Enhancing representation in the context of multiple-channel spam filtering

María Novo-Lourés a,b,c, David Ruano-Ordás a,b,c,d, Reyes Pavón a,b,c, Rosalía Laza a,b,c, Silvana Gómez-Meire a, José R. Méndez a,b,c,*

a Department of Computer Science, Universidade de Vigo, ESEI - Escola Superior de Enxeñaría Informática, Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004-Ourense, Spain
b CINBIO, Universidade de Vigo, Research Group SI4, Department of Computer Science, 32004-Ourense, Spain
c SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur). SERGAS-UVIGO. Hospital Álvaro Cunqueiro Bloque técnico, Estrada de Clara Campoamor, 341, 36312 Vigo, Pontevedra, Spain
d Computer Graphics and Data Engineering Group, Department of Electronics and Computer Science, University of Santiago de Compostela, Campus Vida, 15782 Santiago de Compostela, Spain

ARTICLE INFO

Keywords:
Spam filtering
Feature engineering
Machine learning
Text representation

ABSTRACT

This study addresses the usage of different features to complement synset-based and bag-of-words representations of texts in the context of using classical ML approaches for spam filtering (Ferrara, 2019). Despite the existence of a large number of complementary features, in order to improve the applicability of this study, we have selected only those that can be computed regardless of the communication channel used to distribute content. Feature evaluation has been performed using content distributed through different channels (social networks and email) and classifiers (Ada-boost, Flexible Bayes, Naïve Bayes, Random Forests, and SVMs). The results have revealed the usefulness of detecting some non-textual entities (such as URLs, Uniform Resource Locators) in the addressed distribution channels. Moreover, we also found that compression properties and/or information regarding the probability of correctly guessing the language of target texts could be successfully used to improve the classification in a wide range of situations. Finally, we have also detected features that are influenced by specific fashions and habits of users of certain Internet services (e.g. the existence of words written in capital letters) that are not useful for spam filtering.

Introduction

Spam filtering continues to be an important research problem due to the interest and need to protect Internet users against the massive delivery of junk content. Although many approaches have been developed to identify spam, one of the most successful measures for spam protection is based on text categorization methods using ML (Machine Learning) techniques which represent texts by using features extracted from them (Kowsari, Meimandi, Heidarysafa, Mendu, Barnes & Brown, 2019). This is one of the fundamental tasks in NLP (Natural Language Processing) with broad applications to most kinds of documents including content from SMS (Short Message Service), websites, or e-mails.

* Corresponding author.
E-mail address: moncho.mendez@uvigo.es (J.R. Méndez).

https://doi.org/10.1016/j.ipm.2021.102812
Received 13 May 2021; Received in revised form 7 October 2021; Accepted 7 November 2021
Available online 24 November 2021
0306-4573/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
An automatic text classification process (Kowsari, Meimandi, Heidarysafa, Mendu, Barnes & Brown, 2019) consists of a first stage of text cleaning and pre-processing to remove noise or words that are redundant or unnecessary and that can negatively affect the overall performance. Automatic text classification requires a representation of each text as a set of characteristics or features that are predictive of the classification problem, which is usually in the form of structured columns. For this purpose, we usually take advantage of domain knowledge to transform raw data into features that better represent the underlying problem, thus facilitating the ML process and improving the classifier performance. This process, usually known as feature engineering, comprises methods to convert a text into features to feed classification algorithms and has a great impact on the success and effectiveness of classification applications (Arif, Li, Iqbal & Liu, 2018). Feature engineering is still one of the most time-consuming and challenging steps in applied NLP and text mining today, and the focus of many research studies in computer science (Nargesian, Samulowitz, Khurana, Khalil & Turaga, 2017).

Possible ways of representing texts include (i) Bag of Words (BoW) or token-based approaches, (ii) n-gram schemes, (iii) word embeddings, (iv) topic-based models and (v) synset-based representations. BoW representations (i) are based on using words (usually known as tokens) as features to represent texts. Different measures can be used with this kind of representation in order to assign a value for these features including the frequency of the word (continuous value), and the presence or absence of the word (binary value). Derived from BoW, n-gram schemes (ii) are based on using n contiguous words as features (for instance the sentence “I am too tall” could be represented as 2-gram features in (“I am”, “am too”, “too tall”). Additionally, the use of multiple consecutive characters (not words) as features is also included in the group of n-gram representation schemes. One of the main limitations of these schemes is the difficulty in representing some semantically similar sentences which have been written using synonymous words (e.g. “special car discounts this week”, “automobile deals during next seven days”). This limitation suggests the introduction of new strategies for representation that take advantage of semantic information.

Word embedding (Hajek, Barushka & Munk, 2020) schemes (iii) are based on using contextual information of words in large document collections to create models able to predict the most probable word in a given context. In these approaches, texts are represented by using the outputs generated by these models. During last years, LSTM (Long Short-Term Memory, a type of RNN-Recurrent Neural Networks) and CNN (Convolutional Neural Networks) have become a popular architecture designed to better capture long-term dependencies for representation (Minae et al., 2021). Moreover, BERT-based (Bidirectional Encoder Representations from Transformers) deep learning approach has been recently introduced and is able to capture semantic and long-distance dependencies in sentences to improve the classification performance (AbdulNabi & Yaseen, 2021). Topic-based models (iv) are probabilistic schemes used to analyse large collections of words to detect which of them are usually included in the same documents (Du et al., 2020). The words that are used jointly are grouped into “topics” which make it possible to determine the similarity of specific documents with these generated topics. Although word embeddings and topic-based models are completely different, they use large collections of documents to automatically compile semantic knowledge.

From another perspective, synset-based representations (v) are newer schemes used (Vélez de Mendizabal, Basto-Fernandes, Ezpeleta, Méndez & Zurutuza, 2020) to take advantage of knowledge compiled manually into ontological dictionaries (such as Wordnet or Babelnet). These dictionaries contain sets of synonyms (i.e. synsets) and the semantic relations between them (e.g. hypernyms, hyponyms, meronyms, holonyms). In these approaches, texts are represented using synsets so that synonymous words (e.g. car and automobile) are included in the same feature. Additionally, as proposed in a recent study (Vélez de Mendizabal et al., 2020), using these representations, synset-features can be grouped by taking advantage of hypernym relationships. In addition to these representation schemes, a large number of descriptive attributes of texts has been introduced in the classification to improve its performance (length of the text, number of words, etc.).

Often, by applying text representation strategies, a large number of attributes are obtained. This situation leads to an increment in the computational resources required for the construction of ML models. To deal with these situations, we usually take advantage of dimensionality reduction schemes (Méndez, Cotos-Yanez & Ruano-Ordas, 2019). For this purpose, the utility of each feature can be evaluated by using well-known measures (Fisher, Wilcoxon, Information Gain), selecting the subset of the original characteristics that will provide the most information in the process of classifying the texts. In this line, keeping irrelevant features can decrease the performance of ML models.

