSA Perspective of Method 3, the Word2Vec Perspective of Method 4, and the FastText Perspective of Method 4, since they attained F1 scores and accuracy of 100%.

The Chi-square perspective of Method 2 attained the results expressed in Table A.49. It best mark was an F1 score of 99.71%, employing the Decision Tree[50] and Random Forest[51] algorithms.

Table A.49 Results for Dataset 2 - Employing Method 2 through the Chi-Square perspective with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9964	0.9964	0.9964	0.9964
Naive Bayes	0.9724	0.9775	0.9724	0.9736
Logistic Regression	0.9956	0.9956	0.9956	0.9956
KNN	0.9927	0.9927	0.9927	0.9927
Decision Trees	**0.9971**	**0.9971**	**0.9971**	**0.9971**
Random Forest	**0.9971**	**0.9971**	**0.9971**	**0.9971**
XGBoost	0.9964	0.9964	0.9964	0.9964
MLP	0.9956	0.9956	0.9956	0.9956

The LSA perspective of Method 3 reached the results expressed in Table A.50. It best mark was an F1 score of 99.64%, employing the SVM[52] algorithm.

The Word2Vec perspective of Method 4 (with its vocabulary built from the Dataset 2) reached the results expressed in Table A.58. This approach obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.854%, which are, to the best of our knowledge, the best results in phishing detection research employing Dataset 2. These highly prized

[50]This measure was achieved through the Decision Trees classification algorithm, with the gini as function to measure the quality of a split, four as the depth maximum, and the rest of its parameters in the default setting.

[51]This measure was achieved through the Random Forest classification algorithm, with the gini as function to measure the quality of a split, auto as the number of features to consider when looking for the best split, 2 as minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[52]This measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 1,000, the kernel type as linear, the kernel coefficient - gamma as 0.001, and the rest of its parameters in the default setting.

Table A.50 Results for Dataset 2 - Employing Method 3 through the LSA perspective with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9971**	**0.9964**	**0.9964**	**0.9964**
Naive Bayes	0.9782	0.9788	0.9782	0.9784
Logistic Regression	0.9956	0.9956	0.9956	0.9956
KNN	0.9956	0.9956	0.9956	0.9956
Decision Trees	0.9906	0.9906	0.9906	0.9906
Random Forest	0.9956	0.9957	0.9956	0.9956
XGBoost	0.9956	0.9957	0.9956	0.9956
MLP	0.9956	0.9957	0.9956	0.9956

measures were achieved through the SVM[53], Logistic Regression[54], or MLP[55] algorithms.

Table A.51 Results for Dataset 2 - Employing Method 4 through the Word2Vec perspective with word vector built over Dataset 2

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9985**	**0.9985**	**0.9985**	**0.9985**
Naive Bayes	0.9433	0.9533	0.9433	0.9465
Logistic Regression	0.9985	0.9985	0.9985	0.9985
KNN	0.9964	0.9964	0.9964	0.9964
Decision Trees	0.9906	0.9905	0.9906	0.9905
Random Forest	0.9956	0.9956	0.9956	0.9956
XGBoost	0.9949	0.9949	0.9949	0.9949
MLP	0.9971	0.9971	0.9971	0.9971

The Word2Vec perspective of Method 4 (with its vocabulary sourced from the pre-trained Google News corpus word vector model) reached the results expressed in Table A.52. This approach obtained F1 score of 99.71%. This measure were achieved through

[53]This highly prized measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of its parameters in the default setting.

[54]This highly prized measure was achieved through the Logistic Regression algorithm, with the inverse of regularization - C as 1000, penalty as l2, and the rest of the parameters in the default setting.

[55]This highly prized measure was achieved through the MLP algorithm, with mini-batch size as 40 and the maximum number of iterations as 100 (hyperparameters), and the rest of the parameters in the default setting.

the Logistic Regression[56] and MLP[57] algorithms.

