Pattern Recognition Letters 000 (2018) 1–9

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Opinion fraud detection via neural autoencoder decision forest

Manqing Dong[*], Lina Yao, Xianzhi Wang, Boualem Benatallah, Chaoran Huang, Xiaodong Ning

School of Computer Science and Engineering, University of New South Wales, Sydney 2052, Australia

A R T I C L E I N F O

Article history:
Available online xxx

MSC:
41A05
41A10
65D05
65D17

Keywords:
Autoencoder
Neural decision forest
Opinion fraud detection

A B S T R A C T

Online reviews play an important role in influencing buyers' daily purchase decisions. However, fake and meaningless reviews, which cannot reflect users' genuine purchase experience and opinions, widely exist on the Web and pose great challenges for users to make right choices. Therefore, it is desirable to build a fair model that evaluates the quality of products by distinguishing spamming reviews. We present an end-to-end trainable unified model to leverage the appealing properties from Autoencoder and random forest. A stochastic decision tree model is implemented to guide the global parameter learning process. Extensive experiments were conducted on a large Amazon review dataset. The proposed model consistently outperforms a series of compared methods.

1. Introduction

As a result of the popularity of the e-commerce and experience sharing websites, online reviews influence increasingly on people' purchase decisions. More and more people are referring to others' opinions before buying something. In view of the significance of online reviews to the success of e-commerce, many sellers intentionally post fake reviews, either positive to promote the sales of their own products or negative ones to pull down the sales of other sellers' products. In fact, nowadays, online fraud, including the posting of fake reviews, has become a common phenomenon driven by profits [33]. In the worst cases, some even hire a group of people for fraud.

Under such circumstances, it is paramountly important to evaluate products based on those credible reviews rather than the suspicious (or fake) ones, to help online users make wise purchase decisions. Until now, there have been a large body of research efforts on detecting fake or spam reviews [21,28], as well as spammers (i.e., the person who provides spam reviews) and spammer groups [21]. Most existing approaches achieve these purposes by extracting features from the review text, ratings, product metadata (category and price), or review feedback (helpful/unhelpful votes etc.) [10]. Some researchers improve performance by considering user behavior patterns (e.g., user's interactions with products or other users) [33] or the probabilistic distribution of users' behaviors(spamming and non-spamming) [16]. Rayana and Akoglu [24] design a collective unified framework to exploit linguistic clues of deception, behavioral footprints, or relational ties between agents in a review system.

As for the learning techniques, several previous works mainly use traditional classification methods such as Support Vector Machine (SVM) [22], Naive Bayes Classifiers, and logistic regression [10]. Some researchers consider to use text analysis, for example, learning the written patterns [25]. Some used probabilistic methods such as unsupervised Bayesian approach [20,36,37], and Hidden Markov models [16]. And limited works considered using deep learning based methods [15] for detecting the fake reviews [30]. Deep learning methods, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [15], have achieved a great amount of success in works such as Image Recognition [5]. They show advantages in dealing with high dimensional data and utilize the non-linear combination of input features – where for detecting the fake news, we are also dealing with various features and try to find their link to the signal of fake. Some related works include Ma's attempt [18], they used RNN for detecting rumors in microblogs; and in Baohua's work [30], they designed semi-supervised recursive autoencoders for detecting review spam in microblogs. However, is there a way that could combine the deep learning methods and traditional classification methods, to fully extend the advantages of different methods?

In this work, we present an end-to-end trainable unified model to leverage the appealing properties of Autoencoder [14] and random forest [17]. And the reason for combining these two methods is shown in the following. Autoencoder has been proved to be a ro-

[*] Corresponding author.
E-mail address: manqing.dong@student.unsw.edu.au (M. Dong).

https://doi.org/10.1016/j.patrec.2018.07.013
0167-8655/© 2018 Elsevier B.V. All rights reserved.

bust algorithm which can produce unsupervised representations in feature patterns, and such representations are often more concise. Random forest is the ensemble of several decision trees, which can help prevent over fitting in each tree and shows good performance in practice. In our unified model, we use autoencoder to generate the hidden representations of the features, and take them as the input of the random forest. The entire model is trained jointly via the stochastic and differentiable decision tree model, and the decision forest generates the final prediction. To summarize, we make the following contributions to the field of opinion fraud detection.

- We employ statistical analysis to define a list of quality measures, which utilize multiple metrics such as reviewer behavioral patterns, and review content analysis. These measures will be used as discriminative indicators of spamicity.
- We propose to detect spam reviews with the learned quality features to infer the salient correlations between reviews and heterogeneous cross-domain historical user information. We propose a joint model by fusing autoencoder and random forest to harvest the merits in an end-to-end trainable way. The model is learned by back propagation using stochastic and differentiable decision tree model. As such, the global optimization of model parameters like splits, leaves and nodes can be obtained.
- Experimental results on the real dataset demonstrate that the proposed model achieves the best prediction accuracy compared to several baselines and state-of-art methods. We have made the related code and dataset open-sourced for reuse.

2. Related work

Opinion spam detection has been an active field of research in recent years, which covers broad topics such as detecting singleton fake reviewers, fake reviews, or even spam groups. For example, Jindal and Liu [10] built a logistic regression classifier with review feedback features, content characteristics, and rating related features to detect fake reviewers; Feng et al. [4] investigated syntactic stylometry for deceptive reviews detection by using SVR classifiers; Mukherjee et al. [21] propose a model for group opinion spam spotting with assistance of human experts.

Most detection methods are learning discriminate features from the linguistic [4,22], relational [1], and behavioral aspects [4,21] based on review text, ratings, product meta-data (category and price), and review feedback (helpful votes), reviewer behavioral patterns and linkage structures [38,39]. Original fake review detection methods mainly consider only the review text information, for example, Ott et al. [22] consider text categorization and the sentiment of the text. And later on, a lot of works combine those different aspects of features for improvements. In Rayana's work [24], they combine features such as review texts, timestamps, user behavioral information and the review network. However, limited work found have conducted quality feature analysis (e.g., the importance of a feature) for evaluating the selected features.

The techniques used in detecting the opinion fraud vary a lot. Some researchers use classical classification methods such as Support Vector Machine (SVM) [22], Naive Bayes Classifiers, and logistic regression [10]. Such as Sandulescu and Ester [28] employ bag-of-words and bag-of-opinions Latent Dirichlet Allocation (LDA) model to detect singleton review spammers based on semantic similarity, in which they use cosine similarity as a measure, and topics and extracted from the reviews text, with parts-of-speech (POS) as patterns. And others may prefer statistical methods: Mukherjee et al. [20] use unsupervised Bayesian approach and considered user's behavior features and bigrams; A most recent work [16] models the distribution of reviewers' posting rates and utilizes a coupled hidden Markov model to capture both reviewers' posting behaviors and co-bursting signals. There are also some trials on deep learning, which achieves great success in several areas in recent years. Several attempts include Wang's work [31], which proposes using a hybrid convolutional neural network to detect fake news. Wang et al. [30] designed semi-supervised recursive autoencoders for detecting review spam in Weibo, a Chinese alternative to Twitter. Still, to the best of our knowledge, very limited work on hybrid deep learning models are proposed for fake opinion detection.

