EI - UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO DE E-MAILS SPAM UTILIZANDO REDES NEURAIS ARTIFICIAIS

JOÃO MARINHO DE CASTRO ASSIS

ITAJUBÁ, DEZEMBRO DE 2006

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO DE E-MAILS *SPAM* UTILIZANDO REDES NEURAIS ARTIFICIAIS

JOÃO MARINHO DE CASTRO ASSIS

Dissertação submetida ao corpo docente da coordenação dos programas de pós-graduação em Engenharia da Universidade Federal de Itajubá como parte dos requisitos necessários à obtenção do grau de Mestre em Ciências em Engenharia Elétrica.

Orientador: Otavio Augusto S. Carpinteiro, PhD.
Co-Orientador: Prof. B. Isaías de Lima Lopes, Dr.

Banca Examinadora:
Prof. Cairo Nascimento, PhD.
Prof. Edmilson Marmo Moreira, Dr.
Prof. Otavio Augusto S. Carpinteiro, PhD.
Prof. B. Isaías de Lima Lopes, Dr.

Itajubá, dezembro de 2006

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

A848d

Assis, João Marinho de Castro
Detecção de e-mails spam utilizando redes neurais artificiais / João Marinho de Castro. -- Itajubá, (MG) : UNIFEI, 2006.
112 p. : il.

Orientador : Prof. PhD. Otávio Augusto S. Carpinteiro.
Co-orientador : Prof. Dr. B. Isaías de Lima Lopes.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Detecção de spams. 2. Classificação de e-mails. 3. Redes neurais artificiais. 4. Inteligência artificial. I. Carpinteiro, Otávio Augusto S., orient. II. Lopes, B. Isaías de Lima, co-orient. III. Universidade Federal de Itajubá. IV. Título.

CDU 004.492.3(043)

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu oportunidade à vida.

A meus orientadores, pela ajuda nas horas mais difíceis e por sempre acreditarem no trabalho que realizava.

A minha família, pelo apoio incondicional.

A minha namorada, por estar sempre presente e pela grande ajuda.

Amigos e colegas, que me apoiaram e ajudaram em mais esta etapa.

Enfim, agradeço a todos que, de alguma forma, contribuíram para a realização desse trabalho.

RESUMO

O e-mail, a correspondência eletrônica da Internet, cresceu sensivelmente nos últimos anos e, simultaneamente, houve um aumento considerável no número de e-mails indesejados recebidos pelos usuários, os *spam* e-mails.

Os *spams* geralmente contêm publicidade não solicitada e, muitas vezes, apresentam informações falsas, sendo invariavelmente postados indiscrimidamente, podendo causar grandes prejuízos às pessoas e empresas.

Podem-se citar alguns problemas básicos com os spams: o desperdício de largura de banda da rede, a perda de tempo e de produtividade, a perda de e-mails legítimos ou normais, além de atrasos no recebimento de mensagens importantes.

Diversas soluções foram propostas na literatura, porém muitos resultados são algumas vezes ineficientes ou limitados na classificação de *spam* e-mails. Muitas vezes apresentam altas taxas de falsos positivos e falsos negativos, que é a classificação errada de e-mails normais e de *spams*, respectivamente.

A proposta deste trabalho é o desenvolvimento de um sistema anti-spam, com maior ênfase na pré-filtragem dos e-mails. O pré-filtro é o responsável por transformar as informações complexas, presentes em cada mensagem, em informações mais simples e mais uniformes, permitindo um melhor desempenho na classificação. Como agente classificador utilizou-se Redes Neurais Artificiais, o *multi-layer perceptron* ou MLP, cuja capacidade de generalização e adaptabilidade são características importantes para o problema em questão.

Diferentes métodos de seleção das características envolvidas e de modelos de redes

neurais MLP são apresentados na seção de resultados, que se mostraram bastante promissores, com casos em que houve 100% de classificação correta de e-mails normais e de *spams*.

ABSTRACT

The e-mail service has significantly grown in recent years and also the number of spam e-mails received by the users. A spam e-mail usually contains not requested advertising and it presents false information being sent deliberately. Spams can cause a great worry to people and companies, because they occupy significant Internet bandwidth and they increase the delay in receiving important messages.

Some solutions have been proposed in the literature; however some results are insufficient or limited, for example, the wrong classification of normal e-mails and spam e-mails as false positive or false negative, respectively.

The proposal of this work is the development of an anti-spam filter, with an initial selective filter of e-mails. The pre-filter is responsible for transforming the information in the e-mails into consistent characteristic values, with an improvement in the classification process. Artificial Neural Networks, the multi-layer perceptron (MLP), were used as classifier, whose generalization and adaptation's capability has been important for the spam-filter.

The MLP architecture has been shown in situations where the filter did not have any incorrect classification of e-mails.

CONTEÚDO

LISTA DE TABELAS

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

Nos últimos anos, a Internet tem ficado cada vez mais presente na vida das pessoas. Diversos tipos de serviços e utilidades, além de uma gama de informações, são ofertadas. Dentre os serviços oferecidos, tem-se o correio eletrônico ou *e-mail,* que é uma forma de mensagem eletrônica enviada a um ou mais destinatários.

À medida que a Internet cresceu, o e-mail tornou-se uma ferramenta popular, tendo o seu uso foi generalizado. Assim qualquer pessoa com acesso à Internet pode ter uma conta de e-mail de fácil uso - existem diversos serviços gratuitos oferecidos por provedores[1]. Com a popularização do serviço, o correio eletrônico passou a ser utilizado como veículo de publicidade indesejada. Assim, enviar e-mails de propagandas indesejados tornou-se uma mania presente na correspondência eletrônica. Este tipo de e-mail é chamado de *spam e-mail* (ou *junk mail*, *bulk mail* e *unsolicited commercial e-mail*). É um nome geral para indicar todas estas mensagens indesejadas, postadas indiscriminadamente[2] para milhares de destinatários. Os tipos de *spams* são variados, dentre os quais podem ser citados: publicidade de sites pornográficos, propagandas de dinheiro fácil e financiamentos, além de medicações dos mais variados tipos. Em contrapartida, os e-mails normais ou legítimos que trafegam pela rede são

[1] Alguns serviços de e-mail disponíveis gratuitamente são: Yahoo (http://mail.yahoo.com.br), Hotmail (http://www.hotmail.com) e Gmail (http://mail.google.com/mail)

[2] Isto é, sem saber quem são os destinatários.

denominados de *ham e-mails*.

1.1 Descrição do problema

A quantidade de *spams* recebidos varia muito de pessoa para pessoa e depende de algumas informações, como o endereço de e-mail, a sua exposição na rede[3], o tempo de existência e a sua publicidade em rede. Além disso, o volume de *spams* varia de acordo com a época do ano [1]. No ano de 2002 foram criados servidores *proxy* e *relay* abertos[4], e o número de *spams* aumentou drasticamente. Estes servidores não oferecem proteção em relação à detecção de *spams*, e com isto houve um surto na quantidade de *spams* trafegados.

Em relação à quantidade de e-mails normais ou *hams*, a variabilidade é individual, ou seja, de acordo com os critérios de cada pessoa. A figura 1.1 mostra a variação do número de *spams* reportados em um serviço especializado, o SpamCop [2]. O SpamCop é usado por muitas pessoas para filtrar *spams* e também para enviar reclamações aos originadores de *spams*.

A grande maioria dos *spams* contêm informações falsas, e de acordo com Cournane e Hunt [3], a Comissão Federal de Comércio dos Estados Unidos prevê que dois entre três *spams* contenham informações enganosas de algum tipo, como endereço falso de retorno, assunto não condizente com a mensagem e propaganda sobre produtos questionáveis e ilicitudes na consecução de recursos financeiros, etc. Desta forma os *spams* podem ser bastante prejudiciais caso o usuário acredite nas informações que o

[3]Por exemplo, se o e-mail foi colocado em alguma página da Internet, o mesmo pode ter sido capturado por alguma ferramenta de busca usada por *spammers* - enviadores de *spams*.

[4]Isto é, que não exigem registro dos seus usuários.

