Filtragem Adaptativa de *Spam* com o Princípio *Minimum Description Length*

Ígor Assis Braga, Marcelo Ladeira

Departamento de Ciência da Computação
Universidade de Brasília (UnB) – Brasília, DF – Brasil

igorab@icmc.usp.br, mladeira@unb.br

Abstract. *The spreading of the spamming practice has motivated the adoption of filters that learn how to separate unwanted messages from the legitimate ones. This paper presents a classification scheme based on the Minimum Description Length (MDL) principle. Experiments conducted on TREC 2005 and TREC 2006 public email corpora show that the MDL classification results compare to those of state-of-the-art techniques, however, outperforming them with respect to execution speed. The good performance of MDL-based classification has led us to develop UnBeaten: a spam filter aimed at email users and integrated to the Mozilla Thunderbird mailer.*

Resumo. *A proliferação da prática de spamming no universo do correio eletrônico vem motivando a adoção de filtros que aprendem a fazer a separação automática entre as mensagens legítimas e as indesejadas. Este artigo apresenta um esquema de classificação baseado no princípio Minimum Description Length (MDL). Experimentos realizados nas bases de mensagens TREC 2005 e TREC 2006 mostram que o esquema tem resultados comparáveis aos de técnicas no estado da arte, se destacando, no entanto, em relação à rapidez no processamento de mensagens. O bom desempenho da classificação por MDL inspirou a criação do UnBeaten: um filtro com foco em usuários de correio eletrônico e integrado ao leitor de e-mail Mozilla Thunderbird.*

1. Introdução

Spam, mensagens indesejadas partindo de remetentes que não possuem ligação alguma com os destinatários, são disparadas para um grande número de contas por indivíduos conhecidos como *spammers*, constituindo, aproximadamente, 70% das mensagens que circulam hoje pela Internet [1]. No lado do usuário, quanto mais mensagens inúteis se misturam às mensagens legítimas nas caixas de entrada, mais tempo é perdido na tentativa de separá-las. Já em relação aos provedores de Internet e às empresas que disponibilizam o serviço de *e-mail*, há o desperdício de *links* de comunicação e de recursos computacionais com a transmissão, o processamento e o armazenamento de mensagens que vão, com certeza, ser ignoradas pelos seus usuários.

A versatilidade do protocolo de *e-mail* SMTP proporcionou aos *spammers* a capacidade de mandar mensagens a muitos usuários por um custo muito baixo, gerando lucro mesmo com uma baixa taxa de resposta. No entanto, não é desejável acabar com a

[1] Dados da empresa de segurança digital Symantec – http://www.symantec.com

possibilidade de se enviar mensagens a diversos usuários simultaneamente. Assim, algumas técnicas de controle que atuam no momento da transmissão da mensagem vêm sendo desenvolvidas, dentre elas as listas de bloqueio baseadas em DNS [Jung e Sit 2004] e os protocolos de autenticação DKIM [Leiba e Fenton 2007] e SPF. Apesar de promissoras, essas técnicas não tem sido amplamente utilizadas, pois requerem alterações nos servidores de recepção e de envio de mensagens .

Enquanto uma solução que diminua os incentivos ao *spamming* não se propaga, mecanismos de filtragem por conteúdo são empregados para fazer a separação automática entre as mensagens legítimas e as indesejadas. As técnicas mais recentes de filtragem envolvem o uso de algoritmos de aprendizado de máquina juntamente com técnicas de mineração de textos para treinar classificadores que aprendam com mensagens previamente julgadas pelo usuário. Isso permite a criação automática de modelos personalizados do que é uma mensagem indesejada e do que não é.

A filtragem automática de *spam* constitui uma área de pesquisa ativa e desafiadora pois os *spammers* sempre tentam burlar os filtros utilizando novas técnicas de ofuscação de conteúdo, além de haver a possibilidade da desastrosa ocorrência de falsos positivos (isto é, classificações inadvertidas de mensagens legítimas como *spam*). Esses desafios fazem com que seja necessário o emprego de bons algoritmos adaptativos, que se atualizam rapidamente frente às mudanças nas mensagens. O esquema de classificação baseado no princípio *Minimum Description Length* (MDL) [Grünwald 2005], proposto neste artigo, constitui uma contribuição nesse sentido.

