Análise de Métodos de Aprendizagem de Máquina para Detecção Automática de *Spam Hosts*

Renato Moraes Silva[1], **Tiago A. Almeida**[2], **Akebo Yamakami**[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP) – Campinas – SP – Brasil

[2]Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar) – Sorocaba – SP – Brasil

{renatoms,akebo}@dt.fee.unicamp.br, talmeida@ufscar.br

Abstract. *Web spamming is one of the main problems that affect the quality of search engines. The number of web pages that use this technique to achieve better positions in search results is growing. The main motivation is the profit achieved with the online advertising market, besides attacks on Internet users through malware that steal information to facilitate bank thefts. Given this scenario, this paper presents an analysis of machine learning techniques employed to detect spam hosts. Experiments performed with a real, public and large dataset, indicate that ensemble of decision trees are promising in the task of spam hosts detection.*

Resumo. *Web spamming é um dos principais problemas que afeta a qualidade das ferramentas de busca. O número de páginas web que usam esta técnica para conseguir melhores posições nos resultados de busca é cada vez maior. A principal motivação são os lucros obtidos com o mercado de publicidade online, além de ataques a usuários da Internet por meio de malwares, que roubam informações para facilitar roubos bancários. Diante disso, esse trabalho apresenta uma análise de técnicas de aprendizagem de máquina aplicadas na detecção de spam hosts. Experimentos realizados com uma base de dados real, pública e de grande porte indicam que as técnicas de agregação de métodos baseados em árvores são promissoras na tarefa de detecção de spam hosts.*

1. Introdução

As ferramentas de busca são grandes aliadas dos usuários da Internet para encontrar informações e por isso são responsáveis por uma porcentagem expressiva das visitas recebidas pela maioria dos *web sites*. Logo, para ter sucesso, é importante que o *web site* mantenha um alto *ranking* de relevância nos motores de busca (*pagerank*). Dessa forma, poderá melhorar seu posicionamento nas páginas de resultado dessas ferramentas, quando forem consultados termos relacionados ao seu conteúdo ou serviços oferecidos. Para atingir esse propósito, podem ser usadas diversas estratégias, conhecidas como otimização para motores de busca (SEO – *search engine optimization*) [Ledford 2009].

Existem diversas estratégias éticas de SEO, porém como afirma [Ledford 2009], para aprender as mais bem sucedidas, é preciso muito tempo e dedicação. Todos os elementos do *web site* devem ser projetados para maximizar seu *ranking* nos motores de

busca. No entanto, como mencionado no guia de SEO publicado pela empresa Google[1], esse processo deve ser pensado principalmente para os consumidores do *web site* e não para as ferramentas de busca. É importante que o projeto do *web site* ofereça conteúdo relevante, facilidade de navegação e outras características que beneficiem o usuário. O problema é que muitos *sites* preferem investir em técnicas antiéticas de SEO, enganando os motores de busca para ganhar relevância sem merecê-la. Os maiores prejudicados são os usuários que ao fazerem suas consultas recebem respostas inesperadas, irrelevantes e muitas vezes infectadas por conteúdos maliciosos e prejudiciais, como *malwares* ou outras pragas virtuais. Essa técnica é conhecida como *web spamming* ou *spamdexing* [Svore et al. 2007] e, segundo [Gyongyi e Garcia-Molina 2005], tais páginas podem possuir tanto conteúdo *spam* quanto *spam links*. O primeiro consiste em criar páginas com milhares de palavras-chaves irrelevantes e o segundo consiste em adicionar *links* que apontam para as páginas que pretende-se promover.

Um dos motivos que fomenta a criação e o crescimento do volume de *web spam* é o incentivo econômico a essas práticas. Geralmente, os *spammers* criam páginas com conteúdo irrelevante, que conseguem boas posições no *ranking* dos motores de busca por meio de técnicas de *spamming*, e colocam anúncios para produtos de *sites* de compras ou para outras modalidades de *sites*. Assim, ele recebe algum dinheiro por cada clique dado pelos usuários [Gyongyi e Garcia-Molina 2005].

Estudos recentes apontam que o volume de *web spam* vem aumentando consideravelmente nos últimos anos. Segundo [Ntoulas et al. 2006], cerca de 13,8% da páginas de língua inglesa, 25% das francesas e 22% das germânicas são *spam*. Em outro estudo, [John et al. 2011] observaram que 36% dos resultados dos motores de busca Google e Bing contém URLs maliciosas. Um relatório produzido pela empresa Websense[2] mostra que 22,4% dos resultados de busca sobre entretenimento levam a *links* maliciosos. Além disso, segundo um relatório publicado pela empresa McAfee[3], 49% dos termos de busca mais populares retornam algum *site* malicioso entre os 100 primeiros resultados da busca. A mesma pesquisa aponta que 1,2% das consultas retornam *links* de *sites* maliciosos entre os 100 primeiros resultados.