In this work, we propose to integrate neural networks with neural random forest, which is a differentiable model that could be fused into deep learning methods. And there have been already several attempts on leveraging the benefits of deep learning methods and random forest [40]. One of the most important work in this area is the studies that cast the neural random forest as the structure of neural networks. The first few attempts started in the 1990s when Sethi [29] proposed entropy net, which encoded decision trees into neural networks. Welbl et. al [32] further proposed that any decision trees can be represented as two-layer Convolutional Neural Networks (CNN). Since the classical random forest cannot conduct back-propagation, other researchers tried to design differentiable decision tree methods, which can be traced from Montillo's work [19]. They investigated the use of sigmoidal functions for the task of differentiable information gain maximization. In Peter et al.'s work [12], they introduce a stochastic, differentiable, back propagation compatible version of decision trees, guiding the representation learning in lower layers of deep convolutional networks. Different from their work, we develop a hybrid model by combining neural decision forest with lightweight autoencoder to detect fraud opinion.

3. Quality feature representation

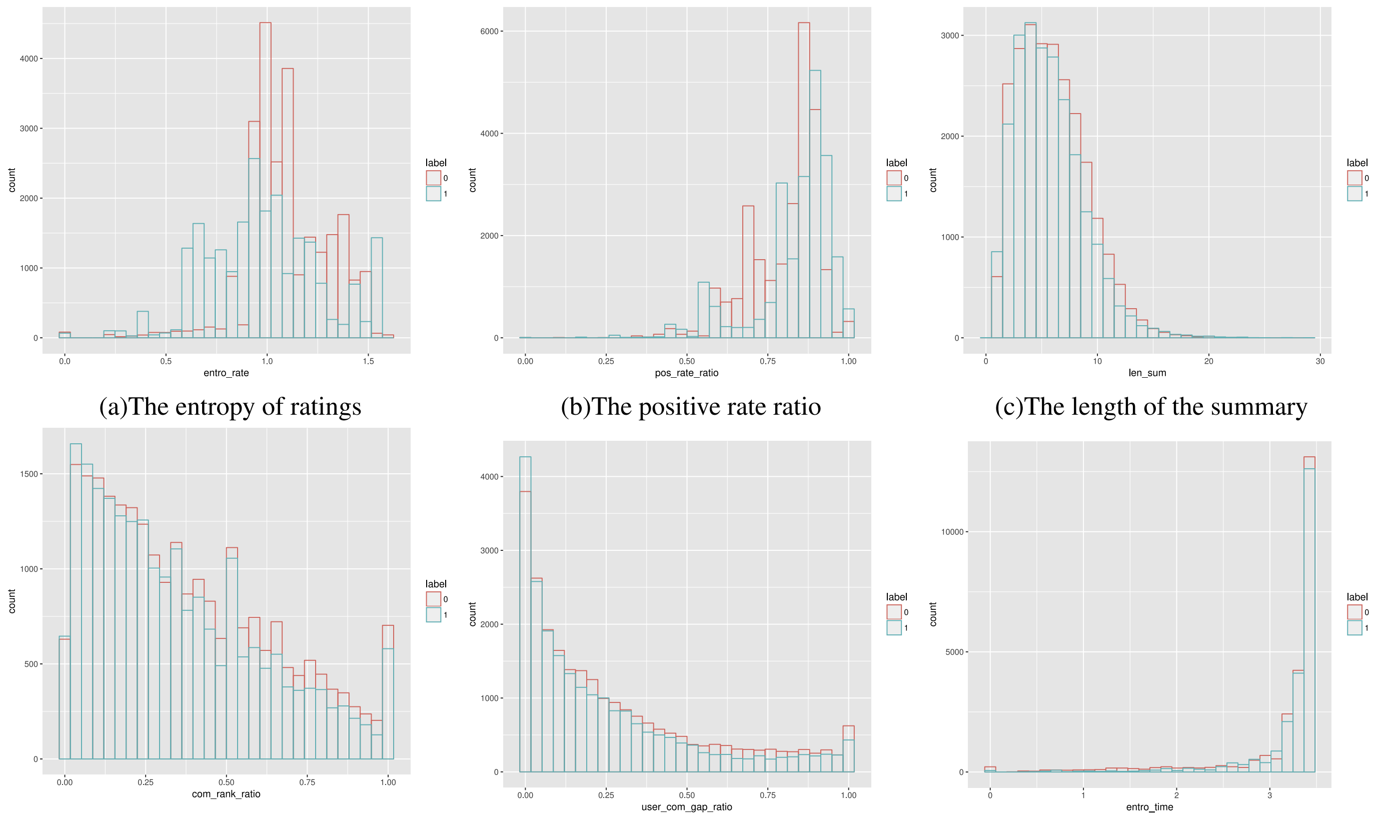
For detecting the fake reviews, the first thing is defining the features. In this paper, we mainly considered two scopes of features, one is user's behavioral features and one is user's review content features. The selected numerical features and the description of those features are listed in Table 1. And the categorical features will be shown later.

We extensively studied the reviewer behavior patterns in this section for the good understanding of the difference between spamicity and non-spamicity.

First, we investigate the difference between spammers and non-spammers. We used Wilcoxon Signed-Rank Test [11] to evaluate this difference over the continuous data such as entropy of ratings and review time entropy. This is a non-parametric statistical hypothesis test used to compare two related samples, and determines whether their population means ranks differ. Also there is no need to make an assumption on data distribution while using the *Wilcoxon Signed-Rank Test*.

Table 2 shows the p-value of the Wilcoxon Signed-Rank Test of three different alternative hypotheses: not equal to 0, less than 0, and greater than 0. A small p-value, especially a p-value is less than .05, stands for the high possibility to accept the alternative hypothesis, which also indicates a significant difference between spammer and non-spammer. For example, from the first line from Table 2, we could know that the entropy of ratings for spammers is significantly less than non-spammers, which indicates the spammers are tending to give similar rates. The features in bold typeface are the features that show low significance in the distinguishing spammers and non-spammers.

