Análise de desempenho de redes neurais artificiais para classificação automática de *web spam*

Renato M. Silva [1]
Tiago A. Almeida [2]
Akebo Yamakami [1]

Resumo: Com o crescente aumento do volume de informações disponíveis na web, as ferramentas de busca tornam-se cada vez mais necessárias no dia a dia dos usuários da internet. Entretanto, pessoas mal-intencionadas veem esse fenômeno como uma oportunidade para obter lucro e, como consequência, um problema conhecido como *web spam* vem se tornando cada vez mais frequente na vida dos usuários da internet, provocando prejuízos pessoais e econômicos. Diversas técnicas vêm sendo propostas para detecção automática de *web spam*, porém, a alta capacidade de aperfeiçoamento dos mecanismos empregados pelos *spammers* exige que os métodos de classificação sejam cada vez mais genéricos e eficientes. Técnicas bastante conhecidas que possuem tais características são as redes neurais artificiais. Diante desse cenário desafiador, estse trabalho apresenta uma análise de desempenho de redes neurais artificiais perceptron de múltiplas camadas no combate de tal problema.

***Palavras-chave*:** Aprendizagem de máquina. Redes neurais artificiais. Web spam.

Abstract: *Due to the increasing volume of information available on the web, search engines become increasingly necessary in day to day of Internet users. However, people with bad intention see this phenomenon as an opportunity to get profit and, consequently, a problem known as web spam is becoming increasingly common in the lives of Internet users, causing personal and economic losses. Fortunately, some methods have been proposed in the literature for automatic detection of this plague. However, the constant improvement of techniques used by spammers requires that the filtering approaches be more generic, efficient and with high capacity of adaptation. Well known techniques that have such characteristics are the artificial neural networks. Given this scenario, this paper presents a performance evaluation of multi-layer Perceptron artificial neural networks employed to solve such problem.*

***Keywords*:** *Artificial neural networks. Machine learning. Web spam.*

1 Introdução

Junto com o constante aumento do número de usuários, a *web* também vem crescendo de maneira impressionante e está se tornando cada vez mais importante na vida das pessoas que a utilizam. Tal crescimento, aliado ao consequente aumento no volume de informações disponíveis, faz aumentar a importância dos motores de busca, que são ferramentas que ajudam os usuários a encontrar as informações desejadas, visando apresentar os resultados de forma organizada, rápida e eficiente. Porém, existem métodos mal-intencionados que tentam burlar os mecanismos de busca manipulando o *ranking* de relevância das páginas apresentadas, degradando a eficiência dessas ferramentas. Esses métodos induzem os algoritmos de busca a classificar algumas páginas com maior relevância do que realmente têm, o que deteriora os resultados e frustra os usuários, além de os expor a conteúdo inadequado e inseguro. Essa técnica enganosa é conhecida como *web spamming* [35].

[1]Faculdade de Engenharia Elétrica e de Computação, Unicamp, Av. Albert Einstein, 400 - Barão Geraldo - 13083-852, Campinas/SP - Brasil
{renatoms,akebo@dt.fee.unicamp.br}
[2]Departamento de Computação, UFSCar, Rod. João Leme dos Santos, Km 110 - SP-264, Itinga - 18052-780, Sorocaba/SP - Brasil
{talmeida@ufscar.br}

http://dx.doi.org/10.5335/rbca.2012.2195

Web spam pode conter conteúdo *spam* e/ou *spam links* [19, 33, 34]. Segundo Gyongyi and Garcia-Molina [19], um exemplo de conteúdo *spam* é uma página de pornografia que adiciona milhares de palavras-chave em sua página inicial sobre assuntos que não têm ligação com conteúdo pornográfico e deixa essas palavras invisíveis por meio de recursos das linguagens de programação para *web*. Assim, quando uma pessoa realiza uma pesquisa usando um dos termos presentes nessas palavras-chave, os motores de busca podem retornar como resultado essa página de pornografia, o que na maioria das vezes não atende os anseios do usuário. Shen et al. [31] explicam que a técnica de *spam links* é um tipo de *web spamming* no qual os *spammers* adicionam milhares de *links* para as páginas que pretendem promover. Essa técnica eleva o *ranking* das páginas em motores de busca que classificam a relevância de uma página usando a relação da quantidade de *links* que apontam para ela. Outra técnica, menos popular, chamada *cloaking* (camuflagem), consiste em levar a que o conteúdo apresentado aos *web crawlers* (também conhecidos como *bots* ou robôs – programas responsáveis pela indexação das páginas *web*) seja diferente do conteúdo mostrado ao usuário. Dessa forma, para os usuários é apresentado um *web spam*, ao passo que para os *web crawlers* é apresentada uma página *web* normal, sem vestígios de *spam* [19].

Assim como *spam* por e-mail, *web spamming* vem se tornando um transtorno cada vez maior para os usuários da internet. Um dos menores problemas causados por essa praga virtual diz respeito ao incômodo causado pelo forjamento de respostas não merecidas e inesperadas promovendo o anúncio de páginas indesejadas. Além disso, usuários que estão em busca de informações acabam gastando mais tempo para encontrar aquilo que desejam, logo, sofrem prejuízos pessoais e econômicos em razão de perda de produtividade [31]. Porém, existem problemas ainda mais graves que oferecem uma grande ameaça aos usuários, uma vez que *web spam* pode conter conteúdos maliciosos que instalam *malwares* nos computadores das vítimas, facilitando o roubo de senhas bancárias e de informações pessoais, degradando o desempenho dos computadores e da rede, entre outros [13].

Segundo relatórios anuais, o volume de *web spam* está aumentando de forma assustadora. Eiron et al. [14] classificaram cem milhões de páginas da internet e verificaram que, em média, 11 dos vinte melhores resultados eram páginas pornográficas que obtiveram um elevado índice de relevância por meio da manipulação de *links*. Segundo Gyongyi and Garcia-Molina [19], em 2002 havia cerca de 6 a 8% de *web spam* nos resultados de pesquisa dos motores de busca e em 2003 e 2004 esse valor saltou para 13 e 18%, respectivamente. Outro estudo aponta que cerca de 9% dos resultados de pesquisa contêm pelo menos um *link* de uma página de *spam* entre os dez melhores resultados, ao passo que 68% de todas as consultas contêm algum tipo de *spam* nos quatro melhores resultados apresentados pelos motores de busca [19]. Segundo estimativas realizadas pela ferramenta de busca *Blekko*, que possui um "contador de *spams*" *on-line*, a cada hora são criados mais de um milhão de *web spam* e, portanto, entre 1º de janeiro de 2011 e 07 de agosto de 2012 foram criados cerca de 14 billões.[3]

O principal motivo que fomenta o crescimento do volume de *web spam* é o incentivo econômico a essa prática. Muitos preferem investir em técnicas antiéticas para melhorar o índice de relevância dos seus *sites*, pois, além do retorno ser visivelmente mais rápido, o custo é bem mais baixo do que investir em técnicas de melhoria visual e criação de conteúdo que motivem a visita dos usuários, tornando-os mais acessíveis e interessantes. Outros, por sua vez, contratam profissionais sem ter o conhecimento de que estes utilizam métodos de criação de *spam*. Além disso, muitos *spammers* lucram diretamente por meio da participação em programas de afiliados. Um exemplo característico é aquele em que o *spammer* cria uma página com conteúdo mínimo, ou mesmo sem conteúdo, fornecendo um *link* para um produto de um determinado *site* de compras. Dessa forma, utilizando técnicas de *spamming*, consegue aumentar o índice de relevância do *site* e, quando um usuário clica no anúncio, o *spammer* recebe dinheiro por isso. Às vezes o valor recebido por clique é ínfimo, porém o altíssimo volume de cliques torna o montante atrativo [19].

