An Automated Online Spam Detector Based on Deep Cascade Forest

Kangyang Chen[1], Xinyi Zou[1], Xingguo Chen[1,2](✉), and Huihui Wang[3,4]

[1] Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, People's Republic of China
chenxg@njupt.edu.cn

[2] National Engineering Laboratory for Agri-product Quality Traceability, Beijing Technology and Business University, Beijing, People's Republic of China

[3] PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing, People's Republic of China

[4] Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, People's Republic of China

Abstract. With the development of internet communication, spam is quite ubiquitous in our daily life. It not only disturbs users, but also cms. Although there exists many methods of spam detection in both the area of cyber security and natural language processing, their performance is still not capable to satisfy requirements. In this paper, we implemented deep cascade forest for spam detection, a deep model without using back-propagation. With less hyperparameters, the training cost can be easily controlled and declines compared with that in neutral network methods. Furthermore, the proposed deep cascade forest outperforms other machine learning models in the F1 Score of detection. Therefore, considering the lower training cost, it can be considered as a useful online tool for spam detection.

Keywords: Spam detection · Deep forest · Deep learning · Machine learning · Ensemble methods

1 Introduction

In the era of information explosion, communication through digital media becomes prevalent in peoples' daily life. There is a lot of complicated information in our daily life. However, among these messages, there exists a large amount of information that is false, violent, or illegal. These messages not only affect user's product experience, but may also lead to cyber security issues such as financial fraud and personal privacy leaks [20]. According to the report presented by Kaspersky, an independent cybersecurity company, spammers continuously exploit new methods to propogate malicious messages to their "audience", including instant messengers and social networks [13].

Owing to the fact that spam is too multitudinous to be recognized and filtered in advance manually, spam detection is of great significance. There are two major difficulties in spam detection. One is how to convert text information into numerical information, especially those conveys less information such as word abbreviations, expressions, and symbols. The other is how to build an online tool that can recognize spam in time.

Spam detection plays a predominant way in cyber security. Early in 1999, Harris Drucker et al. applied Support Vector Machines (SVM) to email spam detection [7]. In order to encode texts into computer-readable mathematical features, he combined Term Frequency-Inverse Document Frequency (TF-IDF) representation method with binary representation method. According to his research, compared with boosting decision tree, SVM is considered as the most suitable model at that time. In 2011, McCord used a variety of traditional machine learning methods to detect spam on twitter. In his work, several classic classification algorithms such as random forest (RF), naive bayes (NB), support vector machine (SVM), and k nearest neighbors (KNN) are compared. Among them, RF shows the best performance as an ensemble learning method [14]. When combining with multiple base classifiers, ensemble classifier provides a more accurate prediction than single classifier. In 2013, Cailing Dong developed an ensemble learning framework for online web spam detection, and their experimental results reflect the effectiveness of integrated learning [6].

In around 2006, the idea of Deep Learning started to take shape. Hinton believes that neural networks can be used to reduce dimensions of data so as to contribute to feature extraction [8]. Its true in many areas such as speech recognition [5], image recognition [11]. Deep learning is also widely used in natural language processing (NLP) [12,19], Long Short-Term Memory (LSTM) is one of its best-known models. Hochreiter discusses [9] how LSTM is trained to store information over a period of time. Recent years, some researchers began to use deep learning models to detect spam. In 2017, Wu et al. compared multilayer perception with RF, decision tree (DT) and NB [22]. Later, Ren empirically explored a neural network model to learn document-level representation for detecting deceptive opinion spam [15]. They compared gated recurrent neural network (GRNN) with convolutional neural network (CNN) and recurrent neural network (RNN), finding that GRNN outperforms others on datasets which consist of truthful and deceptive reviews in three domains. Furthermore, Gauri Jain et al. firstly used LSTM to categorize SMS spam and Twitter spam [10] in 2019. The results show that LSTM is superior to traditional machine learning methods in SMS and Twitter spam datasets. Due to the fact that some datasets contain not only texts but also images, Yang et al. used CNN for image extraction and LSTM for text extraction respectively [23].

When dealing with text information, a traditional approach is vector space model [17]. It is designed to encode each word respectively. Therefore, it washes away semantic information and generates high dimensional and sparse features that are not suitable for neural networks. Another mainstream approach is semantic-based textual representation, which translates textual information into continuous dense features to learn the distributed representation of words [4].

This method is also called word embedding, which has high autocorrelation and is suitable for use in neural networks.

Our goal is to find promising methods and settings that can recognize spam in social networks. Many developers have developed anti-spam tools for spam detection, but they are not efficient. Most of these spam detection methods are based on traditional machine learning methods. In recent years, with the development of NLP, deep learning methods emerged in spam detection, which have achieved satisfying results in accuracy. Nevertheless, the deeper the network is, the higher training cost and complexity the model will be.