Table A.52 Results for Dataset 2 - Employing Method 4 through the Word2Vec perspective with vocabulary Source from pre-trained Google News corpus word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9964	0.9964	0.9964	0.9964
Naive Bayes	0.9157	0.9427	0.9157	0.9237
Logistic Regression	**0.9971**	**0.9971**	**0.9971**	**0.9971**
KNN	0.9935	0.9934	0.9935	0.9934
Decision Trees	0.9680	0.9675	0.9680	0.9677
Random Forest	0.9876	0.9878	0.9876	0.9873
XGBoost	0.9913	0.9914	0.9913	0.9911
MLP	**0.9971**	**0.9971**	**0.9971**	**0.9971**

The FastText perspective of Method 4 (with its vocabulary built from the Dataset 2) reached the results expressed in Table A.53. This approach obtained accuracy, precision, recall, F1 score, and Specificity rates of 99.78%, which are, to the best of our knowledge, the best results in phishing detection research employing Dataset 2. This measure were achieved through the SVM[58] and Logistic Regression[59] algorithms.

The FastText perspective of Method 4 (with its vocabulary sourced from pre-trained Common Crawl word vector model) reached the results expressed in Table A.54. This approach obtained F1 score of 99.71%. This measure were achieved through the MLP[60] algorithm.

[56]This highly prized measure was achieved through the Logistic Regression algorithm, with the inverse of regularization - C as 1000, penalty as l2, and the rest of the parameters in the default setting.

[57]This highly prized measure was achieved through the MLP algorithm, with mini-batch size as 10 and the maximum number of iterations as 100 (hyperparameters), and the rest of the parameters in the default setting.

[58]This measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 1,000, the kernel type as rbf, and the rest of its parameters in the default setting.

[59]This measure was achieved through the Logistic Regression algorithm, with the inverse of regularization - C as 100, penalty as l2, and the rest of the parameters in the default setting.

[60]This highly prized measure was achieved through the MLP algorithm, with mini-batch size as 10 and the maximum number of iterations as 100 (hyperparameters), and the rest of the parameters in the default setting.

Table A.53 Results for Dataset 2 - Employing Method 4 through the FastText perspective with word vector built over Dataset 2

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9978**	**0.9978**	**0.9978**	**0.9978**
Naive Bayes	0.9157	0.9441	0.9157	0.9239
Logistic Regression	**0.9978**	**0.9978**	**0.9978**	**0.9978**
KNN	0.9956	0.9956	0.9956	0.9956
Decision Trees	0.9891	0.9891	0.9891	0.9891
Random Forest	0.9949	0.9949	0.9949	0.9949
XGBoost	0.9949	0.9949	0.9949	0.9949
MLP	0.9956	0.9956	0.9956	0.9956

Table A.54 Results for Dataset 2 - Employing Method 4 through the FastText perspective with word vector sourced from Common Crawl word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9949	0.9949	0.9949	0.9949
Naive Bayes	0.9542	0.9580	0.9542	0.9556
Logistic Regression	0.9964	0.9963	0.9964	0.9964
KNN	0.9920	0.9921	0.9920	0.9919
Decision Trees	0.9767	0.9764	0.9767	0.9765
Random Forest	0.9869	0.9870	0.9869	0.9866
XGBoost	0.9920	0.9921	0.9920	0.9919
MLP	**0.9971**	**0.9971**	**0.9971**	**0.9971**

A.12 Obtained Results for Dataset 3

As stated in Chapter 4, we propounded an additional analysis of the proposed methods for the perspectives that presented the best results for Dataset 1. It was performed using Dataset 2 and Dataset 3. Section A.12 presents the results for Dataset 3.

The perspectives/methods used in this analysis were Chi-Square Perspective of Method 2, the LSA Perspective of Method 3, the Word2Vec Perspective of Method 4, and the FastText Perspective of Method 4, since they attained F1 scores and accuracy of 100%.

The Chi-square perspective of Method 2 attained the results expressed in Table A.55. It best mark was an F1 score of 97.51%, employing the Random Forest[61] algorithm.