Para combater esse problema, diversos métodos vêm sendo propostos na literatura, sendo que alguns analisam apenas *spam links* [16, 31], outros fazem somente análise de conteúdo [26, 36] e, ainda, aqueles que analisam tanto os *links* quanto o conteúdo [1, 11, 28]. Entre tais propostas, as que vêm obtendo maior sucesso são as técnicas de aprendizado de máquina, tais como seleção de conjuntos (*ensemble selection*) [15, 17], *clustering* [11, 22], floresta aleatória [15], *boosting* [15, 17], máquinas de vetores de suporte [29, 35] e árvores de decisão [11, 16].

Apesar da existência desse conjunto de métodos, ainda há características do problema que precisam ser levadas em consideração, dentre os quais está o grande crescimento e evolução de técnicas de *web spam* empregadas especificamente para burlar as técnicas de classificação. Logo, é necessário que métodos mais genéricos sejam

[3]Consulte http://www.spamclock.com/

utilizados para possibilitar o acompanhamento da evolução dessas técnicas, visando manter o bom desempenho ao longo do tempo.

Sem razão aparente, existem poucos trabalhos na literatura que empregam técnicas de redes neurais artificiais (RNAs) para a classificação de *web spam*, apesar de estarem entre as técnicas de reconhecimento de padrões mais conhecidas e bem sucedidas. Diante disso, o objetivo deste trabalho é apresentar uma análise do desempenho obtido por essas técnicas, mais especificamente por meio da rede neural artificial perceptron de múltiplas camadas (MLP – multilayer perceptron) usando o algoritmo de treinamento de primeira ordem *backpropagation* com gradiente descendente e o algoritmo de segunda ordem Levenberg-Marquardt.

Este artigo está estruturado da seguinte maneira: na seção 2 são apresentados alguns trabalhos correlatos que abordam diferentes técnicas empregadas para o combate de *web spam*. As principais definições e características das redes neurais artificiais avaliadas neste trabalho são oferecidas na seção 3. Na seção 4 são detalhados os experimentos realizados, a base de dados empregada, os vetores de características, as medidas de desempenho e os resultados obtidos. Finalmente, conclusões e direções para pesquisas futuras são descritas na seção 5.

2 Revisão bibliográfica

Ao contrário da extensiva quantidade de trabalhos presentes na literatura que abordam o problema de classificação de *spam* por e-mail [2–8, 12, 18, 37], existem poucas pesquisas publicadas a respeito de técnicas para combater *web spam*, uma vez que tal problema ainda é bastante recente.

Em Leon-Suematsu et al. [24], é apresentado um algoritmo de detecção de *web spam* que se baseia na análise de *links*. A primeira etapa do método consiste na decomposição de webgrafos em subgrafos densamente conectados e no cálculo das *features* de cada subgrafo. Em seguida, é usado um classificador SVM para identificar os subgrafos que são compostos por *web spam*. Então, foi feita a propagação das predições sobre webgrafos por um algoritmo de *PageRank* para expandir o escopo da identificação. Para isso, os *hosts*, identificados como *spam* foram usados como sementes para identificar outros *hosts*, já que os nós de *spams* tendem a estar ligados a outros nós de *spams*. De acordo com os autores, o método produz bons resultados e o principal fator que colabora para isso é o uso da estrutura de *links* dos webgrafos.

Já Shengen et al. [32] propõem criar novas características para a identificação de *web spam* por meio de programação genética usando *features* baseadas em *links* que já existem e são usadas em outros trabalhos da literatura. Também foram apresentados os efeitos do número de indivíduos, do número de gerações e da profundidade da árvore binária nos resultados do algoritmo. As novas *features* foram usadas em um classificador SVM e em um classificador baseado em programação genética. Segundo os autores, os classificadores que usam as novas *features* geradas em seu trabalho têm resultados muito melhores se comparados com as *features* fornecidas na base de dados que eles usaram para os experimentos.

No trabalho de Zhang et al. [38] é proposto um algoritmo de aprendizagem para detecção de *web spam* chamado *Harmonic functions based semi-supervised learning*(HFSSL). No método proposto, as páginas web, rotuladas ou não, são representadas como vértices de um grafo ponderado e o problema de aprendizagem é modelado como um campo Gaussiano aleatório nesse gráfico. A média desse campo é caracterizada por funções harmônicas que possuem uma forma fechada e podem ser obtidas por meio de manipulação matricial. Os resultados indicam que uma das vantagens do método proposto é que ele permite a utilização de uma grande quantidade de dados sem rótulos, diferente da maioria dos métodos usados para a detecção de *web spam*. Essa característica é importante, pois a rotulagem das páginas web como *spam* ou não *spam* é um trabalho demorado e que exige experiência quando é feito por humanos. Além disso, os resultados dos experimentos mostraram que o algoritmo é eficaz e melhor do que outros métodos de aprendizagem semissupervisionada.

Em Largillier and Peyronnet [23] considera-se que os *spammers* usam páginas web com arquiteturas específicas dedicadas ao aumento do *PageRank* em torno de uma determinada página de destino. Dessa forma, a sua pontuação é maximizada e, ao mesmo tempo, evita-se a sua detecção automática. Para contornar esse impasse, os autores propuseram uma técnica para identificação de *web spam* que aborda *spam links*, analisando a estrutura de ligação de páginas *spams*. Os resultados indicam que a técnica proposta é eficaz, pois foi capaz de identificar *spams* usando padrões simples, além de ser capaz de suportar pequenas alterações na estrutura.

Em Rungsawang et al. [30] são explorados tanto o conteúdo *spam* quanto os *links* para a criação de um método de detecção de *spam hosts*. O modelo desenvolvido pelos autores é baseado no algoritmo de otimização por colônia de formigas, usando heurísticas padrões e funções básicas de atualização do feromônio. Os resultados obtidos indicam que o método adotado é melhor que os classificadores baseados no método de árvores de decisão e SVM.

No trabalho de Castilho et al. [11] é apresentado um sistema de detecção de *web spam* que combina *features* baseadas em links e *features* baseadas em conteúdo. Além disso, é usada uma topologia de grafo modelado através da exploração da dependência de *links* entre as páginas web. Segundo os autores, os *hosts* que possuem conexões entre si tendem a pertencer à mesma classe. Além disso, foi proposto no trabalho um conjunto de *features* que podem ser extraídas em relação ao conteúdo das páginas e que são utilizadas por outros autores em outros trabalhos e nos eventos *Web Spam Challenge Track* I e II.

Em Svore et al. [35] é apresentado um método de detecção de *web spam* que emprega *features* baseadas em conteúdo e que consideram o tempo de *rank*. Os experimentos foram realizados usando um classificador SVM linear com *features* que consideram o tempo de *rank* com e sem dependência de consultas. Os resultados indicam que o primeiro método tem um desempenho melhor que o segundo.