In summary, this article uses the gcForest method, the deep ensemble method proposed by Zhou et al. in 2017 [24], and its structure has been improved to adapt to spam detection problems. It is a highly-ensemble learning model with fewer hyperparameters than deep neural network. Furthermore, its model complexity can be determined in a data-dependent way which makes our model less time-consuming. Compared with previous machine learning methods, our model shows a higher accuracy and efficiency and solves training overhead problems simultaneously.

The rest of the paper is organized as follows: Sect. 2 offers the description of the cascade structure of gcForest approach (Deep Cascade Forest, DCF) that is the core of our method. Section 3 briefly describes the text processing methods. In Sect. 4, we illustrate models elaborately including datasets we use and parameter settings in experiments. After that, we compare the accuracy, F1 Score, training time, etc. of models. Finally, Sect. 5 describes the main conclusion and offers guidelines for future work.

2 Deep Cascade Forest

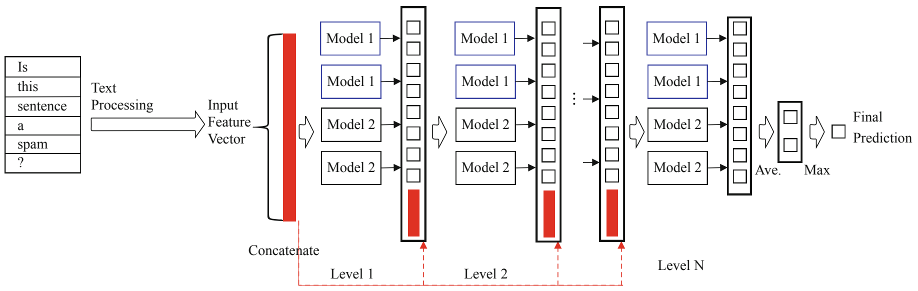


Fig. 1. Example of the cascade structure of gcForest (DCF) for spam detection. Suppose each level of the cascade consists of two kinds of base models. There are two classes to predict (spam or ham), thus, each model will output a two-dimensional class vector, which is then concatenated for re-representation of original input.

In 2017, Zhou et al. proposed an ensemble approach with a deep cascade structure, named gcForest [24]. The basic form of gcForest contains 2 parts:

Multi-**G**rained Scanning and **C**ascade Forest. The former is used for feature preprocessing, and for spam detection tasks, we replace it with text preprocessing method. In order to train the model, we only use the cascade structure and call the model the deep cascade forest (DCF).

As is elaborately illustrated in Fig. 1, first, it is necessary to split the input document into words, and then carried out the text processing procedure that extracting the textual information as a feature vector. Inspired by the well-known recognition that representation learning in deep neural network mostly relies on the layer-by-layer processing of raw features, DCF then feed the feature vector into the layer-by-layer cascade structure. After training various base models on the feature vector, the output predictions will then be concentrated with raw features and fed to the next layer together. In each layer, different types of base models can be selected to encourage the ensemble diversity. Base model can even be an ensemble model, e.g. random forest and this constitutes an "ensemble of ensembles". The output of each base model is a class vector, indicating the probability of predicting a sentence as a class. In the spam detection task, there are two classes (spam and ham), that is, each model outputs a two-dimensional probability class vector. To reduce the risk of overfitting, which is common in deep learning models, class vectors are generated by k-fold cross validation. In detail, each instance will be trained k-1 times to generate k-1 class vectors, and then averaged to get the final class vector. Before generating a new layer, the performance of the entire cascade structure is estimated on validation data, and if the performance does not improve, the training process terminates. So that the number of cascade levels is automatically determined.

DCF can achieve good results in different dimensions of input data. However neural networks are difficult to get good results on high dimensional and sparse text features. Comparing with deep neural networks, DCF has much fewer hyper-parameters and lower training cost, and it opens the door of deep learning based on non-NN (Neural Network) styles, or deep models based on non-differentiable modules.

3 Text Processing

In order to turn text into information that computers can recognize, following different text processing methods are used:

Remove the Stop Words. A stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We can remove them easily, by storing a list of words that considered as stop words.

Build Word Count Vector. To build the word count vector for each sample, we firstly create a dictionary of words and their frequency. Once the dictionary is ready, we can extract word count vectors from training set. A sample corresponds to a word count vector. The dimension of the word count vector is the total number of words in the training set. If the sample contains a word in the training set, the value in the vector is the frequency of the word in the training set. If not, the value is zero. All word count vectors are combined into a word count matrix, rows represent each sample, and columns represent each word.

Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining, to produce a composite weight for each term in each document. [16].