Fig. 1 shows a set of distinguishable patterns of review metadata regarding density distribution. The red column stands for non-spammers and the green column stands for spammers. Fig. 1(a) shows the entropy of ratings for the spammers is lower than non-



(a)The entropy of ratings (b)The positive rate ratio (c)The length of the summary

(d)The proportion of the user's comment rank (e)The user's time gap ratio (f)The entropy of the rating time

Fig. 1. The histogram of the spammer and non-spammer in (a)the entropy of the ratings, (b) the positive rate ratio, (c) the length of the summary, (d) the proportion of the user's comment rank around all reviews, (e) the user's time gap ratio for the comment time, (f) the entropy of the rating time.

spammers, which is in consistency with our p-value showed above. Fig. 1(b) reflects that the spammers are holding the higher value than non-spammers in both low and high positive rate ratio, but lower than non-spammers in the middle. This can be explained that the spammers would likely to keep his/her rating habit, that is, a user who always rates very low scores or very high scores may likely be a spammer. Fig. 1(c) describes the number of words of the summary, we could say that spammers would like to use fewer words than non-spammers; Fig. 1(d) indicates that the spammers tend to comment early on the products; Fig. 1(e) shows that spammers would be more likely appeared in products which is released for a long time; and from Fig. 1(f) we could see spammers comment more randomly than non-spammers.

For categorical features, we apply Pearson χ^2 test [3] to evaluate the correlation between the features and the labels (whether a reviewer is a spammer or not). Pearson χ^2 test is a statistical test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance. The null hypothesis of the test is stating that the frequency distribution of certain events observed in a sample is consistent with a particular theoretical distribution. So if p-value is less than .05 stands the feature has a high correlation with the labels.

In Table 3, min/max rate is the minimal/maximal score a user had through his/her review history; common username is a binary variable – we extracted the normal English name list from world-english.org, the weirdness shows whether user's name showed in the name list, if so is valued as 0 otherwise is 1; having memo or not stands for whether a user has self-description; semantic of the summary stands for the semantic value for the review summary content; Semantic of the summary is the text analysis over review comments. The features in bold typeface don't demonstrate the significant difference between spammers and non-spammers.

4. Proposed methodology

In this section, we introduce our proposed methodology, which can be referred as Fig. 2. The whole idea is firstly to initialize all the parameters, and feed the features into the input layer X_I of autoencoder; also the cleaned features (the reconstruction of input layer X_I) are regarded as X_C. Then we get the hidden layer H, whose nodes will be followed by fully connected layers and reaches the nodes in the tree. After that, the nodes in the tree layer are delivered to the nodes in k decision trees. Decision nodes will make a decision and give a prediction by averaging the decision results. Then we update all the parameters by minimizing the cost between the prediction and the real value, and iteratively optimize the whole process till to reach the optimal results.

Auto-encoder is a neural network [15] that is trained to attempt to copy its input to its output. And it always consists of two parts, namely the encoder and the decoder. Internally, it has a hidden layer H describes a code used to represent the input.

An auto-encoder takes an input $X_I \in \mathbb{R}^n$, where X is the input feature vectors, and n is the number of the features.

First we map it (with an encoder) to a hidden representation $H \in \mathbb{R}^m$,

$$H = f_E(W_E X_I + b_E) \tag{1}$$

where $f_E(*)$ is a non-linearity such as the sigmoid function [7]. And W_E and b_E are weights and bias variables used in encoder part

Table 1
Feature explanation.

User's behavioral data
Number of reviewed products
Length of name
Reviewed product structure: the products include 54 different categories, such as book and music, here indicates the ratio of user's reviewed product's categories.
Minimum score and maximum score
Ratio of each score: 1 to 5
Number of each score
Ratio of positive and negative ratings: the proportion of high rates (score 4 and 5) and low rates (score 1 and 2) of a user
Entropy of ratings: as a measure of skewness in ratings. The entropy of the user's rating can be explained by $-\sum_{i=1}^{5} p_i \log p_i$, where p_i is the proportion that the user rated score i.
Total number of helpful and unhelpful votes
Average number of helpful and unhelpful votes
The ratio of the helpful and unhelpful votes
Median, min, max number of helpful and unhelpful votes.
Day gap: the time gap between the user's first review and last review by days.
Review time entropy: as a measure of skewness in user's review time by years, which could be denoted by formula $-\sum_{j=1}^{m} t_j \log t_j$, where m means the year gap between the user's first review and last review, and t_j represent the proportion of the number of reviews a user generated for year j.
Same date indicator: whether the user's first comment and the last comment are on the same date, 1 indicate the same date.
Active ratio: the proportion that a user was active during m years.
User's review data
The Average review's ratings for the product
Number of reviews
Entropy of the scores: as a measure of the skewness for the product's review ratings. Can be denoted by $-\sum_{i=1}^{5} s_i \log s_i$, where s_i denotes the proportion for score i in the whole reviews.
Comment time gap: the duration of the first review and the last review by days.
Entropy of product's comment time: as a measure of the skewness for the product's review times, *month* stands for the duration of the reviews for the product by month, then the entropy of product's comment time can be denoted by $-\sum_{i=1}^{month} r_i \log r_i$, where r_i means the ratio of the reviews created in month i.
User's rate for the product.
User's helpful and unhelpful votes from others.
User's comment time gap ratio: the days gap between user's comment time and the first comment divided by whole comment time gap for the product.
User's comment time rank: rank equal to 1 means this user is the first person giving the product review ratio.
The ratio of the user's comment time rank and the total number of comments.
Review summary length: the number of words of the summary text.
Review text semantic: the semantic value for the review content, 1 stands for positive, 0 stands for neutral, -1 stands for negative.

Table 2
p-Value of Wilcoxon Signed-Rank Test for continuous features.

Features	**(p-Value) with alternative hypothesis**		
	Not equal to 0	Less than 0	Greater than 0
	User's history features		
Entropy of ratings	2.8e−05	1	1.4e−05
Number of reviewed products	.0492	.0246	.9754
Ratio of positive ratings	.0003	.0001	.9998
Ratio of negative ratings	.1007	.9497	.0503
Number of helpful votes	.0001	5.3e−05	.9999
Sum unhelp	4.2e−06	2.1e−06	1
Average help	8.1e−08	4.1e−08	1
Average unhelp	1.5e−11	7.3e−12	1
Time gap	.0024	.0012	.9988
Entropy of rating time	.0006	.0003	.9997
Active ratio	.0085	.9958	.0042
Memo length	.8915	.5544	.4457
Mean rate	.0004	.0002	.9998
	User's review data		
Overall product score	5.8e−09	2.9e−09	1
Number of comments	2.2e−16	1	2.2e−16
Number of first day's comments	.4217	.7892	.2108
Product's comment time gap	2.2e−16	2.2e−16	1
Comment rank	2.2e−16	1	2.2e−16
Comment rank ratio	2.2e−16	1	2.2e−16
User help	2.2e−16	2.2e−16	1
User unhelp	2.2e−16	2.2e−16	1
Length of the summary	2.2e−16	1	2.2e−16
Length of the review text	.5556	.2778	.7222
User comment time gap	5.4e−05	2.7e−05	1
User comment time gap ratio	2.2e−16	1	2.2e−16

Table 3
p-Value of Pearson χ^2 test for categorical features.