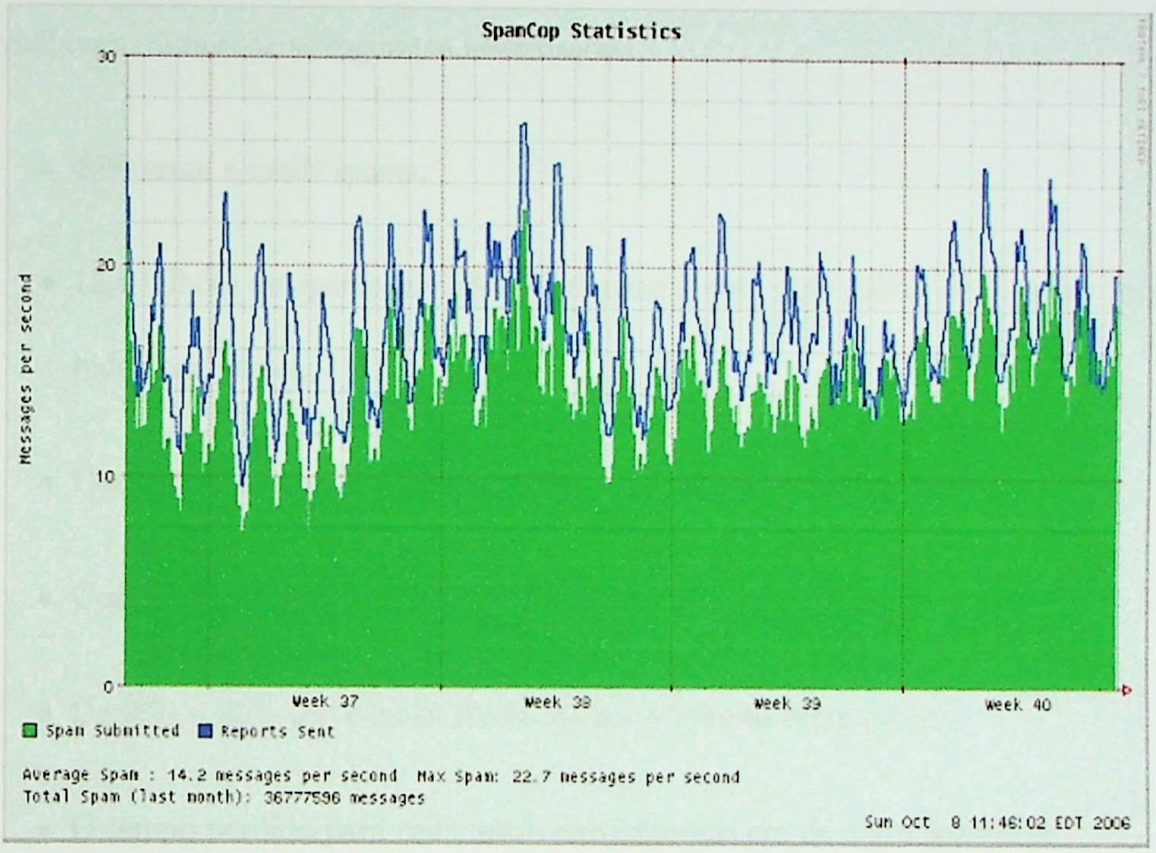

Figura 1.1: Estatísticas de *spams* recebidos e reportados para a SpamCop em Outubro de 2006 [2]

e-mail contenha.

Os *spams* podem causar prejuízos às pessoas e às empresas, na medida que se aumenta a quantidade de e-mails circulando nas redes de computadores [1] [8] [9].

Evett publicou um resumo estatístico [4] que mostra dados alarmantes sobre a quantidade de *spams* circulando na Internet[5]. A partir de um levantamento de fontes confiáveis, obteve-se as seguintes informações:

- 40% eram e-mails *spams*;
- 12,4 bilhões de *spams* enviados diariamente, com uma média de 6 *spams* recebidos por pessoa;
- Custo operacional de U$ 255 milhões para usuários não corporativos;
- Custo operacional de U$ 8,9 bilhões para usuários corporativos;
- De 20% a 25% dos e-mails recebidos nas empresas eram *spams*;
- O tempo perdido para cada *spam* nas empresas era de 4 a 5 segundos;
- Estimativa de crescimento de 63% no número de *spams* para o ano de 2007.

Em 1998 [5], a quantidade de *spams* era de aproximadamente 10% de todos os e-mails enviados e este número teve um aumento para 40%. Portanto, observa-se a seriedade do problema e a necessidade preventiva no tratamento do tráfego de e-mails. Os problemas abordados até o momento serão tratados com maiores detalhes nas próximas seções.

[5]Em Outubro de 2006

1.2 Importância do problema

Os *spams* causam uma série de problemas tanto para os usuários individuais quanto para as empresas. Pode-se citar dentre eles: enchimento das caixas de correio eletrônico (fazendo com que o usuário perca mensagens legítimas e relevantes, descartadas por falta de espaço), desperdício de recursos de rede, perda de tempo por parte dos usuários (conseqüente perda de produtividade em ambiente corporativo), divulgação de pornografia, entre outros. Cournane e Hunt fizeram uma compilação dos problemas causados pelos *spams* em [3], que serão listados a seguir.

1.2.1 Deslocamento de Custo

Os custos de envio de *spams* são baixos para o *spammer*. Com uma conexão por um modem de 56 Kbps, até milhares de e-mails podem ser enviados por hora. E quem paga o custo de recebimento é o usuário final, que já custeia a conexão com a Internet.

O custo causado pelo *spam* pode ser medido de diversas formas. Entre elas, a mais comum e simples é a medição do tráfego no envio e recebimento de *spams*, principalmente quando se cobra pelos dados trafegados. Este desperdício de recursos é altamente indesejável e precisa ser eliminado.

Para os provedores de Internet (ISPs), os *spams* são um grande problema, já que os servidores de e-mail perdem tempo com o envio e ficam sobrecarregados. Assim as mensagens legítimas (prioritárias) podem demorar a ser entregues, uma vez que o servidor está ocupado no envio de *spams*. Os provedores de Internet podem, neste

caso, repassar os custos causados pelos *spammers* ao usuário final na forma de uma conexão mais lenta ou preços mais altos. Isto é chamado de deslocamento de custo.

1.2.2 Fraude

Como os *spams* são indesejados, os *spammers* usam táticas fraudulentas para que suas mensagens sejam abertas pelo usuário final de qualquer forma. Uma das técnicas utilizadas é alterar o assunto da mensagem para algo que não seja relacionado com *spams*.

Encaminhamento fraudulento de mensagens também é usado para mascarar o originador do *spam*. Neste caso, o *spam* é enviado para um servidor de encaminhamento (*relay*) de terceiros ou anônimo que foi configurado para encaminhar a mensagem para destinatários escolhidos pelo *spammer*. Este é um típico caso de mascaramento, com a impossibilidade de se descobrir a origem do *spam*. E reclamações podem ser enviadas para o servidor de encaminhamento em vez da verdadeira origem. Assim, o *spammer* evita a responsabilidade pelos custos envolvidos e mantém-se anônimo.

1.2.3 Desperdício de Recursos

O congestionamento do tráfego na Internet sempre foi um problema sério. Mensagens enviadas para milhares de destinatários podem causar grandes congestionamentos nos roteadores e redes intermediárias, antes de serem entregues ao destino final. As filas de entradas de pacotes dos roteadores acabam ficando lotadas pela incapacidade de lidar com o enorme fluxo de dados, gerando grandes atrasos no funcionamento normal das redes envolvidas. Assim e-mails *ham* podem demorar a chegar

ao destino e, além disso, outras funcionalidades da Internet ficam prejudicadas. Isto ocorre porque os roteadores trabalham sobre a lei do "melhor esforço", isto é, fazem o possível para entregar os dados que recebem, inclusive os *spams*, tratados como dados normais.

Os efeitos são visíveis aos usuários finais na forma de atrasos na navegação e no recebimento de e-mails. Com o aumento da quantidade de *spams*, os problemas de congestionamento tendem a aumentar.

1.2.4 Perda de e-mails legítimos

Algumas contas de e-mail contêm espaço de armazenamento limitado. Antigamente este problema era mais sério, já que o espaço era de poucos *megabytes*. Desta forma, os *spams* podem encher o espaço que poderia ser usado para os *hams* e-mails. Quando o limite é atingido, os novos e-mails que chegam são descartados pelo servidor, podendo causar sérios prejuízos.

1.2.5 Listas Negras

Alguns provedores de Internet estão implementando as chamadas listas negras[6]. Estas listas contêm endereços de e-mails ou até mesmo domínios relacionados à prática de *spam*. Os provedores as utilizam na tentativa de reduzir o volume de tráfego de *spams* que circula em suas redes.

O problema com estas listas é que algumas delas[7] bloqueiam completamente um domínio, não apenas o endereço que envia o *spam*. Assim, usuários deste domínio têm

[6] Exemplos de listas negras podem ser encontradas em *http://spamlinks.net/filter-bl.htm*
[7] A lista *MAPS RBL group* é conhecida por fazê-lo [3].

seu e-mail bloqueado, mesmo não sendo *spammers*, o que é altamente prejudicial.

1.3 Soluções existentes e problemas

Muitos métodos já foram propostos para tratar o problema dos *spams*, mas eles não são completamente satisfatórios. Conforme Özgür, Güngör e Gürgen [5], os métodos podem ser divididos em duas categorias principais: estáticos e dinâmicos. As características, soluções propostas e problemas encontrados em cada uma das categorias serão tratados nesta seção.

1.3.1 Métodos estáticos

Os métodos estáticos têm como principal característica a necessidade de intervenção humana para realizar a tarefa de filtragem. Normalmente são baseados em listas que o usuário cria ao reportar um e-mail como *spam*. Assim, o seu servidor de e-mail (ou mesmo gerenciador de e-mail[8]) bloqueia ou apaga diretamente estas mensagens sem que o usuário saiba. O problema com este método é que os *spammers* conhecem as soluções e sempre mudam o endereço de envio. Desta forma, sempre chegam *spams* novos na caixa do usuário.

Outro método implementado é o bloqueio de mensagens que não fazem parte da lista de contatos do usuário. Assim, se uma mensagem chega de uma fonte desconhecida, ela é tratada como *spam*. O servidor envia uma mensagem ao originador notificando que a sua mensagem foi bloqueada. O usuário recebe uma notificação de mensagem bloqueada, podendo recebê-la caso queira. Este sistema é adotado pelos

[8]Gerenciador (ou cliente) de e-mail é o programa responsável por carrregar as mensagens do servidor e disponibilizá-las ao usuário.

servidores de e-mail do UOL e BOL, por exemplo. Esta solução também não é eficaz, já que mensagens importantes podem ser tratadas como *spam*, e os *spammers* também conseguem burlar esta proteção (simplesmente confirmando o e-mail enviado pelo servidor de destino, assim a mensagem é enviada ao usuário).

Finalmente, existem soluções que se baseiam na análise do assunto e de outros campos do cabeçalho do e-mail. O usuário pode criar filtros especiais no seu gerenciador de e-mails, ou seja, quando certas palavras ocorrerem, a mensagem é tratada como *spam* e, pode ser apagada ou movida para uma pasta especial para análise posterior.

1.3.2 Métodos dinâmicos

Os métodos mais complexos e eficazes são dinâmicos por natureza. A principal característica destes métodos é que levam em consideração o conteúdo dos e-mails e realizam tomada de decisão com base em tais conteúdos.

