IADIS International Journal on Computer Science and Information Systems
Vol. 11, No. 1, pp. 99-114
ISSN: 1646-3692

LEARNING SPAM FEATURES USING RESTRICTED BOLTZMANN MACHINES

Luis Alexandre da Silva, Kelton Augusto Pontara da Costa, Patricia Bellin Ribeiro, Gustavo Henrique de Rosa and João Paulo Papa. *Department of Computing São Paulo State University, São Paulo, Brazil.*

ABSTRACT

Nowadays, spam detection has been one of the foremost machine learning-oriented applications in the context of security in computer networks. In this work, we propose to learn intrinsic properties of e-mail messages by means of Restricted Boltzmann Machines (RBMs) in order to identity whether such messages contain relevant (ham) or non-relevant (spam) content. The main idea contribution of this work is to employ Harmony Search-based optimization techniques to fine-tune RBM parameters, as well as to evaluate their robustness in the context spam detection. The unsupervised learned features are then used to feed the Optimum-Path Forest classifier, being the original features extracted from e-mail content and compared against the new ones. The results have shown RBMs are suitable to learn features from e-mail data, since they obtained favorable results in the datasets considered in this work.

KEYWORDS

Spam Detection, Machine Learning, Restricted Boltzmann Machines, Optimum-Path Forest.

1. INTRODUCTION

The increasing volume of unsolicited messages sent daily, also known as spam, has fostered the design of reliable anti-spam filters. Spam messages have been of great concern in the last years, mainly due to the problems associated with their impact in our lives. Many companies have invested in machine learning research techniques in order to improve e-mail spam detection. The increasing workload in computer networks traffic, loss of productivity, improper use of offensive content and financial losses are among the most common shortcomings related to spam messages, just to name a few. Kurose & Ross (2012) pointed out other problems related to spam messages as well, such as illegal accesses to confidential information and black-market-oriented advertisements.

As aforementioned, a number of techniques can be employed to address the problem of spam detection. Silva, Yamakami & Almeida (2012), for instance, proposed different approaches based on intelligent techniques to detect spam messages by analyzing the impact of different sets of features on the accuracy of supervised classifiers. Jia, Li, Gao & Xia (2012) employed the well-known Support Vector Machines (SVMs) for web spam classification, and Behjat, Mustapha, Nezamabadi-pour, Sulaiman & Mustapha (2012) highlighted the importance of feature selection for spam detection using Genetic Algorithms and Artificial Neural Networks with Multilayer Perceptrons (ANN-MLP). Another significant study by Fernandes et al. (2015) introduced the Optimum-Path Forest (OPF) classifier (Papa, Falcão & Suzuki 2009, Papa, Falcão, Albuquerque & Tavares 2012) to the context of spam filtering in SMS messages, as well as OPF was compared against some state-of-the-art supervised pattern recognition techniques. The work was validated using two distinct supervised OPF techniques against SVMs, ANN-MLP and a k-nearest neighbors classifier in a recent developed dataset. The experiments showed promising results for both OPF-based classifiers, although SVMs have obtained the best recognition rates, but at the price of a high computational load.

Some works are oriented to features based on e-mail properties as well. Shams & Mercer (2013), for instance, proposed a novel spam classification approach that considers the frequency of words, html tags, and language-centric features, such as grammar and spell errors, among others. The experimental setup considered several supervised machine learning techniques and four benchmark datasets (CSDMC2010, Spam Assassin, Ling Spam, and Enron-Spam), obtaining excellent classification results.

Although we may have a number of supervised machine learning techniques out there with very interesting results in the context of spam detection, the scientific community has focused on deep learning techniques in the last years. Such sort of approaches are based on the hierarchical-working-mechanism of the human brain, which aims at extracting a different kind of information at each step of the knowledge perception process. Convolutional Neural Networks (LeCun, Bottou, Bengio & Haffner 1998), Restricted Boltzmann Machines (RBMs) (Ackley, Hinton & Sejnowski 1988, Hinton 2002) and Deep Belief Networks (DBNs) (Hinton, Osindero & Teh 2006) are among the most widely used approaches for deep learning-based applications, obtaining outstanding results in several important tasks, such as face and object detection, as well as speech recognition.

However, there have been a few works that deal with problem of spam detection using some sort of deep learning-based technique. Tzortzis & Likas (2007) presented a methodology for spam detection using DBNs, being their performance compared against Support Vector Machines in three widely used datasets (Enron 1, Spam Assassin and Ling Spam). The experiments showed the similarity or even better performance of DBNs when compared to SVMs. Very recently, Fiore, Palmieri, Castiglione & De Santis (2013) used a Discriminative Restricted Restricted Boltzmann Machine (DRBM) to address the problem of anomaly detection in computer networks. Although the results are promising, when the classifier is tested over a network with different properties of the one it has been trained, the classifier's performance may be degraded.

One of the main shortcomings related to RBMs concerns with their fine-tuning parameter task, which aims at selecting a suitable set of parameters in such a way the reconstruction error is minimized. As far as we know, there are a few works involving this subject only. One of the first studies that have attempted at modeling the problem of RBM model selection by means of meta-heuristic optimization techniques was conducted by Papa, Rosa, Marana, Scheirer & Cox (2015), which employed the well-known Harmony Search (HS) (Geem 2009) to fine-tune

DRBMs in the context of image reconstruction, and another study proposed by Papa, Rosa, Costa, Marana, Scheirer & Cox (2015) addressed the very same problem, but now with respect to Bernoulli RBMs. In both works, the authors concluded that HS-based approaches are suitable for such task, since they are usually faster than swarm-based meta-heuristic techniques. However, such approaches still suffer with their own parameter setting up, the so-called meta-optimization problem. Silva et al. (2015) proposed the use of RBM as a means to identify spam messages in e-mail messages. In this work, the authors used two well-known datasets to classify e-mail messages: SPAMBASE and LINGSPAM. In this case, an RBM was used to unsupervised learn the characteristics of the datasets. For such purpose, the authors used three techniques based on Harmonic Search for the refinement of the RBM parameters.