Diante desse cenário, este trabalho apresenta uma análise de desempenho de diversas técnicas bem conhecidas de aprendizado de máquina que podem ser aplicadas para auxiliar no combate desse problema. O objetivo é encontrar métodos promissores que podem ser explorados e empregados para auxiliar na detecção automática de *spam hosts*. Resultados preliminares obtidos com redes neurais artificiais usando atributos extraídos do conteúdo das páginas *web* foram publicados em [Silva et al. 2012a, Silva et al. 2012b]. Contudo, neste artigo são oferecidas as seguintes contribuições:

- experimentos realizados com diversos outros métodos, tais como métodos de agregação de múltiplos classificadores, métodos baseados em árvores, métodos baseados em *cluster* e máquinas de vetores de suporte.
- experimentos realizados com diferentes vetores de características (atributos) extraídos de páginas *web* disponíveis em uma base de dados grande, real e pública;

[1] *Search Engine Optimization Starter Guide*. Consultar: `http://www.google.co.jp/intl/en/webmasters/docs/search-engine-optimization-starter-guide.pdf`

[2] *Websense 2010 Threat Report*. Consultar: `http://www.websense.com/assets/reports/report-websense-2010-threat-report-en.pdf`

[3] *McAfee Threats Report: First Quarter 2011*. Consultar: `http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2011.pdf`

- experimentos que ilustram os impactos na acurácia dos métodos devido ao balanceamento dos dados de treinamento;
- comparação dos resultados obtidos nesse trabalho com os resultados obtidos por métodos propostos por outros autores da literatura;

Este artigo está estruturado da seguinte forma: na Seção 2 são brevemente descritos os trabalhos correlatos disponíveis na literatura. Na Seção 3 são apresentados os conceitos básicos sobre os métodos classificadores avaliados neste trabalho. Na Seção 4 são descritas as configurações adotadas nos experimentos e a base de dados empregada. Os resultados experimentais são apresentados na Seção 5. Por fim, conclusões e direções para trabalhos futuros são descritos na Seção 6.

2. Trabalhos correlatos

O problema de *web spamming* é relativamente recente e, portanto, existem poucos trabalhos que oferecem avanços significativos no sentido de resolvê-lo. Em um dos trabalhos da literatura de *web spam*, [Gyongyi e Garcia-Molina 2005] descrevem uma variedade de técnicas empregadas pelos *spammers* para disseminar *web spam*. Dentre elas, as mais comuns são: inserção de palavras chaves populares no título das páginas ou nas *meta tags*, métodos que tornam o texto invisível para o usuário, que exploram *links* e que redirecionam o usuário para outras páginas.

Em outro trabalho [Castillo et al. 2007] propõem um conjunto de características baseadas no conteúdo das páginas *web* para separar os *spam hosts* dos *ham hosts* e apresentam um sistema de detecção de *web spam* que combina características baseadas nos *links* e no conteúdo. Além disso, eles usam a topologia do grafo *web* através da exploração da dependência de *links* entre as páginas *web*. Em contrapartida, em uma abordagem diferente da adotada no trabalho anterior, [Svore et al. 2007] usam o método de SVM na detecção de *web spam* através de características baseadas no conteúdo e que consideram o tempo de *rank*.

[Shengen et al. 2011] propõem novas características para a identificação de *web spam* por meio de programação genética usando características baseadas em *links* e apresenta os efeitos do número de indivíduos, do número de gerações e da profundidade da árvore binária nos resultados do algoritmo. [Rungsawang et al. 2011] apresentam um algoritmo de otimização por colônia de formigas para a detecção de *spam hosts* que explora tanto características baseadas no conteúdo quanto características baseadas nos *links*.

[Jayanthi e Sasikala 2012] apresentam um algoritmo para detecção de *web spam* chamado `WESPACT` que usa algoritmos genéticos para classificar atributos baseados em *links* e em conteúdo. Seguindo a mesma linha do trabalho anterior, com o uso de algoritmos de computação natural, [Taweesiriwate et al. 2012] propõem o uso do algoritmo de otimização por colônia de formigas para a detecção de web spam, porém seguem a estratégia de *TrustRank* [Gyongyi et al. 2004] a fim de gerar regras de classificação para a detecção de *web spam*. [Largillier e Peyronnet 2012] usam uma abordagem diferente para o problema e apresentam diversos métodos de agrupamento de nós para a diminuição dos efeitos do *web spamming* no algoritmo *PageRank* (algoritmo de *ranking* dos motores de busca). Por outro lado, [Liu et al. 2012] propõem algumas características baseadas no comportamento dos usuários para separar as páginas *spam* das páginas legítimas (*ham*).

3. Classificadores

Apesar da existência de trabalhos cujo intuito é filtrar *web spam*, ainda não há um consenso se a aplicação de técnicas de aprendizado de máquina é realmente eficaz na detecção

automática de *spam hosts*. Tendo isso em vista, foram avaliados neste trabalho diversos métodos de classificação bem conhecidos.

Nessa seção, são apresentados os métodos classificadores avaliados nesse trabalho: redes neurais artificiais (RNAs) perceptron de múltiplas camadas (MLP – *multilayer perceptron*), máquina de vetores de suporte (SVM – *support vector machines*), métodos baseados em árvores (C4.5 e floresta aleatória), IBK, *boosting* adaptativo (AdaBoost – *adaptive boosting*), *bagging* e LogitBoost. A escolha de tais métodos reside no fato de terem sido avaliados e listados como as melhores técnicas de mineração de dados atualmente disponíveis [Wu et al. 2008].

As RNAs não fazem parte da lista proposta por [Wu et al. 2008], mas foram escolhidas para serem avaliadas devido a sua alta capacidade de generalização. A técnica de floresta aleatória também foi escolhida, mesmo não estando na lista, pois ela faz uma combinação de árvores de decisão. Logo, como o método C4.5 é um algoritmo de árvore de decisão e está na lista dos melhores métodos, acredita-se que a combinação de árvores de decisão também possa gerar bons resultados. Por fim, também foram feitos experimentos com o método OneR pois ele tem um baixo custo computacional e é um dos algoritmos de aprendizagem de máquina mais simples. Logo, o seu desempenho pode ser usado para analisar o desempenho obtido por algoritmos mais complexos.