3 Redes neurais artificiais

Redes neurais artificiais (RNAs) são sistemas paralelos distribuídos e constituídos de unidades de processamento simples, chamadas "neurônios", que têm capacidade computacional relacionada à aprendizagem e á generalização. Nesse sistema, o conhecimento é adquirido por um processo chamado "treinamento"ou "aprendizagem"que fica armazenado em forças de conexões entre os neurônios, chamadas pesos sinápticos [21].

Segundo Braga et al. [10], as RNAs são capazes de aprender através de um conjunto reduzido de exemplos e depois generalizar o conhecimento adquirido, sendo capaz de dar respostas coerentes para dados desconhecidos. Além disso, também são capazes de extrair informações não apresentadas de forma explícita e possuem grande capacidade de auto-organização.

Um modelo básico de RNA possui os seguintes componentes [10]:

- *conjunto de sinapses:* conexões entre os neurônios da RNA. Cada uma delas possui um peso sináptico;
- *integrador:* realiza a soma dos sinais de entrada da RNA, ponderados pelos pesos sinápticos;
- *função de ativação:* restringe a amplitude do valor de saída de um neurônio;
- *bias:* valor aplicado externamente a cada neurônio e tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação.

Existem diversos tipos de funções de ativação, sendo que as mais populares são apresentadas a seguir [21].

- *Função limiar ou degrau:* normalmente restringe a saída da RNA em valores binários [0,1] ou bipolares [-1,1]. Logo, pode ser representada por

$$\varphi(u) = \begin{cases} 1 & \text{se} \quad u \geq 0 \\ 0 & \text{se} \quad u < 0 \end{cases} \tag{1}$$

- *Função sigmoidal:* trata-se da função mais comum. É definida como uma função crescente com balanceamento adequado entre o comportamento linear e não linear e assume um intervalo de variação entre 0 e 1 e é dada por

$$\varphi(u) = \frac{1}{1 + \exp(-au)} \tag{2}$$

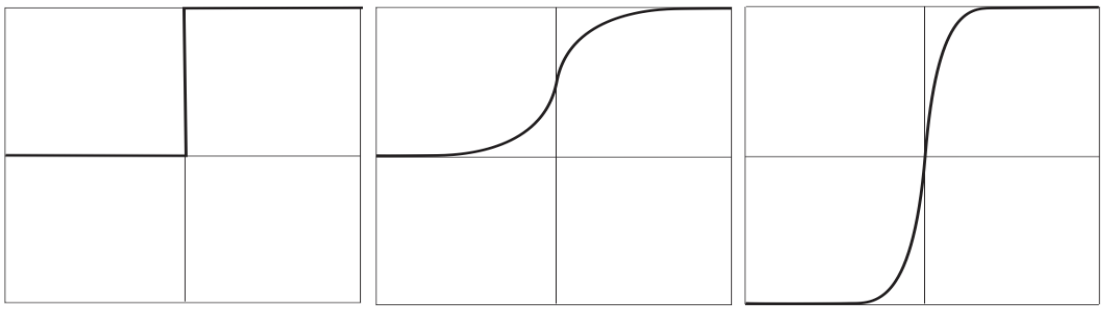
sendo a o parâmetro de inclinação da função.

- *Função tangente hiperbólica:* assume um intervalo de variação de -1 a 1 e é definida por:

$$\varphi(u) = \tanh(u) \tag{3}$$

Segundo Haykin [21], a característica da função tangente hiperbólica de assumir valores negativos traz benefícios analíticos.

O comportamento de cada função de ativação é apresentado na Figura 1.



(a) Degrau (b) Sigmoidal (c) Tangente hiperbólica

Figura 1: Funções de ativação degrau, sigmoidal e tangente hiperbólica, respectivamente

Outro aspecto que pode diferenciar duas RNAs é quanto a forma de aprendizagem, que pode usar os seguintes paradigmas [9]:

- *aprendizado supervisionado:* a saída desejada em relação a um padrão de entrada é fornecida para a RNA por um supervisor externo. Dessa forma, a saída da RNA é comparada com a saída desejada, recebendo informações sobre o erro obtido pela resposta atual. Diante disso, os pesos sinápticos são ajustados de forma a minimizar o erro;
- *aprendizado não supervisionado:* nessa categoria não existe um supervisor para acompanhar o processo de aprendizagem. Portanto, a RNA deve procurar algum tipo de correlação ou redundância nos dados de entrada.

Existem diversos tipos de RNAs, tais como perceptron simples, Adaline, perceptron de múltiplas camadas (*multi-layer Perceptron – MLP*), mapas auto-organizáveis de Kohonen, redes de Hopfield, redes recorrentes, entre outras. Dentre estas, uma das RNAs que vêm obtendo maior sucesso na solução de problemas de classificação e reconhecimento de padrões é a MLP [10, 21], empregada neste trabalho.

3.1 Rede neural artificial perceptron de múltiplas camadas (*multi-layer Perceptron – MLP*)

Uma rede neural artificial perceptron de múltiplas camadas (MLP – multilayer perceptron) é uma RNA do tipo perceptron que possui um conjunto de unidades sensoriais que formam a camada de entrada, uma ou mais camadas ocultas ou intermediárias de neurônios computacionais e uma camada de saída. Esse tipo de RNA é totalmente conectada, ou seja, todos os neurônios de qualquer camada estão conectados a todos os neurônios da camada anterior [21].

Por padrão, o treinamento de uma MLP é do tipo supervisionado por meio de algoritmo *backpropagation* (retropropagação do erro), que tem a função de encontrar as derivadas da função de erro com relação aos pesos e bias da RNA. A função de erro calcula a diferença entre a saída fornecida pela RNA e a saída desejada em relação a um determinado padrão de entrada [9].

O algoritmo *backpropagation* é constituído de duas etapas. Na primeira, chamada *forward*, um padrão de entrada é apresentado à RNA e seu efeito é propagado, camada a camada, até produzir um conjunto de saídas,

que correspondem a resposta em relação ao padrão de entrada. Na segunda etapa, conhecida como *backward*, são calculadas as derivadas da função de erro com respeito aos pesos da RNA. O sinal do erro é propagado da camada de saída para a camada de entrada. Tal sinal corresponde à diferença entre a saída real na etapa *forward* em relação a saída desejada [21]. Também é importante destacar que, segundo Bishop [9], na etapa *forward* os pesos sinápticos são mantidos fixos, ao passo que na *backward* os pesos são ajustados de acordo com algum método de otimização, sendo que um dos mais populares é o método do gradiente descendente [10]. O ajuste é realizado para que a saída da RNA se aproxime da saída desejada, diminuindo o valor do sinal de erro.

O algoritmo *backpropagation* pode ser resumido nos seguintes passos [21, 25]:

1. *inicialização:* nessa etapa os pesos e o bias da RNA devem ser inicializados. Uma das formas de realizar esse procedimento é usar aleatoriamente valores de uma distribuição uniforme que esteja no intervalo entre 0 e 1, considerando que a função de ativação a ser utilizada será a função sigmoidal;

2. *apresentação dos padrões de treinamento:* um conjunto de padrões de treinamento (época) deve ser apresentado à RNA. Para cada padrão são realizados os passos *forward* e *backward*;

3. *etapa forward:* suponha que um exemplo de treinamento seja representado por $(x(n), d(n))$, onde $x(n)$ é um padrão de entrada e $d(n)$ é a saída desejada. Então, o sinal é propagado pela RNA, camada a camada da seguinte forma:

$$u_j^l(n) = \sum_{i=0}^{m^{l-1}} w_{ji}^l(n) y_i^{l-1}(n) \tag{4}$$

sendo $l = 0, 1, 2, ..., L$ o índice das camadas da RNA. Quando $l = 0$, representa a camada de entrada e $l = L$, a camada de saída. Já, $y_i^{l-1}(n)$ é a função de saída do neurônio i na camada anterior $l-1$, $w_{ji}^l(n)$ é o peso sináptico do neurônio j na camada l e m^l é a quantidade de neurônios na camada l. Para $i = 0$, $y_0^{l-1}(n) = +1$ e $w_{j0}^l(n)$ representa o bias aplicado ao neurônio j da camada l.