In this work, we employed a parameterless Harmony Search algorithm called Parameter-Setting Free Harmony Search (PSF-HS) (Geem & Sim 2010) to fine-tune RBM parameters, and we validate the robustness of the evaluated approach in the context of spam detection in two public datasets. In addition, we compare PSF-HS against vanilla HS and Improved Harmony Search (IHS) (Mahdavi, Fesanghary & Damangir 2007). Given a dictionary-based set of features, the main idea is to build a compact and representative set of features by means of an unsupervised fashion using RBM, and then to use the learned features to feed a supervised pattern recognition technique. Although we have used the Optimum-Path Forest (Papa, Falcão & Suzuki 2009, Papa, Falcão, Albuquerque & Tavares 2012) for such task, one can employ any other supervised pattern recognition technique. The remainder of this paper is organized as follows. Section 2 introduces the theory background about RBMs. In Section 3, we describe the OPF classifier. In the Section 4, we present the methodology, and Section 5 presents the experimental setup as well as it discusses the experiments. Finally, Section 6 states conclusions and future works.

2. RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann Machines are energy-based stochastic neural networks composed by two layers of neurons (visible and hidden), in which the learning phase is conducted by means of an unsupervised fashion. The RBM is similar to the classical Boltzmann Machine (Ackley et al. 1988), except that no connections between neurons of the same layer are allowed. Figure 1 depicts the architecture of a Restricted Boltzmann Machine, which comprises a visible layer **v** with m units and a hidden layer **h** with n units. The real-valued $m \times n$ matrix **W** models the weights between visible and hidden neurons, where w_{ij} stands for the weight between the visible unit v_i and the hidden unit h_j.

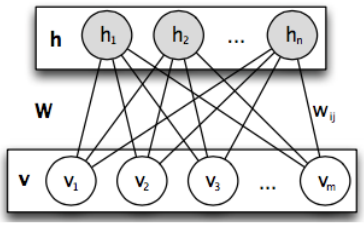

Figure 1. The RBM architecture

At first, RBMs were designed using only binary visible and hidden units, the so-called Bernoulli Restricted Boltzmann Machines (BRBMs). Later on, Welling, Rosen-zvi & Hinton (2005) shed light over other types of units that can be used in an RBM, such as Gaussian and binomial units, among others. Since in this paper we are interested in BRBMs, we will introduce their main concepts, which are the basis for other generalizations of RBMs.

Let us assume **v** and **h** as the binary visible and hidden units, respectively. In other words, $\mathbf{v} \in \{0, 1\}^m$ and $\mathbf{h} \in \{0, 1\}^n$. The energy function of a Bernoulli Restricted Boltzmann Machine is given by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i h_j w_{ij}, \tag{1}$$

where **a** and **b** stand for the biases of visible and hidden units, respectively. The probability of a configuration (**v, h**) is computed as follows:

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}, \tag{2}$$

where the denominator of above equation is a normalization factor that stands for all possible configurations involving the visible and hidden units[1]. In short, the BRBM learning algorithm aims at estimating **W**, **a** and **b**.

The parameters of a BRBM can be optimized by performing stochastic gradient ascent on the log-likelihood of training patterns. Given a training sample (visible unit), its probability is computed over all possible hidden vectors, as follows:

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}. \tag{3}$$

In order to update the weights and biases, it is necessary to compute the following derivatives:

$$\frac{\partial \log P(\mathbf{v})}{\partial w_{ij}} = E[h_j v_i]^{data} - E[h_j v_i]^{model}, \tag{4}$$

$$\frac{\partial \log P(\mathbf{v})}{\partial a_i} = v_i - E[v_i]^{model}, \tag{5}$$

$$\frac{\partial \log P(\mathbf{v})}{\partial b_j} = E[h_j]^{data} - E[h_j]^{model}, \tag{6}$$

[1] Note this normalization factor is extremely hard to be computed when the number of units is too large.

where $E[\cdot]$ stands for the expectation operation, and $E[\cdot]^{data}$ and $E[\cdot]^{model}$ correspond to the data-driven and the reconstructed-data-driven probabilities, respectively.

In practical terms, we can compute $E[h_j v_i]^{data}$ considering **h** and **v** as follows:

$$E[\mathbf{hv}]^{data} = P(\mathbf{h}|\mathbf{v})\mathbf{v}^T, \tag{7}$$

where $P(\mathbf{h}|\mathbf{v})$ stands for the probability of obtaining **h** given the visible vector (training data) **v**:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right), \tag{8}$$

where $\sigma(\cdot)$ stands for the logistic sigmoid function, notice that the logistic sigmoid function can be computed by the following equation: $\sigma(x) = 1/(1 + \exp(-x))$. Therefore, it is straightforward to compute $E[\mathbf{hv}]^{data}$: given a training data $x \in \chi$, where χ stands for a training set, we just need to set $\mathbf{v} \leftarrow \mathbf{x}$ and then employ Equation 8 to obtain $P(\mathbf{h}|\mathbf{v})$. Further, we use Equation 7 to finally obtain $E[\mathbf{hv}]^{data}$.