Na Seção 2, abordaremos alguns trabalhos correlatos. O esquema de classificação proposto está detalhado na Seção 3. Os experimentos conduzidos nas bases TREC 2005 e TREC 2006 estão descritos na Seção 4, enquanto que os resultados obtidos e as comparações com técnicas relacionadas são apresentados na Seção 5. Por último, na Seção 6, são descritas as principais características do filtro UnBeaten, uma ferramenta desenvolvida para os usuários do leitor de *e-mail* Mozilla *Thunderbird* e que coloca em prática a classificação por MDL.

2. Trabalhos Relacionados

A filtragem de *spam* tem uma interpretação clara como um problema de classificação em aprendizado supervisionado. Neste caso, o aprendiz é um classificador e o conjunto de treinamento consiste de exemplos de mensagens de *e-mail* com a indicação da classe, *spam* ou legítima, de cada mensagem. A tarefa do classificador é determinar a classe de uma nova mensagem a partir das informações de treinamento oferecidas a ele. Por razões de eficiência, trabalha-se com uma representação do conteúdo de uma mensagem. A obtenção dessa representação é chamada de extração de características. Se compararmos o classificador a uma função que leva mensagens a rótulos de classes, as características da mensagem são os parâmetros dessa função. O método de extração mais utilizado para mensagens de *e-mail* é utilizar caracteres de espaço e/ou de pontuação como delimitadores para quebrar o texto em palavras e usá-las como características.

O uso de algoritmos de aprendizado de máquina para a filtragem de *spam* foi proposto primeiramente por [Sahami et al. 1998], que introduziu o classificador *naïve* Bayes para essa tarefa. As primeiras implementações práticas não vieram até que [Graham 2002] detalhasse a técnica com indicações mais diretas de como estimar os parâmetros do mod-

elo. Algum tempo depois, [Metsis et al. 2006] fizeram uma comparação da eficácia de variantes do *naïve* Bayes na filtragem de *spam*. Eles mostraram que o *naïve* Bayes multinomial (MNB), utilizando atributos binários, era superior às outras variantes. Em combinação com um classificador *naïve* Bayes, foi utilizado em [Siefkes et al. 2004] um método de extração de características chamado *Orthogonal Sparse Bigrams* (OSB). Esse último visa modelar o contexto em que as palavras foram usadas nas mensagens.

Algoritmos baseados em *Support Vector Machines* (SVM) são considerados o estado da arte em tarefas de classificação de textos. A filosofia de SVM é escolher o classificador linear mais simples, dentre aqueles que se ajustam bem aos dados de treinamento, o que torna a técnica bastante atrativa para uso em problemas com entradas de alta dimensionalidade. Os únicos senões de SVM são a dificuldade em se fazer o treinamento incremental e a necessidade de se armazenar mensagens antigas. Em [Sculley e Wachman 2007], é apresentado o RO-SVM (*Relaxed Online* SVM) como uma possível solução para o treinamento incremental aplicado à filtragem de *spam*. Apesar das reduções consideráveis no tempo de treinamento reportados nas bases TREC, o custo do uso de SVM ainda continua sendo muito alto.

Recentemente, uma abordagem utilizando o modelo estatístico de compressão de dados *Prediction by Partial Matching* (PPM) mostrou-se ser adequada para a filtragem de *spam*, principalmente porque essa técnica opera sobre seqüências de caracteres [Bratko et al. 2006]. Isso é particularmente interessante para o problema em questão, já que existe uma tendência de que as mensagens venham com seu conteúdo ofuscado com espaços e pontuações para poder quebrar o processo de extração de palavras. Os autores concluem, no entanto, que o emprego desses modelos requer a disponibilidade de muita memória, o que acaba comprometendo a escalabilidade em número de usuários. Neste artigo, introduzimos uma abordagem de classificação por MDL similar à de Bratko et al., porém usando o tradicional processo de extração de palavras das mensagens.