A saída do neurônio j na camada l é dada por

$$y_j^l(n) = \varphi_j(u_j^l(n)) \tag{5}$$

onde φ_j é a função de ativação do neurônio j. Agora, o erro deve ser calculado da seguinte forma:

$$e_j^l(n) = y_j^l(n) - d(n) \tag{6}$$

sendo que $d(n)$ é a saída desejada para o padrão de entrada $x(n)$;

4. *etapa backward:* nessa etapa, inicia-se a derivação do algoritmo *backpropagation*, a partir da camada de saída, onde tem-se:

$$\delta_j^L(n) = \varphi_j'(u_j^L(n)) e_j^L(n) \tag{7}$$

sendo φ_j' a derivada da função de ativação. Para $l = L, L-1, ..., 2$, calcula-se:

$$\delta_j^{l-1}(n) = \varphi_j'(u_j^{l-1}(n)) \sum_{i=1}^{m^l} w_{ji}^l(n) * \delta_j^l(n) \tag{8}$$

para $j = 0, 1, ..., m^l - 1$.

Agora os pesos sinápticos podem ser atualizados da seguinte forma:

$$w_{j,i}^l(n+1) = w_{j,i}^l(n) - \alpha \delta_j^{l-1}(n) y_j^L(n) \tag{9}$$

onde α é o parâmetro que define o tamanho do passo de aprendizagem;

5. repetição dos passos 3 e 4 para todos os padrões de treinamento, completando uma época de treinamento;

6. retorno ao passo 3 e repetição até que o critério de parada seja satisfeito.

Segundo Haykin [21], um dos critérios de paradas mais comuns para a MLP e outros tipos de RNAs é o número de épocas de treinamento. A execução do algoritmo também pode ser interrompida quando a RNA atingir um valor mínimo de erro, que pode ser calculado pela seguinte equação:

$$EQM = \frac{1}{n} e^T e, \tag{10}$$

sendo n o número de padrões de entrada e e um vetor que armazena o erro relativo a todos os padrões de entrada da RNA. O valor encontrado por essa equação é chamado de "erro quadrático médio"(EQM).

Outro critério de parada é a validação cruzada. De acordo com Bishop [9] e Braga et al. [10], essa técnica evita a ocorrência de *overfitting*, estimando o erro de generalização da RNA durante o processo de treinamento. O *overfitting* ocorre quando a RNA memoriza os padrões de treinamento em vez de extrair características gerais para ser eficiente na generalização dos padrões, ou seja, para acertar padrões que a RNA não viu durante o processo de treinamento. Dessa forma, na validação cruzada, o conjunto de padrões é dividido de forma aleatória em conjunto de treinamento e conjunto de validação. O primeiro é utilizado para treinar a RNA e o segundo é usado para calcular a capacidade de generalização da RNA durante o processo de aprendizagem. Logo, o processo de treinamento é interrompido quando o erro do conjunto de validação começar a aumentar. Tal erro também pode ser calculado pela Equação 10.

3.1.1 Algoritmo de Levenberg-Marquardt

O algoritmo Levenberg-Marquardt é um método de otimização e aceleração da convergência do algoritmo *backpropagation*, sendo mais poderoso do que a técnica convencional do gradiente descendente [9]. É considerado um método de segunda ordem, assim como os métodos do gradiente conjugado e do método quase-Newton, pois utiliza informações sobre a derivada segunda da função de erro. Além disso, assim como o método quase-Newton, o algoritmo de Levenberg-Marquardt pode tornar-se computacionalmente impraticável em RNAs muito grandes. Porém, é mais eficiente do que método do gradiente conjugado em RNAs com algumas centenas de pesos e possui menor custo computacional do que o método quase-Newton [20].

Segundo Liu [25], o algoritmo de Levenberg-Marquardt propõe uma solução de compromisso entre o algoritmo do gradiente descendente e o método iterativo de Gauss-Newton, incorporando a velocidade de convergência de uma RNA de segunda ordem.

Considerando que a função de erro usada na MLP é dada pelo EQM (Equação 10), a equação usada no método de Gauss-Newton para atualização dos pesos da RNA e consequente minimização do valor do EQM é:

$$W_{i+1} = W_1 - H^{-1} \nabla f(W) \tag{11}$$

O gradiente $\nabla f(W)$ pode ser representado por

$$\nabla f(W) = J^T e \tag{12}$$

e a matriz Hessiana pode ser calculada por

$$\nabla^2 f(W) = J^T J + S \tag{13}$$

onde J é a matriz Jacobiana

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial x_1} & \frac{\partial e_1}{\partial x_2} & \cdots & \frac{\partial e_1}{\partial x_n} \\ \frac{\partial e_2}{\partial x_1} & \frac{\partial e_2}{\partial x_2} & \cdots & \frac{\partial e_2}{\partial x_3} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_n}{\partial x_1} & \frac{\partial e_n}{\partial x_2} & \cdots & \frac{\partial e_n}{\partial x_n} \end{bmatrix},$$

x_i é o i-ésimo padrão de entrada da RNA e

$$S = \sum_{i=1}^{n} e_i \nabla^2 e_i \tag{14}$$

Pode-se supor que S é um valor pequeno, se comparado ao produto da matriz Jacobiana, logo, a matriz Hessiana pode ser representada por

$$\nabla^2 f(W) \approx J^T J \tag{15}$$

Diante disso, a atualização dos pesos no método de Gauss-Newton apresentado na Equação 11 pode ser expressada por

$$W_{i+1} = W_i - (J^T J)^{-1} J^T e \tag{16}$$

Uma limitação do método de Gauss-Newton é que uma matriz Hessiana simplificada não pode ser invertida. Nesse ponto, o algoritmo de Levenberg-Marquardt apresenta uma modificação e, portanto, a atualização dos pesos é realizada da seguinte forma:

$$W_{i+1} = W_i - (J^T J + \mu I)^{-1} J^T e \tag{17}$$

sendo que I é a matriz identidade e μ, um parâmetro que torna a matriz Hessiana definida positiva.

Maiores detalhes sobre o algoritmo de Levemberg-Marquardt podem ser encontrados em Bishop [9], Hagan and Menhaj [20] e Liu [25].

4 Experimentos e resultados

Para dar credibilidade aos resultados, todos os experimentos foram realizados com a coleção WEBSPAM-UK2006.[4] Trata-se de uma base de dados pública e real de *web spam* que possui um conjunto de 77,9 milhões de páginas hospedadas em 11 mil *hosts* nos domínios do Reino Unido. Essa coleção foi utilizada no *Web Spam Challenge Track I* e *II*[5] que se trata de uma competição de técnicas de detecção de *web spam*.

