Université du Québec en Outaouais

Département d'Informatique et d'Ingénierie

A Learning Approach for Spam Detection Using Semantic Representation

Par

Nadjate Saidani

Presented to the Department of computer science and engineering as fulfillment of the thesis requirement for the degree of Doctor of philosophy in Science and Information Technologies

Evaluation jury

Chairman	Dr. Luigi Logrippo, Université du Québec en Outaouais
External examiner	Dr. Marc Frappier, Université de Sherbrooke
Internal examiner	Dr. Alan Davoust, Université du Québec en Outaouais
Research advisor	Dr. Kamel Adi, Université du Québec en Outaouais
Research co-advisor	Dr. Mohand Saïd Allili, Université du Québec en Outaouais

Gatineau, Quebec, Canada, 2021.

Abstract

Emails can be considered as the most popular means of communication in the last decades. However, email popularity comes with a major problem related to the reception of unsolicited or unwanted emails, commonly known as spam, which represent a major threat for individuals and organizations.

Even though the proposed solutions for spam detection in the literature have come a long way, spam emails still represent a real problem for IT infrastructure security. The most recent spam detection methods use content-based approaches which have shown promising results. These methods usually use text representations in the form of feature spaces allowing spam/legitimate emails discrimination using classification algorithms. However, most of these methods use holistic and high-dimensional spaces that do not consider high-level semantical aspects of the text and ignore spam specificity in different thematic domains.

Our work tackles this problem by proposing an original approach for spam detection that interprets email content on two distinct semantic levels. In the first level, we categorize emails by specific thematic domains (e.g., Health, Education, Finance, etc.) to enable a separate conceptual view for spam in each domain. In the second level, we automatically extract in each domain a set of semantic features from labeled emails, represented in the form of rules, for spam detection. These features are meant to summarize the email content into topics forming compact feature spaces that efficiently discriminate spam from legitimate emails.

Experiments on a large corpus of emails have shown that the proposed method provides an efficient representation of the internal semantic structure of email content, which allows for more precise spam detection results compared to existing methods. They have also demonstrated that having a specialized classifier to target the spam of each domain can enhance the within domain spam/legitimate email discrimination, and boost the overall spam detection performance.

Sommaire

L'utilisation du courriel est considérée comme le moyen de communication le plus populaire depuis des décennies. Cependant, sa popularité a engendré un problème majeur lié à la réception de courriels non sollicités et indésirables. Ces courriels, communément appelés spam, représentent une menace majeure pour les individus et les organisations. Même si, dans la littérature, beaucoup de travaux ont été consacrés à la détection de spam, ce dernier représente toujours un réel problème pour la sécurité des infrastructures informatiques.

La plupart des travaux de recherche pour la détection de spam utilisent des approches basées sur le contenu, qui ont montré des résultats prometteurs. Ces dernières utilisent habituellement des représentations textuelles, sous forme d'espaces de caractéristiques, permettant la discrimination courriel spam/courriel légitime à l'aide des algorithmes de classification. Cependant, la plupart de ces méthodes utilisent des espaces holistiques et de grandes dimensions qui ne considèrent pas les aspects sémantiques de haut niveau du texte et ignorent la spécificité du spam dans différents domaines thématiques.

Cette thèse traite ce problème en proposant une approche originale pour la détection de spam qui interprète le contenu des courriels sur deux niveaux sémantiques différents. Dans le premier niveau, nous catégorisons les courriels par domaines thématiques spécifiques (ex., Santé, Education, Finance, etc.) pour permettre une vue conceptuelle distincte pour les courriels spam dans chaque domaine. Dans le second niveau, nous extrayons automatiquement dans chaque domaine un ensemble de caractéristiques sémantiques à partir de courriels étiquetés, que nous représentons sous forme de règles permettant la détection de spam. Ces caractéristiques résument le contenu des courriels en un ensemble de sujets formant des espaces de caractéristiques compacts qui distinguent efficacement les courriels spam des courriels légitimes.

Des expériences sur un large corpus de courriels ont montré que la méthode proposée fournit une représentation efficace de la structure sémantique interne du contenu des

courriels, ce qui permet d'obtenir des résultats de filtrage anti-spam plus précis et plus efficaces par rapport aux méthodes existantes. Ils ont également démontré que le fait de disposer d'un classificateur spécialisé pour cibler les messages spam de chaque domaine peut améliorer la discrimination spam/courriels légitimes au sein du domaine et améliorer les performances globales de détection de spam.

Remerciements

Ce travail de thèse est réalisé au Laboratoire de Recherche en Sécurité Informatique (LRSI) de l'Université de Québec en Outaouais sous la direction bienveillante du Professeur Kamel Adi et de l'accompagnement éclairé du Professeur Mohand Saïd Allili. Son achèvement, après plusieurs années de recherche, me procure aujourd'hui fierté et satisfaction, et là, justement, c'est l'occasion de se remémorer toutes les étapes accomplies, les nombreuses difficultés qu'il a fallu surmonter, mais surtout les personnes qui m'ont permis d'en arriver à la soutenance d'une thèse tant attendue par ma famille.

Je voudrais tout d'abord présenter mes remerciements les plus vifs à Monsieur Kamel Adi, professeur au Département d'informatique de l'Université de Québec en Outaouais, pour avoir accepté de m'accueillir dans son équipe et de diriger en toute rigueur ce travail de recherche. Je lui exprime également ma reconnaissance pour la confiance qu'il m'a constamment témoignée, pour les connaissances qu'il m'a prodiguées et pour ses conseils et remarques aussi pertinentes que constructives; qu'il soit assuré de mon profond respect et de ma totale gratitude.

Le long de la préparation de ma thèse, j'ai eu aussi le privilège et l'opportunité de travailler avec Monsieur Mohand Saïd Allili, professeur au Département d'informatique de l'Université de Québec en Outaouais. L'occasion m'est donnée pour exprimer mes remerciements les plus sincères pour m'avoir fait bénéficier de ses compétences et pour avoir été constamment présent et à l'écoute.

Je tiens à témoigner ma profonde reconnaissance à Monsieur Luigi Logrippo, professeur au Département d'informatique de l'Université de Québec en Outaouais pour avoir accepté de présider mon jury de thèse et de sacrifier une partie de son temps dans l'évaluation scientifique du présent travail. Je le remercie très chaleureusement.

J'adresse ma profonde gratitude à Monsieur Marc Frappier, professeur au Département d'informatique de l'Université de Sherbrooke et à Monsieur Alan Davoust, professeur au Département d'informatique de l'Université de Québec en Outaouais pour l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs du présent travail. Leurs remarques et suggestions vont me permettre certainement d'ouvrir de nouvelles perspectives dans mes recherches.

J'adresse mes remerciements les plus chers à mon mari qui m'a fortement soutenu tout au long de cette thèse et qui a été à la fois tendre et patient pour partager la joie de ma soutenance, après avoir vécu ensemble le bonheur de la naissance de notre fille Celena.

Je voudrais particulièrement souligner le soutien de ma famille et de celle de ma belle-famille qui a été permanent et sans faille. Je les remercie, un par un du fond de mon cœur et leur dire combien je les aime.

To my beloved parents Noura et Boualem

To my sweet daughter Celena

Contents

List of Tables

List of Figures

List of Abbreviations

AUC	Area Under Curve
BoW	Bag-of-Words
$CBOW$	Continuous Bag-Of-words
$CSBC$	Conceptual Similarity Based on Corpus
DL	Deep Learning
$EGPA$	Email Geographic Path Analysis
$eTVSM$	enhanced Topic-based Vector Space Model
$FERC$	Federal Energy Regulatory Commission
GA	Genetic Algorithm
$GloVe$	Global Vectors for word representation
HMM	Hidden Markov Models
IG	Information Gain
IT	Information Technology
KNN	K-Nearest Neighbor
$K-NNC$	K-Nearest Neighbor Classifier
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
MI	Mutual Information
ML	Machine Learning
$MMA-MF$	Multi-Modal Architecture based on Model Fusion
$MRMR$	Minimum Redundancy Maximum Relevance
NB	Naive Bayes
NLP	Natural Language Processing
NNC	Nearest Neighbor Classifier
$OCFS$	Orthogonal Centroid Feature Selection

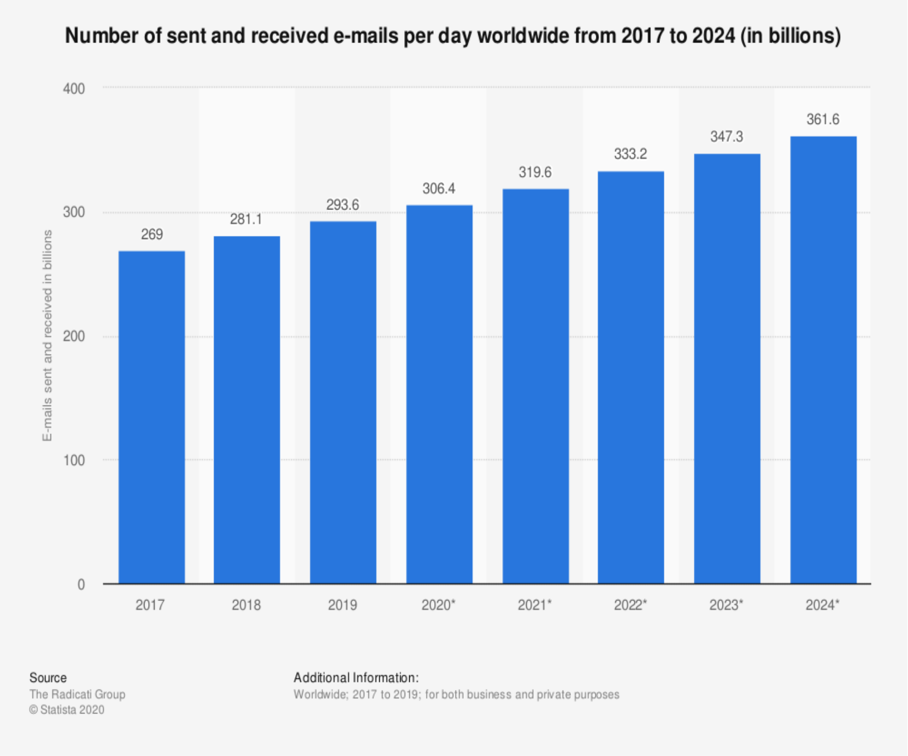
$OMFS$	Orthogonal Minimum Feature Selection
OR	Odds Ratio
$pLSA$	probabilistic Latent Semantic Analysis
REP	Reduced Error Pruning
SD	Subgroup Discovery
SG	Skip Gram
$SSBWL$	Semantic Similarity Based on the Wikipedia Links
$SNARE$	Spatio-temporal Network-level Automatic Reputation Engine
SVD	Singular Value Decomposition
SVM	Support Vector Machine
$TBSR-SD$	Text-Based Semantic Representation for Spam Detection
tf	term frequency
$tf-idf$	term frequency-inverse document frequency
$TVSM$	Topic-based Vector Space Model
VSM	Vector Space Model
$WRAcc$	Weighted Relative Accuracy

1

Introduction

1.1 Context and motivation

Electronic mail or email is one of the most used services on the Internet given the advantages it offers in terms of transmission speed, the ability to handle multimedia documents, and broadcasting messages at a very low cost. As can be seen in Figure 1.1, around 306.4 billion emails were exchanged per day in 2020. This number is supposed to reach 361.6 billion by the end of 2024. However, email's popularity comes with a major problem related to the reception of unsolicited or unwanted emails. These emails, commonly known as spam, represent a major threat for individuals and organizations. Indeed, spam overload mailboxes with unwanted messages, causing a loss in network bandwidth and storage space. It also favors a fast distribution of false information and the spreading of malicious codes. As shown in Figure 1.2 [23], spam represented over 55% of the total email traffic in January 2020.


Number of sent and received e-mails per day worldwide from 2017 to 2024 (in billions)

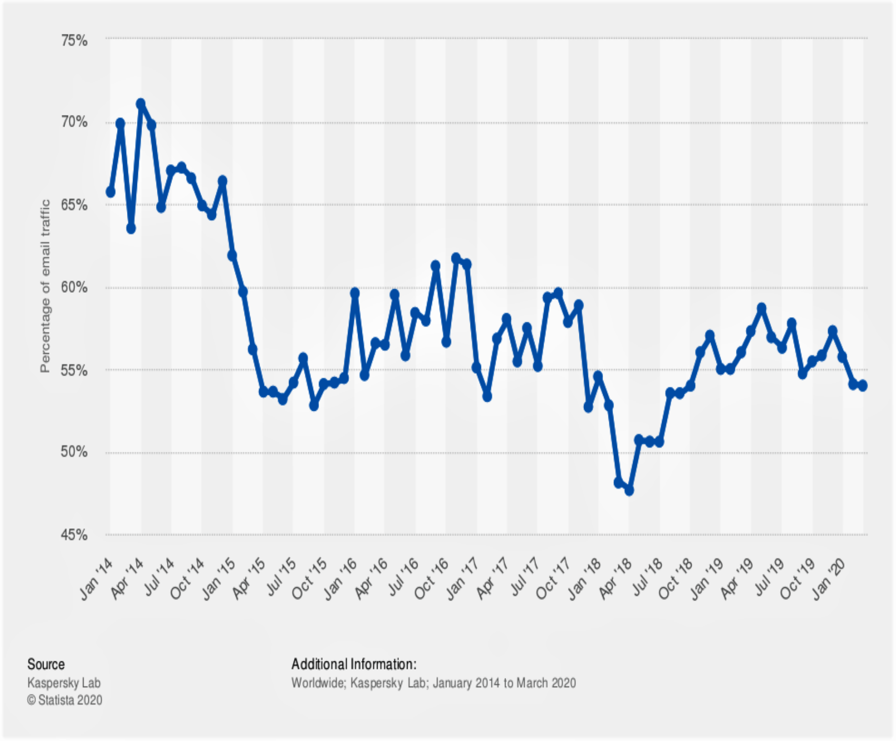
Source
The Radicati Group
© Statista 2020

Additional Information:
Worldwide; 2017 to 2019; for both business and private purposes

Figure 1.1: Number of sent and received emails per day, 2017-2024.

Spam is at the origin of a drastic decrease in productivity within organizations, causing a loss of billions of dollars [11]. According to scientific researchers, M. Rao of Microsoft Research and H. David of Google, spam costs US businesses and consumers about $20 billion per year, while spammers and spam-advertised merchants make about $200 million a year in profit [13]. For instance, the Nigerian Prince scams gross over $700,000 a year [74].

In the literature, several definitions have been used for spam. For instance, Cranor *et al.* [28] defined spam as "*unsolicited bulk email*". However, this definition seems to be very narrow, as it means that unsolicited emails that are not sent in bulk do not have a negative impact on systems which is not always true in practice. A more precise definition was provided by Cormack *et al.* [27] and say that "*spam is unsolicited and unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient*". In [26], Cormack provided a more general definition of



Source
Kaspersky Lab
© Statista 2020

Additional Information:
Worldwide; Kaspersky Lab; January 2014 to March 2020

Figure 1.2: Proportion of spam in email traffic, 2014-2020.

spam based on the adversarial nature of spam and spam filters. He defined spam as "*unwanted communication intended to be delivered to an indiscriminate target, directly or indirectly, notwithstanding measures to prevent its delivery*". In addition, he defined spam filters as "*an automated technique to identify spam for the purpose of preventing its delivery*". Other definitions of spam can be found in [106]. Ham emails are the opposite of spam: they represent all the legitimate emails that are accepted by the recipients.

To deal with the problem of spam, several detection methods have been proposed in the literature [11, 17, 30, 124]. These methods generally use two types of information contained in emails, namely non-content and content-based information [11]. Non-content information refers to email headers and related sender information such as the email sender address domain (IP address), his reputation, his writing style and sending time. List-based approaches are among the earliest proposed solutions for spam detection [10]. Commonly used techniques in list-based filters are *Whitelist* and *Blacklist* [26, 54]. The *Whitelist* method allows the recognition of trusted sender addresses while the *Black-*

list method can block unsolicited emails based on fake sender addresses [115], known as spammer addresses. However, mailing lists must be continually updated to maintain the effectiveness of these methods, which limits their applicability with a highly evolving spam content.

Content-based information refers to textual information contained within email bodies and subjects. Recently, machine learning techniques have shown tremendous success for content-based spam detection [10, 17], after their indisputable success in text categorization [49]. In fact, spam filtering can be seen as a two-class (spam, ham) text categorization and, therefore, several automatic classification methods were applied, such as Support Vector Machines (SVM) [112], Naive Bayes (NB) [116], Artificial Neural Networks (ANN) [93], etc. These methods mainly consist of building classification models by learning from email features (e.g., email text words, n-grams, etc.). Email features can usually be extracted manually using hand-crafted rules, also called knowledge engineering, or inferred automatically using text mining techniques [10, 30]. For manual feature extraction, rules are built by experts and regularly updated to maintain the efficiency of spam detection systems. They are generally coded using compact regular expressions to specify complex text patterns [86]. This technique has been used, for instance, in the SpamAssassin system [110] and by some other researchers such as Sahami *et al.* [95] and Schleimer *et al.* [101]. In automatic features extraction, a corpus of annotated emails is automatically analyzed using text mining algorithms to extract useful information such as words, characters, HTML markup and various document statistics [10]. Our work focuses on content-based approaches for spam detection.

1.2 Problem statement

Since content-based spam filtering uses mainly text classification, text representation constitutes a cornerstone for this type of methods. Among the most popular methods for text representation, we can find the Bag-of-Words (BoW) model [4] where an unstructured set of word tokens are used to discriminate between spam and legitimate messages. More precisely, the BoW model describes the text of an email as a vector of words where each word represents an individual feature of the email. Unfortunately, this representation is very high-dimensional, as each word is treated as an attribute in the feature space. In addition, this model may incur an important loss in the email semantics since words are taken independently. Indeed, individual word tokens can be plagued with polysemy and sometimes intentionally modified by spammers to prevent their detection by spam filters [99]. To alleviate this issue, researchers have used n-gram models [14] where emails are represented by sequences of words which carry more semantical content than mere words, which in turn leads to more refined models. However, this approach increases exponentially the size of the vocabulary, which leads to highly sparse spaces for representing email documents.

Inspired by natural language processing (NLP) techniques, several other researchers have used semantic features for spam detection [94]. For example, methods have used word synonyms created from WordNet Ontology [99] or word sense disambiguation [65] to identify the appropriate meaning of polysemous words in specific contexts. These methods have reported a better performance compared to the BoW and n-gram models. However, these methods do not consider the different writing styles between legitimate users and spammers. In other words, semantic approaches project the email text contents into semantic space without considering whether the email is spam or ham. Moreover, they do not capture all the semantic subtleties discriminating ham/spam as the lexicon used by spammers is greater than that of natural language.

Due to the increasing sophistication of content obfuscation techniques, spammers have been able to generate emails with content close to the natural language, which makes their discrimination with legitimate emails very difficult [117]. Hence, having a richer semantic representation close to the natural language is required to depict and capture, in an explicit way, the information conveyed by emails which can enable discrimination between legitimate and spam emails [17]. A major problem then is the extraction of high-level semantic meanings from the content of emails and using these as features to build vector spaces that could decrease the dimension of vectors and, at the same time, enhance the spam classification accuracy.

On the other hand, spam content can drastically vary between different domains (cat-

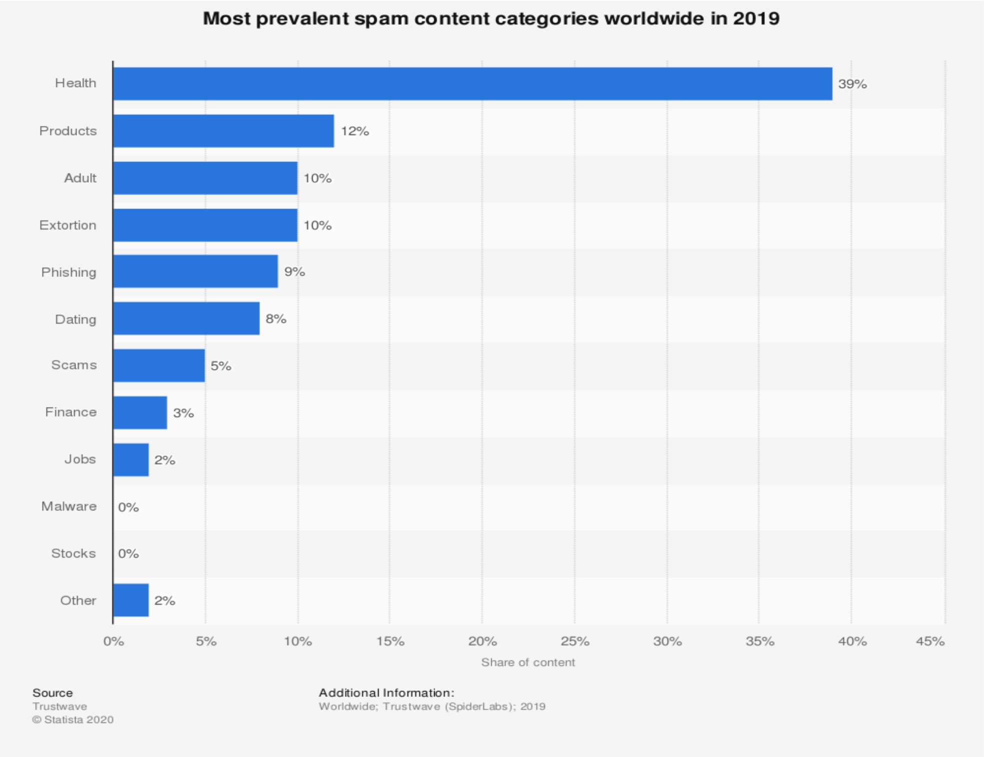


Figure 1.3: The most common categories of spam content sent in 2019.

egories) targeted by spammers. For instance, in spam campaigns related to the subject of health, spam content is mainly oriented to medicine or false therapy campaigns advertisement, whereas in finance, spam carry advertisements for dubious financial services and products. Figure 1.3 shows the most targeted domains by spammers in 2019. Given the variety of semantics carried by different domains, trying to have a general and unified semantic representation for all domains can be a hard or even counter-productive pursuit. Several research works have proved that complex representations of texts do not always improve the efficiency of spam classification [26]. Therefore, we argue that using domain-specific spam representation can be more advantageous for spam discrimination than using a general-purpose one. In other words, having a specialized classifier to target the spam of each domain can enhance the within domain spam/legitimate email discrimination, and therefore enhance the overall spam detection task.

1.3 Contributions

The main goal of this thesis is to propose a new approach for efficient and accurate spam filtering using semantic representation learning. To achieve this goal, we propose:

- a method for automatic extraction of semantic features for domain-specific email spam detection. Our approach analyses email contents at two semantic levels.

  The first level broadly categorizes the subject of emails in order to define a domain-specific spam characterization using NLP and machine learning techniques. To categorize emails by domain, five most targeted categories by spammers are considered: Computer, Adult, Education, Finance, and Health. The second level uses an approach inspired by the CN2-SD algorithm [22] to automatically extract semantic rules for spam detection. Each rule has a binary class outcome (spam/ham) and acts by itself as a weak classifier for discriminating spam. The combination of these rules produces a strong classifier enabling robust and accurate spam detection compared to existing methods [98]. This work resulted in the following publication :

  - Nadjate Saidani, Kamel Adi, and Mouhand Said Allili. A supervised approach for spam detection using text-based semantic representation. In *Int'l Conference on E-Technologies*, pages 136-148. Springer, 2017.

- to extend our method using a hybrid scheme composed of manually-specified and automatically-extracted semantic rules for spam detection. Automatic rules extraction allow an efficient extraction of basic semantics, while manually-specified rules allow semantics tuning by incorporating domain-specific knowledge of experts and end-users. Moreover, to allow optimal integration of the two types of rules, we propose an algorithm for eliminating redundancy and conflicts between rules. Finally, we validate our approach using extensive experiments on a large corpus of emails composed of six different domains: Computer, Adult, Education, Finance, Health and Others. The category "Others" is used to ensure the completeness of the categories. We compared our approach to a method based on the BoW model and some existing semantic models, and demonstrate that our approach yields a higher performance for spam detection compared to the other methods [97]. This work resulted in the following publication:

  - Nadjate Saidani, Kamel Adi, and Mohand Said Allili. A semantic-based classification approach for an enhanced spam detection. *Computers & Security*, 94:101716, 2020.

- to explore the use of more elaborated semantic features based on deep learning and semantic ontology to spam filtering. For this purpose, we use the Word2Vec model [78] to categorize email documents by domains at the first level. In the second level, we use an approach integrating eTVSM semantic ontology [99] and CN2-SD [22] to generate semantic rules. Finally, we compare our approach with the previously obtained results and some other recent proposed approaches [96]. This work resulted in the following publication :

  - Nadjate Saidani, Kamel Adi, and Mohand Said Allili. Semantic representation based on deep learning for spam detection. In *Int'l Symposium on Foundations and Practice of Security*, pages 72–81. Springer, 2019

1.4 Thesis outline

The rest of this thesis is structured as follows:

In chapter 2, we present an overview of text classification algorithms. We introduce the general framework of machine learning and define different machine learning types. We also provide a background on text mining and outline the most common machine learning algorithms, especially those used in the classification of text documents. Finally, we describe some popular evaluation methods for text classification algorithms. In chapter 3, we provide an overview of the state-of-the-art related to spam filtering techniques based on content and non-content features. As our work focuses on semantic methods, we pay particular attention to techniques based on semantic content analysis. In chapter 4, we propose a two-level semantic analysis of emails to enable domain-specific spam detection. In chapter 5, we propose a hybrid approach which is an extension of the semantic features introduced in chapter 4. In chapter 6, we explore the use of the embedding words approach based on deep learning for spam detection. Finally, in chapter 7, we summarize the main contributions of the present thesis and we suggest some possible avenues for further research.

2

Background on text classification algorithms

2.1 Introduction

Automatic text classification is one of the most important tasks of Machine Learning (ML), it automatically assigns one or more predefined categories to a text document. To further elaborate the text classification process using machine learning techniques, we have divided this chapter into five main parts. The first part summarizes the different paradigms of machine learning. The second part includes the different types of machine learning algorithms. The third and the fourth parts provide background on text mining and text classification algorithms, respectively. Finally, the fifth part presents a synthesis of different methods used to evaluate machine learning systems.

2.2 Machine learning

Machine learning is one of the fields of artificial intelligence [32]. This field refers to development, analysis and implementation of methods that allow a machine to improve its current performance based on learning from data. A general scheme of automatic learning process is given in Figure 2.1.

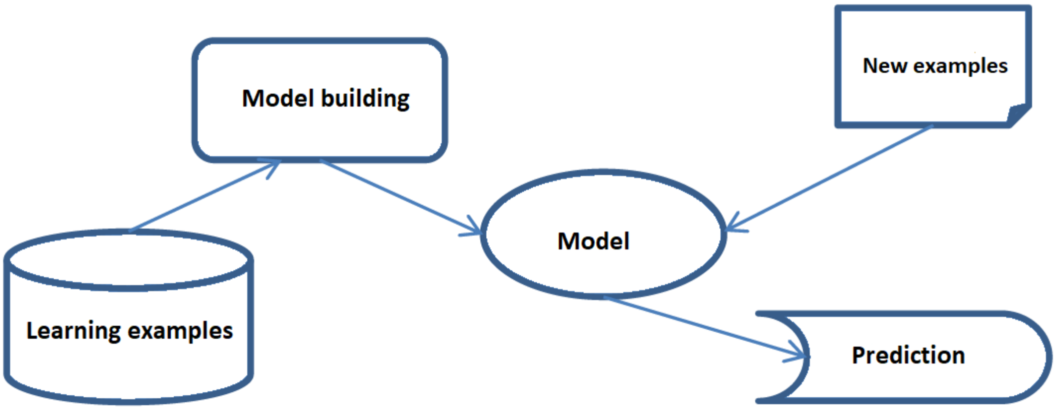


Figure 2.1: Machine learning process.

Machine learning systems use datasets containing pre-collected examples, known as training or learning data (e,g. emails, images, etc.), of the problem to be addressed in order to make decisions (predictions). More specifically, in the learning phase, the training data is used to extract knowledge in order to build a model that hopefully generalizes to all possible examples of the treated problem. Afterward, this model is used as a reference to classify new examples not observed during learning stage. In other words, the main task of machine learning is to create a model with good prediction performance on the test data that contains new examples. The algorithms derived from machine learning can be applied to various situations and are particularly suited to the problem of automated decision-making. These include Natural Language Processing (NLP), such as text categorization and spam detection, video analysis, image classification, voice recognition, pattern identification, web research, medical diagnosis, fault control, etc. [16, 44, 103].

2.3 Machine learning types

Any machine learning algorithm works using a set of training data. Depending on the input data used by the algorithm, machine learning can be categorized into two main types: supervised learning and unsupervised learning. Each category contains several algorithms tailored for different problems. These types of learning can also be combined in the same system to increase its accuracy. Figure 2.2 shows the common groupings of machine learning algorithms. In this chapter, we will mainly talk about supervised learning algorithms as our contributions are based on this method and we will briefly discuss unsupervised learning.