3.1. Rede neural artificial perceptron de múltiplas camadas

Uma RNA perceptron de múltiplas camadas (MLP – *multilayer perceptron*) é uma rede do tipo *perceptron* que possui um conjunto de unidades sensoriais que formam a camada de entrada, uma ou mais camadas intermediárias de neurônios computacionais e uma camada de saída [Haykin 1998]. Por padrão, o seu treinamento é supervisionado e usa o algoritmo *backpropagation* (retropropagação do erro), que tem a função de encontrar as derivadas da função de erro com relação aos pesos e bias da rede. Esse algoritmo pode ser resumido em duas etapas: *forward* e *backward* [Bishop 1995].

Na etapa *forward*, o sinal é propagado pela rede, camada a camada, da seguinte forma: $u_j^l(n) = \sum_{i=0}^{m^{l-1}} w_{ji}^l(n) y_i^{l-1}(n)$, sendo $l = 0, 1, 2, ..., L$ o índice das camadas da rede. Quando $l = 0$, ele representa a camada de entrada e quando $l = L$ representa 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 é dada por: $y_j^l(n) = \varphi_j(u_j^l(n))$, onde φ_j é a função de ativação do neurônio j. O erro pode ser calculado por: $e_j^l(n) = y_j^l(n) - d(n)$, sendo que $d(n)$ é a saída desejada para o padrão de entrada $x(n)$.

Na etapa *backward*, inicia-se a derivação do algoritmo *backpropagation*, a partir da camada de saída, onde tem-se: $\delta_j^L(n) = \varphi'_j(u_j^L(n)) e_j^L(n)$, sendo φ'_j a derivada da função de ativação. Para $l = L, L-1, ..., 2$, calcula-se: $\delta_j^{l-1}(n) = \varphi'_j(u_j^{l-1}(n)) \sum_{i=1}^{m^l} w_{ji}^l(n) * \delta_j^l(n)$, para $j = 0, 1, ..., m^l - 1$.

Para maiores detalhes sobre as MLPs, consulte [Bishop 1995, Haykin 1998].

Algoritmo de Levenberg-Marquardt

O algoritmo de Levenberg-Marquardt é um método de otimização e aceleração da convergência do algoritmo *backpropagation*. Ele 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 [Bishop 1995].

3.2. Máquinas de vetores de suporte

Máquinas de vetores de suporte (SVM – *support vector machines*) [Cortes e Vapnik 1995] é um método de aprendizagem de máquina que pode ser usado para problemas de classificação e regressão e outras tarefas de aprendizagem [Haykin 1998, Chang e Lin 2011]. Elas foram conceitualmente implementadas seguindo a ideia de que vetores de entrada são não-linearmente mapeados para um espaço de atributos de alta dimensão. Nesse espaço, é construída uma superfície de decisão que permite distinguir as classes dos exemplos de entrada.

Para conseguir separar dados linearmente ou não-linearmente separáveis, um dos principais elementos usados pelo método SVM é uma função de *kernel*. Através dela, o SVM constrói uma superfície de decisão que é não-linear no espaço de entrada, mas é linear no espaço de atributos [Haykin 1998]. As principais funções de *kernel* que podem ser utilizadas no SVM são [Haykin 1998, Hsu et al. 2003]: linear, radial basis function (RBF), polinomial e sigmoidal.

Para a escolha dos parâmetros do SVM, a técnica recomendada em [Hsu et al. 2003] é a *grid search* (busca em grade). Então, considerando o SVM com *kernel* RBF, em que deve-se definir o parâmetro de regularização C e o parâmetro γ, eles propõem testar as seguintes sequências exponenciais: $C = 2^{-5}, 2^{-4}, 2^{-3}..., 2^{15}$ e $\gamma = 2^{-15}, 2^{-14}..., 2^{3}$.

3.3. C4.5

O C4.5 [Quinlan 1993] é um dos mais clássicos algoritmos de árvores de decisão e trabalha tanto com atributos categóricos quanto contínuos. Além disso, ele permite o uso de atributos desconhecidos, desde que sejam representados por "?". O C4.5 usa um método de dividir e conquistar para aumentar a capacidade de predição das árvores de decisão. Dessa forma, um problema é dividido em vários sub-problemas, criando-se sub-árvores no caminho entre a raiz e as folhas da árvore de decisão.

3.4. Floresta aleatória

Uma floresta aleatória (*random forest*) [Breiman 2001] é uma combinação de árvores de decisão, em que cada árvore depende dos valores de vetores aleatórios amostrados de forma independente e distribuídos igualmente para todas as árvores na floresta. Nesse método, depois que um determinado número de árvores são geradas, cada uma lança um voto para uma classe do problema, considerando um vetor de entrada. Então, a classe mais votada será escolhida na predição do classificador.

3.5. IBK

O algoritmo IBK é um algoritmo de aprendizagem baseado em instâncias (IBL – *instance-based learning*) [Aha et al. 1991]. Esse tipo de algoritmo é derivado do método de classificação k-vizinhos mais próximos (KNN – *k-nearest neighbor*). Porém, este último

é um algoritmo não-incremental e tem como objetivo principal manter uma consistência perfeita com o conjunto inicial de treinamento. Já o algoritmo do tipo IBL é incremental e tem como objetivo maximizar a acurácia sobre novas instâncias do problema [Aha et al. 1991].