Table A.55 Results for Dataset 3 - Employing Method 2 through the Chi-Square perspective with 100 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9744	0.9742	0.9744	0.9734
Naive Bayes	0.9540	0.9571	0.9540	0.9552
Logistic Regression	0.9726	0.9720	0.9726	0.9720
KNN	0.9586	0.9586	0.9586	0.9554
Decision Trees	0.9610	0.9602	0.9610	0.9605
Random Forest	**0.9755**	**0.9750**	**0.9755**	**0.9751**
XGBoost	0.9651	0.9641	0.9651	0.9636
MLP	0.9720	0.9714	0.9720	0.9715

The LSA perspective of Method 3 reached the results expressed in Table A.56. It best mark was an F1 score of 97.71%, employing the XGBoost[62] algorithm.

The Word2Vec perspective of Method 4 (with its vocabulary built from the Dataset 3) reached the results expressed in Table A.57. This approach obtained an F1 score of 98.12%. This measure was achieved through the XGBoost[63] algorithm.

[61]This measure was achieved through the Random Forest classification algorithm, with the gini as function to measure the quality of a split, log2 as the number of features to consider when looking for the best split, 10 as minimum number of samples required to split an internal node, and the rest of its parameters in the default setting.

[62]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[63]This measure was achieved through the XGBoost classification algorithm, with the subsample as 0.6, the minimum split loss reduction - gamma as 0.5, the maximum depth of a tree as 4, the minimum sum of

Table A.56 Results for Dataset 3 - Employing Method 3 through the LSA perspective with 25 features

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9720	0.9714	0.9720	0.9712
Naive Bayes	0.7065	0.9156	0.7065	0.7597
Logistic Regression	0.9674	0.9665	0.9674	0.9662
KNN	0.9761	0.9761	0.9761	0.9761
Decision Trees	0.9645	0.9634	0.9645	0.9636
Random Forest	0.9773	0.9769	0.9773	0.9769
XGBoost	**0.9773**	**0.9770**	**0.9773**	**0.9771**
MLP	0.9726	0.9720	0.9726	0.9718

Table A.57 Results for Dataset 3 - Employing Method 4 through the Word2Vec perspective with word vector built on Dataset 3

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9790	0.9788	0.9790	0.9789
Naive Bayes	0.9016	0.9378	0.9016	0.9120
Logistic Regression	0.9703	0.9699	0.9703	0.9701
KNN	0.9668	0.9676	0.9668	0.9671
Decision Trees	0.9616	0.9627	0.9616	0.9621
Random Forest	0.9785	0.9781	0.9785	0.9781
XGBoost	**0.9814**	**0.9811**	**0.9814**	**0.9812**
MLP	0.9738	0.9732	0.9738	0.9733

The Word2Vec perspective of Method 4 (with its vocabulary sourced from pre-trained Common Crawl word vector model) reached the results expressed in Table A.54. This approach obtained F1 scoreof 98.28%. This measure was achieved through the MLP[64] algorithm.

Table A.58 Results for Dataset 3 - Employing Method 4 through the Word2Vec perspective with its vocabulary sourced from pre-trained Google News word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9796	0.9799	0.9796	0.9797
Naive Bayes	0.8008	0.9107	0.8008	0.8330
Logistic Regression	0.9808	0.9808	0.9808	0.9808
KNN	0.9808	0.9808	0.9808	0.9808
Decision Trees	0.9418	0.9418	0.9418	0.9418
Random Forest	0.9738	0.9734	0.9738	0.9729
XGBoost	0.9808	0.9805	0.9808	0.9804
MLP	**0.9825**	**0.9832**	**0.9825**	**0.9828**

The FastText perspective of Method 4 (with its vocabulary built from the Dataset 3) reached the results expressed in Table A.59. This approach obtained F1 score of 97.88%. This measure was achieved through the KNN[65] algorithm.