2.3.1 Supervised learning

Supervised learning algorithms are widely used in various research problems, such as intrusion detection, spam filtering, speech recognition, text categorization [32]. In this type of learning, the actual output values (labels) of all examples in input data (training data) are known and defined in advance. The goal of these algorithms is to bring out, from a known set of input data and known responses (output values) to the data, a model allowing to assign right predictions for the response to new data. Supervised learning is typically done in the context of classification or regression to develop predictive models (see Figures 2.2 and 2.3 (a,b)).

In supervised learning, there are two main techniques: classification and regression. Mainly, the classification technique predicts discrete responses, usually called classes, labels or categories. For example, in the case of spam filtering problem, predict whether the received email belongs to spam or ham class. The regression technique predicts continuous responses such as predicting the stock prices using historical data. Commonly used algorithms in supervised learning include logistic regression, naive Bayes, Support vector machines, Artificial neural networks and Random forests.

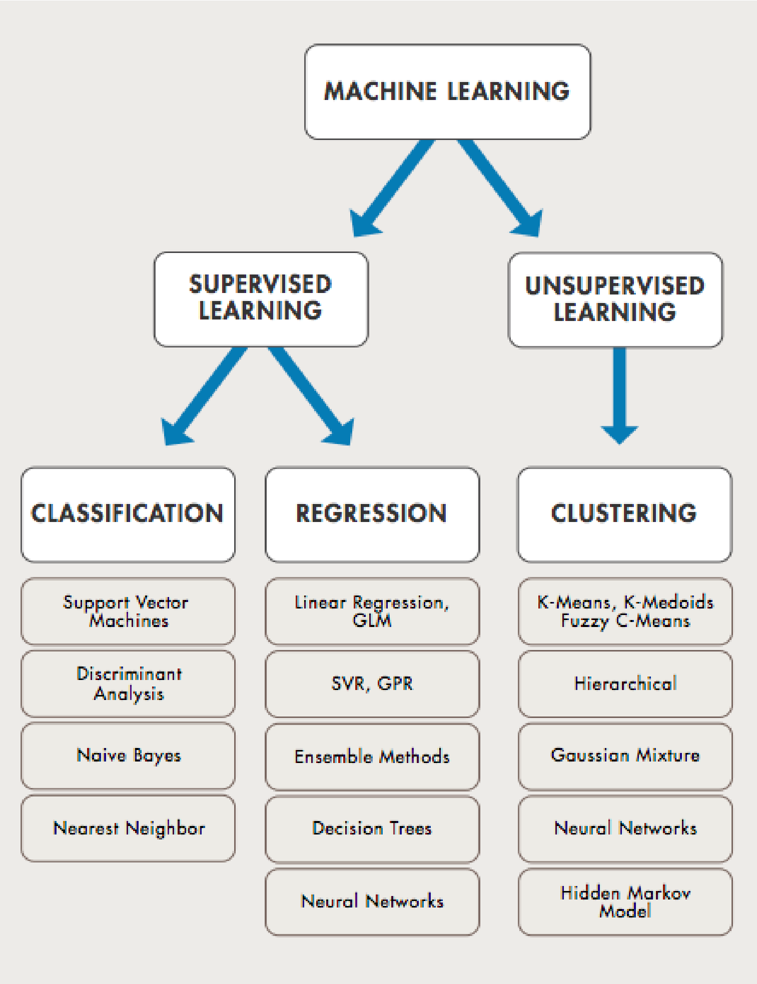


Figure 2.2: Machine learning techniques include both unsupervised and supervised learning [111].

2.3.2 Unsupervised learning

In this type of learning, unlike supervised learning, only the information relating to input examples is known and the desired output values are unknown. Unsupervised learning method looks to group examples of input data into homogeneous spectral output values. Therefore, the algorithm must determine its output values itself according to the similarities detected between examples of input data using metrics such as Euclidean or probabilistic distance. Clustering is the most common unsupervised learning method, it seeks to group examples so that those within the same group are sufficiently similar, and

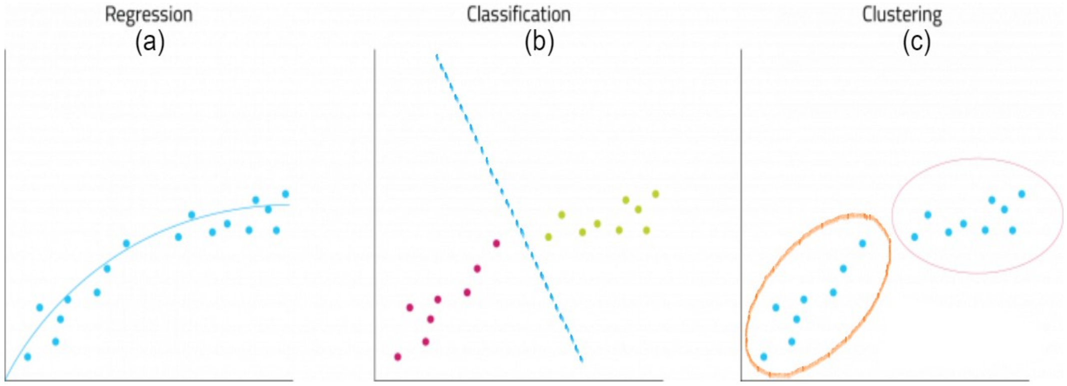


Figure 2.3: (a) Regression, (b) Classification and (c) Clustering techniques in machine learning.

those within different groups are sufficiently different (Figure 2.3). Commonly used algorithms in clustering include k-means clustering, Hierarchical clustering, Hidden Markov models and Gaussian mixture models.

2.4 Text classification

Text classification uses machine learning and NLP techniques to automatically classify text documents into one or more predefined categories according to their content. Over the last few decades, text classification techniques have been applied in many applications, such as sentiment analysis, spam detection, news-article topic labeling, etc. Text classification based on ML comprises four main components: feature extraction, feature selection or dimension reduction, model learning and evaluation. Figure 2.4 illustrates an overview of the text classification process. The two first components, feature extraction and dimension reduction are detailed in the following subsections, while classification algorithms and evaluation methods are presented in the further sections.

2.4.1 Text feature extraction

Text feature extraction is the process of taking out a list of terms (terms could be words, n-grams, characters, phrases, part of speech, topics, etc.) from the text documents (corpus)

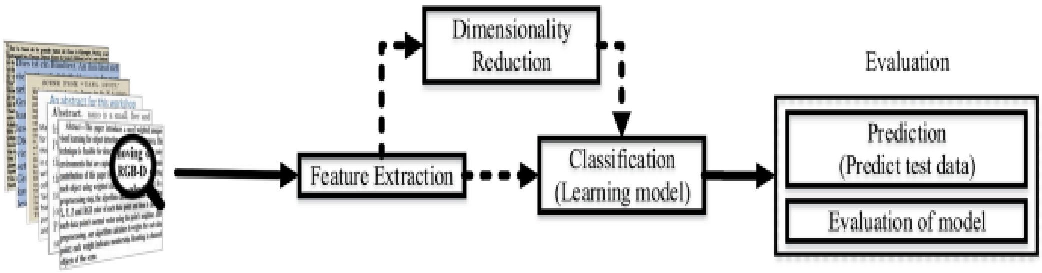


Figure 2.4: Text classification process [64].

and converting them into feature space. Usually, the text documents are both highly dimensional and unstructured data. However, to make this data easy to handle by ML algorithms, it is transformed from a raw text data to numerical data using mathematical modeling. Before this transformation, a text preprocessing phase is needed to clean up noisy and unnecessary text data to reduce feature dimensionality and improve classifier performance. Text preprocessing includes four main steps, namely: Tokenization, Noise and Stop-word removal, Stemming and Lemmatization. Tokenization permits segmenting email content into a set of words, phrases, etc. Noise and Stop-word removal deletes unnecessary characters such as punctuation and special characters and most common words in a text such as articles, prepositions, etc. Stemming allows transforming words into their roots, for example the stem of "studying" is "study". Lemmatization groups together the different inflected forms of a word to its meaningful base form, for example, the lemma of gone, going and went is go. This section discusses two common techniques of text feature extraction: *term (word) weighting* and *word embedding* techniques.

1) **Term weighting**

   Term or word weighting schemes require that all documents in the dataset are first converted to feature vectors. Then, assign appropriate weights to each item in the feature vector. The weight of each term in a document indicates its importance in representing the content of that document. Below, we discuss the most commonly used weighting schemes.

- **Binary weighting:**

  Binary weighting is the simplest scheme of feature extraction methods. The scheme uses binary feature values, it assigns "1" to any term present in the document and "0" otherwise.

- **Term frequency:**

  Term frequency (tf) measure calculates $f_{t,d}$, the number of occurrences of each term t in a document d and assigns it to feature space. To avoid bias for long documents (overweighting), this measure can be normalized by $\sum_{t' \in d} f_{t',d}$, the total number of occurrences of all terms in a document. The tf can be formulated as follows:

  $$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}. \tag{2.1}$$

  There are several other variants of the tf, the most common one is the logarithmic term frequency. This variant uses a logarithmic function to dampen the importance of high-frequency terms that can affect rare but important terms. The logarithmic tf scheme is calculated as follows:

  $$tf(t,d) = \log(1 + f_{t,d}). \tag{2.2}$$

  By adding one to the log function, we keep weight equal to 0 on absent terms.

- **Term frequency-inverse document frequency:**

  Term frequency-inverse document frequency ($tf{-}idf$) is among the most successful term weighting scheme. The $tf{-}idf$ was proposed to resolve the problem of common terms (words) in a document. Its principle is to determine the importance of a word in the document and the entire dataset (corpus). Two statistical methods are used by $tf{-}idf$: term frequency (tf) and inverse document frequency (idf). The tf refers to the normalized term frequency as described above. The idf computes the logarithm of the total number of documents $|D|$ in the dataset divided by the number of documents where the word t appears. The *idf* method increases the weight of important words and decreases the weight of insignificant words. Thus, given a set of training data D, where a document $d \in D$ and a term $t \in d$, the *idf* equation can be given as:

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}. \tag{2.3}$$

By multiplying the tf method and the *idf* method, we can obtain high-frequency terms that provide a particularly important context for a single document in a set of documents. The formal representation of the $tf - idf$ is given as follow:

$$tf{-}idf(t, d, D) = tf(t, d) \cdot idf(t, D). \tag{2.4}$$

As described previously, several weighting methods are used to determine the value of a term in the feature space. The weighting methods can also be viewed as a form of Bag of Word (BoW) models. However, these methods suffer from some limitations. Firstly, they don't take into account context and synonyms of terms in a document. In addition, the resulting matrices of these methods are usually huge and sparse which induces a problem of high dimensionality. For more details about the advantages and drawbacks of different feature extraction techniques see [64].

2) **Word embedding**

Word embedding is NLP technique that involves deep learning. This technique maps the document words to high-dimensional vectors of real numbers using a neural network. This technique aims to capture the contexts and semantics of a word in a document using low-dimensional vectors. The principle is to group together vector representations of words with similar meaning in the vector space. In other words, all words with similar meaning have a similar vector representation. Various word embedding models have been proposed such as *word2vec* and *Global Vectors for*

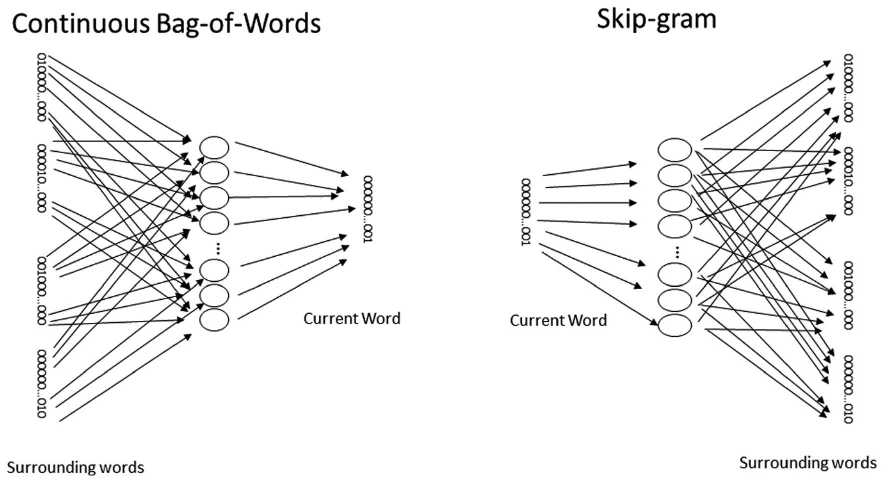


Figure 2.5: The Continuous Bag-Of-Words (CBOW) and the Skip Gram (SG) models [59].

word representation (GloVe). We review both of these models in the following of this section.

- **Word2Vec:**

  Word2vec method has been developed by Mikolov *et al.* [80] in 2013 at Google. This method uses shallow neural networks with three layers: an input layer, a hidden layer and an output layer, to learn the word representations based on their context. Two distinct models are used by Word2vec to learn word embeddings: *Continuous Bag-Of-words* (CBOW) and *Skip Gram* (SG) models. The CBOW model learns the representations by predicting the current word based on its surrounding context words, whereas the SG model learns the representations by predicting the surrounding context words based on the current word. A general framework of both models are represented in Figure 2.5. The Word2Vec model is described in more detail in chapter 6.

- **Global vectors for word representation:**

  Another well-known word embedding method proposed by Pennington *et al.* [85], is the Global Vectors for word representation (GloVe). The method leverages the global contexts in the form of global co-occurrence statistics for learning word embeddings. More specifically, the method combines the advantages of two commonly used models to learn word vector representations: the global matrix factorization model such as Latent Semantic Analysis (LSA) [43] and the local context window model such as Word2Vec [80]. The author claims that LSA (Latent Semantic Analysis) performs well in capturing significant statistical information using global statistics of word co-occurrence, but relatively fails in analogy tasks such as capturing semantic similarity between words. In contrast, word2vec performs better on analogy task, but fails to capture the corpus' global statistics. For example, Word2Vec doesn't recognize if two words occur together because one of them is very frequent, such as "the", or because there is a real semantic connection between the words.

Both popular models Word2vec and Glove have improved their performance in learning word embedding. However, these models perform better if they are trained from a huge dataset (corpus), the reason why researchers often opt for choosing pre-trained word embedding vectors [64]. Hence, these models tend to ignore all words that are outside their vocabulary (words that are missing from the trained model).

2.4.2 Text feature selection

Feature selection technique is an important task for vector dimensionality reduction in text classification [34]. The basic idea is to identify the most relevant text features and discard the redundant and the irrelevant ones. Consequently, feature selection often improves the accuracy of the classifiers which reduces the computation time.

In this section, we describe the most commonly used feature selection techniques in text classification. We use the following notations for the used formulas: for a set D of training data, $|D|$ is the number of documents in the training set. $P(t)$ is the probability that the term t occurs in a document, $\bar{t}$ represents the absence of the term t. $P(c)$ is the probability that a document belongs to the class c, $\bar{c}$ represents the non-membership in c. $P(t, c)$ is the probability of observing the term t in a document belonging to the class c.

- **Mutual information:**

  Mutual Information (MI) calculates the mutual dependency between two random variables. Formally, the MI between a term t and a class c is calculated as follows:

$$MI(t, c) = \sum_{t,c} P(t, c) \log \frac{P(t, c)}{P(t)P(c)}. \tag{2.5}$$

- **Chi-square:**

  Chi-square, noted also χ^2, is one of the most common method in text classification, it is a statistical metric that measures the lack of independence between a term t and a class c. In the case where a term t and a class c are completely independent, the Chi-square value is equal to zero, i.e., the number of documents containing a term t in predefined classes is equal. The expression for calculating Chi-square static is given as:

$$\chi^2(t, c) = \frac{|\mathcal{D}| \left(P(t, c)P(\bar{t}, \bar{c}) - P(t, \bar{c})P(\bar{t}, c)\right)^2}{P(t)P(\bar{t})P(c)P(\bar{c})}. \tag{2.6}$$

- **Odds ratio:**

  Odds Ratio (OR) is a feature selection method, initially proposed by [82] for binary text classification using naive Bayes algorithm. The method measures the ratio between the odds of a term occurring in one class and the odds of it occurring in another class. The OR is calculated as:

$$OR\left(t, c\right) = \frac{P\left(t, c\right)\left(1 - P\left(t, \bar{c}\right)\right)}{\left(1 - P\left(t, c\right)\right)P\left(t, \bar{c}\right)}. \tag{2.7}$$

  OR gives a score greater than zero for terms that are more probable in c, and a score less than zero for those who are less probable in c. The zero score is given for terms that are equally probable in all classes.

2.4.3 Topic modeling

Topic modeling is a statistical technique that can automatically extract latent topics or concepts from a given set of documents. The basic idea of this technique is that each document is considered as a mixture of topics and each topic as a distribution of terms [127]. More specifically, a topic is a set of terms that frequently co-occur in the documents. In other words, terms that are semantically related are mapped to the same topic [13, 83]. For example, the words: "patient", "doctor", "disease", "cancer" and "health" can be mapped to the topic "healthcare". Hence, terms or words with similar meaning will occur in similar text documents. Here, we describe the most popular topic modeling methods: *latent semantic analysis* [33], *probabilistic latent semantic analysis* [55] and *latent dirichlet allocation* [13].

- **Latent semantic analysis:**

  The first topic model was introduced by Deerwester *et al.* [33] in 1990, called Latent Semantic Analysis (LSA) or Latent Semantic Indexing (LSI). The LSA model consists of three main steps: the first step preprocesses the raw text data and converts it into a term-document matrix. In practice, the term-document matrix can be implemented using a term-weighting scheme like tf, $tf{-}idf$, etc. The second step performs matrix decomposition on the term-document matrix X to learn latent topics. More specifically, the LSA applies an algebraic method called Singular Value Decomposition (SVD) to project documents to a lower-dimensional semantic space. The final step generates a matrix that maps the documents according to their topics that came out of the SVD technique. As shown in Figure 2.6, the SVD of a $|V| \times c$ matrix X, where $|V|$ is the number of terms and c is the number of documents, is the product of three different matrices: W, Σ and a transpose of matrix C:

$$X = W \Sigma C^{\top}, \tag{2.8}$$

  where W and C are orthogonal matrices $W^T W = C^T C = I$. The matrix Σ is a diagonal matrix of singular values which are sorted in decreasing order, $\sigma_1 \geq \sigma_2 \geq$

$\cdots \geq \sigma_m$, where m is the number of linearly independent rows of the matrix X called the rank.

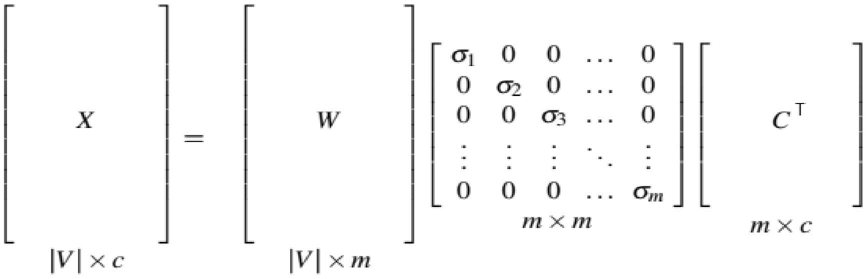


Figure 2.6: SVD applied to term-document matrix X .

For discovering latent themes, a truncated SVD method is used to generate a reduced-rank matrix approximation of X by selecting only the k biggest singular values in Σ, where $k \leq m$. In other words, the k most significant (biggest) singular values are kept, and the remaining (smallest) singular values are ignored (set to zero). Figure 2.7 shows a visualization of such a process. The matrix $\hat{X}$ of rank k that best approximates the term-document matrix X is given as:

$$\hat{X} = W_k \Sigma_k C_k^\top \approx X. \tag{2.9}$$

This dimensionality reduction enables the reduction of noise in the latent space by removing irrelevant dimensions. In addition, by limiting the number of dimensions to k, this allows to strengthen correspondence between words and contexts in order to improve the measure of similarity between documents[113].

- **Probabilistic latent semantic analysis:**
  Probabilistic Latent Semantic Analysis (pLSA) model, also called aspect model, was proposed by Hofmann [55] which tries to improve the LSA model by using a probabilistic method. Unlike the LSA, which applies the SVD method to the term-document co-occurrence matrix, the pLSA performs probabilistic mixture decomposition on the co-occurrence matrix using a generative latent class model (latent topics). Mainely, pLSA model associates an unobserved class variable (a latent

$$\underset{|V|\times c}{\begin{bmatrix} & & \\ & & \\ & X & \\ & & \\ & & \end{bmatrix}} = \underset{|V|\times k}{\begin{bmatrix} & & \\ & & \\ & W_k & \\ & & \\ & & \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k\times c}{\begin{bmatrix} & C_k^{\mathrm{T}} & \end{bmatrix}}$$

Figure 2.7: Reduced-rank SVD performed on term-document matrix X.

topic variable) $z \in Z = \{z_1, \ldots, z_K\}$ with each observation, i.e., the occurrence of each term t in a document d. Thus, the pLSA model for the term-document co-occurrence matrix can be expressed as a joint probability $p(d, w)$ between a term t and a document t:

$$P(d,t) = P(d) \sum_{z \in Z} P(t \mid z) P(z \mid d), \tag{2.10}$$

where $P\left(d\right)$ represents the probability of a term t occurring in a given document d, $P\left(t \mid z\right)$ denotes the probability of a term t conditioned on latent topic variable z, $P\left(z \mid d\right)$ is the probability distribution of a document d over the latent topics.

In the pLSA model, the conditional probabilities $P\left(t \mid z\right)$ and $P\left(z \mid d\right)$ are determined using the Expectation Maximization (EM) algorithm to maximize the following likelihood function:

$$\mathcal{L} = \sum_{d} \sum_{t} n(d,t) \log p(d,t), \tag{2.11}$$

where $n(d, w)$ denotes the number of occurrence of t in d.

The EM algorithm alternates between two steps: an Expectation step, denoted by E-Step, and a maximization step, denoted by M-step. The E-Step computes the posterior probabilities of the latent variables, i.e., the probability of topic z given the observed document d and term $t \in d$, according to the following equation:

$$P(z \mid t, d) = \frac{P(t \mid z) P(z \mid d)}{\sum_{z'} P\left(t \mid z'\right) P\left(z' \mid d\right)}. \tag{2.12}$$

The M-step updates the conditional probabilities $P(t|z)$ and $P(z|d)$ to maximize the likelihood function $\mathcal{L}$.

$$P(t \mid z)=\frac{\sum_{d} n(d, t) P(z \mid t, d)}{\sum_{t^{\prime}} \sum_{d} n\left(d, t^{\prime}\right) P\left(z \mid t^{\prime}, d\right)}. \tag{2.13}$$

$$P(z \mid d)=\frac{\sum_{t} n(d, t) P(z \mid t, d)}{\sum_{z^{\prime}} \sum_{t} n(d, t) P\left(z^{\prime} \mid t, d\right)}. \tag{2.14}$$

- **Latent Dirichlet allocation:**

  Latent Dirichlet Allocation (LDA) [13] extends the pLSA generative model to address some of its shortcomings. Like pLSA, LDA assumes that each document is a mixture of different topics. However, the authors have shown that pLSA suffers from overfitting [5] where the number of parameters increases linearly with the number of documents in the collection (training data). Another shortcoming that the authors have underlined is that pLSA model is not a generative model for new documents as the topic distribution is learned directly from the original data (training data). To address these issues, LDA model uses the Dirichlet distribution to model topics and words. In LDA model, the joint distribution of a topic mixture is given by:

$$P(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)=P(\theta \mid \alpha) \prod_{n=1}^{N} P\left(z_{n} \mid \theta\right) P\left(w_{n} \mid z_{n}, \beta\right), \tag{2.15}$$

  where t is a term of a given document d and z is the topic assignment to the term t, θ is the topic distribution for a document d. The two parameters α and β are the parameters of Dirichlet prior on the per-document topic distributions and the per-topic word distribution, respectively. Finally, N denotes the number of words or terms in a document.

2.5 Supervised machine learning algorithms

Formally, a supervised machine learning algorithm receives as input a set of data $D = \{(x^{(i)}, y^{(i)}) | (x^{(i)}, y^{(i)}) \in X \times Y\}_{i=1}^{n}$ and produces as result a hypothesis $h : X \rightarrow Y$. In this case, D corresponds to a set of n pairs of inputs $x^{(i)}$ and associated targets $y^{(i)}$. The elements of $X = \{x^{(1)}, \cdots, x^{(n)}\}$ are the n representations of examples we wish to study where each example $x \in X$ is described by a vector of m attributes (features) $x = (x_1, \cdots, x_m)$ and the elements of $Y = \{y^{(1)}, \cdots, y^{(n)}\}$ are values that we can associate with each example x. The hypothesis h built by a learning algorithm allows to associate an element $\hat{y}$ of the set Y with an element x of the set X absent in the set of learning examples:

$$\hat{y} = h(x). \tag{2.16}$$

The nature of the set Y depends on the type of problem to be solved. When the prediction values of Y belong to the set $\mathbb{R}$, it is a regression application, otherwise it is a classification application. For example, in spam filtering problem, the training data D consists of a certain number of pairs (x, y), where $x \in X$ is an email document represented by a vector of word occurrences or frequencies and $y \in Y$ is a response value assigned to email such as spam or ham class. In the case of classification, the system seeks for a function h able to automatically assign the right class spam or ham to a new email document. We would like this new classification to be most often identical to what we could have done manually. In the case of regression, the system can for example use score or weight to assign a spam ranking for a new email document. If the score exceeds a certain threshold, the email will likely end up in the spam folder. Thus, the input data is always discrete but the function sought is with real values.

In the rest of this chapter, we will limit ourselves to the problems of two-class discrimination $y \in \{-1, +1\}$, where one of the possible classes is represented by -1 and the other by $+1$. Much of the literature focuses on the binary case because the classifications involving more than 2 categories (multi-class) can always be re-expressed in the form of binary classifications. Indeed, if we consider a multi-class classification, we can always carry out an independent binary classification for each category and then look for which category the membership has manifested.

2.5.1 Naive Bayes

Bayesian classifier is a simple and effective probabilistic model based on Bayes' theorem. The theorem provides a way of calculating a conditional probability $p(y|x)$ (shown in Equation 2.17), called a posterior probability. More specifically, the Bayesian classifier assigns the most likely class $y \in Y$ to a given example $x \in X$ described by its feature vector $x = (x_1, \cdots, x_m)$. The probability that an example x belongs to a class y is given as follows:

$$p(y|x) = \frac{p(y)p(x|y)}{p(x)}. \tag{2.17}$$

- $p(x|y)$ is the probability of generating example x given class y.
- $p(y)$ is the probability of occurrence of class y.
- $p(x)$ is the probability of occurrence of example x.

The *naive* hypothesis of Bayes' theorem comes into play when we assume the independence of features. Therefore, this hypothesis allows to write:

$$p(x|y) = p(x_1, \cdots, x_m|y) = \prod_{i=1}^{m} p(x_i|y). \tag{2.18}$$

The probability $p(x_i|y)$ is the ratio between the number of times the attribute x_i appears in class y and the total number of examples that include class y. In practice, it is possible to simplify the naive Bayes formula by ignoring $p(x)$ because this parameter remains the same for each class. This gives us the following formulation:

$$p(y|x) \propto p(y) \prod_{i=1}^{m} p(x_i|y). \tag{2.19}$$

This calculation is done for each class $y \in Y$, and we consider the highest probability to select the class of the example that we want to classify. Therefore, the decision rule that maximizes this probability is given by the following equation:

$$\hat{y} \propto \operatorname*{argmax}_{y \in Y} \; p(y) \prod_{i=1}^{m} p(x_i|y). \tag{2.20}$$

2.5.2 Support vector machine

The Support Vector Machine (SVM) is an efficient machine learning algorithm based on geometric interpretations. The algorithm uses three main concepts, namely: optimal hyperplane, maximum margin and support vectors. The examples closest to the separation boundary of the two classes $y = -1$ and $y = +1$ shown in the scheme of Figure 2.8 are called support vectors. The margin is the distance between the support vectors on either side of the separation boundary. The main objective of SVM is to look for the maximum margin to find an optimal hyperplane which gives a better separation of the learning examples.

