Computers & Security 94 (2020) 101716

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A semantic-based classification approach for an enhanced spam detection

Nadjate Saidani*, Kamel Adi, Mohand Saïd Allili

Département d'Informatique et d'Ingénierie, Université du Québec en Outaouais, Gatineau, J8Y 3G5, QC, Canada

ARTICLE INFO

Article history:
Received 29 June 2019
Revised 31 December 2019
Accepted 7 January 2020
Available online 9 January 2020

Keywords:
Spam detection
Domain-specific analysis
Semantic features
Multilevel analysis
Classification

ABSTRACT

In this paper, we explore the use of a text semantic analysis to improve the accuracy of spam detection. 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.) to enable a separate conceptual view for spams in each domain. In the second level, we combine a set of manually-specified and automatically-extracted semantic features for spam detection in each domain. These features are meant to summarize the email content into compact topics discriminating spam from non-spam emails in an efficient way. We show that the proposed method enables a better spam detection compared to existing methods based on *bag-of-words* (BoW) and semantic content, and leads to more interpretable results.

1. Introduction

Unsolicited or unwanted emails, commonly known as spam emails, are one of harmful threats to Internet security. Spam emails overloads the mailboxes with unwanted messages causing a loss in network bandwidth and storage space, and favoring fast distribution of false information and spread of malicious codes. According to a recent study of the Kaspersky Lab (Gudkova et al., 2017), spam represented over 56% of the total email traffic. To deal with this problem, several spam detection methods have been proposed in the recent years (Caruana and Li, 2012; Dada et al., 2019; Hu et al., 2010; Wu et al., 2018). Two main types of information have been used in these methods, namely non-content and content based information (Hu et al., 2010). Non-content information refers to the use of email headers and related sender information such as email sender address domain (IP address), reputation, writing style and sending time, for detecting spams. Content-based information refers to using textual information contained in the email bodies and subjects. Our work focuses on content-based approaches for spam detection.

Recently, machine learning techniques have shown a tremendous success for content-based spam detection (Bhowmick and Hazarika, 2016; Caruana and Li, 2012). They mainly consist in learning classification models mapping email features (e.g., words, n-grams, etc.) into spam/ham classes. Email features can usually be extracted manually using hand-crafted rules, also called knowledge engineering, or automatically using text mining techniques (Bhowmick and Hazarika, 2016; Dada et al., 2019). In the former, rules are built by experts and regularly updated to maintain the efficiency of spam detection systems. In the latter, a corpus of annotated emails is automatically analyzed using text mining algorithms to extract useful information such as words, HTML markup and various document statistics, enabling spam discrimination (Bhowmick and Hazarika, 2016). From these features, an email representation can be constructed in the form of *bag-of-words* (BoW) models (Aggarwal and Zhai, 2012) where unstructured set of word tokens are used to discriminate between spam and legitimate messages. BoW models assume word tokens independent which can prevent them from conveying useful semantic information for email representation.

In general, spam content can drastically vary between different domains targeted by the spammers. For example, in spam campaigns related to the Health domain, spam content is mainly oriented to medicine or false therapy campaigns advertisement, whereas in the Finance domain, spam can carry advertisement for dubious financial services and products. Then, using domain-specific spam features can be more efficient than general-purpose ones for spam discrimination. Furthermore, individual word tokens can be plagued with polysemy and sometimes intentionally modified by spammers to prevent their detection by spam filters (Santos et al., 2012). Therefore, using more elaborate features than mere words such as in BoW is required for better discriminating between legitimate and spam emails. Having a richer semantic support enabling to distinguish between emails containing identical sets of words but different semantic structures is one way of

* Corresponding author.
E-mail address: sain06@uqo.ca (N. Saidani).

https://doi.org/10.1016/j.cose.2020.101716
0167-4048/

achieving this objective. Recently, the enhanced *topic-based vector space model* (eTVSM) has been proposed in Santos et al. (2012) to extract latent semantic aspects from text messages for spam classification. Other methods based on *latent semantic indexing* (LSI) (Renuka and Visalakshi, 2014) and *latent dirichlet allocation* (LDA) (Liu et al., 2016) have been proposed for spam detection. These methods have reported a better performance than BoW. However, since the semantic topics are extracted independently of domains, they do not guarantee optimal performance for spam detection within each domain.