Finally, the extracted features are used to train a classifier. Different ML algorithms have been used for text classification including Naïve Bayes classifier, decision trees, rule-based classifiers, maximum entropy, logistic regression, and support vector machines (Bahgat, Rady & Gad, 2016; Guzella & Caminhas, 2009). Semi-supervised and unsupervised approaches are used in cases when there is a lack of labelled data (Keyvanpour & Imani, 2013; Witten, Frank, Hall & Pal, 2017). Moreover, the use of deep learning strategies (neural networks) has gained popularity in recent years within the context of text classification (Kumar Sharma, Kishor Sharma & Singh, 2019; Minae et al., 2021; Wu, Liu & Wang, 2020). The core component of these approaches is a machine-learning embedding model that maps text into a low-dimensional continuous feature vector that outperforms other representations in classification performance.

Content-based spam classification is very popular and takes advantage of techniques brought from pure text classification to analyse the textual content of communications and decide whether they are legitimate or spam (AbdulNabi & Yaseen, 2021; Baco-couche, Ahmed, Sierra-Sosa & Elmaghraby, 2020; Das, Dash, Das & Panda, 2020; Jain, Goel, Agarwal, Singh & Bajaj, 2020). In addition, in this case, one of the most important aspects of the ML pipeline is feature engineering.

One of the most important problems when fighting spam is the large list of protocols used to disseminate junk textual contents (e-
mail, weblogs, short message sending, social media, etc.). Some feature engineering studies addressed the filtering of spam contents by using specific features of one of the protocols used to disseminate spam. However, from a logical perspective, spam refers to a set of subject matters that are irrelevant for a specific user. Keeping these ideas in mind, the classification process should only include textual information in order to provide a unified solution for filtering spam contents regardless of the channels used for their distribution.

The goal of this study is to find textual features that can be successfully used to complement text representation and improve the performance of classic ML approaches (AdaBoost, Flexible Bayes, Naïve Bayes, Random Forests and SVMs) for spam filtering tasks. For this purpose, we have selected some complementary features introduced in previous feature engineering studies in the field of spam filtering together with some new ones (probability of accurately guessing language, interjection ratio, etc.). All selected features, which are applicable regardless of the distribution channel, have been analysed to evaluate their utility for spam filtering in combination with two different representation methods (BoW and synset-based). The experiment included spam and ham (legitimate) content distributed through two different Internet services (social networks and e-mail) together with five classifiers (Naïve Bayes, Flexible Bayes, Adaboost, Random Forests and Support Vector Machines). In detail: (i) we use content-based features, ensuring they are applicable to identify spam distributed through analysed channels; (ii) we compare the performance of several types of features in conjunction with synset and BoW (token-based) representations; and finally, (iii) we discuss their influence on the identification of spam contents delivered via the studied protocols.

Our main contributions include: (i) the development and execution of an experimental protocol to study the impact of selected features, (ii) the identification of complementary features that can be successfully used to detect spam distributed through social networks and e-mail and (iii) the identification of irrelevant features and discussing their weaknesses.

The remainder of the paper is organized as follows: Section 2 describes the state of the art in feature engineering for textual contents, whilst Section 3 identifies and categorizes those features that can be used regardless of text distribution channels. From the features identified, a group of 10 were selected to analyse in detail their impact on spam filtering accuracy. The experimental setup and results are addressed in Section 4. Finally, Section 5 provides a discussion of experimentation results and outlines the future research direction.

Related work

The transformation of voluminous unstructured data from a set of documents into a reasonable set of effective and predictive variables requires an initial decision about which features seem most appropriate, implementing their extraction and studying their impact on classifier performance. In the case of spam, different types of features were extracted from texts to address their classification.

Spam is everywhere, particularly in SMS, e-mails, indexed web sites, instant messaging platforms, forums, weblogs, wikis and social networks (Ferrara, 2019). Usually, the features used for the classification process depend on the specific distribution channel (communication protocols). However, some protocol specific features are not available for all forms of spreading spam contents. Therefore, the identification of specific features that can be computed only from text could introduce new methods to provide a unified spam filtering solution for the wide range of spreading methods. Below is a review of the state of the art in feature engineering compatible with the most frequent communication channels such as e-mail, SMS, web spam and social networks.

Some studies have shown how to take advantage of features for improving the accuracy of e-mail spam filtering. In particular, a published study (Alqatawna, Faris, Jaradat, Al-Zewairi & Adwan, 2015) analysed the effect of adding ninety different features to the data used to improve the results of classical data mining classifiers on an unbalanced dataset. Most of these features are easy to compute, such as the number of digit characters or tabulations. Some content-based and sender account features (sender country, sender IP address, sender email, sender & recipient age, sender reputation) were successfully combined to increase the accuracy of e-mail filtering (Dada et al., 2019). Finally, a recent study (Gangavarapu, Jaidhar & Chanduksa, 2020) introduced the use of forty features (nine based on the body, eight on the subject line, four on the sender address, thirteen on the URL and six on the script).

Moreover, the distribution of spam using SMS presents many new difficulties for analytics algorithms, and feature engineering is more critical because SMS messages are short, and the text is riddle with abbreviations, slang words, and idioms. Although much research has been done to improve SMS spam filtering over the past decade (Zepeleta, Zurutuza, & Gómez Hidalgo, 2016; Zainal & Jali, 2016), nowadays people tend to communicate information more regularly and faster via social media than through SMS (Kolajo, Daramola, Adebisi & Seth, 2020) and messages are sent usually by companies, who provide bulk SMS sending services for commercial needs, e.g., verification codes, e-commercial notifications, express delivery notifications, etc.

Several works have also addressed the utilization of appropriate features for the detection and elimination of web spam (also known as black-hat SEO) (Oskue & Razavi, 2014). However, due to various characteristics of this kind of data (such as large scale, high dimension, sparsity, and changing patterns) their classification remains a challenge. The previous approach was improved by incorporating new high-level web page features related to the URL, the HTML (HyperText Markup Language) content and page ranking information in the famous Google search engine (Xiang, Hong, Rose & Cranor, 2011). Regarding the analysis of web contents to detect spam, some useful properties were introduced, including the use of specific sentences, the inclusion of encoding/decoding functions in HTML, HTML injection, etc. (Prieto, Álvarez, López-García & Cacheda, 2012).