Features	p-value
Min rate	1.4e−06
Max rate	0.1158
Common name	0.1721
Having memo or not	1
User rate	2.2e−16
Semantic of the summary	0.1554
Semantic of the review text	2.2e−16

The latent representation H or code is then mapped back (with a decoder) into a reconstruction X_C,

$$X_C = f_D(W_D H + b_D) \tag{2}$$

where X_C is seen as a prediction of X_I, given the code H. And $W_D$and b_D are weights and bias variables used in decoder part, they are optimized such that the average reconstruction error is minimized.

The reconstruction error can be measured in many ways, depending on the appropriate distributional assumptions on the input given the code. For example, the traditional squared error $L_{AE}(X_I, X_C) = \| X_I - X_C \|^2$ can be used.

Then let the hidden layer H go through several fully connected layers, we get X_T as the input nodes for the neural decision forest

$$X_T = g(W_F H + b_F) \tag{3}$$

where W_F and b_F are the weights used in the fully connection layers, and similarly, the $g(*)$ is the activation function. And in the next we will introduce how input those tree nodes X_T into the tree layer.

For an end-to-end neural network, each part should be differentiable, where traditional random forest is not suitable. Thus here in our work we utilized a differentiable version of random forest, which is called neural decision forest.

The neural decision forest [12] is designed by Kontschieder et al., each node in the decision tree holds probabilistic distribution thus make the decision tree differentiable. Below explain the details.

Suppose we have number of K trees, for each tree, it is a structured classifier consisting of decision(or split) nodes $\mathcal{D}$ and prediction(or leaf) nodes $\mathcal{L}$. Where the leaf nodes are the terminal nodes of the tree, and each prediction node $\ell \in \mathcal{L}$ holds a probability distribution p_ℓ over Y. Each decision node $d \in \mathcal{D}$ holds a decision function $D_d(X_T; \Theta) \in [0, 1]$, which stands for the probability that a sample reached decision node d and then be sent to the left sub-tree, for each decision nodes, suppose it holds the representation:

$$D_d(X_T; \Theta) = \sigma(f_d(X_T)) \tag{4}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, and $f_d(X_T)$ is the transfer function for X_T,

$$f_d(X_T) = W_T X_T \tag{5}$$

Θ stands for the set of previous parameters $\{W_E, W_D, W_F, b_E, b_D, b_F\}$.

About constructing the tree, here we suppose all the trees are following a classical binary tree structure, which means each node

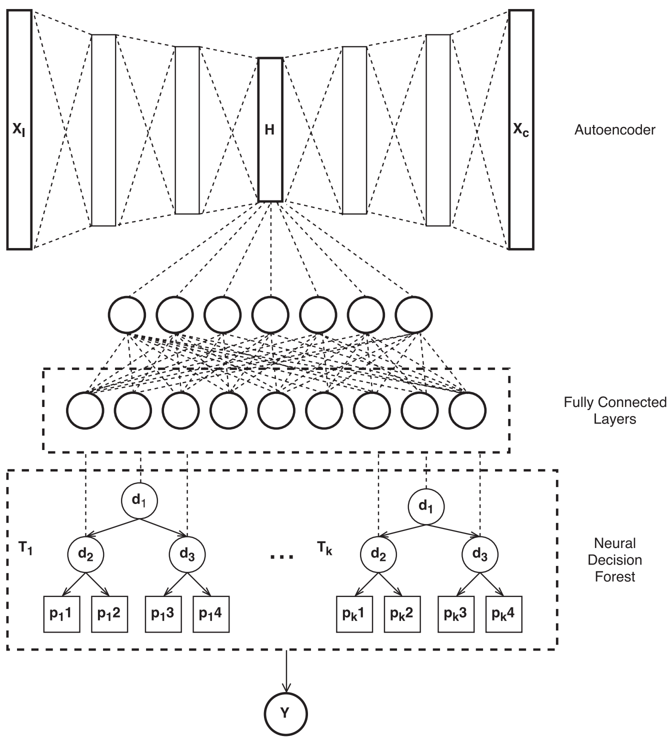


Fig. 2. Proposed neural autoencoder decision forest.

has two subtrees, then after defining the depth *n_depth* of a tree, suppose the depth is 1, just like what shows in Fig. 1, then the decision nodes in depth 1 is 2 and the number of leaf nodes is 4. More generally, once define the depth *n_depth*, the number of decision nodes would be 2^{n_depth}, and the number of leaf nodes would be $2^{n\_depth+1}$.

And then the probability of a sample reach tree k to be predicted as class y would be:

$$\mathbb{P}_{T_k}[y|X_T,\Theta,P]=\Sigma_{\ell\in\mathcal{L}}P_{\ell_y}\left(\Pi_{d\in\mathcal{D}}D_d(X_T;\Theta)^{\mathbb{1}_{left}}\overline{D_d}(X_T;\Theta)^{\mathbb{1}_{right}}\right) \quad (6)$$

where $\overline{D_d}(h;\Theta)=1-D_d(h;\Theta)$, and $\mathbb{1}_{left}$ indicates the indicator function for the nodes go left.

For example, in tree 1 T_1 in Fig. 1, to get to the leaf node p_1, here the $\Pi_{d\in\mathcal{D}}D_d(h;\Theta)^{\mathbb{1}_{left}}\overline{D_d}(h;\Theta)^{\mathbb{1}_{right}}$ equals to d_1d_2, which means the probability that a sample reach leaf node p_1, and for p_2, this probability will be $d_1\overline{d_2}$. and P_{ℓ_y} means the probability for the nodes in leaf ℓ predicted to be label y. Here, in our problem, the P_{ℓ_y} will have the form $P_{\ell_y}=(p_0,p_1)$, where p_0 stands for the probability of the leaf node to be 0 non spammer.

Then for the forest of decision trees, it is an ensemble of decision trees $\mathcal{F}=\{T_1,\dots,T_K\}$, which delivers a prediction for sample x by averaging the output of each tree, which can be showed from:

$$\mathbb{P}_{\mathcal{F}}[y|x]=\frac{1}{K}\Sigma_{k=1}^{K}\mathbb{P}_{T_k}[y|x] \quad (7)$$

Then the prediction for the label y would be:

$$\hat{y}=argmax_y\mathbb{P}_{\mathcal{F}}[y|x] \quad (8)$$

The loss for the neural decision forest is the average of the loss of each tree. And in the training process, the total loss is the average of all the training samples.