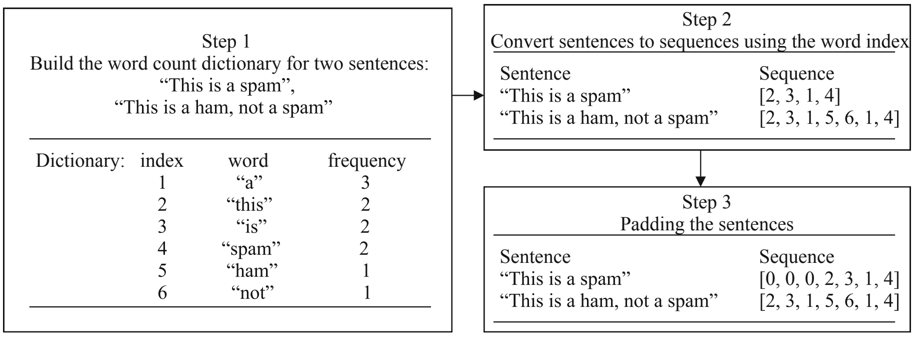
TF determines a terms (a word or a combination) relative frequency within a document. The $TF(w_i)$ is the number of times that word w_i appears in a document.

TF-IDF uses the above TF multiplied by the IDF, the inverse document frequency (IDF) is defined as:

$$IDF(w_i) = \log(\frac{|D|}{DF(w_i)}) \tag{1}$$

Where $|D|$ is the number of documents, and the document frequency $DF(w_i)$ is the number of times that word w_i appears in all documents.

Texts to Sequences. This approach will create a vector for each sample, converting words to their index in the word count dictionary. An example of texts to sequences is shown in Fig. 2.



Step 1
Build the word count dictionary for two sentences:
"This is a spam",
"This is a ham, not a spam"

Dictionary:	index	word	frequency
	1	"a"	3
	2	"this"	2
	3	"is"	2
	4	"spam"	2
	5	"ham"	1
	6	"not"	1

Step 2
Convert sentences to sequences using the word index

Sentence	Sequence
"This is a spam"	[2, 3, 1, 4]
"This is a ham, not a spam"	[2, 3, 1, 5, 6, 1, 4]

Step 3
Padding the sentences

Sentence	Sequence
"This is a spam"	[0, 0, 0, 2, 3, 1, 4]
"This is a ham, not a spam"	[2, 3, 1, 5, 6, 1, 4]

Fig. 2. An example of texts to sequences method

4 Experiments and Results

The experiments use two public data sets to train the model. In order to compare models performance on detecting spam and their training cost, we used F1 Score to evaluate the model and calculated the training and testing time of each model. Our experiments use a PC with Intel Core i5 7260u CPUs (2 cores), and the performance and running efficiency of DCF is good.

4.1 Datasets and Evaluation Measures

The experiments are performed on SMS spam dataset and YouTube comments spam dataset. All the information is available in UCI repository [1,21]. The SMS spam dataset is a public set of SMS labeled messages that are collected for mobile phone spam research. It has one collection composed by 5,574 English, real and non-encoded messages, tagged in legitimate (ham) or spam [2]. The YouTube comments spam dataset was collected using the YouTube Data API v3. The samples were extracted from the comments section of five videos that were among the 10 most viewed on YouTube during the collection period [3]. For these two datasets, each is divided into 70% training sets and 30% test sets. The training set is used to train the model, and the test set is used to assess the effectiveness of models. The overview of these datasets is reported in Table 1.

Table 1. Datasets overview

Dataset	No. of instances	No. of ham	No. of spam
SMS spam	5574	4825	747
YouTube spam	1956	951	1005

The experiments were performed on four processed datasets, shown in Table 2, where Count in the first row denotes the approach of building word count vector, Sequence denotes the approach of texts to sequences.

Table 2. Number of features in four processed datasets

Dataset	TF-IDF	Count	Sequence
SMS spam	8710	8710	200
YouTube spam	4454	4454	500
SMS spam (stop)	9403	9403	200
YouTube spam (stop)	4185	4185	500

In order to assess the effectiveness of proposed methods, this paper uses different evaluation indicators, including accuracy, recall, precision and F1 score, which are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

Where true positive (TP) means the number of spam that are correctly classified, false positive (FP) means the number of legitimate emails (ham) that are misclassified, true negative (TN) means the number of legitimate emails (Ham) that are correctly classified and false negative (FN) is the number of misclassified spam.

4.2 Parameter Settings

The grid search method is used to select hyperparameters of models [18]. Both machine learning methods and deep learning methods are the benchmark, including support vector machine (SVM), k nearest neighbors (KNN), Naive Bayes (NB), decision tree (DT), logistic regression (LR), random forest (RF), adaptive boosting (Adaboost), Bagging, extra-trees classifier (ETC, standing for extremely randomized trees), and long short-term memory (LSTM).