Some new lexical-based features, such as unusual letter combinations, consonant clusters together with statistics on syllables, words, and sentences were introduced (Luckner, Gad & Sobkowiak, 2014) to improve the classification performance. Another study (Kumar, Gao, Welch & Mansoori, 2016) presented novel cloaking-based features for web spam classification, which helps classifier models to achieve high precision and recall evaluations, thereby reducing false-positive rate. Furthermore, another research (Alsaleh & Alarifi, 2016) analysed the advantages of using linguistic-based features and introduced a set of seven new features for filtering websites, such as the number of inline frames,
The background of Instagram spam filtering also comprises several studies. In particular, a new scheme that applies feature-based methods and supervised learning techniques was introduced for filtering spam in Instagram (W. Zhang & Sun, 2017). Authors consider not only statistics from user profiles and posts, but also information gathered from photos. Another study (Akbar Septiani & Wibisono, 2017) experimented with three sets of features: (i) hand-engineered, such as comment length, number of capital letters, and number of emojis; (ii) keywords, representing the presence or absence of advertising words or product-related words; and (iii) text, namely bag-of-words, TF-IDF, and fastText embeddings (each combined with latent semantic analysis). The use of different kinds of algorithms was introduced to detect fake and automated accounts of Instagram. Finally, new aspects of the dataset (i.e., independence, separability, complex relations) were analysed in another study (Akyon & Esat Kalfaoglu, 2019).

Filtering YouTube comments was targeted by another approach (Alsaleh, Alarifi, Al-Quayed & Al-Salman, 2015) that takes advantage of similarity of post/comments, the interval between post and comments, number of words in the comments, a number of sentences in the comments, comment length, phone information, e-mail information, URL link, black word list, stop word ratio and word duplication ratio. Other study (Perveen, M., Rasool & Akhtar, 2016) analysed the impact of some features including negative word count, negative word ratio, URL, positive word count and positive word ratio. Finally, a recent research (Samsudin et al., 2019) proposes the use of ML techniques to introduce a spam comment detection framework in which the authors used three types of features: presence of links, length of comments and spam keywords.

Weblog spam is targeted in a study released in 2019 (Li, Wu & Wang, 2019) that evaluates the use of a Word2vec representation (content features) combined with inherent features extracted from the comments (attribute features). Authors define three attribute features: (i) number of comments posted by the same user in the weblog, (ii) the time elapsed between posting two consecutive comments and (iii) the number of responses to the current comment.

There is an incremental use of social networks to share opinions, experiences or feelings not only about products and services, but also about several issues in society. Since anyone can easily produce unrestricted reviews, some users take advantage of this situation to promote their products, brands and shops, or to denigrate their competitors. This practice, where reviews are manipulated or poisoned for profit or gain, is known as Opinion or Review Spam. Researchers have proposed methods and algorithms in the field of review spam detection (Tian, Mirzabagheri, Tirandazi & Bamakan, 2020; X. Zhang & Ghorbani, 2020). However, the detection of review spam cannot be done by content analysis or text classification alone, and can often be done more effectively by analysing spammers (e.g. whether they have actually experienced the product they have reviewed), their relationships (i.e. much spam is produced by collaboration between spammers) and the timing time of appearance of the reviews (Hussain, Mirza & Hussain, 2019; Wijnhoven & Pieper, 2019). For this reason, this type of spam is outside the scope of this review.

Most of the previous studies are specifically bound to a particular type of protocol or channel for the distribution of spam contents. In fact, some features introduced may not be available for all spam distribution channels and therefore, their utility is limited for detecting of spam contents. To the best of our knowledge, only a few studies addressed the feature-based detection of spam regardless of the distribution channel used. In particular, some researchers (Cormack, Gómez Hidalgo & Puertas Sánz, 2007) considered the problem of content-based spam filtering for short text messages that emerges through three different channels: (i) mobile (SMS) communication, (ii) blog comments, and (iii) e-mail summary information. In 2011, Monarch (Thomas, Grier, Ma, Paxson & Song, 2011) was introduced. It is a real-time system that crawls URLs, as they are submitted to all Internet services (e-mail, Twitter, etc.), and determines whether the URLs direct to spam content by examining a wide range of features extracted from HTML contents, IP address analysis, HTTP (HyperText Transfer Protocol) issues and DNS (Domain Name System) information. Additionally, another study (Adewole, Anuar, Kansin & Sangaiyah, 2019) introduced a unified framework for detecting both spam message and spam accounts from SMS and twitter microblogging platforms. Moreover, several feature sets compatible with e-mail and SMS distribution channels were evaluated (El-Alfy & AlHasan, 2016).
In contrast to the existing approaches, our study presents a novel set of generic features for the detection of spam content regardless of the distribution channel, or Internet communication protocol used for the distribution of junk contents. Finding a scheme able to perform better by using only the features extracted from text would be a reliable solution that would unify the wide range of software applications currently used to filter contents exchanged through specific protocols.

**Channel-independent features**

The successful operation of ML techniques depends greatly on the selection of an appropriate feature set for the target problem. Although its main distribution channel is e-mail, spam is distributed through a multitude of additional channels, which have their own peculiarities. To cope with this issue, our proposal uses only features that are independent of the distribution channel (particularly, text-based ones).

In this study, we analyse the impact of text-features on the classification that are included in the following groups: (i) Stylistic text features, (ii) non-text entities, (iii) features based on NER (Named Entity Recognition), (iv) sentiment analysis, (v) language identification easiness and (vi) compression-based features.

Stylistic text features (i) comprise features that are connected with the author of the text. Some examples are the excessive use of capital letters, or the average number of words per sentence. Although these features can be easily computed, this work only took into account the use of interjections, and capital letters.

Non-text entities (ii) are unnatural language components included in the contents, in particular hashtags, URLs or mentions (@userName). In our work, we paid special attention to the number of URLs included in the text. For this purpose, e-mail addresses were also considered as URLs in the form mailto:user@domain.com.

We believe some NER properties (iii) could be successfully used for the identification of spam contents. Some frameworks currently implement the functionality of extracting different kinds of entities (date, currency, numbers, or locations). As these properties could contribute valuable information to identify specific subject matters, our study included the use of these properties. In particular, we take advantage of the Stanford NLP Framework\(^3\) to find entities in the analysed contents.

Sentiment analysis (iv) features have been successfully used in previous studies to improve the detection of spam in multiple datasets. As these features can be easily computed by using only text data, they were also considered in this study.

In our experiments, we used a language guesser\(^4\) and computed the reliability of the classification (v). The reliability of this kind classification is poor when the length of the available text is very short or when there are spelling errors that seriously impede the correct identification of the language. The presence of spelling errors could provide valuable information to increase the accuracy of classifiers.

Finally, compression-based features (vi) could identify some spammer tricks such as mixing up legitimate texts or words (sometimes invisible) to avoid the detection of their spam contents. We included for evaluation the compression ratio of contents to assess these parameters.

Ratio-based features were used to conduct the experimentation. When measuring some elements in the text, including URLs, interjections, and NER elements, the results were divided by the number of words included in the text. New features were combined with token and synset-based frequency features to determine their impact in the classification and estimate their value. The design and results of the experiments is shown in the next section.