3. Classificação por MDL

O princípio MDL – *Minimum Description Length*– oferece uma solução geral ao problema de seleção de modelos [Grünwald 2005]. Quando vários modelos competem pela explicação de um conjunto de dados, o princípio diz que devemos selecionar aquele modelo que melhor comprime os dados. Como a capacidade de comprimir um conjunto de dados está diretamente relacionada à descoberta de regularidades nesses dados, pode-se dizer que mais compressão implica em um maior aprendizado sobre os dados.

Dado um conjunto de treinamento $\left\{(M_i, C_i)_{i=1}^{N}\right\}$, o critério de classificação por MDL testa, para cada classe $C \in \{spam, legitima\}$, a hipótese de que a classe de uma nova mensagem M é C. Isso é feito com a inserção da mensagem no subconjunto de treinamento de cada classe e com a verificação do aumento no tamanho da descrição do conjunto de treinamento total. A hipótese escolhida, e portanto a classe da nova mensagem, será aquela que causar o menor aumento no tamanho de descrição dos dados.

Quando uma nova mensagem M chega, ela é comprimida com os modelos das classes *spam* e legítima. O modo como construímos esse modelo é o que diferencia a nossa abordagem da apresentada em [Bratko et al. 2006]. Cada classe C é vista como uma seqüência de palavras extraídas das mensagens e inseridas no conjunto de treinamento. Para cada palavra de M, atribuímos um tamanho de código L que é calculado baseado

na seqüência de palavras que já ocorreram anteriormente nas mensagens do conjunto de treinamento de C. O tamanho de M em C é a soma do tamanho dos códigos das palavras contidas em M. Da Teoria da Informação, sabemos que podemos obter um código prefixado (e passível de decodificação direta) fazendo $L_{palavra} = -\log P_{palavra}$, sendo P uma distribuição de probabilidades sobre as palavras da classe. Seja $n_C(i)$ o número de vezes que uma palavra i aparece nas mensagens anteriores de uma classe C, então a probabilidade de uma palavra qualquer ocorrer nesta classe é obtida usando o estimador de máxima verossimilhança

$$\frac{n_C(palavra)+\frac{1}{|\chi|}}{n_C+1},$$

sendo n_C a soma de $n_C(i)$ para cada cada palavra i que aparece nas mensagens anteriores de C e $|\chi|$ o tamanho do vocabulário (que aqui assumimos ser 2^{32}, isto é, cada palavra é um símbolo de 32 bits na sua forma não comprimida). Esse estimador reserva uma "porção" de probabilidade a palavras ainda não vistas.

Uma boa característica desse esquema é que é possível começar com um conjunto de treinamento vazio e ir construindo os modelos de cada classe à partir do zero, conforme o *feedback* do usuário. Além disso, não é necessário guardar as mensagens do conjunto de treinamento para classificar novas mensagens, pois esses modelos são construídos incrementalmente a partir da contagem da freqüência do aparecimento das palavras nas mensagens. Esse abordagem também pode ser estendida para além da classificação binária, bastando-se criar mais modelos para representar as outras classes.

Um mecanismo que pode ser utilizado para controlar a taxa de falsos positivos constitui em se definir um limiar (*threshold*) e utilizá-lo para classificar uma mensagem como *spam*, por exemplo, somente se uma medida de ordem da mensagem for maior que o valor do limiar. Nesta pesquisa, uma ordem O foi derivada do tamanho da mensagem comprimida nos dois modelos. Seja L_1 o menor desses tamanhos e L_2 o outro tamanho. Então definimos $O = c \times \left(1 - \left(\frac{L_1}{L_2}\right)\right)$, sendo $c = -1$ quando a mensagem é classificada como legítima e $c = 1$ caso contrário. Essa ordem foi utilizada somente para avaliar nosso esquema, mas pode ser usada também quando há uma classificação com custos.

