Neural Computing and Applications (2020) 32:4239–4257
https://doi.org/10.1007/s00521-019-04331-5

EMERGING TRENDS OF APPLIED NEURAL COMPUTATION - E_TRAINCO

Spam detection on social networks using cost-sensitive feature selection and ensemble-based regularized deep neural networks

Aliaksandr Barushka[1] · Petr Hajek[1]

Received: 1 November 2018 / Accepted: 28 June 2019 / Published online: 2 July 2019

Abstract
Spam detection on social networks is increasingly important owing to the rapid growth of social network user base. Sophisticated spam filters must be developed to deal with this complex problem. Traditional machine learning approaches such as neural networks, support vector machines and Naïve Bayes classifiers are not effective enough to process and utilize complex features present in high-dimensional data on social network spam. Moreover, the traditional objective criteria of social network spam filters cannot cope with different costs assigned to type I and type II errors. To overcome these problems, here we propose a novel cost-sensitive approach to social network spam filtering. The proposed approach is composed of two stages. In the first stage, multi-objective evolutionary feature selection is used to minimize both the misclassification cost of the proposed model and the number of attributes necessary for spam filtering. Then, the approach uses cost-sensitive ensemble learning techniques with regularized deep neural networks as base learners. We demonstrate that this approach is effective for social network spam filtering on two benchmark datasets. We also show that the proposed approach outperforms other popular algorithms used in social network spam filtering, such as random forest, Naïve Bayes or support vector machines.

Keywords Neural network · Social networks · Regularization · Ensemble learning · Misclassification cost

1 Introduction

Generally, spam can be defined as unwanted and unsolicited messages sent to usually a large number of recipients [1]. Spam message can be sent over multiple communication channels, such as e-mail, SMS and social networks. Statistics show that a large proportion of all messages in social networks are spam messages. For instance, the study by Proofpoint, a major company specialized in cyber security, reported that during the first half of 2013 there has been a 355% growth of social spam. For every seven new social media accounts, five new spammers are detected [2].

✉ Petr Hajek
petr.hajek@upce.cz

Aliaksandr Barushka
aliaksandr.barushka@student.upce.cz

[1] Institute of System Engineering and Informatics, Faculty of Economics and Administration, University of Pardubice, Studentská 84, 532 10 Pardubice, Czech Republic

The growing opportunities of social networks and their popularity have attracted many users. These days, the base of social network users is steadily growing and considerable amount of communication is done through social networks. For instance, the Twitter microblogging service averaged at 335 million monthly active users in the second quarter of 2018 [3]. However, along with legitimate and useful information, inappropriate and unwanted content is also released on these networks. Indeed, spam senders target social network users as well. Moreover, business social networks like Linkedin are also affected [4]. This has serious economic and social consequences. Spam messages decrease work productivity, increase IT support-related resources (help desk) and may even result in security incidents. Moreover, personal privacy can be threatened and the development of social networks can be hindered [5]. This is why a considerable attention is given to spam filtering in social networks.

Spam messages can be filtered either manually or automatically. Obviously, manual spam filtering (also known as crowdsourcing) by identifying spam message and

removing it is a time consuming task. Moreover, spam messages may contain a security threat, such as links to phishing web sites or servers hosting malware. Therefore, over a number of decades researches and practitioners have worked on improving automatic spam filtering algorithms. Notably, two types of methods occur in the literature specifically addressing the problem social network spam detection, namely graph-based and machine learning methods [6]. The first models social network as a social graph [7, 8], making use of graph properties, graph clusters and trust relationships. However, the majority of earlier work has been aimed at using machine learning techniques. In fact, those are particularly known to be highly accurate in detecting spam messages [6]. There are a number of existing machine learning algorithms applied to spam filtering, including neural networks [9], support vector machines (SVMs) [10], Naïve Bayes [11] and random forest [12].

Spam filtering task belongs to binary classification problem; each message should be identified either as spam or legitimate. Besides, a high accuracy algorithm should also perform well when it comes to false positive ratio (legitimate message is classified as spam) to avoid situations where legitimate message is not delivered to the intended receiver. The main concept of the machine learning algorithms is to make use of content-based features; this is to build a word list and assign a weight to each word accordingly. However, spammers tend to include common legitimate messages into the spam message in order to decrease the probability of being detected. In social network spam filtering, additional attributes are therefore used, such as those related to sender's profile and behaviour in the social network, see [6] for an overview.

The state-of-the-art methods in social network spam filtering have recently been surveyed by [6, 13, 14]. According to the surveys, ensemble learning methods, such as bagging and random forest, outperform traditional single classifiers. The ensemble methods combine the predictions of several base machine learning algorithms in order to improve accuracy and robustness over single algorithms. In previous studies, ensemble methods employed traditional classifiers like decision trees to effectively filter spam messages. However, surprisingly little attention has been paid to neural networks with ensemble learning. Recent evidence showed that neural networks equipped with regularization techniques may be highly accurate in detecting e-mail and SMS spam [9]. This can be attributed to better optimization convergence and resistance to overfitting. To take advantage of these qualities, here we integrate regularized neural networks with ensemble learning methods for social network spam filtering. Another problem is the choice of the objective function of the prediction model. Using accuracy, a traditional classification performance measure does not take account of different costs associated with type I and type II errors. Moreover, using accuracy for often highly imbalanced spam datasets might lead to erroneous conclusions because the minority class (usually the class of spam messages) has little effect on accuracy compared to the majority class of legitimate messages [15]. Therefore, more accurate measures are needed to improve the effectiveness of spam filtering methods. As earlier work failed to address this problem, here we propose a cost-sensitive social network spam filter that considers misclassification cost (combination of type I and type II errors) in both stages of spam detection, feature selection and classification. Feature selection aims to both reduce the dimensionality of the feature space and improve the prediction performance of the model [16]. Therefore, for the first stage, we adapt a multi-objective evolutionary feature selection (MOEFS) algorithm called Evolutionary NOn-dominated Radial slots-based Algorithm (ENORA) [17] to minimize both the number of attributes necessary for spam filtering and the misclassification cost of the model. Then, in the second stage, cost-sensitive ensemble learning is performed with regularized deep neural networks (RDNNs) as base learners. In summary, the contributions of this study are:

- Developing a novel social network spam detection method integrating a modified MOEFS algorithm and cost-sensitive ensemble-based RDNNs. The novelty of our method lies in considering cost-sensitive learning in both feature selection and spam classification. This is also the first study using ensemble-based RDNNs for spam detection.
- Using the benchmark data from the social networking sites Hyves and Twitter, here we show that the proposed approach can be more effective than state-of-the-art spam filtering methods in terms of misclassification cost, as well as other classification performance measures.

This paper is a significantly extended version of [18]. The earlier version was limited to ensemble-based RDNNs without considering cost-sensitive learning and feature selection. The extension further includes an in-depth literature review. Furthermore, here we propose a modified MOEFS algorithm and examine the effect of feature selection on spam detection performance. We also compare the results with traditional feature selection methods and examine the effect of different misclassification cost ratios on the performance of spam detection models.

The rest of this paper is organized in the following way. Section 2 reviews recent development in spam detection on social networks. Section 3 presents the benchmark datasets. In Sect. 4, we introduce the proposed spam filter. Section 5 shows the results of the experiments and a

comparative analysis with several state-of-the-art methods used for social network spam filtering. In Sect. 6, we conclude this paper and discuss possible future research directions.

2 Spam detection on social networks: a literature review

User base of social networks is growing over the number of years. For instance, Facebook, one of the biggest social networks in the world, grew from one billion to two billion users just in 5 years [19]. Social network spam has become a major concern of industry and academia because it may include unwanted content, such as insults, hate speech or malicious links. Such messages can be seen by the recipient's followers. Moreover, they may lead to confusions and misdirection in public discussions [20]. Fighting social network spam with traditional legal methods has serious limitation, since spam messages in social networks can be sent from different countries. It is important to note that spammers may use anonymizers, making it difficult to trace them. To overcome this problem, several social network spam filters have recently been developed [6, 14]. A list of related studies is shown in Table 1, presenting the method used, the datasets and the resulting performance evaluation.

Features related to tweet content and user behaviour were identified and used in machine learning by SVM [24]. Song et al. [28] utilized relation features, such as the connectivity and distance between a tweet sender and receiver, to detect spam messages. A statistical analysis of language used in tweets represents an alternative approach [32], which identifies spam tweets in isolation (i.e. without user information) using their trending topics. Similarly, Antonakaki et al. [38] exploited trending topics to detect spam campaigns in Twitter.

An SVM classifier was used by Lee and Kim [33] to detect suspicious URLs in tweets. Their system makes use of correlated URL redirect chains extracted from tweets. URLs in social media have also been used in the behaviour-based spam detection system proposed in [37]. More precisely, the behavioural signals were obtained from both the URL sender and receiver. In other words, a high accuracy was achieved without using other tweets' attributes such as those based on message content.

In addition to spam messages detection, recent studies have also considered an alternative task of social spammer (profile) detection. A Naïve Bayes classifier was proposed in [23] to detect spammers in Twitter. In [54], the so-called "social bridges" were identified to detect spammers in Twitter. These are reported as the major supporters of malicious users and a graph-topology-based classifier was used to detect such bridge linkages. A hybrid approach for identifying spam profiles was proposed in [49], combining social media analytics and firefly algorithm with chaotic maps for spam detection in Twitter marketing. A large Twitter dataset was used in [5] to demonstrate that feature distributions between spammers and legitimate users are different. These feature distributions were used in a social spammer detection framework that integrated this information with a social regularization term incorporate into a classification model. Another way to tackle the issue of detecting spammers in Twitter was described in [50]. A multilayer social network was defined, and the identification of spammers was based on the existence of overlapping community-based features of users represented in the form of Hypergraphs, such as structural behaviour and URL characteristics. A unified approach was proposed in [40], utilizing the fact that social spammers tend to post more spam messages. Indeed, it was shown that combining social spammer filtering with spam message filtering improves the performance of both tasks.

Although Twitter represents the most frequently used source of data, alternative social networks have also been examined. For example, data from Sina Weibo were used to study features related to message content and user behaviour in [20, 40]. The most important features were then used in the SVM classifier for spam detection. Extreme learning machines were used by [41] on a similar dataset. A semi-supervised social media spammer filtering approach was developed in [46]. This approach outperformed traditional supervised classifiers for the spammer detection task. Similar results were obtained for spam message detection in Hyves social network [30]. Using the same dataset, significant improvements were achieved by combining data oversampling with RDNNs [53].