A maioria dos métodos dinâmicos usa técnicas de classificação de textos baseada em inteligência artificial, isto é, métodos onde a máquina "aprende" a classificar as mensagens nas duas categorias existentes. O classificador é construído automaticamente a partir de uma base de dados, onde e-mails coletados foram anteriormente classificados em *spam* e *ham*. Este processo é chamado de treinamento indutivo da máquina, e o classificador analisa documentos ainda não vistos baseado no aprendizado adquirido. Este treinamento também é conhecido como supervisionado.

Dentre os métodos dinâmicos, podem ser citados:

- Algoritmos *Naive Bayesian* - o seu uso é bastante comum em problemas de classificação de textos, assim como em filtros anti-spam. A idéia básica é usar a probabilidade na estimativa de uma dada categoria presente em um documento ou texto. *Naive* em inglês quer dizer ingênuo, sem conhecimento. No caso deste método, *naive* é usado porque o modelo assume que existe independência entre as palavras, isto é, a probabilidade de ocorrer uma palavra não depende da existência de outra(s). Assumindo isto, o cálculo computacional deste classificador é mais simples, o que torna o filtro eficiente e rápido quando comparado a outros métodos *bayesianos* não *naive*, onde a complexidade é exponencial. Esta solução já foi mostrada nos trabalhos de Özgür, Güngör e Gürgen [5], Androutsopoulos *et al.* [17] e Zhang *et al.* [20], e já é utilizada em alguns programas, como por exemplo, o gerenciador de e-mail Mozilla Thunderbird[9];

- *Support Vector Machine* (SVM) - SVMs são conjuntos de métodos de treinamento supervisionados utilizados para classificação e regressão. Pertencem a uma família de classificadores lineares generalizados. Tem como propriedade especial minimizar simultaneamente o erro na classificação empírica e maximizar a divisão geométrica entre as classes. Assim, também é conhecido como classificador com margem máxima (*maximum margin classifier*). O SVM já foi usado por Zhang *et al.* [20], Drucker, Wu e Vapnik [29] e Hidalgo [33];

- *K-Nearest Neighbour* (k-NN) - é um método usado para reconhecimento de padrões. A idéia básica é que, dado um documento qualquer na entrada, o sis-

[9]Mais detalhes do filtro podem ser obtidos em http://wiki.mozilla.org/Thunderbird2:JunkMail

tema dá uma pontuação para seus vizinhos mais próximos entre os documentos de treinamento, e usa as categorias dos k documentos com maior pontuação para adivinhar a categoria do documento de entrada. Os pesos das categorias às quais os vizinhos pertencem são então ajustados, usando pontuações próximas, e a soma total da pontuação dos k vizinhos mais próximos é usada para classificar documentos. Se mais de um vizinho compartilha a mesma categoria, então as pontuações são totalizadas para produzir um peso. Este método resulta na produção de uma lista de categorias propostas para o documento que está sendo testado na entrada. O k-NN é usado por Yang e Pedersen [24];

- *Boosting trees* - o propósito é encontrar uma regra de classificação precisa, fazendo combinações de várias hipóteses de base, que são pouco precisas. Um algoritmo particular é chamado de AdaBoost. É relativamente novo, foi proposto por Freund e Schapire em 1996. Esta versão é util na classificação em categorias simples. Ele mantém um conjunto de pesos representativos para os padrões de treinamento. Tais pesos são usados no algoritmo de aprendizagem, cujo objetivo é buscar uma hipótese de base com erro relativamente baixo em comparação aos pesos. Assim, o algoritmo usa estes pesos para forçar o método de aprendizagem a concentrar-se nos exemplos de mais difícil classificação. O AdaBoost é usado por Zhang *et al.* [20] e Carreras e Marquez [21];

- Aprendizado baseado em memória - é um método de aprendizado indutivo que armazena dados de treinamento em uma estrutura de memória na qual predições sobre novos dados são baseadas. O método assume que o ato de raciocinar é

baseado diretamente na reutilização de experiências armazenadas em vez do conhecimento adquirido (como regras ou árvores de decisão). Novos dados são classificados estimando a similaridade com os exemplos armazenados. Normalmente a classificação é feita usando uma variação do algoritmo básico do k-NN. Este método é usado por Sakkis e Androutsopoulos *et al.* [19] e Zhang *et al.* [20].

Além destas propostas apresentadas, existe também a utilização das redes neurais artificiais como classificadoras, que serão apresentadas no capítulo 2.

Todas estas soluções apresentaram resultados que foram considerados bons. Entretanto, a taxa de falsos positivos sempre foi alta, o que não é bom para o usuário final, conforme será explicado a seguir.

1.3.3 Problemas com os falsos positivos e falsos negativos

A classificação de e-mails enfrenta um grave problema, que é a classificação incorreta de mensagens legítimas. O falso positivo ocorre quando um *ham* é classificado erroneamente como um e-mail *spam*. O inverso, ou seja, um *spam* classificado como e-mail legítimo, também ocorre, e é chamado de falso negativo.

Quando um usuário recebe na sua caixa de entrada de e-mails um *spam* (falso negativo), o problema não é grave, ele pode simplesmente apagar a mensagem. Agora, se o mesmo usuário tem uma mensagem legítima classificada como *spam*, e esta é movida para uma pasta exclusiva, surge aí um problema ainda mais grave: o usuário pode simplesmente apagar as mensagens da pasta, sem ver o conteúdo, o que seria

bastante prejudicial.

O falso positivo faz com que o usuário sempre esteja verificando a pasta de *spams* regularmente, tentando encontrar mensagens erroneamente classificadas. O pior caso ocorre quando o servidor ou gerenciador de e-mail apaga automaticamente a mensagem.

Desta forma, os falsos positivos são altamente indesejáveis, e devem ser evitados a qualquer custo. Os falsos negativos, apesar de não tão graves, também devem ser diminuídos. Existem na literatura vários trabalhos sugerindo a utilização de modelos estatísticos [5] [17] [20] e de modelos baseados em redes neurais [5] [28] [29]. Porém, em ambos os casos, os filtros anti-spam compartilham do mesmo problema, ou seja, a alta quantidade alta de falsos positivos e falsos negativos [10]. Outrossim, os métodos apresentados na seção 1.3.2 apresentam problemas com a alta taxa de falsos positivos e falsos negativos.

1.4 Proposta de Solução

Este trabalho tem duas propostas principais. A primeira proposta é a utilização de redes neurais como classificador de e-mails *hams* e *spams*. As redes neurais têm sido extensivamente aplicadas no reconhecimento de padrões, assim como em diversos outros problemas de classificação em diferentes categorias [7].

Através do aprendizado indutivo, as redes neurais estabelecem funções que mapeiam um conjunto de entradas em um conjunto de saídas durante o treinamento [11]. A maior vantagem, no entanto, é o fato das redes neurais serem capazes de generalizar, ou seja, produzir resultados corretos para padrões nunca antes apresentados, através

Figura 1.2: Visão Geral da Proposta de Solução

do mapeamento suave entre as entradas e as saídas [11].

A segunda proposta, é a realização de um intensivo pré-processamento de dados, com o desenvolvimento de um pré-filtro avançado para o tratamento da base de dados e aplicação de diferentes métodos de seleção de características. O uso destes métodos simplifica a tarefa da rede neural em classificar os e-mails em *hams* e *spams*, garantindo uma maior precisão. Esta proposta é inédita e, é o diferencial deste trabalho comparado aos outros já realizados.

O sistema funcionará como representado na figura 1.2: o pré-filtro realiza o pré-processamento em cada e-mail e a rede neural artificial é responsável por classificar os dados do pré-filtro, indicando se o e-mail da entrada é spam ou não.

O presente trabalho é dividido da seguinte forma: o capítulo 2 explicará em detalhes a necessidade e o desenvolvimento do pré-filtro e a descrição da arquitetura da rede neural utilizada; o capítulo 3 apresentará a base de dados escolhida e a seleção das características relevantes. Também será explicada a obtenção do vetor característico, que é a entrada da rede neural; no capítulo 4 serão mostradas as medidas de desempenho, assim como os experimentos e os resultados obtidos. Finalmente, o capítulo 5 trará as conclusões sobre o trabalho proposto e suas contribuições, bem como sugestões para trabalhos futuros.

CAPÍTULO 2

PRÉ-FILTRO E REDES NEURAIS

O pré-filtro é um software desenvolvido para processar e-mails, tornando-os mais simples e uniformes, com a eliminação de partes desnecessárias, transformando o conteúdo abrangente e representativo. O pré-processamento deve ser realizado tanto em textos quanto em imagens e anexos dos e-mails.

Este capítulo é dedicado ao desenvolvimento do pré-filtro, porém trará uma introdução sobre a aplicação de Redes Neurais no trabalho.

2.1 Pré-filtro e a Classificação de E-mails

Para entender a necessidade da pré-filtragem dos dados é preciso apresentar a estrutura de um e-mail, como é feito a seguir.

2.1.1 Descrição de um e-mail

Um e-mail é composto de duas partes: o cabeçalho (*header*) e o corpo (*body*).

O cabeçalho é constituído de vários campos, apresentados no formato:

nome_do_campo: conteúdo do campo

Cada campo ocupa uma linha. Um cabeçalho típico apresenta o remetente, o destinatário, data, horário e assunto do e-mail.

O corpo do e-mail contém a informação a ser transmitida, que pode conter textos e anexos. A informação textual pode vir na forma de *texto plano*[1], exibido na forma em que está, não havendo qualquer processamento antes de sua exibição.

Outra forma comum é o texto HTML[2] (*text/html*), que contém diretivas ou *tags* específicas para apresentação do e-mail de maneira adequada.