Assim como empregado no campeonato, nos experimentos realizados foi utilizado um conjunto de 8.487 vetores de características. Essas informações foram extraídas da coleção de dados e fornecidas pelos organizadores do evento aos times participantes.

Um problema encontrado é que os vetores fornecidos não são rotulados. Dessa forma, comparou-se cada *host* referente a cada vetor de característica fornecido na competição com os *hosts* rotulados fornecidos junto com a base de dados. Nela, cada *host* é rotulado como *spam*, não *spam* (*ham*) ou indefinido. Dessa forma, os vetores referentes aos *hosts* que não possuem rótulos ou rotulados como indefinido foram descartados. Após esse pré-processamento restaram 6.509 (76.6%) *hosts* rotulados como *ham* e 1.978 (23.4%) rotulados como *spam*.

[4] *Yahoo! Research: "Web Spam Collections"*. Disponível em http://barcelona.research.yahoo.net/webspam/datasets/.
[5] *Web Spam Challenge:* http://webspam.lip6.fr/

4.1 Vetores de características

Os vetores de características usados neste trabalho para discriminar os *spam hosts* dos *hosts* legítimos são os mesmos empregados no *Web Spam Challenge* e estão documentados em Castilho et al. [11]. A seguir são apresentados os campos de cada vetor:

- *número de palavras na página, número de palavras no título e média do tamanho das palavras:* essas características consideram apenas as palavras visíveis de uma página e que são formadas apenas por caracteres alfanuméricos;
- *fração do texto âncora:* fração do número de palavras no texto âncora para o número total de palavras no texto visível na página. Texto âncora é uma anotação em um *link* que descreve o conteúdo da página de destino dessa ligação;
- *fração do texto visível:* fração do número de palavras visíveis em relação ao número de palavras na página (inclui *tags* HTML e outros textos invisíveis);
- *taxa de compressão:* o texto visível da página foi compactado usando o programa `bzip`. Então, a taxa de compressão é a relação entre o tamanho do texto compactado em relação ao tamanho do texto não compactado;
- *precisão e revocação da coleção:* foram encontradas as k palavras mais frequentes dos dados coletados. Então, a precisão da coleção é a dada pela fração de palavras em uma página que aparecem no conjunto de termos populares e revocação é a fração de termos populares que aparecem na página. Foram extraídas quatro características em relação a precisão e quatro em relação a revocação, usando k = 100, 200, 500 e 1000;
- *precisão e revocação da consulta:* foram considerados os q termos mais populares nas consultas. Dessa forma, a precisão da coleção é dada pela fração de palavras em uma página que aparecem no conjunto de termos populares das consultas e revocação é a fração de termos populares das consultas que aparecem na página. Foram extraídas quatro características em relação a precisão e quatro em relação a revocação, usando q = 100, 200, 500 e 1000;
- *probabilidade de trigrama independente:* um trigrama é um conjunto de três termos (*tokens*) consecutivos. Dada p_w, a probabilidade de distribuição de trigramas em uma página, e dado $T = w$, o conjunto de todos os trigramas em uma página, e $k = |T(p)|$, o número de trigramas diferentes. Então, a probabilidade de ocorrência de trigramas diferentes é uma medida de independência da distribuição de trigramas e é definida por $\frac{1}{k} \sum_{w \in T} \log p_w$;
- *entropia de trigramas:* entropia é a medida de compressibilidade de uma página, mas nesse caso considera-se apenas a distribuição de trigramas p_w, sendo calculada por $H = -\sum_{w \in T} p_w \log p_w$.

A lista de características apresentada fornece um total de 24 que são usadas para discriminar os *hosts*. Assim, para cada *host* são consideradas essas informações em relação à página inicial, à página com maior *pagerank*, à média dos valores para todas as páginas e desvio padrão de todas as páginas [11]. Logo, cada *host* é associado a um vetor composto por 96 características.

4.2 Medidas de desempenho

Para avaliar os resultados obtidos pelas RNAs, foram utilizadas as seguintes medidas [11] (Tabela 1):

- *acurácia:* taxa de acerto global, ou seja, proporção de predições corretas em relação ao tamanho do conjunto de dados;
- *sensitividade (recall):* proporção de padrões da classe positiva (*spam*) identificada corretamente. Indica o quão bom o classificador é para identificar a classe positiva;

- *especificidade:* proporção de padrões da classe negativa (*ham*) identificada corretamente. Indica o quão bom o classificador é para identificar a classe negativa;

- *precisão (precision):* porcentagem de padrões classificados como pertencentes à classe positiva e que realmente pertencem à classe positiva;

- *F-medida:* média harmônica entre precisão e sensitividade.

Tabela 1: Matriz de confusão

Classe	*Spam*	*Ham*
Spam	v_p	f_n
Ham	f_p	v_n
Acurácia:	$Acc = \frac{v_p+v_n}{v_p+v_n+f_p+f_n}$	
Sensitividade:	$Sen = \frac{v_p}{v_p+f_n}$	
Especificidade:	$Esp = \frac{v_n}{v_n+v_p}$	
Precisão:	$Pre = \frac{v_p}{v_p+f_p}$	
F-medida:	$Fmed = 2 * \frac{Pre*Sen}{Pre+Sen}$	

Na Tabela 1, v_p (verdadeiros positivos) referem-se ao número de exemplos corretamente classificados como *spam*, f_p (falsos positivos) referem-se aos exemplos que foram incorretamente classificados como *spam*, v_n (verdadeiros negativos) referem-se ao número de exemplos classificados corretamente como *ham* e f_n (falsos negativos) referem-se ao número de exemplos classificados incorretamente como *ham*.

4.3 Configurações gerais

Neste trabalho foram avaliadas as RNAs MLP treinadas com o algoritmo *backpropagation* e com o método do gradiente descendente e outra usando o método de segunda ordem de Levemberg-Marquardt.

Conforme usualmente empregado na literatura [11], cada classificador foi avaliado por meio do método de validação cruzada k-*folds*, usando $k = 5$. Dessa forma, os dados foram separados em cinco grupos de maneira circular. Logo, em cada simulação, quatro grupos (que representam 80% dos dados) foram usados para treinamento e um grupo, que representa 20% dos dados, foi dividido para ser usado para validação e teste. Portanto, os resultados apresentados a seguir foram obtidos pela média simples dos resultados das simulações.

Todas as MLPs implementadas usam uma única camada intermediária e possuem um neurônio na camada de saída. Além disso, adotou-se uma função de ativação linear para o neurônio da camada de saída e uma função de ativação do tipo tangente hiperbólica para os neurônios da camada intermediária. Dessa forma, os pesos e o bias da RNA foram inicializados com valores aleatórios entre 1 e -1 e os dados usados para a classificação foram normalizados para o intervalo $[-1, 1]$, por meio da seguinte equação:

$$x = 2 * \frac{x - x_{min}}{x_{max} - x_{min}} - 1,$$

sendo x a matriz com todos os vetores de características e x_{min} e x_{max} o menor e o maior valor da matriz x, respectivamente.

As saídas desejadas para a RNA são -1 (*ham*) e 1 (*spam*). Logo, na análise dos resultados, as saídas com valores maiores ou iguais a 0 são consideradas *spams*, caso contrário são *hams*. Além disso, em todas as simulações, os critérios de parada adotados foram: número de épocas maior que um limiar $Epocas_{max}$, erro quadrático médio (EQM) do conjunto de treino menor que um limiar EQM_{min} ou aumento do EQM do conjunto de validação (verificado a cada dez épocas).