In recent years, there has been an increasing interest in dimensionality reduction techniques with the aim of improving the prediction performance and stability of social network spam filters [16]. Several researchers employed feature selection and extraction methodologies to identify the most important features for social network spam filtering. The concept of rough set theory was applied in [8], concluding that the used methodology selected a smaller subset of features than those of the baseline methodologies (information gain, consistency subset evaluation, correlation-based feature selection (CBF), community detection and χ^2 evaluation). By considering important features of the posts and their corresponding comments, and finally applying the feature selection techniques, the method proposed in [52] selected the most effective features to detect spam using machine learning techniques. A probabilistic generative model (latent Dirichlet allocation) was proposed in [42] to detect the latent semantics from user-generated comments. Incremental learning was then used to address the issue of the

Table 1 Summary of previous studies on social network spam filtering

Study	Feature selection	Classification method	Dataset (spam/legitimate)	Performance
[21]	No	RF	Facebook profiles (173/827)	FPR = 2%, FNR = 1%
			Twitter profiles (500/500)	FPR = 2.5%, FNR = 3%
[22]	No	Decorate	MySpace profiles (627/388)	Acc = 99.21%
			Twitter profiles (168/104)	Acc = 88.98%
[23]	No	NB	Twitter profiles (14/486)	F1-score = 0.917
[24]	IG, χ^2	SVM	Twitter messages (355/710)	Acc = 87.2%
[25]	No	Boosting RF	Twitter profiles (22,223/19,297)	Acc = 98.42%
[26]	No	Active learning	Facebook profiles	–
[27]	No	Suspension algorithm	Twitter profiles (100/200)	–
[28]	Relief, IG, χ^2	LogitBoost, Bayes Net	Twitter messages (10 K/10 K)	TPR = 99.7%, FPR = 0.6%
[29]	No	RF	Twitter campaigns (744/580)	Acc = 94.5%
[30]	No	SSL	Hyves messages (698/497)	AUC = 0.801
[31]	No	RF	Twitter profiles (2060/20,000)	F1-score = 0.900
[32]	No	SVM	Twitter messages (168 K/340 K)	F1-score = 0.883
[33]	No	SVM	Twitter messages (26,950/156,896)	Acc = 91.87%
[34]	No	ADTree	Facebook profiles (1000/1000)	AUC = 0.985
[35]	No	NB, C4.5	Facebook profiles (165/155) and Twitter profiles (160/145)	Acc = 95.7%
[36]	No	DenStream + K-means	Twitter profiles (208/3031)	Acc = 98.00%
[37]	No	RF	Twitter messages (124/214)	F1-score = 0.859, AUC = 0.921
[20]	No	SVM	Sina Weibo profiles (11,488/17,646)	F1-score = 0.996
[38]	No	DT	Twitter (63,612/6.6 M)	TPR = 81%, FPR = 0.59%
[39]	No	ELM	Sina Weibo profiles (14,796/64,419)	F1-score = 0.996
[40]	No	Co-detection of spammers and messages	Sina Weibo messages (25,681/27,803)	F1-score = 0.927
			Sina Weibo profiles (1496/3594)	F1-score = 0.795
[41]	No	ELM	Sina Weibo messages (500/500)	F1-score = 0.996
[42]	χ^2	SVM	YouTube messages (210,283/845,092)	Acc = 88.05, AUC = 0.872
[7]	No	Unsupervised graph-based approach	Twitter profiles (2072/17,322; 1617/19,312; 3109/12,128)	Acc = 92.3%
[16]	GI, IG, MDA	RF	Twitter messages (30 K/120 K)	AUC = 0.920
[43]	No	RF + unsupervised learning	Twitter messages (1 M/1 M)	Acc = 95%
[44]	χ^2	RF	Twitter messages (3648/4000)	Acc = 93.2%, AUC = 0.983
[5]	No	SVM	Twitter profiles (4414/5666)	F1-score = 0.879
[45]	No	RF	Facebook messages (600/600)	F1-score = 0.987
[46]	No	SSL	Sina Weibo profiles (135/2865)	F1-score = 0.920
[47]	IG, χ^2	RF	Twitter (9945/90,055)	Acc = 97.11%, F1-score = 0.838
[48]	WOA-wrapper	SVM + WOA	Twitter profiles (204/196)	Acc = 93.73%
[49]	No	K-Means + FA	Twitter profiles (4923/9312)	Acc = 97.87%
[50]	No	Unsupervised SpamCom	Twitter profiles (22,223/19,276)	F1-score = 0.880
[8]	RST	Graph-based greedy algorithm	Twitter messages (93,791/250 K)	Acc = 81.04%
[51]	No	SSL	Twitter messages (48,849/22,185)	Acc > 95%
[52]	PSO	Decision tree	Facebook profiles (200 K)	Acc = 92%
[53]	CBF, IWSS	RDNN	Hyves messages (466/355)	Acc = 92.81%, AUC = 0.961

Table 1 (continued)

Study	Feature selection	Classification method	Dataset (spam/legitimate)	Performance
This study	MOEFS	Cost-sensitive ensembles of RDNNs	Hyves messages (466/355) Twitter messages (4198/57,476)	

Acc accuracy, AUC area under ROC curve, *CBF* correlation-based filter, *ELM* extreme learning machine, *FA* firefly algorithm, *GI* Gini index, *IG* information gain, *IWSS* incremental wrapper feature subset selection, *MDA* mean decrease accuracy, *NB* Naïve Bayes, *PSO* particle swarm optimization, *RF* random forest, *RST* rough set theory, *SSL* semi-supervised learning, *WOA* whale optimization algorithm

changing feature space. Three traditional feature selection methods were used in [16], including information gain, Gini index and mean decrease accuracy. The latter measures attribute importance based on the accuracy of the RF classifier. Evolutionary search algorithm was used in combination with χ^2 evaluation criterion by [44] to identify the reduced set of attributes for spam filtering in Twitter microblogging social network. Even better accuracy than the previously mentioned filter-based methods can be achieved using wrapper-based feature selection [48]. However, this approach is reportedly computationally intensive because the classifier must be trained on each feature subset. The main limitation of the wrapper-based approach proposed in [48] is the use of classification accuracy as the evaluation measure due to its unsuitability for different misclassification cost of spam and legitimate classes.

Regarding the classification methods used to categorize spam and legitimate messages (profiles), traditional machine learning methods have dominated in earlier research, such as Naïve Bayes, SVM and random forest. To make use of unlabelled messages in the dataset, several studies have used methods with unsupervised learning in addition to supervised learning [43, 51]. Ensemble-based approaches, such as Decorate [22] and boosting [25], have been effectively used in a few studies, demonstrating that those methods can be more accurate in detecting spam than single classifiers. This can be attributed to the diversity of the base learners that reduces the problem of overfitting. However, the main limitation of the mentioned studies is the application of decision trees as base learners, which suffer from several drawbacks, such as poor capacity to deal with high-dimensional datasets [53].

In summary, previous related literature attempted to overcome the problem of high-dimensional data (the curse of dimensionality) by selecting the most important features, regardless of whether content-based features or user behaviour features. This was mainly due to the risk of overfitting or poor convergence of the used classification methods. However, useful information may be hidden in higher-order features that can be extracted by using deep neural networks [9]. In fact, additional hidden layers enable the recombination of features and thus to capture higher complexity and abstraction in high-dimensional datasets [55]. Moreover, ensemble methods have become popular in social network spam detection tasks due to their capacity to reduce the risk of overfitting and variance [14]. In order to take advantage of these approaches, here we use RDNNs as base learners in several ensemble learning schemes, including boosting, bagging and random subspace.

3 Datasets

In this study, we used two benchmark datasets, one obtained from Hyves, the Dutch social networking site, while the other obtained from Twitter.

3.1 Hyves dataset

The original Hyves dataset contained both labelled and unlabelled messages.[1] As we use a supervised learning approach here, we excluded the unlabelled (unannotated) messages from the dataset. Since the details on this dataset can be found in [30], we provide only a brief description in this study.

The dataset includes the following types of information: message content, spam report and user information. The messages were collected from publicly accessible profile or group pages. At least two spam reports were created for each message to obtain reliable categorization of messages into "spam" or "not spam". The user policy of Hyves was used to define spam messages. Specifically, unsolicited and promotional messages were labelled as spam.

The Hyves dataset contained 466 spam messages and 355 legitimate messages. The messages were represented as the arrays of JSON objects with the following fields: the annotation of the object (either spam or legitimate), anonymized IDs of the reporters of the message, anonymized ID of the author of the message and bag-of-words representation of the message (an anonymized ID was assigned to each word).

[1] http://ilps.science.uva.nl/framework-unsupervised-spam-detection-social-networking-sites/.

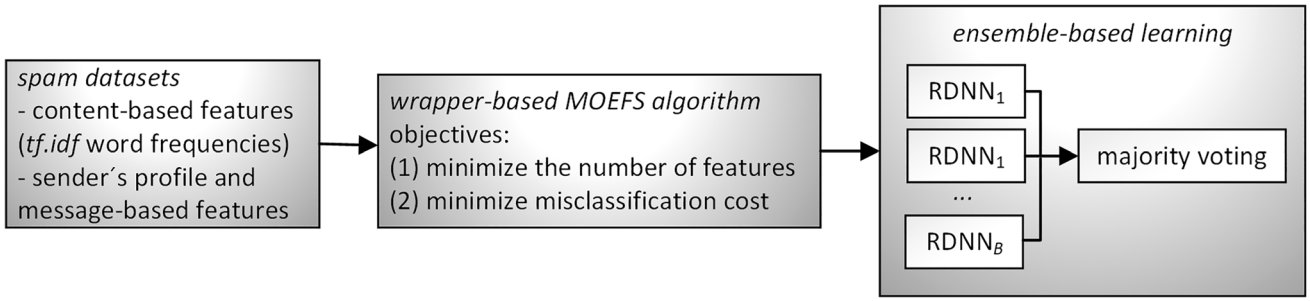

Fig. 1 Conceptual framework for social network spam filtering

3.2 Twitter dataset

The Twitter dataset was originally used by [47]. The original Twitter dataset consisted of 100,000 tweets collected from 92,720 users in May 2016. The authors labelled the dataset manually and provided the links to the used tweets along with their labels.[2] Among those messages, 90,055 were identified as legitimate. We attempted to download all the tweets in July 2018. However, many messages were filtered and removed by that time. As a result, our dataset consisted of 61,675 tweets, 4198 of them labelled as spam.

3.3 Data preprocessing

To represent message content in both datasets, we used tf.*i*df weighting scheme in the bag-of-words fashion. The weight v_{ij} for the i-the word in the jth message can be calculated as follows:

$$v_{ij} = \left(1 + \log\left(\mathrm{tf}_{ij}\right)\right) \times \log(N/\mathrm{df}_i), \quad (1)$$

where tf_{ij} represents term frequency, df_i is document frequency, and N is the number of messages. We used top 2000 words according to their weights. This number was reported to be sufficient to perform document classification in previous studies [56].

For the Hyves dataset, the information about spam reports and users were considered as additional attributes, namely the number of user spam reports (it is assumed that the more spam reports received, the more likely the message is spam) and the number of all messages written by author (assuming that author with many spam messages is more likely to post another spam message).

In addition to the text representation of the Twitter dataset, the information included in the JSON objects was extracted in accordance with the original study [47]. Those features can be categorized into: (1) tweet-based, such as type of tweet, numbers of retweets, hashtags and mentions, and (2) profile-based, such as the numbers of followers, friends and statuses (see [47] for all the details).

[2] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0182487#pone.0182487.s003.

4 Methods

Since the main interest of the paper is the proposal of a social network spam filter based on the combination of MOEFS and cost-sensitive RDNN with ensemble learning, we provide a description of these methods in this section. The proposed conceptual framework for social network spam filtering is presented in Fig. 1.

4.1 Multi-objective evolutionary feature selection

Related studies have demonstrated that the prediction performance of models for spam filtering can be significantly improved using feature selection [16]. This step aims to improve model accuracy and stability by choosing a subset of only relevant input attributes. Thus, the dimensionality of the feature space is reduced. This may have considerable importance for the spam filtering models, because social network administrators can better understand the characteristics of spam messages and the attributes' acquisition time and cost can be reduced.