The big question now is how to obtain $E[\mathbf{hv}]^{model}$, which is the model learned by the system, notice that we are now writing $E[h_j v_i]^{model}$ in terms of **h** and **v**. One possible strategy is to perform alternating Gibbs sampling starting at any random state of the visible units until a certain convergence criterion, such as k steps, for instance. The Gibbs sampling consists of updating hidden units using Equation 8 followed by updating the visible units using $P(\mathbf{h}|\mathbf{v})$, given by:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right), \tag{9}$$

and then updating the hidden units once again using Equation 8. In short, it is possible to obtain an estimative of $E[\mathbf{hv}]^{model}$ by initializing the visible unit with random values and then performing Gibbs sampling, which may be time-consuming. Fortunately, Hinton (2002) introduced a faster methodology to compute $E[\mathbf{hv}]^{model}$ based on contrastive divergence. Basically, the idea is to initialize the visible units with a training sample, to compute the states of the hidden units using Equation 8, and then to compute the states of the visible unit (reconstruction step) using Equation 9. Roughly speaking, this is equivalent to perform Gibbs sampling using $k = 1$.

Based on the above assumption, we can now compute $E[\mathbf{hv}]^{model}$ as follows:

$$E[\mathbf{hv}]^{model} = P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T. \tag{10}$$

Therefore, the equation below leads to a simple learning rule for updating the weight matrix **W**, as follows:

$$\begin{aligned} \mathbf{W}^{t+1} &= \mathbf{W}^t + \eta(E[\mathbf{hv}]^{data} - E[\mathbf{hv}]^{model}) \\ &= \mathbf{W}^t + \eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T), \end{aligned} \tag{11}$$

where $\mathbf{W}^t$ stands for the weight matrix at time step t, and η corresponds to the learning rate. Additionally, we have the following formulae to update the biases of the visible and hidden units:

$$\begin{aligned} \mathbf{a}^{t+1} &= \mathbf{a}^t + \eta(\mathbf{v} - E[\mathbf{v}]^{model}) \\ &= \mathbf{a}^t + \eta(\mathbf{v} - \tilde{\mathbf{v}}), \end{aligned} \tag{12}$$

and

$$\begin{aligned} \mathbf{b}^{t+1} &= \mathbf{b}^t + \eta(E[\mathbf{h}]^{data} - E[\mathbf{h}]^{model}) \\ &= \mathbf{b}^t + \eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})), \end{aligned} \tag{13}$$

where $\mathbf{a}^t$ and $\mathbf{b}^t$ stand for the visible and hidden units biases at time step t, respectively. In short, Equations 11, 12 and 13 are the vanilla formulation for updating the RBM parameters.

Later on, Hinton (Hinton 2012) introduced a weight decay parameter λ, which penalizes weights with large magnitude, notice that the weights may increase during the convergence process, as well as a momentum parameter α to control possible oscillations during the learning process. Therefore, we can rewrite Equations 11, 12 and 13 as follows:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T) - \lambda\mathbf{W}^t + \alpha\Delta\mathbf{W}^{t-1}}_{=\Delta\mathbf{W}^t}, \tag{14}$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \underbrace{\eta(\mathbf{v} - \tilde{\mathbf{v}}) + \alpha\Delta\mathbf{a}^{t-1}}_{=\Delta\mathbf{a}^t} \tag{15}$$

and

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})) + \alpha\Delta\mathbf{b}^{t-1}}_{=\Delta\mathbf{b}^t}. \tag{16}$$

3. OPTIMUM-PATH FOREST CLASSIFICATION

Let $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ be a labeled dataset, such that $\mathcal{D}_1$ and $\mathcal{D}_2$ stands for the training and test sets, respectively. Let $S \subset \mathcal{D}_1$ be a set of prototypes of all classes (i.e., key samples that best represent the classes). Let $(\mathcal{D}_1, A)$ be a complete graph whose nodes are the samples in $\mathcal{D}1$ and any pair of samples defines an arc in A = $\mathcal{D}_1 \times \mathcal{D}_1$, as displayed in Figure 2a. Additionally, let π_s be a path in $(\mathcal{D}_1, A)$ with terminus at sample $s \in \mathcal{D}_1$.

The OPF algorithm proposed by Papa et al. (2009, 2012) employs the path-cost function $f_{\max}$ due to its theoretical properties for estimating prototypes (Section 3.1 gives further details about this procedure):

$$f_{max}(\langle s \rangle) = \begin{cases} 0 & \text{if } s \in S \\ +\infty & \text{otherwise,} \end{cases}$$
$$f_{max}(\pi_s \cdot \langle s,t \rangle) = \max\{f_{max}(\pi_s), d(s,t)\}, \quad (1)$$

where *d(s,t)* stands for a distance among nodes *s* and *t*, such that *s*, $t \in \mathcal{D}_1$. Therefore, $f_{\max}(\pi_s)$ computes the maximum distance between adjacent samples in π_s, when π_s is not a trivial path. In short, the OPF algorithm tries to minimize $f_{\max}(\pi_t)$, $\forall t \in \mathcal{D}_1$.

3.1 Training

We say that S^* is an optimum set of prototypes when Algorithm 1 minimizes the classification errors for every s $\in \mathcal{D}_1$. We have that S^* can be found by exploiting the theoretical relation between the minimum-spanning tree and the optimum-path tree for $f_{\max}$.