Assim como no método KNN, no método IBK existe uma função de similaridade que obtém um valor numérico obtido pelo cálculo da distância euclidiana. Então a classificação gerada para um padrão i será influenciada pelo resultado da classificação dos seus k-vizinhos mais próximos, pois padrões similares devem ter classificações similares [Aha et al. 1991, Witten e Frank 2005].

3.6. *Boosting* adaptativo

O método de *boosting* adaptativo (AdaBoost – *adaptive boosting*) [Freund e Schapire 1996] é um algoritmo de *boosting* amplamente utilizado para problemas de classificação. Em geral, assim como qualquer método de *boosting*, ele faz uma combinação de classificadores. Porém, segundo [Freund e Schapire 1996], ele possui algumas propriedades que o tornam mais prático e fácil de ser implementado do que os algoritmos de *boosting* que o antecederam, pois ele não necessita de nenhum conhecimento prévio das predições obtidas por classificadores ruins. Em vez disso ele se adapta as predições ruins e gera uma hipótese majoritária ponderada, em que o peso das predições fornecidas pelos classificadores ruins torna-se uma função de sua predição.

3.7. *Bagging*

O *bagging* (abreviação de *bootstrap aggregating*) [Breiman 1996] é um método de geração múltiplas versões de um classificador que são combinadas para a obtenção de um classificador agregado. O processo adotado pelo método de *bagging* é semelhante ao do método de *boosting*, porém de acordo com [Witten e Frank 2005], diferente do que ocorre no segundo método, no *bagging*, os diferentes modelos de classificadores recebem o mesmo peso na geração de uma predição.

No método de *bagging* quando a predição do classificador deve ser numérica, é feita uma média sobre os resultados de todos os modelos de classificadores. Por outro lado, se a predição deve ser uma classe, é feita uma votação e a classe com maior pontuação será escolhida para representar o padrão [Breiman 1996].

3.8. LogitBoost

O método LogitBoost [Friedman et al. 1998] é uma versão estatística do método de *boosting* e, segundo [Witten e Frank 2005], possui algumas semelhanças com o método Adaboost, porém ele otimiza a probabilidade de ocorrência de uma classe, enquanto o Adaboost otimiza uma função de custo exponencial. Em [Friedman et al. 1998] esse método é definido como um algoritmo para montagem de modelos aditivos de regressão logística.

3.9. OneR

O método OneR ou 1R (abreviação de 1-*rules*) [Holte 1993] pode ser considerado uma árvore de decisão com 1 nível, pois gera um conjunto de regras, uma para cada atributo do conjunto de dados e classifica uma amostra com base em um único atributo. O atributo escolhido é aquele, cuja regra produz o menor erro.

3.10. Naive Bayes

O método naive Bayes [John e Langley 1995] é um classificador probabilístico simples baseado no teorema de Bayes. Esse método é denominado ingênuo (*naive*) porque ele assume que os atributos contribuem de forma independente para formar probabilidades estimativas da ocorrência de uma determinada classe do problema durante o processo de classificação. Logo, segundo [Witten e Frank 2005] a existência de redundância nos atributos pode distorcer o processo de aprendizagem.

4. Base de dados e configurações

Os experimentos foram realizados com a base de dados pública *WEBSPAM-UK2006 collection*[4]. Ela é composta por 77,9 milhões de páginas *web* hospedadas em 11.000 *hosts*. Essa base foi utilizada no *Web Spam Challenge Track I* e *II*[5], que trata-se de uma competição de técnicas de detecção de *web spam*.

Assim como empregado nos campeonatos, nos experimentos realizados foram utilizados 3 conjuntos de 8.487 vetores de características pré-computados. Cada conjunto é composto por 1.978 *hosts* rotulados como *spam* e 6.509 *hosts* rotulados como *ham* (não-spam). Essas informações foram extraídas da coleção de dados e fornecidas pelos organizadores do evento aos times participantes. O primeiro conjunto de vetores de características é composto por 96 características baseadas no conteúdo [Castillo et al. 2007], o segundo é composto por 41 características baseadas nos *links* [Becchetti et al. 2006] e o terceiro é composto por 138 características baseadas nos *links* transformados [Castillo et al. 2007], que são simples combinações ou operações logarítmicas sobre as características baseadas em *links*.

Para avaliar o desempenho de cada classificador foi usada uma validação por subamostragem aleatória, também conhecida como validação cruzada de Monte Carlo [Shao 1993]. Essa método de validação foi escolhido pois dá mais liberdade na seleção dos subconjuntos de treinamento e teste, já que diferente do método de validação k-folds, permite que sejam feitas quantas repetições forem desejadas, usando qualquer porcentagem de dados para treinamento e teste. Então, foram feitas 10 simulações com cada classificador e em cada uma, 80% dos dados foram usados para treinamento e 20% para teste. Eles foram selecionados aleatoriamente e com reposição a cada simulação. Ao final de todas as simulações foi calculada a média e o desvio padrão dos resultados. Para avaliar e comparar os resultados dos classificadores foram utilizadas as seguintes medidas de desempenho amplamente empregadas na literatura:

- *Sensitividade (recall):* proporção de padrões da classe positiva (*spam*) identificada corretamente.
- *Precisão (precision):* porcentagem de padrões classificados como pertencentes a classe positiva e que realmente pertencem a classe positiva.
- *F-medida:* média harmônica entre precisão e sensitividade, dada por:

$$F - medida = 2 \times \left(\frac{precisao \times sensitividade}{precisao + sensitividade} \right).$$

[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. Configurações

Para tornar os resultados completamente reprodutíveis, são apresentadas nessa seção as configurações adotadas para cada classificador.