In literature [57], three categories of feature selection methods are distinguished, namely filters, embedded methods and wrappers. For filters, no learning algorithms are necessary to be involved because only data characteristics are used to evaluate the importance of attributes. In contrast, a specific prediction model is used in embedded methods. The relevance of each attribute is then evaluated while training that model. Wrappers are more computationally intensive, compared to the two approaches, because they employ enumeration algorithms to find the subset of attributes with the best performance of the prediction model.

As indicated above, two main objectives should be taken into account in feature selection for social network spam filtering: minimizing the prediction error and minimizing the number of selected features. The latter objective is important for improving the prediction model's stability and interpretability. However, as suggested in [17], these two objectives are often in conflict with each other. To

address this issue, MOEFS has been successfully used in wrapper-based feature selection [17, 58]. Notably, the ENORA algorithm was recently employed for this task [58], outperforming other baseline algorithms.

A wrapper-based approach has been effectively applied to optimize the SVM performance in social network spam filtering [48]. However, to the best of our knowledge, no related work has investigated the use of multiple objectives in feature selection. Here, we propose a wrapper-based MOEFS to minimize the misclassification cost (MC) of the spam filtering model on the one hand while minimizing the number of selected features on the other hand. In other words, we adapt ENORA to feature selection for cost-sensitive classification task. The detailed description of the ENORA algorithm is presented in [17]. Therefore, for the sake of clarity, we summarize only our modifications in this section.

Adopting the recommendation of Jiménez et al. [17], we use one crossing and one mutation operator in the ENORA algorithm. More precisely, self-adaptive crossover and mutation are used, which provides favourable diversity in the population of solutions and maintains the convergence of the algorithm. As a result, only the probability of variation must be decided to control the evolution of the population.

To initialize the population of solutions, the number of individuals (solutions) in the population must be fixed. Each solution is represented by a fixed-length bit set, one bit for each attribute (1 if the attribute was selected and 0 if not selected). Thus, the length corresponds to the number of attributes. Additionally, two discrete parameters d_I and e_I represent crossover and mutation, respectively. As a result, an individual I is represented as follows:

$$I = \left\{b_I^1, b_I^2, \ldots, b_I^m, d_I, e_I\right\}, \tag{2}$$

where $b_I^j \in \{0,1\}, j = 1,2,\ldots,m$, m is the number of attributes, $d_I \in \{0,1\}$, and $e_I \in \{0,1\}$. Each individual is evaluated using two fitness functions, $f_1(I)$ and $f_2(I)$, representing two objectives in MOEFS:

$$\begin{cases} f_1(I) = \mathrm{MC}(I) \\ f_2(I) = C(I) \end{cases}, \tag{3}$$

where $C(I)$ is the cardinality of the subset of attributes (i.e. the number of selected attributes) and the calculation of MC(I) is as follows.

Table 2 Confusion matrix for social network spam filtering

Prediction/actual	Positive	Negative
Positive (spam)	TP	FP (type I error)
Negative (legitimate)	FN (type II error)	TN

TP, FP, FN and TN are the numbers of messages classified as true positive, false positive, false negative and true negative

On the one hand, the wrong prediction of a message that is spam (type II error) leads to the loss of time because the user needs to read the message, delete it and report a spam message (or spamming profile), respectively. On the other hand, predicting a spam message when it would be legitimate (type I error) may result in its automatic filtering and ignoring by the user or eventually in its automatic deletion. This case is considered more serious than the former one because we want to avoid labelling legitimate message as spam [59]. Several spam filtering studies have combined those two errors into a MC [59, 60], which is considered a crucial criterion in the evaluation of spam filtering effectiveness [60]. However, this measure has rarely been utilized as the objective criterion in learning by classification models [59]. In other words, previous literature has inadequately studied cost-sensitive social network spam filtering.

Table 2 shows the confusion matrix used to calculate MC(I), which combines type I and type II errors as follows:

$$\mathrm{MC}_\lambda(I) = \frac{1}{1+\lambda} \times \mathrm{FNR} + \frac{\lambda}{1+\lambda} \times \mathrm{FPR}, \tag{4}$$

where λ is a misclassification cost ratio comparing the degree of seriousness of type I error compared to type II error, and FNR and FPR denote false positive and false negative rate, respectively:

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP}+\mathrm{FN}}, \tag{5}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}. \tag{6}$$

In our experiments, we adopted the values of the misclassification cost ratio λ considered in previous studies [59, 60], resulting in three different ratios used, $\lambda = 1$, $\lambda = 3$ and $\lambda = 7$. Note that for $\lambda = 1$, $\mathrm{MC}_\lambda(I)$ is the average of FNR and FPR. The proposed ENORA algorithm for MOEFS can be defined as follows:

Algorithm 1: ENORA for MOEFS

Input: The number of individuals in population N; the number of attributes in dataset m; the number of iterations T
Output: The population of non-dominated individuals P_f
Initialize population P;
Evaluate all individuals of P using f_1 and f_2;
For t=1 to T {
  $Q \leftarrow \emptyset$;
  For I=1 to N {
  *Parent*1, *Parent*2 ← Binary tournament selection from P;
  *Child*1, *Child*2 ← Self-adaptive variation *Parent*1, *Parent*2;
  Evaluate *Child*1, *Child*2 using f_1 and f_2;
  $Q \leftarrow Q \cup \{Child1, Child2\}$;
  $I \leftarrow I + 2$;
  }
  $R \leftarrow P \cup Q$;
  $P \leftarrow N$ Best individuals from R according to the rank-crowding-better function in R;
}
$P_f \leftarrow P$;

First, the initial population P is randomly generated as described in [17, 58]. Specifically, a number $q \in \{1, 2, \ldots, m\}$ is randomly generated, and q random bits in the individual I are fixed to 1, while the remaining m–n bits are fixed to 0. Parameters d_I and e_I are randomly generated from {0, 1}. Then, binary tournament selection is used to obtain parents *Parent*1 and *Parent*2. The parents are crossed and mutated using self-adaptive variation, and the offspring is evaluated and added to an initially empty auxiliary population Q. Then, the rank-crowding-better function [58] is used to choose the N best individuals of an auxiliary population R. This function is based on the crowding distance of an individual I in a population P:

$$\text{crowding\_distance}(P, I) = \frac{f_1^{\sup_1^I} - f_1^{\inf_1^I}}{f_1^{\max} - f_1^{\min}} + \frac{f_2^{\sup_2^I} - f_2^{\inf_2^I}}{f_2^{\max} - f_2^{\min}}, \tag{7}$$

where $f_1^{\sup_1^I}$ and $f_2^{\sup_2^I}$ (resp. $f_1^{\inf_1^I}$ and $f_2^{\inf_2^I}$) are the values of the objective functions for the individual higher (lower) adjacent in the objective function to the individual I.

4.2 Cost-sensitive ensemble-based regularized deep neural networks

The model of RDNN used in this study is the multilayer perceptron neural network with multiple hidden layers that process complex relations between the input features and output categories. However, such a structure results in the large number of connections, leading to sampling noise. Therefore, intensive adaptation of training data may result in overfitting. To address this issue, we used dropout regularization. Indeed, increased accuracy may be achieved by dropping units from the neural network, including all their incoming and outgoing connections. The dropout regularization randomly changes the given ratio of the activations' values to zero, while training is performed and therefore hidden units that produce the same result are ignored. In addition, we employed rectified linear units instead of traditional sigmoidal units in order to avoid poor local minima of training error and slow optimization convergence [61]. This is done by producing partial derivative equal to one, in case the rectified linear unit is activated. It is also worth to add that these units saturate when reaching one. This might be useful when hidden activations are selected as input features for the classifier. The mini-batch gradient descent was used as a training algorithm for the RDNN. Connection weights are updated for every mini-batch of training data in the following way:

$$w_{t+1} = w_t - \eta \nabla_\theta J\left(w_t; x^{(i:i+n)}; y^{(i:i+n)}\right), \tag{7}$$

where w is connection weight, t denotes time, η is learning rate, J is an objective function, x^i and y^i are the inputs and output of the i-the data sample within every mini-batch, and n is the number of data samples in the mini-batch. By using the mini-batches, a stable convergence can be achieved during the RDNN learning.

The goal of ensemble learning algorithms is to combine the predictions of multiple base estimators constructed with the defined learning algorithm. This approach leads to better generalizability and robustness over single estimators. There are two main classes of ensemble learning algorithms: averaging and boosting. The fundamental concept of averaging is to construct several estimators independently from each other and calculate the average of their predictions. By reducing variance, the combined estimator is more accurate than single base estimator. By contrast, boosting builds the base estimators sequentially. Thus, several sequential weak models are combined to achieve a good ensemble. Here, we used three conventional ensemble learning algorithms, namely AdaBoost M1 [62], bagging [63] and random subspace [64].

The AdaBoost M1 algorithm was developed to produce predictions with high accuracy utilizing a number of weak base learners. The algorithm keeps building the learners until there are no errors in training data predictions or the limit numbers of models are exceeded. This is done by increasing the weights of incorrectly predicted data. Finally, the predictions from all the models are combined by using a weighted majority vote to obtain the final predictions. The algorithm is defined as follows:

The random subspace algorithm was proposed to handle the problem of trade-off between overfitting and achieving the highest accuracy. In fact, the random subspace algorithm is similar to bagging. The main difference is in the way they draw the random subsets of training data. In random subspace, these subsets are produced as the random subsets of the features. The random subspace algorithm applied here for social network spam filtering can be defined as follows:

Algorithm 2: Adaboost M1 with cost-sensitive RDNNs as base learners

Input: The set D of training data $(x^i; y^i)$, i=1,2, … ,n; the number B of base cost-sensitive RDNNs
Output: Ensemble of base cost-sensitive RDNNs $\{C_b\}$
For b=1 to B {
  Construct a base RDNN C_b on weighted training data $D^*=(w_1D^1{}_b, w_2D^2{}_b, \ldots, w_nD^n{}_b)$;
  Calculate the probability estimates of the expected cost $e_b=1/n\ \Sigma\ w_{ib}\times\xi^i{}_b$ ($\xi^i{}_b$=0 if D^i classified correctly, $\xi^i{}_b$=1 if D^i classified as FN, $\xi^i{}_b=\lambda$ otherwise);
  Set weight $c_b=0.5\times\log((1-\mathrm{err}_b)/\mathrm{err}_b)$;
  If $\mathrm{err}_b<0.5$, set $w_{ib+1}=w_{ib}\times\exp(c_b\xi^i{}_b)$;
  Otherwise, set all weights w_{ib}=1 and restart the algorithm;
}
Combine base cost-sensitive RDNNs C_b, b=1,2,…,B into an ensemble $\{C_b\}$ by weighted majority voting;

The main idea behind bagging is to construct multiple instances of black-box estimator on the random subsets of the original training data. To produce an aggregated prediction, separate predictions are then combined by using the voting procedure. Thus, the variance of base estimator is reduced by applying randomization during the process of building ensembles. The bagging algorithm employed here can be defined as follows:

Algorithm 3: Bagging with cost-sensitive RDNNs as base learners

Input: The set D of training data $(x^i; y^i)$, i=1,2, … ,n; the number B of base cost-sensitive RDNNs
Output: Ensemble of base cost-sensitive RDNNs $\{C_b\}$
For b=1 to B {
  Create a bootstrapped replicate D_b of the training data set D;
  Construct a base cost-sensitive RDNN C_b on D_b;
}
Combine base cost-sensitive RDNNs C_b, b=1,2,…,B into an ensemble $\{C_b\}$ by simple majority voting;