The FastText perspective of Method 4 (with its vocabulary sourced from pre-trained Common Crawl word vector model) reached the results expressed in Table A.60. This approach obtained accuracy, precision, recall, F1 score, and Specificity rates of 98.43%, which are, to the best of our knowledge, the best results in phishing detection research employing Dataset 3. These highly prized measures were achieved through the the SVM[66] algorithm.

instance weight needed in a child as 1, and the rest of its parameters in the default setting.

[64]This measure was achieved through the MLP algorithm, with mini-batch size as 10 and the maximum number of iterations as 100 (hyperparameters), and the rest of the parameters in the default setting.

[65]This measure was achieved through the use of KNN algorithm, with the number of neighbors as one, the weights as uniform, and the rest of its parameters in the default setting.

[66]This measure was achieved through the Support Vector Machine algorithm, with the penalty parameter of the term - C as 100, the kernel type as linear, and the rest of its parameters in the default setting.

Table A.59 Results for Dataset 3 - Employing Method 4 through the FastText perspective with word vector built over Dataset 3

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	0.9779	0.9776	0.9779	0.9777
Naive Bayes	0.6605	0.8955	0.6605	0.7219
Logistic Regression	0.9773	0.9770	0.9773	0.9771
KNN	**0.9790**	**0.9788**	**0.9790**	**0.9788**
Decision Trees	0.9441	0.9454	0.9441	0.9447
Random Forest	0.9668	0.9665	0.9668	0.9650
XGBoost	0.9755	0.9751	0.9655	0.9749
MLP	0.9750	0.9757	0.9750	0.9753

Table A.60 Results for Dataset 3 - Employing Method 4 through the FastText perspective with its vocabulary sourced from pre-trained Common Crawl word vector model

Algorithm	Accuracy	Precision	Recall	F1 score
SVM	**0.9843**	**0.9843**	**0.9843**	**0.9843**
Naive Bayes	0.6045	0.8864	0.6045	0.6744
Logistic Regression	**0.9843**	**0.9842**	**0.9843**	**0.9842**
KNN	0.9732	0.9732	0.9732	0.9732
Decision Trees	0.9458	0.9465	0.9458	0.9461
Random Forest	0.9715	0.9710	0.9715	0.9703
XGBoost	0.9819	0.9817	0.9819	0.9817
MLP	0.9808	0.9819	0.9808	0.9811

Bibliography

[1] Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-phishing Working Groups 2Nd Annual eCrime Researchers Summit*, eCrime '07, pages 60–69, New York, NY, USA. ACM.

[2] Akinyelu, A. and Adewumi, A. (2014). Classification of phishing email using random forest machine learning technique. *Journal of Applied Mathematics*, **2014**.

[3] Aleroud, A. and Zhou, L. (2017). Phishing environments, techniques, and countermeasures. *Comput. Secur.*, **68**(C), 160–196.

[4] Almomani, A., Gupta, B. B., Atawneh, S., Meulenberg, A., and Almomani, E. (2013). A survey of phishing email filtering techniques. *IEEE Communications Surveys & Tutorials*, **15**, 2070–2090.

[5] Alpaydin, E. (2014). *Introduction to Machine Learning*. The MIT Press.

[6] (APWG), A.-P. W. G. (2020). Phishing activity trends reports: 1st quarter 2020 plus covid-19 coverage. Technical report.

[7] Baki, S., Das, A., Elaassal, A., Goyal, P., Marchette, D., Moraes, L. F. T. D., Rebeiro, C., and Verma, R., editors (2018). *Proceedings of the 1st Anti-Phishing Shared Task at 4th ACM IWSPA (IWSPA-AP)*, number 2124 in CEUR Workshop Proceedings, Aachen.

[8] Barathi Ganesh, H., Vinayakumar, R., Anand Kumar, M., and Soman, K. (2018). Distributed representation using target classes: Bag of tricks for security and privacy analytics amrita-nlpiwspa-2018. In *CEUR Workshop Proceedings, CEUR-WS*, volume 2124, pages 10–15.