In this paper, we propose an approach incorporating semantic analysis for spam detection. Our approach considers semantic information at two levels. In the first level, emails are categorized into subject domains (e.g., Finance, Health, etc.), which enables tailoring semantic features for each specific domain for spam detection. To categorize emails into domains, we trained a supervised classifier with a selected vocabulary maximizing domain discrimination. In the second level, we build a semantic feature space for discriminating emails in each domain. The semantic feature space consists of a mix of manually-specified and automatically-extracted rules from labeled emails. Manually-specified rules are encoded by regular expressions, whereas automatically-extracted ones are generated using the CN2-SD method (Lavrac et al., 2004). Each rule has a binary class outcome (spam/ham) and acts by itself as a weak classifier for spam detection. The combination of these rules produces a strong classifier enabling a robust and accurate spam detection. Our main contributions are summarized in the following points:

- We propose a two-level semantic analysis of emails to enable domain-specific spam detection. The first level aims at categorizing the broad subject of emails to define a domain-specific spam characterization. The second level extracts semantic features in each domain in the form of rules distinguishing between spam and legitimate messages.
- Our approach uses a hybrid scheme composed of manually-specified and automatically-extracted of semantic rules. Manually-specified rules allow to incorporate domain specific knowledge by experts or end-users, and can be continuously enriched. Automatic rules extraction is based on binary classification trees and exploit the CN2-SD method (Lavrac et al., 2004) for assessing the strength of each rule for spam/ham discrimination. We also propose an algorithm eliminating redundancy and conflicts between rules.
- We validate our approach using extensive experiments on a large corpus of emails composed of six different domains, namely: Computer, Adult, Education, Finance, Health and Others. These domains are among the most targeted by spammers (Gudkova et al., 2015; 2016; Symantec, 2016). In addition of testing several classification techniques for our semantic features, we compared our approach to the BoW-SF (Androutsopoulos et al., 2000), the Doc2Vec (Wu et al., 2017) and the eTVSM (Santos et al., 2012) approaches. We demonstrate that our approach yields a higher performance for spam detection compared to the other methods.

Early results of this work have been published in Saidani et al. (2017). Herein, we give a more in-depth theoretical analysis and a broader literature review. We also propose an extension of our semantic features using manually-specified rules for spam classification. Finally, we present a more extensive validation results including comparisons with other classification methods and datasets. In comparison to our early work (Saidani et al., 2017), our performance has been significantly improved.

The rest of this paper is organized as follows: Section 2 gives a brief overview of related work. Section 3 details the proposed approach for efficient spam detection. Section 4 discusses the evaluation results. Finally, Section 5 presents the conclusion and some directions for future work.

2. Related Work

In recent years, content-based spam detection has received a lot of attention from the research community, where numerous approaches have been proposed. Here, we present an overview of these approaches for spam detection which can be classified into three broad categories: rule-based approaches, vector-space-based approaches and semantic-based approaches. Hereafter, we discuss each of these categories.

2.1. Rule-based approaches

Rule-based approaches are among the earliest proposed solutions for spam detection (Bhowmick and Hazarika, 2016; Cormack, 2007). Commonly used techniques within this category are *whitelist, blacklist* and *hand-crafted* rules, which can be considered as signatures for spam identification (Cormack, 2007; Herzberg, 2009). The *Whitelist* method can recognize trusted sender addresses while the *Blacklist* method can block unsolicited emails based on fake sender addresses (Vorakulpipat et al., 2012), known as spammer addresses. Hand-crafted rules method uses 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.) (Cormack, 2007). These rules are in general coded using compact regular expressions to specify complex text patterns (Pérez-Díaz et al., 2012). This technique has been used, for example, in the SpamAssassin system (The Apache SpamAssassin Project, 2014) and by some other researchers such as Sahami et al. (1998) and Schleimer et al. (2003). However, the set of rules need to be continuously updated to maintain the effectiveness of the method. Unfortunately, this limits the applicability of the method in a context of dynamically evolving spam content, which requires developing techniques for automatic rule generation.

2.2. Vector-space-based approaches

With recent advances in machine learning, a significant effort has been made for developing automatic approaches for spam detection (Bhowmick and Hazarika, 2016; Bhuiyan et al., 2018; Caruana and Li, 2012; Dada et al., 2019). In this context, several classification-based methods have demonstrated their effectiveness for spam detection using techniques such as the naive Bayes (Kadam et al., 2018; Singh and Bhardwaj, 2018; Wang et al., 2015), the decision trees (Pérez-Díaz et al., 2012) and the *support vector machine* (SVM) (Singh and Bhardwaj, 2018; Torabi et al., 2015). These methods cast spam filtering as a classical text categorization problem (Caruana and Li, 2012), where a set of relevant features (e.g., words, n-grams) are extracted from email messages using *natural language processing* (NLP) techniques. The most common model for features representation is the BoW (Caruana and Li, 2012; Sahami et al., 1998). This model describes the text as a set of unstructured and independent tokens where each token represents a separate dimension in the vector space used for email classification. For example, in the BoW-SF (*bag-of-words spam filtering*) method (Androutsopoulos et al., 2000), the authors use the BoW model to represent email documents which are classified using the naive Bayes algorithm. However, this method suffers from the problem of polysemy and lack of semantic representation. To mitigate this issue, the n-grams model (Bozkir et al., 2017) extends the BoW model by considering sequences of words or characters