4.1.1. Redes neurais artificiais

Neste trabalho, foram avaliadas as seguintes RNAs: MLP treinada com o algoritmo *backpropagation* e método do gradiente descendente (MLP-GD) e MLP treinada com o método de Levenberg-Marquardt (MLP-LM).

Todas as redes MLP 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 rede 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 equação $x = 2 \times \frac{x - x_{min}}{x_{max} - x_{min}} - 1$, 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. Além disso, em todas as simulações com as RNAs MLP os critérios de parada adotados foram: número de épocas maior que um limiar θ , erro quadrático médio (EQM) do conjunto de treino menor que um limiar γ ou aumento do EQM do conjunto de validação (verificado a cada 10 épocas).

Os parâmetros de cada RNA foram empiricamente calibrados e são apresentados na Tabela 1.

Tabela 1. Parâmetros das RNAs.

Parâmetro	MLP-GD	MLP-LM
Limite máximo de iterações θ	10000	500
Limite mínimo do EQM γ	0.001	0.001
Passo de aprendizagem α	0.005	0.001
Número de neurônios na camada intermediária	100	50

4.1.2. Máquinas de vetores de suporte

O SVM foi implementado utilizando a biblioteca LIBSVM [Chang e Lin 2011] disponível para a ferramenta MATLAB. Foram feitas simulações com as funções de *kernel* linear, RBF, sigmoidal e polinomial. A técnica de *grid search* foi empregada para a definição dos parâmetros. Porém, nas SVMs com *kernel* polinomial e sigmoidal, que possuem um maior número de parâmetros a serem definidos, optou-se por realizar a *grid search* apenas sobre os parâmetros C e γ, devido ao excessivo custo computacional. Neste caso, adotou-se os valores padrões da LIBSVM para os demais parâmetros.

A *grid search* foi realizada com um conjunto de treino (80% dos dados) e teste (20% dos dados), escolhidos aleatoriamente para cada configuração de classificação. Depois de executado, os melhores parâmetros foram escolhidos e usados para realizar os experimentos com o SVM. Nesses experimentos, todos os tipos de *kernel* foram avaliados. Porém, optou-se por apresentar apenas os resultados obtidos com o *kernel* RBF, uma vez que, este obteve o melhor desempenho. A Tabela 2 apresenta os parâmetros usados nas simulações com o método SVM.

Tabela 2. Parâmetros usados no método SVM com *kernel* RBF, obtidos por *grid search*.

Tipos de vetores de características	Classes balanceadas?	C	γ
Conteúdo	Sim	2^{14}	2^{3}
Conteúdo	Não	2^{15}	2^{3}
Links	Sim	2^{15}	2^{-1}
Links	Não	2^{15}	2^{-1}
Links Transformados	Sim	2^{5}	2^{-5}
Links Transformados	Não	2^{10}	2^{-9}
Links + conteúdo	Sim	2^{14}	2^{-4}
Links + conteúdo	Não	2^{15}	2^{-2}

4.1.3. Demais métodos

Os demais classificadores foram implementados usando a ferramenta WEKA [Hall et al. 2009]. Os métodos AdaBoost e *bagging* foram treinados com 100 iterações, sendo que ambos empregam agregação de múltiplas versões do método C4.5. Para os demais métodos foram empregados os parâmetros padrões da ferramenta utilizada.

5. Resultados

Nessa seção, são apresentados os resultados da detecção automática de *spam hosts* obtidos pelos métodos de aprendizado de máquina considerados os melhores atualmente disponíveis [Wu et al. 2008]. Para cada método e conjunto de vetores de características, foram feitas simulações usando classes desbalanceadas, conforme originalmente fornecido na competição de *web spam*, sendo 6.509 *hosts* (76,6%) da classe *ham* e 1.978 (23,4%) da classe *spam*. Em seguida, para avaliar se o desbalanceamento dos dados interfere na acurácia dos métodos, também foram realizados experimentos usando o mesmo número de amostras em cada classe. Neste caso, ambas as classes passaram a ter 1.978 representantes cada, aleatoriamente selecionados.

As Tabelas 3 e 4 apresentam os resultados obtidos por cada método de classificação explorando as características baseadas no conteúdo, nos *links*, nos *links* transformados e na combinação de *links* com o conteúdo. Os resultados estão ordenados pelo valor da F-medida. Os valores em negrito indicam os melhores resultados para cada tipo de atributo. Os valores em negrito precedidos pelo símbolo "*" indicam os melhores resultados considerando todas as simulações.

De maneira geral, os métodos avaliados obtiveram bons resultados, pois foram capazes de detectar com alta precisão uma quantidade expressiva de *spam hosts*, independentemente da característica empregada. Além disso, os resultados indicam que entre as técnicas avaliadas, o método de *bagging*, na média, apresentou os melhores resultados. Ele foi capaz de detectar em média 83,5% dos *spam hosts* com uma precisão média de 82,9%, independente do tipo de atributo utilizado. Por outro lado, o classificador SVM, em média, apresentou o pior desempenho. Outro método que merece destaque é o método naive Bayes que diferente dos outros métodos, nos experimentos com classes desbalanceadas, obteve uma ótima taxa de sensitividade, próxima a 100%. Porém esse resultado foi obtido a custa de muitos falsos positivos e por isso, a taxa de precisão foi baixa.