[9] Basnet, R. B., Sung, A. H., and Liu, Q. (2014). Learning to detect phishing urls. In *International Journal of Research in Engineering and Technology*, volume 03.

[10] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[11] Blei, D. M. (2012). Probabilistic topic models. *Commun. ACM*, **55**(4), 77–84.

[12] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, **3**, 993–1022.

[13] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *CoRR*, **abs/1607.04606**.

[14] Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., and Blei, D. (2009). Reading tea leaves: How humans interpret topic models. volume 32, pages 288–296.

[15] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

[16] Chiew, K. L., Yong, K. S. C., and Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems With Applications*, **106**, 1–20.

[17] Chin, T., Xiong, K., and hu, C. (2018). Phishlimiter: A phishing detection and mitigation approach using software-defined networking. *IEEE Access*, **PP**, 1–1.

[18] CISCO (2020). Cisco annual internet report (2018-2023) white paper. Technical report.

[19] da Costa, J. P. C. L., Freitas, E. P., Serrano, A. M. R., and de Sousa Jr, R. T. (2012). Improved parallel approach to pca based malicious activity detection in distributed honeypot data. *International Journal of Forensic Computer Science (IJoFCS)*.

[20] Daeef, A. Y., Ahmad, R., Yacob, Y., Na'imah, Yaakob, and Azir, K. N. F. K. (2016). Multi stage phishing email classification. Journal of Theoretical and Applied Information Technology.

[21] Danasingh, A. A., Balamurugan, S., and EPIPHANY, J. L. (2016). Literature review on feature selection methods for high-dimensional data. *International Journal of Computer Applications*, **136**.

[22] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, **41**(6), 391–407.

[23] Dependencies, U. (2020a). Universal features.

[24] Dependencies, U. (2020b). Universal pos tags.

[25] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, **10**(7), 1895–1923.

[26] Dinov, I. D. (2018). *Data Science and Predictive Analytics - Biomedical and Health Applications using R*. Springer.

[27] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley, New York, 2 edition.

[28] Erk, K. and Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 897–906, USA. Association for Computational Linguistics.

[29] et al., J. M. (????). The apache spamassassin public corpus. Technical report.

[30] Fang, Y., Zhang, C., Huang, C., Liu, L., and Yang, Y. (2019). Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism. *IEEE Access*, **7**, 56329–56340.

[31] Fette, I., Sadeh, N., and Tomasic, A. (2007). Learning to detect phishing emails. *International World Wide Web Conference Committee (IW3C2)*, pages 649–656.

[32] Galibus, T., de B. Vieira, T. P., de Freitas, E. P., de Oliveira Albuquerque, R., da Costa, J. P. C. L., de Sousa Júnior, R. T., Krasnoproshin, V., Zaleski, A., Vissia, H. E. R. M., and Galdo, G. D. (2017). Offline mode for corporate mobile client security architecture. *MONET*, **22**(4), 743–759.

[33] Gangavarapu, T. and Jaidhar, C. D. (2020). A novel bio-inspired hybrid metaheuristic for unsolicited bulk email detection. In V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, editors, *Computational Science – ICCS 2020*, pages 240–254, Cham. Springer International Publishing.

[34] Gangavarapu, T., Jaidhar, C., and Chanduka, B. (2020a). Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*.

[35] Gangavarapu, T., Jaidhar, C., and Chanduka, B. (2020b). Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*.

[36] Ghojogh, B., Samad, M. N., Mashhadi, S. A., Kapoor, T., Ali, W., Karray, F., and Crowley, M. (2019). Feature selection and feature extraction in pattern analysis: A literature review. *CoRR*, **abs/1905.02845**.

[37] Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers.

[38] Group, C. T. I. (2019). Total global email & spam volume for june 2019. Technical report.

[39] Gualberto, E. S., De Sousa, R. T., De Brito Vieira, T. P., Da Costa, J. P. C. L., and Duque, C. G. (2020a). The answer is in the text: Multi-stage methods for phishing detection based on feature engineering. *IEEE Access*, **8**, 223529–223547.