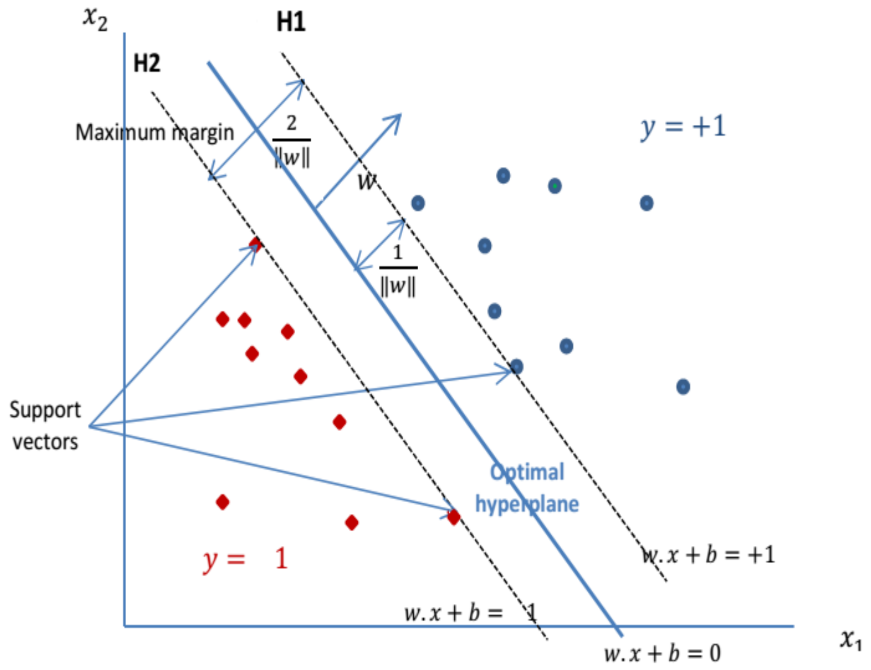


Figure 2.8: Classification of data by Support Vector Machine (SVM).

In the case of linear SVM, for a set of n training data (examples) labelled into two classes $y \in \{+1, -1\}$. The problem consists of finding a hyperplane $w \cdot x + b = 0$ which separates the classes with the largest margin, where $w \in \mathbb{R}^m$, $b \in \mathbb{R}$ and $w \cdot x$ is the scalar product of the two vectors w and x. The search for the maximum margin for determining the w and b parameters of the hyperplane leads to a quadratic optimization problem. We are looking for a point that minimizes or maximizes a certain function subject to certain constraints. The discriminant function h is obtained by a linear combination of an input vector x and it is written as follows:

$$h(x) = w \cdot x + b. \tag{2.21}$$

The class being given by the sign of $h(x)$:

$$\hat{y} = sign(h(x)) = \begin{cases} +1 \quad if \quad h(x) \geq 0, \\ -1 \quad if \quad h(x) < 0 \end{cases} \tag{2.22}$$

The separating hyperplane is then defined by the equation: $w \cdot x + b = 0$. Let $(x^{(i)}, y^{(i)})$ be one of the n couples of the learning set D, the goal is to find the classifier h such that:

$$y^{(i)}(w \cdot x^{(i)} + b) \geq 0. \tag{2.23}$$

As we mentioned earlier, the SVM approach seeks to find the optimal hyperplane among the set of possible hyperplanes, thus allowing to correctly classify the data. Referring to Figure 2.8, we note that on the two hyperplanes $H1$ and $H2$ parallel to the optimal hyperplane $w \cdot x + b = 0$, we have the support vectors whose respective equations are $w \cdot x + b = -1$ and $w \cdot x + b = +1$. Thus, in this case, we cannot find training examples located in the margin because they should satisfy the following constraints:

$$w \cdot x^{(i)} + b \geq +1 \quad if \quad y^{(i)} = +1 \tag{2.24}$$

$$w \cdot x^{(i)} + b \leq -1 \quad if \quad y^{(i)} = -1 \tag{2.25}$$

It is possible to combine the two constraints 2.24 and 2.25 into a single inequality as follows:

$$y^{(i)}(w \cdot x^{(i)} + b) - 1 \geq 0 \tag{2.26}$$

In vector geometry, the margin is equal to $\frac{1}{\|w\|}$ To find the optimal hyperplane which maximizes the margin, we must determine the vector w which has the minimum Euclidean norm $min\|w\|$ equivalent to $min\ \frac{1}{2}\|w\|^2$ which checks the constraint of equation 2.26

of good classification of training examples. The optimal separator hyperplane can be obtained by solving the equation:

$$\begin{cases} min\ \frac{1}{2}\|w\|^2 \\ \\ y^{(i)}(w \cdot x^{(i)} + b) - 1 \geq 0 \quad \forall_{i=1,\cdots,n} \end{cases} \tag{2.27}$$

In this type of problem, to calculate the variables w and b, we could use suitable optimization methods like the principle of duality and the Lagrange multipliers to show that the vector w^*, which achieves the optimum, can be written in the form:

$$w^* = \sum_{i=1}^{n} \alpha_i^* y^{(i)} x^{(i)}, \tag{2.28}$$

where α_i^* are the Lagrange multipliers, they are non-zero only for the examples $x^{(i)}$ lying exactly on the border of the margin, that is to say the support vectors. Let $vs = \{j \in \{1, \cdots, l\} |\ \alpha_j^* \neq 0\}$ be the set of indices of the support vectors. The decision rule for a new observation x based on the hyperplane with maximum margin is given by:

$$\hat{y} = sign(\sum_{j \in vs} \alpha_j^* y^{(j)} x \cdot x^{(j)} + b^*). \tag{2.29}$$

The calculation of b^* can be performed from any support vector by the equation:

$$b^* = y^{(i)} - w \cdot x^{(i)}. \tag{2.30}$$

In terms of precision, we can use the average over all the support vectors to have a robust value of b^*.

The SVM classifier presented above assumes that the data of the two classes are linearly separable, which is not always the case in practice. The extension of SVM to the non-linear case uses the kernel trick that projects the data into a new higher-dimensional space called a representation space or feature space and then applies a linear SVM in this new space.

2.5.3 Logistic regression

As with Bayesian classifier, Logistic regression is based on a probabilistic interpretation since its final decision is based on the posterior probability of class y. In general, the algorithm assumes that the logarithm of the posterior probability can be described as a linear function of the feature vector $x = (x_1, \cdots, x_m)$. Thus, the posterior probability $p(y|x)$ is represented by sigmoid function acting on the feature vector x.

$$p(y|x) = \frac{1}{1 + e^{-(w \cdot x + b)}}, \tag{2.31}$$

where, $w \cdot x + b = w_1 x_1 + \cdots + w_m x_m + b$. This implies that $(b, w_1, \cdots, w_m)$ represents the vector which defines a hyperplane that separates the two classes -1 and $+1$. The result sign indicating the class to be assigned to a new observation is represented as follows:

$$\hat{y} = sign(w \cdot x + b) = \begin{cases} +1 & si \quad w \cdot x + b \geq 0, \\ -1 & si \quad w \cdot x + b < 0 \end{cases} \tag{2.32}$$

This hypothesis formulation avoids the simulation of the generation process of the example x as would the Bayes algorithm and allows a direct evaluation of the posterior probability of the class y. The training of this algorithm consists of determining the parameters w and b which are estimated from the training data.

2.5.4 Decision trees

Classifier algorithm based on decision tree such as ID3 [89], C4.5 [90], RIPPER [24] and CART [71] are made up of a set of rules making it possible to classify a set of data into homogeneous groups. Each rule associates a conjunction of tests on descriptive variables. The top of a decision tree represents the root, the variables corresponding to non-terminal nodes (tests on features) are classification variables, each branch corresponds to a modality of the variable (response to a test) considered at this level of the tree and leaves (terminal nodes) represent labeling classes. For example, in the case of binary tests, one of the branches corresponds to a positive response to the test and the other branch to a negative response. This process is repeated on each node of the tree, the nodes which are not pure are segmented until pure leaves are obtained.

Decision trees construction requires:

1. a good choice of the classification variable on a node.

2. a stop criterion for the learning algorithm. This requires:
    - predefine a proportion threshold of examples of a class in a node to avoid over-fitting;
    - set a homogeneity threshold below which we refuse to split a vertex;
    - reach a pure node.
3. an optimal decision rule when a leaf is not pure: generally, we label the current node by the majority class.

In a given node, the measure used to select the best attribute to be tested should allow us to search among the different attributes of the learning examples, which one has the greatest factor of discrimination for class distribution. Three popular measures for choosing the best attribute are: Information gain [89, 90], Gain ratio [71], Gini index [118] and Chi-square criterion [72].

Once a decision tree is built, its size can be very important, which can deplete computing and storage resources. In addition, the tree may have a high error rate due to over-fitting; that is, an ability to perfectly describe training examples while having weak capacities for generalizing or predicting new examples. Generally, over-fitting can occur when the learning set contains noisy data, too little data, or too specific data. To overcome these problems, pruning operations involve removing branches from the tree that reduce error rates.

Several pruning techniques [81] have been proposed to avoid over-fitting in a decision tree. These can be categorized into two main approaches: Pre-pruning and post-pruning. The pre-pruning approach controls the growth of a decision tree during its development (i.e., we decide to stop the building of our tree before it is fully grown). The post-pruning approach involves growing an entire decision tree in its entirety, then pruning the tree's nodes in a bottom-up fashion.

2.5.5 K-nearest neighbor

K-Nearest Neighbors (KNN) classifier is a simple but a powerful supervised machine learning algorithm, it assumes that similar examples are near each other. The kNN principle consists of determining for each new example to be classified, the list of k nearest neighbors among the examples already classified (where k is an integer). The new example is assigned to the class among its k nearest neighbors. A particular case is when k=1, the new example, in this case, is assigned to the class of its single nearest neighbor.

When we speak about neighbor it implies the notion of distance or dissimilarity. The most commonly used distance measure is Euclidean distance. Let $dis(x^{(1)}, x^{(2)})$ be the Euclidean distance between two vectors $x^{(1)} = (x_1^{(1)}, \cdots, x_m^{(1)})$ and $x^{(2)} = (x_1^{(2)}, \cdots, x_m^{(2)})$, which is computed as follows:

$$dis(x^{(1)}, x^{(2)}) = \sqrt{\textstyle\sum_{i=1}^{m}(x_i^{(1)} - x_i^{(2)})^2}. \tag{2.33}$$

The algorithm calculates the distance between each train and test data (new example) point and then select the top nearest according to the value of k. For KNN, the choice of the number of neighbors k is very crucial because the classification results change according to this value. For example, in Figure 2.9, we can see the effect of the choice of k on the final classification result. The new example to predict (noted "?" In green circle) could be classified either in the class of blue squares or the class of red triangles. Indeed, if k = 3, it is assigned to the class of red triangles because there are two triangles and only one square in the circle indicated in a continuous line. If k = 5, it is assigned to the first class because there are more squares than triangles in the circle indicated in a dotted line.

2.5.6 Adaboost (Adaptative Boosting):

Adaboost or Adaptative Boosting is a type of "Ensemble Learning" where multiple learners (known as "weak learners") are employed to build a stronger learning algorithm. AdaBoost works by choosing a base algorithm (e.g., decision trees) and iteratively improving it by accounting for the incorrectly classified examples in the training set. Initially, the

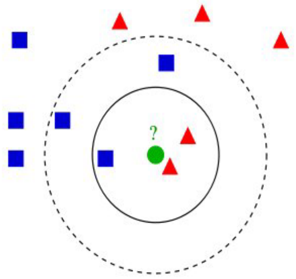


Figure 2.9: k-Nearest Neighbor classification.

algorithm assigns equal weights to all the training examples and chooses a base algorithm. At each step of iteration, it applies the base algorithm to the training set and increases the weights of the incorrectly classified examples. It iterates n times, each time applying base learner on the training set with updated weights. The final model is the weighted sum of the n learners.

2.5.7 Random forests

Random forests are machine learning algorithms based on decision trees. When given a set of class-labeled data, random forests build a set of classification trees. Each tree is developed from a bootstrap sample from the training data. When developing individual trees, an arbitrary subset of attributes is drawn (hence the term "random"), from which the best attribute for the split is selected. Classification is based on the majority vote from individually developed tree classifiers in the forest. Figure 2.10 shows a general random forest architecture [77].

The bootstrap method generates n training samples. These samples learn a set of weak classifiers (C trees). The majority vote on the learned classifier outputs represents the classifier's final prediction [39].

2.5.8 Artificial Neural Networks

Artificial Neural Networks (ANN) are inspired by the human brain to build classification models. The goal is to have a network of artificial neurons (represented by nodes) able to

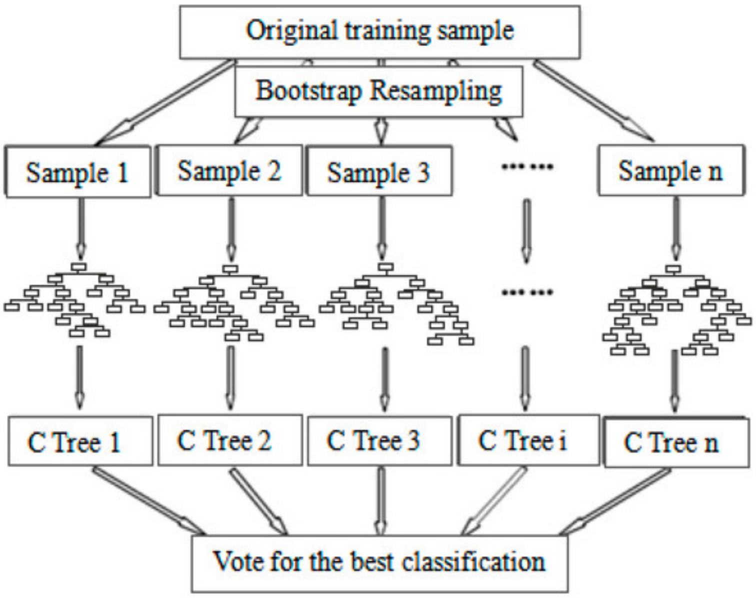


Figure 2.10: Random forest classifier.

learn over time and identify different features using knowledge obtained from historical data. Basically, the ANN contains three connected layers, including the input layer, one or more hidden layer(s), and the output layer, as shown in Figure 2.11. The output of the input layer is the input of the hidden layer and the output of the last layer is the result. The connections between the layers are represented by specific weights adjusted over time to get more accurate results. In other words, the weights are adjusted if the evaluated output is different from the desired output. This process is repeated until the desired results are obtained. ANN nodes apply an active function such as sigmoid or the hyperbolic tangent functions to get the result.

2.6 Classification evaluation metrics

Given the large number of classification algorithms proposed in the literature, it is necessary to compare them and see which one responds best to a specific task. There are many

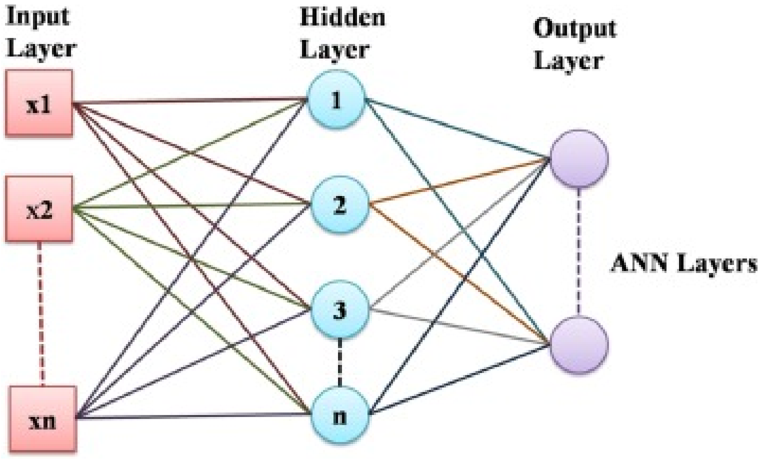


Figure 2.11: Artificial Neural Networks.

evaluation metrics used to test the performance of the classification algorithms [40]. Some of the most common metrics are: accuracy, precision, recall and F1-measure. In general, all of these measures are built from a confusion matrix (contingency table). Table 2.1 illustrates the confusion matrix of a binary classification. The row of the matrix states the actual class, while the column states the predicted class.

	Predicted class	
Actual class	Spam	Ham
Spam	TP	FN
Ham	FP	TN

Table 2.1: Confusion matrix of a binary classification.

Here, we consider the case of email filtering with two classes spam (the positive class) and ham (the negative class). The ham and spam labels in the rows of the table represent the emails' actual classes, and those in the columns represent the classifier's decision classes. The meaning of the types of boxes in the contingency table are represented as follows:

- *TP (True Positive):* it represents the number of emails whose actual class is spam and predicted class is spam.

- *TN (True Negative):* it represents the number of emails whose actual class is ham and predicted class is ham.

- *FP (False Positive):* it represents the number of emails whose actual class is ham and predicted class is spam.

- *FN (False Negative):* it represents the number of emails whose actual class is spam and predicted class is ham.

From the statistics of the confusion matrix resulting from a classification task, the evaluation measures: accuracy, precision, recall and F1-measure are defined and calculated as follows:

- **Accuracy:** it measures the proportion of correct predictions over the total evaluated documents(emails).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.34}$$

- **Precision:** it measures the proportion of correctly predicted positive documents over the total predicted documents in a positive class.

$$\text{Precision } = \frac{TP}{TP + FP} \tag{2.35}$$

- **Recall / Sensitivity:** it measures the proportion of correctly predicted positive documents over the total positive documents.

$$\text{Recall/Sensitivity} = \frac{TP}{TP + FN} \tag{2.36}$$

- **F1-measure:** it measures the harmonic mean between recall and precision.

$$\text{F1-measure} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.37}$$

2.7 Conclusion

The task of text classification, also known as text categorization, is considered as one of the most popular tasks in NLP. The purpose of text classification is to use the labeled or unlabeled training data (corpus) to build models that can automatically predict the relevant label (class or category) of new data. The process of structuring textual data into one or more class labels is based on the following steps: feature extraction, feature selection, machine learning algorithms and evaluation measures. Feature extraction is the process of converting text documents into a feature space, such as term weighting (e.g., $tf{-}idf$, tf, etc.) and word-embedding models (e.g., Word2Vec, GloVe, etc.). Feature selection is the process of reducing the feature vector dimensionality, such as MI, Chi-square, Odds ratio, etc. Another type of dimensionality reduction technique is the topic modeling which maps document keywords into a small number of topics, such as LSA, pLSA, LDA, etc. Machine learning algorithms are used to create text classification models, such as the logistic regression, naive Bayes, KNN, SVM, etc. Finally, evaluation measures are used to test the performance of the classifiers, such as accuracy, precision, recall, etc. Text classification techniques have been adopted in spam filtering systems to distinguish between spam and ham emails. The following chapter reviews the different spam filtering methods based on automatic text classification methods.

3

Literature review

3.1 Introduction

Various spam filtering methods have been proposed in the literature to deal with the problem of spam [10, 17, 61]. Most of these methods have some success for filtering specific spam, but fail to solve the problem efficiently. This issue is due to the fact that spammers are using increasingly sophisticated attack techniques. In this chapter, we review some well-known work in the field of spam detection. We also compare the strengths and drawbacks of existing methods in spam filtering. These methods are generally categorized into two main approaches, namely non-content and content-based features. There are also combined methods that take advantage of both techniques to improve the accuracy of spam filters. Such methods are usually called hybrid approaches. In the following sections, we provide a description of each approach and its related work.

3.2 Non-content based approaches

Non-content-based features, also called header-based features, refer to the use of email headers, related sender information, and general characteristics of the message for detecting spam. For instance, all information related to sender address domain (IP address), sender reputation, writing style, sending time, type of attachment, message size, etc., are explored in spam detection problem. This part briefly reviews the important contributions done by the researchers using non-content based spam filtering approaches.

3.2.1 List-based filters

Filters based on list analysis block or allow the delivery of emails sent from specific senders. The emails are then blocked and considered as spam if the senders are categorized as spammers and they are allowed and considered as ham if the senders are categorized as trusted users. Commonly used techniques within this category are: *Whitelist*, *Blacklist* and *Greylist*, which can be considered as signatures for spam identification [26, 54, 61].

- **Whitelist:** the Whitelist method is a list of trusted senders, domains or IP addresses from which users tend to receive emails, and all other addresses out of this list are considered as spam. This method is very strict and may introduce a high false-positive rate. It can be useful for some internal company addresses or extremely private email addresses, but it needs to be updated regularly. Furthermore, spammers may easily use spoofed addresses if they can identify or guess the Whitelisted addresses to fool spam filters [15, 26]. The user can maintain his Whitelist using his mailbox tools or automated tools such as auto-whitelisting of SpamAssassin [110]. The automated tools can deduce the Whitelist from email traffic based on the history of legitimate emails. The false-positive rate of the Whitelist technique can be significantly reduced if it cooperates with other anti-spam techniques [15].
- **Blacklist:** unlike methods based on the Whitelist, the Blacklist methods can block and reject unsolicited emails based on senders, domains or IP addresses, known as spammer addresses [115]. Some of the existing popular Blacklist methods include:

auto-blacklisting of SpamAssassin[110], SpamHaus [1], etc. However, this method can misidentify legitimate senders and increase the false-positive rate if a spammer sends spam from an IP address that legitimate email users also use. In addition, as the Whitelist, Blacklist filters also need to be updated frequently, which can be costly in terms of resources and time for email servers. For example, spammers use botnets by examining a large number of IP addresses before putting them on the Blacklist. According to the authors in [115], the effectiveness of a Blacklist method depends on its completeness, accuracy, and spammer inability to spoof addresses that are not on the Blacklist.

- **Greylist:** the Greylist method initially rejects emails received from unknown senders, domains or IP addresses and returns temporary error messages to the senders that invite them to resend the emails. The error messages indicate to the sender servers to wait and try again to send later. This method assumes that legitimate senders are more likely to resend the emails after delay, while many spammers usually send once and would not retry. This technique has the advantage of being very easy to implement and low-costly in terms of human and physical resources. However, the process of the Greylist method can be very annoying to users as it may delay mail delivery. Currently, the evolution of Web technology has reduced the effectiveness of this method as spammers can use "zombie computers" to resend the spam email [15, 61].

3.2.2 Temporal features analysis

The basic idea is to extract useful features from the temporal information usually present in the header part of the emails. Hao *et al.* [51] built the Spatio-temporal Network-level Automatic Reputation Engine (SNARE) system to analyze the email sender reputation based on statistical features at network-level. The authors combine various spatial and temporal features of email senders. Temporal features such as the time of a day when the message was sent are considered. Spatial features can be the geographical distance between the sender and the receiver. The authors demonstrate that the SNARE system can achieve comparable accuracy to existing static IP blacklists. They also showed that the system can cooperate with other filtering systems to yields a higher performance for spam detection.

Qian Xu *et al.* [125] proposed a method based on three types of features namely, static, temporal and network features and incorporated these features into an SVM classification algorithm. The static features include the number of messages and message size. Temporal features such as the size of messages during a day and on each day of the week. Finally, network features include the number of recipients and clustering coefficient. The evaluation results using AUC (Area Under Curve) metric showed that the methods based on temporal features and network features can be effective compared to those that are only based on conventional static features.

In general, systems based on temporal information analysis for email spam detection cannot replace content-based analysis systems. However, they can be useful as an intermediate filter like Blacklist and Whitelist methods to enhance the classification accuracy [51].

3.2.3 SMTP path analysis

SMTP path analysis is a filtering method based on the reputation of IP addresses. The method learns if IP addresses are spam or not by analyzing the history of emails sent using that IP address [67].

Leiba *et al.* [68] presented an algorithm to learn the reputation of IP addresses and email domains by analyzing the paths used to transmit known spam or ham emails. This method considers only the IP addresses contained in the "*received*" lines of the email headers to train the used classification algorithm. The basic idea of this technique is that an email from the same or similar IP addresses is likely to share the same classification. The authors demonstrated that the combination of the SMTP path analysis method with a Bayesian filter increases the accuracy of the spam filter.

Yu Jiang *et al.* [57] developed Email Geographic Path Analysis (EGPA) technique for spam filtering. In this method, the authors first build an email path using route information and then determine the geographic location of each node in the path. To detect spam, the filter checks the geographic information deviation of nodes in the path. The performance of the EGPA method has been tested using email traffics captured from one backbone link that crosses the geographic boundary of China. Experimental results show a reduction of 13.9% in spam email traffic.

3.2.4 Behavior analysis

Spammers have typical behaviors that distinguish them from legitimate senders. For example, spam email can be sent with an anonymous username, transmitted without permission, sent with illegal accounts, delivered in bulk repeatedly without authorization and to different recipients, etc. Wu *et al.* [122] claim that such behavior can be analyzed to identify spam emails. They built a spam filtering model by observing the transactions of sending messages and identifying different behaviors applied to send a message. The behavior features are obtained by analyzing header and syslog parts of messages. On the header part, they analyzed 6 fields: `Received`, *Return-Path*, *From*, *Delivered-To*, *To* and *Date*. On the syslog part, they analyzed 3 fields: *from*, *to*, *nrcpts* and *date*. A total of 26 features extracted from these fields are used for learning neural network classifiers to discriminate between typical spam behavior and legitimate messages. Thus, the performance of the system is compared with some existing content-based methods, it achieves best results in terms of false positive and false negative rates.

Qaroush *et al.* [88] studied the information contained in the email header and proposed a method based on statistical header features and sender behavior to detect spam email. The statistical information represents a total of 48 features extracted from the most appearance fields in email header which are: *From*, *To*, *CC*, *Received*, *Return-Path*, *Date*, *Reply-To*, *Error-To*, *Sender*, *References*, *In-Reply-To* and *Message-ID*. The sender behavior uses a "Trust" feature obtained by analyzing *From* field of email header. This feature can take one of the following values: strongly spam, weakly spam, weakly ham, or strongly ham. The value of the trust feature depends on the sender's reputation which can be achieved based on his historical behavior. The idea is to check if a sender sent a significant amount of spam emails in the past, then the sender is more likely to be a spammer. In contrast, if most of the sent messages were ham, then the sender is more likely to be a legitimate email user rather than a spammer. Finally, by combining the statistical header features and trust value together, the experimental results show that among different used machine learning techniques, random forest has achieved the best performance.

3.2.5 Social network analysis

Social network analysis method is a strategy for investigating social structures using networks and graph theory [84]. The method is considered very useful for discovering the relationships among a group of people and judging the trustworthiness of outsiders. Hence, various anti-spam filters have adopted this technique to exploit the knowledge embedded in the social network interactions to distinguish between spam and ham.

Hu *et al.* [56] introduced new metrics found to be efficient to detect the compromised email accounts from the perspective of graph topology. More specifically, the authors adapted the widely used social network analysis metrics to detect compromised email accounts. The used metrics are as follows: Success Outdegree Proportion, Reverse Pagerank, Recipient Clustering Coefficient and Legitimate Recipient Proportion. The authors used and adapted these metrics according to the features of mail log analysis. For the social network graph construction, the authors built a directed multigraph where each node corresponds to email account and each edge corresponds to entry in the mail logs. Each mail log entry contains the following metadata: *date*, *time*, *from*, *to*, *rcpttype*, result. The "*from*" field is considered as source node and the "*to*" field as target node. Evaluation results demonstrate that the approach can detect identity spoofing on email accounts with false-positive rates between 20-40%. However, in reality many organizations receive a considerable amount of emails each day, so a false positive rate of 20% would result in a big number of false alerts every day.

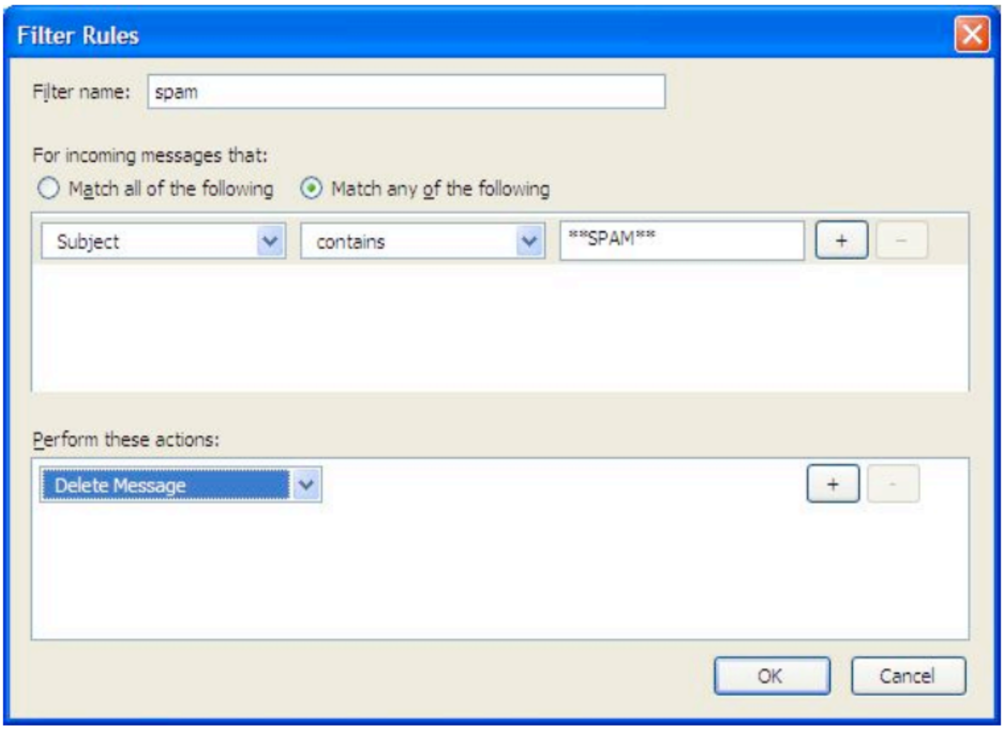
3.3 Content-based approaches

Content-based spam filtering approaches are among the most popular filters for spam email detection[31]. These approaches analyze the email content and extract a set of features from the email's subject and body parts. This kind of filter can be manually built using a set of hand-made rules, often called heuristic filters. It can also be built by using machine learning algorithms applied to a set of categorized messages spam or ham. In this section, we cover the most relevant approaches to content-based spam filtering.