4. Experimentos

As bases de mensagens de *e-mail* TREC 2005 [Cormack e Lynam 2005a] e TREC 2006 [Cormack 2006a] foram utilizadas para avaliar o desempenho da técnica de filtragem por MDL. Essas bases se originaram do *Spam Track*, um evento da conferência anual TREC (*Text Retrieval Conference*), no qual filtros aprendizes desenvolvidos pela indústria e por grupos de pesquisa podem ser submetidos e avaliados em uma competição.

A base TREC 2005 é composta de 39399 mensagens legítimas (42,7%) e 52790 mensagens de *spam* (57,3%). As mensagens legítimas vieram das contas que se tornaram públicas de 150 empregados da Enron, empresa americana do setor de energia que sofreu investigação federal após decretar falência. As mensagens de *spam* foram coletadas em 2005 e alteradas para parecerem ter sido enviadas ao servidor de *e-mail* da Enron. Já a base TREC 2006 é composta de 12910 mensagens legítimas (34,1%) e 24912 mensagens de *spam*, todas coletadas da *web* em maio de 2006.

As mensagens foram processadas pelos organizadores a fim de se remover qualquer traço que poderia dar a informação direta de uma classificação anterior. Além disso,

essas bases passaram por uma etapa de pré-processamento antes da realização dos experimentos. Com a ajuda do utilitário *normalizemime*[2], fizemos a decodificação de conteúdos em *base*64 e a codificação de todas as áreas de textos em UTF-8.

Através de ferramentas computacionais distribuídas com as bases, mensagens a serem classificadas são repassadas uma por uma, em ordem cronológica, ao filtro sendo testado. Este, por sua vez, devolve a classificação de cada mensagem juntamente com uma ordem que reflete o quanto cada mensagem é considerada como *spam* (quanto maior esse valor, maior a certeza de que a mensagem é indesejada). Logo após a classificação, a ferramenta fornece ao filtro a classe real a que pertence a mensagem recém-classificada. Esse *feedback* permite a evolução do filtro, principalmente em caso de erro.

Neste artigo, o classificador MDL foi avaliado considerando apenas uma vez as palavras contidas em uma mensagem. Essa abordagem é a recomendada em [Metsis et al. 2006] e [Drucker et al. 1999]. As palavras, nesse contexto, são obtidas através da quebra da cadeia formada pelos 3000 primeiros caracteres de uma mensagem nos pontos que possuem caracteres de espaço. Esse mesmo método de extração foi utilizado na nossa execução do MNB e na execução de SVM em [Sculley e Wachman 2007]. Para essas três técnicas, a comparação fica, então, mais justa. O mesmo não acontece com os filtros mais bem colocados nas duas *Spam Tracks* e com o software livre *Bogofilter* 1.1.5, que também têm seus desempenhos comparados à classificação por MDL.

4.1. Critérios de Avaliação

As ordens e as classificações retornadas para cada mensagem são utilizadas na avaliação dos classificadores. Daqui em diante, usaremos fn para indicar a taxa de falsos negativos e fp para indicar a taxa de falsos positivos. Para avaliar os filtros de uma maneira independente de limiar, utilizou-se uma variante da análise da curva ROC [Fawcett 2004]. Neste caso foi considerado o gráfico de $1 - fn$ em função de fp. A curva é obtida através da variação do limiar do classificador e a apuração dos valores (fn, fp) resultantes. A área acima da curva ROC ($1 - AUC$) representa a probabilidade de um classificador dar a uma mensagem legítima aleatória uma ordem maior que um *spam* aleatório. Sendo assim, quanto menor for esse valor, melhor é o classificador.

Há situações em que as curvas ROC de dois classificadores se cruzam, e, nesse caso, reportar ($1 - AUC$) não é suficiente. Assim, reportamos também fn quando fp é igual a $0,1\%$, ou seja, quanto *spam* é classificado erroneamente quando se permite ao filtro errar a classificação de uma mensagem legítima em 1000. Esse critério é importante porque nos interessa o desempenho da técnica para taxas de falsos positivos baixas.