[40] Gualberto, E. S., De Sousa, R. T., De B. Vieira, T. P., Da Costa, J. P. C. L., and Duque, C. G. (2020b). From feature engineering and topics models to enhanced prediction rates in phishing detection. *IEEE Access*, **8**, 76368–76385.

[41] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, **3**, 1157–1182.

[42] Halgaš, L., Agrafiotis, I., and Nurse, J. R. C. (2020). Catching the phish: Detecting phishing attacks using recurrent neural networks (rnns). In I. You, editor, *Information Security Applications*, pages 219–233, Cham. Springer International Publishing.

[43] Hamid, I. R. A., Abawajy, J., and Kim, T.-h. (2013). Using feature selection and classification scheme for automating phishing email detection. *Studies in Informatics and Control*, **22**.

[44] Harikrishnan, N. B., Vinayakumar, R., , and Soman, K. P. (2018). A machine learning approach towards phishing email detection cen-securityiwspa 2018. In *CEUR Workshop Proceedings, CEUR-WS*, volume 2124, pages 21–28.

[45] Hassanpour, R., Dogdu, E., Choupani, R., Goker, O., and Nazli, N. (2018). Phishing e-mail detection by using deep learning algorithms. In *Proceedings of the ACMSE 2018 Conference*, ACMSE '18, New York, NY, USA. ACM.

[46] Howard, J. and Ruder, S. (2018). Fine-tuned language models for text classification. *CoRR*.

[47] Islam, M. R. and Abawajy, J. (2013). A multi-tier phishing detection and filtering approach. *Journal of Network and Computer Applications*, **36**, 324–335.

[48] Jolliffe, I. (2002). *Principal Component Analysis*. Springer Verlag.

[49] Jurafsky, D. and Martin, J. H. (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

[50] Kaji, N. and Kobayashi, H. (2017). Incremental skip-gram model with negative sampling. *CoRR*, **abs/1704.03956**.

[51] Kenter, T., Borisov, A., and de Rijke, M. (2016). Siamese CBOW: optimizing word embeddings for sentence representations. *CoRR*, **abs/1606.04640**.

[52] Landauer, T., Foltz, P., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, **25**, 259–284.

[53] Landauer, T., McNamara, D., Dennis, S., and Kintsch, W. (2007). *Handbook of Latent Semantic Analysis*. Taylor & Francis.

[54] Landauer, T. K. and Dutnais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *PSYCHOLOGICAL REVIEW*, **104**(2), 211–240.

[55] Lastdrager, E. E. H. (2014). Achieving a consensual definition of phishing based on a systematic review of the literature. *Crime Science*, **3**(9), 1–10.

[56] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, **abs/1405.4053**.

[57] L'Huillier, G., Hevia, A., Weber, R., and Ríos, S. A. (2010). Latent semantic analysis and keyword extraction for phishing classification. In C. C. Yang, D. Zeng, K. Wang, A. Sanfilippo, H. H. Tsang, M.-Y. Day, U. Glässer, P. L. Brantingham, and H. Chen, editors, *ISI*, pages 129–131. IEEE.

[58] Li, B., Drozd, A., Guo, Y., Liu, T., Matsuoka, S., and Du, X. (2019). Scaling word2vec on big corpus. *Data Science and Engineering*, **4**(2).

[59] Liu, K., Bellet, A., and Sha, F. (2015). Similarity Learning for High-Dimensional Sparse Data. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 653–662, San Diego, California, USA. PMLR.

[60] Maldonado, S. and L'Huillier, G. (2013). Svm-based feature selection and classification for email filtering. In P. Latorre Carmona, J. S. Sánchez, and A. L. Fred, editors, *Pattern Recognition - Applications and Methods*, pages 135–148, Berlin, Heidelberg. Springer Berlin Heidelberg.

[61] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

[62] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[63] Martin, C. D. and Porter, M. A. (2012). The extraordinary svd. *The American Mathematical Monthly*, **119**(10), 838–851.