**Experiment design and results**

To demonstrate the utility of the features shown above, we designed and executed a specific empirical experiment. The following subsections show, respectively, the criterions used for selecting the datasets, the design of the experimental protocol, measures used for the evaluation, and the results.

4.1. **Dataset election**

The election of datasets for experimentation was made by carefully examining the features of publicly available datasets. A recent study (Vázquez et al., 2021) introduced a significant number of datasets comprising spam communications distributed over different Internet channels. All of them are available for testing the performance of spam filtering schemes. The aim of this study is to find features to represent spam texts transmitted in different ways. To improve the statistical significance of the results with moderate computational expense, medium-sized datasets should be used. Keeping in mind these considerations, we have selected YouTube Spam Collection and SpamAssasin dataset.

YouTube Spam Collection (Almeida, Lochter & Túlio, 2015) is a compilation of YouTube comments labelled as ham/spam. It comprises comments made by users on five highly accessed videos. The comments include the full text, comment identifier, author identifier, date, and comment tag (spam/ham). To ensure privacy of users, we kept only the comment id and the label. Comments were re-downloaded using the YouTube API to ensure that those that were removed by owners are not present in the dataset used in the experiment. The texts included in the dataset (a total amount of 1956) are short and the percentage of spam messages is 51%. This

\(^3\) Available at https://nlp.stanford.edu/software/

\(^4\) Available at https://github.com/optimaize/language-detector

5
dataset was used in some previous similar studies (Baccouche et al., 2020; Das et al., 2020) and is therefore also suitable to the present work.

SpamAssassin dataset consists of medium-sized, well-written texts (more than 6000) with a 31% of spam ratio. As the text is usually free of spelling errors, this dataset has been used to represent e-mail messages (spam e-mail) and websites (WebSpam). This corpus has been previously used for research in recent works (Ruano-Ordoñez, Fdez-Riverola, & Méndez, 2018).

4.2. Experimental design and configuration

The experimental protocol involves the use of two datasets (YouTube Collection and SpamAssassin corpora) which were represented by using two schemes (token and synset based). We evaluated the performance achieved by five classifiers (Naïve Bayes, Flexible Bayes, Random Forests, Adaboost, Support Vector Machines) using these representations together with each of the analysed features. The selected classification techniques have been widely used in the context of spam filtering and cover different types (two probabilistic-based approaches, two classifier ensembling methods and a geometric model). Fig. 1 graphically presents the experimental protocol designed for this purpose.

As shown in Fig. 1, the text data from original instances (YouTube comments and e-mail messages in RFC 5322 format (Leiba, 2013)) has been extracted and cleaned for the representation. Raw text data achieved from this step has been converted into high-dimensional feature vectors using token-based and synset-based approaches. The value of each token/synset feature was computed by dividing the number of occurrences of each token/synset by the total number of tokens/synsets identified in the message. In order to achieve synsets from text we took advantage of the Babelfy API (Application Programmers Interface) (Moro, Cecconi & Navigli, 2014a; Moro et al., 2014b). All pre-processing steps were executed using NLPA software (Novo-Lourés, Pavón, Laza, Ruano-Ordoñez & Méndez, 2020). The dimensionality of feature vectors was reduced to 1000 features using Information Gain (IG) scheme (Pérez-Díaz, Ruano-Ordoñez, Fdez-Riverola & Méndez, 2016). We used the well-known Java implementations of NaïveBayes (switching the use of kernels to on/off), RandomForests, AdaboostM1 and SMO (Sequential Minimal Optimization) algorithms provided in the Weka6 package as classifiers. These classifiers were trained using a stratified subset of the dataset containing the 80% of its instances. The remaining 20% of instances (which are also stratified) were used for evaluation purposes. The evaluation of the different configurations was made using Kappa statistic, accuracy, recall/precision, f-score and TCR (Total Cost Ratio) for \( \lambda = 1, 9 \) and 999 (Dada et al., 2019).

To study the impact on performance for each property, they were added in a separate form. In this study we selected ten properties for analysis: ALLCAPSRatio, InterjectionsRatio, URLSRatio, NERDateRatio, NERMoneyRatio, NERNumberRatio, NERLocationRatio, Polarity, CompressionRatio and LanguageReliability.

ALLCAPSRatio and InterjectionsRatio are stylistic text features. The former helps to detect an excessive use of capitalized words by finding the proportion of words, which have been written using capital letters. Although this feature became quite popular in the domain, the benefits achieved from its use could be quite limited given that this form of writing was only a trend in some specific kinds of spam distributions forms (e-mail) during some years (before the inclusion of rules filtering these kinds of messages in some popular spam filtering frameworks such as SpamAssassin7). The latter feature calculates the proportion of interjections to the total number of words in the text.

URLSRatio is a non-text entity property, which is computed as the number of URLS divided by the number of content words.

We also take advantage of NER by computing the properties NERDateRatio, NERMoneyRatio, NERNumberRatio and NERLocationRatio as proportions of the number of entities detected in relation to the total number of words. Thus, NERDateRatio considers only entities related to dates as dates of the week, dates in several formats, or time of the day; NERMoneyRatio detects references to money amounts; NERNumberRatio takes numbers into account, excluding those belonging to dates, time, money or another entity; and NERLocationRatio detects locations contained in the text such as places, cities, countries or continents.

Polarity is a sentiment analysis feature that is computed by using lexicons, an inventory of lexemes and their individual evaluation. The polarity score is calculated within the range [−1.0, 1.0] and considers the opinion expressed by the texts. Text polarity can be defined as negative (those with pessimistic contents that will achieve lower scores) or positive (which will achieve high polarity scores).

We also used LanguageReliability property, which is defined as the confidence of the identified language. This property helps to identify misspellings while writing the contents.

Finally, CompressionRatio is a compression-based feature computed by dividing the original text size by its compressed size.

The next section compiles the experimental results achieved following this protocol.

4.4. Experimental results

Having described the evaluation protocol, this section now shows the results and direct conclusions achieved from its application. In order to carry out a deep analysis of achieved results, a wide range of performance measurements of the analysed configurations was included as additional materials. Table shows the f-score evaluation of all studied scenarios (using different representation methods,

---

5 https://spamassassin.apache.org/old/publiccorpus/
6 Available at http://www.cs.waikato.ac.nz/∼ml/weka/
7 See information about ALLCAPS and SUBJ_ALLCAPS SpamAssassin rules.
features, classifiers, and datasets). For each configuration (Dataset, Representation method and Classifier), those features that help the classifier to perform better are represented in bold and the best configuration are highlighted in red.