For SMS spam detection, SVM uses a sigmoid kernel with gamma set to 1.0. KNN uses 49 neighbors. NB uses a multinomial kernel with alpha set to 0.2. The minimum of DT samples in each split is 7, and the best *gini* value is used to measure the quality of a split. LR uses the L1 penalty, random forest contains 31 decision trees. The Adaboost classifier contains 62 decision trees. The Bagging classifier contains 9 decision trees. ETC contains 9 decision trees.

For YouTube spam detection, SVM uses a sigmoid kernel with gamma set to 1.0. KNN uses 5 neighbors. NB uses a multinomial kernel with alpha set to 1. The minimum of DT samples in each split is 2, and the best *gini* value is used to measure the quality of a split. LR uses the L2 penalty, and random forest contains 10 decision trees. The Adaboost classifier contains 50 decision trees. The Bagging classifier contains 10 decision trees. ETC contains 10 decision trees.

For SMS spam detection, each layer of DCF contains 1 RF with 31 decision trees, and an NB classifier with a multinomial kernel. For YouTube spam detection, each layer of DCF contains 1 DT, an NB classifier with a multinomial kernel and a LR classifier. There are many hyperparameters used by LSTM. The specific structure and settings refer to Appendix B. It can be seen that the complexity of DCF is much smaller than that of LSTM.

4.3 Results and Analysis

Datasets shown in Table 2 are split into 70% training data and 30% test data. All models use two text processing approaches: building word count vectors and TF-IDF. In addition, given that LSTM is better suited to use semantic-based text processing methods, the texts to sequences approach is used and compared with other approaches. In order to further validate the performance of those models, we compared their training time which is an important factor in building an online detector.

Figures 3 and 4 show the accuracy and F1 score of different models on the SMS dataset and YouTube dataset, respectively. More details of precision, recall, training and testing time are shown in Tables 3 and 4.

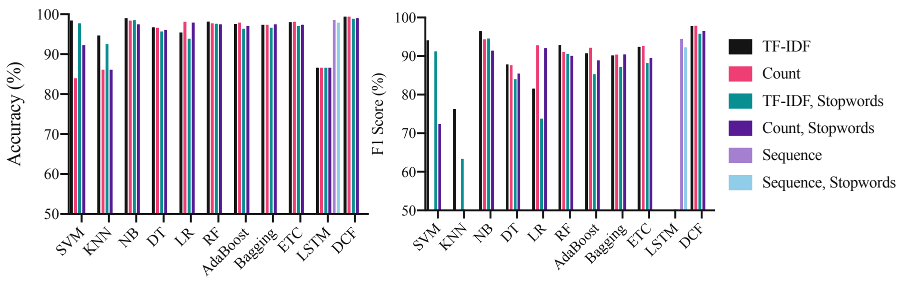


Fig. 3. Accuracy and F1 Score of different models on the test dataset of SMS spam

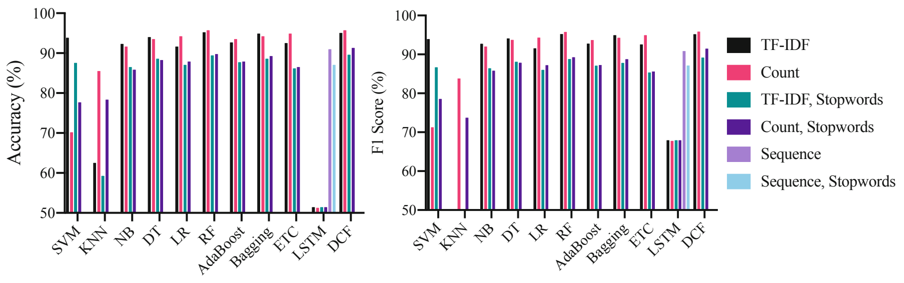


Fig. 4. Accuracy and F1 Score of different models on the test dataset of YouTube spam

From Fig. 3, we can conclude that DCF outperforms others in both the accuracy and F1 Score on the SMS spam dataset. After building word count vectors, DCF achieves the highest accuracy of 99.40% and highest F1 Score of 97.84% which can be found in Table 3. Simultaneously, DCF gets the highest accuracy of 99.40% after using TF-IDF method. The DCF's performance is better on datasets without removing stop words. The training time of DCF is much less than LSTM after building word count vectors and using TF-IDF method. LSTM performs poorly after using the building word vector and using the TF-IDF method, because high dimensional and sparse samples produced by these two methods are not well handled by LSTM. However, after using the texts to sequences method, LSTM performance has been greatly improved but still does not exceed DCF Among many machine learning models, NB not only has a short training time, but also has an accuracy of 99.04%. In addition, since TP is equal to zero, the accuracy, recall, and F1 score of some models are zero in Table 3, which means that all spam is incorrectly classified.