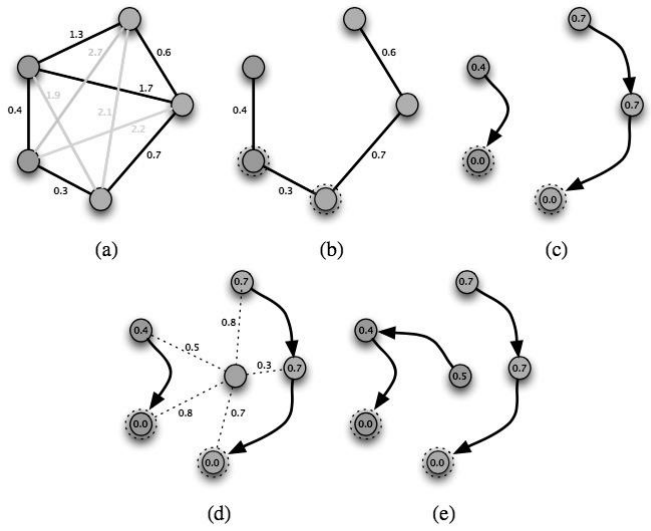

Figure (2). (a) Training set modeled as a complete graph, (b) a minimum spanning tree computation over the training set (prototypes are highlighted), (c) optimum-path forest over the training set, (d) classification process of a “green" sample, and (e) test sample is finally classified.

By computing an MST in the complete graph ($\mathcal{D}_1$, *A*) (Figure 2b), we obtain a connected acyclic graph whose nodes are all samples of $\mathcal{D}_1$ and the arcs are undirected and weighted by the distances *d* between adjacent samples. The spanning tree is optimum in the sense that the

sum of its arc weights is minimum as compared to any other spanning tree in the complete graph. In the MST, every pair of samples is connected by a single path, which is optimum according to $f_{\max}$. Hence, the minimum-spanning tree contains one optimum-path tree for any selected root node.

The optimum prototypes are the closest elements of the MST with different labels in $\mathcal{D}_1$ (i.e., elements that fall in the frontier of the classes, as displayed in Figure 2c). By removing the arcs between different classes, their adjacent samples become prototypes in S^*, and Algorithm 1 can define an optimum-path forest with minimum classification errors in $\mathcal{D}_1$ (Figure 2d). Algorithm 1 implements the above procedure for OPF training phase.

Algorithm 1. OPF Training Algorithm.

Input: A labeled training set D_1.
Output: Optimum-path forest P, cost map C,
label map L, and ordered set D'_1.
Auxiliary: Priority queue Q, set S of prototypes,
and cost variable *cst*.

1. Set $D'_1 \leftarrow 0$ and compute the prototype set by MST $S \subset D_1$.
2. For each $s \in D_1 \setminus S$, set $C(s) \leftarrow +\infty$.
3. For each $s \in S$, do
4. $C(s) \leftarrow 0$, $P(s) \leftarrow nil$, $L(s) \leftarrow \lambda(s)$, and insert s in Q.
5. While Q is not empty, do
6. Remove from Q a sample s such that $C(s)$ is minimum.
7. Insert s in D'_1.
8. For each $t \in D_1$ such that $C(t) > C(s)$, do
9. Compute $cst \leftarrow \max\{C(s), d(s,t)\}$.
10. If $cst < C(t)$, then
11. If $C(t) \neq +\infty$, then remove t from Q.
12. $P(t) \leftarrow s$, $L(t) \leftarrow L(s)$, $C(t) \leftarrow cst$.
13. Insert t in Q.
14. Return a classifier $[P,C,L,D'_1]$.

Lines 1 − 4 initialize maps and insert prototypes in Q (the function $\lambda(\cdot)$ in Line 4 assigns the true label to each training sample), notice that the cost map C stores the optimum-cost of each training sample.The main loop computes an optimum path from S to every sample s in a non-decreasing order of cost (Lines 5−13). At each iteration, a path of minimum cost $C(s)$ is obtained in P when we remove its last node s from Q (Line 6). Ties are broken in Q using first-in-first-out policy. That is, when two optimum paths reach an ambiguous sample s with the same minimum cost, s is assigned to the first path that reached it. Note that $C(t) > C(s)$ in Line 8 is false when t has been removed from Q and, therefore, $C(t) = +\infty$ in Line 11 is true only when $t \in Q$. Lines 10−13 evaluate if the path that reaches an adjacent node t through s is cheaper than the current path with terminus t, and update the position of t in Q, $C(t)$, $L(t)$ and $P(t)$ accordingly. The complexity for training OPF is done by $\theta(|\mathcal{D}_1|^2)$, due to the main (Lines 5-13) and inner loops (Lines 8-13) in Algorithm 1, which run $\theta(|\mathcal{D}_1|)$ times each.

3.2 Classification

For any sample $t \in \mathcal{D}_2$, we consider all arcs connecting t with samples $s \in \mathcal{D}_1$, as though t were part of the training graph (Figure 1d). Considering all possible paths from S^* to t, we find the optimum path $S^*(t)$ from S^* and label t with the class $\lambda(R(t))$ of its most strongly connected prototype $R(t) \in S^*$. This path can be identified incrementally, by evaluating the optimum cost

$$C(t) = \min\{\max\{C(s), d(s,t)\}\}, \ \forall s \in D_1 \qquad (2)$$

$C(t)$ as:

Let the node $s^* \in \mathcal{D}_1$ be the one that satisfies Equation 2 (i.e., the predecessor $P(t)$ in the optimum path $P^*(t)$. Given that $L(s^*) = \lambda(R(t))$, the classification simply assigns $L(s^*)$ as the class of t (Figure 1e). An error occurs when $L(s^*) \neq \lambda(t)$. Algorithm 2 implements this idea.