Using rows as criteria (datasets, representations, and classifiers), a comparison indicates the good functioning of Random Forest’s strategy to filter spam and the existence of very small performance differences when varying the representation (synset/tokens) of medium-large datasets (SpamAssassin). However, taking into account the main target of this study (features), URLsRatio, LanguageReliability, CompressionRatio and ALLCAPSRatio (the latter one only in SpamAssassin) could all be successfully used to increase performance. As the presence of URLs in spam e-mails is essential to promote the purchase of goods online, any communication without an URL would probably be legitimate. The URLsRatio attribute is therefore of great relevance for the classification of the texts. Moreover, the utility of LanguageReliability and CompressionRatio features can be explained by the usage of some tricks to avoid spam filters (e.g. e-leet-speak (Ferrante, 2008)) and language style of spam contents (abbreviations, misspellings and typos, etc. (Aycock & Friess, 2006). Finally, the use of capitalized words is very common in the context of spam e-mail and therefore, the performance of the filters can be improved by using features that collect this information. Inspired in Sankey diagrams (Lupton & Allwood, 2017), we graphically plotted the impact of using features on classification performance (f-score) together with representations (see Fig. 2) and ML models (refer to Fig. 3).

In Fig. 2, each configuration is represented as a flow connecting a type of representation, a feature, and its impact on performance with respect to the baseline configuration (the performance for the same configuration without including additional features). The colours red and green correspond to a deterioration and improvement of performance, respectively, while the grey colour means that the accuracy does not vary. The width of each flow models the number of classifiers (min=1, max=5) that fit the represented situation. A right-to-left (RTL) analysis of the Figure is recommended to observe which configurations improve, worsen, or have the same f-score as the baseline configuration.

As shown in Fig. 2, the use of additional features improves or keeps the performance in about 85 percent of the analysed configurations. Furthermore, a more random behaviour of the classifiers is observed when using synset-based representations with small texts. This is probably due to the difficulty in disambiguating the meaning of words (WSD) in short texts. From all analysed features, we found URLsRatio, LanguageReliability, CompressionRatio and ALLCAPSRatio as the most promising to improve classification results. Additionally, the use of the Polarity feature only impaired the accuracy of the classifiers in one of the 20 configurations analysed.

In order to determine the impact of the analysed features in the operation of classifiers, Fig. 3 graphically represents flows connecting attributes (named as Models, Features, and Performance). The width of each flow (min=1, max=2) is associated to the number of representations (synsets or tokens) that fit the connected attributes. This figure has been designed to be interpreted using a LTR (left-to-right) direction. To this end, the colour palette used to fill the flows highlights the different classifiers used (one colour per ML model).

As shown in Fig. 3, additional features allow us to improve up to 50 percent of configurations (47% and 49% respectively on YouTube and SpamAssassin datasets) while only 16 percent of analysed configurations performed poorly. LanguageReliability, CompressionRatio and ALLCAPSRatio can clearly improve the performance for almost all classifiers. There are a few exceptions to this situation, such as the use of Random Forest with datasets where the texts are very short, which is probably the cause of the difficulty in carrying out the disambiguation process. Additionally, the use of URLs Ratio can improve a great number of configurations. However, by using this feature, a larger number of configurations achieve poor results. Keeping in mind the best configurations identified above, we ran a cost-sensitive performance evaluation of the studied analysis using TCR scores for $\lambda=1$, 9 and 999. The results of this evaluation are included in Table . For each configuration (Dataset, Representation method and Classifier), those features that help the classifier to perform better are represented in bold and the best configuration is highlighted in red.
Fig. 2. Analysis of the impact of the considered features across representations.
Fig. 3. Analysis of the impact of the considered features across classifiers.
### Table 1
Evaluation of analysed configurations using f-score statistic.