4.4 Resultados

A arquitetura que apresentou os melhores resultados para a MLP com o método do gradiente descendente é composta por cem neurônios na camada intermediária. Os parâmetros usados foram: passo de aprendizagem $\alpha = 0.01$, $Epocas_{max} = 3000$ e $EQM_{min} = 0.001$.

Com tais parâmetros, a RNA atingiu o máximo de épocas adotado e, portanto, o treinamento foi abortado. Porém, tal interrupção não provocou grandes perdas de desempenho já que o EQM encontrava-se bem próximo de convergir. O EQM do treinamento atingiu o valor de 0.13 e os resultados obtidos podem ser observados na Tabela 2.

Tabela 2: Resultados obtidos pela rede neural artificial MLP treinada com o método do gradiente descendente

			Acurácia:	0.853
Classe	*Spam*	*Ham*	Sensitividade:	0.532
Spam	0.126	0.111	Especificidade:	0.952
Ham	0.037	0.727	Precisão:	0.775
			F-medida:	0.631

Conforme pode ser observado na Tabela 2, apesar da alta taxa de acurácia (85.3%), a taxa de sensitividade foi inferior a 54%, ao passo que a especificidade foi de 95.2%. Tais valores indicam que a RNA implementada é muito boa na classificação de padrões pertencentes à classe ham, mas é ineficiente para dados da classe $spam$. Outra métrica importante, a F-medida, também obteve apenas um valor razoável.

Outra RNA avaliada é a MLP com o método de segunda ordem de Levemberg-Marquardt. A arquitetura que apresentou os melhores resultados é composta por cinquenta neurônios na camada intermediária. Os parâmetros usados foram: passo de aprendizagem $\alpha = 0.001$, $Epocas_{max} = 3000$ e $EQM_{min} = 0.001$. O treinamento foi interrompido após 37 épocas, pois observou-se um aumento do erro de validação. O EQM obtido para essa simulação foi 0.05. Logo, observa-se que a RNA convergiu em tempo bem inferior ao que ocorreu na simulação anterior e atingiu um erro menor, mesmo com uma quantidade menor de neurônios na camada intermediária. Os resultados são apresentados na Tabela 3.

Tabela 3: Resultados obtidos pela rede neural artificial MLP treinada com o método de Levenberg-Marquardt

			Acurácia:	0.903
Classe	*Spam*	*Ham*	Sensitividade:	0.692
Spam	0.164	0.073	Especificidade:	0.969
Ham	0.024	0.740	Precisão:	0.874
			F-medida:	0.772

Os resultados apresentados claramente indicam que a RNA com método de Levenberg-Marquardt foi superior à RNA com método de primeira ordem. Note-se que a taxa de acurácia atingiu 90.3% e a taxa de sensitividade alcançou quase 70%. Logo, conclui-se que esse classificador aumentou a capacidade de predição de dados da classe positiva. Além disso, a taxa de precisão subiu para 87.4% e, portanto, a chance de um padrão ser classificado como *spam* e realmente ser *spam* também aumentou. Em consequência, o valor da F-medida também teve um aumento significativo.

Conforme mencionado, cada vetor de características é composto pela repetição dos atributos extraídos da página principal do *host*, da página com maior *pagerank*, a média das características nas páginas e o desvio padrão. Após analisar tais informações, cogitou-se a hipótese de que poderiam haver redundâncias que estariam impactando nos resultados.

Diante desse cenário, foi empregado o método de análise de componentes principais (PCA) [21] para tentar encontrar combinações lineares de variáveis que descrevem as principais tendências do conjunto de dados e, posteriormente, obter uma estrutura simplificada dos dados revelando a existência de redundâncias nas informações. Tal recurso é amplamente empregado no processo de redução de dimensionalidade em problemas de classificação e reconhecimento de padrões [27].

Por meio do PCA foi possível constatar que com apenas 23 dimensões é possível representar cerca de 99%

das informações presentes nos vetores de características disponíveis. De acordo com os resultados, as informações mais relevantes estão nas primeiras 21 dimensões do vetor. Logo, é possível inferir que as características extraídas da página principal dos *hosts* já apresentam grande parte das informações que podem caracterizá-lo como *spam* ou *ham*.

Após reduzir os vetores de características de 96 para 23 dimensões, foram feitas novas simulações.

O primeiro teste foi realizado com a RNA, com método do gradiente descendente usando os seguintes parâmetros: $\alpha = 0.01$, $Epocas_{max} = 3000$, $EQM_{min} = 0.001$ e cem neurônios na camada intermediária. Com essa configuração, o treinamento foi interrompido após atingir três mil épocas e EQM igual a 0.14. Os resultados obtidos são apresentados na Tabela 4.

Tabela 4: Resultados obtidos pela rede neural artificial MLP treinada com o método do gradiente descendente após redução de dimensionalidade usando PCA

			Acurácia:	0.837
Classe	*Spam*	*Ham*	Sensitividade:	0.437
Spam	0.101	0.131	Especificidade:	0.959
Ham	0.032	0.736	Precisão:	0.761
			F-medida:	0.555

Apesar dos novos vetores de características obtidos pelo PCA representarem mais de 99% das informações dos dados originais, os resultados obtidos nessa nova simulação foram inferiores. Isso pode ser facilmente notado pela taxa de acurácia e valor da F-medida menores do que as obtidas com os dados originais. Porém, a degradação dos resultados não foi tão expressiva e pôde ser compensada pelo ganho computacional, já que o número de dimensões que precisaram ser tratadas pela RNA diminuiu mais que 50%.

Uma nova simulação também foi realizada com a MLP treinada com o método de Levenberg-Marquardt. Os parâmetros usados foram: passo de aprendizagem $\alpha = 0.001$, $Epocas_{max} = 3000$, $EQM_{min} = 0.001$ e cinquenta neurônios na camada intermediária. Com essa configuração o treinamento foi interrompido após 24 épocas devido ao aumento no erro de validação. Os resultados podem ser observados na Tabela 5.

Tabela 5: Resultados obtidos pela rede neural artificial MLP treinada com o método de Levemberg-Marquardt após redução de dimensionalidade usando PCA

			Acurácia:	0.843
Classe	*Spam*	*Ham*	Sensitividade:	0.554
Spam	0.121	0.098	Especificidade:	0.925
Ham	0.059	0.722	Precisão:	0.673
			F-medida:	0.608

Novamente, os resultados obtidos também foram inferiores usando os dados com dimensão reduzida pelo PCA. Apesar de a taxa de especificidade não ter diminuído significativamente, a taxa de sensitividade foi bastante inferior e, consequentemente, também houve uma redução na F-medida.

Diante dos resultados obtidos pela classificação dos dados com dimensão reduzida, verificou-se que houve um ganho em relação ao custo computacional, no entanto, houve também uma degradação na qualidade dos resultados.

A diferença constante entre as altas taxas de especificidade e as baixas taxas de sensitividade provavelmente poderia ser explicada pela grande diferença na quantidade de padrões presentes na classe *ham* em relação à classe *spam* expostas ao classificador durante a fase de treinamento. Dessa forma, optou-se por igualar a quantidade de dados nas duas classes para usar durante o treinamento dos algoritmos e, portanto, foram desconsiderados 4571 representantes aleatórios da classe *ham*.