3.3.1 Rule-based approaches

Heuristic or rule-based spam Filtering approaches are among the earliest proposed solutions for spam detection [10, 26]. Filters based on these approaches are also called *Hand-Crafted rules* or *Knowledge Engineering*. The principle of these techniques consists of using a set of hand-coded rules. These methods use a set of logical rules to detect emails containing specific keywords, sentences or suspicious patterns such as words containing punctuation symbols (e.g., Money back!, f*r*e*e, etc.) [26]. The rules are usually encoded using compact regular expressions to specify complex text patterns [86]. Commonly, the set of rules is maintained by a community of expert users and administrators [100]. These techniques have been used, for example, in the SpamAssassin system [110] and by some other researchers such as Sahami *et al.* [95] and Schleimer *et al.* [101]. These types of approaches are direct and require no learning. However, the set of rules needs to be regularly updated to maintain the effectiveness of the approaches.

The simplest method of rule-based spam filtering uses only a list of blacklisted keywords or phrases such as: "Viagra", "BILLIONS OF DOLLARS!!", "MAKE MONEY!!!", etc. Most modern mailboxes allow writing user rules through simple forms to adapt them to their needs. For instance, a *Thunderbird straightforward* spam filter illustrated in Figure 3.1 shows that the user has created a filter named "*spam*" to delete all messages (sent to trash) in which the word "***spam***" occurs in the Subject part of a message. However, these filters are not very effective and poorly designed. They are not very effective as they have weak robustness to masked words such as the word "Vi@gr@" and they are



Filter name: spam

For incoming messages that:

○ Match all of the following ● Match any of the following

Subject contains **SPAM**

Perform these actions:

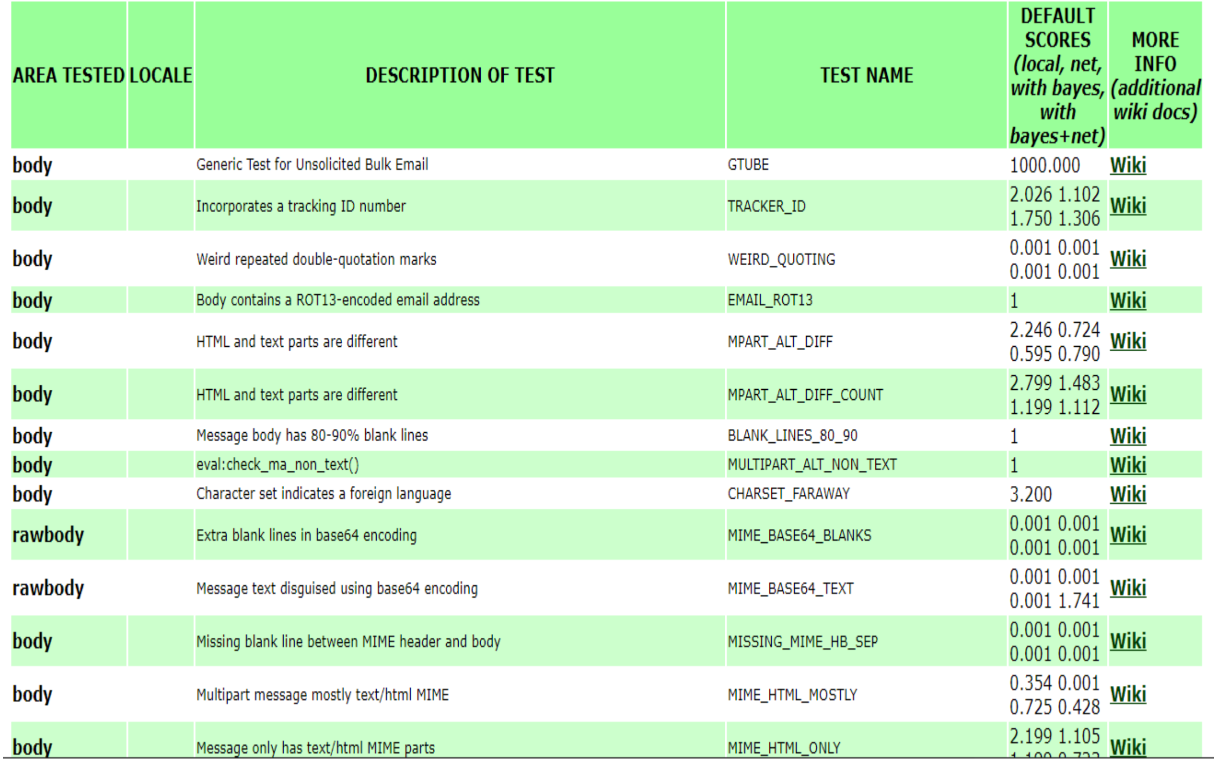
Delete Message

Figure 3.1: A simple rule-based spam filter coded as a Thunderbird mail client rule.

poorly designed as they cannot deal with the Scunthorpe problem such as detecting the word "cialis" in the word "specialist".

Shortly afterward, keyword-based filters became a little more relevant with the use of regular expressions. These filters use a set of patterns applied to a character chain to determine the different variations of keywords (e.g., free!, f r e e, f*r*e*e, etc.) often used by spammers. Meanwhile, spammers have continued to invent increasingly sophisticated techniques to escape the filters. Therefore, system administrators saw the need to integrate other heuristic rules to improve the classification error rate such as: length of words and sentences, presence of HTML code (Hypertext Markup Language), presence of images, etc. Usually, these rules are designed and modified in response to threats and specific user needs [100].

A few years later, in 1999, the open-source SpamAssassin, currently maintained by the Apache foundation, has emerged. SpamAssassin is considered as one of the best examples of rule-based spam filters. The rules of SpamAssassin (also named "tests") are used to



AREA TESTED	LOCALE	DESCRIPTION OF TEST	TEST NAME	DEFAULT SCORES *(local, net, with bayes, with bayes+net)*	MORE INFO *(additional wiki docs)*
body		Generic Test for Unsolicited Bulk Email	GTUBE	1000.000	Wiki
body		Incorporates a tracking ID number	TRACKER_ID	2.026 1.102 1.750 1.306	Wiki
body		Weird repeated double-quotation marks	WEIRD_QUOTING	0.001 0.001 0.001 0.001	Wiki
body		Body contains a ROT13-encoded email address	EMAIL_ROT13	1	Wiki
body		HTML and text parts are different	MPART_ALT_DIFF	2.246 0.724 0.595 0.790	Wiki
body		HTML and text parts are different	MPART_ALT_DIFF_COUNT	2.799 1.483 1.199 1.112	Wiki
body		Message body has 80-90% blank lines	BLANK_LINES_80_90	1	Wiki
body		eval:check_ma_non_text()	MULTIPART_ALT_NON_TEXT	1	Wiki
body		Character set indicates a foreign language	CHARSET_FARAWAY	3.200	Wiki
rawbody		Extra blank lines in base64 encoding	MIME_BASE64_BLANKS	0.001 0.001 0.001 0.001	Wiki
rawbody		Message text disguised using base64 encoding	MIME_BASE64_TEXT	0.001 0.001 0.001 1.741	Wiki
body		Missing blank line between MIME header and body	MISSING_MIME_HB_SEP	0.001 0.001 0.001 0.001	Wiki
body		Multipart message mostly text/html MIME	MIME_HTML_MOSTLY	0.354 0.001 0.725 0.428	Wiki
body		Message only has text/html MIME parts	MIME_HTML_ONLY	2.199 1.105	Wiki

Figure 3.2: A part of SpamAssassin rule list.

check several parameters on email headers and body text to identify spam signatures. The current list of tests (rules) of the version 3.x has over 1000 tests. Figure 3.2 shows a small part of the list of tests as they are illustrated in the project Web page. Each individual rule is scored, typically between ±0.01 and ±0.5. The filter assigns a total score to the incoming email which is the sum scores of all rules satisfied by that email. The email is labeled as spam if the total score reaches or exceeds the spam threshold set by the administrator or the user (the standard score is set at 5.0). The initial associated score to each rule is defined using a neural network trained with error backpropagation.

One of the major problems with rule-based spam filtering is the divulgation and the sharing of these rules on open source sites. Consequently, spammers are always able to check whether their messages can avoid these types of filters [26]. This problem leads to the production of false positives and the rejection of legitimate messages, while most users prefer to receive spam rather than losing legitimate email. Furthermore, as mentioned above, the set of rules needs to be updated regularly to maintain the effectiveness of the filters. Unfortunately, this limits the applicability of the method in the context of dynamically evolving spam content, which requires developing techniques for automatic rule generation.

3.3.2 Vector-space-based approaches

With recent advances in machine learning, a significant effort has been made for developing automatic approaches for spam detection [10, 12, 17, 30]. In this context, several classification-based methods have demonstrated their effectiveness for spam detection using techniques such as naive Bayes [58, 104, 116], Decision trees [86] and Support Vector Machines (SVM) [104, 112]. These methods cast spam filtering as a classical text categorization problem [17], where a set of relevant features (e.g., words, sentences) are extracted from email messages using Natural Language Processing (NLP) techniques. The information extracted from email content is typically represented as a feature vector and used as input data for machine learning algorithms. Generally, vector-space-based approaches perform two main steps: feature extraction and feature selection. In feature extraction, as mentioned earlier, the information contained in the email content is extracted and displayed in a unified form. In feature selection, only the relevant information is chosen to improve the performance of machine learning algorithms. Several models have been proposed in the literature to represent the email content as a vector of features. We present below the most popular models.

- **Bag of Words (BoW):** the BoW is considered as the most common model for feature representation in spam filtering area [17, 95]. This model is the simplest representation of text documents, it describes the text as a set of unstructured and independent terms (words) where each term represents a separate dimension in the vector space used for email classification. For example, the authors in [42] proposed a method based on the BoW model to classify emails to spam or ham. The proposed method is based on four steps including: data pre-processing, Vector Space Model (VSM) for email representation, feature selection, and email classification.

  In the preprocessing step, the features are first extracted from email content using subject and body parts. This is followed by applying a stop-word list and a stemming procedure to remove the irrelevant and redundant features to reduce the vocabulary size. In the vector space model step, the approach uses the BoW model to represent each email as a vector of words (terms). The authors used term frequency (tf) for assigning weights to the terms, it is equal to the number of occurrences of the terms in each document. In the feature selection step, each term in the vector space model is evaluated using the Mutual Information (MI) measure. This measure has the ability to measure the degree of association between the feature and the corresponding class. In the classification step, six different classifiers are evaluated in the experimentation, namely: Naive Bayes, Bagging, Logistic Regression, Decision Tree, J4.8, and Adaboost.

M. Basavaraju *et al.* [9] proposed a text clustering technique based on the VSM for spam detection. The authors used the BoW model to extract features from the text of email documents. First, Porters stemming and stopping algorithms are used in this work for text preprocessing to reduce vocabulary size and help in information retrieval and classification. Next, $tf-idf$ (term frequency-inverse document frequency) model is chosen as weighting scheme to evaluate the importance of words in email documents. Then, the algorithms: Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) or K-means are used to cluster email documents. Finally, the Nearest Neighbor Classifier (NNC) or K-Nearest Neighbor Classifier (K-NNC) algorithms are used to classify test documents.

Several other works based on the BoW model for spam detection have been proposed in the literature [6, 18, 35, 63, 76, 108]. Feature selection methods such as information gain [18], term frequency variance [63] and Chi Square[35] are some metrics used in these works to determine more significant features that enable learning classifiers to identify spam emails. According to the literature review, the Information Gain is the most widely used and the most effective feature selection metric in spam filtering domain [49, 76]. For more detail about the feature selection strategies in VSM refer to [76, 107, 108].

The BoW model representation has proven good efficiency compared to other methods relying only on network analysis or Blacklists [109]. However, this model has also its limitations since textual features that are extracted independently can usually miss features correlation and semantic content description of the email. In addition, this representation often yields to a high-dimensional sparse matrix and may cause a waste of computational and storage resources because each word in the document is treated as an attribute in the feature space.

- **N-Gram:** to mitigate the issue of semantic representation in the BoW model, the use of n-grams extends this model by considering sequences of n words occurring in the text as basic tokens to discriminate between spam and ham. For example, in the case of 2-grams, the tokens of the expression "*machine learning techniques*" are: "*machine learning*" and "*learning techniques*". In this way, the textual feature extracted from the email contents could carry more semantic information in comparison to the BoW model. Hence, many works have used n-gram representation to deal with the problem of spam [8, 14, 20, 73]. Although such representation attempts to take advantage of contextual phrasal information (e.g., 'Additional income', 'Call free', etc.), the size of the vocabulary increases exponentially which leads to highly sparse spaces for representing email documents. In addition, they do not carry high-level semantic information since n-grams are extracted independently. Several works have proved that complex representations of texts do not always improve the efficiency of the classification and sometimes may even deteriorate it [26].

  Other studies in the n-gram model have attempted to use a character level to represent the email contents [60, 102]. Unlike the word n-gram model, the character n-gram model represents the sequence of n adjacent characters instead of words to represent the messages. For instance, in the case of 5-gram, the expression "*Hello world*" would be mapped to the tokens: "*Hello*", "*ello* ", "*llo w*", "*lo wo*", etc. The current technique is robust to grammatical errors (e.g., the word "income" and "incame" share the majority of character n-gram) and strange usage of abbreviations, punctuation marks, etc. in addition, the model does not require any text preprocessing (lemmatization, stemming, etc.) [60]. However, this method suffers from the problem of polysemy and fails to capture the meaning of the words from the text.

3.3.3 Semantic-based approaches

Given the above-cited limitations, some researchers have recently introduced semantic-based approaches for improving spam detection [17]. The major drawbacks of the previously cited methods (content-based approaches) lie in the features' independence and redundancy. For example, the words "pill", "tablet", and "Cialis" are considered independent features,but in reality, they are all related to the word "drug". This problem could affect the performance of some ML classifiers, such as naive Bayes. To address this issue, researchers studied the benefits of discovering the relationship between words to reduce the feature vector dimensionality by grouping words according to their semantics [46]. The authors in [17] define the semantic information extracted from spam emails as follows: "*semantics are related to the ability to portray and understand the meaning of information in an expressive way and their combination allows to define and understand spam in a more formal and explicit way*".

Ontologies are considered as one of the basic concepts of the semantic Web, they are widely explored in spam filtering. Generally, Ontologies are defined as a formal explicit specification of a shared conceptualization of a domain of interest [46]. One of the most popular examples of ontologies is the WordNet database (i.e. an English dictionary designed for natural language processing.). The ontology groups synonymous words that express the same concept by hierarchic levels into synsets. WordNet also provides information about word definitions, word usage examples, and different semantic relations between synsets. Various semantic relations can be distinguished in the WordNet database. For instance, hyponym (X is a kind of Y), hypernym (X includes the notion of Y among others), meronym (X is a part of Y), holonym (X contains Y among others). Below, Figure 3.3 gives an example of each semantic relation using the standardized OWL ontology representation. Usually, each relation is defined by the type of semantic relation and the sense number for the referenced synset, noted by #.

In spam detection area, the researchers have adopted ontologies to ensure a consistent textual interpretation of spam contents. For example, Méndez *et al.*[75] introduced a new semantic-based feature selection method to detect spam email by taking advantage of WordNet ontology. The basic idea of this approach is to group words into topics (e.g., the words "viagra", "cialis", "tadalafil" or "xanax" could be related to the topic "drug" or "chemical_substance") and used them to generate feature vectors to train ML algorithms. The proposed method is mainly based on four steps: 1) loading the corpus, 2) email parsing process, 3) email topic extractor and guesser, 4) compute the topic-related significance of each feature. The first step consists of loading emails into memory. The second step extracts each email's header and body parts and pre-processes them to transform the text content into valuable information. The third step guesses the topics that best match each email using WordNet Lexical Database. The authors only considered the hyponym and hypernym relations to obtain the subject corresponding to each word. Finally, the fourth step uses ML techniques to represent the extracted knowledge. The proposed method is compared against the Information Gain and the Dirichlet Latent Distribution. Topic Guessing' technique showed a significant increase in performance. However, the root topic's manual specification is not the best way of operating, and the authors are aware that this gap requires more attention.

In addition to the semantic representation of textual information in the email contents, ontologies are also used to represent user preferences in order to ensure a more consistent interpretation of spam filters [17]. For example, Kim *et al.* [62] built their ontology-based on user preferences and their reaction to new received emails such as sports, news, etc., personal information such as age, gender, etc. The authors classified the user reaction into four types: Reply, Delete, Store, and Spam. Here, the receiving action is taken into account because it is a kind of real connection between the email recipients and the way of treating these emails.

There is another type of email called Gray Email that does not provide enough features to establish a degree of confidence to determine whether the email is spam or ham. Hempelmann *et al.* [52] presented a method called "*a meaning based method*" to dis-

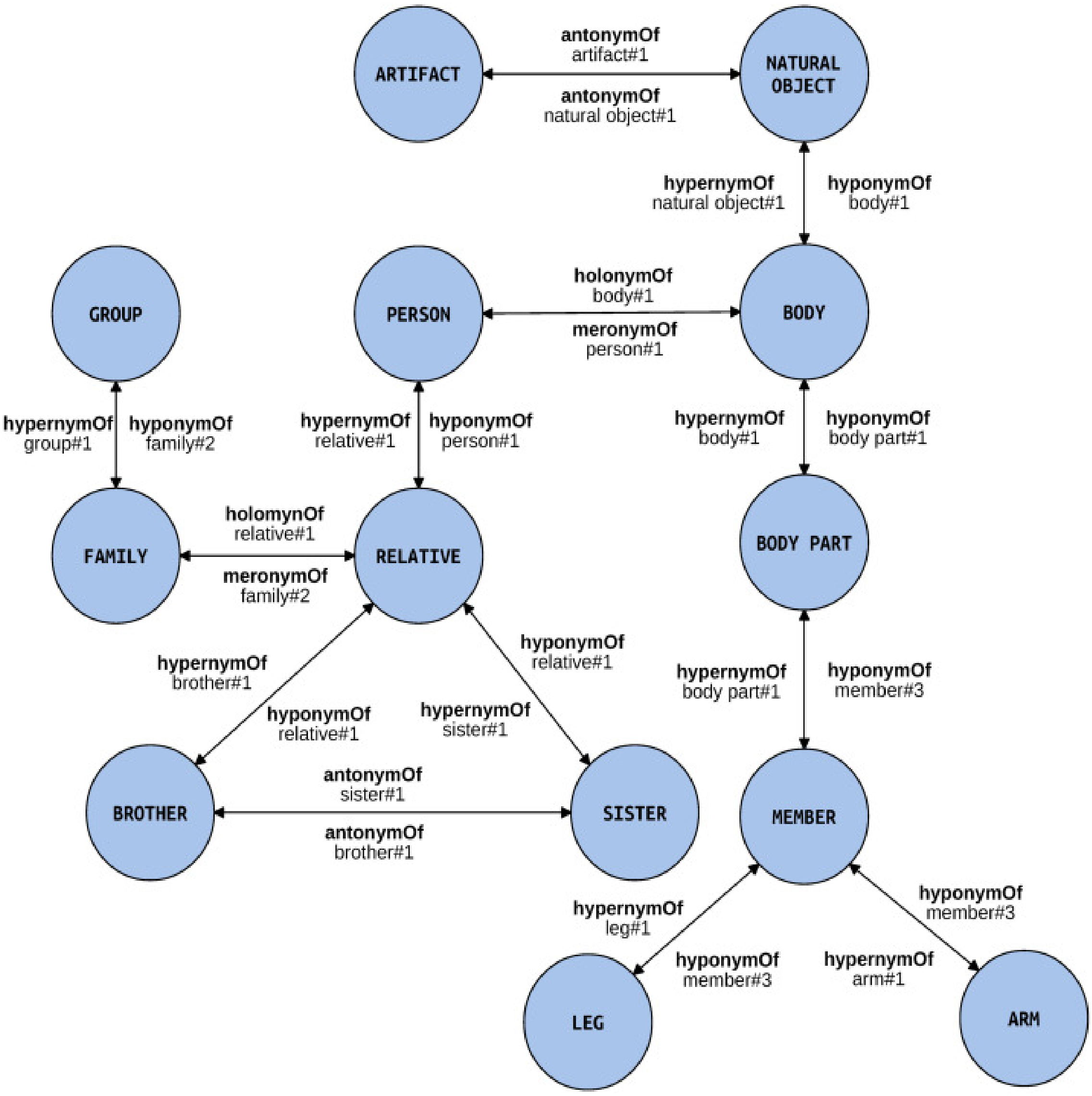


Figure 3.3: Representation of several WordNet semantic relations using standardized OWL methodology [46].

tinguish text without or with little semantic content. The idea is to apply a semantic analyzer and with the help of a semantic ontology a threshold "meaning density" is applied, if the text content is below this threshold the email is classified as spam if not is classified as ham. The results showed better accuracy in categorizing emails by adding semantic information to classifiers.

We have seen a few separate initiatives that tend to develop independent ontologies in the area of spam filtering. This can create difficult situations with regard to ontological interpretability requirements. With the increasing access to several independent ontologies, the problem of discrepancy between ontologies becomes more and more complex. The ontologies must be linked and cooperated with each other while ensuring global coherence to make possible the tasks of: integration, sharing information, seeking information from several sources, etc. So far, no work has addressed this kind of problem in the area of spam detection and no standard ontology for spam filtering has been established [17].

In [43, 94], authors studied the effectiveness of a spam classifier using the Latent Semantic Indexing (LSI) method. This method uses singular value decomposition to build a latent space in which the hidden semantics of documents are better represented. A similar model has been proposed called probabilistic Latent Semantic Analysis (pLSA) [91]. These models (LSI and pLSA) represent documents as a set of topic distributions and the topics as a set of word distributions. Other popular models such as the Latent Dirichlet Allocation (LDA) [70] and labeled-LDA [105] have been used for building latent semantic spaces for spam detection. In [99], the authors introduced the eTVSM (enhanced Topic-based Vector Space Model) method using a semantic ontology to extract the most dominant topics in a text message. The eTVSM uses the database WordNet to create a vectors of word synonyms for computing document similarity. The authors evaluated several machine learning classifiers: NB (Naive Bayes), KNN, SVM and decision tree and showed that the representation of eTVSM model provides high percentages of spam detection. These methods have generally yielded a better performance than the BoW and n-gram models.

Recently, Venkatraman *et al.* [114] proposed a spam detection approach based on conceptual and semantic similarity to enhance the naive Bayesian classifier. The approach is suggested to overcome the ambiguity raised by the polysemy of spam email contents. The authors assume that in the naive Bayesian classifier, the problem of ambiguity is raised when a set of terms are related to the same concept. For this reason, they propose integrating the naive Bayesian classifier with the Conceptual Similarity Based on Corpus (CSBC) approach and Semantic Similarity Based on the Wikipedia Links (SSBWL) approach to combat the problem of ambiguity in spam emails. The CSBC approach calculates the relationship between two terms based on their co-occurrence in a body of spam email dataset, while The SSBWL approach uses hyperlinks that exist in Wikipedia representative articles of spam terms. The proposed method has good performance in detecting spam emails. However, this method performs the conceptual and semantic similarity techniques only on spam emails and ham emails are only analyzed by NB classifier to optimize the time and space complexity.

In [65], the authors explored the use of word semantics by introducing a word sense disambiguation procedure to enhance spam detection. In [36], the authors used sentiment analysis to improve the efficiency of spam detection. They demonstrated that the polarity of the message (i.e., identification of the positive or negative nature of a message) is a useful feature for spam classification. In their work, the authors assume that the semantics of a spam message should be shaped with a positive meaning. Other studies [69, 92] explored the use of NLP for deceptive opinion spam detection in different web pages. These methods had some success in narrowing the semantic meaning of words with regard to the message content to enhance the spam detection accuracy. However, they are unable to extract high-level semantic concepts of emails which can be helpful for discriminating between legitimate and spam emails.

With the advent of Deep Learning (DL) methods, several DL-based methods for spam detection have been proposed. For example, Yang *et al.* [126] described a Multi-Modal Architecture based on Model Fusion (MMA-MF) to classify email. This model's primary concern is to separately process the email's text and image information using the LSTM (Long Short-Term Memory) model and the CNN (Convolutional Neural Network) model to detect spam. The LSTM model is used to extract semantic information hidden in the email text content and get the text part's classification probability value as spam, whereas the CNN model is used to obtain the classification probability of the attached image part. Then, the two classification probability values are combined into a fusion model using the logistic regression method to identify whether the email is spam or ham. As a result, the study showed that the proposed model could archive classification accuracy in the range of 92.64–98.48%. However, the approaches based on deep learning such as LSTM and CNN usually require big data compared to traditional ML algorithms. Moreover, the output results of these methods are hard to interpret as they are a kind of black-box models. Besides, these methods result in high computational complexity during the training stage.

Further work based on deep learning approach in [123] where the authors used the Doc2Vec model for training spam classifier on twitter messages. In the same vein, the authors in [58] used the Word2Vec word embedding to overcome the drawback of feature independence in the naive Bayes algorithm. The basic idea of the word embedding technique is that words occurring in a similar context tend to be closer in the generated vector space. Thanks to this property, these methods have increased the performance of spam detection. However, given that the spam language is generally much larger than the natural language vocabulary used to train the Doc2Vec and Word2Vec models, some discrimination information can be lost during the encoding process.

3.4 Hybrid approaches

A spam filter can combine several learning methods at the same time to help improve its accuracy. These approaches are generally aimed at minimizing the rate of false positives. Feng *et al.* [38] proposed a hybrid method called OMFS (Orthogonal Minimum Feature Selection). The OMFS method includes two main stages, namely: the OCFS (Orthogonal Centroid Feature Selection) and MRMR (Minimum Redundancy Maximum Relevance). The first stage consists of using the OCFS algorithm for feature selection on the training data space. The second stage applies the MRMR algorithm to reduce the redundant features. Three machine learning algorithms were tested by the authors to evaluate the performance of the proposed method which are: naive Bayes, KNN and SVM algorithms.

Gordillo *et al.* [45] adapted the Hidden Markov Models (HMM) to the problem of misspelled words. The authors considered a blacklist of words frequently used by spammers and a list of their variants to train one model for each forbidden word. Thereafter, to determine if a word extracted from email content represents a variant of the forbidden word, a correspondence threshold is returned by the model. This work combines the HMM model, ANN and the Genetic Algorithm (GA) which leads to a new learning algorithm in order to detect any new form of masked words. The results showed that the method is able to identify more than 95% of the variants of a word. However, no email classification results were presented by the authors.

In spam detection problem, decision tree tends to have over-sensitivity to noise and overfitting. To tackle this issue, the authors proposed hybrid combination logistic regression and decision tree for email spam detection. Logistic regression is used with certain false negative threshold to reduce noisy data before being fed by a decision tree. Evaluating the system on Spambase dataset, the results of experiments revealed that accuracy recorded 91.67% without using any feature engineering methods.

Another predictive hybrid algorithm proposed by Chitra *et al.* [19] which combines three algorithms: fuzzy logic, GA and SVM. The principle of the algorithm is to detect malicious behavior using fuzzy rules and GA. Various other methods based on hybrid algorithms have been proposed in the literature to design effective spam filters for more works see [61].

3.5 Conclusion

In this chapter, we have reviewed some of the main techniques for email spam detection. Most of the proposed methods are based on non-content and content information. Non-content-based approaches analyze the information contained in the email header, such as IP addresses and sending time. Content-based approaches analyze the email body to extract textual information, such as words, and build feature vectors to learn ML algorithms. The content approaches are widely explored in the literature due to their effectiveness. To further improve ML classifiers' performance, content approaches introduced new feature extraction techniques able to take advantage of semantic information to group words into topics or concepts. These methods have showed their efficiency compared to traditional models. However, most of these methods use holistic and general-purpose semantic features that are extracted regardless of the email content type (domain). Therefore, since semantic features are extracted independently of domains, they do not guarantee optimal spam detection performance within each domain.

4

Spam detection using automatic semantic feature extraction

4.1 Introduction

Several spam detection methods have been proposed in the literature [17] (see chapter 2 for more details). Among recent approaches, learning-based methods using text mining have gained more and more popularity [49]. Text mining techniques are based on an automatic extraction of knowledge from the content of pre-collected messages (emails). These methods generally represent email content using text features such as words, n-grams, etc., which aim to distinguish between spam and legitimate emails (ham). Other methods based on text semantic analysis have been proposed to improve the accuracy of spam detection [6] such as eTVSM, pLSA and LDA. However, most of these methods use holistic and general-purpose semantic features which are agnostic of the subject of email content (email domain). Indeed, spam content can be very dependent on the domain and the targeted users. For example, in health subject, spam can be targeted to medicine or false therapy campaigns advertisement, whereas in finance, spam can carry advertisement for dubious financial services and products. Therefore, using a specific spam discrimination for each domain can be more efficient than a general-purpose spam filter. Moreover, using more semantic cues for each domain in addition to raw text features can offer better discrimination between legitimate and spam emails. For example, emails advertising health products can be of interest if they are only informative and not targeting money extortion.