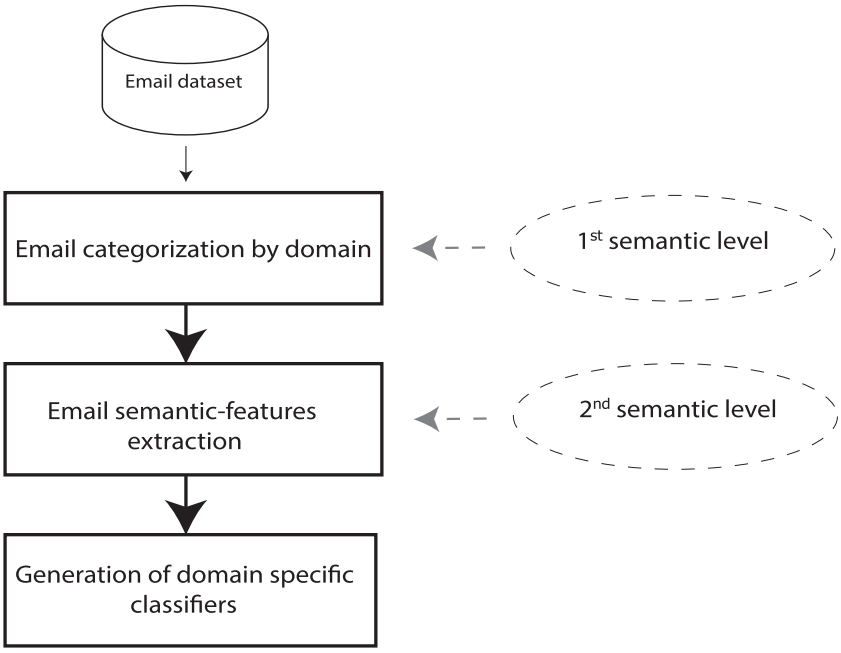

Fig. 1. Layout of steps composing our approach.

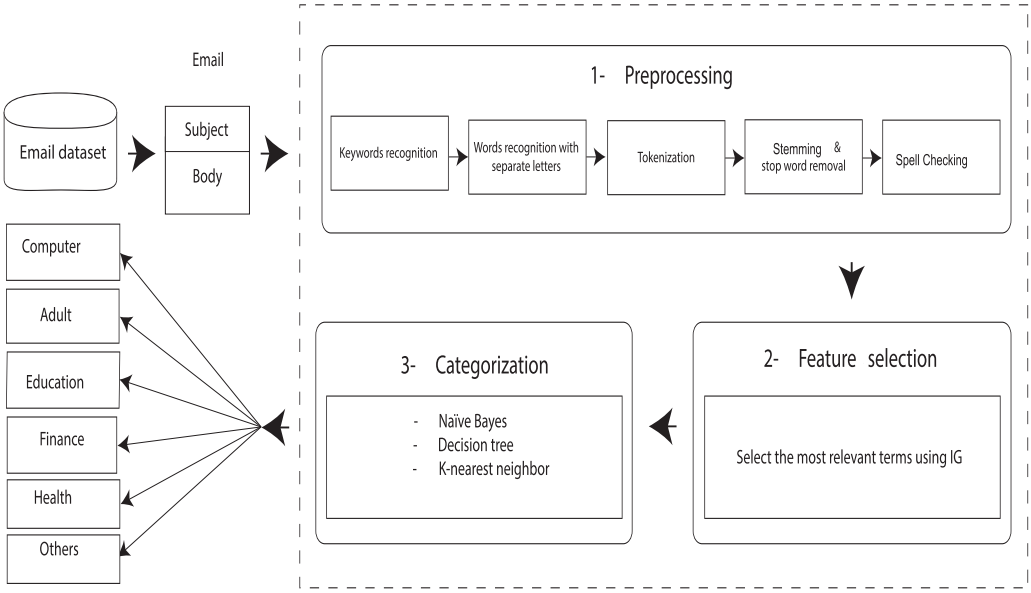

Fig. 2. Outline of the different steps used for email categorization by domain.

occurring in the text as basic tokens, which are used to discriminate between spam and legitimate emails. Contrarily to individual words, n-grams carry more semantic information. However, their representation is usually very high-dimensional and the representation space is sparse. In addition, they do not carry high-level semantic information since n-grams are extracted independently.

2.3. Semantic-based approaches

Given the above-cited limitations, some researchers have recently introduced semantic-based approaches for improving spam detection (Caruana and Li, 2012). In Gansterer et al. (2008) and Renuka and Visalakshi (2014), 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. Other popular models such as the *latent Dirichlet allocation* (LDA) (Liu et al., 2016) and labeled-LDA (Song et al., 2017) have been used for building latent semantic spaces for spam detection. In Santos et al. (2012), the authors introduced the eTVSM method using a semantic ontology to extract the most dominant topics in a text message. These methods have generally yielded a better performance than the BoW model.