[64] Martínez, J., Comesaña, C., and García Nieto, P. (2019). Review: machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics*.

[65] May, C., Duh, K., Durme, B. V., and Lall, A. (2017). Streaming word embeddings with the space-saving algorithm. *CoRR*, **abs/1704.07463**.

[66] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[67] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013b). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, **2013**.

[68] Mikolov, T., Yih, S. W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics.

[69] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).*

[70] Miller, G. A. (1995). Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, **38**, 39–41.

[71] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[72] Moradpoor, N., Clavie, B., and Buchanan, B. (2017). Employing machine learning techniques for detection and classification of phishing emails. In *2017 Computing Conference*, pages 149–156.

[73] Mujtaba, G., Shuib, L., Raj, R., Majeed, N., and al garadi, M. (2017). Email classification research trends: Review and open issues. *IEEE Access*, **PP**, 1–1.

[74] Najafabadi, M., Villanustre, F., Khoshgoftaar, T., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, **2**.

[75] Nazario, J. (????). Phishing Corpus. Technical report.

[76] Nidhin A Unnithan, Harikrishnan NB, V. R. and KP, S. (2018). Detecting phishing e-mail using machine learning techniques. In *Proceedings of the 1st Anti-Phishing Shared Task Pilot at 4th ACM IWSPA co-located with 8th ACM Conference on Data and Application Security and Privacy (CODASPY 2018).*

[77] on Kim, J. and Mueller, C. W. (1978). Factor analysis: Statistical methods and practical issues.

[78] Osuna, E., Freund, R., and Girosi, F. (1997). Support vector machines: Training and applications. Technical report, USA.

[79] Puschmann, D., Barnaghi, P. M., and Tafazolli, R. (2018). Using lda to uncover the underlying structures and relations in smart city data streams. *IEEE Systems Journal*, **12**, 1755–1766.

[80] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

[81] R, V., HB, B. G., M, A. K., and Kp, S. (2018). Deepanti-phishnet: Applying deep neural networks for phishing email detection cen-aisecurity@iwspa-2018. In [7], pages 40–50.

[82] Ramanathan, V. and Wechsler, H. (2012). phishgillnet—phishing detection methodology using probabilistic latent semantic analysis, adaboost, and co-training. *EURASIP Journal on Information Security*, **2012**.

[83] Ramanathan, V. and Wechsler, H. (2013). Phishing detection and impersonated entity discovery using conditional random field and latent dirichlet allocation. *Comput. Secur.*, **34**, 123–139.

[84] Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. Technical report, Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 08855e.

[85] Rea, A. and Rea, W. (2016). How many components should be retained from a multivariate time series pca?

[86] Rosner, F., Hinneburg, A., Röder, M., Nettling, M., and Both, A. (2014). Evaluating topic coherence measures. *CoRR*, **abs/1403.6397**.

[87] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

[88] Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 399–408.

[89] Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, **10**, e0118432.

[90] Shin, D.-h. (2018). Applications of artificial intelligence in the export control domain. In Y. Bi, S. Kapoor, and R. Bhatia, editors, *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, pages 1005–1011, Cham. Springer International Publishing.

[91] Shlens, J. (2005). A tutorial on principal component analysis. In *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*.

[92] Singh, J. (2011). Detection of phishing e-mail. *International Journal of Computer Science and Technology*, **2**(3), 547–549.

[93] Sonowal, G. and Kuppusamy, K. S. (2017). Phidma – a phishing detection model with multi-filter approach. In *Journal of King Saud University – Computer and Information Sciences*, pages 1–14.

[94] Stats, I. W. (2020). Internet usage statistics - the internet big picture: World internet users and 2020 population stats. Technical report.

[95] Stevens, K., Kegelmeyer, P., Andrzejewski, D., and Buttler, D. (2012a). Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, page 952–961, USA. Association for Computational Linguistics.

[96] Stevens, K., Kegelmeyer, P., Andrzejewski, D., and Buttler, D. (2012b). Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, page 952–961, USA. Association for Computational Linguistics.