It can be inferred from Fig. 4 and Table 4, as for YouTube spam dataset, DCF also achieved the highest accuracy (95.74%) and F1 Score (95.87%). LSTM performs worst on the dataset after building word count vector and using TF-IDF method, with the lowest accuracy and F1 Score, as well as the longest training time, which denotes that it is less likely to be applied in this field. Regardless of the text processing method used, KNN works very poorly.

Overall, DCF shows the highest accuracy and F1 Score in the spam detection mission. This model not only has quite robust performance to different datasets, but also has lower training cost than the deep neural network due to its automatically determined complexity. The LSTM model is much suitable to use the texts to sequences processing approach, but when facing high-dimension and sparse data (e.g. the YouTube spam dataset), the model not only performs worse, but also has a long training time.

Among other machine learning and ensemble learning methods, NB seems more suitable for spam detection on account of its relatively higher accuracy and lower training cost.

5 Conclusion

In this paper, differing from other researches who use machine learning methods and deep learning methods to carry out spam detection, we attempted deep forest, a non-NN style deep model based on non-differentiable modules. We concluded that deep forest shows the highest accuracy and F1 Score on both SMS spam datasets and YouTube spam datasets. Deep forests are suitable for input data of different kinds of dimensions, however neural networks are difficult to produce good results on high dimensional and sparse samples. Owing to the fact that deep forest has fewer hyperparameters and lower training cost than LSTM, it can be considered as a more suitable model for building an online detector.

In the future, we hope to use new techniques to solve problems with more datasets that include both images and texts. We also aim to explore more text processing methods to further improve performance of classifiers. As an alternative towards deep neural networks, we intend to apply deep forest to other tasks that can not be well handled by deep neural networks. With regard to online tools, we plan to develop web browser and mobile phone plugins to filter spam directly.

Acknowledgement. This work was supported by the National Natural Science Foundation, China (Nos. 61806096, 61872190, 61403208).

A Performance Comparison Between Deep Cascade Forest and Other Classifiers

Tables 3 and 4 shows the precision, recall, precision, accuracy and training/testing time of different models by different kinds of text processing methods on SMS datasets and YouTube datasets, respectively.

Table 3. Performance comparison between deep cascade forest and other classifiers on SMS spam dataset