4.2. Vencedores do *Spam Track*

Os melhores resultados reportados pelos organizadores do *Spam Track* 2005 e 2006 são reproduzidos aqui, mas eles só podem ser comparadas com o classificador MDL se olharmos o processo de filtragem como um todo, já que a extração de características é diferente. O filtro com cognome *ijsSPAM2* [Bratko e Filipič 2005] obteve os melhores resultados no *Spam Track* de 2005. Ele é baseado no modelo PPM em conjunto com o MDL.

Para o *Spam Track* 2006, o filtro baseado em PPM foi novamente submetido com o cognome *ijsS1*. Outra importante implementação é o *oflS1* [Assis et al. 2006], que, no

[2]Disponível em http://hyvatti.iki.fi/∼jaakko/spam

geral, obteve os melhores resultados no evento. O *oflS1* implementa o extrator de características OSB. As características obtidas são selecionadas de acordo com sua capacidade de discriminação entre as classes, e o classificador empregado é uma extensão do *naïve* Bayes. O treinamento no *oflS1* é feito somente nas mensagens em que houve erro de classificação e naquelas que estão mais próximas da região de decisão.

5. Resultados e Análise

Os resultados de todos os filtros participantes do TREC 2005 podem ser encontrados em [Cormack e Lynam 2005b], enquanto que os do TREC 2006 foram publicados em [Cormack 2006b]. Os resultados de SVM e tempos de execução relacionados foram disponibilizados por [Sculley e Wachman 2007], mas não foram reportados os valores de fn associados à taxa de $fp = 0,1\%$ e nem a configuração da máquina em que os experimentos foram executados. Ao lado de cada valor de $1 - AUC$ encontra-se, entre parênteses, o respectivo intervalo de confiança de 95%.

5.1. TREC 2005

Pela análise da Tabela 1, pode-se perceber que a classificação por MDL tem um bom desempenho na base TREC 2005, superando o *Bogofilter* e o MNB (que apresentou fn muito elevada nesta base). O intervalo de 95% de confiança dos três melhores resultados se sobrepõem, não sendo possível fazer uma distinção segura entre eles. Destacamos, porém, que o nosso filtro é melhor que o filtro SVM em relação ao tempo de execução, processando toda a base em aproximadamente 3 minutos (tempo de relógio) quando utilizamos um microcomputador Pentium 4, 3.0 GHz, com 1 GB de memória. O tempo de execução utilizando-se SVM na TREC 2005 foi de 1108 minutos de CPU. Alertamos que esses números devem ser interpretados somente para dar uma idéia da ordem de grandeza dos tempos gastos, pois os ambientes de execução das duas técnicas são diferentes.

Filtro	(1-AUC)%	$fn\%$ com $fp = 0,1\%$
SVM	0,015 (0,011 – 0,022)	–
PPM (ijsSPAM2)	0,019 (0,015 – 0,023)	1,78
MDL	**0,022 (0,018 – 0,027)**	**3,43**
Bogofilter	0,048 (0,038 – 0,062)	3,41
MNB	0,268 (0,251 – 0,290)	21,9

Tabela 1. Desempenho de classificadores na base TREC 2005

5.2. TREC 2006

A Tabela 2 mostra que o MDL tem um desempenho muito bom na base TREC 2006, também superior ao *Bogofilter* e ao MNB. Os classificadores MDL e SVM se destacam em relação ao PPM e ao OSB, mas não é justo falar que aqueles são melhores porque os filtros *oflS1* e *ijsS1* foram submetidos à competição antes de que qualquer pessoa tivesse acesso à base. Nesse caso e no caso da TREC 2005, os grupos responsáveis não puderam executar seus filtros nas bases para acertar uma configuração de parâmetros que pudesse levar a melhores resultados. Novamente, como já era esperado, o filtro baseado em SVM perde para o MDL em tempo de execução. Enquanto o filtro MDL processa toda a base em pouco mais de 1 minuto, o SVM leva 223 minutos.