Os resultados também indicam que os classificadores obtiveram melhor desempenho quando foram treinados usando classes balanceadas. Portanto, verifica-se que, em ge-

Tabela 3. Resultados obtidos na detecção de *spam hosts* usando classes desbalanceadas.

	Sensitividade	Precisão	F-medida
	Atributos extraídos do conteúdo		
Bagging	68.7±2.3	**84.4**±1.9	**0.757**±0.014
Floresta aleatória	65.7±4.4	83.4±3.7	0.734±0.024
MLP-LM	69.3±4.2	77.6±4.6	0.731±0.032
AdaBoost	66.6±3.2	78.4±2.3	0.720±0.024
IBK	64.6±1.3	72.8±2.0	0.685±0.011
C4.5	67.7±4.6	68.0±0.6	0.678±0.024
MLP-GD	57.0±4.6	77.5±2.7	0.656±0.039
LogitBoost	54.2±3.9	71.6±3.2	0.616±0.023
SVM	36.0±2.4	76.2±2.8	0.488±0.023
OneR	38.3±2.4	65.0±2.7	0.482±0.023
Naive Bayes	***97.4**±1.0	25.4±1.3	0.402±0.015
	Atributos extraídos dos *links*		
Bagging	79.2±1.6	**77.2**±2.0	**0.781**±0.009
Floresta aleatória	76.5±7.9	77.0±3.7	0.764±0.027
AdaBoost	71.8±1.8	74.4±2.2	0.731±0.017
C4.5	73.2±2.1	72.3±1.4	0.727±0.011
MLP-LM	70.8±6.6	74.1±3.7	0.723±0.049
LogitBoost	71.8±4.0	71.4±2.1	0.715±0.009
MLP-GD	61.6±3.1	75.3±2.9	0.677±0.026
IBK	67.8±2.8	64.2±2.4	0.659±0.023
Naive Bayes	**94.6**±1.3	46.5±1.2	0.624±0.012
SVM	47.1±2.5	62.8±1.9	0.538±0.021
OneR	49.4±3.5	54.7±1.9	0.518±0.023
	Atributos extraídos dos links transformados		
Bagging	78.9±1.2	**77.6**±2.1	**0.782**±0.014
Floresta aleatória	76.5±3.1	75.8±4.1	0.760±0.012
MLP-LM	74.9±3.9	76.1±2.1	0.754±0.027
SVM	76.6±2.3	73.9±1.6	0.752±0.012
MLP-GD	75.2±2.2	73.9±3.1	0.745±0.014
AdaBoost	72.6±3.0	74.9±1.9	0.737±0.019
LogitBoost	70.2±4.9	72.3±3.2	0.711±0.019
C4.5	73.5±1.8	68.5±2.2	0.709±0.016
IBK	67.5±2.4	67.3±3.5	0.673±0.011
OneR	64.3±2.1	63.2±2.8	0.637±0.018
Naive Bayes	**96.5**±1.0	26.2±0.3	0.412±0.004
	Combinação dos atributos extraídos do conteúdo e dos links		
AdaBoost	80.4±1.9	***86.3**±1.2	***0.832**±0.014
MLP-LM	81.7±2.0	84.4±1.9	0.830±0.008
Bagging	81.1±1.8	84.0±1.7	0.825±0.016
Floresta aleatória	74.4±2.5	85.0±2.4	0.793±0.006
MLP-GD	74.2±3.2	79.5±2.5	0.767±0.016
C4.5	75.9±2.3	73.8±0.9	0.748±0.014
LogitBoost	70.2±2.6	77.7±2.3	0.737±0.019
IBK	68.7±2.3	72.5±2.0	0.705±0.015
Naive Bayes	**95.3**±1.0	45.9±0.8	0.620±0.008
SVM	52.4±2.0	69.6±2.5	0.598±0.018
OneR	50.1±2.4	54.4±2.2	0.521±0.018

ral, os métodos de classificação analisados tendem a favorecer a classe com maior número de representantes apresentados na fase de treinamento. Isso pode ser confirmado, pois os valores da sensitividade obtidos pelos métodos usando classes desbalanceadas foram, em geral, mais baixos, o que mostra que os classificadores erraram mais na identificação dos padrões pertencentes à classe *spam*.

Note que, o melhor resultado médio, considerando todas as técnicas, foi obtido no cenário em que foram utilizados os vetores de características baseados em *links* transformados (Tabela 4). Porém, o maior valor de F-medida foi obtido pelo método *AdaBoost*

Tabela 4. Resultados obtidos na detecção de *spam hosts* usando classes balanceadas.