Algorithm 4: Random subspace with cost-sensitive RDNNs as base learners

Input: The set D of training data $(x^i; y^i)$, i=1,2, … ,n; the number B of base cost-sensitive RDNNs
Output: Ensemble of base cost-sensitive RDNNs $\{C_b\}$
For b=1 to B {
  Select an r-dimensional random subspace D_b from the original training data set D;
  Construct a base cost-sensitive RDNN C_b in D_b;
}
Combine base cost-sensitive RDNNs C_b, b=1,2,…,B into an ensemble $\{C_b\}$ by simple majority voting;

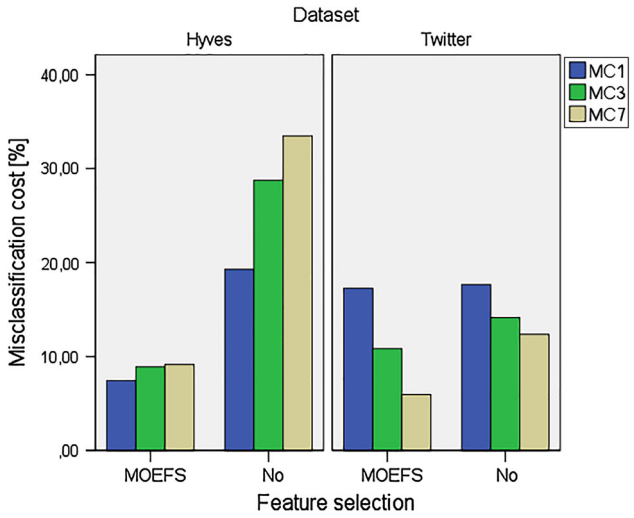

Fig. 2 Misclassification cost for random forest without feature selection versus random forest with MOEFS

The time complexity of the proposed method can be obtained as follows. When disregarding MOEFS, the time complexity of the cost-sensitive ensemble-based RDNN is $O(B \times n_{\text{mb}} \times T \times (m \times n_1 + n_1 \times n_2 + n_2 \times n_3 + n_3 \times n_4))$, where B is the number of the base learners, n_{mb} is the number of mini-batches, T is the number of epochs, m is the number of features, n_1, n_2 and n_3 are the numbers of neurons in the first, second and third hidden layer, respectively, and n_4 is the number of neurons in the output layer. When MOEFS is performed using ENORA with l objectives and N individuals in the population, the time complexity is $O(l \times N^2 + B \times n_{\text{mb}} \times T \times (m_{\text{selected}} \times n_1 + n_1 \times n_2 + n_2 \times n_3 + n_3 \times n_4))$ in the worst case. Note that the time complexity of ENORA for MOEFS is only $O(l \times N)$ in the best case. Hence, besides the number of neurons and hidden layers in the base RDNNs, the time complexity largely depends on the feature reduction rate of MOEFS.

5 Experimental results

In this section, we first describe the setting of all experiments, and then, we present the results in terms of five prediction measures: MC, detection accuracy (Acc), AUC, FNR and FPR, and F1-score. FNR represents the percentage of spam messages incorrectly predicted as legitimate, while FPR is the percentage of legitimate messages incorrectly predicted as spam. F1-score combines precision and recall, where precision is a fraction of messages correctly classified as spam out of all the messages the algorithm classifies as spam, whereas recall is the fraction of messages correctly classified as spam out of all the spam messages. Tenfold cross-validation was used to avoid overfitting and evaluate the prediction performance. To address the problem of the imbalanced Twitter dataset, we used a distribution-based balancing algorithm to undersample the majority class of legitimate messages in the training data. More precisely, the original training data in the majority class were replaced based on learning their Gaussian probability distribution. This reportedly reduced

Table 3 Results of the experiments for $\lambda = 1$

Method	MC_1 (%)	Acc (%)	AUC	FNR (%)	FPR (%)	F1-score
Hyves dataset						
Naïve Bayes	25.97 ± 3.74	76.61 ± 3.41	0.931 ± 0.033	45.1 ± 8.0	**6.9 ± 5.1**	0.819 ± 0.026
SVM	9.70 ± 3.38	**90.13 ± 3.47**	0.903 ± 0.034	8.5 ± 4.2	10.9 ± 5.0	**0.911 ± 0.032**
AdaBoost M1	9.88 ± 3.42	89.16 ± 3.68	0.906 ± 0.037	**2.8 ± 1.8**	16.9 ± 5.4	0.896 ± 0.038
Random forest	**7.43 ± 2.49**	**92.33 ± 2.78**	**0.961 ± 0.026**	5.7 ± 1.8	**9.2 ± 4.7**	**0.930 ± 0.026**
Single RDNN	**9.04 ± 3.01**	**90.86 ± 3.10**	**0.954 ± 0.023**	8.4 ± 3.5	**9.6 ± 4.3**	**0.918 ± 0.029**
RDNN with AB	**8.70 ± 3.80**	**91.23 ± 3.85**	**0.947 ± 0.029**	8.2 ± 3.9	**9.2 ± 4.5**	**0.921 ± 0.035**
RDNN with bagging	**7.39 ± 2.26**	**92.33 ± 2.38**	**0.958 ± 0.023**	5.3 ± 2.3	**9.4 ± 3.6**	**0.930 ± 0.023**
RDNN with RanSub	**7.76 ± 1.72**	**91.84 ± 1.81**	**0.964 ± 0.017**	**4.8 ± 2.2**	10.7 ± 2.8	**0.925 ± 0.017**
Twitter dataset						
Naïve Bayes	23.91 ± 1.12	76.76 ± 0.41	0.818 ± 0.010	24.7 ± 2.4	23.1 ± 0.5	0.818 ± 0.010
SVM	**16.54 ± 0.90**	87.27 ± 0.66	0.835 ± 0.009	**20.9 ± 1.9**	12.1 ± 0.7	0.835 ± 0.009
AdaBoost M1	26.48 ± 1.46	79.22 ± 0.52	0.758 ± 0.019	33.1 ± 2.9	19.9 ± 0.5	0.758 ± 0.019
Random forest	17.26 ± 0.85	**88.17 ± 0.48**	**0.899 ± 0.009**	23.5 ± 1.8	**11.0 ± 0.5**	0.902 ± 0.003
Single RDNN	**16.04 ± 1.07**	**88.98 ± 1.71**	**0.914 ± 0.008**	**21.8 ± 3.0**	**10.2 ± 2.0**	**0.914 ± 0.009**
RDNN with AB	**17.03 ± 0.61**	86.09 ± 1.71	**0.896 ± 0.007**	**20.7 ± 2.9**	13.4 ± 2.0	0.888 ± 0.011
RDNN with bagging	**15.42 ± 1.02**	**90.02 ± 0.50**	**0.918 ± 0.007**	**21.7 ± 2.1**	**9.1 ± 0.6**	**0.915 ± 0.004**
RDNN with RanSub	17.32 ± 1.26	87.58 ± 0.77	**0.906 ± 0.010**	23.0 ± 2.2	**11.6 ± 0.8**	0.898 ± 0.005

Table 4 Results of the experiments for $\lambda = 3$

Method	MC_3 (%)	Acc (%)	AUC	FNR (%)	FPR (%)	F1-score
Hyves dataset						
Naïve Bayes	19.28 ± 3.24	71.01 ± 3.95	0.927 ± 0.037	59.1 ± 8.6	**6.0 ± 3.8**	0.787 ± 0.026
SVM	**10.96 ± 4.32**	**89.16 ± 3.91**	0.892 ± 0.038	**10.4 ± 4.4**	11.1 ± 5.2	**0.903 ± 0.036**
AdaBoost M1	**11.00 ± 3.68**	**88.18 ± 3.63**	**0.938 ± 0.029**	14.4 ± 4.7	9.9 ± 4.1	**0.896 ± 0.032**
Random forest	**8.90 ± 3.14**	**90.38 ± 2.96**	**0.959 ± 0.027**	**11.8 ± 3.3**	**7.9 ± 3.6**	**0.916 ± 0.026**
Single RDNN	**9.81 ± 3.71**	**88.79 ± 3.61**	**0.946 ± 0.026**	15.5 ± 5.0	**7.9 ± 4.2**	**0.903 ± 0.031**
RDNN with AB	**8.83 ± 2.98**	**90.02 ± 3.25**	**0.941 ± 0.026**	**13.5 ± 5.9**	**7.3 ± 3.4**	**0.913 ± 0.028**
RDNN with bagging	**9.28 ± 2.21**	**89.89 ± 1.96**	**0.953 ± 0.022**	**12.7 ± 3.8**	**8.1 ± 3.1**	**0.912 ± 0.017**
RDNN with RanSub	**10.22 ± 4.39**	**86.98 ± 6.01**	**0.956 ± 0.023**	21.6 ± 12.0	**6.4 ± 3.4**	**0.892 ± 0.046**
Twitter dataset						
Naïve Bayes	23.72 ± 0.83	76.83 ± 0.44	0.819 ± 0.012	**26.0 ± 3.9**	23.0 ± 0.6	0.823 ± 0.003
SVM	12.75 ± 0.83	91.32 ± 0.85	0.817 ± 0.013	**29.5 ± 2.8**	7.2 ± 1.0	0.923 ± 0.006
AdaBoost M1	22.80 ± 2.65	89.96 ± 5.45	0.756 ± 0.025	75.4 ± 24.1	5.3 ± 7.4	0.894 ± 0.029
Random forest	**10.85 ± 0.93**	94.16 ± 0.95	**0.903 ± 0.011**	31.5 ± 2.3	4.0 ± 1.0	0.944 ± 0.008
Single RDNN	**10.69 ± 0.71**	94.62 ± 0.74	**0.906 ± 0.010**	32.6 ± 2.3	3.4 ± 0.8	**0.948 ± 0.006**
RDNN with AB	11.86 ± 1.15	92.41 ± 1.89	0.889 ± 0.007	**29.5 ± 2.9**	6.0 ± 2.2	0.932 ± 0.014
RDNN with bagging	**10.10 ± 0.43**	**95.60 ± 0.41**	**0.914 ± 0.011**	33.6 ± 1.6	**2.3 ± 0.5**	**0.956 ± 0.003**
RDNN with RanSub	**10.86 ± 0.76**	**96.16 ± 0.37**	**0.910 ± 0.010**	39.8 ± 2.7	**1.2 ± 0.3**	**0.959 ± 0.004**

Bold values significantly better performance at $p = 0.05$ level

overlapping between spam and legitimate class in related spam filtering studies [53, 65].

In the first run of experiments, MOEFS was performed on both datasets. In agreement with [17], random forest was used as the prediction method in the wrapper-based MOEFS because it is considered a state-of-the-art benchmark algorithm in the spam filtering literature [44, 45, 47]. For the ENORA search algorithm, the number of individuals in the population was set to 100 and the number of generations to evolve the population was 10. As hinted above, the MOEFS was performed for different values of misclassification cost ratio, $\lambda = 1$, $\lambda = 3$ and $\lambda = 7$. The results of random forest without feature selection and with MOEFS are presented in Fig. 2. It demonstrates that MC was substantially decreased when applying random forest with the set of features reduced using MOEFS. In fact, the differences in MC_1, MC_3 and MC_7 were significant at $p = 0.01$ (using the Student's paired t test), except for MC_1 for the Twitter dataset, suggesting that MOEFS is effective in decreasing MC of the social network spam filter. In addition, at the same time, the number of features was reduced for both datasets. For the Hyves dataset, the use of MOEFS resulted in 854, 725 and 721 features on average for $\lambda = 1$, $\lambda = 3$ and $\lambda = 7$, respectively, while for the Twitter dataset, the average number of features was 741, 738 and 734. In other words, more than half of the attributes were removed as irrelevant and thus the dimensionality of the feature space was substantially reduced. The feature subsets obtained in this stage were further used in training the RDNN with ensemble learning.