Filtro	(1-AUC)%	$fn\%$ com $fp = 0,1\%$
MDL	**0,028 (0,022 – 0,037)**	**2,48**
SVM	0,034 (0,025 – 0,046)	–
OSB (oflS1)	0,054 (0,034 – 0,085)	2,49
PPM (ijsS1)	0,061 (0,048 – 0,076)	2,24
MNB	0,065 (0,054 – 0,078)	10,1
Bogofilter	0,087 (0,066 – 0,115)	3,45

Tabela 2. Desempenho de classificadores na base TREC 2006

6. O Filtro UnBeaten

O bom desempenho da classificação por MDL inspirou a criação do UnBeaten: um filtro com foco em usuários de correio eletrônico e integrado ao leitor de *e-mail* Mozilla *Thunderbird* 2.0, que é um software livre desenvolvido sob a supervisão da Fundação Mozilla. O UnBeaten foi projetado com duas partes independentes: o classificador e a interface com o usuário. O classificador foi implementado em Java 1.5, enquanto que a interface com o usuário foi desenvolvida como uma extensão do *Thunderbird*.

Além da filosofia de software livre, o desenvolvimento do *Thunderbird* adota os princípios da modularidade, da portabilidade, da extensibilidade e da separação entre a lógica do programa e a sua apresentação. Tudo isso é possível através de uma combinação entre as tecnologias XUL, XPCOM e *JavaScript*, que também são usadas para construir extensões. O XUL é empregado para criar interfaces com o usuário da aplicação. Já o XPCOM é responsável por disponibilizar as funcionalidades das aplicações Mozilla como componentes. Por último, o *JavaScript* dá ao desenvolvedor a possibilidade de modificar o comportamento do *Thunderbird* e acessar componentes XPCOM. Uma boa referência para essas tecnologias é o XUL *Planet*[3].

6.1. Interface com o Usuário

A Figura 1 mostra os elementos de interface com o usuário. Na parte superior, encontra-se o menu UnBeaten; no centro, uma mensagem de alerta; no canto inferior-esquerdo, as pastas inseridas no UnBeaten com o ícone em formato de "U"; e, no canto inferior-direito, o ícone e o rótulo de status do UnBeaten. Quando o ícone exibir a cor vermelha, é sinal de que a requisição informada no rótulo retornou com erro. Se o ícone exibir a cor alaranjada, significa que alguma condição anormal ocorreu. Quando a cor do ícone for verde, a requisição foi executada com sucesso.

O uso da classificação por MDL enriquece o UnBeaten com a possibilidade de se fazer uma filtragem multiclasses. Ao invés do esquema original de separação entre *spam* e mensagens legítimas, pode-se empregar, por exemplo, uma separação entre mensagens de esportes, de amigos, de outras legítimas e de *spam*. Isso pode ser particularmente útil para deixar mais concisas as classes de separação, implementadas como pastas locais no UnBeaten. A seguir, relatamos as principais operações realizadas pela ferramenta.

Manutenção de Pastas. Para inserir uma pasta no UnBeaten, o usuário precisa criá-la primeiro. A pasta deve ser criada como subpasta das "Pastas Locais". A pasta