In Laorden et al. (2012), authors explored the use of word semantics by introducing a word sense disambiguation procedure to enhance spam detection. In Ezpelet et al. (2016), 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

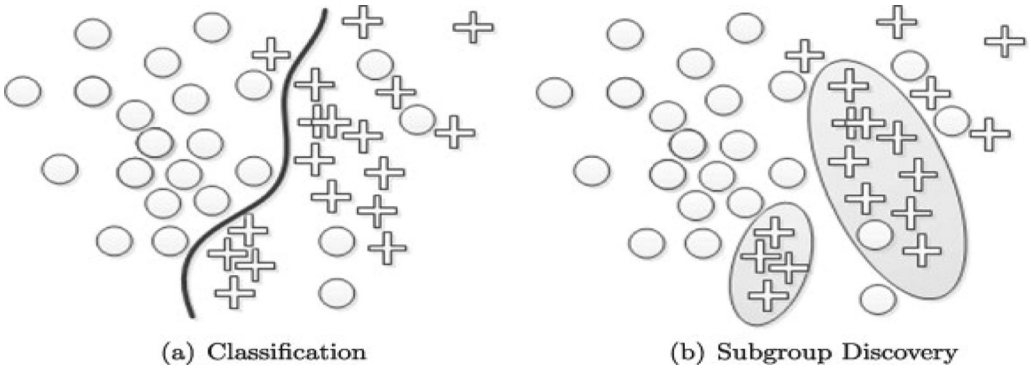

Fig. 3. Difference between classification model (a) and subgroup discovery model (b).

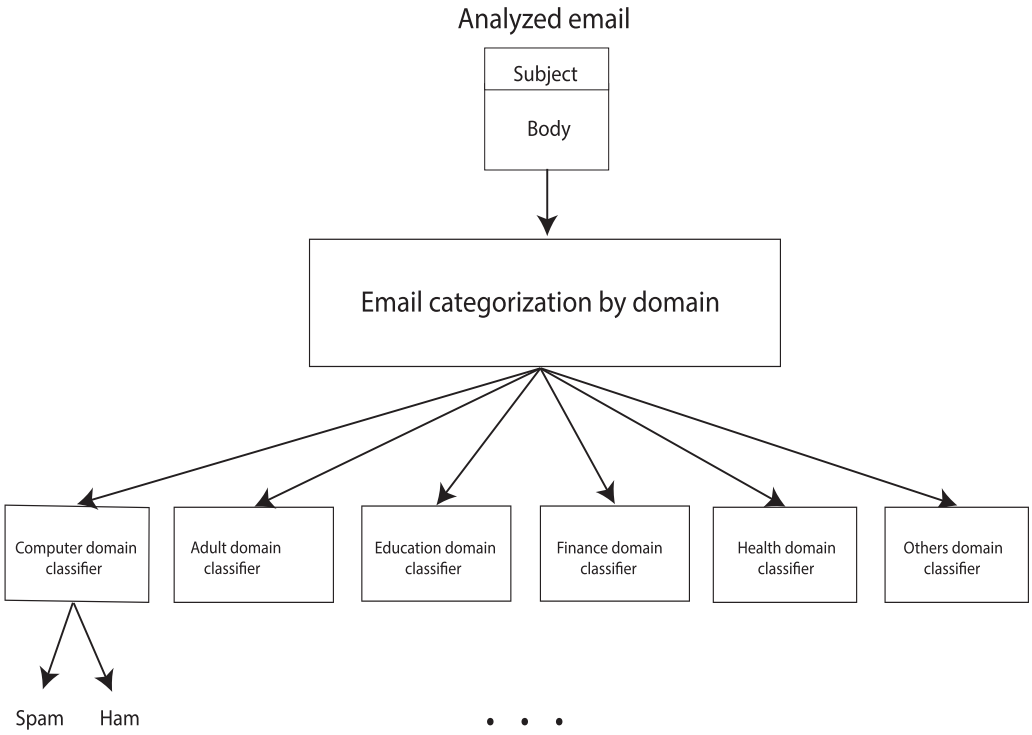

Fig. 4. Outline of the approach for spam detection using domain-specific classifiers.

the semantics of a spam message should be shaped with a positive meaning. Other studies Li et al. (2017) and Ren and Ji (2017) 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, authors in Wu et al. (2017) used the Doc2Vec model for training spam classifier on twitter messages. In the same vein, the authors in Kadam et al. (2018) 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 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.

Previous methods have shown the importance of using semantic information to improve spam detection. However, most of these methods use holistic and general-purpose semantic features which are agnostic of the subject of email content (email domain). 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 (Gansterer et al., 2008). 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

Table 1
Syntax for manually-specified rules.

```
Rule ::= Patterns '→' Class
Patterns ::= Pattern '∧' Patterns | Pattern
Pattern ::= Words | Number | Symbol | Space
Words ::= Word, Words | Word
Word ::= Letters+
Letters ::= A | ⋯ | Z | Number | Symbol | Space
A ::= 'A'| 'a'| '@'| ⋯
⋯
Z ::= 'Z'| 'z'| 'ż'| ⋯
Number ::= '0'|'1'|⋯|'9'
Symbol ::= '$'| '%'| '€'| ⋯
Space ::= '_' Space | ϵ
Class ::= 'Spam' | 'Ham'
```

Table 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

Table 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

(e.g., invitation to click on a specified link, obfuscation attempt, etc.). Obtained results demonstrate the improvement in term the accuracy of spam detection of the proposed approach compared to recent works.