Classifier	Text processing	Precision (%)	Recall (%)	Accuracy (%)	F1 (%)	Training time (s)	Testing time (s)
SVM	TF-IDF	99.05	89.66	98.45	94.12	0 m 0.4593 s	0 m 0.1385 s
	Count	40.82	34.48	83.97	37.38	0 m 0.4723 s	0 m 0.1495 s
	TF-IDF, Stopwords	98.02	85.34	97.73	91.24	0 m 0.3706 s	0 m 0.4958 s
	Count, Stopwords	71.91	72.84	92.28	72.38	0 m 0.1763 s	0 m 0.1140 s
KNN	TF-IDF	100.00	61.64	94.68	76.27	0 m 0.0041 s	0 m 0.2161 s
	Count	0.00	0.00	86.12	0.00	0 m 0.0010 s	0 m 0.3007 s
	TF-IDF, Stopwords	99.08	46.55	92.52	63.34	0 m 0.0014 s	0 m 0.9363 s
	Count, Stopwords	0.00	0.00	86.12	0.00	0 m 0.0010 s	0 m 0.3494 s
NB	TF-IDF	98.21	94.83	99.04	96.49	0 m 0.0018 s	0 m 0.0003 s
	Count	91.50	97.41	98.39	94.36	0 m 0.0019 s	0 m 0.0004 s
	TF-IDF, Stopwords	95.20	93.97	98.50	94.58	0 m 0.0017 s	0 m 0.0005 s
	Count, Stopwords	87.11	96.12	97.49	91.39	0 m 0.0022 s	0 m 0.0016 s
DT	TF-IDF	90.05	85.78	96.71	87.86	0 m 0.2270 s	0 m 0.0006 s
	Count	88.21	87.07	96.59	87.64	0 m 0.1238 s	0 m 0.0008 s
	TF-IDF, Stopwords	86.70	81.47	95.69	84.00	0 m 0.1950 s	0 m 0.0104 s
	Count, Stopwords	87.39	83.62	96.05	85.46	0 m 0.1386 s	0 m 0.0026 s
LR	TF-IDF	93.33	72.41	95.45	81.55	0 m 0.0107 s	0 m 0.0002 s
	Count	96.73	89.22	98.09	92.83	0 m 0.0153 s	0 m 0.0015 s
	TF-IDF, Stopwords	90.06	62.50	93.84	73.79	0 m 0.0085 s	0 m 0.0005 s
	Count, Stopwords	97.13	87.50	97.91	92.06	0 m 0.0104 s	0 m 0.0022 s
RF	TF-IDF	100.00	86.64	98.15	92.84	0 m 0.9180 s	0 m 0.0150 s
	Count	100.00	83.62	97.73	91.08	0 m 0.9301 s	0 m 0.0290 s
	TF-IDF, Stopwords	100.00	82.76	97.61	90.57	0 m 1.2973 s	0 m 0.0437 s
	Count, Stopwords	100.00	81.90	97.49	90.05	0 m 1.3429 s	0 m 0.0448 s
AdaBoost	TF-IDF	95.26	86.64	97.55	90.74	0 m 2.4091 s	0 m 0.0185 s
	Count	96.24	88.36	97.91	92.13	0 m 2.1926 s	0 m 0.0189 s
	TF-IDF, Stopwords	96.72	76.29	96.35	85.30	0 m 2.5404 s	0 m 0.0370 s
	Count, Stopwords	93.78	84.48	97.07	88.89	0 m 2.4455 s	0 m 0.0234 s
Bagging	TF-IDF	93.52	87.07	97.37	90.18	0 m 1.1530 s	0 m 0.0579 s
	Count	91.59	89.22	97.37	90.39	0 m 0.7393 s	0 m 0.0214 s
	TF-IDF, Stopwords	91.08	83.62	96.59	87.19	0 m 1.0032 s	0 m 0.0284 s
	Count, Stopwords	95.67	85.78	97.49	90.45	0 m 0.8343 s	0 m 0.0408 s
ETC	TF-IDF	99.50	86.21	98.03	92.38	0 m 0.6672 s	0 m 0.0115 s
	Count	99.02	87.07	98.09	92.66	0 m 0.6012 s	0 m 0.0131 s
	TF-IDF, Stopwords	100.00	78.88	97.07	88.19	0 m 0.9153 s	0 m 0.0454 s
	Count, Stopwords	100.00	81.03	97.37	89.52	0 m 0.7918 s	0 m 0.0201 s
LSTM	TF-IDF	0.00	0.00	86.60	0.00	14 m 54.3540 s	0 m 35.1124 s
	Count	0.00	0.00	86.60	0.00	20 m 19.1406 s	0 m 32.3467 s
	TF-IDF, Stopwords	0.00	0.00	86.60	0.00	9 m 49.5561 s	0 m 28.2616 s
	Count, Stopwords	0.00	0.00	86.60	0.00	15 m 53.9231 s	0 m 30.3698 s
	Sequence	98.08	91.07	98.56	94.44	0 m 32.2483 s	0 m 3.2479 s
	Sequence, Stopwords	91.27	93.30	97.91	92.27	0 m 45.1065 s	0 m 1.2113 s
DCF	TF-IDF	98.68	96.98	**99.40**	97.83	1 m 41.7678 s	0 m 9.7851 s
	Count	98.26	97.41	**99.40**	**97.84**	0 m 46.6826 s	0 m 4.6078 s
	TF-IDF, Stopwords	98.63	93.10	98.86	95.79	1 m 54.8653 s	0 m 2.4483 s
	Count, Stopwords	97.79	95.26	99.04	96.51	1m 6.5370 s	0 m 1.3066 s

Table 4. Performance comparison between deep cascade forest and other classifiers on YouTube spam dataset