[3]http://www.xulplanet.com

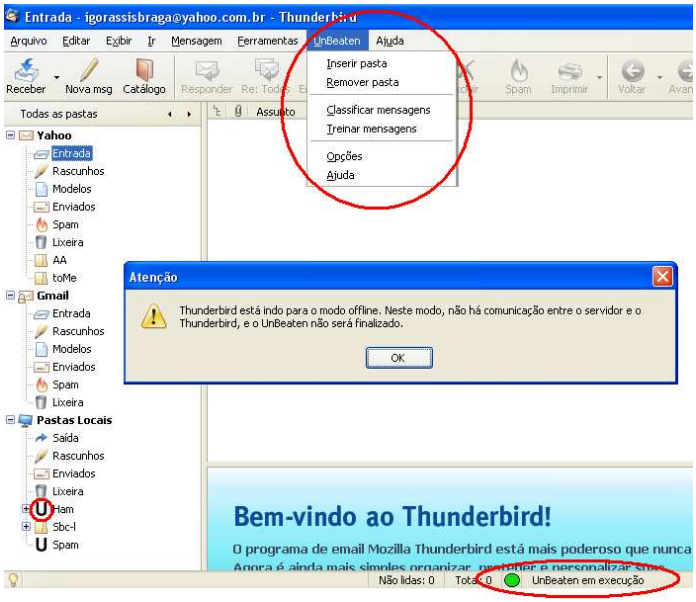

Figura 1. Elementos da interface gráfica do UnBeaten

"Spam" de "Pastas Locais", caso exista, também pode ser usada como uma pasta do UnBeaten. Depois de criada a pasta, é possível inserí-la através do comando "Inserir pasta" do menu UnBeaten. Para removê-la, pode-se usar o respectivo comando do menu. A renomeação ou movimentação da pasta não afeta o UnBeaten.

Treinamento. Há três maneiras de se treinar mensagens em uma pasta do UnBeaten. A primeira é selecionando as mensagens que já estão dentro da pasta e acionar o comando "Treinar mensagens" do menu UnBeaten. A segunda acontece automaticamente quando uma mensagem é classificada em uma pasta. A terceira maneira é movendo ou copiando as mensagens para a pasta. A pasta "Entrada" tem um significado especial no treinamento, pois mensagens movidas para ela causam o "destreinamento" da mensagem da pasta de origem. Quando o destino da mensagem não é uma pasta do UnBeaten e a de origem é, não há o "destreinamento" da mensagem (por exemplo, quando apagamos uma mensagem). Se ao invés de movida, uma mensagem for copiada de uma pasta do UnBeaten para qualquer outra pasta, também não haverá o "destreinamento".

Classificação. A classificação também pode ser feita de duas maneiras. A primeira acontece quando se usa o comando "Classificar mensagens" em mensagens selecionadas. A segunda, quando uma mensagem é recebida na pasta "Entrada" proveniente de uma conta POP3. Só é possível a classificação de mensagens recebidas de uma conta IMAP quando o usuário marca como *offline* a pasta "Entrada" da respectiva conta.

6.2. Integração

A extensão se comunica com o núcleo do UnBeaten através de uma conexão TCP/IP em uma porta que pode ser definida pelo usuário. A conexão é local porque o servidor que implementa o classificador é executado na máquina do usuário. O servidor e a extensão operam com um protocolo baseado em tarefas. As tarefas podem ser, por exemplo, classificar uma mensagem ou criar uma classe. Cada conexão feita pela extensão irá mandar somente uma requisição de tarefa para o servidor, o qual responde, encerra a conexão

e aguarda até que uma nova conexão seja feita. A resposta para todas as tarefas, com exceção da classificação, é um aviso de falha ou de sucesso na execução da tarefa.

A comunicação com o servidor, a partir da extensão, é feita através do objeto `XMLHTTPRequest`. Esse objeto *JavaScript* é um envoltório para o componente XPCOM correspondente. Com o `XMLHTTPRequest` é possível realizar conexões assíncronas com um servidor HTTP. Como somente a requisição da tarefa é o que importa, o servidor descarta os cabeçalhos HTTP.

7. Conclusão e Trabalhos Futuros

Apresentamos um esquema de classificação de mensagens de *e-mail* baseado no princípio *Minimum Description Length*. Essa técnica, quando testada sobre os conjuntos de mensagens TREC 2005 e 2006, obteve resultados comparáveis aos classificadores no estado da arte da filtragem de *spam*, como SVM. Porém, a classificação por MDL tem a grande vantagem de ser bastante rápida, processando, em média 400 mensagens por segundo na base TREC 2005, incluindo o tempo de extração das palavras das mensagens.

Inspirados pelo bom desempenho da técnica, criamos o filtro adaptativo de *spam* UnBeaten. O UnBeaten é usado como uma extensão no ambiente do leitor de *e-mail* Mozilla *Thunderbird*. O mecanismo de extensões do *Thunderbird* permitiu que a técnica de classificação, implementada em Java, fosse separada da interface com o usuário. Isso favorece trabalhos futuros que queiram modificar a técnica de classificação sem precisar modificar a interface ou vice-versa. O UnBeaten também permite a criação de múltiplas classes, o que o torna uma interessante ferramenta organizadora de mensagens. O leitor pode ter acesso à ferramenta fazendo um pedido por *e-mail* a um dos autores.