| Dataset      | Repres. | Classifier | Features | none | allcaps | Ratio | Compr. Ratio | Interjec. Ratio | Lang. Reliab. | NER Date Ratio | polarity | Locat. Ratio | URLs Ratio | Money Ratio | Number Ratio |
|--------------|---------|------------|----------|------|---------|-------|--------------|----------------|--------------|----------------|----------|-------------|-----------|-------------|--------------|
| Youtube      | tokens  | Adaboost   | 0.5652   | 0.5652 | 0.5333  | 0.5652 | 0.5652       | 0.5652         | 0.5652      | 0.5652         | 0.5652   | 0.5652      | 0.5652    | 0.5652      | 0.7429       |
|              |         | Flexible Bayes | 0.7874 | 0.7874 | 0.7874 | 0.7874 | 0.7874   | 0.7874         | 0.7874 | 0.7874 | 0.7874 | 0.7874 | 0.7874 | 0.7874 | 0.7969       |
|              |         | Naïve Bayes | 0.6667  | 0.6705 | 0.6784 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6705 | 0.6867       |
|              |         | Random Forest | 0.7965 | 0.8276 | 0.8333 | 0.8276 | 0.8276 | 0.8276 | 0.8276 | 0.8276 | 0.8276 | 0.8276 | 0.8205 | 0.8276 | 0.8908       |
|              |         | SMO (SVM) | 0.7000 | 0.7059 | 0.7731 | 0.7059 | 0.7059 | 0.7059 | 0.7059 | 0.7059 | 0.7059 | 0.7059 | 0.6555 | 0.7059 | 0.6555       |
| Synsets      | tokens  | Adaboost | 0.4628 | 0.4286 | 0.4651 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4286 | 0.4096       |
|              |         | Flexible Bayes | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6555 | 0.6667 | 0.6555 | 0.6777       |
|              |         | Naïve Bayes | 0.6296 | 0.6545 | 0.6726 | 0.6545 | 0.6545 | 0.6545 | 0.6545 | 0.6545 | 0.6545 | 0.6545 | 0.6667 | 0.6667 | 0.6296       |
|              |         | Random Forest | 0.6667 | 0.6613 | 0.7310 | 0.6613 | 0.6613 | 0.6613 | 0.6613 | 0.6613 | 0.6613 | 0.6613 | 0.6721 | 0.7000 | 0.7395       |
|              |         | SMO (SVM) | 0.5833 | 0.5833 | 0.6949 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833 | 0.5833       |
| SpamAssassin | tokens  | Adaboost | 0.7343 | 0.7465 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343 | 0.7343       |
|              |         | Flexible Bayes | 0.8937 | 0.8968 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937 | 0.8937       |
|              |         | Naïve Bayes | 0.8927 | 0.9045 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938 | 0.8938       |
|              |         | Random Forest | 0.9488 | 0.9539 | 0.9543 | 0.9427 | 0.9496 | 0.9496 | 0.9510 | 0.9513 | 0.9475 | 0.9427 | 0.9481       |
|              |         | SMO (SVM) | 0.9418 | 0.9447 | 0.9418 | 0.9420 | 0.9461 | 0.9418 | 0.9404 | 0.9435 | 0.9418 | 0.9420 | 0.9432       |
| Synsets      | tokens  | Adaboost | 0.7043 | 0.7059 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043 | 0.7043       |
|              |         | Flexible Bayes | 0.8867 | 0.8914 | 0.8867 | 0.8898 | 0.8883 | 0.8838 | 0.8851 | 0.8851 | 0.8851 | 0.8851 | 0.8851 | 0.8851 | 0.8835       |
|              |         | Naïve Bayes | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871 | 0.8871       |
|              |         | Random Forest | 0.9401 | 0.9409 | 0.9517 | 0.9339 | 0.9435 | 0.9353 | 0.9457 | 0.9374 | 0.9415 | 0.9401 | 0.9437       |
|              |         | SMO (SVM) | 0.9275 | 0.9306 | 0.9248 | 0.9302 | 0.9290 | 0.9277 | 0.9220 | 0.9275 | 0.9277 | 0.9275 | 0.9275 | 0.9292 | 0.9292       |
| Dataset       | Repres. | Classifier | TCR | λ = 1 | 9 | 999 | 1 | 9 | 999 | 1 | 9 | 999 | 1 | 9 | 999 | 1 | 9 | 999 | 1 | 9 | 999 |
|--------------|---------|------------|-----|-------|---|----|---|---|----|---|---|----|---|---|----|---|---|----|---|---|----|
| Youtube tokens | Adaboost | 1.65 | 1.65 | 1.65 | 1.65 | 1.65 | 1.571 | 1.571 | 1.571 | 1.65 | 1.65 | 1.65 | 2.444 | 2.444 | 2.444 |
|              | Flexible Bayes | 2.444 | 0.574 | 0.006 | 2.444 | 0.574 | 0.006 | 2.444 | 0.574 | 0.006 | 2.444 | 0.574 | 0.006 | 2.538 | 0.579 | 0.006 |
|              | Naive Bayes | 1.158 | 0.150 | 0.001 | 1.158 | 0.147 | 0.001 | 1.2 | 0.153 | 0.001 | 1.158 | 0.147 | 0.001 | 1.270 | 0.167 | 0.002 |
|              | Random Forest | 2.870 | 1.692 | 0.033 | 3.3 | 1.833 | 0.033 | 3 | 0.6 | 0.006 | 3.3 | 1.833 | 0.033 | 5.077 | 5.077 | 5.077 |
|              | SVM | 1.833 | 0.5 | 0.005 | 1.886 | 0.537 | 0.006 | 2.444 | 0.795 | 0.009 | 1.886 | 0.537 | 0.006 | 1.610 | 0.431 | 0.005 |
| synsets      | Adaboost | 1.375 | 1.375 | 1.375 | 1.375 | 1.375 | 1.435 | 1.435 | 1.435 | 1.435 | 1.375 | 1.375 | 1.375 | 1.347 | 1.347 | 1.347 |
|              | Flexible Bayes | 1.610 | 0.431 | 0.005 | 1.610 | 0.431 | 0.005 | 1.610 | 0.431 | 0.005 | 1.610 | 0.431 | 0.005 | 1.692 | 0.437 | 0.004 |
|              | Naive Bayes | 1.65 | 0.635 | 0.008 | 1.737 | 0.647 | 0.008 | 1.784 | 0.606 | 0.007 | 1.737 | 0.647 | 0.008 | 1.941 | 0.673 | 0.008 |
|              | Random Forest | 1.610 | 0.391 | 0.004 | 1.571 | 0.371 | 0.004 | 1.692 | 0.267 | 0.003 | 1.571 | 0.371 | 0.004 | 2.129 | 0.641 | 0.007 |
|              | SVM | 1.32 | 0.327 | 0.003 | 1.32 | 0.327 | 0.003 | 1.833 | 0.532 | 0.006 | 1.32 | 0.327 | 0.003 | 1.32 | 0.327 | 0.003 |
| SpamAssassin tokens | Adaboost | 2.173 | 0.816 | 0.010 | 2.279 | 0.893 | 0.012 | 2.173 | 0.816 | 0.010 | 2.173 | 0.816 | 0.010 | 2.173 | 0.816 | 0.010 |
|              | Flexible Bayes | 5.081 | 2.575 | 0.042 | 5.222 | 2.611 | 0.042 | 5.081 | 2.575 | 0.042 | 5.151 | 2.593 | 0.042 | 5.081 | 2.575 | 0.042 |
|              | Naive Bayes | 4.585 | 0.821 | 0.008 | 5.083 | 0.817 | 0.008 | 4.642 | 0.837 | 0.008 | 4.642 | 0.837 | 0.008 | 4.642 | 0.837 | 0.008 |
|              | Random Forest | 6.641 | 1.628 | 0.016 | 10.743 | 1.852 | 0.018 | 11.059 | 2.725 | 0.029 | 11.75 | 2.136 | 0.021 | 9.641 | 1.969 | 0.020 |
|              | SVM | 8.744 | 2.307 | 0.025 | 9.171 | 2.335 | 0.025 | 8.744 | 2.307 | 0.025 | 9.4 | 2.35 | 0.025 | 8.952 | 2.321 | 0.025 |
| synsets      | Adaboost | 1.938 | 0.642 | 0.008 | 1.928 | 0.607 | 0.007 | 1.938 | 0.642 | 0.008 | 1.938 | 0.642 | 0.008 | 1.938 | 0.642 | 0.008 |
|              | Flexible Bayes | 4.759 | 2.149 | 0.031 | 4.947 | 2.186 | 0.031 | 4.759 | 2.149 | 0.031 | 4.821 | 2.161 | 0.031 | 4.642 | 2.124 | 0.031 |
|              | Naive Bayes | 4.585 | 1.229 | 0.013 | 4.585 | 1.229 | 0.013 | 4.585 | 1.229 | 0.013 | 4.585 | 1.229 | 0.013 | 4.585 | 1.229 | 0.013 |
|              | Random Forest | 8.356 | 1.701 | 0.017 | 8.545 | 2.000 | 0.021 | 10.444 | 2.410 | 0.025 | 8.952 | 2.112 | 0.022 | 8.952 | 2.022 | 0.021 |
|              | SVM | 7.094 | 2.077 | 0.023 | 7.373 | 2.011 | 0.022 | 6.836 | 1.969 | 0.022 | 7.231 | 2.089 | 0.023 | 7.231 | 2.000 | 0.022 |
As shown in Table 1 and 2, the best feature for datasets with small messages is the presence of URLs. We believe this information is also suitable for filtering a wide variety of contents such as the comments in social networks (such as Instagram) or SMS messages. The use of URLs in larger texts (SpamAssassin) seems to have a limited impact on performance because their use in legitimate contents is more frequent. Particularly, in the case of SpamAssassin dataset, the existence of words written completely in capital letters is appropriate for spam identification. In this case, we believe that this finding is very dependant on the specific dataset and cannot be extrapolated to other contents. In our opinion, the use of LanguageReliability and/or URLsRatio properties could achieve better results for large contents.