A partir dessa nova configuração de treinamento, foram realizadas novas simulações empregando os vetores de características originais. A RNA com o método do gradiente descendente foi avaliada com os seguintes parâmetros: $\alpha = 0.01$, $Epocas_{max} = 3000$, $EQM_{min} = 0.001$ e cem neurônios na camada intermediária. Com essa configuração, o treinamento foi interrompido ao atingir 3000 épocas e o EQM obtido foi de 0.12. Os novos resultados são apresentados na Tabela 6.

Tabela 6: Resultados obtidos pela rede neural artificial MLP treinada com o método do gradiente descendente usando classes de tamanhos iguais durante o treinamento

			Acurácia:	0.811
Classe	*Spam*	*Ham*	Sensitividade:	0.793
Spam	0.386	0.101	Especificidade:	0.828
Ham	0.088	0.424	Precisão:	0.814
			F-medida:	0.803

É claramente notável que igualar a quantidade de representantes nas duas classes durante o treinamento melhorou os resultados da classificação. Isso pode ser observado pelo aumento do valor da F-medida, que foi bem superior ao encontrado nas demais simulações. Outro ponto a ser observado é que a taxa de sensitividade também aumentou, o que significa que o classificador especializou-se mais em acertar os padrões que são da classe *spam*.

Também foi realizada uma nova simulação com a RNA com método de Levenberg-Marquardt, treinando-a com classes de tamanhos iguais. Os parâmetros usados foram $\alpha = 0.001$, $Epocas_{max} = 3000$, $EQM_{min} = 0.001$ e cinquenta neurônios na camada intermediária. Com essa configuração, o treinamento foi interrompido com 55 épocas devido ao aumento no erro de validação e o EQM obtido foi de 0.04. Os resultados podem ser observados na Tabela 7.

Tabela 7: Resultados obtidos pela rede neural articial MLP treinada com o método de Levemberg-Marquardt usando classes de tamanhos iguais durante o treinamento

			Acurácia:	0.899
Classe	*Spam*	*Ham*	Sensitividade:	0.902
Spam	0.439	0.048	Especificidade:	0.897
Ham	0.053	0.460	Precisão:	0.892
			F-medida:	0.897

Conforme pode ser observado, houve uma melhora ainda mais significativa nos resultados. Note-se que o valor obtido para a F-medida foi quase igual a 0.9, demonstrando que tanto a taxa de sensitividade quanto a de precisão foram igualmente altas.

Para uma análise mais confiável do desempenho das RNAs na classificação de *web spam*, foi feita uma comparação (Tabela 8) dos resultados das MLPs obtidos neste trabalho com os resultados de uma das técnicas estado da arte da literatura de *web spam*, a técnica de bagging de árvores de decisão c4.5, empregada por Castilho et al. [11] e Ntoulas et al. [26]. Para que a comparação seja justa, nós implementamos a técnica de bagging de c4.5 através da ferramenta Weka, usando a mesma base de dados e sistemática de validação usada nos testes com as RNAs (k-folds com $k = 5$).

Tabela 8: Comparação dos resultados obtidos pelas RNAs MLP com outros trabalhos da literatura

Classificador	Características	Precisão	Sensitividade	F-medida
Resultados da literatura				
Bagging de árvores de decisão [11, 26]	conteúdo	0.807	0.677	0.736
Bagging de árvores de decisão [11, 26]	conteúdo e *links*	0.826	0.788	0.806
Bagging de árvores de decisão + classes iguais [11, 26]	conteúdo	0.837	0.831	0.834
Bagging de árvores de decisão + classes iguais [11, 26]	conteúdo e *links*	0.873	**0.920**	0.896
Resultados obtidos pelas redes neurais artificiais MLP				
Gradiente descendente	conteúdo	0.775	0.532	0.631
Levenberg-Marquardt	conteúdo	0.874	0.692	0.772
Gradiente descendente + Classes com tamanhos iguais	conteúdo	0.814	0.793	0.803
Levenberg-Marquardt + Classes com tamanhos iguais	conteúdo	**0.892**	0.902	**0.897**

Conforme pode ser observado no método usado por Castilho et al. [11] e Ntoulas et al. [26], o maior valor da F-medida obtida na classificação de *web spam* por análise de conteúdo foi igual a 0.834, quando foram utilizadas classes balanceadas na classificação. Em outro teste foram analisadas características relacionadas tanto ao conteúdo das páginas quanto dos *links*, o que aumentou o valor da F-medida para 0.896. Porém, usando redes

neurais artificiais MLP treinadas com o algoritmo de Levenberg-Marquardt, mesmo considerando apenas características relacionadas ao conteúdo, foi obtido um resultado superior quando foram usadas classes balanceadas, fato esse que demonstra que o emprego e o estudo das RNAs podem ser bastante promissores para auxiliar na solução do problema apresentado.

5 Conclusões e trabalhos futuros

Este trabalho apresentou uma análise de desempenho de classificação de *web spam* usando técnicas de redes neurais artificiais perceptron de múltiplas camadas.

Em geral, foi observado que a MLP treinada com o algoritmo *backpropagation* com gradiente descendente é inferior à MLP treinada com o método de Levenberg-Marquardt. Além do quê, os resultados obtidos pela MLP treinada com o método de Levenberg-Marquardt foram superiores aos resultados da técnica estado-da-arte da literatura de *web spam*, a técnica de bagging de árvores de decisão [11, 26].

Outro ponto importante observado é que, para o problema analisado, treinar as redes neurais com classes de tamanhos muito diferentes faz os resultados convergirem em benefício da classe com maior número de representantes, o que aumenta a taxa de falsos positivos ou de falsos negativos. Logo, é importante balancear a quantidade de representantes de cada classe utilizada como entrada do algoritmo.

A análise dos vetores de características inicialmente propostos por Castilho et al. [11] indicam que esses possuem muitas informações redundantes, já que, com o método PCA, foi possível diminuí-los de 96 para apenas 23 dimensões. Contudo, tal redução não contribuiu para a melhoria dos resultados.

Em trabalhos futuros pretende-se pesquisar e propor outras características que auxiliem na discriminação das classes. Também será analisada a viabilidade do emprego de outras técnicas de redes neurais, em especial os mapas auto-organizáveis de Kohonen, já que alguns trabalhos na literatura indicam ter obtido êxito na classificação de *spam* enviados por e-mail. Outro tema de estudo a ser abordado é a implementação de métodos auxiliares que levem em consideração a relação topológica entre os *links* das páginas para auxiliar na classificação dos *hosts*.

Agradecimentos

Os autores são gratos à Capes, à Fapesp e ao CNPq pelo apoio financeiro.

Referências

[1] ABERNETHY, J.; CHAPELLE, O.; CASTILLO, C. Graph regularization methods for web spam detection. *Machine Learning*, v.81, n.2, p.207-225, 2010.

[2] ALMEIDA, T.; YAMAKAMI, A. Content-Based Spam Filtering. In *Proceedings of the 23rd IEEE International Joint Conference on Neural Networks*. Barcelona: Spain, 2010. p.1-7.

[3] ALMEIDA, T.; YAMAKAMI, A. Redução de Dimensionalidade Aplicada na Classificação de Spams Usando Filtros Bayesianos. *Revista Brasileira de Computação Aplicada*, v.3, n.1, p.16-29, 2011.

[4] ALMEIDA, T.; YAMAKAMI, A. Facing the Spammers: A Very Effective Approach to Avoid Junk E-mails. *Expert Systems with Applications*. 2012.