O corpo do e-mail pode utilizar uma extensão especial chamada MIME [27], que estende a capacidade do e-mail, permitindo a inclusão de textos diferentes do padrão ASCII, de imagens e outros tipos de anexos. É um formato bem flexível que permite a inclusão de qualquer tipo de arquivo em uma mensagem de e-mail. Um cabeçalho MIME é incluído antes de cada conteúdo MIME, apresentando o seguinte formato:

tipo:sub-tipo

Exemplos de cabeçalhos MIME:

Exemplo 1 *Content-type: audio/wav*

Exemplo 2 *Content-type: text/plain*

Existe ainda o MIME chamado *multipart*. Um conteúdo *multipart* pode possuir outros tipos MIME. Assim, os e-mails podem apresentar a estrutura de uma árvore, que precisa ser processada recursivamente, conforme o exemplo da figura 2.1.

2.1.2 A informação nos e-mails

A informação presente em um e-mail está contida tanto nos cabeçalhos, quanto no corpo propriamente dito. Ela está nas palavras, na forma e disposição das mesmas,

[1] Texto incluído sem diretivas de formatação, como por exemplo um texto em **negrito** ou *itálico*.
[2] HyperText Markup Language

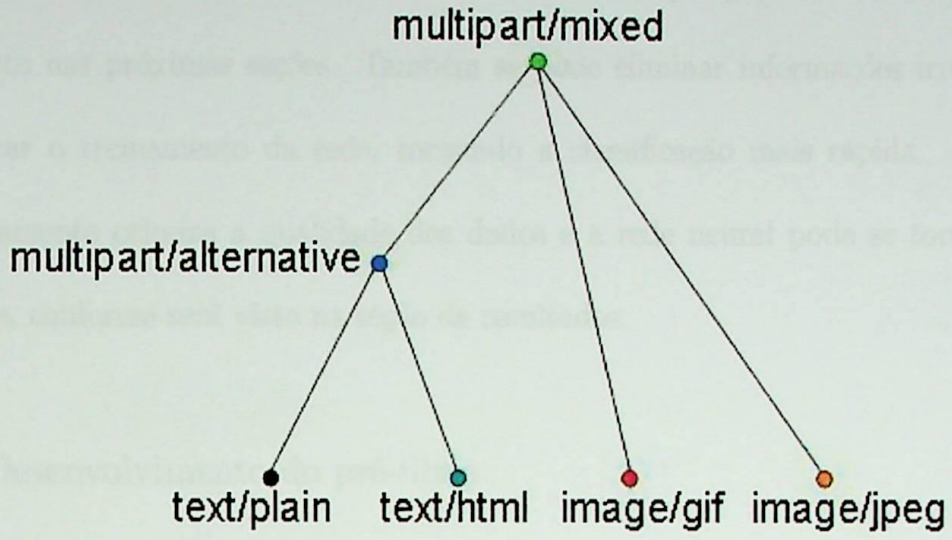

Figura 2.1: Exemplo de um e-mail com estrutura MIME *multipart*

no tipo de conteúdo e forma como são apresentados, e também nos anexos.

A informação precisa ser apresentada à rede neural para que a mesma possa ser treinada e, posteriormente, classificar corretamente novos e-mails. No entanto, o e-mail na sua forma original não tem como ser apresentado à rede neural, pois há muita informação redundante e inútil, que invalidaria o treinamento e a correta classificação.

Desta forma, observa-se a necessidade de processar tais dados (informações) para que possam ser apresentadas à rede neural. Este pré-processamento deve ser feito na etapa inicial do processo, o pré-filtro, que é um pacote computacional em Java, descrito na seção 2.2.

2.1.3 A importância do pré-filtro

É necessário observar que o pré-filtro tem um papel importante no desempenho do sistema de classificação [17] [27]. Com o pré-filtro, a rede neural encontra carac-

terísticas de *spam* que não seriam identificáveis sem o pré-processamento, conforme será visto nas próximas seções. Também se pode eliminar informações irrelevantes e facilitar o treinamento da rede, tornando a classificação mais rápida. Um pré-processamento otimiza a qualidade dos dados e a rede neural pode se tornar mais eficiente, conforme será visto na seção de resultados.

2.2 Desenvolvimento do pré-filtro

Foi desenvolvido um pacote computacional em Java, responsável por realizar todo o processamento dos e-mails para a obtenção dos dados de entrada da rede neural. A figura 2.2 representa as classes presentes e uma breve descrição das mesmas.

A primeira classe, `MessageInterface`, é reponsável por acessar os arquivos de e-mails gravados em dado local do computador, também permitindo que os e-mails alterados sejam gravados após as alterações do pré-filtro. A classe `ProcessaMensagens` é responsável por abrir os e-mails e acessar suas diferentes partes (cabeçalhos, corpo e anexos), incluíndo os conteúdos MIME citados na seção 2.1.1.

A classe `TokenSubst` foi criada para separar as palavras encontradas no corpo dos e-mails e enviá-las à classe `DetectaPadroes`. Ambas serão explicadas em mais detalhes ao longo desta seção.

2.2.1 "*Tokenização*" e processamento HTML

A classe de *tokenização*[3] `TokenSubst` faz parte do núcleo principal de processamento dos e-mails, junto com a classe `DetectaPadroes`. Ambas são responsáveis pela

[3] "tokenizar", do inglês, *tokenize*, quer dizer dividir algo em partes menores, chamados *tokens*.

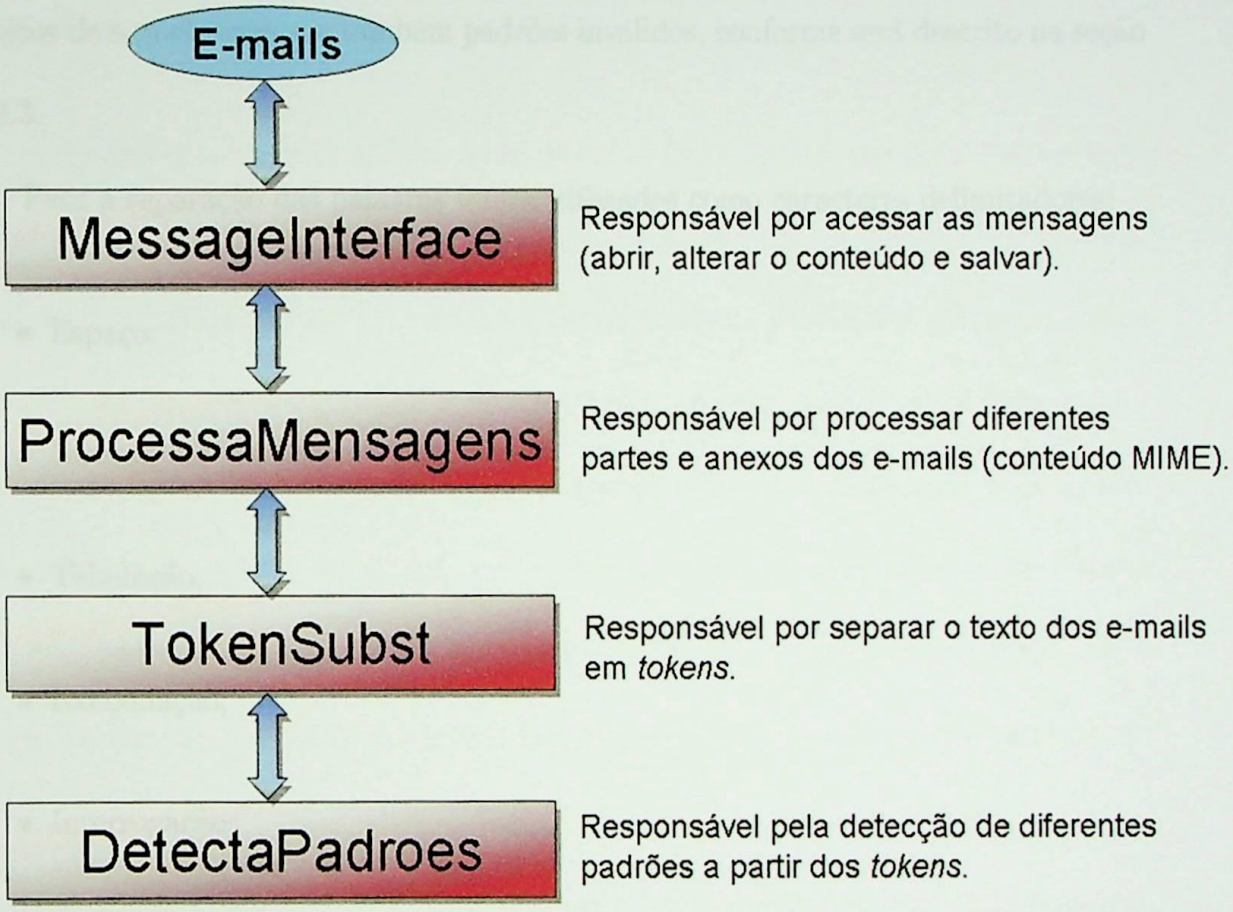

Figura 2.2: Classes do Sistema de Pré-Processamento

qualidade do processamento do pré-filtro.

A classe `TokenSubst` recebe como entrada um texto, que pode ser simples ou HTML. Quando o texto é simples, o processamento é feito apenas separando o texto recebido nos chamados *tokens*, ou seja, em simples palavras. As palavras são enviadas para a classe `DetectaPadroes`, que realiza processamentos para encontrar padrões típicos de e-mails *spams*, e também padrões inválidos, conforme será descrito na seção 2.2.3.

Para a separação das palavras foram utilizados como caracteres delimitadores:

- Espaço;
- Nova linha;
- Tabulação;
- Exclamação;
- Interrogação;
- Vírgula;
- Ponto-e-vírgula.