Finally, to assess the significance of improvements from a statistical point of view, a statistical modelling based on the multi-way ANOVA was carried out for each of the best-identified features (URLsRatio, LanguageReliability and CompressionRatio). For this purpose, we compute two models to explain kappa and f-score (dependant variables) in relation to (i) dataset, (ii) representation (synsets/tokens), (iii) ML model and (iv) the presence of feature (categorical independent variables). The data used in the statistical test were obtained by performing a 10-fold cross-validation, with all the considered algorithms, representations, and datasets. Instead of considering one single summary f-score/kappa value for each cross-validation, each of the 10 partial results obtained from the cross-validation were included in the sample. The size sample for each dependant variable was 400 (2 values for presence of feature variable x 2 representations x 2 datasets x 5 models x 10 folds). Using model for kappa, we found statistically significant differences using URLsRatio (p-value < 0.0001 with an estimated average improvement of 0.0356) and CompressionRatio (p-value < 0.1 with an estimated average improvement of 0.0145). Moreover, using model for f-score we found that URLsRatio presents statistically significant differences (p-value < 0.0001 with and estimated average improvement of 0.0368).

The next section introduces some conclusions of the experiments made and outlines future work for our research.

4.5. Theoretical and practical implications

A large number of previous works have addressed feature-engineering tasks for the spam-filtering domain. Some of them introduce properties that can be successfully used to improve the identification of spam messages in specific domains (e.g. link-based features that are only applicable to web spam). However, we believe that content and intentionality of spam messages makes them different from other messages. Moreover, we also believe that issues related to the protocol or service that was used to deliver the message to the target user could help to classify the message, but should not be the main data used for its classification.

In this study, we identified that some features used to successfully improve the performance of classical ML approaches in the past are no longer valid because they model fashion or user habits that are now extinct. Thus, writing messages with many capital letters (whole words) is not a suitable feature for detecting spam in the analysed distribution forms. Furthermore, we also identified some features that can be used to improve filtering performance but only in a few forms of message exchange (e.g. social networks). Moreover, we identified some features that slightly increased the classification performance for all forms of spam distribution analysed, such as URLsRatio, CompressionRatio and LanguageReliability. Links are necessary in commercial spam messages to provide the receiving user with a mechanism to purchase the advertised goods. The CompressionRatio feature helps to identify misspellings and/or the use of linguistic tricks to avoid detection of spam messages in spam filters. On the other hand, the LanguageReliability feature (in which the improvement achieved is not significant from a statistical point of view) is also indicative of misspellings, mistyped texts, and too short messages.

The current proposal takes advantage of a wide variety of findings introduced in previous research studies. However, some recent studies have detailed proposals that may be complementary to the present one. Some of them have even used the same datasets to evaluate their suitability. Particularly, the study performed by Baccouche et al. (Baccouche et al., 2020) discusses the usage of recurrent neural networks (particularly Long Short-Term Memory architectures) for spam detection. Since the representation used by these neural networks is in the form of matrices, the inclusion of features suggested in this study is not trivial. However, Das et al. (Das et al., 2020) use some classifiers based on the use of simpler artificial neural networks. They conclude that these classifiers generally outperform classical approaches. Therefore, combining the findings of both studies could improve the efficiency of filtering processes.

Discussion and future work

This study analyses the impact on the spam filtering performance of ten distribution channel independent features. The impact of the features was analysed using two different datasets, two different text representation schemes, and five classification techniques. To evaluate performance of classifiers we took advantage of five standard and popular measures including Accuracy, Kappa Statistic, Recall/Precision, f-score and TCR. Although the findings seem to be applicable for different distribution channels, they have been identified and tested only in spam distributed via social networks and e-mail.

Although it is not the aim of this work, we have noted the low level of accuracy of classifiers when using synset-based representations with small texts (YouTube Comments Dataset). These difficulties probably derive from the process of text disambiguation. In fact, results achieved with synset-based representations in longer texts (SpamAssassin) are more similar to those obtained with token-based representations.

Moreover, avoiding the use of specific features that are dependant on the distribution channel is possible. However, in order to design an effective strategy to filter spam, the types of contents usually shared on the target distribution channels should be analysed in detail. In particular, legitimate texts distributed in social-media distribution channels (social networks, YouTube comments) usually contain reflections, opinions, and criticisms from users. Therefore, the presence of specific non-text entities (such as URLs) could
provide significant information to improve spam filtering in the analysed scenarios. This conclusion is evident in the results of the experiment carried out with the YouTube comments dataset. In addition, for this feature we have achieved significant differences from a statistical point of view.

The use of some popular features that are common in spam texts of a specific dataset (such as the existence of words written in capital letters) could lead to a very poor increase in accuracy when used in contexts that are similar to those analysed. Moreover, identifying relevant data (sums of money, addresses, dates, etc.) using NER in large texts (e-mails) did not achieve significant improvements in the datasets analysed. Despite this, we believe that these properties could successfully allow classifiers to detect legitimate contents (mainly those containing this information). These entities are commonly used in messages exchanged in a professional context. However, SpamAssassin was not built with messages extracted from this kind of context and therefore, classifiers can take advantage of NER information to improve filtering accuracy.

Furthermore, some more complex features such as LanguageReliability or CompressionRatio could successfully increase performance in a lot of situations due to their capabilities to assess and model intrinsic spam properties such as the use of shorter messages with portions of text that are not (or barely) repeated, a multitude of spelling mistakes, and/or tricks to prevent the recognition of text by filters. In particular, CompresionRatio have shown better performance and statistical differences in kapa evaluations in most of the situations analysed and with independence of the dataset and representations used. Finally, we want to highlight the fact that taking advantage of LanguageReliability is more difficult in short texts (such as those included in YouTube Comments Dataset) because the reliability of guessing their language is always poor. In the context of our experimentation, this feature provided better performance when applied on large texts (such as those included SpamAssassin dataset). Additionally, the improvements achieved by using it cannot be considered significant from a statistical point of view.

In particular, we are convinced that it is possible to filter out spam content by taking into account only the text and discarding information that depends on the communication protocols used for its distribution (premise in this work). Our interest is to advance in this line of research and we believe the outcomes of this study can be successfully used to improve spam filtering in multiple channel distributions. Particularly, findings included in this study needs to be verified in some additional distribution forms (e.g. SMS messages, web spam, weblog, forum, etc.). The future work also includes experimentation in spam detection using deep learning architectures (such as neural embedding, attention mechanism, self-attention, Transformer, BERT, and XLNet). We find relevant addressing the identification of additional features (feature engineering) that can be used to increase the performance of these approaches and executing a comparative benchmarking with the results included in this study (traditional classification algorithms).