Classifier	Text processing	Precision (%)	Recall (%)	Accuracy (%)	F1(%)	Training time (s)	Testing time (s)
SVM	TF-IDF	95.86	92.05	93.87	93.92	0 m 0.1041 s	0 m 0.0281 s
	Count	70.68	71.85	70.19	71.26	0 m 0.1014 s	0 m 0.0229 s
	TF-IDF, Stopwords	96.36	78.81	87.56	86.70	0 m 0.0908 s	0 m 0.0288 s
	Count, Stopwords	77.67	79.47	77.68	78.56	0 m 0.0521 s	0 m 0.0117 s
KNN	TF-IDF	98.81	27.48	62.52	43.01	0 m 0.0017 s	0 m 0.0258 s
	Count	98.65	72.85	85.52	83.81	0 m 0.0011 s	0 m 0.0202 s
	TF-IDF, Stopwords	98.46	21.19	59.28	34.88	0 m 0.0008 s	0 m 0.0279 s
	Count, Stopwords	98.34	58.94	78.36	73.71	0 m 0.0007 s	0 m 0.0243 s
NB	TF-IDF	90.79	94.70	92.33	92.71	0 m 0.0018 s	0 m 0.0004 s
	Count	90.42	93.71	91.65	92.03	0 m 0.0009 s	0 m 0.0001 s
	TF-IDF, Stopwords	89.68	83.44	86.54	86.45	0 m 0.0010 s	0 m 0.0001 s
	Count, Stopwords	88.69	83.11	85.86	85.81	0 m 0.0011 s	0 m 0.0013 s
DT	TF-IDF	95.88	92.38	94.04	94.10	0 m 0.0450 s	0 m 0.0003 s
	Count	93.71	93.71	93.53	93.71	0 m 0.0247 s	0 m 0.0004 s
	TF-IDF, Stopwords	95.02	82.12	88.59	88.10	0 m 0.0481 s	0 m 0.0003 s
	Count, Stopwords	93.63	82.78	88.25	87.87	0 m 0.0348 s	0 m 0.0007 s
LR	TF-IDF	95.34	88.08	91.65	91.57	0 m 0.0081 s	0 m 0.0001 s
	Count	95.27	93.38	94.21	94.31	0 m 0.0045s	0 m 0.0001s
	TF-IDF, Stopwords	96.69	77.48	87.05	86.03	0 m 0.0030 s	0 m 0.0001 s
	Count, Stopwords	95.65	80.13	87.90	87.21	0 m 0.0034 s	0 m 0.0001 s
RF	TF-IDF	97.90	92.72	95.23	95.24	0 m 3.5051 s	0 m 0.1002 s
	Count	97.92	93.71	95.74	95.77	0 m 3.3739 s	0 m 0.1228 s
	TF-IDF, Stopwords	97.62	81.46	89.44	88.81	0 m 4.3453 s	0 m 0.0874 s
	Count, Stopwords	97.27	82.45	89.78	89.25	0 m 4.5593 s	0 m 0.1163 s
AdaBoost	TF-IDF	94.20	91.39	92.67	92.77	0 m 0.4922 s	0 m 0.0071 s
	Count	94.30	93.05	93.53	93.67	0 m 0.4322 s	0 m 0.0068 s
	TF-IDF, Stopwords	94.92	80.46	87.73	87.10	0 m 0.4404 s	0 m 0.0071 s
	Count, Stopwords	95.29	80.46	87.90	87.25	0 m 0.4357 s	0 m 0.0071 s
Bagging	TF-IDF	97.22	92.72	94.89	94.92	0 m 0.2403 s	0 m 0.0085 s
	Count	95.89	92.72	94.21	94.28	0 m 0.2488 s	0 m 0.0081 s
	TF-IDF, Stopwords	97.57	79.80	88.59	87.80	0 m 0.2903 s	0 m 0.0083 s
	Count, Stopwords	96.14	82.45	89.27	88.77	0 m 0.2522 s	0 m 0.0082 s
ETC	TF-IDF	94.48	90.73	92.50	92.57	0 m 0.1329 s	0 m 0.0024 s
	Count	96.90	93.05	94.89	94.93	0 m 0.1476 s	0 m 0.0035 s
	TF-IDF, Stopwords	94.02	78.15	86.20	85.35	0 m 0.1577 s	0 m 0.0025 s
	Count, Stopwords	95.14	77.81	86.54	85.61	0 m 0.1473 s	0 m 0.0030 s
LSTM	TF-IDF	51.45	100.00	51.45	67.94	2 m 56.7070 s	0 m 8.4437 s
	Count	51.37	99.67	51.28	67.79	3 m 20.1492 s	0 m 7.8555 s
	TF-IDF, Stopwords	51.45	100.00	51.45	67.94	2 m 7.2586 s	0 m 9.3072 s
	Count, Stopwords	51.45	100.00	51.45	67.94	2 m 7.2586 s	0 m 8.1658 s
	Sequence	94.95	87.09	90.97	90.85	0 m 47.4032 s	0 m 1.1193 s
	Sequence, Stopwords	88.97	85.43	87.05	87.16	1 m 10.3903 s	0 m 1.0290 s
DCF	TF-IDF	95.65	94.70	95.06	95.17	0 m 8.7054 s	0 m 0.2663 s
	Count	95.71	96.03	**95.74**	**95.87**	0 m 8.6446 s	0 m 0.2523 s
	TF-IDF, Stopwords	95.82	83.44	89.61	89.20	0 m 10.4687 s	0 m 0.0996 s
	Count, Stopwords	92.54	90.40	91.31	91.46	0 m 11.8891 s	0 m 0.3085 s

B Parameters of LSTM in our Experiment

The LSTM layer contains 64 units for SMS spam detection and 100 units for YouTube spam detection. On both datasets, the batch size is set to 128 In addition, an embedding layer is used to convert each word in the sequence into a dense vector in advance. The embedding layer follows the LSTM layer, a fully connected layer with 256 units, an activation layer using ReLu function, a dropout layer with a dropout rate of 0.1, a fully connected layer with 1 unit, and an activation layer using sigmoid function.