O processamento de texto HTML é mais complexo. O formato HTML permite adicionar vários parâmetros de formatação de texto, inclusão de tabelas, hyperlinks, imagens, informações multimídia, entre outros. Estas informações são adicionadas por meio de elementos chamados *tags*, no seguinte padrão:

<nome_do_tag parâmetro1=valor1 parâmetro2=valor2>Texto do tag</nome_to_tag>

Podem haver variações, como *tags* que não contêm parâmetros e textos, além de também existir *tags* inseridas dentro de outras *tags*. Assim, o algoritmo de processamento deve ser capaz de analisar todos estes casos e produzir uma saída sem que haja perda de informação útil.

Uma das principais propostas deste trabalho foi o aproveitamento de quase toda a informação das *tags* presentes nos *spams* HTML, ao contrário de outras propostas, onde toda a informação HTML sempre é descartada [17] [19] [20] [28], ou parcialmente descartada [23]. Propõe-se que *tags* apresentem padrões típicos tanto para *spams*, quanto de *hams* - sem descarte. Algumas *tags* HTML são consideradas importantes na detecção de *spams* (*html*, *src*, *script*, *href*), já que estas especialmente contêm imagens, *links* e *scripts*. Além disso, o uso de HTML é característica típica de e-mails *spams* [23], cuja a detecção é mais complexa. Cabe ressaltar que também há e-mails *hams* em HTML.

Assim, foi proposta a divisão das *tags* em três categorias distintas, cada uma com um tipo de processamento especial. A tabela 2.1 representa algumas das *tags* e suas respectivas categorias.

Na primeira categoria tudo é ignorado, isto é, o nome da *tag*, seus parâmetros e conteúdos. Neste caso, supõe-se que a informação presente seja irrelevante. Desta forma, o bloco HTML

<STYLE TYPE="text/css" MEDIA="screen, print, projection"></STYLE>

Tag	Categoria	Tag	Categoria
a	*3*	html	*2*
abbr	*2*	i	*2*
acronym	*2*	img	*3*
b	*2*	input	*3*
base	*3*	ins	*2*
body	*2*	label	*2*
br	*2*	li	*2*
button	*3*	map	*3*
caption	*2*	marquee	*1*
col	*2*	ol	*2*
comment tag	*1*	option	*2*
del	*2*	p	*2*
font	*3*	style	*1*
form	*3*	table	*2*
frame	*2*	textarea	*2*
h1-h6	*2*	title	*1*
head	*2*	tr	*2*
hr	*2*	var	*2*

Tabela 2.1: Processamento *Tags* HTML

é totalmente descartado durante o pré-processamento.

As *tags* da segunda categoria têm seus atributos removidos durante o pré-processamento. A *tag* em si é substituída por outra específica, composta dos caracteres "!_*in*_" mais a *tag*. O texto HTML "*<label for="e-mail">e-mail address</label>*" é substituído por "*!_in_label e-mail address*", por exemplo.

Na terceira e última categoria, a *tag* é processada integralmente. Neste caso o nome da *tag*, os parâmetros e o conteúdo são utilizados e adicionados à saída, como o exemplo, onde o bloco HTML *"<form action="results.php"> conteúdo </form>*" é transformado em "*!_in_form action conteudo*" durante o pré-processamento. É válido observar que o conteúdo dentro de uma *tag* da terceira categoria também é pré-processado, porque possui conteúdo HTML. A listagem de todas *tags* utilizadas

está presente no apêndice **B.**

O texto HTML, após a fase descrita acima, já está pronto para ser processado, como o texto plano que foi citado anteriormente. A classe `TokenSubst` realiza em cada *token* a transformação em caracteres minúsculos e a retirada dos caracteres acentuados, para melhorar a uniformização dos dados. Após isto, os *tokens* são passados para a classe `DetectaPadroes`, que ajuda na detecção de outros padrões típicos que podem ser utilizados em *spams*, conforme será explicado posteriormente na seção 2.2.3.

2.2.2 Métodos utilizados pelos *spammers*

Com o passar do tempo, os métodos utilizados pelos *spammers* se tornaram bastante sofisticados nas tentativas de enganar os filtros anti-spam. Existem alguns padrões que são conhecidos por serem utilizados em *spams* [1] [3].

Os primeiros *spams* que surgiram utilizavam assuntos como "GANHE DINHEIRO FÁCIL", ou os destinatários eram ocultos [1]. Este tipo de *spam* foi facilmente detectado e separado dos *hams* utilizando-se simples filtros de cabeçalho, com regras simples, do tipo: "se o assunto contém a palavra dinheiro, mova para a pasta de *spams*".

Outro método simples para detecção era procurar no corpo das mensagens por palavras típicas, como "viagra" ou "money". Assim, os *spammers* perceberam a necessidade de utilizar técnicas para enganar estes filtros e fazer com que suas mensagens fossem lidas pelos destinatários. Eles começaram a utilizar técnicas mais avançadas, como:

- *Esconder palavras chaves com caracteres inválidos.* Neste caso, as pessoas são capazes de ler e identificar o que o *spammer* quer, mas conseguem facilmente enganar os filtros mais simples. Exemplo: "v.ia.g.ra", "M O N E Y", "j-o*b f-or y.o.u".

- *Uso de tags HTML inválidas.* Com o uso de *tags* HTML inválidas, os *spammers* podem fazer com que textos considerados *hams* sejam processados pelo filtro, sendo exibidos no e-mail. Assim, com o uso de palavras consideradas normais, os *spammers* conseguem enganar os filtros que levam em consideração o "peso" das palavras, como os filtros bayesianos [3]. Um exemplo pode ser encontrado no apêndice A.

- *Uso de texto invisível.* O *spammer* neste caso esconde o texto considerado legítimo dentro da mensagem, para que enganem o filtro, da mesma forma que o caso anterior. O texto não é exibido pelo gerenciador de e-mail. Três exemplos de texto invisível são mostrados no apêndice A.

- *Uso de imagens com texto.* É uma técnica que envolve o envio de mensagens onde o *spam* está contido em uma imagem e não no texto do e-mail (que pode vir em branco ou com texto considerado legítimo para enganar filtros). Desta forma, o conteúdo do *spam* não pode ser filtrado pelos filtros anti-spam convencionais (pois necessitaria-se de muitos recursos computacionais, além do uso de inteligência artificial na detecção de textos, caso sejam camuflados[4] na men-

[4]Camuflado para um reconhecedor de caracteres comum (OCR), já que é legível ao ser humano.

sagem). As mensagens podem ter uma ou mais imagens, ou *links* HTML para as mesmas. Um exemplo deste caso pode ser encontrado no apêndice A.

- *Uso de comentários HTML.* Como muitos filtros anti-spam usam dicionários de palavras chaves em *spams*, os *spammers* podem esconder palavras usando comentários HTML ou outras *tags* que serão removidas pelo gerenciador de e-mail na exibição. Assim a palavra é exibida corretamente e o filtro é enganado. Para um melhor entendimento desta técnica, um exemplo é mostrado no apêndice A.

- *Uso de texto redundante.* Com o uso das extensões MIME, é possível enviar o texto HTML e plano ao mesmo tempo (o que foi feito no início por questões de compatibilidade com os gerenciadores de e-mail antigos), que não exibiam mensagens HTML. Nos gerenciadores atuais, o texto HTML é exibido por padrão, caso os dois tipos estejam presentes. Assim, o *spam* é colocado na parte HTML e um texto *ham* qualquer é colocado na outra parte. Como os filtros mais comuns não fazem distinção, as duas partes são processadas e o e-mail é considerado *ham*. Um exemplo ilustrando este caso é encontrado no apêndice A.

- *Uso de acentuação incorreta.* O uso de acentuação de forma inválida também pode enganar os filtros que não retiram a acentuação para fazer a verificação, como por exemplo, "Trábâlhe èm câsa, gañhe mûito dinheíro".

Nota-se que a tarefa de identificar corretamente um *spam* tornou-se complexa e um filtro eficiente deve saber lidar com todas estas formas de camuflagem utilizadas. Por esta razão, foi desenvolvida a classe `DetectaPadroes`, que será descrita a seguir.

2.2.3 Detecção de padrões conhecidos

A classe `DetectaPadroes` foi desenvolvida para permitir ao pré-filtro detectar padrões de texto conhecidos e utilizados por *spammers* e também para unificar outros padrões, permitindo que dados de certo tipo (como um *link*) tenham uma saída única após o processamento.

A classe detecta os seguintes padrões:

- *XXX = YYY.* É utilizado para unificar os valores de saída quando é processado um parâmetro de uma *tag* HTML. Exemplo: "<table color=blue>", a saída seria "!_table color" somente. Assim, se houvesse outro caso como "<table color=red>" a saída seria a mesma, permitindo maior uniformização;

- *E-mails.* Toda referência a um endereço de e-mail encontrado na mensagem é uniformizado na saída. Exemplo: a frase "Mande e-mail para compreaqui@loja.com.br" ficaria "mande e-mail para !_e-mail". Qualquer tipo de endereço e-mail é colocado na saída como "!_e-mail";

- *URLs ou hyperlinks.* Assim como no caso de e-mails, qualquer *hyperlink* tem como saída "!_link";

- *Caracteres inválidos no meio de palavras.* Os caracteres encontrados no meio das palavras são detectados e qualquer palavra com este padrão é substituída na saída por "!_HIDEWORDS". Exemplo: ".M.0-N-E_Y" é detectada como "_HIDEWORDS";

- *Palavras muito grandes.* Palavras muito grandes, normalmente sem sentido

algum e apenas usadas para enganar filtros, são detectadas e na saída é colocada uma palavra padrão indicando a ocorrência destes casos;

- *Números ou strings inválidas no sujeito.* Números ou *strings* inválidas colocadas após o sujeito são uma das técnicas usadas para burlar filtros. O programa detecta estas *strings* e coloca na saída uma *string* padrão "!_NUMERO_SUBJECT". Exemplo: "Get the best Life can offer you @e90jaakdfd". A saída seria "get the best life can offer you !_NUMERO_SUBJECT";

- *Dinheiro e porcentagem.* Da mesma forma que nos casos anteriores, qualquer ocorrência de quantidades monetária e porcentagem são transformadas na string "!_MONEY" e "!_PORCENTAGEM", respectivamente.