The RDNN with ensemble learning was trained with the following setting: number of hidden layers = {1, 2, 3}, number of units in hidden layers = {10, 20, 50}, and learning rate was set to $\eta = 0.1$, size of mini-batches = 100, dropout rate for input layer = 0.2; dropout rate for hidden layers = 0.5, and number of iterations = 1000. The best setting of the RDNN structure (in most experiments, it was 2 hidden layers with 50 and 20 units, respectively) was obtained by using grid search. Furthermore, 10 iterations were used in the learning of AdaBoost M1 ensemble, the size of each bag in bagging was 100, and the size of each subspace was 50% of all attributes in the random subspace algorithm. The learning of bagging and random subspace was also performed in 10 iterations.

To demonstrate the effectiveness of the proposed social network spam filter, we compared its performance with several methods used in previous studies for spam filtering, namely the single RDNN [18], Naïve Bayes [23], SVM [5, 42], AdaBoost M1 with decision stump as base learner [66] and random forest [44, 45, 47]. Note that all the compared methods were trained in the cost-sensitive mode; this is with, respectively, MC_1, MC_3 and MC_7 as objective functions. The settings of these algorithms were as follows: single RDNN (the same setting as for the RDNN with ensemble learning); SVM (sequential minimal optimization algorithm with $C = \{2^0, 2^1, \ldots, 2^6\}$ ($C = 2^2$ worked best) and polynomial kernel function); AdaBoost M1 with

Table 5 Results of the experiments for $\lambda = 7$

Method	MC_7 (%)	Acc (%)	AUC	FNR (%)	FPR (%)	F1-score
Hyves dataset						
Naïve Bayes	12.29 ± 2.99	69.06 ± 2.85	0.927 ± 0.037	65.3 ± 7.2	**4.7 ± 3.8**	0.778 ± 0.019
SVM	10.85 ± 3.65	87.21 ± 2.52	0.868 ± 0.025	16.3 ± 5.3	10.1 ± 4.5	0.888 ± 0.022
AdaBoost M1	10.46 ± 4.08	**88.06 ± 3.86**	**0.939 ± 0.028**	**14.7 ± 4.6**	9.9 ± 4.3	**0.895 ± 0.034**
Random forest	**9.16 ± 3.22**	**89.40 ± 2.26**	**0.959 ± 0.027**	**13.2 ± 2.2**	8.6 ± 3.7	**0.907 ± 0.021**
Single RDNN	**8.86 ± 4.13**	**88.79 ± 3.48**	**0.945 ± 0.028**	15.5 ± 5.2	7.9 ± 4.6	**0.903 ± 0.030**
RDNN with AB	**8.94 ± 3.82**	**90.14 ± 3.31**	**0.941 ± 0.022**	**11.6 ± 3.7**	8.6 ± 4.1	**0.913 ± 0.030**
RDNN with bagging	**8.97 ± 2.76**	86.53 ± 2.32	**0.952 ± 0.024**	**13.3 ± 3.1**	8.4 ± 3.1	**0.908 ± 0.020**
RDNN with RanSub	**8.30 ± 2.61**	84.53 ± 6.71	**0.956 ± 0.022**	28.9 ± 17.9	**5.4 ± 3.5**	0.877 ± 0.042
Twitter dataset						
Naïve Bayes	23.64 ± 0.65	76.72 ± 0.57	0.817 ± 0.013	**29.1 ± 3.7**	22.9 ± 0.6	0.822 ± 0.004
SVM	7.66 ± 0.58	94.18 ± 0.53	0.802 ± 0.012	36.0 ± 2.2	3.6 ± 0.5	0.944 ± 0.005
AdaBoost M1	12.31 ± 0.28	92.69 ± 0.61	0.734 ± 0.025	89.2 ± 13.3	1.3 ± 1.6	0.734 ± 0.025
Random forest	**5.94 ± 0.51**	**96.26 ± 0.53**	**0.904 ± 0.013**	39.8 ± 2.5	1.1 ± 0.6	**0.960 ± 0.005**
Single RDNN	**6.00 ± 0.27**	**96.28 ± 0.18**	**0.899 ± 0.012**	41.0 ± 3.2	1.0 ± 0.3	**0.960 ± 0.002**
RDNN with AB	6.66 ± 0.74	95.36 ± 0.96	**0.898 ± 0.011**	37.7 ± 3.8	2.2 ± 1.2	**0.953 ± 0.007**
RDNN with bagging	**5.77 ± 0.37**	**96.64 ± 0.24**	**0.913 ± 0.011**	42.8 ± 2.6	**0.5 ± 0.1**	**0.963 ± 0.011**
RDNN with RanSub	6.46 ± 0.29	**96.38 ± 0.17**	**0.908 ± 0.011**	50.0 ± 2.7	**0.2 ± 0.2**	**0.959 ± 0.002**

Bold values significantly better performance at $p = 0.05$ level

10 iterations and decision stump as base learner; and 100 random trees were used in random forest. All the experiments were performed in Weka 3.7.13 environment.

A brief description of the comparative methods is given as follows:

The Naïve Bayes classifier utilizes information learnt from training data in order to calculate the probability of spam or legitimate class taking into consideration words found in the message. The SVM learning is based on structural risk minimization. Specifically, SVM finds the optimal separating hyperplane that represents the maximum margin between two classes and the corresponding decision boundaries are defined by the so-called support vectors. As a result, this algorithm can effectively handle high-dimensional data. Finally, random forest consists of multiple tree predictors. Each of them is influenced by the values of an independently sampled random vector. Therefore, all the trees in the forest share the same distribution.

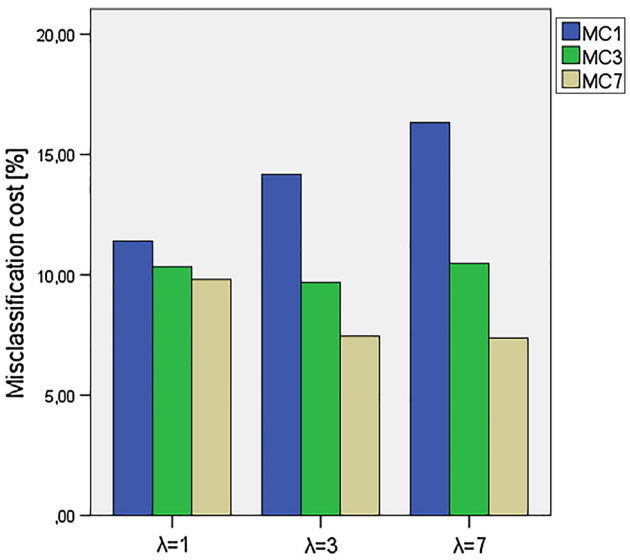

Fig. 3 Effect of MC ratio λ on the misclassification cost of RDNN with bagging

The results of the experiments for MC ratio $\lambda = 1$ are summarized in Table 3. The results show that the RDNN with bagging performed best in terms of MC_1 for both datasets. Besides the RDNN with ensemble learning, the single RDNN and random forest also performed well. Student's paired t tests were performed to compare the results statistically. The results that are statistically similar to the best performer at $p = 0.05$ are in bold in Table 3.

The results in Table 3 show that the RDNN with ensemble learning performed well in terms of both FNR and FPR for both datasets. In other words, the performance was well balanced for both spam and legitimate classes. This was also confirmed with the high values of AUC. Moreover, the RDNN with bagging, together with random forest and single RDNN, respectively, performed best in terms of F1-score, indicating a balanced performance in both the precision and the recall for the Hyves and Twitter datasets. Notably, we obtained better results than those reported in the original study using the Hyves dataset

Table 6 Comparison of MOEFS with other feature selection methods

Method	$\lambda = 1$		$\lambda = 3$		$\lambda = 7$		Features
	MC_1 (%)	Acc (%)	MC_3 (%)	Acc (%)	MC_7 (%)	Acc (%)	
Hyves dataset							
CFS + PSO	9.10 ± 2.75	90.26 ± 3.08	11.30 ± 2.81	**90.01 ± 2.48**	12.46 ± 0.11	56.88 ± 0.78	58
GR	**7.78 ± 2.30**	**92.33 ± 2.78**	**9.17 ± 3.18**	**90.38 ± 3.47**	**9.28 ± 3.12**	81.00 ± 4.70	1221
IG	**7.70 ± 2.74**	**92.08 ± 2.80**	**9.42 ± 3.14**	**90.25 ± 3.09**	**8.78 ± 2.95**	**82.70 ± 4.22**	1221
Relief	**7.50 ± 2.66**	**92.21 ± 2.79**	**9.45 ± 3.44**	**89.89 ± 2.99**	**8.99 ± 3.22**	81.48 ± 5.80	1959
MOEFS	**7.39 ± 2.26**	**92.33 ± 2.38**	**8.83 ± 2.98**	**90.02 ± 3.25**	**8.30 ± 2.61**	**84.53 ± 6.71**	767
Twitter dataset							
CFS + PSO	20.60 ± 1.08	85.89 ± 3.69	10.95 ± 0.48	**95.87 ± 0.47**	6.13 ± 0.29	**96.37 ± 0.25**	91
GR	17.09 ± 0.89	85.54 ± 1.20	10.94 ± 0.77	94.54 ± 0.79	**6.04 ± 0.31**	**96.31 ± 0.28**	1278
IG	16.36 ± 0.76	85.70 ± 0.96	11.08 ± 0.74	94.13 ± 0.72	6.15 ± 0.27	96.11 ± 0.35	1278
Relief	16.37 ± 1.23	**89.04 ± 0.97**	10.73 ± 0.43	94.04 ± 0.75	6.18 ± 0.42	96.03 ± 0.47	1727
MOEFS	**15.42 ± 1.02**	**90.02 ± 0.50**	**10.10 ± 0.43**	**95.60 ± 0.41**	**5.77 ± 0.37**	**96.64 ± 0.24**	738

Bold values significantly better performance at $p = 0.05$ level

(AUC = 0.801) [30]. However, this original study was based on a simple spam score combining messages' own and neighbour characteristics. More precisely, the best performance in terms of accuracy was achieved by using the RDNN with bagging. As presented in Table 3, this can be mainly attributed to the low values of both FNR and FPR on both datasets. In other words, this method performs particularly well in predicting both the spam and legitimate classes. For the Hyves dataset, the RDNN with random subspace performed even better in terms of FNR, but it failed to classify the legitimate messages compared with the remaining ensemble methods. Finally, the RDNN with AB performed reasonably well with respect to both classes, resulting in high accuracy and AUC. Regarding the other comparative methods, they were significantly outperformed by the best spam filter in terms of MC_1, except random forest and single RDNN. For the Twitter dataset, SVM performed also reasonably well in terms of MC_1 and FNR. This agrees with the original study [47]. We must admit that even higher accuracy was achieved in [47] which can be explained by the fact that Twitter removed the most obvious spam messages in the meantime. Moreover, the compared method in [47] performed well on the majority (legitimate) class only. Therefore, comparable accuracy can be easily obtained by giving priority to FPR in our method; this is when increasing MC ratio λ.