In *Algorithm* 2, the main loop (Lines 1 − 9) performs the classification of all nodes in $\mathcal{D}_2$. The inner loop (Lines 4 − 9) visits each node $k_{i+1} \in D'_1$, $i = 1,2,...,|Z'_1|-1$ until an optimum path $\pi_{ki+1} \cdot \langle k_{i+1}, t \rangle$ is found. In the worst case, the algorithm visits all nodes in D'_1. Line 5 evaluates $f_2(\pi_{ki+1} \cdot \langle k_{i+1}, t \rangle)$ and Lines 7 − 8 updates cost, label and predecessor of t whenever $\pi_{ki+1} \cdot \langle k_{i+1}, t \rangle$ is better than the current path π_t (Line 6). Although the reader can note the complexity of the OPF classification phase is given by $\theta(|\mathcal{D}_1| \mid \mathcal{D}_2|)$ (for each classification node we have to visit all training samples), Papa et al. (2012) showed that, in practice, the complexity is given by $O(p \mid \mathcal{D}_2|)$, in which $p \in O(|\mathcal{D}_1|)$.

Algorithm 2. OPF Classification Algorithm.

Input: Classifier $[P,C,L,D'_1]$ and test set D_2.
Output: Label L_0 and predecessor P_0 maps defined for D_2.
Auxiliary: Cost variables *tmp* and *mincost*.

1. For each $t \in D_2$, do
2. $i \leftarrow 1$, $mincost \leftarrow \max\{C(k_i), d(k_i,t)\}$.
3. $L_0(t) \leftarrow L(k_i)$ and $P_0(t) \leftarrow k_i$.
4. While $i < |D'_1|$ and $mincost > C(k_{i+1})$, do
5. Compute $tmp \leftarrow \max\{C(k_{i+1}), d(k_{i+1},t)\}$.
6. If $tmp < mincost$, then
7. $mincost \leftarrow tmp$.
8. $L_0(t) \leftarrow L(k_{i+1})$ and $P_0(t) \leftarrow k_{i+1}$.
9. $i \leftarrow i + 1$.
10. Return$[L_0,P_0]$.

4. MATERIALS AND METHODS

We propose to improve spam detection by learning features using Restricted Boltzmann Machines optimized through HS-based techniques. As aforementioned, the learning step has four parameters: the learning rate η, weight decay λ, penalty parameter α, and the number of

hidden units n. For the comparison of results, we define two experiments: EXP1 and EXP2, which aim at evaluating different ranges concerning the RBM parameters. In regard to EXP1, we have the following ranges: $\eta \in [0.0, 1.0]$, $\lambda \in [0.0, 1.0]$, $\alpha \in [0.0, 0.001]$ and $n \in [5, 15]$; and with respect to EXP2, we used the following ranges: $\eta \in [0.0, 1.0]$, $\lambda \in [0.0, 1.0]$, $\alpha \in [0.0, 0.001]$ and $n \in [30, 45]$.

In order to establish the accuracy, we performed a cross-validation procedure using 10 runnings, being 50% of the dataset used for training, and the remaining 50% employed for testing purposes. The next section describes the datasets employed in this work. Note we used an accuracy measure proposed by Papa et al. (2009).

4.1 Datasets

We used three public and well-known datasets considering the task of spam detection:

- SPAMBASE: this dataset is available in the UCI Machine Learning Repository, and it consists of 4,601 instances, being 39.4% samples classified as spam messages (each instance is represented by 57 attributes). The features represent the frequency of occurrence of each word based on a dictionary established by the authors of this work (Bache & Lichman 2013).
- LINGSPAM: this dataset contains spam and non-spam e-mail messages collected via a Linguist mailing list, including 2,893 e-mail messages out of which 2,412 are labeled as non-SPAM, and 481 are labeled as spam (Androutsopoulos, Koutsias, Chandrinos, Paliouras & Spyropoulos 2000).
- CSDMC: the dataset is composed of a selection of messages, divided in two parts, one for training and another for testing. The training set contains 4,327 messages, being 2,949 messages classified as ham, and 1,378 messages labeled as spam. (Group, 2010).

For the sake of simplicity, we used the very same dictionary of words (tokens) learned from SPAMBASE dataset for both LINGSPAM and CSDMC datasets. Note all samples are labeled as being ham or spam for all datasets. It is important to highlight the attributes were binarized between 0 and 1, since are using binary-valued RBMs.

4.2 Parameter Setting-up

In order to provide a more robust analysis of the results, we conducted two experiments, each one employing a cross-validation with 10 runnings. Notice that the ORIGINAL dataset is composed of 57 features. After learning features with RBM, the EXP1 comprised 10 features on average, and EXP2 contained 35 features on average.

Finally, we employed 10 agents over 50 iterations for convergence considering all techniques. Table 1 presents the parameter configuration for each optimization technique, notice these values have been empirically setup. We also have employed 100 epochs and mini-batches of size 100 for the RBM learning weights procedure by means of the Contrastive Divergence (HINTON, 2012). The Harmony Memory Considering Rate (HMCR) and the Pitch Adjusting Rate (PAR) avoid HS and IHS to get trapped from local optima, and ρ is used to adjust the step size of new solutions. Considering PSF-HS, we start with $HMCR = 0.7$.