In this chapter, we propose a general approach for incorporating semantic analysis for spam detection. Semantic analysis of email content is carried out at two levels. These levels consist respectively on first categorizing emails by domains and then extracting explicit semantic features within each domain to classify emails into spam and ham categories. For email domain categorization, without loss of generality, we have considered five domains: 1) Computer, 2) Adult, 3) Education, 4) Finance, and 5) Health. These categories are among the most targeted by spam [47, 48, 119]. To assign emails to these domains, we train a supervised classifier on labeled data after operating feature selection on the vocabulary of email text content using information gain. Semantic features are then extracted automatically for each domain using CN2-SD method [66], which will serve as weak spam classifiers with outputs combined in a more general and robust classifier for discriminating spam from legitimate emails. Experiments on a large corpus of emails have shown that our approach yields very good spam classification results compared to recent text-based filtering techniques.

The rest of the chapter is organized as follows: in section 4.2 we give a general model of our first contribution to spam detection using automatic extraction of semantic features. In section 4.3 we discuss the process of email categorization by domains. In section 4.4 we describe the used method to extract the semantic features from the email data. In section 4.5 we present how to generate domain specific classifiers. We report our evaluation results in section 4.6. Finally, in section 4.7 we present the conclusion of this first contribution.

4.2 General model of automatic semantic feature extraction

In our approach for spam detection, a fundamental step is the semantic characterization of the considered specific domains (computer, adult, education, finance and health), each described by a set of semantic features. Each set of semantic features is then used to provide basic attributes for building a domain-specific classifier for spam detection. Figure 4.1 summarizes our overall approach. As we can see in figure 4.1, we proceed by two levels of semantic analysis. In the first level, we use a classification algorithm to automatically partition a global training dataset (emails) into the considered five domains. In the second level, we automatically extract a set of semantic features from the dataset in each domain. The semantic features are then used to build specialized classifiers for detecting domain-specific spam.

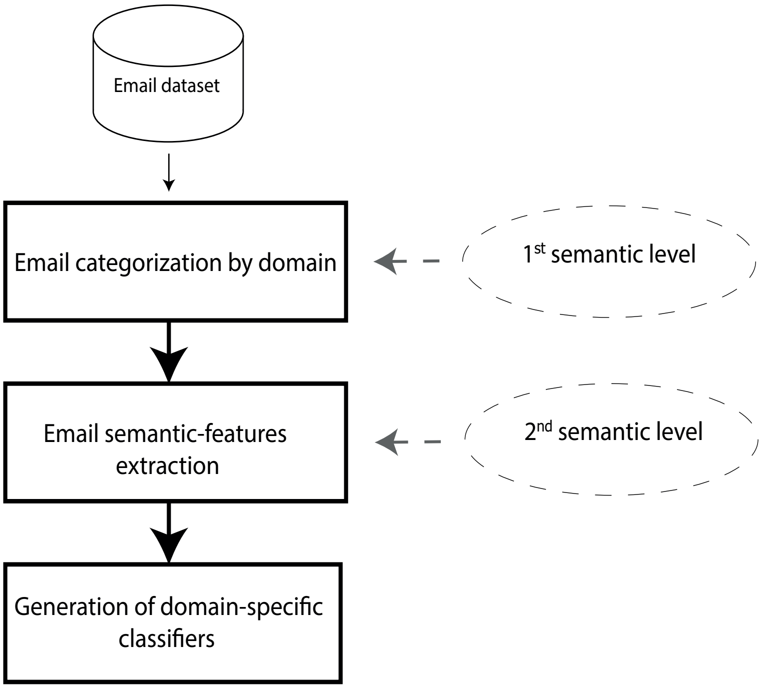

Email dataset
Email categorization by domain
1st semantic level
Email semantic-features extraction
2nd semantic level
Generation of domain-specific classifiers

Figure 4.1: General model of our approach.

4.3 Email categorization by domains

Spammers usually target domains that may interest users to promote products and services (e.g., Health, Education, Finance, etc.). According to the annual reports by Kaspersky [47, 48] and Symantec [119], nearly half of all spam is categorized as health, computer, finance, education and adult content. The reports contain a thorough analysis of the new targeted domains with the different techniques used to send spam emails. Without loss of generality, we limited our study in this chapter to these five domains. The following describes each of these most common spam domains.

- *Health care*: in the Health domain, spam may contain ads for health services or miraculous medication products that offer amazing results. Typical spam messages include advertisements for weight loss, improved posture, nutritional supplements, etc.

- *Finance*: in the Finance domain, spam may contain offers for debt consolidation services, low-interest loans, insurance, etc. Spammers target people desperate for money who need a loan or get out of debt fast and easily. These emails also target those who wish to lower their mortgage rate or have their student loans forgiven. This is a dangerous category as it can lead people to give up their personal financial information and open themselves to identity theft.

- *Computer*: in the Computer domain, typical spam messages involve special offers of bargain-priced hardware or software and services for website hosting, domain registration, website optimization, etc. Ironically, this type of spam may also falsely advertise how to eliminate computer threats and report spam. The unsuspecting victim clicks on a false link to remove their email from the spam list only to find their mailbox filled with even more spam the next day.

- *Adult*: in the Adult domain, spam messages include offers for supplements designed to increase sexual ability, pornographic sites and other adult products. Regardless if someone has ever searched for porn, these emails will still show up in a mailbox.

This type of spam almost always includes offers from dating sites and pornographic sites.

- *Education*: in the Education domain, spam may contain offers from fake online universities or training sites. These emails often include phone numbers to find out more about a degree and may also include a false number to call to opt-out of future correspondence. This category, like all others, also attempts to scam victims into giving out personal information. This type of spam can lead victims to a fake degree or education fraud.

For email categorization by domain, we assign a category to each email of the global training dataset. Notice that an appropriate email preprocessing steps are required before we can efficiently exploit the information contained in the subject and the body of emails for the categorization. We present in Figure 4.2 the global process for email categorization.

Let $D = \{d_1, \cdots, d_n\}$ be the set of email documents, $C = \{c_1, \cdots, c_p\}$ be the set of categories and $T = \{t_1, \cdots, t_m\}$ be the set of characteristics called terms. Note that in the present work, we consider five categories ($p = 5$) $C = \{Health, Finance, Computer, Adult, Education\}$ and each document $d \in D$ is associated to a unique category $c(d) \in C$. For the categorization process, we consider three main steps. The first step allows text preprocessing on documents in D, where each document is represented by a vector of terms. The second step is used to extract the relevant features. A special attention is paid for the reduction of data dimensionality to avoid deterioration of classification accuracy in the presence of noisy data. The third step consists of learning a classifier to build a better classification model for categorizing email documents by domain.

4.3.1 Preprocessing

The preprocessing phase includes the following steps:

- *Keywords recognition*: is tasked with the automatic identification of keywords and abbreviations from predefined categories. For this, we used a list of regular expres-

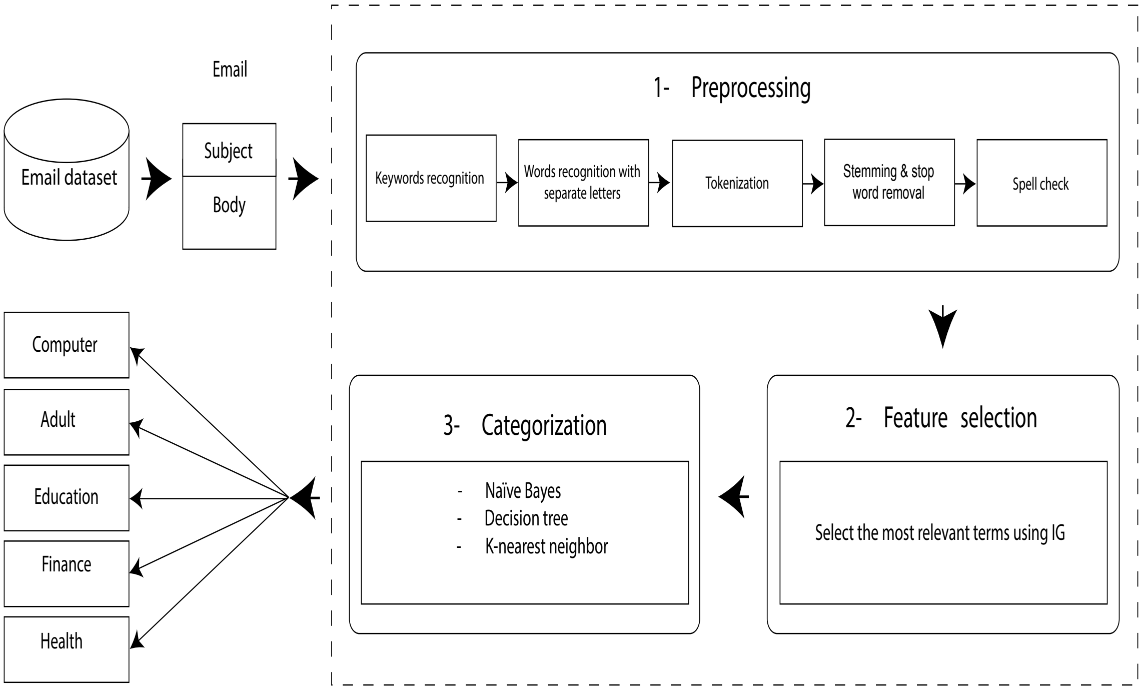


Figure 4.2: An outline of the different steps used for email categorization by domain.

sions to recognize keywords and their variants used by spammers. For instance, in the computer science category, we identified the abbreviations VLC, CD, PDF, etc.; in the adult category, we spot the taboo words, etc. Keywords are coded by using regular expressions.

- *Word with separate letters recognition*: is involved to locate words containing letters separated by blank spaces, and recognize their corresponding in a dictionary of natural language. For this, we implemented a tree search algorithm of the longest word on segments of text strings separated by empty spaces (see Example 1). This step is used to avoid alphabet letters in the feature vectors at the end of the preprocessing process. This step is important for text segmentation phase (tokenization), which is the next phase of our process.

Example 1:

Here is an example of email spam document: `"Howdy There C E R T I F I E D U N I V E R S I T Y D I P L O M A S and d e g r e e s Wouldn’t it be`

great to get a Masters degree".
The chain segments with blank spaces are: "C E R T I F I E D U N I V E R S I T Y D I P L O M A S" and "d e g r e e s". The words recognized by our algorithm on these segments are: "certified", "university", "diplomas", and "degrees".

- *Tokenization*: is the step of text segmentation to extract the words or the terms of the email. Our algorithm performs a text division into tokens by using white space as a separator. So each email is coded as a vector of tokens representing the vocabulary used in the email.

- *Stop-word removal*: is the process of removing most common words in a text, such as articles, prepositions, pronouns, etc. These words often do not carry much meaning (e.g., "to", "a", "for", etc.).

- *Stemming*: is used to perform the process of reducing variant word forms to a single "stem" form. For instance, the words: user, users, used, using all can be stemmed to the word "use". For this, we used the tool PorterStemmer [2] known as the most common algorithm for English stemming.

- *Spell checking*: is applied to check a word spelling. The extracted words are checked using the function "Spellcheck" [37]. This function checks automatically misspelled words and returns suggestions of correct words. In our work, each misspelled words is replaced with the first suggestion given by the function. This step enhances the recognition of key words in each category.

The removal of the stop words and the standardization (stemming) are very useful as they allow the dimensionality reduction of the feature vector. However, this still remain insufficient. Hence, to reduce further the dimensionality we use a statistical method for the selection of relevant features.

4.3.2 Feature selection

Feature selection is an important step and aims to reduce the number of features for improving classifier performance. To select the most representative terms we, used the Information Gain (IG) which is widely used in text categorization and spam detection [10, 49, 128]. It measures the discriminating power of a word, i.e.: the information amount provided by the knowledge of the appearance or not of a term in the decision process. Given the set T containing all email's terms obtained during the preprocessing phase from D and a set of categories C, the IG provided by a term $t \in T$ for a category $c \in C$ is defined as follows:

$$IG(t,c) = \sum_{\acute{c} \in \{c,\bar{c}\}} \sum_{\acute{t} \in \{t,\bar{t}\}} p(\acute{t},\acute{c}) log \left(\frac{p(\acute{t},\acute{c})}{p(\acute{t})p(\acute{c})} \right), \tag{4.1}$$

with:

- $p(t,c)$ (resp. $p(t,\bar{c})$) is the probability of observing the term t in a document belonging to the category c (resp. one of the other categories);
- $p(\bar{t},c)$ (resp. $p(\bar{t},\bar{c})$) is the probability of not observing the term t in a document belonging to the category c (resp. one of the other categories);
- $p(t)$ (resp. $P(\bar{t})$) is the probability of observing the term t in a document (resp. not observing the term t in a document);
- $p(c)$ (resp. $P(\bar{c})$) is the probability of the category c (resp. one of the other categories).

An IG is determined for each term t in the training collection D, and all terms are sorted according to the highest IG. The first q terms with the highest score are retained while the rest of the terms are ignored. Thus, the selected terms represent the relevant features (vocabulary) of the training data D. In our experimentation, we chose $q = 500$, the terms having the highest information gain for each category.

4.3.3 Categorization

The final step in the framework of email categorization by domain is to learn classification algorithms using the selected features in the previous step. The learned models are used to classify the new email documents (test data) into one of the predefined categories: Health, Education, Finance, Adult or Computer. In the experimental part of this chapter, we compared the following classification algorithms: naive Bayes, KNN and decision tree to categorize emails by specific domains. The used algorithms are briefly described below. See chapter 2 for more details.

- *Bayesian classifier:* is based on Bayes' theorem. For a set of training data D, the classifier calculates for each category, the probability that a document $d \in D$ represented as a vector of m terms $d = (t_1, \cdots, t_m)$, belongs to a category $c \in C$. This calculation is done for each category, and we consider the highest probability to select the category of an email.

- *K-Nearest Neighbor (KNN):* is one of the most popular used methods for text classification. The approach looks at the K email documents in the training dataset that are the closest to the email under classification: it is classified according to the class to which the majority of the K-nearest neighbors belong.

- *Decision tree:* is a structure that includes a root node, branch with internal nodes and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label.

4.4 Email semantic features extraction

This step aims at extracting semantic meanings for email text. As semantics of emails, we mean a set of hidden concepts describing the email's content. The final goal is to create a very precise semantic representation for an efficient detection of spam. In this regard, we applied CN2-SD algorithm [66] for automatic extraction of semantic features. In our case, these features are represented in the form of rules which are a conjunction of a set of terms. Each rule describes some semantic meaning in the text and its output takes on binary values (1 if the rule is satisfied and 0 otherwise).

The CN2-SD algorithm is built on top of two algorithms: the classification rule learner CN2 [21, 22] and the Subgroup Discovery (SD) [53]. In this algorithm, SD is used to discover interesting patterns in the training data and CN2 is used to induce classification rules to predict the class labels. The main difference between a classification and subgroup discovery is that classification is a predictive task, while discovering subgroups is a descriptive task. Finally, the reasons behind our choice of the CN2-SD algorithm are as follows:

- it allows an efficient and automatic induction of rules.
- it allows an automatic generation of population description. This is particularly useful for the extraction of the semantic features.
- it ensures precise discrimination between populations.

In the rest of this section, we first introduce the two algorithms, SD and CN2 , then we describe the CN2-SD algorithm.

4.4.1 Subgroup discovery algorithm

The subgroup discovery algorithm allows a descriptive induction that seeks to discover patterns that best describe the data. This algorithm is used in our work to describe semantic concepts of email contents. Semantic concept description is an important task in data mining; its goal is to summarize in a concise and understandable manner the properties of a target population (domain) in the form of a set of patterns. According to [53, 121], the SD can be defined as a data mining technique for discovering the relations among different objects (emails) with respect to certain properties of a target variable (class). These relations are encoded by rules of the form:

$$r : cond \rightarrow y, \tag{4.2}$$

where, *cond* is a conjunction of features of the form <attribute> <operator> <value> and y is the target variable (in our case spam or ham). As already mentioned in Section

4.4, SD does not aim to generate global models. Instead, it allows to identify individual patterns of interest in order to extract understandable and interpretable knowledge for descriptive purposes. Figure 4.3 shows the difference between subgroup discovery and classification tasks. Generally, in classification, a good model should be able to assign the correct class value to a new document, while in SD, the main goal is to find interesting descriptive patterns in the learning data.

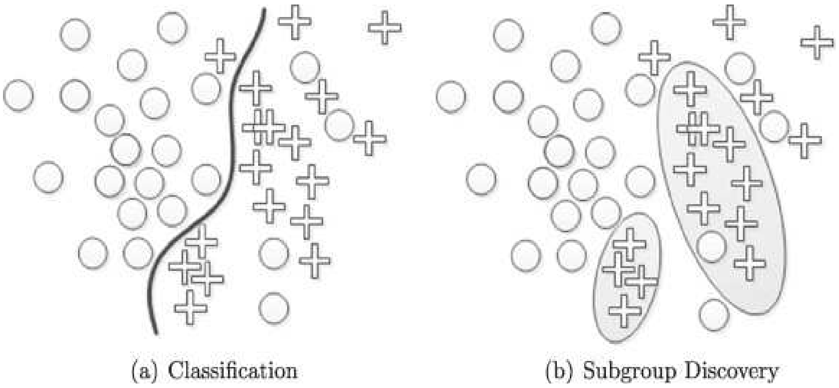


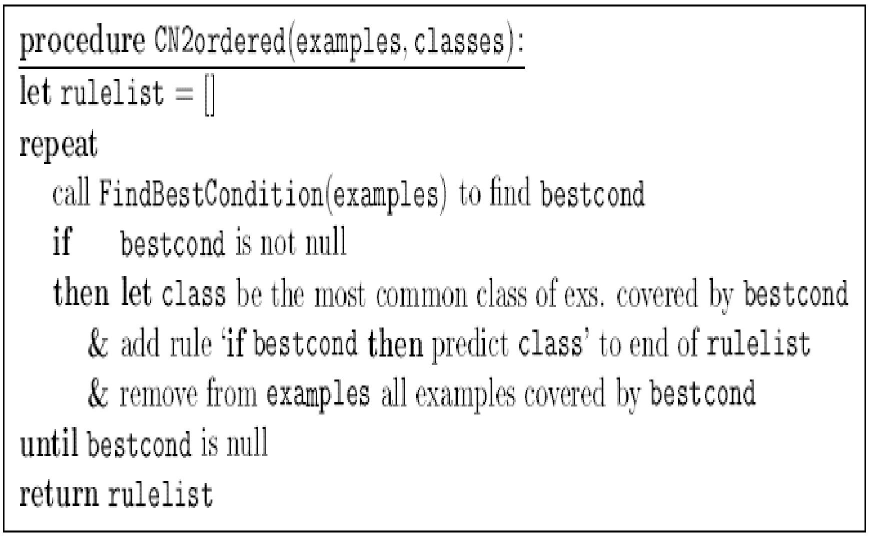
Figure 4.3: Difference between classification model (a) and subgroup discovery model (b).

4.4.2 CN2 rule induction algorithm

CN2 algorithm is one of the classical rule-based learning algorithms for inducing propositional classification rules. The algorithm consists of two main parts: a low-level part and a high-level part. A low-level part (also called a search procedure) performs the search for a single rule that covers a number of examples. A high-level part (also called control procedure) repeatedly executes the lower level to induce a set of rules. Several heuristic metrics are used in the literature to assess the quality of an induced rule at the low level. One of the most used metrics is accuracy, and it is described as follows:

$$Acc(r : cond \rightarrow y) = \frac{n(y.cond)}{n(cond)}, \tag{4.3}$$

where, $n(cond)$ represents the number of examples (emails) covered by the rule $r : cond \rightarrow y$ and $n(y.cond)$ is the number of correctly classified examples (true positives).



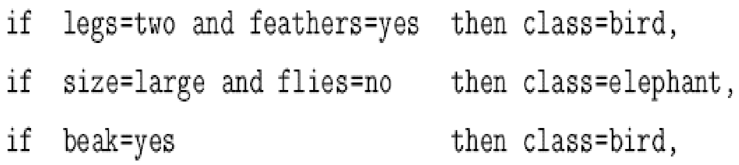
```
procedure CN2ordered(examples, classes):
let rulelist = []
repeat
    call FindBestCondition(examples) to find bestcond
    if    bestcond is not null
    then let class be the most common class of exs. covered by bestcond
        & add rule 'if bestcond then predict class' to end of rulelist
        & remove from examples all examples covered by bestcond
until bestcond is null
return rulelist
```

Figure 4.4: CN2 ordered rules algorithm [21].

```
      If    feathers = yes    then   class = bird
else  if    legs = two        then   class = human
else   ...
```

Figure 4.5: An example of ordered set of rules [21].

CN2 algorithm can use two different high-level control procedures: a procedure for inducing an ordered list of rules (see Figure 4.4 and 4.5) and a procedure for inducing an unordered list of rules (see Figure 4.6 and 4.7). For inducing an ordered list of rules, the low-level part searches for the best rule in the training data using heuristic metrics. At each search iteration, the high-level part deletes all examples covered by the induced (learned) rule until all examples in the training data are covered. In an unordered list of rules, the control procedure (high-level) is reiterated to learn rules for each class individually. For each learned rule, instead of deleting all the covered examples, which is the case for an ordered list, only the covered examples belonging to the rule class are deleted. CN2 removes the examples covered by learned rules to avoid the induction of the same rule in the next iterations.



```
if  legs=two and feathers=yes  then class=bird,
if  size=large and flies=no    then class=elephant,
if  beak=yes                   then class=bird,
```

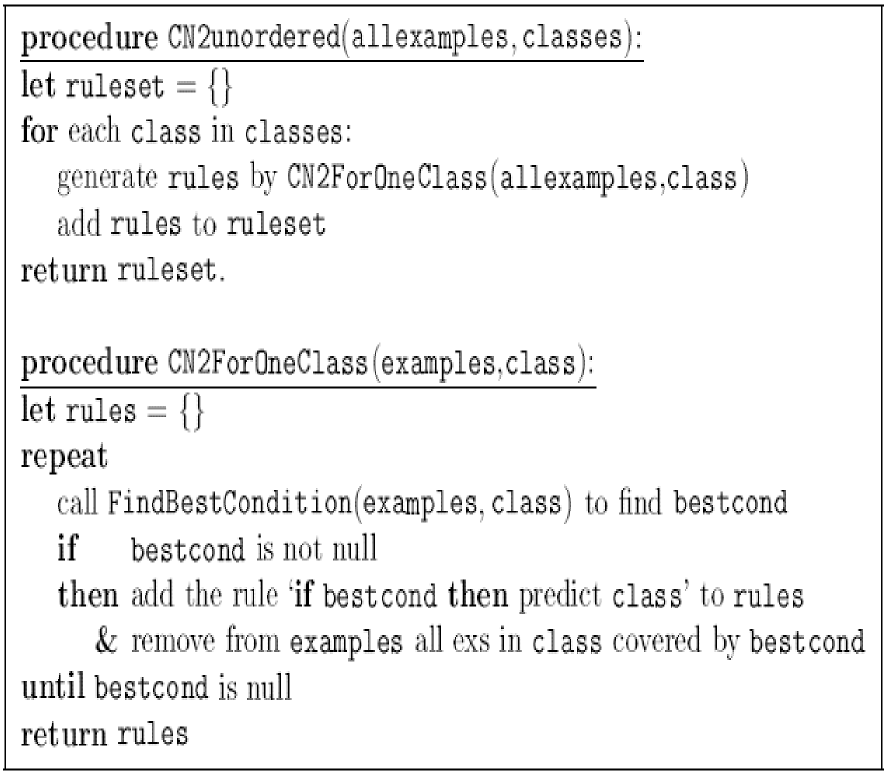
Figure 4.6: An example of unordered set of rules [21].

4.4.3 CN2-SD algorithm

The CN2-SD algorithm is an adaptation of the classifier CN2 to the task of subgroup discovery. The main modifications of the CN2 algorithm, making it appropriate for SD, involve the implementation of the weighted covering algorithm by incorporating example weights into the Weighted Relative Accuracy (WRAcc) heuristic. Algorithm 1 describes the main steps of the CN2-SD. In the first iteration of the algorithm, all examples are assigned to the same weight: $w(d_i,\ 0) = 1$, which means the email d_i have not been covered by any rule. In the following iterations, the weights of emails covered by one or more rules will decrease according to a weighting scheme. Two weighting schemes can be used in CN2-SD, the additive weights and the multiplicative weights. In our spam detection framework, we used the additive weighting scheme because of its simplicity and efficiency [66]. The equation of this scheme is defined as follows:

$$w(d_i,\ j) = \frac{1}{j+1}. \tag{4.4}$$

The CN2-SD uses a general-to-specific (top-down) search strategy to learn a set of rules. For each of these rules, the algorithm starts with the most general form having a condition part made of one (attribute, value) pair, which is extended iteratively by adding conjunctions of (attribute, value) pairs to the conditions part of the rule. The attributes are chosen so as to decrease the weight value of the covered examples by the rule, and therefore decrease the classification error. To evaluate and compare the induced rules, the algorithm uses the heuristic function WRAcc as a quality measure, which is defined as follows:



```
procedure CN2unordered(allexamples, classes):
let ruleset = {}
for each class in classes:
    generate rules by CN2ForOneClass(allexamples,class)
    add rules to ruleset
return ruleset.

procedure CN2ForOneClass(examples,class):
let rules = {}
repeat
    call FindBestCondition(examples, class) to find bestcond
    if     bestcond is not null
    then add the rule 'if bestcond then predict class' to rules
        & remove from examples all exs in class covered by bestcond
until bestcond is null
return rules
```

Figure 4.7: CN2 unordered rules algorithm [21].

$$\text{WRAcc}(r : cond \to y) = \frac{s(r \downarrow D)}{S} \left(\frac{s(r \downarrow\!* \, D)}{s(r \downarrow D)} - \frac{s(y)}{S} \right), \tag{4.5}$$

where S is the sum of the weights of all examples in D, $s(r \downarrow D)$ is the sum of the weights of all covered examples by r, $s(r \downarrow\!* \, D)$ is the sum of the weights of all correctly-covered examples by r and $s(y)$ is the sum of the weights of all examples of class y.

The algorithm CN2-SD uses the metric WRAcc to select rule candidates. It also yields unordered sets of rules but combines them in terms of a uniform weighting scheme. Furthermore, covered examples at each iteration are not removed but only re-weighted.

```
Input: learning dataset $D$.
Output: set of rules $R$.
Assign all examples in $D$ to the same weight: $w(d_i,\ 0) = 1$;
$R \leftarrow \emptyset$;
while there are examples in $D$ having a weight 1 do
    $r \leftarrow LearnRule(D)$;
    $R \leftarrow R \cup \{r\}$;
    Decrease the weight of the covered examples by $r$;
End.
return $R$.
```

Algorithm 1: CN2-SD

4.5 Generation of domain-specific classifiers

For each specific domain, the set of semantic features extracted from the previous step is used as learning attributes to build a domain-specific classifier. To this end, we use a labeled training set of emails in each domain and perform supervised learning of candidate classifiers such as naive Bayes, decision trees, and KNN.

Once domain-specific classifiers are generated, a newly coming email should be first automatically assigned to one of the considered domains. The domain-specific classifier is, then, used to classify the email as legitimate or spam. See figure 4.8 for the process of spam detection.

4.6 Experimental results

To evaluate our approach, we have built a test dataset using messages collected from several public sources: Enron [25], Ling-spam [3] and some specialized forums. Ling-spam corpus contains messages collected from a mailing list on the science and profession of linguistics. The corpus comprises 2893 messages, of which 2412 are legitimate and 481 are spam. Enron corpus is a large dataset of more than $600,000$ real emails, owned by Enron's 158 employees and acquired by the Federal Energy Regulatory Commission

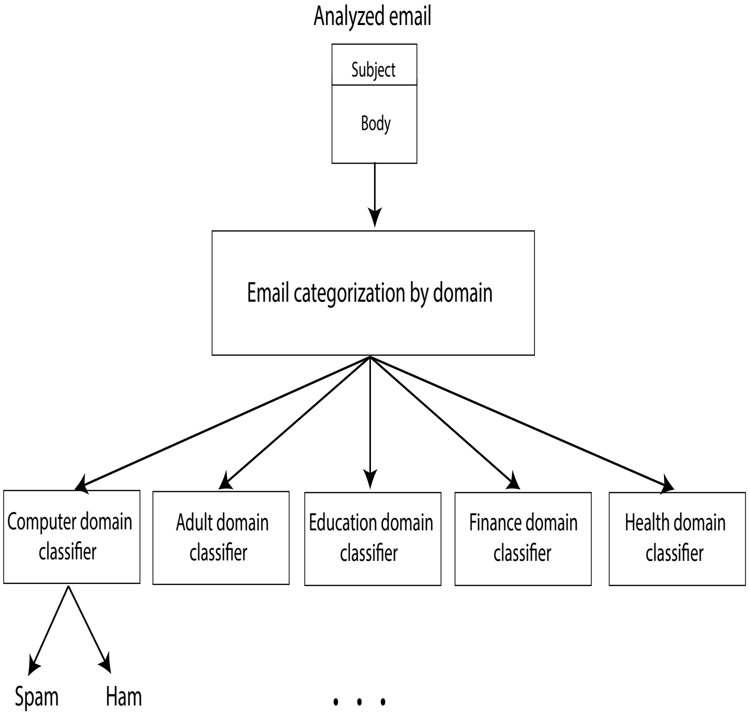

Analyzed email
Subject
Body
Email categorization by domain
Computer domain classifier
Adult domain classifier
Education domain classifier
Finance domain classifier
Health domain classifier
Spam
Ham
. . .