Assim, com a uniformização, os padrões são mais facilmente reconhecidos pela rede neural, já que o número de entradas da mesma é limitado.

2.2.4 Processamento dos e-mails

Para processar cada e-mail individualmente - através do assunto e do corpo da mensagem, que serão analisados por `TokenSubst` - foi desenvolvida a classe `ProcessaMensagens`, acessada através da interface gráfica mostrada na figura 2.3.

Primeiramente, a classe retira do cabeçalho o assunto da mensagem e envia para `TokenSubst`. Após isto, é verificado o campo "Content-Type", que diz qual é o tipo da mensagem. Se for somente texto simples ou HTML, ele é enviado para `TokenSubst` processar.

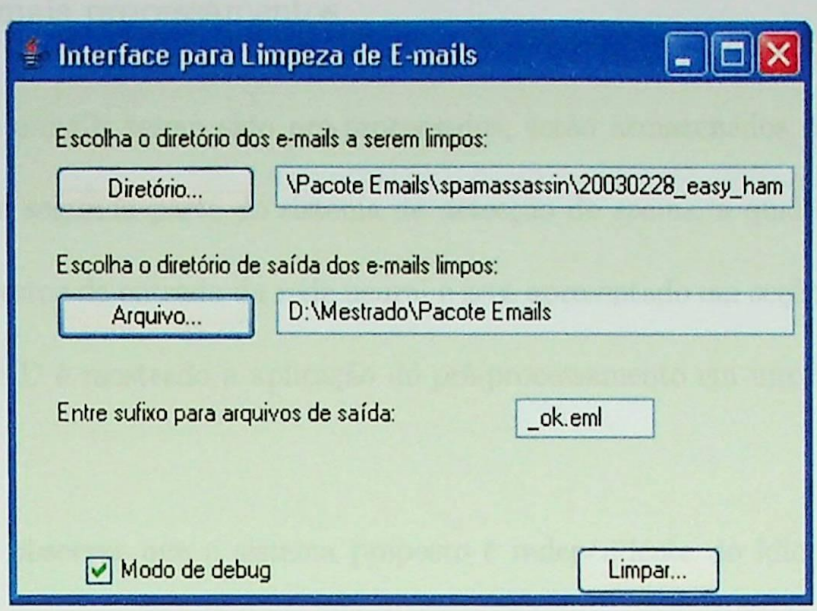

Figura 2.3: Software em Java - Pré-processamento dos e-mails

No caso de ser do tipo "multipart/alternative", explicado nas seções 2.1.1 e 2.2.2, somente o texto HTML é processado e o texto plano é ignorado.

Quando o conteúdo é uma mensagem tipo MIME, ele é tratado como uma árvore recursiva para obter as partes que interessam. Os textos válidos são pré-processados por `TokenSubst`.

Todos os anexos não textuais são descartados, e no seu lugar é adicionado ao corpo do e-mail a seguinte string: "!_ANEXO_TIPO?nome_anexo". Assim, os anexos também são considerados e uniformizados. E o seu tipo é colocado na string de saída, pois alguns tipos de anexos podem ser indicadores de *spam* [23]. Como exemplo, uma imagem em formato GIF, que é um arquivo binário, anexada ao e-mail é substituída pela string "!_ANEXO_TIPO?image/gif".

2.2.5 Demais processamentos

Após os e-mails terem sido pré-processados, serão armazenados em arquivos e utilizados na segunda parte do sistema de detecção de *spams*, a qual é responsável por gerar o vetor de entrada da rede neural e será apresentado em seções posteriores. No apêndice D é mostrado a aplicação do pré-processamento em um *ham* e em um *spam*.

Deve-se observar que o sistema proposto é independente do idioma, o processamento realizado é genérico e automático. Desta forma, ele pode ser aplicado a qualquer base de dados de e-mails para treinar a rede neural. Portanto, a princípio, o que determinará a linguagem que o sistema filtrará será essa base utilizada no treinamento. Maiores detalhes serão mostrados na seção de trabalhos futuros, 5.1.

2.3 Redes Neurais Artificiais

Há anos o homem vem tentando criar máquinas que se comportem como o ser humano. A ciência chamada de Inteligência Artificial tem conseguido avanços em diversas áreas. Entretanto, ainda não foi criada uma máquina que tenha um comportamento humano desejado, principalmente nas áreas de visão, fala e comportamento.

Diversos componentes interconectados logicamente compõem um computador. O cérebro humano, à primeira vista, é uma massa cinzenta. Entretanto, analisando mais a fundo, percebe-se que é composto por inúmeros componentes, os neurônios, que podem estar ligados a milhares de outros neurônios. Existe uma estrutura complexa e altamente paralelizada. Computadores conseguem realizar cálculos matemáticos em

frações de segundo. Um cérebro humano reage aproximadamente dez vezes por segundo. O cérebro é capaz de realizar tarefas intrinsecamente complexas para qualquer computador. O cérebro realiza várias tarefas simultaneamente, justamente devido à sua estrutura altamente paralelizada. Visão, fala, análise de problemas são situações onde diversos fatores precisam ser combinados para se chegar a um resultado correto, o que o cérebro realiza muito bem e computadores não [6].

O que os cientistas e pesquisadores tentam fazer na área de redes neurais artificiais é descobrir o princípio de funcionamento do cérebro na solução de problemas e aplicar em sistemas computacionais. Não se sabe como o cérebro representa as informações no nível mais alto, mas sim que ele se utiliza de diversas unidades lentas, porém, altamente interconectadas. Em resumo, para modelar o cérebro humano, precisa-se representar os neurônios e criar uma estrutura altamente paralela, ao contrário do que é feito na computação tradicional, com estruturas seriais. Nas próximas seções será explicado a evolução da ciência na área e o funcionamento de uma rede neural artificial.

2.3.1 Neurônio Biológico

O paradigma das Redes Neurais Artificiais surgiu em conseqüência da busca por conhecimento a respeito da mente humana. Nesse sentido, houve um grande interesse em pesquisar o papel do funcionamento de estruturas do Sistema Nervoso (SN), o qual motivou a construção de modelos matemático-computacionais que pudessem auxiliar na elucidação de aspectos neurobiológicos envolvidos em diversas atividades cognitivas. As Redes Neurais Artificiais encontram-se entre estes modelos e refletem

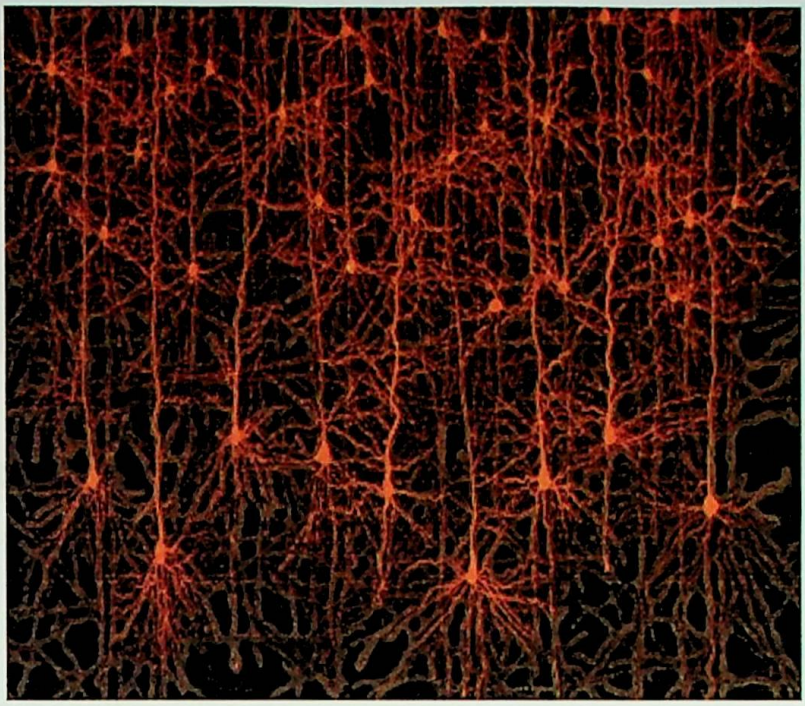

Figura 2.4: Imagem de parte do SN humano [32]

o comportamento de circuitos neuronais (grupos de neurônios funcionalmente conectados) [6].

Os neurônios são as células básicas do SN. Existem dois tipos principais de neurônios, os que realizam processamento e os que interconectam diferentes partes do cérebro entre si, ou conectam o cérebro a outras partes do corpo.

Individualmente, realizam operações relativamente simples, porém a riqueza das conexões entre estes tipos de células proporciona a enorme diversidade de tarefas realizadas pelo SN. O funcionamento detalhado de um neurônio ainda não é conhecido, mas sabe-se que um é composto das seguintes partes: corpo (soma), dentritos e axônio, apresentado na figura 2.5.