So the whole loss functions can be defined as:

$$L_F(x,y,\Theta,P)=E_{x\in X}(E_{T\in\mathcal{F}}(L_T(x,y,\Theta,P))) \quad (9)$$

And here, for we are dealing with a classification problem, the loss function we chose here is defined as:

$$\begin{aligned}L_T(x,y,\Theta,P)&=-(y\times log(\mathbb{P}_T[y|x,\Theta,P])\\&\quad+(1-y)log(\mathbb{P}_T[y|x,\Theta,P]))\\&=-log(\mathbb{P}_T[y|x,\Theta,P])\end{aligned} \quad (10)$$

where y=0 or 1.

Then our final optimal function is minimizing the whole loss function which is finding:

$$\mathcal{L}(X_I,y,\Theta,P)=\arg\min_{\Theta}L_{AE}(X_I,X_C)+L_F(x,y,\Theta,P) \quad (11)$$

For learning the parameters, the optimization function we chose here is the RMSProp [27], which is a mini-batch version of RProp, and RProp is equivalent to using the gradient but also dividing by the size of the gradient, the parameters are updating by:

$$g_t=\nabla\mathcal{L}(\Theta_{t-1};\mathcal{B},P) \quad (12)$$

$$G_t=G_t+g_t\odot g_t \quad (13)$$

$$\Theta_t=\Theta_{t-1}-\frac{\eta}{\sqrt{G_t+\epsilon}}\odot g_t \quad (14)$$

where g_t is the gradient, the η and ϵ are learning rates, and $\mathcal{B}$ is a random subset of training data set.

For learning the leaf node distribution probability P, we still follow a RProp process but add a softmax function to the updated parameters.

$$g_t^P=\nabla\mathcal{L}(P_{t-1};\mathcal{B},\Theta)=\nabla L_F(P_{t-1};x,y,\Theta) \quad (15)$$

$$G_t^P=G_t^P+g_t^P\odot g_t^P \quad (16)$$

$$P_t=P_{t-1}-\frac{\eta^P}{\sqrt{G_t^P+\epsilon}}\odot g_t^P \quad (17)$$

$$P_t=softmax(P_t) \quad (18)$$

The details for how to calculate the derivative of loss function for each parameters could refer from paper [12]. And below is the pseudo-code for our algorithm for both training process and testing process.

4.1. Computational complexity

Our proposed process mainly contains three parts, the autoencoder part, the fully connected layers, and the neural random forest part.

For each epoch, for the autoencoder part, the time complexity is $\mathcal{O}(\Sigma_l^{2*n\_AE}n_{l-1}^{AE}n_l^{AE})$, where *n_AE* is the number of layers from input layer to hidden layer, l is the layer index, n_l^{AE} is the number of nodes in an autoencoder layer l.

For the fully connected layers part, the time complexity is $\mathcal{O}(\Sigma_l^{n\_FC}n_{l-1}^{FC}n_l^{FC})$, where *n_FC* is the number of layers for fully connected layers, and similarly, n_l^{FC} is the number of nodes in fully connected layer l.

As for the neural random forest part, the time complexity is $\mathcal{O}(n_{n\_FC}\times n\_tree\times n\_leafnodes)$, where n_{n_FC} is the number of output nodes from fully connected layers, *n_tree* is the number of trees in a forest, and *n_leafnodes* is the number of leafnodes in a tree.

For the whole training process, the whole time complexity should multiply *n_epoch* and *n_batch*, which are the number of epochs and number of batch of the dataset.

5. Experiments

We evaluate proposed approach using a real-world dataset. In the following section we first describe our dataset, and then compared existing methods with the presentation of results and analysis. Also, experiment settings and parameter tuning are explained, as well as the impact of features.

5.1. Dataset description

The dataset used for our work is based on the public collected Amazon review data set from He and McAuley [6], and ground-truth spamming review dataset from Mukherjee et al. [21]. We preprocessed the dataset as follows. We obtained a manual labeled subset for 815 unique users in more than 2000 potential groups. After averaging the manually labeled spamming score (ranging from 0 and 1, more the score approaches 1, more likely this user is a spammer.), we labeled users with the average score less than 0.5 as a non-spammer and labeled them 0, otherwise, the spammers are labeled 1. Filtering is also carried out to handle missing values. To control the high appearance of some users, who wrote even higher than 500 reviews, we set a threshold (20 reviews) for selecting user's reviews. Then we got 7950 reviews as our dataset, which is used for spamming activity detection. We regard reviews written by spammers as fake reviews and whose number counts to 3363.

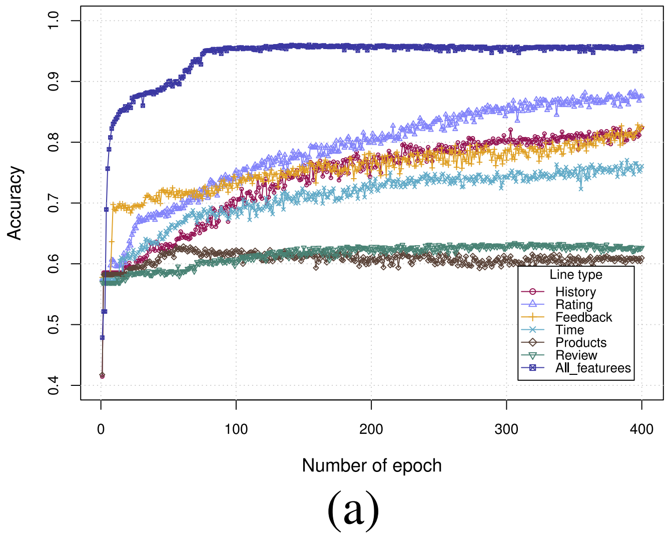
5.2. Impact of features

In this section, we evaluated the impact of different types of feature subsets on the performance. All features are stated in Table 1, and we divide the features into 6 scopes:

- History records: number of reviewed products; length of name; reviewed product structure.
- Rating signal: minimum score and maximum score; Ratio of each score (1–5) ; number of each score; Ratio of positive and negative ratings; entropy of ratings.
- Feedback signal: total number of helpful and unhelpful votes; average number of helpful and unhelpful votes; the ratio of the helpful and unhelpful votes; median, min, max number of helpful and unhelpful votes.
- Time signal: day gap; review time entropy; same date indicator; active ratio.
- Product's comment information: the average reviews ratings for the product; number of reviews; entropy of the scores; comment time gap; entropy of products comment time.
- User's review information: users rate for the product; users helpful and unhelpful votes from others; users comment time gap ratio; users comment time rank; the ratio of the users comment time rank and the total number of comments; review summary length; review text semantic.

We run the proposed neural autoencoder decision forest using these 6 types of feature, as well as full feature set. And we draw the accuracy line chart and the proportion of the predicted results for each type of features, which is shown in Fig. 3.