	Sensitividade	Precisão	F-medida
	Atributos extraídos do conteúdo		
MLP-LM	**86.5**±2.0	***89.1**±2.4	**0.877**±0.017
Bagging	83.5±2.3	86.1±1.3	0.848±0.013
MLP-GD	82.9±2.0	86.1±2.4	0.845±0.015
Floresta aleatória	81.8±3.3	83.4±2.3	0.825±0.015
AdaBoost	81.8±2.5	83.3±1.6	0.825±0.014
IBK	77.0±2.6	81.4±1.5	0.791±0.017
C4.5	79.2±2.5	78.3±1.6	0.787±0.012
LogitBoost	77.0±3.2	78.9±1.1	0.779±0.015
Naive Bayes	83.6±17.7	62.8±10.2	0.694±0.039
SVM	60.6±2.7	76.7±1.5	0.677±0.019
OneR	61.6±1.0	68.3±2.0	0.648±0.013
	Atributos extraídos dos *links*		
Bagging	91.7±1.9	84.9±1.4	**0.882**±0.013
Floresta aleatória	88.8±2.5	**87.1**±1.6	0.879±0.010
MLP-LM	92.1±3.8	82.3±2.6	0.868±0.019
AdaBoost	87.7±1.7	83.9±0.8	0.858±0.009
MLP-GD	90.2±2.6	80.0±2.9	0.848±0.019
LogitBoost	88.0±3.7	80.4±1.0	0.840±0.018
C4.5	85.4±2.2	82.3±1.4	0.838±0.011
Naive Bayes	**94.8**±0.9	73.7±2.0	0.829±0.014
OneR	88.7±1.0	77.5±1.5	0.827±0.010
IBK	80.8±1.4	77.1±0.8	0.789±0.003
SVM	77.1±2.2	75.8±1.7	0.765±0.014
	Atributos extraídos dos links transformados		
Bagging	91.3±1.6	**86.2**±1.1	**0.886**±0.011
AdaBoost	88.8±0.9	85.9±1.7	0.873±0.011
MLP-GD	89.6±2.2	85.9±2.0	0.877±0.019
Floresta aleatória	89.5±3.7	86.0±1.7	0.876±0.016
MLP-LM	87.5±4.0	85.8±3.3	0.866±0.027
SVM	88.5±1.7	84.4±1.3	0.864±0.007
LogitBoost	87.6±3.7	82.7±1.0	0.850±0.021
C4.5	85.6±2.4	83.1±1.8	0.844±0.020
OneR	86.5±2.0	80.5±1.2	0.834±0.008
IBK	81.6±0.7	80.2±0.4	0.809±0.002
Naive Bayes	***96.8**±0.8	56.6±3.2	0.714±0.025
	Combinação dos atributos extraídos do conteúdo e dos *links*		
AdaBoost	92.6±1.4	**88.5**±0.9	***0.905**±0.005
Bagging	**93.2**±0.8	87.8±1.0	0.904±0.005
MLP-GD	92.6±2.4	86.7±2.1	0.895±0.013
Floresta aleatória	92.1±2.1	86.9±1.2	0.894±0.008
MLP-LM	91.9±3.5	85.6±2.7	0.886±0.021
LogitBoost	89.9±1.3	84.3±1.2	0.870±0.005
C4.5	86.9±2.2	85.1±0.9	0.860±0.009
IBK	84.7±2.7	82.4±1.0	0.835±0.012
Naive Bayes	92.7±6.5	75.2±3.5	0.828±0.016
OneR	88.8±1.7	77.5±1.7	0.827±0.012
SVM	71.4±1.7	78.7±2.3	0.749±0.015

usando a combinação de características baseadas no conteúdo da página e na relação dos *links*. Portanto, é importante observar que, a escolha dos atributos mais adequados varia de acordo com a técnica de aprendizagem de máquina escolhida.

Para facilitar a avaliação dos métodos analisados, é apresentado na Tabela 5 uma comparação entre os melhores resultados obtidos nesse trabalho e os disponíveis na literatura de *web spamming*. Para oferecer uma comparação justa, os métodos propostos na literatura foram treinados com a mesma base de dados, atributos e configurações usados nos métodos apresentados nesse trabalho. Foram usados exa-

tamente os mesmos parâmetros descritos nos trabalhos propostos na literatura ou os parâmetros padrão da ferramenta ou biblioteca utilizada. Em resumo, os métodos de *bagging* de árvores de decisão [Castillo et al. 2007, Ntoulas et al. 2006] e *boosting* de árvores de decisão [Ntoulas et al. 2006] foram implementados usando a ferramenta WEKA. Já, as máquinas de vetores de suporte [Svore et al. 2007] foram implementadas usando a biblioteca LIBSVM na ferramenta MATLAB. Por outro lado, método de programação genética [Shengen et al. 2011] não foi reimplementado porque os autores adotaram a mesma base de dados usada nesse trabalho, logo, para a comparação foram usados os resultados originais fornecidos pelos autores.

Tabela 5. Comparação entre os melhores resultados obtidos nesse trabalho e os resultados disponíveis na literatura

Classifiers	Precisão	Sensitividade	F-medida.
Resultados disponíveis na literatura			
Bagging [Castillo et al. 2007, Ntoulas et al. 2006] - conteúdo	81.5	68.0	0.741
Máquinas de vetores de Suporte [Svore et al. 2007] - conteúdo	55.5	**86.5**	0.677
Boosting [Ntoulas et al. 2006] - conteúdo	78.2	68.2	0.728
Programação genética [Shengen et al. 2011] - links	69.8	76.3	0.726
Programaçao genética [Shengen et al. 2011] - links transformados	76.5	81.4	0.789
Melhores resultados obtidos nesse trabalho			
Bagging - conteúdo+*links*	84.0	81.1	0.825
AdaBoost - conteúdo+*links*	**86.3**	80.4	**0.832**
Bagging - conteúdo+*links* (classes balanceadas)	87.8	**93.2**	0.904
AdaBoost - conteúdo+*links* (classes balanceadas)	**88.5**	92.6	**0.905**

É importante observar na tabela 5 que os métodos *bagging* e *AdaBoost* obtiveram desempenho superior aos métodos propostos em outros trabalhos da literatura. Portanto, é conclusivo que técnicas de aprendizado de máquina podem ser empregadas com sucesso para auxiliar o processo de detecção automática de *spam hosts*.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma análise de desempenho dos métodos de classificação mais empregados na literatura para auxiliar na tarefa de detecção automática de *spam hosts*.