De forma geral, os axônios são responsáveis pela transmissão do impulso nervoso

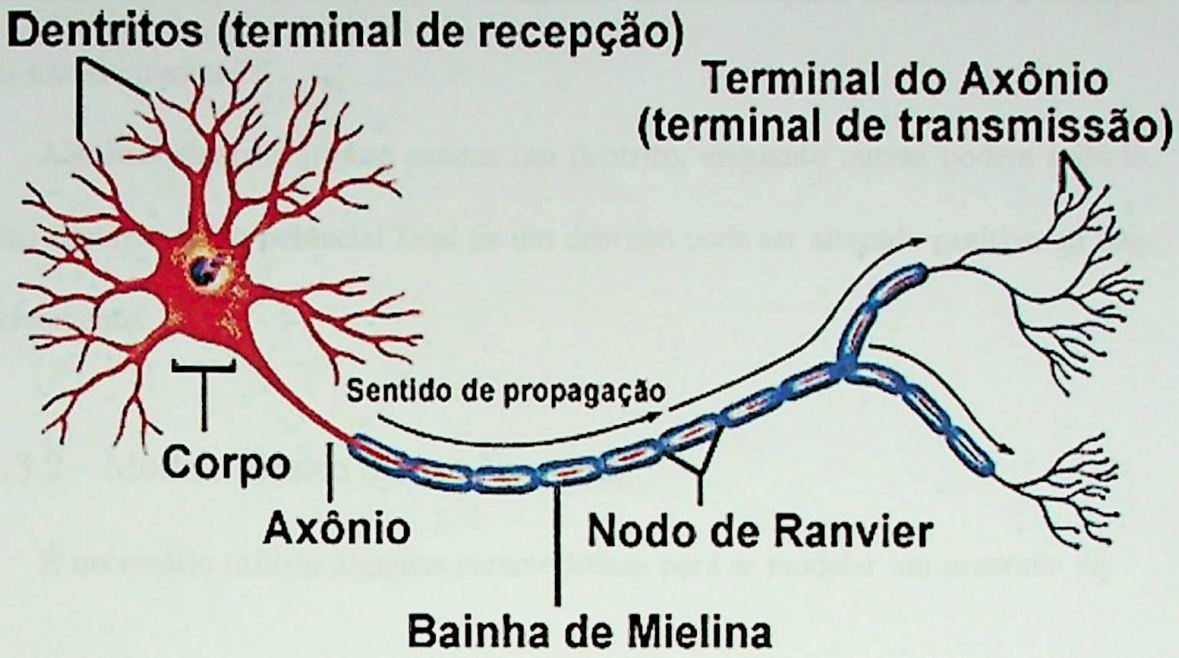

Figura 2.5: Estrutura básica de um neurônio

a outros neurônios, os dentritos relacionam-se com a captação de estímulos, enquanto que no corpo celular encontram-se o núcleo e organelas, responsáveis pelas demais atividades da célula.

Os neurônios são capazes de realizar operações complexas. Entretanto, sabe-se que, simplificadamente, este realiza um somatório dos estímulos recebidos pelos seus dentritos. O axônio é um dispositivo de disparo não-linear que produz em sua saída pulsos de tensão quando o potencial dentro do neurônio atinge um valor limite. O axônio termina em uma espécie de contato, chamado sinapse. Em realidade não existe contato físico, mas sim químico, estabelecido temporariamente quando o axônio é disparado. Os neurotransmissores são responsáveis pelo contato químico. Muitas sinapses podem atuar no mesmo neurônio. Assim, o conjunto de sinapses provoca

estímulos no corpo do neurônio. O somatório destes estímulos determina a ativação ou não do mesmo.

Algumas sinapses podem excitar um dentrito, enquanto outras podem inibi-lo. Isto significa que o potencial local de um dentrito pode ser alterado positiva ou negativamente.

2.3.2 Modelo Básico de um Neurônio

É necessário utilizar algumas características para se modelar um neurônio [6]:

- A saída está ligada ou desligada;
- A saída depende apenas das entradas;
- Cada entrada pode ter um ganho, ou eficiência, de ativação diferente;
- O neurônio realiza o somatório das entradas recebidas;
- O neurônio é disparado a partir de um certo número de entradas ativadas.

Na década de 1940, McCulloch e Pitts criaram um modelo simplificado para o neurônio biológico [12], chamado de *perceptron* McCulloch-Pitts. Este modelo tenta simular as realidades biológicas que ocorrem dentro de uma célula do sistema nervoso. A informação fornecida por outros neurônios correspondem aos estímulos de entrada (X), onde cada entrada possui ligações sinápticas (W). Os pesos sinápticos são fixos e a saída (Y) é obtida pela aplicação de uma função de limiar.

Este modelo, representado na figura 2.6, tem seu comportamento assim definido: o *perceptron* realiza o somatório das entradas ponderadas, compara com o limiar e

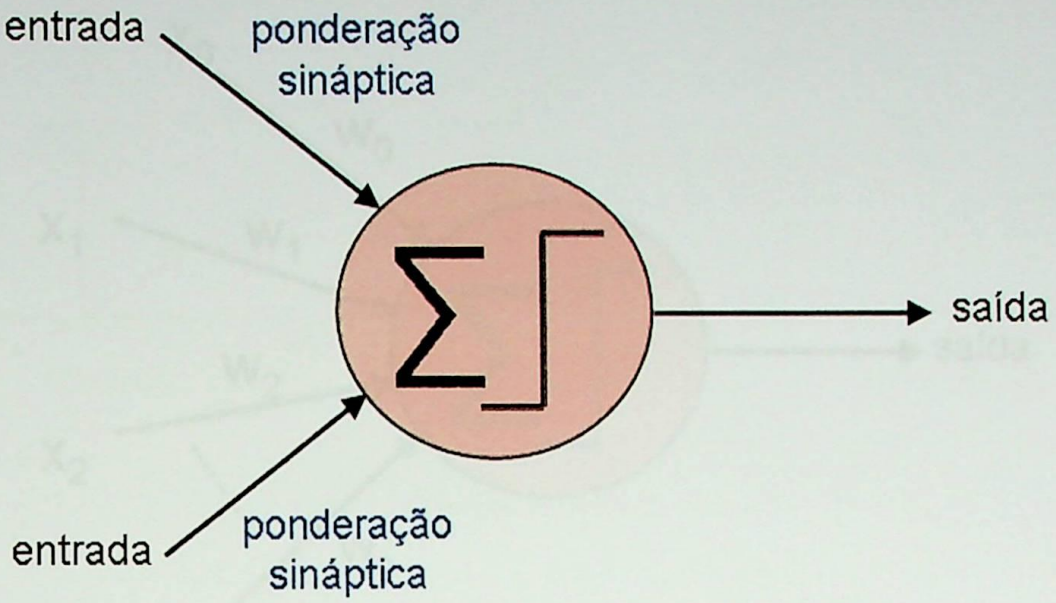

Figura 2.6: Neurônio Artificial

somente ativa a saída caso este nível seja ultrapassado.

Pode-se dizer que o sinal total na entrada (net) é o somatório do nível de cada entrada multiplicado pelo peso da mesma:

$$net = \sum_{i=1}^{n} W_i X_i \tag{2.1}$$

onde W_i é o peso ou ponderação na entrada i e X_i é o sinal na entrada i.

A saída y terá valor 1 se a soma ponderada for maior que o limiar θ, e 0, se for menor que o mesmo.

$$y = f_h \left[\sum_{i=1}^{n} W_i X_i - \theta \right] \tag{2.2}$$

onde f_h é a função de Heaviside [6]:

$$f_h(x) = \begin{cases} 1 \text{ ou "ligado"} & \text{se } x > 0 \\ 0 \text{ ou "desligado"} & \text{for } x \leq 0 \end{cases} \tag{2.3}$$

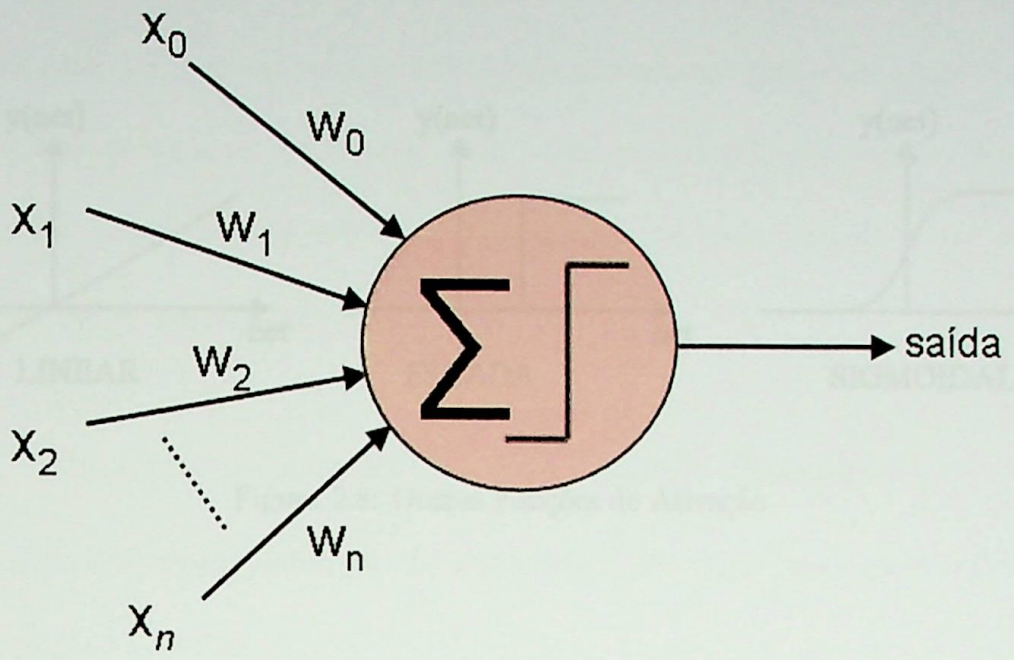

Figura 2.7: Modelagem do Neurônio Básico

Finalmente, chega-se ao modelo representado na figura 2.7.

A função de ativação utilizada no caso do *perceptron* McCulloch-Pitts, função de Heaviside, não é a única maneira de produzir o valor de saída do neurônio. Existe a função linear, que produz uma saída linear contínua; a função escada, que produz uma saída binária (não-linear discreta), e a função sigmoidal, com saída não-linear contínua. A figura 2.8 mostra diferentes tipos de funções de ativação que podem ser utilizadas.