From Fig. 3, we can observe that the performance with full feature set consistently outperforms the prediction results with single type of feature set; among the six type of features, user's rating signal is most helpful for predicting fake reviews. Which demonstrate the effectiveness of designated various types of features in this work, and the effectiveness of our proposed model for combining different scope of features to final prediction.



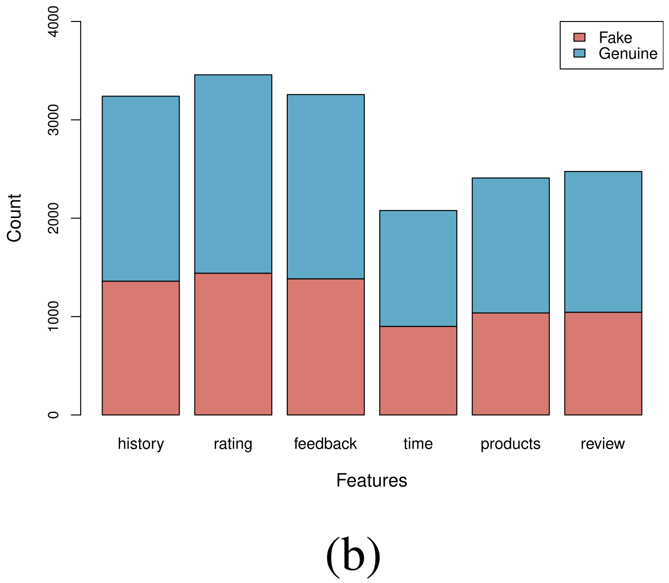


Fig. 3. (a) Accuracy with different feature subset, the top line shows the accuracy using full feature set; (b) predicted results using different features.

5.3. Parameter tuning

Last section we showed the effectiveness of our model for combining the features, and in this part, we are going to tuning the parameters. We mainly considered the parameters: the number of depth of each tree in the neural decision forest; with or without fully connected layers between autoencoder and neural decision forest; number of layers in autoencoder; the normalization methods [9]; the batch size; and the number of trees in a forest.

In the training process, among the 7950 reviews, we randomly selected 4000 samples, and initialized the weights and biases of autoencoder with normally distributed random numbers. The default set for our model is 2 layers of autoencoder, with one fully connected layer after the autoencoder, and 5 trees with depth 2 in the neural random forest. The batch size is 100 and we take z-score as our normalization method. Then we iterated the model with 400 epochs.

We compared the accuracy and time cost among different parameters, which is shown in Fig. 4.

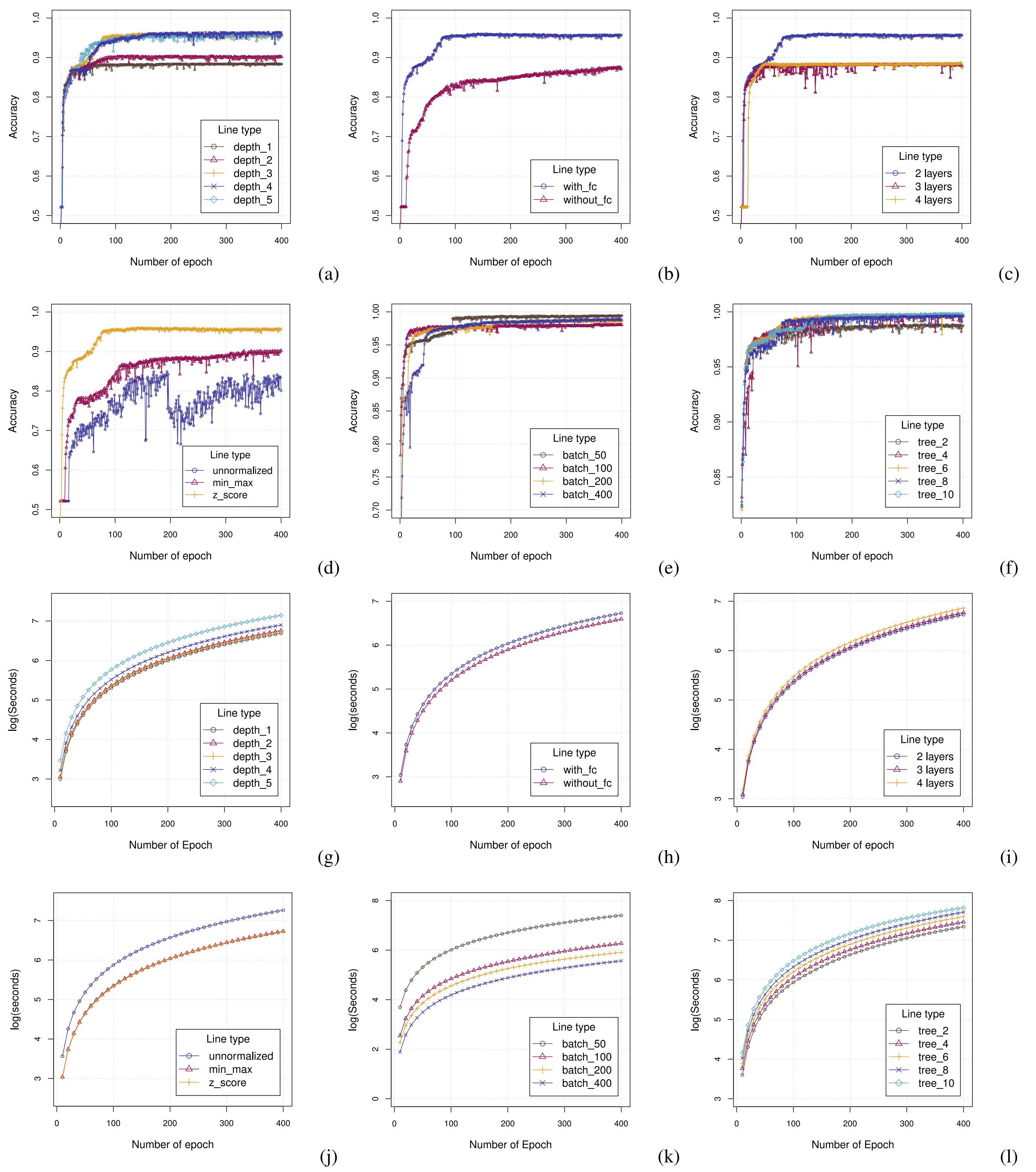


Fig. 4. The accuracy for 400 epochs of the parameter: (a)the number of depths of the tree, (b)with or without fully connected layers between autoencoder and neural decision forest, (c)the number of autoencoder layers, (d)the normalization methods, (e) the batch size, and (f) the number of trees in a forest. And the time for 400 epoches of the according parameters: (g)–(l).