3. The proposed approach

Our approach analyzes emails at two distinct semantic levels. In the first level, emails are categorized by their domains to enable a separate semantic view for spams in each domain. Based on a selected set of features, we categorize emails by specific domains (e.g., 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 (Lavrac et al., 2004). 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 spams. Fig. 1 gives a layout of steps composing our approach.

3.1. Email categorization by domain

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 (Gudkova et al., 2015; 2016) and Symantec (Symantec, 2016), nearly half of all spam is categorized as health, computers, 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 to these six domains. We define as well a sixth category named 'Others' to cover emails not belonging to the five studied categories.

In the Health domain, spams may contain ads for health services or miraculous medication products that offer amazing results. Typical spams include advertisements for weight loss, improved posture, nutritional supplements, etc. In the Finance domain, spams may contain offers for debt consolidation services, low interest loans, insurance, etc. In the Computer domain, typical spams involve special offers of bargain priced hardware or software and services for website hosting, domain registration, website optimization, etc. In the Adult domain, spams include offers for supplements designed to increase sexual ability, pornographic sites and other adult products. Finally, in the Education domain, spams may contain offers from fake online universities or training sites. Any other domain will be identified by the category Others.

Email categorization by domain is the process of automatically assigning a predefined category to an email reflecting its overall content. To implement this process, we consider three main procedures as shown in Fig. 2: 1) Preprocessing, 2) Feature selection and 3) Categorization.

3.1.1. Preprocessing

This procedure implements five main steps, as follows:

- *Keywords recognition*: is tasked with the automatic identification of keywords and abbreviations from predefined categories. For this, we used a list of regular expressions to recognize keywords and their variants used by spammers. For instance, in the adult category, we spot the taboo words, etc.
- *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. This step is used to avoid alphabet letters in the feature vectors at the end of the preprocessing process.
- *Segmentation*: is used to segment email content into a set of words. The email content is then encoded as a vector of words representing the email vocabulary.
- *Stop-word removal*: is the process of removing most common words in a text, such as articles, prepositions, pronouns, etc.
- *Stemming*: is used to perform the process of reducing variant word forms to a single "stem" form. For this, we used the tool PorterStemmer (The Porter Stemming Algorithm, 2006) 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" (Spellcheck, 2004). 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.

3.1.2. Feature selection

Feature selection is an important step and aims to reduce the number of features in order to improve the classifier performance. To select the most discriminative terms we used the *information gain* (IG) which is widely used in text categorization and spam detection (Bhowmick and Hazarika, 2016; Zong et al., 2015). It mea-

Table 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
	Mistimed	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 Bases	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
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

sures the discriminating potential of each word. Given a set T containing all terms obtained during the preprocessing phase from a pre-collected dataset of emails 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:

$$IG(t,c) = \sum_{\acute{c}\in\{c,\bar{c}\}} \sum_{\acute{t}\in\{t,\bar{t}\}} p(\acute{t},\acute{c}) log(\frac{p(\acute{t},\acute{c})}{p(\acute{t})p(\acute{c})}) \quad (1)$$

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).

For email categorization by domain, we chose the 500 terms having the highest IG for each category.

3.1.3. Categorization

After performing feature selection, various classification algorithms are tested to categorize email documents by domain, namely: Naive Bayes, *K-nearest neighbor* (KNN), Decision tree, Adaboost, SVM and Random forest. The choice of these algorithms is motivated by several factors. They can be adapted for both binary and multiclass classification problems, they are performing well with big training data and proved their performance in text categorization and spam detection problem. Furthermore, these algorithms use different statistical models which take advantage of different assumptions for input data (independence or non-independence between features) and constitutes a set of parametric and non-parametric classifiers.

3.2. 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 spams. 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

Table 6
Comparative evaluation with other methods using our collected dataset.

Model	Classifier	Accuracy	Recall	Precision	F1-measure
eTVSM (Santos et al., 2012)	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 (Androutsopoulos et al., 2000)	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 (Wu et al., 2017)	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 7
Comparative evaluation with other methods using CSDMC2010 SPAM dataset.

Model	Classifier	Accuracy	Recall	Precision	F1-measure
eTVSM (Santos et al., 2012)	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 (Androutsopoulos et al., 2000)	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 (Wu et al., 2017)	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 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

(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.

3.2.1. 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 (The Apache SpamAssassin Project, 2014) 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 in addition to general rules. 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.

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 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 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 defines 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 Santos et al. (2012). 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 WordNet (2013) 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. 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 (Wu et al., 2018). 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 It News Africa (2009), 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 spams:

- `Buy your meds online! Viiiagra, also: X@nax, valium,xenical,phentermine, somma, 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`.

3.2.2. Automatic extraction of semantic features

In order to allow automatic extraction of semantic features, we resort to CN2-SD algorithm (Lavrac et al., 2004) for automatic rule induction. The CN2-SD algorithm is an adaptation of CN2 classification rule learner (Clark and Boswell, 1991; Clark and Niblett, 1989) for *subgroup discovery* (SD) (Herrera et al., 2010). 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.