Table 1. Parameter configuration.

Technique	Parameters
HS	$HMCR = 0.7$, $PAR = 0.7$, $\varrho = 0.1$
IHS	$HMCR = 0.7$, $PAR_{MIN} = 0.1$ $PAR_{MAX} = 1.0$, $\varrho_{MIN} = 0.1$ $\varrho_{MAX} = 0.5$

5. EXPERIMENTAL RESULTS

In this section, we describe the experiments conducted to assess the robustness of the proposed approach. As aforementioned, we first extracted the features for both datasets using a dictionary-based approach (bag-of-words), hereinafter called ORIGINAL. Soon after, we conducted experiments in order to learn a more discriminative set of features by means of RBMs optimized with HS (RBM-HS), IHS (RBM-IHS) and PSF-HS (RBM-PSF-HS). Therefore, we have four different versions of each dataset. The extracted/learned features are then used to feed the OPF classifier, thus performing a traditional pattern recognition pipeline, i.e., training, testing and accuracy computation.

Figures **3** and **4** display the experimental results considering SPAMBASE datasets for EXP1 and EXP2 rounds, respectively. From the former figure, we can realize RBM-based features are much more accurate than original ones, except for folds #2, #4, #6, #9 and #10, but even some of the RBM-based folds outperformed the original accuracy values. It is important to highlight that RBM-based reconstruction in EXP1 experiments used only 10 features on average, which is about to 5.7 times lower than the original. Although we observed similar behavior, EXP2 experiments (Figure 4) showed better results than ORIGINAL and EXP1 datasets, which this can be explained by the greater number of hidden neurons.

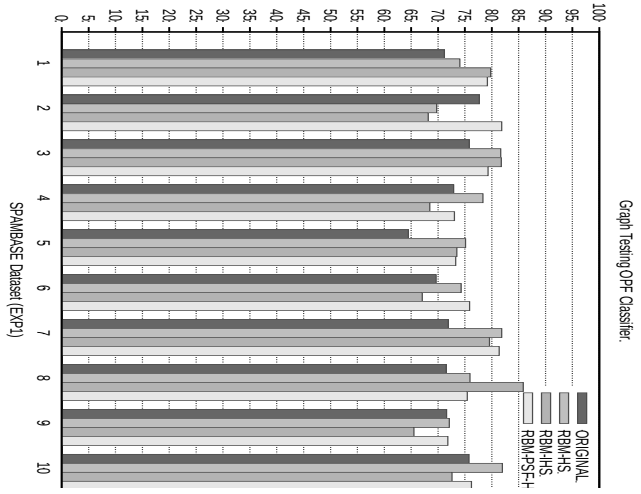

Figure 3. Experimental results over SPAMBASE dataset (EXP1).

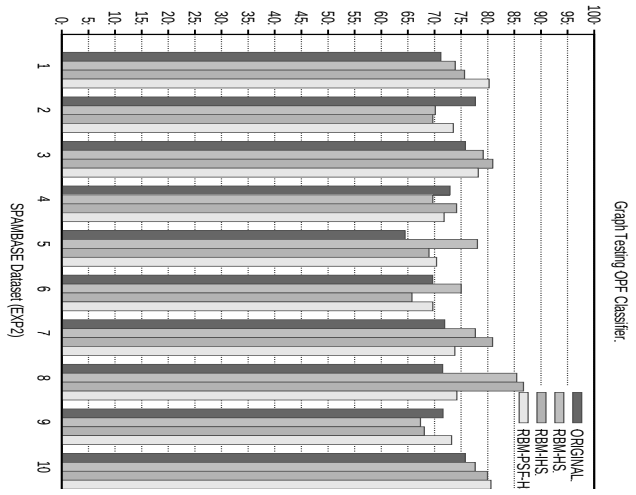

Figure 4. Experimental results over SPAMBASE dataset (EXP2).

Figure 5 and 6 depict the results of EXP1 and EXP2 experiments considering the dataset LINGSPAM, respectively. In regard the average accuracy over all folds for EXP1 and EXP2, we have observed neither experiments achieved values above 66%, which can be explained by the dictionary of tokens based on SPAMBASE. Probably, better results could be obtained with tokens extracted from LINGSPAM dataset. However, the main focus of this work is not to obtain the best results for these datasets, but to show we can improve the classification rates over traditional features by means of RBMs.

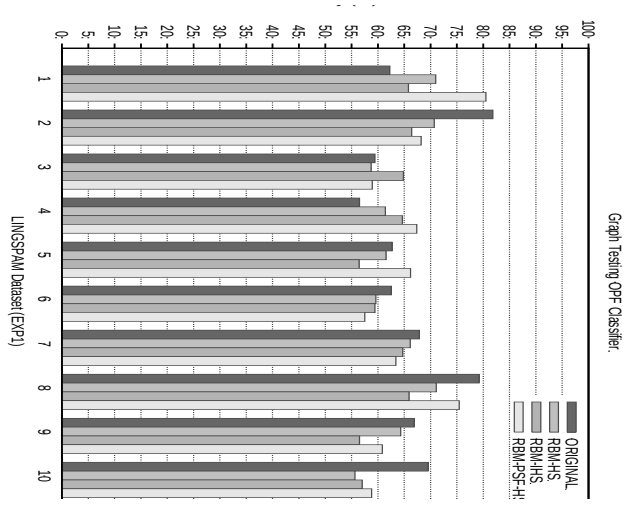

Figure 5. Experimental results over LINGSPAM dataset (EXP1).