From Fig. 4(a)–(f) we can see that the random forest with only 1 depth outputs a bad prediction in performance, and the accuracy becomes better with 2 and onwards in depths, which indicates that a deeper tree may learns more information; adding fully connected layers after getting the hidden representations from the autoencoder can help with the prediction accuracy; a model with 2 layers in autoencoder can get great prediction accuracy in first 100 epochs, and more layers for each autoencoder results in longer to converge and give the result; at least for our dataset, the normalized dataset shows better performance than the unnormalized one, and z_score has a better performance than min-max in terms of normalization; batch size 50 produces the optimal prediction, with a comparable short computation time at the highest prediction accuracy; and finally, the prediction improves with increased number of trees in a forest but takes more time for learning.

Algorithm 1 Training process.

```
Require: Features vector of the review $X$, Label for the review (fake or not) $y$, number of epoch $n\_epoch$, number of trees $n\_tree$, number of depth for a tree $n\_depth$, and the number of batches $n\_batch$.
1: Generalizing the structure for the forest with $n\_tree$ and $n\_depth$
2: Initialize parameters $\Theta, P$ with random variables
3: Randomly shuffle the data sets
4: for i in 1:$n\_epoch$ do
5:     break data sets into $n\_batch$ of $batch\_size$ pieces
6:     for j in 1:$n\_batch$ do
7:         Get the tree input layer through the autoencoder:
8:         $X_T = f(X_{batch}, \Theta)$
9:         Feed the tree input into the neural decision forest:
10:        $\mathbb{P}_{\mathcal{F}}[y_{batch}|x_{batch}] = \frac{1}{K}\sum_{k=1}^{K}\mathbb{P}_{T_k}[y_{batch}|x_{batch}, P]$
11:        Get the loss function as shown in equation (11).
12:        Update $\Theta$ by (12) to (14) .
13:    end for
14:    Update P by (18) to (18)
15: end for
```

Fig. 4(g)–(l) show the time consumption under different parameter settings. The axis y is taking the *log* for the time (seconds). We could see deeper the depth of the tree, more time will be cost; add fully connected layers will increase the time cost; the time cost will increase with bigger number of autoencoder layers; unnormalized dataset cost more time; less batch size takes more training time; and higher tree depths incurs more training time.

Algorithm 2 Prediction with proposed approach.

```
Require: Feature vector of a review $X$, number of batches $I$, and number of trees in the forest $K$.
Ensure: Label for the review (fake or not) y
1: Get hidden representation for $X$ by an encoder:
2: $H = f_E(W_E X_I + b_E)$
3: Get tree input layer from fully connected layers:
4: $X_T = g(W_F H + b_F)$
5: for i in 1:I do
6:     p(y=i) = 0
7:     for k in 1:K do
8:         $p = P_k[y = i|X_T, \Theta, P]$
9:         $p(y = i) = p(y = i) + p$
10:    end for
11:    p(y=i) = $\frac{1}{K}p(y = i)$
12: end for
13: $y = \max_i p(y = i)$
14: return y
```

Thus, considering the time cost and the prediction performance, we choose a batch size with 50, 5 trees in a random forest with depth of 3, z-score as normalization method, one fully connected layer, and 2 layers in autoencoder. In the test we get correctly predicted fake reviews and 2192 correctly predicted genuine reviews, 65 falsely predicted fake reviews and 99 falsely predicted genuine reviews. This means an accuracy of approximately 95.85%.

Table 4
Comparison with the state-of-art methods.

Methods	Precision	Accuracy	F1	Recall
Jindal and Liu	–	78%	–	–
Lai et al.	–	–	–	**96.38%**
Mukherjee et al.	79.6%	86.1%	77.3%	75.1%
Rout et al.	–	92.11%	–	–
Shreesh et al.	87.6%	–	84.3%	82.8%
Heydari et al.	82%	–	86%	88%
Xu and Zhang	86.4%	85.2%	–	–
Zhang et al.	87.12%	87.81%	88.31%	89.63%
Ours	**96.08%**	**95.85%**	**95.11%**	94.15%

5.4. Overall comparison

We compared our model with a set of existing works, which also use Amazon dataset but are different in extracted features and solutions. A brief introduction for these papers is listed below:

- Jindal and Liu [10] first manually labelled easy identify spam reviews and then considered multi-features, they used logistic regression to do the classification.
- Lai et al. [13]: considered the content of the reviews, they used an unsupervised probabilistic language model and a supervised SVM model.
- Mukherjee et al. [20]: This work used unsupervised Bayesian approach and considered user's behavior features and bigrams.
- Rout et al. [26]: considered the sentiment analysis and they extracted sentiment score, linguistic features and unigram as feature set.
- Bhat and Culotta [2]: mainly focused on the content based information and they combined positive unlabeled learning with domain adaptation to train a classifier.
- Heydari et al. [8]: Heydari et al. considered the rating deviation, content based factors and activeness of reviewers, and they captured suspicious time intervals from time series by a pattern recognition technique.
- Xu and Zhang [34]: considered target based, rating based, temporal based and activity based signals. A Latent Collusion Model (LCM) is proposed to work unsupervisedly and supervisedly under different conditions.
- Zhang et al. [35]: considered unverbal features and used four classical supervised classification methods: SVM, Naive Bayes, decision tree and random forest.

To evaluate the performance of our fake news detection algorithms, we used the following criteria [23]: $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F1 = \frac{2TP}{2TP+FP+FN}$. Where TP, FP, TN, FN represent the counts for true positive, false positive, true negative and false negative, respectively.

Table 4 shows the comparison results. And the results for each paper are extracted from the published papers, the "-" means this value is not calculated in the original paper. We could see that our proposed model gives the best prediction accuracy, precision and F1 score values.

Comparing with these methods, we could conclude that our approach demonstrates readily good performance in predicting the fake reviews. It also indicates that the proposed neural autoencoder random forest model is at least an appreciable classification model in predicting the review spam.

6. Conclusion

In this paper, we focus on detecting fake reviews, which is an attractive topic and has drawn attention of certain researchers.

While previous work mainly focused on using classical classification methods, here we proposed an end to end trainable joint model combining autoencoder with neural random forest, to detect the fake reviews. Firstly, we did quality feature analysis to mining the feature efficiency in detecting opinion fraud, and those efficient features are employed as input in our model. And then we declared the design of our model. The model starts with an autoencoder, and then fully connected layers and ends in a differentiable random forest, where the differentiable random forest is trained back propagation. The forest averages the prediction result of each tree and outputs our final prediction. We also analyzed the impact of features and tuned various parameters in our model. The extensive experimental results demonstrate that our method beats a series of state-of-the-art methods yielding 96% of accuracy.