[97] Tang, Y.-K., Mao, X.-L., Huang, H., Shi, X., and Wen, G. (2018). Conceptualization topic modeling. *Multimedia Tools Appl.*, **77**(3), 3455–3471.

[98] Tenório, D. F., da Costa, J. P. C. L., and Sousa Jr, R. T. (2013). Greatest eigenvalue time vector approach for blind detection of malicious traffic. *The International Conference on Forensic Computer Science (ICoFCS).*

[99] Toolan, F. and Carthy, J. (2010). Feature selection for spam and phishing detection. In *2010 eCrime Researchers Summit*, pages 1–12.

[100] Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, **37**(1), 141–188.

[101] Unnithan, N., Harikrishnan, N., Vinayakumar, R., Soman, K., and Sundarakrishna, S. (2018a). Detecting phishing e-mail using machine learning techniques cen-securenlp. In *CEUR Workshop Proceedings, CEUR-WS*, volume 2124, pages 50–56.

[102] Unnithan, N. A., Harikrishnan, N. B., Akarsh, S., Vinayakumar, R., and Soman, K. P. (2018b). Machine learning based phishing e-mail detection security-cenamrita. In *CEUR Workshop Proceedings, CEUR-WS*, volume 2124, pages 64–68.

[103] Vazhayil, A., NB, H., R, V., and KP, S. (2018). Ped-ml: Phishing email detection using classical machine learning techniques censec@amrita. In [7], pages 70–77.

[104] Verleysen, M. and François, D. (2005). The curse of dimensionality in data mining and time series prediction. In *Proceedings of the 8th International Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, IWANN'05, pages 758–770, Berlin, Heidelberg. Springer-Verlag.

[105] Verma, R., Shashidhar, N., and Hossain, N. (2012). Detecting phishing emails the natural language way. In S. Foresti, M. Yung, and F. Martinelli, editors, *Computer Security – ESORICS 2012*, pages 824–841, Berlin, Heidelberg. Springer Berlin Heidelberg.

[106] Verma, R. M., Zeng, V., and Faridi, H. (2019). Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2605–2607, New York, NY, USA. Association for Computing Machinery.

[107] Vieira, T. P., Tenrio, D. F., da Costa, J. P. C., de Freitas, E. P., Galdo, G. D., and de Sousa Jnior, R. T. (2017). Model order selection and eigen similarity based framework for detection and identification of network attacks. *J. Netw. Comput. Appl.*, **90**(C), 26–41.

[108] Vinayakumar, R., Barathi Ganesh, H., Anand Kumar, M., Soman, K., and Poornachandran, P. (2018). Deepanti-phishnet: Applying deep neural networks for phishing email detection cen-aisecurityiwspa-2018. In *CEUR Workshop Proceedings, CEUR-WS*, volume 2124, pages 39–49.

[109] Yang, P., Zhao, G., and Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. *IEEE Access*, **PP**, 1–1.

[110] Yasin, A. and Abuhasan, A. (2016). An intelligent classification model for phishing e-mail detection. *International Journal of Network Security & Its Applications (IJNSA)*, **8**(4), 55–72.

[111] Zareapoor, M. and Seeja, K. R. (2015). Feature extraction or feature selection for text classification: A case study on phishing email detection. *International Journal of Information Engineering and Electronic Business*, **7**.

[112] Zeng, V., Baki, S., Aassal, A. E., Verma, R., De Moraes, L. F. T., and Das, A. (2020). Diverse datasets and a customizable benchmarking framework for phishing. In *Proceedings of the Sixth International Workshop on Security and Privacy Analytics*, IWSPA '20, page 35–41, New York, NY, USA. Association for Computing Machinery.

[113] Zhu, E., Chen, Y., Ye, C., Li, X., and Liu, F. (2019). Ofs-nn: An effective phishing websites detection model based on optimal feature selection and neural network. *IEEE Access*, **PP**, 1–1.