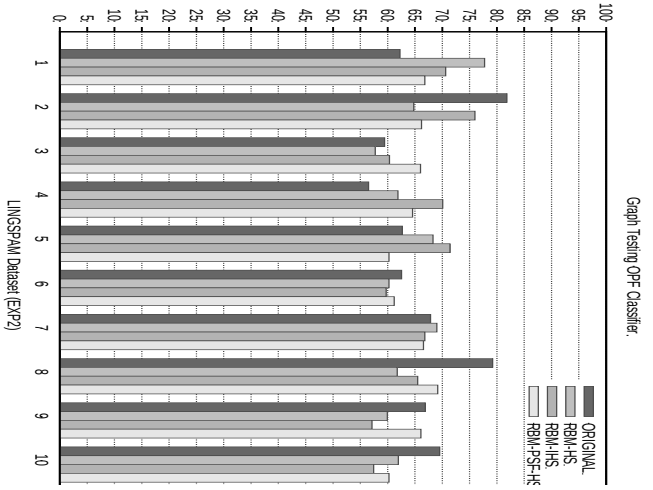

Figure 6. Experimental results over LINGSPAM dataset (EXP2).

In regard to the CSDMC dataset, according to Figures 7 and 8, we observed that none experiment (EXP1 and EXP2) outperformed the results of the ORIGINAL dataset. Therefore, an interesting behavior can be observed for all datasets concerning the quality of the learned features. Additionally, the number of features has been reduced as well. Considering RBM-based, for instance, we have used 10 features on average only, while ORIGINAL dataset was designed with 57 attributes. This means we are using about 5.7 times less features, and with better results for some folds.

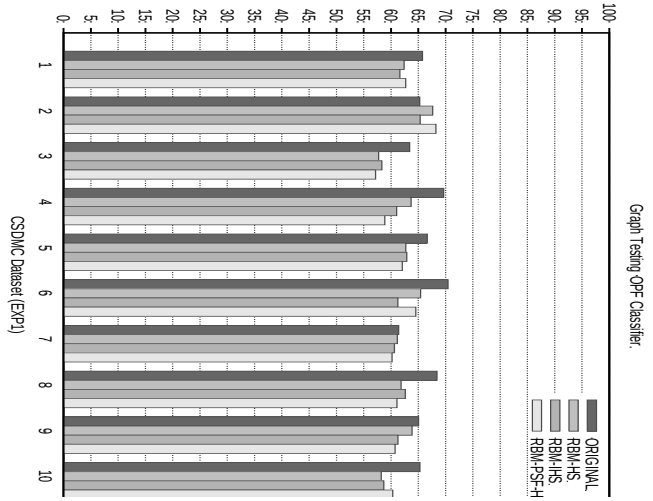

Figure 7. Experimental results over CSDMC dataset (EXP1).

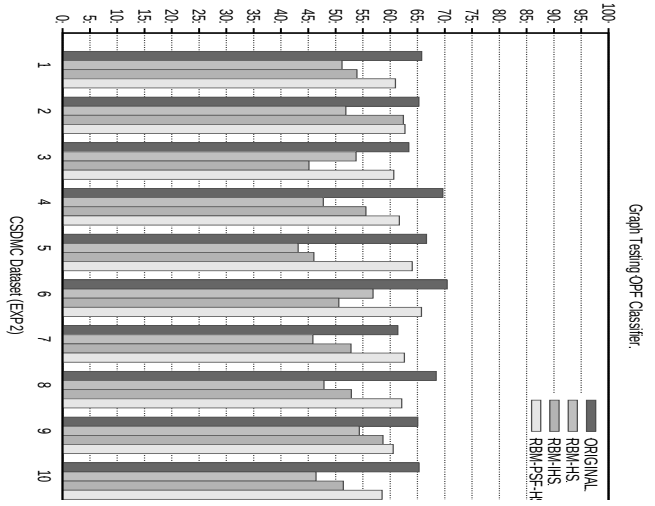

Figure 8. Experimental results over CSDMC dataset (EXP2).

6. CONCLUSION

In this work, we have studied the influence of a more discriminative and compact set of features learned by Restricted Boltzmann Machines, which have been optimized by HS-based techniques. The experiments have been conducted in the context of spam detection in three public datasets: SPAMBASE, LINGSPAM and CSDMC.

In natural language processing applications, words are modeled by an array of frequency distributions, where each dimension of this array represents one of the features in a dataset. The vocabulary size can be easily composed of a large number of words. Since this dimensionality impacts on the training time and computational resources, it is important to use tools as RBMs to learn features. In the experiments, we observed that is possible to achieve suitable results even using 5.7 times less features than the original dataset. The SPAMBASE dictionary used as a basis to LINGSPAM and CSDMC may not have seen a good choice, showing worst results than SPAMBASE dataset. The increase in the number of hidden neurons did not impact in better results with RBMs, reported that the number of hidden neurons does not have a direct dependence in the quality of the training samples. One can observe very promising results, since there are situations in which the new learned features can improve the original ones with a considerable advantage. In regard to future works, an idea for aim better results on the learn features based on RBMs is to eliminate features that are not relevant to the classification of samples, such as those having value 0 for all samples, regardless of whether labeled with ham or spam.