Figure 4.8: An outline of the approach for spam detection using domain-specific classifiers.

(FERC) during its investigation of the Enron collapse. The corpus contains a variety of topics, mainly in business such as energy trading and considerable number of personal messages representing private communication of the employees.

In our collection, spam emails come from the two datasets, Enron and Ling-spam. The ham emails come from specialized discussion forums and the two datasets Enron and Ling-spam. We resorted to the choice of forums to populate our ham dataset due to the lack of specialized open datasets containing all the domain categories addressed by our approach. For example, In Ling-spam and Enron datasets, most of the emails belong to Education and Finance domains, respectively. Therefore, we collected ham messages in specialized forums to cover the other domains (e.g., health, computer, etc.). For example, for Health, we collected messages in Question/Answering virtual clinic sites. We chosed the raw folders of the used datasets without prior preprocessing. We labeled manually the selected emails according to their domain. We used a total of 6679 emails distributed according to their content in the five pre-selected categories. The overall collection includes 3475 ham emails and 3204 spam emails.

We compared three classifiers: KNN, naive Bayes and decision tree to categorize emails by domains. Then we applied the same classifiers to separate spam from legitimate emails in each of the considered domains. In order to evaluate the performance of the machine learning classifiers, we apply the k-fold cross-validation model with $k = 10$, which randomly divides the dataset into k subsets. Each classifier is trained on $k - 1$ sets and evaluated on the remaining set. The final estimation of the classifier is the average of the k results from the subsets. Various metrics have been considered in our experiments to evaluate the performance of the classifiers, namely Precision, Recall, Accuracy and F1-measure which are computed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.8}$$

$$F1 - measure = \frac{2.Precision.Recall}{Precision + Recall} \tag{4.9}$$

where TP, TN, FP and FN are the obtained true positives, true negatives, false positives and false negatives after classification.

Table 4.1: Quantitative evaluation of machine-learning classifiers for email categorization by domains.

Classifier	Accuracy	Recall	Precision	F1-measure
KNN	0.8182	0.8758	0.5482	0.6743
Naive Bayes	0.9684	0.9684	0.9745	0.9714
Decision tree	0.9416	0.8994	0.9718	0.9342

Table 4.1 shows the obtained evaluation results for email categorization by domains. KNN, Decision tree and naive Bayes classifiers were capable of achieving more than 80 % of good prediction. Naive Bayes gives the best results with an accuracy of 0.9684, a recall of 0.9684 and a precision of 0.9745.

Table 4.2 summarizes the results for spam detection on each considered domain. We can see that naive Bayes almost achieved a higher accuracy, recall and precision in all the specified domains (computer, adult, education, finance, health). The last row of the table "Average" represents the overall evaluation results of our approach which are given by computing the average of the obtained results in the five domains. According to the results, naive Bayes classifier clearly outperforms the other algorithms. This achieved an accuracy of 0.9772, a recall of 0.9771 and a considerably high precision of 0.9705.

Table 4.2: Quantitative evaluation of machine-learning classifiers using our semantic approach for spam detection.

	Classifier	Accuracy	Recall	Precision	F1-measure
Computer	KNN	0.9424	0.9448	0.9278	0.9362
	Naive Bayes	0.9633	0.9611	0.9722	0.9666
	Decision tree	0.9677	0.9725	0.9692	0.9708
Adult	KNN	0.9782	0.9710	0.9678	0.9694
	Naive Bayes	0.9759	0.9745	0.9736	0.9740
	Decision tree	0.8763	0.8613	0.8042	0.8318
Education	KNN	0.9565	0.9696	0.9759	0.9727
	Naive Bayes	0.9779	0.9954	0.9774	0.9863
	Decision tree	0.9786	0.9963	0.9774	0.9868
Finance	KNN	0.9669	0.9888	0.9566	0.9724
	Naive Bayes	0.9707	0.9888	0.9626	0.9755
	Decision tree	0.9698	0.9856	0.9639	0.9746
Health	KNN	0.9729	0.9735	0.9620	0.9677
	Naive Bayes	0.9718	0.9655	0.9668	0.9661
	Decision tree	0.9751	0.9655	0.9746	0.9700
Average	KNN	0.9634	0.9695	0.9580	0.9637
	Naive Bayes	0.9772	0.9771	0.9705	0.9738
	Decision tree	0.9554	0.9562	0.9378	0.9469

To prove that domain-based detection allows better prediction of spam, we used a direct approach, where the semantic attributes of all the domains are used to detect spam without going through specific domains. The obtained results are shown in Table 4.3. We can see clearly that the spam detection after categorization by domain outperforms the direct approach.

Table 4.3: Results for spam detection without categorization by domains.

Classifier	Accuracy	Recall	Precision	F1-measure
KNN	0.9070	0.9301	0.8953	0.9124
Naive Bayes	0.8534	0.9494	0.8043	0.8708
Decision tree	0.9175	0.9197	0.9216	0.9206

We compare the filtering capabilities of our approach with two different approaches, eTVSM [99] and BoW-SF [6]. The approach eTVSM uses semantic relationships between terms and the approach VSM uses terms as a bag of words. In Table 4.4 we show the comparative results and we can notice that our approach outperforms the other methods. The Bag-of words (BoW) model, in general, does not carry high-level semantic information since words are taken independently. To enhance their semantic representation, BoW methods usually use n-grams instead of individual words. However, the space dimension increases exponentially, resulting in a very sparse email representation and decreases in an efficiency. The compared methods BoW-SF and eTVSM have this problem, which makes them computationally/memory expensive and less efficient for spam classification. Our approach extracts a set of semantic features per domain in the form of rules of varying lengths term sequences, that are characteristic of spam/ham in the domain. Compared to BoW-SF and eTVSM, our semantic feature space for email representation is significantly lower, yet ensuring the best discrimination between spam and ham in each specific domain. Finally, our results show that using semantic features for each specific domain is more efficient for spam detection than using holistic and general-purpose semantic features which are agnostic of the email general topic.

Table 4.4: Comparative evaluation with other methods.

Model	Classifier	Accuracy	Recall	Precision	F1-measure
eTVSM [99]	KNN	0.9355	0.9067	0.9565	0.9309
	Naive Bayes	0.9739	0.9650	0.9674	0.9662
	Decision tree	0.9657	0.9683	0.9658	0.9670
BoW-SF [6]	KNN	0.8512	0.7709	0.9312	0.8435
	Naive Bayes	0.9361	0.9784	0.9062	0.9409
	Decision tree	0.9088	0.9102	0.9142	0.9122
Our approach	KNN	0.9634	0.9695	0.9580	0.9637
	Naive Bayes	0.9772	0.9771	0.9705	0.9738
	Decision tree	0.9554	0.9562	0.9378	0.9469

4.7 Conclusion

We have proposed a new approach for exploiting semantic information for spam detection. This is achieved by extracting semantic features specific to email domains. For this purpose, we first assign emails to their domains by training a supervised classifier. Next, we apply the algorithm CN2-SD on each domain to extract semantic features to form a more general and robust spam classifier specific to each domain. Conducted experiments have shown that our approach yields better results in terms of spam detection in comparison with approaches based on bag-of-words and/or extracting word-based semantic information (eTVSM).

5

Hybrid approach for spam detection

5.1 Introduction

In the previous chapter, we used automatically generated rules for domain-specific email spam detection. In this chapter, we consider a hybrid approach by combining manually specified and automatically generated rules for domain-specific email spam filtering. More specifically, we propose a method based on two semantic level analysis. In the first level, we categorize emails by specific domains (e.g., Health, Education, Finance, etc.). In the second level, we combine a set of manually-specified and automatically-extracted semantic features for spam detection in each domain. Furthermore, we designed a procedure for solving conflicts and reducing redundancies between the two types of rules.

As already mentioned in the previous chapter, the methods based on a semantic analysis of the email content use holistic and general semantic features which are independent of the email domain [97]. Although spam messages can have some common features, their content in terms of vocabulary and underlying semantics can drastically change from one domain to another. Our approach addresses this issue by building domain-specific spam classifiers. Spam emails in each domain tend to have common semantic features which are not present in ham emails. However, since these features are usually implicit, they are hard to identify directly from the text [43]. Therefore, we propose to use a semi-automatic approach based on manually-specified and automatically-extracted rules. Manually-specified rules enable to inject expert knowledge, while automatically-extracted rules are generated from labeled data. These two sets of rules reinforce each other during spam classification. The extracted rules not only carry NLP meaning, but also model some intended actions by the sender (e.g., invitation to click on a specified link, obfuscation attempt, etc.).

In the experimental part, we have demonstrated that combining manual and automatic types of rules offers a richer set of semantic features to enforce spam classification. These features are meant to summarize the email content into compact topics discriminating spam from non-spam emails in an efficient way. We have also shown that the proposed method enables better spam detection compared to existing methods based on Bag-of-Words (BoW) and semantic content, and leads to more interpretable results.

The rest of this chapter is organized as follows: section 5.2 introduces the general model of hybrid extraction of semantic features. Section 5.3 presents the used method to categorize emails by domain. Section 5.4 details the process of manual extraction and automatic extraction of semantic features. Section 5.5 describes the combination process of the manual and the automatic semantic features to build domain-specific spam classifiers. Section 5.6 discusses the evaluation results. Finally, section 5.7 presents the conclusion.

5.2 General model of hybrid extraction of semantic features

Our approach analyzes emails at two distinct semantic levels. In the first level, emails are categorized by their domains to enable a separate conceptual view for spam in each domain. Based on a selected set of features, we categorize emails by specific domains such as: Health, Finance, Adult, etc. We compared several classification algorithms and identified the best classifier giving the most precise email categorization. In the second level, we build a spam classifier for each specific domain using semantic features. These features combine manually-specified and automatically-generated rules constructed from labeled emails. Manually-specified rules represent expert knowledge; they are built using regular expressions. Automatically-generated features are generated using the CN2- SD method [66]. Each rule has a binary outcome and acts by itself as a weak classifier for spam detection. The obtained semantic features are then used to build specialized classifiers for detecting domain-specific spam. Figure 5.1 gives a layout of steps composing our approach.

5.3 Email categorization

The email data are categorized according to the spammers' most targeted domains, which are: Health, Finance, Computer, Adult and Education (see chapter 4 for more details). It is worth to mention that spammers are constantly seeking to target new domains and develop new techniques, some sectors are quickly evolving and should be monitored closely (e.g., politics). Therefore, for the sake of completeness, we have added in this contribution a new category, called 'Other', to represent domains that are not covered by our chosen spam domain types. See figure 5.2 for more description.

After performing the text preprocessing and feature selection steps, as detailed in chapter 4, various classification algorithms are tested to categorize email documents by domain, namely: Naive Bayes, KNN, Decision tree, SVM, Adaboost and Random forest. The choice of these algorithms is motivated by several factors:

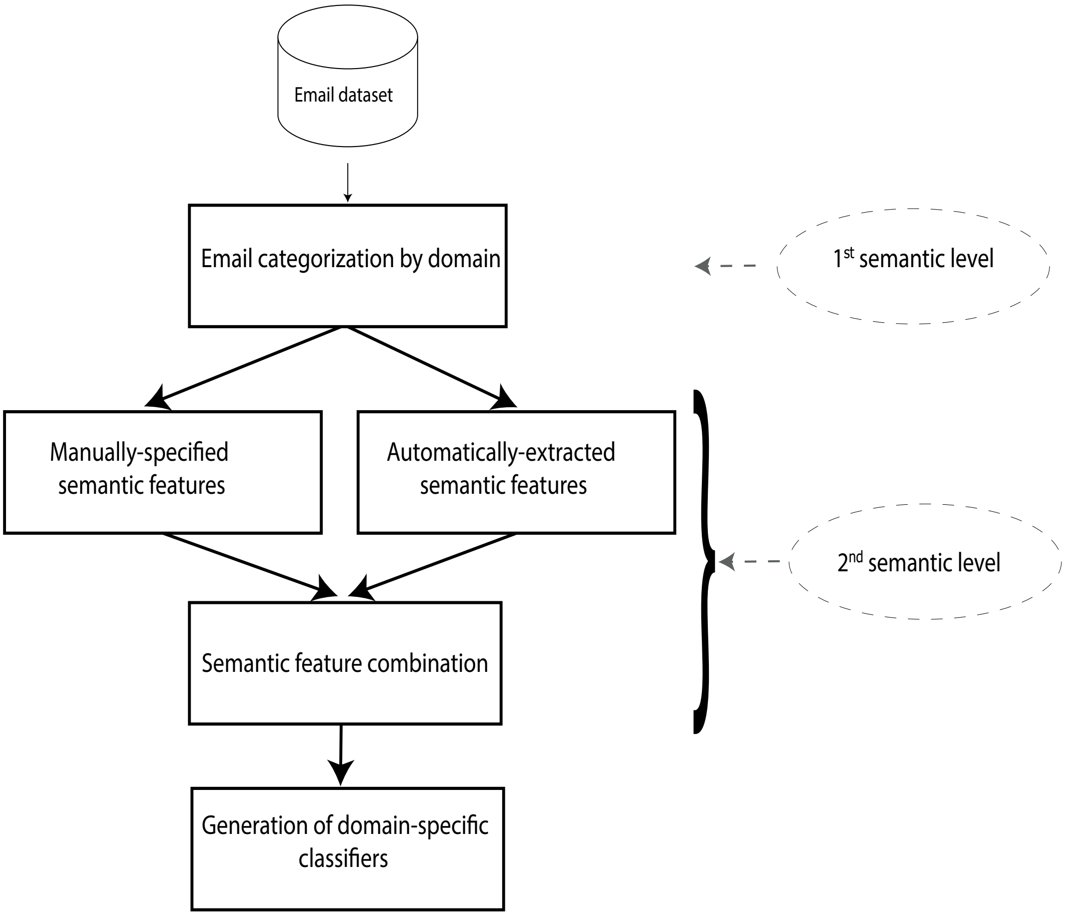

Email dataset
Email categorization by domain
1st semantic level
Manually-specified semantic features
Automatically-extracted semantic features
2nd semantic level
Semantic feature combination
Generation of domain-specific classifiers

Figure 5.1: Layout of steps composing our approach.

- They can be adapted for both binary and multiclass classification problems.
- They perform well with big training data.
- They proved their performance in text categorization and spam detection problems.
- They use different statistical models that take advantage of different assumptions for input data (independence or non-independence between features) and constitute a set of parametric and non-parametric classifiers.

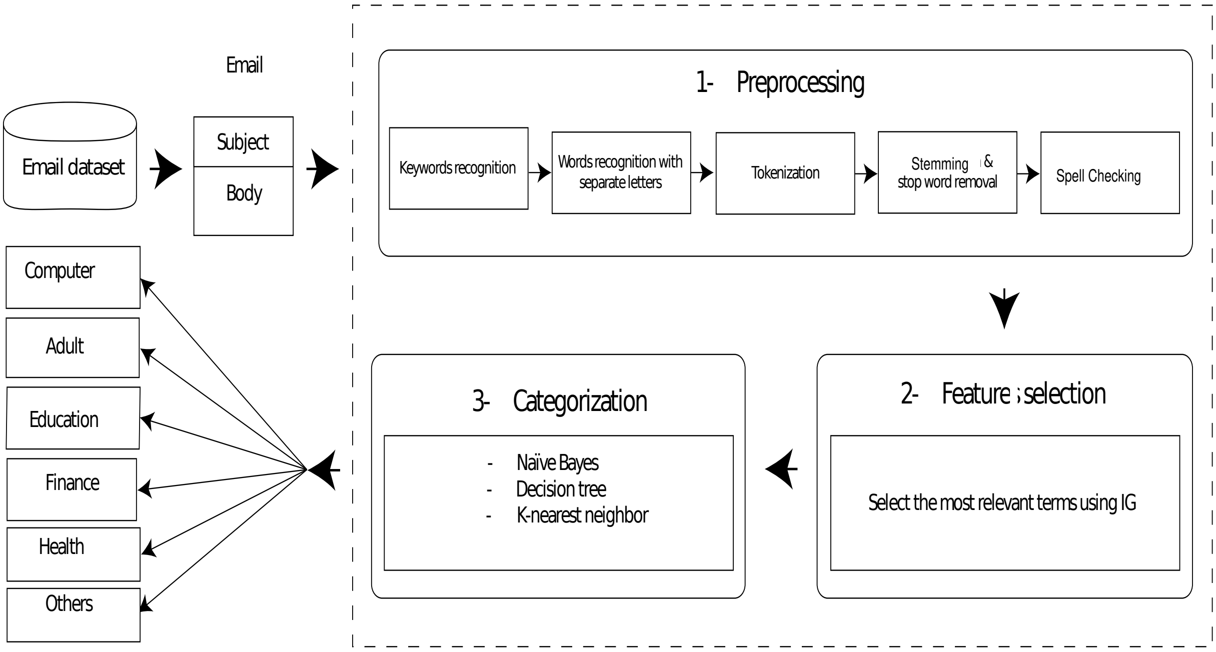


Figure 5.2: Outline of the different steps used for email categorization by domain.

5.4 Domain-specific semantic feature extraction

This step aims at extracting semantic features for representing email content in each domain. Our goal is to create a domain-specific semantic representation for an efficient detection of spam. In this regard, we consider two types of semantic features, manual and automatic. These features are represented by rules of the form $r : cond \rightarrow y$, where $cond = p_1 \wedge \ldots \wedge p_n$ is a conjunction of features (pairs of attributes-values) and $y \in \{spam, ham\}$ is the class value. We denote by $Cond(r) = \{p_1, \ldots, p_n\}$ the set of atomic features of r, by $Class(r) = y$ the class of r and by $\widehat{Cond}(r) = cond$ the conjunction of these features.

5.4.1 Manual extraction of semantic features

Manually built rules provide expert knowledge on spam classification. At this level, we were inspired by the open source SpamAssassin software [110] to build our manually-specified rules for each of the predefined categories. SpamAssassin is considered as the best example of an anti-spam filter based on rules, manually-defined by experts. Each rule condition contains a pattern that can be applied to the body or the subject of an email. We distinguish two types of rules: general and domain-specific rules. Domain-specific rules target individual categories while general rules can be applied to all categories. In other words, each category c_i has its own domain-specific rules noted Rs_i and general rules noted Rg_i. For instance, rules such as: *lose_weight*, *medication*, etc. (see Example 1) can be applied only to the category Health and rules such as: *presence of links*, *presence of specious characters between word's letters*, etc., can be applied to all categories. Note that we have built a total of 389 manually-specified rules.

Example 1:

Here is an example of two manually-specified rules for the category Health:

$lose\_weight : lose \wedge weight \rightarrow Spam$

$medication : viagra \wedge cialis \wedge tadalafil \rightarrow Spam$

Table 5.1 presents the syntax, based on regular expressions, used to describe manually-specified rules. The purpose of using regular expressions is to capture "sensitive" variations of words or sequences of words that spammers may use to fool spam filters. According to Table 5.1, a rule is composed of a set of patterns containing words, numbers, symbols and spaces.

In NLP, two types of linguistic relationships between words can be defined: morphological and semantic relations. For example, *informs*, *informing*, *informed* and *information* are all morphologically related to the root word *inform*. Semantic relation is more complex and includes synonyms, where two or more words are interchangeable because of their similar (or identical) meaning (e.g., buy and purchase); hyponyms that define inclusion relationships between words, considered oriented from most specific to most general (e.g., Cialis, Viagra and Tadalafil are hyponyms of medication) and others see [99]. Hence, we define a pattern "Words" by a set of words that are morphologically and semantically related. For this purpose, we use the Wordnet database [120] for creating morphologically and semantically related sets of words. We have applied Synset to the words of the rules to search a set of synonyms that share a common meaning.

Table 5.1: Syntax for manually-specified rules.

Rule ::= Patterns '$\rightarrow$'Class

Patterns ::= Pattern '$\wedge$' Patterns | Pattern

Pattern ::= Words | Number | Symbol | Space

Words ::= Word , Words | Word

Word ::= Letters$^+$

Letters ::= A | $\cdots$ | Z | Number | Symbol | Space

A ::= 'A'| 'a'| '@'| $\cdots$

$\cdots$

Z ::= 'Z'| 'z'| 'ż'| $\cdots$

Number ::= '0'|'1'|$\cdots$|'9'

Symbol ::= '$'| '%'| '€'| $\cdots$

Space ::= '_'Space | ϵ

Class ::= 'Spam' | 'Ham'

For example, the rule r : *buy* $\wedge$ *drug* $\rightarrow$ *Spam* carries similar semantics with the rule r' : *buying* $\wedge$ *medicine* $\rightarrow$ *Spam* since $Cond(r)$ and $Cond(r')$ are semantically equivalent.

Spammers usually use a broader language than the natural language. In fact, spammers can include empty space between letters, symbols or numbers to obfuscate some sensitive words [124]. In general, they use symbols, special characters and numbers to build forms having the aspect of alphabetical letters which are composed to words that are readable. For example, \/!*AGRĂ* instead of VIAGRA. According to [7], there are over 600 quadrillion ways to spell the word "VIAGRA" using different variations, here the letters V, I and A were replaced by: \/, ! and *Ă*.

Example 2:

The following excerpts represent real contents of some medical spam:

– `Buy your meds online! Viiiagra, also: X@nax, valium,xenical,phentermine, sommÀ , celebre><, va|trex, zyban,` `fi0ricet, adIp3x`

- `Medicati0n purchase`

- `Rx buying`

- `Pi||s online`

- `Buy tablets`

All these text contents are a kind of invitation for the reader to purchase drugs. We notice that in these texts spammers use:

- a derivation for the word 'buy' which is `buying`;

- a synonymous for the word 'buy' which is `purchase`;

- hyponyms for the word 'medication' which are `Viagra`, `Xanax`, `Valium`, `Xenical`, `Phentermine`, `Soma`, `Celebrex`, `Valtrex`, `Zyban`, `Fioricet` and `Adipex`.

- metonymies for the word 'medication' which are: `meds`, `Rx`, `Pills`, and `Tablets`.

5.4.2 Automatic extraction of semantic features

For automatic extraction of semantic features, we followed the same process of chapter 4 that uses the CN2-SD algorithm [66] for automatic rule induction. The CN2-SD algorithm is an adaptation of CN2 classification rule learner [21, 22] for Subgroup Discovery (SD) [53]. It allows to automatically describe a target population in the form of understandable and interpretable rules. This is particularly useful for extracting hidden semantic concepts and ensures accurate discrimination between the described populations. We apply the CN2-SD algorithm on the email training data to induce a set of semantic rules which enables an automatic description of each domain category c_i (see chapter 4 for more details). In this step, we have generated a total of 134 automatic rules to email spam detection. We give here an example of rules induced by the CN2-SD in the Health domain.

1. $linker \wedge \neg\, howev \wedge \neg\, horribl \rightarrow Spam$

2. $onlin \wedge^{\neg} symptom \wedge^{\neg} side \wedge^{\neg} abdomen \wedge^{\neg} coupl \wedge^{\neg} yet \rightarrow Spam$

3. $caus \wedge^{\neg} linker \wedge^{\neg} compani \wedge^{\neg} ship \wedge^{\neg} satisfi \wedge^{\neg} men \rightarrow Ham$

where $^{\neg}$ is the negation symbol, which in our case means the absence of a term.

5.5 Semantic feature combination

Once the manually-specified and automatically-generated rules are built, we combine them to obtain a complete set of rule-based semantic features. This section describes the combination process to build domain-specific spam classifiers. Note that rules can be combined by simply taking the union of the manual and automatic rules. However, this can cause two major problems: redundancy and conflict between rules. We describe in the following two sections how we resolve these two issues.

5.5.1 Merging redundant rules

This step consists of eliminating redundancy in a set of rules. We consider two rules r and r' with the same class y as redundant if one rule is more specific than the other. A rule r is considered as more specific than a rule r' if $Cond(r') \subseteq Cond(r)$. For example, if $r : p_1 \wedge p_2 \wedge p_3 \rightarrow y$ and $r' : p_1 \wedge p_2 \rightarrow y$ then r is more specific than r' and then, r' is more generic than r.

To resolve this problem, we use a post-pruning technique which is applied after the automatic induction of rules. It consists of removing non-meaningful features from the condition part of the more specific rule, which is a way of maintaining its generalization capability on new data. The more generic rule in this case will be removed. In our method, we examine iteratively each pair of rules r and r' to see if one rule is more specific than the other. The removal of features in the more specific rule is carried out iteratively as long as the classification accuracy of the updated rule is increased. To estimate the error rate of the rule on a validation set V (also called pruning set), we use the REP algorithm (Reduced Error Pruning) [41].

More formally, let V be a validation set that corresponds to our test dataset, and r and r' be a pair of rules to analyze. Let $Prune(r)$ be a pruning function that removes iteratively non-meaningful features from r as explained in the previous paragraph. Let $Error(r, V)$ (resp. $Error(r', V)$) be the error rate of r (resp. r') on V. The merging of redundant rules is done by Algorithm 2. Note that for each iteration of the algorithm, we use a procedure to search for the next pair of rules (r, r') to be analyzed.

```
Input: set of rules $I$; validation set $V$.
Output: set of rules $O$.
$O \leftarrow I$;
$b \leftarrow true$;
While $(b = true)$ do
  $b \leftarrow false$;
 If $\exists \{r, r'\} \subseteq O$ where $Cond(r') \subseteq Cond(r)$ then
     $e \leftarrow Error(r, V)$;
     $e' \leftarrow Error(r', V)$;
      While $(e > e')$
           $r \leftarrow Prune(r)$;
           $e \leftarrow Error(r, V)$;
      end
     $O \leftarrow O \setminus \{r'\}$;
     $b \leftarrow true$;
 End

End
```

Algorithm 2: Merging redundant rules

5.5.2 Conflict resolution between rules

Conflict resolution allows to reduce class overlapping and ensure better spam discrimination. We say that two rules $r : cond \rightarrow y$ and $r' : cond' \rightarrow y'$, with $y \neq y'$, are in conflict if there exist a non-empty set of emails in the training data that matches both rules. Our conflict resolution algorithm between rules is inspired from [50] which resolves conflict problems in the case of combining decision trees learned in parallel.

Given a pre-collected dataset of emails D and a pair of rules r and r' to analyze, let $A = r \downarrow D$ and $A' = r' \downarrow D$, where $r \downarrow D$ (resp. $r' \downarrow D$) is the set of emails in D covered by r (resp. r'). If $A' \subseteq A$ then we say that $Class(r)$ is the majority class and $Class(r')$ is the minority class. We define $InferiorRule(r, r', D)$ as a function returning the rule associated with the minority class. Let B be a set of labeled examples (emails) covered by both rules r and r' called "conflicts examples". We define $MajorityClassLabel(B)$ as the function returning the majority class label among the examples in B.