Como trabalho futuro, pensamos em explorar técnicas de aprendizado ativo (*active learning*) e aprendizado semi-supervisionado para a filtragem de *spam*. Em particular, queremos investigar o uso do algoritmo de classificação multivisão Co-Testing para a realização dessas tarefas [Matsubara et al. 2007]. Em uma outra frente, é necessário encontrar alternativas à extração de características usando palavras isoladas a fim de se modelar a interdependência das palavras dentro das mensagens de e-*mail*.

Agradecimentos

Este trabalho foi realizado com apoio parcial do PIC-UnB/CNPq.

Referências

Assis, F., Yerazunis, W., Siefkes, C., e Chhabra, S. (2006). Exponential Differential Document Count: A feature selection factor for improving bayesian filters accuracy. In *The Fifteenth Text REtrieval Conference (TREC) Proceedings*.

Bratko, A. e Filipič, B. (2005). Spam filtering using character-level Markov models: Experiments for the TREC 2005 spam track. In *The Fourteenth Text REtrieval Conference (TREC) Proceedings*, Gaithersburg, MD.

Bratko, A., Filipič, B., Cormack, G. V., Lynam, T. R., e Zupan, B. (2006). Spam filtering using statistical data compression models. *Journal of Machince Learning Research*, 7:2673–2698.

Cormack, G. (2006a). TREC 2006 spam track overview. In *The Fifteenth Text REtrieval Conference (TREC) Proceedings*, Gaithersburg, MD.

Cormack, G. (2006b). TREC 2006 spam track results. In *The Fifteenth Text REtrieval Conference (TREC) Proceedings*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec15/appendices/spam.results.html (Acessado em dezembro de 2007).

Cormack, G. e Lynam, T. (2005a). TREC 2005 spam track overview. In *The Fourteenth Text REtrieval Conference (TREC) Proceedings*, Gaithersburg, MD.

Cormack, G. e Lynam, T. (2005b). TREC 2005 spam track results. In *The Fourteenth Text REtrieval Conference (TREC) Proceedings*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec14/appendices/spam.results.html (Acessado em novembro de 2007).

Drucker, H., Wu, D., e Vapnik, V. (1999). Support Vector Machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054.

Fawcett, T. (2004). ROC graphs: Notes and practical considerations for researchers. Technical Report HPL-2003-4, HP Laboratories Palo Alto.

Graham, P. (2002). A plan for spam. http://www.paulgraham.com/spam.html (Acessado em novembro de 2007).

Grünwald, P. (2005). *A Tutorial Introduction to the Minimum Description Length Principle*. The MIT Press.

Jung, J. e Sit, E. (2004). An empirical study of spam traffic and the use of DNS black lists. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 370–375, New York, NY, USA. ACM.

Leiba, B. e Fenton, J. (2007). DomainKeys Identified Mail (DKIM): Using digital signatures for domain verification. In *Fourth Conference on Email and Anti-Spam (CEAS)*, Palo Alto, CA.

Matsubara, E. T., Monard, M. C., e Prati, R. C. (2007). *Exploring unclassified texts using multi-view semi-supervised learning*. Idea Publishing, Hershey, PA, USA.

Metsis, V., Androutsopoulos, I., e Paliouras, G. (2006). Spam filtering with naive Bayes – Which naive Bayes? In *Third Conference on Email and Anti-Spam (CEAS)*, Palo Alto, CA.

Sahami, M., Dumais, S., Heckerman, D., e Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.

Sculley, D. e Wachman, G. (2007). Relaxed online SVMs for spam filtering. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 415–422. ACM.

Siefkes, C., Assis, F., Chhabra, S., e Yerazunis, W. S. (2004). Combining Winnow and Orthogonal Sparse Bigrams for incremental spam filtering. In *PKDD '04: Proc of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 410–421, New York, NY, USA. Springer-Verlag New York, Inc.