Table 4 presents the results of the experiments for MC ratio $\lambda = 3$. As expected, the performance of almost all methods improved in terms of FPR compared with the results for $\lambda = 1$. However, this was achieved at the cost of higher FNR, resulting in lower accuracy for the Hyves dataset. In this case, the RDNN with AB and RDNN with bagging performed reasonably well in terms of both measures, FNR and FPR. Notably, the RDNN with AdaBoost performed best for MC_3. However, only Naïve Bayes was significantly outperformed for this evaluation measure (at $p = 0.05$). Further improvement in FPR can be seen in the results for $\lambda = 7$ (Table 5). Again, several methods performed relatively weakly in terms of FNR. In case of the RDNN with bagging and random subspace, this resulted in a low overall accuracy. Notably, the accuracy of those methods was significantly lower than that of the RDNN with AdaBoost. In contrast, the RDNN with random subspace outperformed the other methods in terms of MC_7. However, this was achieved at the cost of relatively poor performance on the spam class. Regarding accuracy, similarly as for $\lambda = 1$, Naïve Bayes, SVM and AdaBoost M1 methods were significantly outperformed by the RDNNs with ensemble learning.

By contrast to the Hyves dataset, higher accuracy and F1-score were achieved for the Twitter dataset due to the high imbalance of classes in favour of legitimate messages. For the same reason, FNR dropped substantially, as compared with the baseline for $\lambda = 1$.

Now, the question is how to set the parameter λ value in the classifier training step. To demonstrate the effect of the λ value on different MC measures of the classifier, we opted for the RDNN trained with bagging as this model performed reasonably well in terms of all the used classification measures. Figure 3 shows the average MC obtained across the two datasets. Naturally, the choice of the λ value would largely depend on the actual MC value in the social network. It can be seen from Fig. 3 that if MC_1 is the preferred criterion, $\lambda = 1$ would be the best choice, while for MC_3 and MC_7, the classifier trained with $\lambda = 3$ performed best. This finding suggests that in real-world scenarios (with FPR associated with significantly higher

Table 7 Comparison of MOEFS + RDNN with state-of-the-art social network spam filtering methods

Method	References	MC_1 (%)	MC_3 (%)	MC_7 (%)	Acc (%)	AUC
Hyves dataset						
Naïve Bayes	[23]	14.27 ± 3.92	**10.56 ± 3.58**	**8.71 ± 4.33**	86.73 ± 3.54	**0.943 ± 0.030**
SVM	[20]	**9.35 ± 3.42**	**10.36 ± 3.39**	**10.86 ± 3.65**	**90.37 ± 3.34**	0.906 ± 0.034
Decorate	[22]	**9.70 ± 2.51**	**11.17 ± 3.43**	11.91 ± 3.97	**89.89 ± 2.73**	**0.944 ± 0.014**
ADTree	[34]	11.18 ± 3.19	13.52 ± 3.68	14.70 ± 4.14	88.19 ± 3.25	0.924 ± 0.026
C4.5	[38]	11.59 ± 3.73	13.29 ± 4.91	14.15 ± 5.61	87.95 ± 4.00	0.891 ± 0.038
Random forest (RF)	[45]	**7.75 ± 3.44**	**9.23 ± 4.12**	**9.97 ± 4.52**	**91.84 ± 3.61**	**0.955 ± 0.029**
IWNN + RDNN	[53]	**9.04 ± 3.01**	**9.34 ± 3.49**	**9.49 ± 3.86**	**90.86 ± 3.10**	**0.954 ± 0.023**
IG + RF	[16]	**8.04 ± 3.54**	**9.37 ± 4.18**	**10.04 ± 4.55**	**91.60 ± 3.71**	**0.959 ± 0.026**
Relief + LogitBoost	[28]	**10.11 ± 2.98**	12.34 ± 3.47	13.46 ± 3.96	**89.28 ± 3.03**	0.919 ± 0.026
χ^2 + SVM	[42]	12.23 ± 2.88	12.13 ± 2.90	12.07 ± 3.40	87.82 ± 2.73	0.878 ± 0.029
χ^2 + RF	[47]	**7.94 ± 3.44**	**9.21 ± 4.20**	**9.85 ± 4.63**	**91.72 ± 3.65**	**0.956 ± 0.029**
PSO + C4.5	[52]	**10.20 ± 2.95**	13.89 ± 3.96	15.73 ± 4.50	88.79 ± 3.22	0.900 ± 0.030
MOEFS + RDNN with AB, $\lambda = 3$	This study	**10.39 ± 3.42**	**8.83 ± 2.98**	**8.05 ± 3.10**	**90.02 ± 3.25**	**0.941 ± 0.026**
Twitter dataset						
Naïve Bayes	[23]	23.91 ± 1.12	23.52 ± 0:54	23.33 ± 0.39	76.76 ± 0.41	0.818 ± 0.010
SVM	[20]	**16.54 ± 0.90**	14.34 ± 0.59	13.23 ± 0.61	87.27 ± 0.66	0.835 ± 0.009
Decorate	[22]	31.31 ± 1.15	25.56 ± 0.70	22.69 ± 0.56	78.62 ± 0.54	0.706 ± 0.012
ADTree	[34]	20.88 ± 1.25	19.52 ± 2.82	18.83 ± 4.10	81.48 ± 4.70	0.842 ± 0.015
C4.5	[38]	18.64 ± 1.02	14.14 ± 1.10	11.89 ± 1.39	89.14 ± 1.55	0.854 ± 0.015
Random forest (RF)	[45]	**17.66 ± 1.33**	14.14 ± 0.85	12.36 ± 0.82	88.43 ± 0.87	0.899 ± 0.008
IWNN + RDNN	[53]	19.56 ± 1.04	12.16 ± 0.61	8.47 ± 0.43	93.22 ± 0.37	0.835 ± 0.007
IG + RF	[16]	**17.30 ± 1.26**	14.78 ± 1.04	13.53 ± 1.10	87.04 ± 1.16	0.902 ± 0.008
Relief + LogitBoost	[28]	20.31 ± 1.01	19.28 ± 1.40	18.77 ± 1.81	81.47 ± 2.01	0.844 ± 0.012
χ^2 + SVM	[42]	18.75 ± 1.31	17.96 ± 1.41	17.57 ± 2.14	82.61 ± 2.51	0.813 ± 0.013
χ^2 + RF	[47]	**16.86 ± 1.07**	14.04 ± 1.08	12.64 ± 1.28	88.00 ± 1.40	**0.903 ± 0.010**
PSO + C4.5	[52]	22.29 ± 1.19	21.25 ± 0.77	20.74 ± 0.69	79.50 ± 0.71	0.797 ± 0.016
MOEFS + RDNN with bagging, $\lambda = 3$	This study	17.94 ± 0.76	**10.10 ± 0.43**	**6.18 ± 0.38**	**95.60 ± 0.41**	**0.914 ± 0.011**

Bold values significantly better performance at $p = 0.05$ level

MC than FNR), $\lambda = 3$ would be a reasonable choice irrespective of the actual values of MC.

To show the effectiveness of the proposed MOEFS, we further performed a comparative analysis with several feature selection methods considered in previous social network spam filtering studies due to their capacity to decrease complexity and data dimensionality. Thus, the discriminatory power of the feature space can be increased and the prediction performance of classifiers can be improved.

Specifically, four methods were chosen for comparative purposes: (1) correlation-based filter (CBF) with particle swarm optimization (PSO) as a search method [53], (2) gain ratio (GR) [67], (3) information gain (IG) [16, 24] and (4) relief [28]. Those represent computationally efficient filter methods used in earlier social network spam detection studies. The CBF method used PSO with 20 particles in the swarm to find features with high predictive ability and low redundancy. The ranker search method was employed for the remaining filters. To avoid feature selection bias, we used the feature selection methods separately for all the 10 training datasets. RDNNs with AdaBoost, bagging and random subspace were used as the classification methods for all the compared feature selection methods. The best performance achieved is reported in Table 6. Notably, the MOEFS method performed best in terms of MC. Table 6 also shows that for the Hyves dataset CFS + PSO was significantly outperformed by the proposed MOEFS method, whereas the IG method performed statistically similar. For the Twitter dataset, the compared methods were outperformed in terms of all the measures except accuracy for CFS + PSO with $\lambda = 3$. In summary, our method performed best in misclassification cost in particular, irrespective of the value of MC ratio. In addition, the method was effective regarding the number of selected features.

In the further run of experiments, the results of the proposed model were compared to those obtained by other

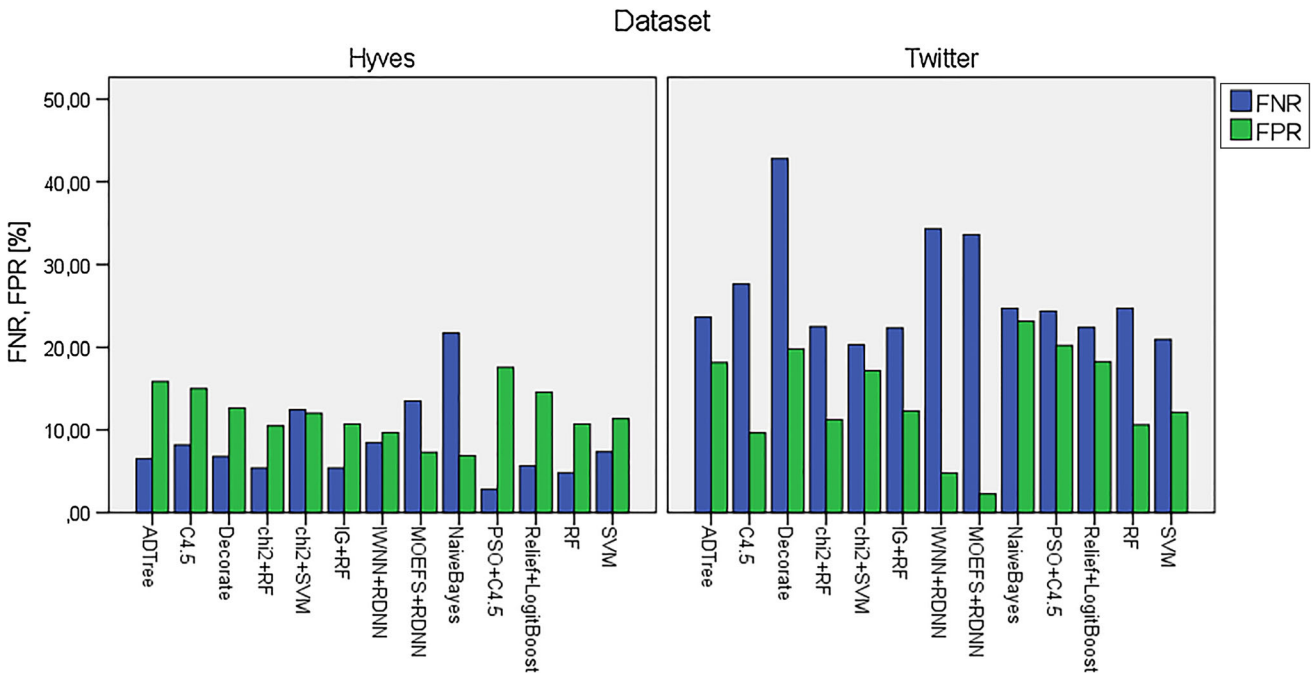

Fig. 4 FNR and FPR for the compared methods

Table 8 Testing time for compared social network spam filtering methods

Method	References	Hyves dataset Testing time [s]	Twitter dataset Testing time [s]
Naïve Bayes	[23]	0.015 ± 0.007	1.898 ± 0.086
SVM	[20]	0.002 ± 0.004	0.006 ± 0.008
Decorate	[22]	0.003 ± 0.006	0.008 ± 0.008
AD Tree	[34]	0.002 ± 0.005	0.005 ± 0.007
C4.5	[38]	0.000 ± 0.000	0.013 ± 0.006
Random forest (RF)	[45]	0.016 ± 0.007	1.483 ± 0.050
IWNN + RDNN	[53]	0.035 ± 0.014	0.059 ± 0.009
IG + RF	[16]	0.013 ± 0.006	1.155 ± 0.037
Relief + LogitBoost	[28]	0.002 ± 0.005	0.008 ± 0.008
χ^2 + SVM	[42]	0.006 ± 0.008	0.006 ± 0.008
χ^2 + RF	[47]	0.016 ± 0.000	1.184 ± 0.063
PSO + DT	[52]	0.002 ± 0.005	0.002 ± 0.005
MOEFS + RDNN with AB/bagging, $\lambda = 3$	This study	0.123 ± 0.022	5.019 ± 0.021