The conflict resolution between rules is presented in Algorithm 3 which stipulates that two rules r and r' are in conflict if they are associated with two different classes $Class(r) \neq Class(r')$ and they share in their cover a set of examples (i.e., $A \cap A' \neq \emptyset$). For example, the following two rules are in conflict:

$r : link \wedge viagra \rightarrow Spam$
$r' : link \wedge doctor \rightarrow Ham$

Following the instructions of Algorithm 3, $L_1 = \{viagra\}$, $L_2 = \{doctor\}$ and $L_3 = \{link\}$. Each rule is strengthened by adding the complement of the conditions part which is not in common:

$r : link \wedge viagra \wedge^{\neg} doctor \rightarrow Spam$
$r' : link \wedge doctor \wedge^{\neg} viagra \rightarrow Ham$

Then, a new rule r'' is created with the condition $L_1 \wedge L_2 \wedge L_3$. If the common part between the conditions of r and r' is empty, then L_3 is also empty and not included. If a majority of the remaining "conflicts" examples are Ham, then the majority class is Ham and the rule will be :

$r'' : link \wedge viagra \wedge doctor \rightarrow Ham$

In case of conflict, therefore, the algorithm returns three rules r, r', and r'', otherwise it returns the original two rules r and r'. When this resolution step does not find new

```
Input: set of rules $I$; learning dataset $D$.
Output: set of rules $O$.
$O \leftarrow I$;
$b \leftarrow true$;
While $(b = true)$ do
  $b \leftarrow false$;
  If $\exists\{r, r'\} \subseteq O$ where $(Class(r) \neq Class(r')) \wedge ((r \downarrow D) \cap (r' \downarrow D) \neq \emptyset))$ then
      If $Cond(r) = Cond(r')$ then
          $O \leftarrow O \setminus \{InferiorRule(r, r', D)\}$;
      Else
          If $Cond(r') \subset Cond(r)$ then
              $L \leftarrow Cond(r) \setminus Cond(r')$;
              $r' \leftarrow (\widehat{Cond(r')} \wedge^{\neg} \widehat{L} \rightarrow Class(r'))$;
          ElseIf $Cond(r) \subset Cond(r')$
              $L \leftarrow Cond(r') \setminus Cond(r)$;
              $r \leftarrow (\widehat{Cond(r)} \wedge^{\neg} \widehat{L} \rightarrow Class(r))$;
          Else
              $D_1 \leftarrow (r \downarrow D) \cap (r' \downarrow D)$;
              $L_1 \leftarrow Cond(r) \setminus Cond(r')$;
              $L_2 \leftarrow Cond(r') \setminus Cond(r)$;
              $L_3 \leftarrow Cond(r) \cap Cond(r')$;
              $r \leftarrow (\widehat{Cond(r)} \wedge^{\neg} \widehat{L}_2 \rightarrow Class(r))$;
              $r' \leftarrow (\widehat{Cond(r')} \wedge^{\neg} \widehat{L}_1 \rightarrow Class(r'))$;
              $D_2 \leftarrow (r \downarrow D) \cap (r' \downarrow D)$;
              $r'' \leftarrow (\widehat{L}_1 \wedge \widehat{L}_2 \wedge \widehat{L}_3 \rightarrow MajorityClassLabel(D_1 \setminus D_2))$;
              $O \leftarrow O \cup \{r''\}$;
          End
      End
      $b \leftarrow true$;
  End
End
```

Algorithm 3: Conflict resolution

conflicts, we go back to Algorithm 2 to remove all redundant rules that might be created by the conflict elimination process.

5.6 Experimental results

To strengthen the validation of our method, we extended the testing dataset used in the first contribution of the previous chapter by collecting more examples from the Enron dataset and the discussion forums. In our current collection, we used a total of 8188 emails distributed according to their content in the six pre-selected categories: Health, Adult Finance, Education, Computer and Others (see Table 5.2). Each category includes both spam and ham emails. Further, to extend the evaluation of our work, we tested another dataset called CSDMC2010 SPAM. The dataset contains a total of 4327 labeled emails where 1378 are spam and 2949 are Ham.

In this experiment, we presented a comparison to other classification methods such as Adaboost, Random forest and SVM in addition to Naive Bayes, KNN and Decision tree classifiers to categorize emails by domain. We also used the same classifiers for spam detection on each of the considered domains. Moreover, to evaluate the effectiveness of the classifiers, we applied k-fold cross validation model, with $k = 10$. In 10-fold cross validation, the dataset is split randomly into 10 partitions called folds. We then fitted the classifiers to a dataset consisting of 9 parts of the original 10 parts and used the remaining portion for validation. The process is then repeated 10 times and the results are averaged.

To address the problem of emails that can belong to multiple domains, we have tested a soft classification procedure based on the Bayes classifier and decision trees, where a performance improvement has been noticed. To evaluate the performance of the classifiers, four metrics have been considered in our experiments, namely *Precision*, *Recall*, *Accuracy* and *F1-measure*.

Table 5.3 shows the results of email categorisation by domain using our approach. Our approach generally provides high performance for domain categorization. We can note that the naive Bayes and SVM performed better than the other tested classifiers. For example, SVM classifier generated an *Accuracy* of 96.66 %, a *Recall* of 95.33 %, and a

Table 5.2: Statistics of our collected dataset.

	Health	Adult	Finance	Education	Computer	Others
Ham	753	466	593	1075	808	1011
Spam	1061	563	465	254	861	278
Total	1814	1029	1058	1329	1669	1289

Precision of 97.69 %.

Table 5.3: Quantitative evaluation of machine-learning classifiers for email categorization by domain.

Classifier	Accuracy	Recall	Precision	F1-measure
KNN	0.8189	0.8945	0.6685	0.7652
Naive Bayes	0.9601	0.9602	0.9668	0.9635
Decision tree	0.9584	0.9456	0.9765	0.9661
Adaboost	0.8701	0.8836	0.8597	0.8715
Random forest	0.9586	0.9441	0.9721	0.9579
SVM	0.9666	0.9533	0.9769	0.9650

To demonstrate the advantage of using domain-specific spam classification, we implemented another version of our algorithm where we omit domain categorisation and concatenate all semantic domain features to train our classifiers. Tables 5.4 and 5.5 show results obtained using our original approach and its modified version, respectively. These results show that the categorization by domain contributes significantly to the improvement of spam detection. The best results have been obtained by the classifiers Adaboost, Random forest and naive Bayes. They achieved an Accuracy of more than 98 % in almost all of the predefined domains. The last row of the Table 5.4 gives averages of spam classification accuracies for all considered domains. The overall evaluation shows that Adaboost outperforms the other classifiers with an *Accuracy* of 98.53%, a *Recall* of 98.99%, *F1-measure* of 98.80%, and a considerably high *Precision* of 98.66%.

In a second experiment, we have compared the performance of our approach with

Table 5.4: Quantitative evaluation of machine-learning classifiers using our semantic approach for spam detection.

	Classifier	Accuracy	Recall	Precision	F1-measure
Computer	KNN	0.9786	0.9694	0.9740	0.9716
	Naive Bayes	0.9671	0.9862	0.9726	0.9793
	Decision tree	0.9785	0.9723	0.9783	0.9752
	Adaboost	0.9896	0.9863	0.9807	0.9834
	Random forest	0.9823	0.9766	0.9721	0.9743
Adult	KNN	0.9791	0.9794	0.9875	0.9834
	Naive Bayes	0.9899	0.9838	0.9792	0.9815
	Decision tree	0.9426	0.9324	0.9778	0.9546
	Adaboost	0.9867	0.9868	0.9896	0.9882
	Random forest	0.9793	0.9888	0.9781	0.9834
Education	KNN	0.9786	0.9854	0.9755	0.9804
	Naive Bayes	0.9862	0.9963	0.9893	0.9928
	Decision tree	0.9816	0.9955	0.9863	0.9909
	Adaboost	0.9926	0.9974	0.9948	0.9961
	Random forest	0.9823	0.9867	0.9871	0.9869
Finance	KNN	0.9580	0.9696	0.9729	0.9712
	Naive Bayes	0.9829	0.9887	0.9873	0.9880
	Decision tree	0.9665	0.9856	0.9728	0.9792
	Adaboost	0.9773	0.9872	0.9809	0.9840
	Random forest	0.9683	0.9700	0.9763	0.9731
Health	KNN	0.9616	0.9835	0.9728	0.9781
	Naive Bayes	0.9954	0.9855	0.9888	0.9871
	Decision tree	0.9873	0.9746	0.9825	0.9785
	Adaboost	0.9939	0.9916	0.9922	0.9905
	Random forest	0.9864	0.9839	0.9867	0.9852
Others	KNN	0.9524	0.9696	0.9646	0.9671
	Naive Bayes	0.9776	0.9869	0.9862	0.9865
	Decision tree	0.9658	0.9822	0.9741	0.9781
	Adaboost	0.9717	0.9900	0.9811	0.9855
	Random forest	0.9712	0.9771	0.9826	0.9798
Average	KNN	0.9681	0.9762	0.9746	0.9753
	Naive Bayes	0.9831	0.9879	0.9839	0.9859
	Decision tree	0.9704	0.9738	0.9786	0.9761
	Adaboost	0.9853	0.9899	0.9866	0.9880
	Random forest	0.9783	0.9805	0.9805	0.9805

Table 5.5: Results for spam detection without categorization by domain.

Classifier	Accuracy	Recall	Precision	F1-measure
KNN	0.9375	0.9501	0.9038	0.9264
Naive Bayes	0.8912	0.9454	0.8387	0.8889
Decision tree	0.9406	0.9476	0.9469	0.9472
Adaboost	0.9592	0.9648	0.9628	0.9638
Random forest	0.9426	0.9424	0.8813	0.9108
SVM	0.9419	0.9553	0.9359	0.9455

several state-of-the-art methods, namely eTVSM [99], Doc2Vec [123] and BoW-SF [6]. Tables 5.6 and 5.7 show the obtained results for each compared method using our dataset and CSDMC2010 SPAM dataset, respectively. We can clearly note that our approach outperforms the other methods for both datasets. This performance can be explained by several factors. BoW-SF [6] does not carry high-level semantic information since words are taken independently. Since eTVSM [99] and Doc2Vec [123] are agnostic of email domain subject, it decreased their performance with regard to our method. Finally, semantic features used in our approach are elaborated from labeled data using manually-defined and automatically-generated rules, which gives them a high spam discrimination potential.

Finally, note that the best results achieved by the Doc2vec [123] in term of accuracy are 85.54% in our dataset and 87.71% in the CSDMC2010 SPAM dataset, which is lesser than the other methods. This indicates that the Doc2Vec features are not very powerful for the problem of email spam detection. We believe that one of the main factors causing this decrease of efficiency is that the spam language is much larger than the natural language vocabulary used to train the Doc2Vec. In other words, when encoding an email document using Doc2Vec, much discrimination information can be lost.

Finally, to test the ability of our approach to adapt to new data, we retrained our classifiers by augmenting our training dataset with 25% of the CSDMC2010 SPAM dataset emails chosen randomly. Obtained classification results on the CSDMC2010 SPAM dataset are shown in Table 5.8. We can see that by augmenting the training dataset, our approach

Table 5.6: Comparative evaluation with other methods using our collected dataset.

Model	Classifier	Accuracy	Recall	Precision	F1-measure
eTVSM [99]	KNN	0.9347	0.9265	0.9573	0.9421
	Naive Bayes	0.9768	0.9633	0.9651	0.9642
	Decision tree	0.9684	0.9659	0.9672	0.9665
	Adaboost	0.9839	0.9758	0.9744	0.9751
	Random forest	0.9686	0.9674	0.9668	0.9671
	SVM	0.9723	0.9689	0.9752	0.9742
BoW-SF [6]	KNN	0.8709	0.8261	0.9255	0.8728
	Naive Bayes	0.9488	0.9650	0.9308	0.9476
	Decision tree	0.9265	0.9312	0.9242	0.9277
	Adaboost	0.9511	0.9783	0.9673	0.9727
	Random forest	0.9302	0.9359	0.9251	0.9305
	SVM	0.9427	0.9354	0.9514	0.9452
Doc2Vec [123]	KNN	0.7526	0.5732	0.8744	0.6925
	Naive Bayes	0.6576	0.8406	0.2456	0.3746
	Decision tree	0.7953	0.6993	0.7947	0.7440
	Adaboost	0.8279	0.7030	0.8672	0.7765
	Random forest	0.8554	0.7671	0.8775	0.8186
	SVM	0.8464	0.7682	0.8557	0.8096
Our approach	KNN	0.9681	0.9762	0.9746	0.9753
	Naive Bayes	0.9831	0.9879	0.9839	0.9859
	Decision tree	0.9704	0.9738	0.9786	0.9761
	Adaboost	0.9853	0.9899	0.9866	0.9880
	Random forest	0.9783	0.9805	0.9805	0.9805
	SVM	0.9769	0.9716	0.9798	0.9757

Table 5.7: Comparative evaluation with other methods using CSDMC2010 SPAM dataset.

Model	Classifier	Accuracy	Recall	Precision	F1-measure
eTVSM [99]	KNN	0.9194	0.9048	0.9323	0.9183
	Naive Bayes	0.9423	0.9448	0.9504	0.9476
	Decision tree	0.9288	0.9156	0.9425	0.9289
	Adaboost	0.9461	0.9352	0.9516	0.9433
	Random forest	0.9533	0.9266	0.9747	0.9500
	SVM	0.9486	0.9381	0.9567	0.9473
BoW-SF [6]	KNN	0.7825	0.5301	0.9215	0.6809
	Naive Bayes	0.9134	0.7814	0.9725	0.8665
	Decision tree	0.8134	0.8960	0.8896	0.8928
	Adaboost	0.9107	0.8700	0.9512	0.9088
	Random forest	0.9128	0.8998	0.9411	0.9200
	SVM	0.9232	0.8940	0.9415	0.9171
Doc2Vec [123]	KNN	0.9217	0.9115	0.8527	0.8811
	Naive Bayes	0.7467	0.5739	0.7946	0.6665
	Decision tree	0.8916	0.8382	0.8244	0.8312
	Adaboost	0.9184	0.8824	0.8643	0.8732
	Random forest	0.9265	0.8701	0.8961	0.8829
	SVM	0.9293	0.9064	0.8771	0.8915
Our approach	KNN	0.9462	0.9492	0.9322	0.9406
	Naive Bayes	0.9464	0.9476	0.9340	0.9408
	Decision tree	0.9298	0.9267	0.9117	0.9191
	Adaboost	0.9554	0.9533	0.9466	0.9499
	Random forest	0.9500	0.9482	0.9409	0.9445
	SVM	0.9597	0.9541	0.9563	0.9552

Table 5.8: Results evaluation of our approach using CSDMC2010 SPAM dataset.

Classifier	Original training dataset	Original training dataset augmented with 25% from CSDMC2010 SPAM dataset
	Accuracy	Accuracy
KNN	0.9253	0.9462
Naive Bayes	0.9419	0.9464
Decision tree	0.9083	0.9298
Adaboost	0.9399	0.9554
Random forest	0.9345	0.9500
SVM	0.9397	0.9597

increased its performance for spam classification. This gives us, among other things, an indication about the potential of our approach to adapt to new email content given a proper training.

5.6.1 Soft vs. hard classification

In a final test, we deal with the situation where an email may have more than one correct category. For example, email content could belong to both domains Education and Finance. To address this issue, we use soft classification in the first semantic level instead of hard classification. Soft classifiers allow text documents to have variable degrees of membership to multiple categories and assign a membership probability for a document between 0 and 1 to each category. Our soft classification is based on the Bayesian probability theory. For a set of training data $D = \{d_1, \cdots, d_n\}$ and a set of domain categories $C = \{c_1, \cdots, c_p\}$ where $p = 6$, the classifier calculates for each class spam/ham, the marginal probability that a document (email) $d \in D$ belongs to a class spam (or ham):

$$P(spam \mid d) = \sum_{i=1}^{p} p(spam, c_i \mid d), \tag{5.1}$$

$$P(spam \mid d) = \sum_{i=1}^{p} p(spam \mid c_i) \cdot p(c_i \mid d). \tag{5.2}$$

where $p(c_i \mid d)$ is the posterior probability of having domain c_i given email document d. This calculation is done for each class spam/ham, and we consider the highest probability to select class label for the document. Comparison between a hard classification and soft classification have been performed to see the effect on spam detection accuracy. Based on the results given in the Tables 5.9 and 5.10, soft classification generally gives better accuracy rates than hard classification for both datasets. This experiment confirms that combining email categorization with soft classification gives a powerful tool for spam detection.

Table 5.9: Comparative evaluation Soft/Hard spam classification using our collected dataset.

Classifier	Hard classification				Soft classification			
	Accuracy	Recall	Precision	F1-measure	Accuracy	Recall	Precision	F1-measure
Naive Bayes	0.9831	0.9879	0.9839	0.9859	0.9892	0.9883	0.9897	0.9890
Decision tree	0.9704	0.9738	0.9786	0.9761	0.9767	0.9766	0.9792	0.9779

Table 5.10: Comparative evaluation Soft/Hard spam classification using CSDMC2010 SPAM dataset.

Classifier	Hard classification				Soft classification			
	Accuracy	Recall	Precision	F1-measure	Accuracy	Recall	Precision	F1-measure
Naive Bayes	0.9464	0.9464	0.9476	0.9340	0.9574	0.9469	0.9598	0.9533
Decision tree	0.9298	0.9298	0.9267	0.9117	0.9281	0.9131	0.9357	0.9243

5.7 Conclusion

We have proposed a new approach using semantic information for spam detection. This approach is composed of two main stages. The first stage categorizes email contents by subject domains, whereas the second stage builds domain-specific semantic features on which spam classification is performed. These features provide a precise description of each domain spam allowing to better targeting their detection. We have shown that domain categorization improves considerably filter performance and promotes good prediction. Experimental results have also shown that our approach outperforms several state-of-the-art methods based on BoW and latent semantic analysis.

6

Deep learning based approach for spam detection

6.1 Introduction

This chapter is an extension of our work in chapter 4, called here TBSR-SD (Text-Based Semantic Representation for Spam Detection). In that work, we considered semantic information at two levels. In the first level, we used Bag-of-Words (BoW) model to categorize emails into subject categories (e.g., finance, medicine, etc.), which enables targeting each domain-specific semantics for spam detection. In the second level, semantic features are extracted for spam detection in each domain. We applied the CN2-SD algorithm [66] for an automatic extraction of semantic features. The features are represented in the form of rules which are a conjunction of words where each rule describes some semantic meaning in the text. In this work, we use in both levels more elaborated semantic features based on deep learning and semantic ontology. In the first level, we use a deep learning approach based on Word2Vec model [80] to learn word embeddings in order to categorize email documents by domains. In the second level, we use eTVSM [99], based on semantic ontology, to extract the most dominant topics in the email content. The resulting topics are then used as input data of the CN2-SD algorithm for the extraction of the semantic features. These features, encoded as rules, are conjunctions of topics rather than words. Each rule has a binary outcome and acts by itself as a weak classifier for spam detection. Experiments on a large corpus of emails have shown that the combination of the whole rules produces a strong classifier enabling robust and accurate spam detection.

The rest of this chapter is organized as follows: section 6.2 detail the process of email categorization by domains using word embeddings. Section 6.3 presents the domain-specific semantic feature extraction using eTVSM and CN2-SD algorithms. Section 6.4 discusses the evaluation results. Section 6.5 gives the overall comparison of our contributions. Finally, section 6.6 presents the conclusion.

6.2 Email categorization by domains using word embeddings

To build our spam detection model, we start by categorizing email documents into the most targeted domains by spammers [47, 48, 119]. We have considered six domains: Computer, Adult, Education, Finance, Health and Others. The category 'Others' is used to ensure the completeness of the domain space. To assign emails to these domains, we train a supervised classifier on labeled data after operating a text preprocessing and applying Word2Vec word embedding [80] on email contents.

6.2.1 Text preprocessing

The preprocessing step converts an input text document into a vector of words. In this work, we consider five preprocessing steps to be applied for the input text document:

keywords recognition, word with separate letters recognition, segmentation, stop-word removal and stemming. Keywords recognition allows automatic identification of keywords and abbreviations from predefined categories. Word with separate letters recognition enables to recognize words containing letters separated by blank spaces. Segmentation permits to segment email content into a set of words. Stop-word removal deletes most common words in a text (e.g., articles, prepositions, etc.). Stemming allows the transformation of words into their roots. See chapter 4, section 4.3.1, for more details.

6.2.2 Word2Vec word embedding

Word2Vec is one of the most popular approaches to learn word embedding using neural networks. The approach converts words into corresponding vectors in such a way that the semantically similar vectors should be close to each other in N-dimensional space (N refers to the dimensions of the vector). The powerful concept behind word2vec is that word vectors that are close to each other in the vector space represent words that are not only of the same meaning but of the same context as well.

Word2Vec is proposed in [80]. Google's Word2Vec is trained on Google News Corpus, which has more than 3 Million running words. The approach comprises two training models: Continuous Bag Of Words model (CBOW) and Skip-Gram (SG) [79]. The CBOW predicts the target or the current word based on the surrounding context information, while the SG predicts surrounding words (context) based on the given current word (noted by w(t) in Figure 6.1). Both models are represented in Figure 6.1. As we used the SG model in this study, we briefly discuss the main techniques of this model below.

- **Skip Gram Model**

  The goal of SG model is to train word vector representations to predict their context (the nearby words) within a window of fixed size c. The window represents the words in the right and the left of the target word. More formally, suppose a sequence of training words denoted by $w_1, w_2, \ldots, w_T$, the objective function of the SG model is to maximize the average log probability of a context word

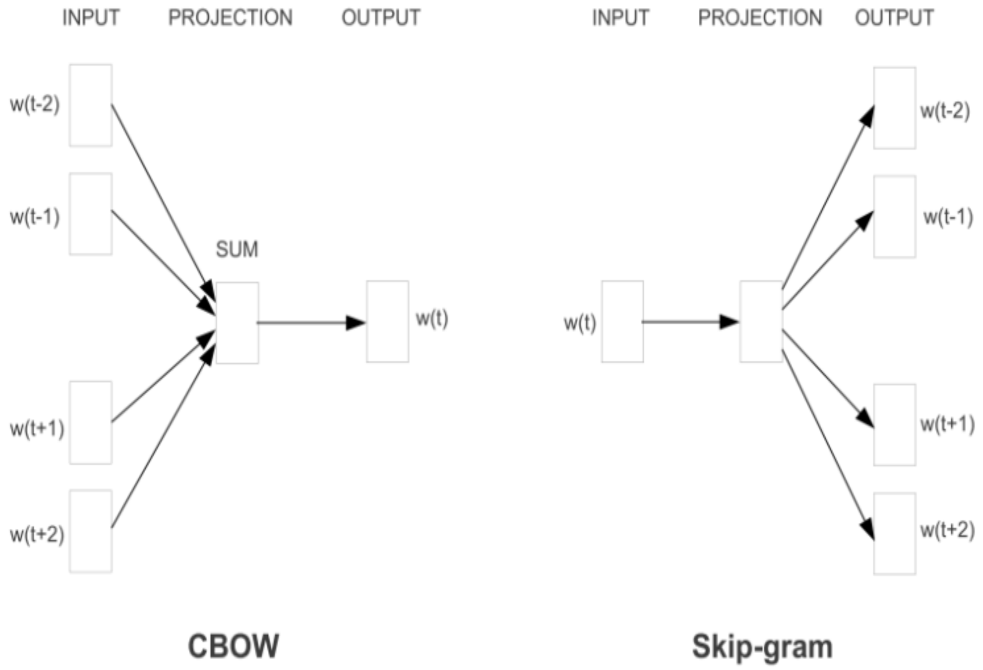


Figure 6.1: Architecture of CBOW and Skip-gram models.

$w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}$ given its target word w_t. The objective function is defined as follows:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p\left(w_{t+j} \mid w_t\right). \tag{6.1}$$

The conditional probability $p\left(w_{t+j} \mid w_t\right)$ is defined using the Softmax function:

$$p\left(w_O \mid w_I\right) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}, \tag{6.2}$$

where v_w and v'_w are the target (input) and the context (output) vector representation of w, and W is the size of unique vocabulary in the corpus. The dot product ${v'_{w_O}}^{\top} v_{w_I}$ compares the similarity of the input and the output candidate word vectors v_{w_I} and v_{w_O} respectively, normalized over the entire vocabulary to give probability distribution.

However, the Softmax function is computationally very expensive in case of big corpus, as the normalization factor in the denominator requires iterating all words in the corpus vocabulary, which is equal to W. Therefore, Mikolov *et al.* [80] in 2013 proposed an alternative called *negative sampling* which uses sigmoid function for binary classification instead of softmax function. More specifically, with negative sampling, the skip-gram model is trained using a binary logistic regression to learn to discriminate the target context words (positive samples) from k randomly-sampled words (negative samples) from the noise distribution $P_n(w)$. The resulting objective function is defined as follows:

$$\log \sigma \left({v'_{w_O}}^\top v_{w_I} \right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma \left(-{v'_{w_i}}^\top v_{w_I} \right) \right], \tag{6.3}$$

where σ is the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{6.4}$$

and k is the number of random negative words sampled from the vocabulary W, it typically has a range of [5,20]. w_i is an i-th negative word drawn from a smoothed unigram probability distribution called noise distribution $P_n(w)$ which it is calculated as follows:

$$P_n(w) = \left(\frac{U(w)}{Z} \right)^{\alpha}, \tag{6.5}$$

where $U(w)$ is the frequency of a word w in the corpus called unigram distribution, Z is a normalization factor, and α is the distribution smoothing hyper-parameter, where $(0 < \alpha \leq 1)$.

6.2.3 Email categorization using Word2Vec representation

We choose to use Word2Vec representation to categorize email documents by domain because it has several advantages over BoW schemes. Word2Vec retains the semantic meaning of different words in a document without losing context information. Another advantage of using the Word2Vec model is the small size of the embedding vector it generates. Each dimension in the embedding vector contains information about one aspect of the word. We do not need huge sparse vectors, unlike the BoW-based approaches.

Figure 6.2 illustrates the steps used to classify email documents by domain. The model takes as input a set of text documents $D = \{d_1, \cdots, d_n\}$ where each document d_i is represented as a vector of words $d_i = \{w_1, \cdots, w_k\}$. The word vectors are then mapped into numerical vectors using Word2Vec. For the words not mapped in word2vec embeddings, spell checking is applied to check the word spelling by using the function Microsoft Word spell checker. If no suggestion of correction is proposed for a given word, a lookup for a similar word is executed. This task is performed in WordNet; it returns a similar word that is semantically related to a given word. Word2vec is again applied to find the word embedding vector corresponding to a similar word. If no similar word is found for a given word in WordNet, the word is deleted. The use of spell check software and WordNet dictionary allows to assign a semantically related vector representation of an unknown word rather than deleting it or assigning it to a random unrelated word vector.

Finally, in this step, we used the pre-trained vectors from Google's Word2Vec embedding model to convert email documents into a vector form. The model contains 300-dimensional vectors for 3 million words and phrases. We choose the SG architecture in this work because it is considered as an efficient method for learning high-quality word embeddings from large-scale unstructured text data like email documents [117]. The Pre-trained Google word2vec model based on SG architecture is available online in [78], but it is only trained in English.

– **Categorization**

To evaluate the performance of the obtained word embeddings from the previous step, various classification algorithms are applied to categorize email documents by domains. Word embeddings are used to generate the document embeddings t_i by averaging all word embeddings in the given document d_i as follows:

$$t_i = \frac{1}{|d_i|} \sum\nolimits_{w \in d_i} T(w) \tag{6.6}$$

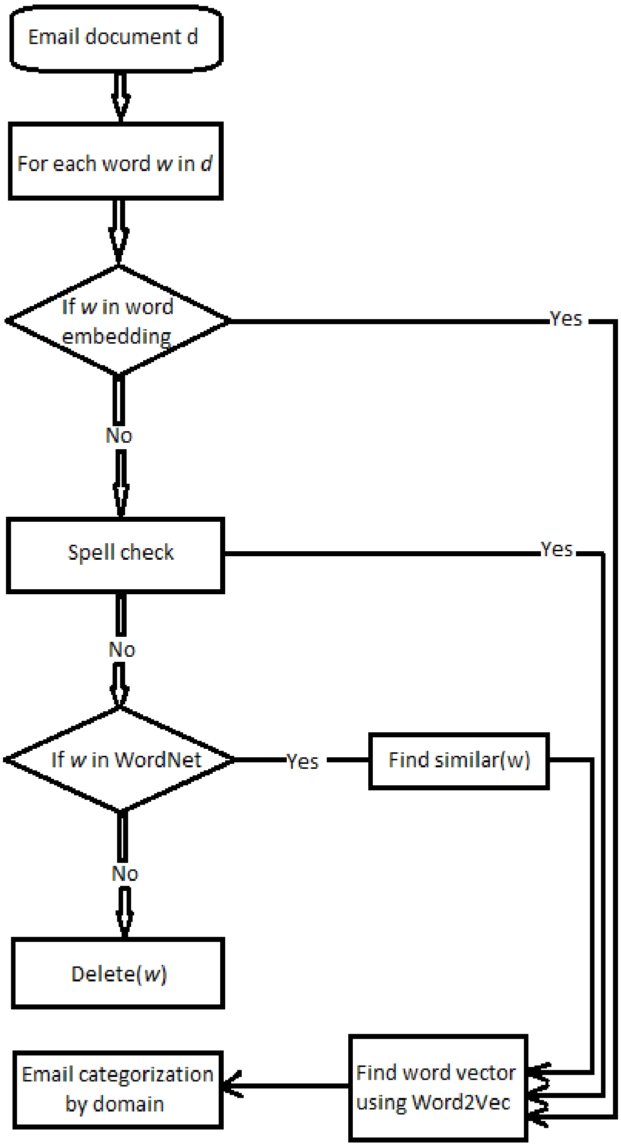


Figure 6.2: Outline of the different steps used for email categorization by domain.

where $T(w)$ is the Word2Vec embedding of the word w.

The obtained results are then used to train a set of classification algorithms to categorize email documents by domain. For email categorization by domain, we compared the following classification algorithms: Naive Bayes, K-Nearest Neighbor (KNN) and Support Vector Machine (SVM) and identified the best classifier giving the most precise email categorization.