Subgroup discovery. 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 e-mail 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 Herrera et al. (2010) and Wrobel (1997), 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 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, 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. Fig. 3 shows the difference between subgroup discovery and classification tasks.

CN2-SD algorithm. 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. 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, 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 (Lavrac et al., 2004). The equation of this scheme is defined as follows:

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.

$$w(d_i,\ j) = \frac{1}{j+1} \tag{2}$$

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 *weighted relative accuracy* (WRAcc) as a quality measure, which is defined as follows:

$$\text{WRAcc}(r : cond \rightarrow y) = \frac{s(r \downarrow D)}{S}\left(\frac{s(r \downarrow * \ D)}{s(r \downarrow D)} - \frac{s(y)}{S}\right) \tag{3}$$

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 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 reweighted. 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. 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 no-presence of a term.

3.3. 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.

3.3.1. 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 *reduced error pruning* (REP) algorithm (Fürnkranz and Gamberger, 2012).

More formally, let V be a validation set which 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 non-meaningful features from r. 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.

3.3.2. 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 Hall et al. (1998) 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 $MinorityRule(r, r', D)$ as a function returning the rule associated with the minority class. Let B be a set of labeled examples. We define $MajorityClass(B)$ 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 :

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**
    $O \leftarrow O \setminus \{MinorityRule(r, r', D)\}$;
    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
          $D_1 \leftarrow (r \downarrow D) \cap (r' \downarrow D)$;
          $L_1 \leftarrow Cond(r) \setminus Cond(r')$;
          $L_2 \leftarrow Cond(r') \setminus Cond(r)$;
          $L_3 \leftarrow Cond(r) \cap Cond(r')$;
          $r \leftarrow (\widehat{Cond(r)} \wedge \neg \widehat{L}_2 \rightarrow Class(r))$;
          $r' \leftarrow (\widehat{Cond(r')} \wedge \neg \widehat{L}_1 \rightarrow Class(r'))$;
          $D_2 \leftarrow (r \downarrow D) \cap (r' \downarrow D)$;
          $r'' \leftarrow (\widehat{L}_1 \wedge \widehat{L}_2 \wedge \widehat{L}_3 \rightarrow MajorityClass(D_1 \setminus D_2))$;
          $O \leftarrow O \cup \{r''\}$;

      End

  End

  $b \leftarrow true$;
End

End

Algorithm 3: Conflict resolution.

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

3.4. 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 a supervised learning of candidate classifiers such as naive Bayes, decision trees and KNN. Whence 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 Fig. 4 for illustration).

4. Experimental results

To evaluate our approach, we have built a test dataset using messages collected from several public sources: Enron (Cohen, 2015), Ling-spam (Ling-Spam corpus, 2009) 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* (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.

Note that since Enron and Ling-spam databases do not cover all our spam domains, we resorted to collect hams from specialized forums. We labeled manually the selected emails according to their domain. 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 2). Each category includes both spam and ham emails. To extend the evaluation of our work, we tested it on 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 tested six classifiers, namely: Naive Bayes, KNN, Decision trees, Adaboost, SVM and Random forests to categorize emails by domain. To evaluate the effectiveness of the classifiers, we applied 10-fold cross validation. 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:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$F1 - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{7}$$

where *TP, TN, FP* and *FN* are the obtained true positives, true negatives, false positives and false negatives after classification. Table 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 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 *Precision* of 97.69 %.

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 domain semantic features to train our classifiers. Tables 4 and 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 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 several state-of-the-art methods, namely eTVSM (Santos et al., 2012), Doc2Vec (Wu et al., 2017) and BoW-SF (Androutsopoulos et al., 2000). Tables 6 and 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 (Androutsopoulos et al., 2000) does not carry high-level semantic information because words are taken independently. Since eTVSM (Santos et al., 2012) and Doc2Vec (Wu et al., 2017) are agnostic of email domain subject, it decreased their performance with

Table 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 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

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 (Wu et al., 2017) in term of accuracy is 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 8. We can see that by augmenting the training dataset, our approach 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.

4.1. 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 degree 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 $D = \{d_1, \cdots, d_k\}$ and a set of domain categories $C = \{c_1, \cdots, c_n\}$ where $n = 6$, 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):

$$P(spam \mid d) = \sum_{i=1}^{n} p(spam, c_i \mid d) \tag{8}$$

$$P(spam \mid d) = \sum_{i=1}^{n} p(spam \mid c_i) \cdot p(c_i \mid d) \tag{9}$$

where $p(c_i|d)$ 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 9 and 10, soft classification gives generally 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.

5. 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 provides a precise description of each domain spams 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-arts methods based on BoW and latent semantic analysis.

For future work, we intend to enhance our approach un several ways. First, we can enhance our semantic features by using word sense disambiguation. Second, to maintain the efficiency of our spam filtering in the long run, 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 feedbacks.

Declaration of Competing Interest

None.