ACKNOWLEDGEMENT

The authors are grateful to FAPESP grants #2014/16250-9 and 2015/00801-9, as well as CNPq grants #470571/2013-6 and #306166/2014-3.

REFERENCES

Book

Kurose, J. F. & Ross, K. W., 2012. *Computer Networking: A Top-Down Approach*, Pearson Education, USA.

Journal

Ackley, D., Hinton, G. & Sejnowski, T. J., 1988. A learning algorithm for boltzmann machines, in D. Waltz & J. Feldman, eds, *Connectionist Models and Their Implications: Readings from Cognitive Science*, Ablex Publishing Corp., Norwood, NJ, USA, pp. 285–307.

Fiore, U., Palmieri, F., Castiglione, A. & De Santis, A., 2013. Network anomaly detection with the restricted boltzmann machine, *In Neurocomputing*, Vol.122, pp. 13–23.

Geem, Z. W., 2009. Music-Inspired Harmony Search Algorithm: Theory and Applications. *In Springer Publishing Company*, Incorporated.

Geem, Z. W. & Sim, K.-B., 2010. Parameter-setting-free harmony search algorithm. *In Applied Mathematics and Computation*, Vol. 8, pp. 3881 – 3889.

Hinton, G., 2002. Training products of experts by minimizing contrastive divergence. *In Neural Computation*, Vol. 14, pp. 1771–1800.

Hinton, G., 2012. A practical guide to training restricted boltzmann machines. *In Neural Networks: Tricks of the Trade*, Vol. 7700, pp. 599–619.

Hinton, G. E., Osindero, S. & Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. *In Neural Computation*, Vol. 18, pp. 1527–1554.

Mahdavi, M., Fesanghary, M. & Damangir, E., 2007. An improved harmony search algorithm for solving optimization problems. *In Applied Mathematics and Computation*, Vol. 188, pp. 1567 – 1579.

Papa, J. P., Falcão, A. X., Albuquerque, V. H. C. & Tavares, J. M. R. S., 2012. Efficient supervised optimum-path

forest classification for large datasets. *In Pattern Recognition*, Vol. 45, pp. 512–520.

Papa, J. P., Falcão, A. X. & Suzuki, C. T. N., 2009. Supervised pattern classification based on optimum-path forest. *In International Journal of Imaging Systems and Technology*, Vol. 19, pp.120–131.

Papa, J. P., Rosa, G. H., Marana, A. N., Scheirer, W. & Cox, D. D., 2015. Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques. *In Journal of Computational Science*,: http://www.sciencedirect.com/science/article/pii/S1877750315000526.

Welling, M., Rosen-zvi, M. & Hinton, G., 2005. Exponential family harmoniums with an application to information retrieval. *In Advances in Neural Information Processing Systems 17*, MIT Press, pp. 1481–1488.

Conference paper or contributed volume

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G. & Spyropoulos, C. D., 2000. An evaluation of naive bayesian anti-spam filtering. *Proceedings of the Workshop on Machine Learning in the New Information Age 11th European Conference on Machine Learning* (ECML 2000). Barcelona, Spain, pp. 9– 17.

Bache, K. & Lichman, M., 2013. UCI machine learning repository. http://archive.ics.uci.edu/ml.

Behjat, A., Mustapha, A., Nezamabadi-pour, H., Sulaiman, M. & Mustapha, N., 2012. GA-based feature subset selection in a spam/non-spam detection system. *International Conference on Computer and Communication Engineering*. pp. 675–679.

C. Group., 2010. *Spam email datasets, CSDMC2010 SPAM corpus.* Retrieved: March, 2011. Available at: http://csmining.org/index.php/spam-email-datasets-.html.

Fernandes, D., Costa, K. A. P., Almeida, T., Papa, J. P., 2015. SMS spam identification through optimum-path forest-based classifiers. *14th International Conference on Machine Learning and Applications (IEEE ICMLA'15)*. Miami, FL, USA.

Hinton, G. 2012. *A practical guide to training restricted boltzmann machines*. Neural Networks: Tricks of the Trade, p. 599–619.

Jia, Z., Li, W., Gao, W. & Xia, Y., 2012. Research on web spam detection based on support vector machine. *International Conference on Communication Systems and Network Technologies*. pp. 517–520.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. pp. 2278–2324.

Papa, J. P., Rosa, G. H., Costa, K. A. P., Marana, A. N., Scheirer, W. & Cox, D. D., 2015. On the model selection of bernoulli restricted boltzmann machines through harmony search. *Proceedings of the Genetic and Evolutionary Computation Conference*.

Shams, R. & Mercer, R., 2013. Classifying spam emails using text and readability features. *IEEE 13th International Conference on Data Mining*. pp. 657–666.

Silva, L. A.; Costa, K. A. P.; Ribeiro, P. B.; Rosa, G.; Papa, J. P., 2015. Parameter-setting free harmony search optimization of restricted boltzmann machines and its applications to spam detection. *12th International Conference Applied Computing*. Dublin, Ireland, pp. 142–150.

Silva, R. M., Yamakami, A. & Almeida, T. A., 2012. An analysis of machine learning methods for spam host detection. *11th International Conference on Machine Learning and Applications*. Washington, DC, USA, pp. 227–232.

Tzortzis, G. & Likas, A., 2007. Deep belief networks for spam filtering. *19th IEEE International Conference on Tools with Artificial Intelligence*. Vol. 2, pp. 306–309.