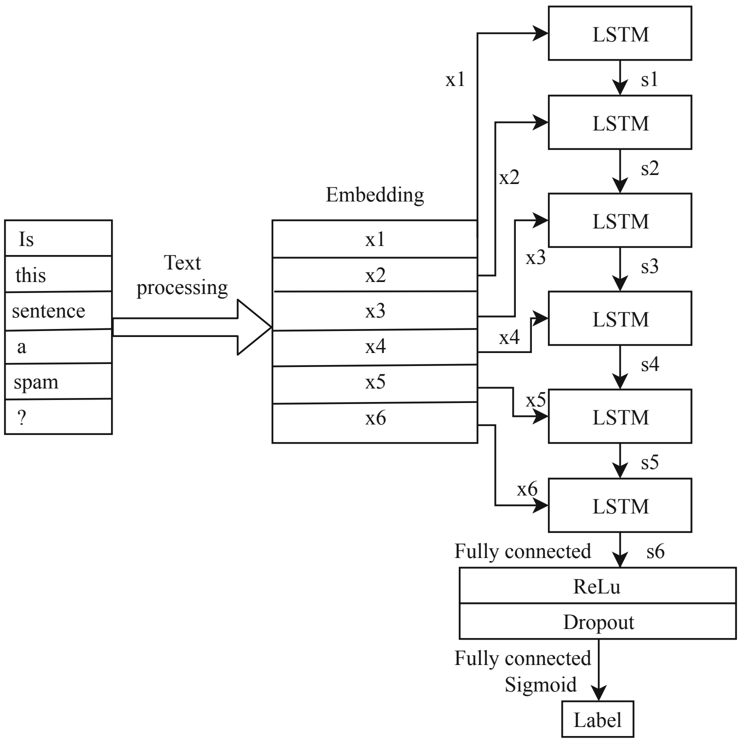


Fig. 5. LSTM classification model

The structure of LSTM is shown in Fig. 5, suppose the texts to sequences processing approach is used, then the output integer sequence is fed to the LSTM model. After embedding each token in the sequence into a 50-dimension word vector x, it is then fed to the LSTM layer with 100 hidden units. Overall, the output 100-dimension vector is processed through a 256 units fully connected layer, a 256 units ReLu layer, a fully connected layer with 1 unit in turn, and finally gets the predicted label through the sigmoid mapping.

References

1. Almeida, T.A.: Sms spam collection data set. https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection. Accessed 25 Apr 2019
2. Almeida, T.A.: SMS spam collection data set from Tiago A. Almeida's homepage. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/. Accessed 25 Apr 2019
3. Almeida, T.A.: Youtube spam collection data set from Tiago A. Almeida's homepage. http://www.dt.fee.unicamp.br/~tiago//youtubespamcollection/. Accessed 25 Apr 2019
4. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
5. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. **20**(1), 30–42 (2012)
6. Dong, C., Zhou, B.: An ensemble learning framework for online web spam detection. In: 2013 12th International Conference on Machine Learning and Applications, vol. 1, pp. 40–45. IEEE (2013)
7. Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Trans. Neural Netw. **10**(5), 1048–1054 (1999)
8. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Jain, G., Sharma, M., Agarwal, B.: Optimizing semantic LSTM for spam detection. Int. J. Inf. Technol. **11**(2), 239–250 (2019)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
13. Vergelis, M., Shcherbakova, T., Sidorina, T.: Spam and phishing in 2018. https://securelist.com/spam-and-phishing-in-2018/89701/. Accessed 25 Apr 2019
14. McCord, M., Chuah, M.: Spam detection on twitter using traditional classifiers. In: Calero, J.M.A., Yang, L.T., Mármol, F.G., García Villalba, L.J., Li, A.X., Wang, Y. (eds.) ATC 2011. LNCS, vol. 6906, pp. 175–186. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23496-5_13
15. Ren, Y., Ji, D.: Neural networks for deceptive opinion spam detection: an empirical study. Inf. Sci. **385**, 213–224 (2017)
16. Salton, G.: Developments in automatic text retrieval. Science **253**(5023), 974–980 (1991)
17. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
18. Sanville, E., Kenny, S.D., Smith, R., Henkelman, G.: Improved grid-based algorithm for bader charge allocation. J. Comput. Chem. **28**(5), 899–908 (2007)
19. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1432 (2015)
20. Tracy, M., Jansen, W., Bisker, S.: Guidelines on electronic mail security. NIST Special Publication (2002)

21. Alberto, T.C., Lochter, J.V., Almeida, T.A.: Youtube spam collection data set. http://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection. Accessed 25 Apr 2019
22. Wu, T., Liu, S., Zhang, J., Xiang, Y.: Twitter spam detection based on deep learning. In: Proceedings of the Australasian Computer Science Week Multiconference, pp. 3:1–3:8. ACM (2017)
23. Yang, H., Liu, Q., Zhou, S., Luo, Y.: A spam filtering method based on multi-modal fusion. Appl. Sci. **9**(6), 1152 (2019)
24. Zhou, Z.H., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3553–3559. AAAI Press (2017)