References

Aggarwal, C.C., Zhai, C., 2012. Mining Text Data. Springer.

Androutsopoulos, I.I., Koutsias, J., Chandrinos, K.V., Spyropoulos, C.D., 2000. 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. ACM, pp. 160–167.

Bhowmick A., Hazarika S.M.. Machine learning for e-mail spam filtering: review, techniques and trends. 2016. arXiv:1606.01042, p. 1–27.

Bhuiyan, H., Ashiquzzaman, A., Juthi, T.I., Biswas, S., 2018. A survey of existing e–mail spam filtering methods considering machine learning techniques. Global J. Comput. Sci. Technol.(C) 18 (1), 20–29.

Bozkir, A.S., Sahin, E., Aydos, M., Sezer, E.A., Orhan, F., 2017. 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), pp. 1–15.

Caruana, G., Li, M., 2012. A survey of emerging approaches to spam filtering. ACM Comput. Surv. 44 (2), 1–27.

Clark, P., Boswell, R., 1991. Rule induction with CN2: Some recent improvements. In: European Working Session on Learning. Springer, pp. 151–163. Vol. 482
Clark, P., Niblett, T., 1989. The CN2 induction algorithm. Mach. Learn. 3 (4), 261–283.
Cohen W.W. Enron email dataset. 2015. https://www.cs.cmu.edu/~enron/.
Cormack, G.V., 2007. Email spam filtering: a systematic review. Found. Trends Inf. Retriev. 1 (4), 335–455.
Ling-Spam corpus. 2009. http://www.stat.purdue.edu/~mdw/598/datasets.html.
Dada, E.G., Bassi, J.S., Chiroma, H., Adetunmbi, A.O., Ajibuwa, O.E., 2019. Machine learning for email spam filtering: review, approaches and open research problems 5 (6), 1–23.
Ezpelet, E., Zurutuza, U., Hidalgo, J.M.G., 2016. Does sentiment analysis help in Bayesian spam filtering? In: International Conference on Hybrid Artificial Intelligence Systems, volume 9648. Springer International Publishing, pp. 79–90.
Fürnkranz, J., Gamberger, D., 2012. Foundations of rule learning. Springer Science and Business Media, Heidelberg New York Dordrecht London. P. 199–207
Gansterer, W.N., Janecek, A.G.K., Neumayer, R., 2008. Spam filtering basedon latent semantic indexing. In: Berry, M.W., Castellanos, M. (Eds.), Survey of Text Mining II - Clustering, Classification, and Retrieval, vol. 2, pp. 165–185.
Gudkova, D., Vergelis, M., Demidova, N., 2015. Spam and phishing in Q2 2015. Kaspersky Lab. P. 1–19.
Gudkova, D., Vergelis, M., Demidova, N., Shcherbakova, T., 2016. Spam and phishing in Q2 2016. Kaspersky Lab. P. 1–22.
Gudkova, D., Vergelis, M., Demidova, N., Shcherbakova, T., 2017. Spam and phishing in q2 2017. Securelsit, Spam and Phishing Reports. https://securelist.com/spam-and-phishing-in-q2-2017/81537/.
Hall, L.O., Chawla, N., Bowyer, K.W., 1998. Combining decision trees learned in parallel. In: KDD Workshop on Distributed Data Mining, pp. 10–15.
Herrera, F., Carmona, C.J., González, p, del Jesus, M.J., 2010. An overview on subgroup discovery: foundations and applications. Knowl. Inf. Syst. 29 (3), 495–525.
Herzberg, A., 2009. DNS-based email sender authentication mechanisms: acritical review. Comput. Secur. 28 (8), 731–742.
Hu, Y., Guo, C., Ngai, E.W.T., Liu, M., Chen, S., 2010. A scalable intelligent non-content-based spam-filtering framework. Expert Syst. Appl. 37 (12), 8557–8565.
It News Africa. 10 tricks used by spammers to get into your inbox. 2009. http://www.itnewsafrica.com/2009/11/10-tricks-used-by-spammers-to-get-into-your-inbox/.
Kadam, S., Gala, A., Gehlot, P., Kurup, A., Ghag, K., 2018. Word embedding based multinomial naive Bayes algorithm for spam filtering. In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). IEEE, pp. 1–5.
Laorden, C., Santos, I., Sanz, B., Alvarez, G., Bringas, P.G., 2012. word sense disambiguation for spam filtering. Electr. Comm. Res. Appl. 11 (3), 290–298.
Lavrac, N., Kavsek, B., Flach, P., Todorovski, L., 2004. Subgroup discovery with CN2-sd. J. Mach. Learn. Res. 5 (2), 153–188.
Li, L., Qin, B., Ren, W., Liu, T., 2017. Document representation and feature combination for deceptive spam review detection. Neurocomputing 254, 33–41.
Liu L., Lu Y., Luo Y., Zhang R., Itti L., Lu J.. Detecting “smart” spammers on social network: a topic model approach. 2016. arXiv:1604.08504.
Pérez-Díaz, N., Ruano-Ordás, D., Mendez, J.R., Galvez, J.F., Fdez-Riverola, F., 2012. Rough sets for spam filtering: Selecting appropriate decision rules for boundary e-mail classification. Appl. Soft Comput. 12 (11), 3671–3682.
Ren, Y., Ji, D., 2017. Neural networks for deceptive opinion spam detection: an empirical study. Inf. Sci. 385, 213–224.
Renuka, K.D., Visalakshi, P., 2014. Latent semantic indexing based SVM model for email spam classification 73 (6), 437–442.
Sahami, M., Dumais, S., Heckerman, D., Horvitz, E., 1998. A bayesian approach to filtering junk e-mail. In: AAAI Workshop on Learning for Text Categorization, pp. 98–105.
Saidani, N., Adi, K., Allili, M.S., 2017. A supervised approach for spam detection using text-based semantic representation. In: International Conference on E-Technologies. Springer, Cham, pp. 136–148. Vol 289.
Santos, I., Laorden, C., Sanz, B., Bringas, P.G., 2012. Enhanced topic-based vector space model for semantics aware spam filtering. Expert Syst. Appl. 39 (1), 437–444.
Schleimer, S., Wilkerson, D.S., Aiken, A., 2003. Winnowing: local algorithms for document fingerprinting. In: ACM SIGMOD International Conference on Management of Data, pp. 76–85.
Singh, V.K., Bhardwaj, S., 2018. Spam mail detection using classification techniques and global training set. In: Intelligent Computing and Information and Communication, 673, pp. 623–632.
Song, L., Lau, R.Y.K., Kwok, R.C.W., Mirkovski, K., Dou, W., 2017. Who are the spoilers in social media marketing? Incremental learning of latent semantics for social spam detection. Electr. Comm. Res. 17 (1), 51–81.
Spellcheck, 2004. https://www.mathworks.com/matlabcentral/fileexchange/5378-spellcheck.
Symantec, 2016. Internet Security Threat Report 21, 1–81.
The Apache SpamAssassin Project. 2014. http://spamassassin.apache.org/.
The Porter Stemming Algorithm. 2006. http://tartarus.org/martin/PorterStemmer/.
Torabi, Z.S., Nadimi-Shahraki, M.H., Nabiollahi, A., 2015. Efficient support vector machines for spam detection: a survey. Int. J. Comput. Sci. Inf. Secur. 13 (1), 11.
Vorakulpipat, C., Visoottiviseth, V., Siwamogsatham, S., 2012. Polite sender: a resource-saving spam email countermeasure based on sender responsibilities and recipient justifications. Comput. Secur. 31 (3), 286–298.
Wang, H., Zheng, G., He, Y., 2015. The improved Bayesian algorithm to spam filtering. In: International Conference on Computer Engineering and Networks, pp. 37–44.
WordNet. A lexical database for english. 2013. https://wordnet.princeton.edu/.
Wrobel, S., 1997. An algorithm for multi-relational discovery of subgroups. In: European Symposium on Principles of Data Mining and Knowledge Discovery, pp. 78–87.
Wu, T., Liu, S., Zhang, J., Xiang, Y., 2017. Twitter spam detection based on deep learning. In: Proceedings of the australasian computer science week multiconference. ACM, pp. 1–8.
Wu, T., Wen, S., Xiang, Y., Zhou, W., 2018. Twitter spam detection: Survey of new approaches and comparative study. Comput. Secur. 76, 265–284.
Zong, W., Wu, F., Chu, L.K., Sculli, D., 2015. A discriminative and semantic feature selection method for text categorization. Int. J. Prod. Econ. 165, 215–222.