References

[1] L. Akoglu, R. Chandy, C. Faloutsos, Opinion fraud detection in online reviews by network effects., ICWSM 13 (2013) 2–11.
[2] S. K. Bhat, A. Culotta, Identifying leading indicators of product recalls from online reviews using positive unlabeled learning and domain adaptation, arXiv preprintarXiv:1703.00518(2017).
[3] G.W. Corder, D.I. Foreman, Nonparametric Statistics: A Step-by-Step Approach, John Wiley & Sons, 2014.
[4] S. Feng, R. Banerjee, Y. Choi, Syntactic stylometry for deception detection, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics, 2012, pp. 171–175.
[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[6] R. He, J. McAuley, Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 507–517.
[7] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: Neural networks for perception, Elsevier, 1992, pp. 65–93.
[8] A. Heydari, M. Tavakoli, N. Salim, Detection of fake opinions using time series, Expert Syst. Appl. 58 (2016) 83–92.
[9] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, 2015, pp. 448–456.
[10] N. Jindal, B. Liu, Opinion spam and analysis, in: Proceedings of the 2008 International Conference on Web Search and Data Mining, ACM, 2008, pp. 219–230.
[11] D.S. Kerby, The simple difference formula: an approach to teaching nonparametric correlation, Compr. Psychol. 3 (2014) 11–IT.
[12] P. Kontschieder, M. Fiterau, A. Criminisi, S.R. Bulo, Deep neural decision forests, in: Computer Vision (ICCV), 2015 IEEE International Conference on, IEEE, 2015, pp. 1467–1475.
[13] C. Lai, K. Xu, R.Y. Lau, Y. Li, L. Jing, Toward a language modeling approach for consumer review spam detection, in: e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on, IEEE, 2010, pp. 1–8.
[14] S. Lange, M. Riedmiller, Deep auto-encoder neural networks in reinforcement learning, in: Neural Networks (IJCNN), The 2010 International Joint Conference on, IEEE, 2010, pp. 1–8.
[15] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[16] H. Li, G. Fei, S. Wang, B. Liu, W. Shao, A. Mukherjee, J. Shao, Bimodal distribution and co-bursting in review spam detection, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 1063–1072.
[17] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, R News 2 (3) (2002) 18–22.
[18] J. Ma, W. Gao, P. Mitra, S. Kwon, B.J. Jansen, K.-F. Wong, M. Cha, Detecting rumors from microblogs with recurrent neural networks., in: IJCAI, 2016, pp. 3818–3824.
[19] A. Montillo, J. Tu, J. Shotton, J. Winn, J.E. Iglesias, D.N. Metaxas, A. Criminisi, Entanglement and Differentiable Information Gain Maximization, in: Decision Forests for Computer Vision and Medical Image Analysis, Springer, 2013, pp. 273–293.
[20] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, R. Ghosh, Spotting opinion spammers using behavioral footprints, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 632–640.
[21] A. Mukherjee, B. Liu, N. Glance, Spotting fake reviewer groups in consumer reviews, in: Proceedings of the 21st International conference on World Wide Web, ACM, 2012, pp. 191–200.
[22] M. Ott, Y. Choi, C. Cardie, J.T. Hancock, Finding deceptive opinion spam by any stretch of the imagination, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, 2011, pp. 309–319.
[23] D.M. Powers, Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation (2011).
[24] S. Rayana, L. Akoglu, Collective opinion spam detection: bridging review networks and metadata, in: Proceedings of the 21th ACMSIGKDDInternational Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 985–994.
[25] P. Ren, Z. Chen, Z. Ren, F. Wei, J. Ma, M. de Rijke, Leveraging contextual sentence relations for extractive summarization using a neural attention model, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 95–104.
[26] J.K. Rout, S. Singh, S.K. Jena, S. Bakshi, Deceptive review detection using labeled and unlabeled data, Multimed. Tools Appl. 76 (3) (2017) 3187–3211.
[27] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprintarXiv:1609.04747(2016).
[28] V. Sandulescu, M. Ester, Detecting singleton review spammers using semantic similarity, in: Proceedings of the 24th international conference on World Wide Web, ACM, 2015, pp. 971–976.
[29] I.K. Sethi, Entropy nets: from decision trees to neural networks, Proc. IEEE 78 (10) (1990) 1605–1613.
[30] B. Wang, J. Huang, H. Zheng, H. Wu, Semi-supervised recursive autoencoders for social review spam detection, in: Computational Intelligence and Security (CIS), 2016 12th International Conference on, IEEE, 2016, pp. 116–119.
[31] W.Y. Wang, “Liar, liar pants on fire”: a new benchmark dataset for fake news detection, arXiv preprintarXiv:1705.00648(2017).
[32] J. Welbl, Casting random forests as artificial neural networks (and profiting from it), in: German Conference on Pattern Recognition, Springer, 2014, pp. 765–771.
[33] Z. Wu, Y. Wang, Y. Wang, J. Wu, J. Cao, L. Zhang, Spammers detection from product reviews: a hybrid model, in: Data Mining (ICDM), 2015 IEEE International Conference on, IEEE, 2015, pp. 1039–1044.
[34] C. Xu, J. Zhang, Towards collusive fraud detection in online reviews, in: Data Mining (ICDM), 2015 IEEE International Conference on, IEEE, 2015, pp. 1051–1056.
[35] D. Zhang, L. Zhou, J.L. Kehoe, I.Y. Kilic, What online reviewer behaviors really matter? Effects of verbal and nonverbal behaviors on detection of fake online reviews, J. Manag. Inf. Syst. 33 (2) (2016) 456–481.
[36] X. Wang, Q.Z. Sheng, L. Yao, X. Li, X.S. Fang, X. Xu, B. Benatallah, Truth discovery via exploiting implications from multi-source data, in: The 25th ACM Conference on Information and Knowledge Management (CIKM 2016), Indianapolis, USA, October 24-28, 2016.
[37] X. Wang, Q.Z. Sheng, L. Yao, X. Li, X.S. Fang, X. Xu, B. Benatallah, Empowering truth discovery with multi-truth prediction, in: The 25th ACM Conference on Information and Knowledge Management (CIKM 2016), Indianapolis, USA, October 24-28, 2016.
[38] X. Zhu, et al., Local and global structure preservation for robust unsupervised spectral feature selection, IEEE Trans. Knowl. Data Eng. 30 (3) (2018) 517–529.
[39] W. Zheng, et al., Unsupervised feature selection by self-paced learning regularization, Pattern Recognit. Lett. (2018).
[40] M. Dong, L. Yao, X. Wang, B. Benatallah, S. Zhang, (2018). GrCAN: Gradient Boost Convolutional Autoencoder with Neural Decision Forest. arXiv preprint arXiv:1806.08079.