Os resultados dos experimentos mostraram que todas as técnicas avaliadas oferecem bons resultados, independente do tipo de atributo utilizado. Isso demonstra que além de eficientes, os métodos apresentados possuem boa capacidade de generalização.

Dentre os métodos avaliados, as técnicas de agregação de métodos classificadores baseados em árvores, tais como *bagging* e *AdaBoost*, obtiveram os melhores desempenhos, demonstrando serem adequadas para auxiliar na detecção de *spam hosts*.

Com relação aos vetores de características, os melhores resultados foram obtidos quando os métodos classificadores utilizaram a combinação de informações extraídas do conteúdo das páginas e relação de *links*. É importante destacar também que, as técnicas avaliadas demonstraram ser mais eficientes quando treinadas com número igual de representantes em cada classe, pois ficou evidente que a classificação torna-se tendenciosa para a classe com maior número de amostras usada na etapa de treinamento.

Trabalhos futuros compreendem o estudo de formas de adaptar os métodos mais promissores para otimizar seu desempenho, a análise das características para verificar a viabilidade do emprego de técnicas de seleção de atributos, além da proposição de novos tipos de atributos que possam aumentar a capacidade de predição dos algoritmos.

Agradecimentos

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

Referências

Aha, D. W., Kibler, D., e Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Becchetti, L., Castillo, C., Donato, D., Leonardi, S., e Baeza-Yates, R. (2006). Using rank propagation and probabilistic counting for link-based spam detection. In *Proc. of the WebKDD'06*, Philadelphia,USA.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford Press, Oxford.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Castillo, C., Donato, D., e Gionis, A. (2007). Know your neighbors: Web spam detection using the web topology. In *Proc. of the 30th SIGIR*, pages 423–430, Amsterdam, The Netherlands.

Chang, C.-C. e Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27.

Cortes, C. e Vapnik, V. N. (1995). Support-vector networks. In *Machine Learning*, pages 273–297.

Freund, Y. e Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proc. of the 13th ICML*, pages 148–156, Bari, Italy. Morgan Kaufmann.

Friedman, J., Hastie, T., e Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407.

Gyongyi, Z. e Garcia-Molina, H. (2005). Spam: It's not just for inboxes anymore. *Computer*, 38(10):28–34.

Gyongyi, Z., Garcia-Molina, H., e Pedersen, J. (2004). Combating web spam with trustrank. In *Proc. of the 30th VLDB*, pages 576–587, Toronto, Canada.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., e Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New York, NY, USA, 2th edition.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90.

Hsu, C.-W., Chang, C.-C., e Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, National Taiwan University.

Jayanthi, S. K. e Sasikala, S. (2012). WESPACT: Detection of web spamdexing with decision trees in GA perspective. In *Proc. of the PRIME'12*, pages 381–386.

John, G. H. e Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proc. of the 11th UAI)*, pages 338–345, Montreal, Quebec;, Canada.

John, J. P., Yu, F., Xie, Y., Krishnamurthy, A., e Abadi, M. (2011). deSEO: combating search-result poisoning. In *Proc. of the 20th SEC*, pages 20–20, Berkeley, CA, USA.

Largillier, T. e Peyronnet, S. (2012). Webspam demotion: Low complexity node aggregation methods. *Neurocomputing*, 76(1):105–113.

Ledford, J. L. (2009). *Search Engine Optimization Bible*. Wiley Publishing, Indianapolis, Indiana, USA, 2th edition.

Liu, Y., Chen, F., Kong, W., Yu, H., Zhang, M., Ma, S., e Ru, L. (2012). Identifying web spam with the wisdom of the crowds. *ACM Trans. on the Web*, 6(1):2:1–2:30.

Ntoulas, A., Najork, M., Manasse, M., e Fetterly, D. (2006). Detecting spam web pages through content analysis. In *Proc. of the WWW*, pages 83–92, Edinburgh, Scotland.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA, USA, 1th edition.

Rungsawang, A., Taweesiriwate, A., e Manaskasemsak, B. (2011). Spam host detection using ant colony optimization. In *IT Convergence and Services*, volume 107 of *Lecture Notes in Electrical Engineering*, pages 13–21. Springer Netherlands.

Shao, J. (1993). Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):486–494.

Shengen, L., Xiaofei, N., Peiqi, L., e Lin, W. (2011). Generating new features using genetic programming to detect link spam. In *Proc. of the ICICTA'11*, pages 135–138, Shenzhen, China.

Silva, R. M., Almeida, T. A., e Yamakami, A. (2012a). Artificial neural networks for content-based web spam detection. In *Proc. of the 14th International Conference on Artificial Intelligence (ICAI'12)*, pages 1–7, Las Vegas, NV, USA.

Silva, R. M., Almeida, T. A., e Yamakami, A. (2012b). Redes neurais artificiais para detecção de web spams. In *Anais do 8th Simpósio Brasileiro de Sistemas de Informação (SBSI'12)*, pages 636–641, São Paulo, Brazil.

Svore, K. M., Wu, Q., e Burges, C. J. (2007). Improving web spam classification using rank-time features. In *Proc. of the 3rd AIRWeb*, pages 9–16, Banff, Alberta, Canada.

Taweesiriwate, A., Manaskasemsak, B., e Rungsawang, A. (2012). Web spam detection using link-based ant colony optimization. In *Proc. of the 26th AINA*, pages 868–873.

Witten, I. H. e Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2th edition.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., e Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.