Nadjate Saidani is a Ph.D. student at Université du Québec en Outaouais (Canada) since 2012. She received the master degree in Networking and Distributed Systems from the University of Abderrahmane Mira, Bejaia (Algeria) in2011. Her primary research is in computer security using data mining techniques.

Kamel Adi holds a master’s degree in Theoretical Computer Science from Pierre et Marie Curie (Paris VI) University and a Ph.D. in Computer Security from Laval University, Quebec, Canada. He is currently a full professor in the Department of Computer Science and Engineering at the University of Quebec in Outaouais, Canada. Kamel Adi is also the co-director of the Computer Security Research Laboratory at Université du Québec en Outaouais, Canada. His research activities focus on the development and application of formal methods for solving problems related to computer security and computer networks.

Mohand Said Allili received the M.Sc. and Ph.D. degrees in computer science from the University of Sherbrooke, Sherbrooke, QC, Canada, in2004 and2008, respectively. Since June2008, he has been an Assistant Professor of computer science with the Department of Computer Science and Engineering, Universit du Qubec en Outaouais, Canada. His main research interests include computer vision and graphics, image processing, pattern recognition, and machine learning. Dr. Allili was a recipient of the Best Ph.D. Thesis Award in engineering and natural sciences from the University of Sherbrooke for2008 and the Best Student Paper and Best Vision Paper awards for two of his papers at the Canadian Conference on Computer and Robot Vision2007 and2010, respectively.