[5] ALMEIDA, T.; YAMAKAMI, A.; ALMEIDA, J. Evaluation of Approaches for Dimensionality Reduction Applied with Naive Bayes Anti-Spam Filters. In *Proceedings of the 8th IEEE International Conference on Machine Learning and Applications*. Miami: USA, 2009. p.517-522.

[6] ALMEIDA, T.; YAMAKAMI, A.; ALMEIDA, J. Filtering Spams using the Minimum Description Length Principle. In *Proceedings of the 25th ACM Symposium On Applied Computing*, Sierre: Switzerland, 2010. p.1856-1860.

[7] ALMEIDA, T.; YAMAKAMI, A.; ALMEIDA, J. Probabilistic Anti-Spam Filtering with Dimensionality Reduction. In *Proceedings of the 25th ACM Symposium On Applied Computing*. Sierre: Switzerland, 2010. p.1804-1808.

[8] ALMEIDA, T.; ALMEIDA, J.; YAMAKAMI, A. Spam Filtering: How the Dimensionality Reduction Affects the Accuracy of Naive Bayes Classifiers. *Journal of Internet Services and Applications*, v.1, n.3, p.183-200, 2011.

[9] BISHOP, C. M. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford. 1995.

[10] BRAGA, A. P.; LUDENIR, T. B.; CARVALHO, A. C. P. L. F. *Redes Neurais Artificiais: Teorias e Aplicações*. Livros Técnicos e Científicos, Rio de Janeiro. 2007.

[11] CASTILHO, C.; DONATO, D.; GIONIS, A. Know your neighbors: web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR Conference on Research and development in information retrieval*, SIGIR'07. Amsterdam: The Netherlands, 2007. p.423-430.

[12] CORMACK, G. Email Spam Filtering: A Systematic Review. *Foundations and Trends in Information Retrieval*, v.1, n.4, p.335-455, 2008.

[13] EGELE, M.; KOLBITSCH, C.; PLATZER, C. Removing web spam links from search engine results. *Journal in Computer Virology*, v.7, p.51-62, 2011.

[14] EIRON, N.; McCURLEY, K. S.; TOMLIN, J. A. Ranking the web frontier. In *Proceedings of the 13rd international Conference on World Wide Web*, WWW'04. New York: USA, 2004. p.309-318.

[15] ERDÉLYI, M.; GARZÓ, A.; BENCZúr, A. A. Web spam classification: a few features worth more. In *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality*, WebQuality'11. Hyderabad: India, 2011. p.27-34.

[16] GAN, Q.; SUEL, T. Improving web spam classifiers using link structure. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb'07. Banff: Canada, 2007. p.17-20.

[17] GENG, G.; WANG, C.; LI, Q.; XU, L.; JIN, X. Boosting the performance of web spam detection with ensemble under-sampling classification. In *Proceedings of the 14th International Conference on Fuzzy Systems and Knowledge Discovery*, FSKD'07. Haikou: China, 2007.p.583-587.

[18] GUZELLA, T.; CAMINHAS, W. A Review of Machine Learning Approaches to Spam Filtering. *Expert Systems with Applications*, v.36, n.7, p.10206-10222, 2009.

[19] GYONGYI, Z.; GARCIA-MOLINA, H. Spam: it's not just for inboxes anymore. *Computer*, v.38, n.10, p.28-34, 2005.

[20] HAGAN, M. T.; MENHAJ, M. B. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, v.5, n.6, p.989-993, 1994.

[21] HAYKIN, S. *Redes Neurais - Príncipios e Práticas*. BOOKMAN, São Paulo, 2th ed. 2001.

[22] LARGILLIER, T.; PEYRONNET, S. Lightweight clustering methods for webspam demotion. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Toronto: Canada, 2010. Volume 1, p.98-104.

[23] LARGILLIER, T.; PEYRONNET, S. Using patterns in the behavior of the random surfer to detect webspam beneficiaries. In *Proceedings of the 2010 International Conference on Web information systems engineering*, WISS'10. Berlin/Heidelberg: Springer, 2011. p.241-253.

[24] LEON-SUEMATSU, Y. I.; INUI, K.; KUROHASHI, S.; KIDAWARA, Y. Web spam detection by exploring densely connected subgraphs. In *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, WI-IAT'11. Lyon: França, 2011. Volume 1, p.124-129.

[25] LIU, H. On the levenberg-marquardt training method for feed-forward neural networks. In *Proceedings of the 2010 International Conference on Natural Computation*, ICNC'10, volume 1. 2010.

[26] NTOULAS, A.; NAJORK, M.; MANASSE, M.; FETTERLY, D. Detecting spam web pages through content analysis. In *Proceedings of the World Wide Web conference*, WWW'06. Edinburgh: Scotland, 2006.p.83-92.

[27] QIU, B.; PRINET, V.; PERRIER, E.; MONGA, G. Multi-block pca method for image change detection. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, ICIAP'03, 2003. p.385-390.

[28] R, B.; SANTANA, R. Linked latent dirichlet allocation in web spam filtering. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb'09. Madrid: Spain, 2009. p.37-40.

[29] REN, Q. Feature-fusion framework for spam filtering based on svm. In *Proceedings of the 7th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, CEAS'10. Redmond: USA, 2010. p.1-6.

[30] RUNGSAWANG, A.; TAWEESIRIWATE, A.; MANASKASEMSAK, B. Spam host detection using ant colony optimization. In *IT Convergence and Services*, volume 107 of *Lecture Notes in Electrical Engineering*. Springer. Netherlands, 2011. p.13-21.

[31] SHEN, G.; GAO, B.; LIU, T.; FENG, G.; SONG, S.; LI, H. Detecting link spam using temporal information. In *Proceedings of the 6th IEEE International Conference on Data Mining*, ICDM'06, 2006. p.1049-1053.

[32] SHENGEN, L.; XIAOFEI, N.; PEIQI, L.; LIN, W. Generating new features using genetic programming to detect link spam. In *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*. Guangdong: China, 2011. Volume 1, p.135-138.

[33] SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. Redes Neurais Artificiais para Detecção de Web Spams. In *Proceedings of the 8th Brazilian Symposium on Information Systems*, SBSI'12. São Paulo, Brazil, 2012. p.135-138.

[34] SILVA, R. M.; ALMEIDA, T. A.; YAMAKAMI, A. Artificial Neural Networks for Content-based Web Spam Detection. In *Proceedings of the 14th International Conference on Artificial Intelligence*, ICAI'12. Las Vegas: USA, 2012. p.1-7.

[35] SVORE, K. M.; WU, Q.; BURGES, C. J. Improving web spam classification using rank-time features. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb'07. Banff: Canada, 2007. p.9-16.

[36] URVOY, T.; CHAUVEAU, E.; FILOCHE, P. Tracking web spam with html style similarities. *ACM Transactions on the Web*, v.2, n.1, p.1-3, 2008.

[37] ZHANG, L.; ZHU, J.; YAO, T. An Evaluation of Statistical Spam Filtering Techniques. *ACM Transactions on Asian Language Information Processing*, v.3, n.4, p.243-269, 2004.

[38] ZHANG, W.; ZHU, D.; ZHANG, Y.; ZHOU, G.; XU, B. Harmonic functions based semi-supervised learning for web spam detection. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11. New York: ACM, 2011. p.74-75.