state-of-the-art social network spam detection models. Specifically, we selected both the models without a feature selection component, such as Naïve Bayes [23], SVM [20], Decorate [22], ADTree [34], C4.5 [38] and random forest [45], as wells as models with feature selection, namely IWNN + RDNN [53], IG + RF [16], Relief + LogitBoost [28], χ^2 + SVM [42], χ^2 + RF [47] and PSO + C4.5 [52].

In agreement with [20], SVM was trained with RBF kernel through the libSVM implementation that uses grid search policy to find the best settings of complexity parameter C and gamma of the RBF kernel. Decision stump was used as base learner in the Decorate spam filter. The C4.5 spam filter was trained using the J48 implementation with the confidence factor of 0.25. To train random forest, 100 random trees were used. Following the setting of the RDNN spam filter in [53], a mini-batch gradient descent algorithm was employed with the mini-batch size of 100 and 1000 iterations. The optimum structure of the RDNN was found using the grid search procedure testing different numbers of hidden layers (1, 2 and 3) and neurons in the hidden layers (10, 20, …, 200). Dropout rates were set to 0.2 and 0.5 for the input and hidden layers, respectively. To train the LogitBoost spam

filter, the C4.5 algorithms were employed as base learners with the desired number of fifteen classifiers in the Decorate ensemble.

Table 7 shows that our method performed best in terms of two MC measures: MC_3 and MC_7. It was also competitive for the remaining evaluation measures. This can be attributed to its excellent performance on the legitimate class (FPR) and reasonably good performance on the spam class (FNR), see Fig. 4. As before, Student's paired t test at $p = 0.05$ was used to test the differences in prediction performance. Interestingly, feature selection was not effective for SVM (χ^2-statistic resulted in 561 and 1277 features for the Hyves and Twitter dataset, respectively). Moreover, SVM, together with IWNN + RDNN (using 52 features) and random forest, performed statistically similar as the proposed model for the Hyves dataset. SVM and random forest methods also performed well for the Twitter dataset which can be attributed to their effectiveness in handling sparse and high-dimensional data [53]. Social network spam datasets are characterized as highly sparse due to the short length of the text in the messages. Note that for the data used in this study, the average legitimate message had 33.15 and 34.78 tokens, while the average spam message had 34.70 and 44.59 tokens for the Hyves and Twitter dataset, respectively. In contrast, the models using the C4.5 algorithm performed relatively poorly, which can be explained by the poor capacity of the algorithm to deal with high-dimensional datasets [53]. For the methods from [34, 52], another possible reason which may explain their relatively poor performance is that they were designed to detect spam profiles, rather than messages.

Finally, the time performance of the compared method was examined. In agreement with [47], we did not include the time needed for training the model as this can be done out of band. As a result, we present only the detection time on testing data in Table 8, which informs about the real-time capability of the proposed model. The results show that although the proposed model was the least time efficient of the compared models, it is still acceptable for real-time spam detection requirement (testing time in a few seconds [47]). The difference in testing times in Table 8 is given by the different sizes of samples in the datasets, with the average testing time (throughput) of approximately 1000 messages per second (667 messages/s and 1229 messages/s on the Hyves and Twitter dataset, respectively). This is considered sufficient for an online system [68]. Note that the testing was performed on an Intel Core CPU with six cores (i5-8400 2.80 GHz) and 16 GB RAM and, therefore, substantial reduction in testing time can further be achieved when implementing the system on a server machine or GPU.

6 Conclusion

In this study, we demonstrated that the combination of MOEFS with cost-based RDNN ensemble learning algorithms is more accurate than the state-of-the-art social network spam filtering methods in terms of MC. The results show that the selection of the ensemble algorithm largely depends on the value of MC ratio. Notably, our model performed best for real-world scenarios, where FP rate is associated with significantly higher MC cost than FN rate. In addition, our model performed well on both classes, spam and legitimate, which can be attributed to the capacity of the ensemble learning algorithms in reducing the risk of overfitting. However, RDNN trained with random subspace cannot be recommended for higher MC ratios due to their relatively poor performance on spam class, suggesting that reducing the number of features with random subspace does not seem to be beneficial in case of RDNNs. For the Hyves dataset, Bagging and AdaBoost performed similarly for various values of MC ratios. This is interesting because bagging usually performs best with complex base learners, whereas weak base learners are preferred in training AB. This suggests that it is important to adjust the complexity of RDNN to the chosen ensemble learning algorithm. Bagging also performed best for the Twitter dataset, supporting the necessity of using complex base learners. We also showed that MOEFS can be effective to further improve the prediction performance of the spam detection model, which can be attributed to the proposed MOEFS modification that aims to reduce the MC of the employed classifier.

To summarize the results, the integration of the modified MOEFS and cost-sensitive RDNN ensemble learning seems to be an effective method for social network spam filtering. Although the improvement in MC might not seem substantial for lower MC ratios (about one per cent over random forest), it was considerable for higher cost related to FP classification. This is encouraging because higher MC ratios are the most realistic ones.

Finally, several limitations of this study need to be mentioned. Here, we used the content of the messages, together with the tweet- and profile-based information. However, the content of the neighbouring messages could be utilized in future studies. This would require a larger dataset to be collected. Another limitation might be the focus on spam messages, rather than spammers. However, note that this model could also be used to predict spammers in addition to spam messages. This could be another promising direction for future work because recent studies showed that such a combination might significantly improve the accuracy of spam and spammer filters [69]. It would also be interesting to investigate the effect of

different content-based (e.g. sentiment, context [70]) or user behaviour features (e.g. geographic-distance [71]), and therefore, further investigation and experimentation is highly recommended. We also did not investigate the potential impact of metadata (e.g. font type, upper/lower case, timestamps) on prediction accuracy. Finally, the study did not evaluate the use of the system in a real environment. However, we believe that the detection accuracy of the system would not deteriorate up to 9 months after its implementation, as indicated in earlier studies [68].

The results obtained here suggest that RDNNs with ensemble learning might have great potential also in other text categorization tasks, such as web-page classification, e-mail spam filtering, sentiment classification and so forth. Moreover, it would be beneficial to investigate whether the proposed model shows similar performance for spam datasets from different countries in different languages and from different social networking platforms and whether the localization of the model is required.

Acknowledgements This article was supported by the scientific research project of the Czech Sciences Foundation Grant No: 16-19590S.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Cormack GV (2006) Email spam filtering: a systematic review. Found Trends Inf Retr 1(4):335–455. https://doi.org/10.1561/1500000006
2. Nexgate (2013) State of social media spam. http://nexgate.com/wp-content/uploads/2013/09/Nexgate-2013-State-of-Social-Media-Spam-Research-Report.pdf. Accessed 20 Apr 2019
3. Statista (2018) Twitter: number of monthly active users 2010–2018. https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/. Accessed 20 Apr 2019
4. Prieto VM, Alvarez M, Cacheda F (2013) Detecting linkedin spammers and its spam nets. Int J Adv Comput Sci Appl (IJACSA) 4(9):189–199
5. Shen H, Ma F, Zhang X, Zong L, Liu X, Liang W (2017) Discovering social spammers from multiple views. Neurocomputing 225:49–57. https://doi.org/10.1016/j.neucom.2016.11.013
6. Adewole KS, Anuar NB, Kamsin A, Varathan KD, Razak SA (2017) Malicious accounts: dark of the social networks. J Netw Comput Appl 79:41–67. https://doi.org/10.1016/j.jnca.2016.11.030
7. Soliman A, Girdzijauskas S (2016) Adaptive graph-based algorithms for spam detection in social networks. KTH Royal Institute of Technology, diva2:998690
8. Dutta S, Ghatak S, Dey R, Das AK, Ghosh S (2018) Attribute selection for improving spam classification in online social networks: a rough set theory-based approach. Soc Netw Anal Min 8(7):1–16. https://doi.org/10.1007/s13278-017-0484-8
9. Barushka A, Hajek P (2016) Spam filtering using regularized neural networks with rectified linear units. In: Adorni G, Cagnoni S, Gori M, Maratea M (eds) Conference of the Italian Association for artificial intelligence. Lecture notes in computer science, vol 10037. Springer, Cham, pp 65–75. https://doi.org/10.1007/978-3-319-49130-1_6
10. Bhowmick A, Hazarika SM (2018) E-mail spam filtering: a review of techniques and trends. In: Kalam A, Das S, Sharma K (eds) Advances in electronics, communication and computing. Lecture notes in electrical engineering, vol 443. Springer, Singapore, pp 583–590. https://doi.org/10.1007/978-981-10-4765-7_61
11. Almeida TA, Almeida J, Yamakami A (2011) Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers. J Internet Serv Appl 1(3):183–200. https://doi.org/10.1007/s13174-010-0014-7
12. Choudhary N, Jain AK (2017) Towards filtering of SMS spam messages using machine learning based technique. In: Singh D, Raman B, Luhach A, Lingras P (eds) Advanced informatics for computing research. Communications in computer and information science, vol 712. Springer, Singapore, pp 18–30. https://doi.org/10.1007/978-981-10-5780-9_2
13. Kaur P, Singhal A, Kaur J (2016) Spam detection on Twitter: A survey. In: 2016 3rd international conference on computing for sustainable global development (INDIACom). IEEE, New Delhi, pp 2570–2573
14. Kaur R, Singh S, Kumar H (2018) Rise of spam and compromised accounts in online social networks: a state-of-the-art review of different combating approaches. J Netw Comput Appl 112:53–88. https://doi.org/10.1016/j.jnca.2018.03.015
15. Sanz JA, Bernardo D, Herrera F, Bustince H, Hagras H (2015) A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data. IEEE Trans Fuzzy Syst 23(4):973–990. https://doi.org/10.1109/TFUZZ.2014.2336263
16. Al-Janabi M, Quincey ED, Andras P (2017) Using supervised machine learning algorithms to detect suspicious URLs in online social networks. In: Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017, ACM, pp 1104–1111. https://doi.org/10.1145/3110025.3116201
17. Jiménez F, Sánchez G, García JM, Sciavicco G, Miralles L (2017) Multi-objective evolutionary feature selection for online sales forecasting. Neurocomputing 234:75–92. https://doi.org/10.1016/j.neucom.2016.12.045
18. Barushka A, Hajek P (2018) Spam filtering in social networks using regularized deep neural networks with ensemble learning. In: Iliadis L, Maglogiannis I, Plagianakos V (eds) Artificial intelligence applications and innovations. AIAI 2018. IFIP advances in information and communication technology, vol 519. Springer, Cham, pp 38–49. https://doi.org/10.1007/978-3-319-92007-8_4
19. Statista (2018) Number of facebook users worldwide 2008–2018. https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. Accessed 20 Apr 2019
20. Zheng X, Zeng Z, Chen Z, Yu Y, Rong C (2015) Detecting spammers on social networks. Neurocomputing 159:27–34. https://doi.org/10.1016/j.neucom.2015.02.047
21. Stringhini G, Kruegel C, Vigna G (2010) Detecting spammers on social networks. In: Proceedings of the 26th annual computer security applications conference. ACM, pp 1–9
22. Lee K, Caverlee J, Webb S (2010) Uncovering social spammers: social honeypots + machine learning. In: Proceedings of the 33rd international ACM SIGIR conference on research and development in information retrieval. ACM, pp 435–442