6.3 Domain-specific semantic feature extraction

This level aims at extracting hidden semantic features for representing email content. Our final goal is to create a domain-specific semantic representation for an efficient detection of spam. In this regard, we consider two main models, eTVSM and CN2-SD. The model eTVSM is used to represent email documents as a vector of topics in each domain. CN2-SD is applied to represent the returned vectors as a set of rules which are used as input semantic features to build specialized classifiers for detecting domain-specific spam.

6.3.1 eTVSM

The eTVSM is an extension of the Topic-based Vector Space Model (TVSM). The basic premise of TVSM is that documents are represented by a vector of topics rather than terms. Therefore, each term vector is weighted and its direction represents term relevance according to fundamental topics. However, this model does not allow the expression of some linguistic phenomena like hyponymy, metonymy, etc. In contrast, the eTVSM model addresses this problem by using domain ontology to represent different relationships between various terms of each domain [87]. The use of an ontology provides a richer natural language retrieval model that is able to accommodate synonyms, homonyms and other linguistic phenomena [99].

In order to apply eTVSM to our database and represent email documents as operational vector space, we follow the steps used in [99]. We used the Themis[1] implementation

[1] https://code.google.com/archive/p/ir-themis/

which is an in-database data-structure. This data-structure is optimized to store email documents and perform document similarity measurements based on the eTVSM [87]. In addition, Wordnet ontology is used to enhance the eTVSM model.

6.3.2 CN2-SD

The resultant vector representations of the email documents by eTVSM are used in our work by CN2-SD using a procedure proposed in chapter 4 to generate a set of semantic features represented as a set of rules. Each rule has a binary outcome and acts by itself as a weak classifier for spam detection. The rationale behind applying CN2-SD is to check if a combination of one or more topics in a single semantic feature can help distinguish junk mail from legitimate mail. The obtained semantic features are then used to build specialized classifiers for detecting domain-specific spam such as naive Bayes, KNN and SVM.

6.4 Experimental results

We used two public datasets of both spam and ham emails and messages from discussion forums to evaluate our approach. The public datasets are Enron [25] and Ling-spam [3]. The discussion forums were necessary to fill the lack of legitimate messages belonging to the six chosen domains in the public datasets Enron and Ling-spam. The collected dataset was categorized manually by domains according to the text content. We used a total of 8188 emails distributed over the six pre-selected categories: Health, Adult Finance, Education, Computer and Others. Each category includes both spam and ham emails. The overall collection includes 4706 ham emails and 3482 spam emails. The evaluation was carried out by 10-fold cross-validation, with a standard accuracy metric, for both semantic levels.

Table 6.1 shows the evaluation results for the first semantic level using Word2Vec model to categorize emails documents by domains. Based on the obtained results from the used classifiers, we see that Support Vector Machine (SVM) has given the highest

Table 6.1: Quantitative evaluation of machine-learning classifiers for email categorization by domains.

Classifier	Accuracy
KNN	92.4%
Naive Bayes	88.2%
SVM	96.2%

accuracy value 96.2%. We see also that the KNN classifier is quite effective and gives a value of 92.4%. In contrast, naive Bayes has the lowest accuracy as compared to other classifiers with 88.2%.

Table 6.2 shows the results of email spam classification in specific domains using the semantic model eTVSM. Bayesian Classifier performs better than the other applied classifiers. It has given almost in all domains an accuracy of more than 97% which allows an average accuracy of 97.8% for the overall model.

As an overall evaluation of our third proposed approach, we have compared its performance with other semantic-based approaches of the literature and we have selected three methods: eTVSM, Doc2Vec and TBSR-SD (our previous contribution detailed in chapter 4). The obtained results are shown in Table 6.3. As we can see, this approach outperforms eTVSM, TBSR-SD and Doc2Vec since we obtained the highest score with 97.8% of accuracy. It is clear that our proposed approaches gets advantages from specialization of the semantic-domains and domain oriented classifiers compared to the general-classifiers approach of eTVSM and Doc2Vec. Moreover, the high performance of this approach compared to the TBSR-SD approach can be explained by the fact that the used Word2Vec embeddings for domain classification allows to overcome the drawback of feature independence of the BoW model used in the TBSR-SD method. Furthermore, in the second level, TBSR-SD uses independent words as input data for the CN2-SD to generate rules, whereas this approach uses a set of topics produced to generate the rules which carry more semantic content than using words.

Table 6.2: Quantitative evaluation of machine-learning classifiers using eTVSM for spam detection in specific domains.

	Classifier	Accuracy
Computer	KNN	90.1%
	Naive Bayes	97.9%
	SVM	89.3%
Adult	KNN	97.6%
	Naive Bayes	97.8%
	SVM	90.4%
Education	KNN	96.6%
	Naive Bayes	98.9%
	SVM	92.3%
Finance	KNN	95.4%
	Naive Bayes	97.8%
	SVM	91.5%
Health	KNN	97.6%
	Naive Bayes	98.5%
	SVM	94.9%
Others	KNN	91.6%
	Naive Bayes	95.9%
	SVM	86.7%
Average	KNN	94.8%
	Naive Bayes	97.8%
	SVM	90.9%

6.5 Overall comparison of our contributions

Summary table 6.4 compares the performance of our three contributions presented in this thesis with the same data set to get an overall idea of the obtained results. Note that, this dataset is large enough to ensure statistically significant result. The best result obtained by the first contribution using the automatic extraction of semantic features is given by the Random Forest classifier with an accuracy of 97.84%. This contribution shows that using semantic features for each specific domain is more efficient for spam detection than using

Table 6.3: Comparative evaluation with other methods.

Model	Classifier	Accuracy
eTVSM [99]	KNN	93.4%
	Naive Bayes	97.6%
	SVM	97.6%
TBSR-SD [98]	KNN	95.5%
	Naive Bayes	97.6%
	SVM	93.2%
Doc2Vec [123]	KNN	75.3%
	Naive Bayes	65.8%
	SVM	84.6%
Our approach	KNN	94.8%
	Naive Bayes	97.8%
	SVM	90.9%

holistic and general-purpose semantic features and provides an efficient representation of internal semantic structure of email content. In addition, email representations based on the automatic extraction of intermediate semantic structures allow to better capture the different subtleties of spam compared to methods based on the word semantics (e.g., Word2Vec, eTVSM) or document semantics (e.g., Doc2Vec). However, most antispam filters based only on automatic extraction of semantic features need to be enriched by expert knowledge to improve the performance of these filters.

The second contribution addresses this problem by proposing a hybrid method that combines expert and automatic knowledge. The method outperforms the first contribution with an accuracy of 98,53% and a precision of 98,66% given by Adaboost. This experiment demonstrates that combining automatic and expert knowledge allows a richer set of semantic features and gives a powerful tool for spam detection. However, automatic features are made up of a set of terms that can sometimes change the meaning of text data. For example, the feature $r : free \wedge mouse \rightarrow spam$ can be true if the text content talks about computers and can be false if the text talks about animals. Therefore, the third contribution proposes to use more elaborated semantic features to group words with similar meanings. This study increases the prediction capacity of the classifiers compared to the two other contributions where the best result is given by naive Bayes with an accuracy of 98.89%.

Table 6.4: Comparative evaluation of our three contributions.

Contributions	Classifier	Accuracy	Precision
Contribution 1 [98]	KNN	0.9600	0.9552
	Naive Bayes	0.9792	0.9751
	Decision tree	0.9652	0.9572
	Adaboost	0.9735	0.9744
	Random forest	0.9784	0.9758
	SVM	0.9703	0.9712
Contribution 2 [97]	KNN	0.9681	0.9746
	Naive Bayes	0.9831	0.9839
	Decision tree	0.9704	0.9786
	Adaboost	0.9853	0.9866
	Random forest	0.9783	0.9805
	SVM	0.9769	0.9798
Contribution 3 [96]	KNN	0.9723	0.9743
	Naive Bayes	0.9889	0.9856
	Decision tree	0.9753	0.9766
	Adaboost	0.9872	0.9887
	Random forest	0.9897	0.9898
	SVM	0.9799	0.9814

6.6 Conclusion

We proposed a new approach using semantic-based model to detect spam emails in specific domains. For this purpose, we use two semantic level analysis. The first level categorizes email documents by domains using Word2Vec word embedding. The ultimate goal of this level is to obtain the global subject of the email content to allow the extraction of relevant semantic features related to a domain. The second level uses the eTVSM model to represent the email documents as a vector of topics and the CN2-SD to create semantic features. The semantic features are then used to train machine learning algorithms for spam detection. The experimental results showed that the application of the topic extraction model in specific domains allows an efficient detection of spam emails.

7

Conclusion and future work

This thesis explores how semantic information conveyed by email text contents can be used more effectively to distinguish between spam and ham emails. Traditional approaches to spam detection based on text mining are confined to detect spam using mostly lexical and general-purpose semantic information. However, our approach analyzes the email contents at two semantic levels to build domain-specific spam classifiers that substantially yield more effective spam detection. This approach exploits in-depth the information contained within the text by distinguishing the semantic context within different domains (categories) targeted by spammers.

Hence, the present research work has given rise to three main contributions. In the first contribution, we automatically categorize the domain (subject) of the analyzed emails (e.g., health, education, finance, etc.) to enable a separate conceptual view of spams in each domain and apply a domain-specific spam detection by using an automatically generated set of semantic features represented in the form of logical rules. These semantic features provide a precise description of each spams domain, allowing better detection. We show that the proposed method provides an efficient representation of internal semantic structure of email contents and allows for more precise and interpretable spam filtering results compared to existing methods. In the second contribution, we enhanced our detection technique by considering a hybrid approach, combining manually specified and automatically generated rules. The automatically built rules capture efficiently the basic semantics, while manually specified rules allow semantics tuning by incorporating domain-specific knowledge of experts and end-users. We have demonstrated that combining manual and automatic types of rules offers a richer set of semantic features to enforce spam detection. In the third contribution, we wanted to explore how the use of semantic models based on deep learning and ontologies can affect spam filtering in the two levels. The experimental study shows promising results in term of the precision of the spam detection.

In future work, we intend to enhance our approach in several ways. We can explore the use of new semantic features, such as message polarity, called sentiment feature, to automatically extract opinions or emotions from email content. An email document's polarity is identified using sentiment analysis that attempts to label the given message into positive, negative, or neutral opinion. Furthermore, to maintain our spam filtering efficiency in the long term, it will be necessary to create a procedure for online and efficient updating of the semantic feature sets and classifiers in all domains. This can be interesting, for example, in the scenario where our system is continuously provided with user feedback. In other words, by considering a relevant feedback algorithm, we can obtain a dynamic system that is refined over time and adapted to the user's needs. In addition, dealing with streaming evolved data may require an updating of the semantic rules. This phenomenon is known as concept drift and can be tackled in several ways, such as the use of rule-based online algorithms or evolutionary rule learning approaches. The latter allows the adaptation of new data to existing knowledge. Finally, to handle a large set of rules after the rule combination step, it could be interesting to elaborate an evaluation step to sort and select the most significant rules by using rule quality measures.

Bibliography

[1] The Spamhaus Project. https://www.spamhaus.org/, 1998–2020. Date accessed: April 2019.

[2] The porter stemming algorithm. http://tartarus.org/martin/PorterStemmer/, note = Date accessed: june 2014, 2006.

[3] Ling-Spam corpus. http://www.stat.purdue.edu/~mdw/598/datasets.html, 2009. Date accessed: May 2017.

[4] Charu C Aggarwal and ChengXiang Zhai. *Mining text data.* Springer Science & Business Media, 2012.

[5] Rubayyi Alghamdi and Khalid Alfalqi. A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1), 2015.

[6] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, and Constantine D Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167, 2000.

[7] Wakama As. It news africa. 10 tricks used by spammers to get into your inbox. http://www.itnewsafrica.com/2009/11/10-tricks-used-by-spammers-to-get-into-your-inbox/, 2009. Date accessed: November 2017.

[8] Aliaksandr Barushka and Petr Hajek. Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*, 48(10):3538–3556, 2018.

[9] M Basavaraju and Dr R Prabhakar. A novel method of spam mail detection using text based clustering approach. *International Journal of Computer Applications*, 5(4):15–25, 2010.

[10] Alexy Bhowmick and Shyamanta M Hazarika. Machine learning for e-mail spam filtering: review, techniques and trends. *arXiv preprint arXiv:1606.01042*, 2016.

[11] Alexy Bhowmick and Shyamanta M Hazarika. E-mail spam filtering: a review of techniques and trends. In *Advances in Electronics, Communication and Computing*, pages 583–590. Springer, 2018.

[12] Hanif Bhuiyan, Akm Ashiquzzaman, Tamanna Islam Juthi, Suzit Biswas, and Jinat Ara. A survey of existing e-mail spam filtering methods considering machine learning techniques. *Global Journal of Computer Science and Technology*, 2018.

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[14] A Selman Bozkir, Esra Sahin, Murat Aydos, Ebru Akcapinar Sezer, and Fatih Orhan. Spam e-mail classification by utilizing n-gram features of hyperlink texts. In *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–5. IEEE, 2017.

[15] Andrej Bratko, Gordon V Cormack, Bogdan Filipič, Thomas R Lynam, and Blaž Zupan. Spam filtering using statistical data compression models. *Journal of machine learning research*, 7(Dec):2673–2698, 2006.

[16] Francesco Camastra and Alessandro Vinciarelli. *Machine learning for audio, image and video analysis: theory and applications*. Springer, 2015.

[17] Godwin Caruana and Maozhen Li. A survey of emerging approaches to spam filtering. *ACM Computing Surveys (CSUR)*, 44(2):1–27, 2008.

[18] Chuanliang Chen, Yingjie Tian, and Chunhua Zhang. Spam filtering with several novel bayesian classifiers. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[19] S Chitra, K Jayanthan, S Preetha, and RU Shankar. Predicate based algorithm for malicious web page detection using genetic fuzzy systems and support vector machine. *International Journal of Computer Applications*, 40(10):13–19, 2012.

[20] Ali Çıltık and Tunga Güngör. Time-efficient spam e-mail filtering using n-gram models. *Pattern Recognition Letters*, 29(1):19–33, 2008.

[21] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *European Working Session on Learning*, pages 151–163. Springer, 1991.

[22] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine learning*, 3(4):261–283, 1989.

[23] J Clement. Spam: share of global email traffic 2014-2020. https://www.statista.com/statistics/420391/spam-email-traffic-share/, 2020. Date accessed: November 2020.

[24] William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[25] WW Cohen. Enron email dataset. https://www.cs.cmu.edu/~./enron/, 2009. Date accessed: May 2017.

[26] Gordon V Cormack. *Email spam filtering: A systematic review*. Now Publishers Inc, 2008.

[27] Gordon V Cormack and Thomas R Lynam. TREC 2005 spam track overview. In *TREC*, pages 500–274, 2005.

[28] Lorrie Faith Cranor and Brian A LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

[29] Geoff Cumming and Robert Calin-Jageman. *Introduction to the new statistics: Estimation, open science, and beyond.* Routledge, 2016.

[30] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, et al. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[31] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa, et al. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[32] Sumit Das, Aritra Dey, Akash Pal, and Nabamita Roy. Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, 115(9), 2015.

[33] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[34] Xuelian Deng, Yuqing Li, Jian Weng, and Jilian Zhang. Feature selection for text classification: A review. *Multimedia Tools and Applications*, 78(3):3797–3816, 2019.

[35] Melvin Diale, Christiaan Van Der Walt, Turgay Celik, and Abiodun Modupe. Feature selection and support vector machine hyper-parameter optimisation for spam detection. In *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pages 1–7. IEEE, 2016.

[36] Enaitz Ezpeleta, Urko Zurutuza, and José María Gómez Hidalgo. Does sentiment analysis help in bayesian spam filtering? In *International Conference on Hybrid Artificial Intelligence Systems*, pages 79–90. Springer, 2016.

[37] Al Mahmood Fahad. spellcheck. https://www.mathworks.com/matlabcentral/fileexchange/5378-spellcheck, 2004. Date accessed: January 2017.

[38] Y Feng and H Zhou. An effective and efficient two-stage dimensionality reduction algorithm for content-based spam filtering. *J. Comput. Inf. Syst*, 9(4):1407–1420, 2013.

[39] Artur J Ferreira and Mário AT Figueiredo. Boosting algorithms: A review of methods, theory, and applications. In *Ensemble machine learning*, pages 35–85. Springer, 2012.

[40] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data.* Cambridge University Press, 2012.

[41] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrač. *Foundations of rule learning.* Springer Science & Business Media, 2012.

[42] Walaa Gad and Sherine Rady. Email filtering based on supervised learning and mutual information feature selection. In *2015 Tenth International Conference on Computer Engineering & Systems (ICCES)*, pages 147–152. IEEE, 2015.

[43] Wilfried N Gansterer, Andreas GK Janecek, and Robert Neumayer. Spam filtering based on latent semantic indexing. In *Survey of Text Mining II*, pages 165–183. Springer, 2008.

[44] Fabio A Gonzalez and Eduardo Romero. *Biomedical image analysis and machine learning technologies: Applications and techniques: Applications and techniques.* IGI Global, 2009.

[45] José Gordillo and Eduardo Conde. An HMM for detecting spam mail. *Expert systems with applications*, 33(3):667–682, 2007.

[46] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In *Handbook on ontologies*, pages 1–17. Springer, 2009.

[47] Darya Gudkova, Maria Vergelis, and Nadezhda Demidova. Spam and phishing in Q2 2015. *Kaspersky Lab*, 2015.

[48] Darya Gudkova, Maria Vergelis, Nadezhda Demidova, and Tatyana Shcherbakova. Spam and phishing in Q2 2016. *Kaspersky Lab*, 18, 2016.

[49] Thiago S Guzella and Walmir M Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[50] Lawrence O Hall, Nitesh Chawla, Kevin W Bowyer, et al. Combining decision trees learned in parallel. In *Working Notes of the KDD-97 Workshop on Distributed Data Mining*, pages 10–15, 1998.

[51] Shuang Hao, Nadeem Ahmed Syed, Nick Feamster, Alexander G Gray, and Sven Krasser. Detecting spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *USENIX security symposium*, volume 9, 2009.

[52] Christian F Hempelmann and Vikas Mehra. Baseline semantic spam filtering. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 273–276. IEEE, 2011.

[53] Francisco Herrera, Cristóbal José Carmona, Pedro González, and María José Del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525, 2011.

[54] Amir Herzberg. DNS-based email sender authentication mechanisms: A critical review. *Computers & security*, 28(8):731–742, 2009.

[55] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[56] Xuan Hu, Banghuai Li, Yang Zhang, Changling Zhou, and Hao Ma. Detecting compromised email accounts from the perspective of graph topology. In *Proceedings of the 11th International Conference on Future Internet Technologies*, pages 76–82, 2016.

[57] Yu Jiang, Ni Zhang, and Binxing Fang. An email geographic path-based technique for spam filtering. In *2007 International Conference on Computational Intelligence and Security (CIS 2007)*, pages 750–753. IEEE, 2007.

[58] Sumedh Kadam, Aayush Gala, Pritesh Gehlot, Aditya Kurup, and Kranti Ghag. Word embedding based multinomial naive bayes algorithm for spam filtering. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–5. IEEE, 2018.

[59] Mehran Kamkarhaghighi, Eren Gultepe, and Masoud Makrehchi. Deep learning for document representation. In *Handbook of Deep Learning Applications*, pages 101–110. Springer, 2019.

[60] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16(06):1047–1067, 2007.

[61] Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, and Mamoun Alazab. A comprehensive survey for intelligent spam email detection. *IEEE Access*, 7:168261–168295, 2019.

[62] Jongwan Kim, Dejing Dou, Haishan Liu, and Donghwi Kwak. Constructing a user preference ontology for anti-spam mail systems. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 272–283. Springer, 2007.

[63] Irena Koprinska, Josiah Poon, James Clark, and Jason Chan. Learning to classify e-mail. *Information Sciences*, 177(10):2167–2187, 2007.

[64] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[65] Carlos Laorden, Igor Santos, Borja Sanz, Gonzalo Alvarez, and Pablo G Bringas. Word sense disambiguation for spam filtering. *Electronic Commerce Research and Applications*, 11(3):290–298, 2012.

[66] Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5(Feb):153–188, 2004.

[67] Barry Leiba, Joel Ossher, Vadakkedathu Thomas Rajan, Richard Segal, and Mark N Wegman. Recognizing spam email, October 1 2013. US Patent 8,549,081.

[68] Barry Leiba, Joel Ossher, VT Rajan, Richard Segal, and Mark N Wegman. SMTP path analysis. In *CEAS*. Citeseer, 2005.

[69] Luyang Li, Bing Qin, Wenjing Ren, and Ting Liu. Document representation and feature combination for deceptive spam review detection. *Neurocomputing*, 254:33–41, 2017.

[70] Linqing Liu, Yao Lu, Ye Luo, Renxian Zhang, Laurent Itti, and Jianwei Lu. Detecting" smart" spammers on social network: A topic model approach. *arXiv preprint arXiv:1604.08504*, 2016.

[71] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[72] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997.

[73] Ben Medlock. An adaptive, semi-structured language model approach to spam filtering on a new corpus. In *CEAS*, 2006.

[74] Leonhardt Megan. Nigerian prince' email scams still rake in over $700,000 a year-here's how to protect yourself. https://www.cnbc.com/2019/04/18/

nigerian-prince-scams-still-rake-in-over-700000-dollars-a-year.html, 2019. Date accessed: May 2020.

[75] Jose R Mendez, Tomas R Cotos-Yanez, and David Ruano-Ordas. A new semantic-based feature selection method for spam filtering. *Applied Soft Computing*, 76:89–104, 2019.

[76] Guyue Mi, Pengtao Zhang, and Ying Tan. Feature construction approach for email categorization based on term space partition. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[77] Si Miao, Xuyi Zhang, Yujie Han, Wen Sun, Chunjiang Liu, and Shan Yin. Random forest algorithm for the relationship between negative air ions and environmental factors in an urban park. *Atmosphere*, 9(12):463, 2018.

[78] Tomas Mikolov. Word2vec software. https://code.google.com/archive/p/word2vec, 2013. Date accessed: June 2018.

[79] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[80] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[81] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

[82] Dunja Mladenić. Feature subset selection in text-learning. In *European conference on machine learning*, pages 95–100. Springer, 1998.

[83] Pinaki Prasad Guha Neogi, Amit Kumar Das, Saptarsi Goswami, and Joy Mustafi. Topic modeling for text classification. In *Emerging Technology in Modelling and Graphics*, pages 395–407. Springer, 2020.

[84] Evelien Otte and Ronald Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, 28(6):441–453, 2002.

[85] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[86] Noemí Pérez-Díaz, David Ruano-Ordas, Jose R Mendez, Juan F Galvez, and Florentino Fdez-Riverola. Rough sets for spam filtering: Selecting appropriate decision rules for boundary e-mail classification. *Applied Soft Computing*, 12(11):3671–3682, 2012.

[87] Artem Polyvyanyy and Dominik Kuropka. A quantitative evaluation of the enhanced topic-based vector space model. 2007.

[88] Aziz Qaroush, Ismail M Khater, and Mahdi Washaha. Identifying spam e-mail based-on statistical header features and sender behavior. In *Proceedings of the CUBE International Information Technology Conference*, pages 771–778, 2012.

[89] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[90] J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014.

[91] Venkatesh Ramanathan and Harry Wechsler. phishGILLNET—phishing detection methodology using probabilistic latent semantic analysis, adaboost, and co-training. *EURASIP Journal on Information Security*, 2012(1):1, 2012.

[92] Yafeng Ren and Donghong Ji. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385:213–224, 2017.

[93] D Karthika Renuka, T Hamsapriya, M Raja Chakkaravarthi, and P Lakshmi Surya. Spam classification based on supervised learning using machine learning techniques. In *2011 International Conference on Process Automation, Control and Computing*, pages 1–7. IEEE, 2011.

[94] Karthika D Renuka and P Visalakshi. Latent semantic indexing based svm model for email spam classification. 2014.

[95] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin, 1998.

[96] Nadjate Saidani, Kamel Adi, and Mohand Said Allili. Semantic representation based on deep learning for spam detection. In *International Symposium on Foundations and Practice of Security*, pages 72–81. Springer, 2019.

[97] Nadjate Saidani, Kamel Adi, and Mohand Said Allili. A semantic-based classification approach for an enhanced spam detection. *Computers & Security*, 94:101716, 2020.

[98] Nadjate Saidani, Kamel Adi, and Mouhand Said Allili. A supervised approach for spam detection using text-based semantic representation. In *International Conference on E-Technologies*, pages 136–148. Springer, 2017.

[99] Igor Santos, Carlos Laorden, Borja Sanz, and Pablo G Bringas. Enhanced topic-based vector space model for semantics-aware spam filtering. *Expert Systems with applications*, 39(1):437–444, 2012.

[100] Enrique Puertas Sanz, José María Gómez Hidalgo, and José Carlos Cortizo Pérez. Email spam filtering. *Advances in computers*, 74:45–114, 2008.

[101] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, 2003.

[102] David Sculley and Gabriel M Wachman. Relaxed online SVMs for spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422, 2007.

[103] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[104] Vishal Kumar Singh and Shweta Bhardwaj. Spam mail detection using classification techniques and global training set. In *Intelligent Computing and Information and Communication*, pages 623–632. Springer, 2018.

[105] Long Song, Raymond Yiu Keung Lau, Ron Chi-Wai Kwok, Kristijan Mirkovski, and Wenyu Dou. Who are the spoilers in social media marketing? incremental learning of latent semantics for social spam detection. *Electronic commerce research*, 17(1):51–81, 2017.

[106] Thamarai Subramaniam, Hamid A Jalab, and Alaa Y Taqa. Overview of textual anti-spam filtering techniques. *International Journal of Physical Sciences*, 5(12):1869–1882, 2010.

[107] Ying Tan. *Anti-Spam Techniques Based on Artificial Immune System*. CRC Press, 2016.

[108] Ying Tan, Quanbin Wang, and Guyue Mi. Ensemble decision for spam detection using term space partition approach. *IEEE transactions on cybernetics*, 50(1):297–309, 2018.

[109] Guanting Tang, Jian Pei, and Wo-Shun Luk. Email mining: tasks, common techniques, and tools. *Knowledge and Information Systems*, 41(1):1–31, 2014.

[110] SpamAssassin Team. The apache spamassassin project, 2012.

[111] The MathWorks Team. Introducing machine learning. https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/i/88174_92991v00_machine_learning_section1_ebook.pdf, 2004. Date accessed: April 2020.

[112] Zahra S Torabi, Mohammad H Nadimi-Shahraki, and Akbar Nabiollahi. Efficient support vector machines for spam detection: a survey. *International Journal of Computer Science and Information Security*, 13(1):11, 2015.

[113] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[114] S Venkatraman, B Surendiran, and P Arun Raj Kumar. Spam e-mail classification for the internet of things environment using semantic similarity approach. *The Journal of Supercomputing*, 76(2):756–776, 2020.

[115] Chalee Vorakulpipat, Vasaka Visoottiviseth, and Siwaruk Siwamogsatham. Polite sender: A resource-saving spam email countermeasure based on sender responsibilities and recipient justifications. *Computers & Security*, 31(3):286–298, 2012.

[116] Hongling Wang, Gang Zheng, and Yueshun He. The improved bayesian algorithm to spam filtering. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*, pages 37–44. Springer, 2015.

[117] Peng Wang, Jiaming Xu, Bo Xu, Chenglin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 352–357, 2015.

[118] Louis Wehenkel. On uncertainty measures used for decision tree induction. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, page 6, 1996.

[119] P Wood, B Nahorney, K Chandrasekar, S Wallace, K Haley, et al. Symantec internet security threat report. *Symantec Corporation, Tech. Rep.*, 21, 2016.

[120] WordNet. A lexical database for english, 2013. Accessed: 2015. URL: https://wordnet.princeton.edu/.

[121] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87. Springer, 1997.

[122] Chih-Hung Wu. Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. *Expert systems with Applications*, 36(3):4321–4330, 2009.

[123] Tingmin Wu, Shigang Liu, Jun Zhang, and Yang Xiang. Twitter spam detection based on deep learning. In *Proceedings of the australasian computer science week multiconference*, pages 1–8, 2017.

[124] Tingmin Wu, Sheng Wen, Yang Xiang, and Wanlei Zhou. Twitter spam detection: Survey of new approaches and comparative study. *Computers & Security*, 76:265–284, 2018.

[125] Qian Xu, Evan Wei Xiang, Qiang Yang, Jiachun Du, and Jieping Zhong. Sms spam detection using noncontent features. *IEEE Intelligent Systems*, 27(6):44–51, 2012.

[126] Hong Yang, Qihe Liu, Shijie Zhou, and Yang Luo. A spam filtering method based on multi-modal fusion. *Applied Sciences*, 9(6):1152, 2019.

[127] Bin Zheng, David C McLean, and Xinghua Lu. Identifying biological concepts from a protein-related corpus with a probabilistic topic model. *BMC bioinformatics*, 7(1):58, 2006.

[128] Wei Zong, Feng Wu, Lap-Keung Chu, and Domenic Sculli. A discriminative and semantic feature selection method for text categorization. *International Journal of Production Economics*, 165:215–222, 2015.