CRediT authorship contribution statement

María Novo-Lourés: Software, Validation, Investigation, Data curation. David Ruano-Ordás: Methodology, Investigation, Writing – original draft, Writing – review & editing, Visualization. Reyes Pavón: Investigation, Writing – original draft, Writing – review & editing. Rosalía Laza: Investigation, Writing – original draft, Writing – review & editing, Funding acquisition. Silvana Gómez-Meire: Investigation, Writing – review & editing. José R. Méndez: Conceptualization, Methodology, Investigation, Supervision, Writing – original draft, Writing – review & editing, Project administration, Funding acquisition.

Acknowledgements

D. Ruano-Ordás was supported by a postdoctoral fellowship from Xunta de Galicia (ED481D-2021/024). Additionally, this work was partially endorsed by the project Semantic Knowledge Integration for Content-Based Spam Filtering (TIN2017-84658-C2-1-R) from the Spanish Ministry of Economy, Industry and Competitiveness (SMEIC), State Research Agency (SRA) and the European Regional Development Fund (ERDF).

SING group thanks CITI (Centro de Investigación, Transferencia e Innovación) from University of Vigo for hosting its IT infrastructure. Funding for open access charge: Universidade de Vigo/CISUG.

References

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

Adewole, K. S., Anuar, N. B., Kamsin, A., & Sangaiah, A. K. (2019). SMSAD: A framework for spam message and spam account detection. Multimedia Tools and Applications, 78(4), 3925–3960. https://doi.org/10.1007/s11042-017-5018-x

Akbar Septiandri, A., & Wibisono, O. (2017). Detecting spam comments on Indonesia’s Instagram posts. Journal of Physics: Conference Series, 801, Article 012069. https://doi.org/10.1088/1742-6596/801/1/012069

Aleyon, F. C., & Esat Kallauglu, M. (2019). Instagram Fake and Automated Account Detection. In 2019 Innovations in Intelligent Systems and Applications Conference (ASIUY) (pp. 1–7). https://doi.org/10.1109/ASIUY48272.2019.8946437

Almaatouq, A., Shmueli, E., Noah, M., Alabdulkareem, A., Singh, V. K., Alsaleh, M., et al. (2016). If it looks like a spammer and behaves like a spammer, it must be a spammer: Analysis and detection of microblogging spam accounts. International Journal of Information Security, 15(5), 475–491. https://doi.org/10.1007/s10207-016-0321-5

Almeida, T. A., Lochter, J. V., & Túlio, C., A. (2015). TubeSpam: Comment spam filtering on YouTube. In Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA-15) (pp. 1–6).

Alom, Z., Carminati, B., & Ferrari, E. (2018). Detecting spam accounts on twitter. In +(ASONAM) (pp. 1191–1198). https://doi.org/10.1109/ASONAM.2018.8508495

Alqatrawna, J., Faris, H., Jaradat, K., Al-Zewairi, M., & Adwan, O. (2015). Improving Knowledge Based Spam Detection Methods: The Effect of Malicious Related Features in Imbalance Data Distribution. International Journal of Communications, Network and System Sciences, 08(05), 118–129. https://doi.org/10.4236/ijcns.2015.85014
Perveen, N., M. M., Rasool, Q., & Akhtar, N. (2016). Sentiment based twitter spam detection. *International Journal of Advanced Computer Science and Applications, 7*(7). https://doi.org/10.14569/IJACSA.2016.070777

Prieto, V.M., Álvarez, M., López-García, R., & Cacheda, F. (2012). Analysis and detection of web spam by means of web content (pp. 43–57). https://doi.org/10.1007/978-3-642-31274-8_4

Ruano-Ordás, D., Fdez-Riverola, F., & Méndez, J. R. (2018). Using evolutionary computation for discovering spam patterns from e-mail samples. *Information Processing and Management, 54*(2). https://doi.org/10.1016/j.ipm.2017.12.001

Samsudin, N. M., Mohd Foozy, C. F. binti, Alias, N., Shamala, P., Othman, N. F., & Wan Din, W. I. S. (2019). Youtube spam detection framework using naïve bayses and logistic regression. *Indonesian Journal of Electrical Engineering and Computer Science, 14*(3), 1508. https://doi.org/10.11591/ijeecs.v14i3.pp1508-1517

Subba Reddy, K., & Srinivasa Reddy, E. (2019). Detecting spam messages in twitter data by machine learning algorithms using cross validation. *International Journal of Innovative Technology and Exploring Engineering, 8*(12), 2941–2946. https://doi.org/10.35940/ijitee.K1913.1081219

Thomas, K., Grier, C., Ma, J., Paxson, V., & Song, D. (2011). Design and evaluation of a real-time URL spam filtering service. In *2011 IEEE Symposium on Security and Privacy* (pp. 447–462). https://doi.org/10.1109/SP.2011.25

Tian, Y., Mirzabagheri, M., Tirandazi, P., & Bamakan, S. M. H. (2020). A non-convex semi-supervised approach to opinion spam detection by ramp-one class SVM. *Information Processing & Management, 57*(6), Article 102381. https://doi.org/10.1016/j.ipm.2020.102381

Vázquez, I., Novo-Lourès, M., Pavón, R., Laza, R., Méndez, J. R., & Ruano-Ordás, D. (2021). Improvements for research data repositories: The case of text spam. *Journal of Information Science,*. Article 0165551521998636. https://doi.org/10.1177/0165551521998636

Vinodhini, M., Prithvi, D., & Balaji, S. (2020). Spam detection framework using ML Algorithm. *International Journal of Recent Technology and Engineering, 8*(6), 5326–5329. https://doi.org/10.35940/ijrte.F1120.038620

Wijshoven, F., & Pieper, A. (2019). Review spam criteria for enhancing a review spam detector. In *39th International Conference on Information Systems* (p. 15).

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). Beyond supervised and unsupervised learning. *Data mining* (pp. 467–478). Elsevier. https://doi.org/10.1016/8978-0-12-804291-5.00011-8

Wu, H., Liu, Y., & Wang, J. (2020). Review of text classification methods on deep learning. *Computers, Materials & Continua, 63*(3), 1309–1321. https://doi.org/10.32604/cmc.2020.010172

Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). CANTINA+. *ACM Transactions on Information and System Security, 14*(2), 1–28. https://doi.org/10.1145/2019599.2019606

Zainal, K., & Jali, M. Z. (2016). A review of feature extraction optimization. *SMS spam messages classification* (pp. 158–170). https://doi.org/10.1007/978-981-10-2777-2_14

Zhang, W., & Sun, H.-. M. (2017). Instagram spam detection. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 227–228). https://doi.org/10.1109/PRDC.2017.43

Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management, 57*(2), Article 102025. https://doi.org/10.1016/j.ipm.2019.03.004