23. Wang AH (2010) Don't follow me: spam detection in Twitter. In: Proceedings of the 2010 international conference on security and cryptography (SECRYPT). IEEE, pp 1–10
24. Benevenuto F, Magno G, Rodrigues T, Almeida V (2010) Detecting spammers on twitter. In: 6th collaboration, electronic messaging, anti-abuse and spam conference (CEAS), pp 1–12
25. Lee K, Eoff BD, Caverlee J (2011) Seven months with the devils: a long-term study of content polluters on Twitter. In: Proceedings of the 5th international AAAI conference on weblogs and social media, pp 185–192
26. Jin X, Lin C, Luo J, Han J (2011) A data mining-based spam detection system for social media networks. Proc VLDB Endow 4(12):1458–81461
27. Thomas K, Grier C, Song D, Paxson V (2011) Suspended accounts in retrospect: an analysis of twitter spam. In: Proceedings of the 2011 ACM SIGCOMM conference on internet measurement conference. ACM, pp 243–258
28. Song J, Lee S, Kim J (2011) Spam filtering in twitter using sender-receiver relationship. In: International workshop on recent advances in intrusion detection. Springer, Berlin, pp 301–317
29. Chu Z, Widjaja I, Wang H (2012) Detecting social spam campaigns on twitter. In: International conference on applied cryptography and network security. Springer, Berlin, pp 455–472. https://doi.org/10.1007/978-3-642-31284-7_27
30. Bosma M, Meij E, Weerkamp W (2012) A framework for unsupervised spam detection in social networking sites. In: Baeza-Yates R et al (eds) European conference on information retrieval. Springer, Berlin, pp 364–375. https://doi.org/10.1007/978-3-642-28997-2_31
31. Yang C, Harkreader R, Gu G (2013) Empirical evaluation and new design for fighting evolving twitter spammers. IEEE Trans Inf Forensics Secur 8(8):1280–1293. https://doi.org/10.1109/TIFS.2013.2267732
32. Martinez-Romo J, Araujo L (2013) Detecting malicious tweets in trending topics using a statistical analysis of language. Expert Syst Appl 40(8):2992–3000. https://doi.org/10.1016/j.eswa.2012.12.015
33. Lee S, Kim J (2013) Warningbird: a near real-time detection system for suspicious urls in twitter stream. IEEE Trans Dependable Secure Comput 10(3):183–195. https://doi.org/10.1109/TDSC.2013.3
34. Bhat SY, Abulaish M (2013) Community-based features for identifying spammers in online social networks. In: 2013 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE, pp 100–107
35. Ahmed F, Abulaish M (2013) A generic statistical approach for spam detection in online social networks. Comput Commun 36(10–11):1120–1129. https://doi.org/10.1016/j.comcom.2013.04.004
36. Miller Z, Dickinson B, Deitrick W, Hu W, Wang AH (2014) Twitter spammer detection using data stream clustering. Inf Sci 260:64–73. https://doi.org/10.1016/j.ins.2013.11.016
37. Cao C, Caverlee J (2015) Detecting spam urls in social media via behavioral analysis. In: European conference on information retrieval. Springer, Cham, pp 703–714. https://doi.org/10.1007/978-3-319-16354-3_77
38. Antonakaki D, Polakis I, Athanasopoulos E, Ioannidis S, Fragopoulou P (2016) Exploiting abused trending topics to identify spam campaigns in Twitter. Soc Netw Anal Min 6(1):48. https://doi.org/10.1007/s13278-016-0354-9
39. Liu C, Wang G (2016) Analysis and detection of spam accounts in social networks. In: 2016 2nd IEEE international conference on computer and communications (ICCC). IEEE, pp 2526–2530. https://doi.org/10.1109/compcomm.2016.7925154
40. Wu F, Shu J, Huang Y, Yuan Z (2016) Co-detecting social spammers and spam messages in microblogging via exploiting social contexts. Neurocomputing 201:51–65. https://doi.org/10.1016/j.neucom.2016.03.036
41. Zheng X, Zhang X, Yu Y, Kechadi T, Rong C (2016) ELM-based spammer detection in social networks. J Supercomput 72(8):2991–3005. https://doi.org/10.1007/s11227-015-1437-5
42. Song L, Lau RYK, Kwok RCW, Mirkovski K, Dou W (2017) Who are the spoilers in social media marketing? Incremental learning of latent semantics for social spam detection. Electron Commer Res 17(1):51–81. https://doi.org/10.1007/s10660-016-9244-5
43. Chen C, Wang Y, Zhang J, Xiang Y, Zhou W, Min G (2017) Statistical features-based real-time detection of drifted twitter spam. IEEE Trans Inf Forensics Secur 12(4):914–925. https://doi.org/10.1109/TIFS.2016.2621888
44. Adewole KS, Anuar NB, Kamsin A, Sangaiah AK (2019) SMSAD: a framework for spam message and spam account detection. Multimed Tools Appl 78(4):3925–3960. https://doi.org/10.1007/s11042-017-5018-x
45. Watcharenwong N, Saikaew K (2017) Spam detection for closed Facebook groups. In: 2017 14th international joint conference on computer science and software engineering (JCSSE). IEEE, pp 1–6. https://doi.org/10.1109/jcsse.2017.8025914
46. Yu D, Chen N, Jiang F, Fu B, Qin A (2017) Constrained NMF-based semi-supervised learning for social media spammer detection. Knowl-Based Syst 125:64–73. https://doi.org/10.1016/j.knosys.2017.03.025
47. Chen W, Yeo CK, Lau CT, Lee BS (2017) A study on real-time low-quality content detection on Twitter from the users' perspective. PLoS ONE 12(8):e0182487. https://doi.org/10.1371/journal.pone.0182487
48. Al-Zoubi AM, Faris H, Hassonah MA (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104. https://doi.org/10.1016/j.knosys.2018.04.025
49. Aswani R, Kar AK, Ilavarasan PV (2017) Detection of spammers in twitter marketing: a hybrid approach using social media analytics and bio inspired computing. Inf Syst Front. https://doi.org/10.1007/s10796-017-9805-8
50. Bindu PV, Mishra R, Thilagam PS (2018) Discovering spammer communities in twitter. J Intell Inf Syst. https://doi.org/10.1007/s10844-017-0494-z
51. Sedhai S, Sun A (2018) Semi-supervised spam detection in Twitter stream. IEEE Trans Comput Soc Syst 5(1):169–175. https://doi.org/10.1109/TCSS.2017.2773581
52. Sohrabi MK, Karimi F (2018) A feature selection approach to detect spam in the Facebook social network. Arab J Sci Eng 43(2):949–958. https://doi.org/10.1007/s13369-017-2855-x
53. Barushka A, Hajek P (2018) Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. Appl Intell 48(10):3538–3556. https://doi.org/10.1007/s10489-018-1161-y
54. Gogoglou A, Theodosiou Z, Kounoudes T, Vakali A, Manolopoulos Y (2016) Early malicious activity discovery in microblogs by social bridges detection. In: 2016 IEEE international symposium on signal processing and information technology (ISSPIT). IEEE, Limassol, pp 132–137. https://doi.org/10.1109/isspit.2016.7886022
55. Hinton G, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580
56. Dhillon IS, Mallela S, Kumar R (2003) A divisive information-theoretic feature clustering algorithm for text classification. J Mach Learn Res 3:1265–1287. https://doi.org/10.1162/153244303322753661

57. Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Comput Electr Eng 40(1):16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024
58. Jiménez F, Marzano E, Sánchez G, Sciavicco G, Vitacolonna N (2015) Attribute selection via multi-objective evolutionary computation applied to multi-skill contact center data classification. In: 2015 IEEE symposium series on computational intelligence. IEEE, pp 488–495. https://doi.org/10.1109/ssci.2015.78
59. Zhang Y, Wang S, Phillips P, Ji G (2014) Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. Knowl-Based Syst 64:22–31. https://doi.org/10.1016/j.knosys.2014.03.015
60. Jia X, Shang L (2014) Three-way decisions versus two-way decisions on filtering spam email. In: Transactions on rough sets XVIII, Springer, Berlin, pp 69–91. https://doi.org/10.1007/978-3-662-44680-5_5
61. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th international conference on machine learning, pp 1–6
62. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: 13th international conference on machine learning, San Francisco, pp 148–156
63. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140. https://doi.org/10.1007/BF00058655
64. Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844. https://doi.org/10.1109/34.709601
65. Bermejo P, Gámez JA, Puerta JM (2011) Improving the performance of Naive Bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. Expert Syst Appl 38(3):2072–2080. https://doi.org/10.1016/j.eswa.2010.07.146
66. Pérez-Díaz N, Ruano-Ordás D, Fdez-Riverola F, Méndez JR (2012) SDAI: an integral evaluation methodology for content-based spam filtering models. Expert Syst Appl 39(16):12487–12500. https://doi.org/10.1016/j.eswa.2012.04.064
67. Cao J, Fu Q, Li Q, Guo D (2017) Discovering hidden suspicious accounts in online social networks. Inf Sci 394:123–140. https://doi.org/10.1016/j.ins.2017.02.030
68. Gao H, Chen Y, Lee K, Palsetia D, Choudhary AN (2012) Towards online spam filtering in social networks. NDSS 12(2012):1–16
69. Masood F, Almogren A, Abbas A, Khattak HA, Din IU, Guizani M, Zuair M (2019) Spammer detection and fake user identification on social networks. IEEE Access 7:68140–68152. https://doi.org/10.1109/ACCESS.2019.2918196
70. Barushka A, Hajek P (2019). Review spam detection using word embeddings and deep neural networks. In: MacIntyre J, Maglogiannis I, Iliadis L, Pimenidis E (eds) Artificial intelligence applications and innovations. AIAI 2019. IFIP Advances in information and communication technology, vol 559. Springer, Cham, pp 340–350. https://doi.org/10.1007/978-3-030-19823-7_28
71. Jang B, Jeong S, Kim CK (2019) Distance-based customer detection in fake follower markets. Inf Syst 81:104–116. https://doi.org/10.1016/j.is.2018.12.001