No final da década de 1950, Rosenblatt propôs um novo modelo de *perceptron* [13] [14], composto do neurônio de McCulloch-Pitts, com função de limiar e aprendizado supervisionado. A arquitetura deste modelo consiste de camadas de entrada e saída, conforme a figura 2.9. O nome *perceptron* foi dado para este modelo de neurônios conectados de forma simples. Rosenblatt foi o pioneiro na simulação de redes neu-

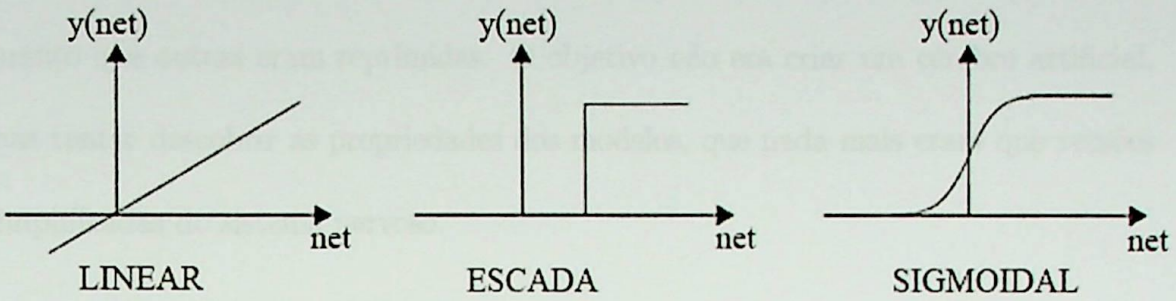

Figura 2.8: Outras Funções de Ativação

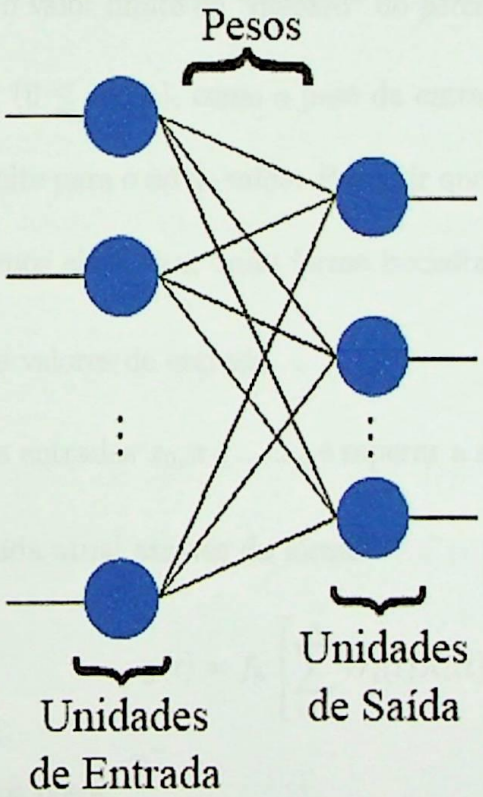

Figura 2.9: Perceptron de Rosenblatt

rais em computadores, assim como sua análise formal [6]. Os *perceptrons* eram redes simplificadas, onde certas propriedades do sistema nervoso real eram reforçadas, enquanto que outras eram reprimidas. O objetivo não era criar um cérebro artificial, mas tentar descobrir as propriedades dos modelos, que nada mais eram que versões simplificadas do sistema nervoso.

2.3.3 Algoritmo de Aprendizagem da Unidade *Perceptron*

O algoritmo de aprendizagem de uma unidade *perceptron* pode ser descrito conforme os passos abaixo:

1. Iniciar os pesos e o valor limite de "disparo" do *perceptron.*

   Definir $w_i(t)$, $(0 \leq i \leq n)$, como o peso da entrada i no tempo t e θ para ser o valor limite para o nó de saída. Permitir que os valores de $w_i(0)$ sejam valores pequenos aleatórios, desta forma inicializando todos os pesos.

2. Apresentar os valores de entrada.

   Apresentar as entradas $x_0, x_1, ..., x_n$ e esperar a saída $y(t)$.

3. Calcular a saída atual através da fórmula

$$y(t) = f_h \left[\sum_{i=1}^{n} W_i(t) X_i(t) \right] \tag{2.4}$$

4. Adaptação dos pesos.

   Se correto, $w_i(t+1) = w_i(t)$

   Se a saída é 0, e deveria ser 1, então $w_i(t+1) = w_i(t) + x_i(t)$

   Se a saída é 1, e deveria ser 0, então $w_i(t+1) = w_i(t) - x_i(t)$

Este é o algoritmo básico do *perceptron*. Os passos de 2 a 4 devem ser repetidos até que a saída apresente os resultados esperados, ou seja, classifique corretamente as entradas apresentadas. Algumas modificações foram sugeridas para melhorar o aprendizado da rede, como o caso de adicionar ponderações na adaptação dos pesos. Assim, o passo 4 ficaria descrito como:

4. Adaptação dos pesos (com ganho)

   Se correto, $w_i(t+1) = w_i(t)$

   Se a saída é 0, e deveria ser 1, então $w_i(t+1) = w_i(t) + \eta x_i(t)$

   Se a saída é 1, e deveria ser 0, então $w_i(t+1) = w_i(t) - \eta x_i(t)$

onde $0 \leq \eta \leq 1$, é o ganho positivo que controla a taxa de adaptação dos neurônios da rede.

Este fator tem o efeito de diminuir a mudança nos pesos, fazendo com que a rede dê passos menores rumo à solução, dando mais estabilidade ao sistema.

2.3.4 Perspectiva Vetorial e Limitações da Unidade *Perceptron*

Para uma análise vetorial da unidade *perceptron*, pode-se escrever a entrada X como um vetor $X = (x_0, x_1, ..., x_n)$ e o vetor de pesos como $W = (w_0, w_1, ..., w_n)$. Assim, chega-se à equação:

$$\sum_{i=1}^{n} w_i x_i = W \cdot X \tag{2.5}$$

O algoritmo de aprendizagem garante que os pesos são modificados a cada iteração para chegar ao resultado esperado, ou seja, reduzir o erro. Para duas entradas A e B, pode-se observar, no espaço bidimensional, o comportamento do vetor à medida

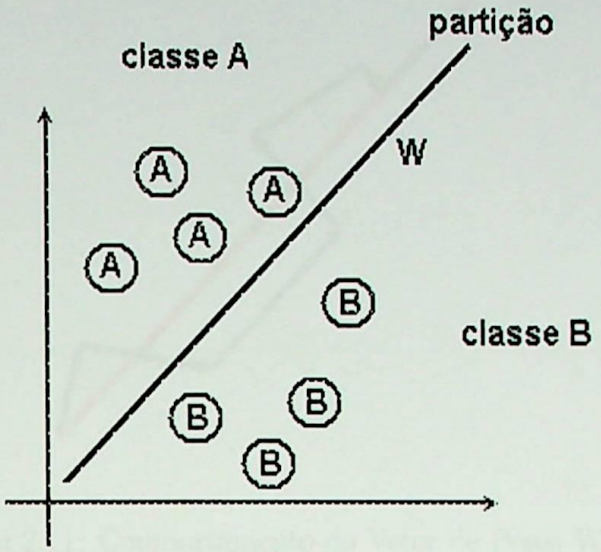

Figura 2.10: Representação Vetorial da Divisão em Classes

que a rede aprende. A linha de partição W é objetivo final, que o *perceptron* deve encontrar para realizar corretamente a classificação entre os padrões A e B, conforme a figura 2.10.

O *perceptron* encontra a linha divisória W ajustando os pesos da rede, como descrito na seção anterior. O valor de W inicial é aleatório, apontando para qualquer direção no espaço. Uma vez que um elemento de uma classe é apresentado, o procedimento de aprendizagem diz que o vetor será alterado para reduzir o erro. No final do processo, os pesos são ajustados para o valor ideal, que divide as classes existentes. O comportamento do vetor pode ser descrito na figura 2.11 - as diversas retas pretas correspondem ao processo de ajuste de W e valor ideal corresponde à reta vermelha.

O efeito do processo de aprendizagem, linha W, pode ser visto na figura 2.12. Nota-se na iteração 0 que a classificação é totalmente incorreta, justamente porque o vetor W foi iniciado com valores aleatórios. À medida que o *perceptron* é treinado,

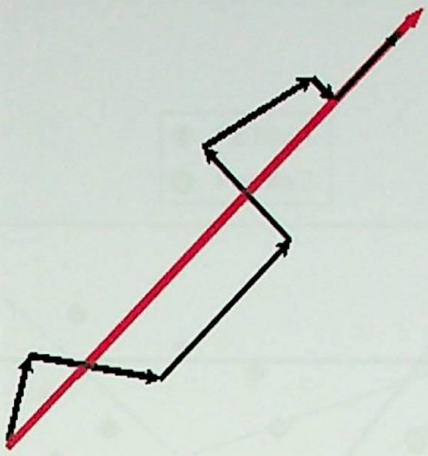

Figura 2.11: Comportamento do Vetor de Pesos W.

vê-se que a precisão de classificação é aumentada. No exemplo, a partir da vigésima iteração, a classificação torna-se correta para todos os padrões.

Pode-se observar que o *perceptron* tenta encontrar uma linha que divide corretamente os dois padrões apresentados. Porem, há situações onde não existe uma divisão linear entre as classes. Neste caso, o *perceptron* não consegue encontrar uma solução. O exemplo clássico deste problema é o caso da função lógica OU-EXCLUSIVO (XOR). A tabela verdade desta função é dada pela tabela 2.2.

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 2.2: Tabela Verdade OU-EXCLUSIVO

No espaço bidimensional da operação XOR, mostrada na figura 2.14, pode-se ver que não é possível desenhar uma linha que separe as saídas 0 e 1 corretamente. Este é um problema linearmente inseparável e impossível de ser solucionado pelo modelo

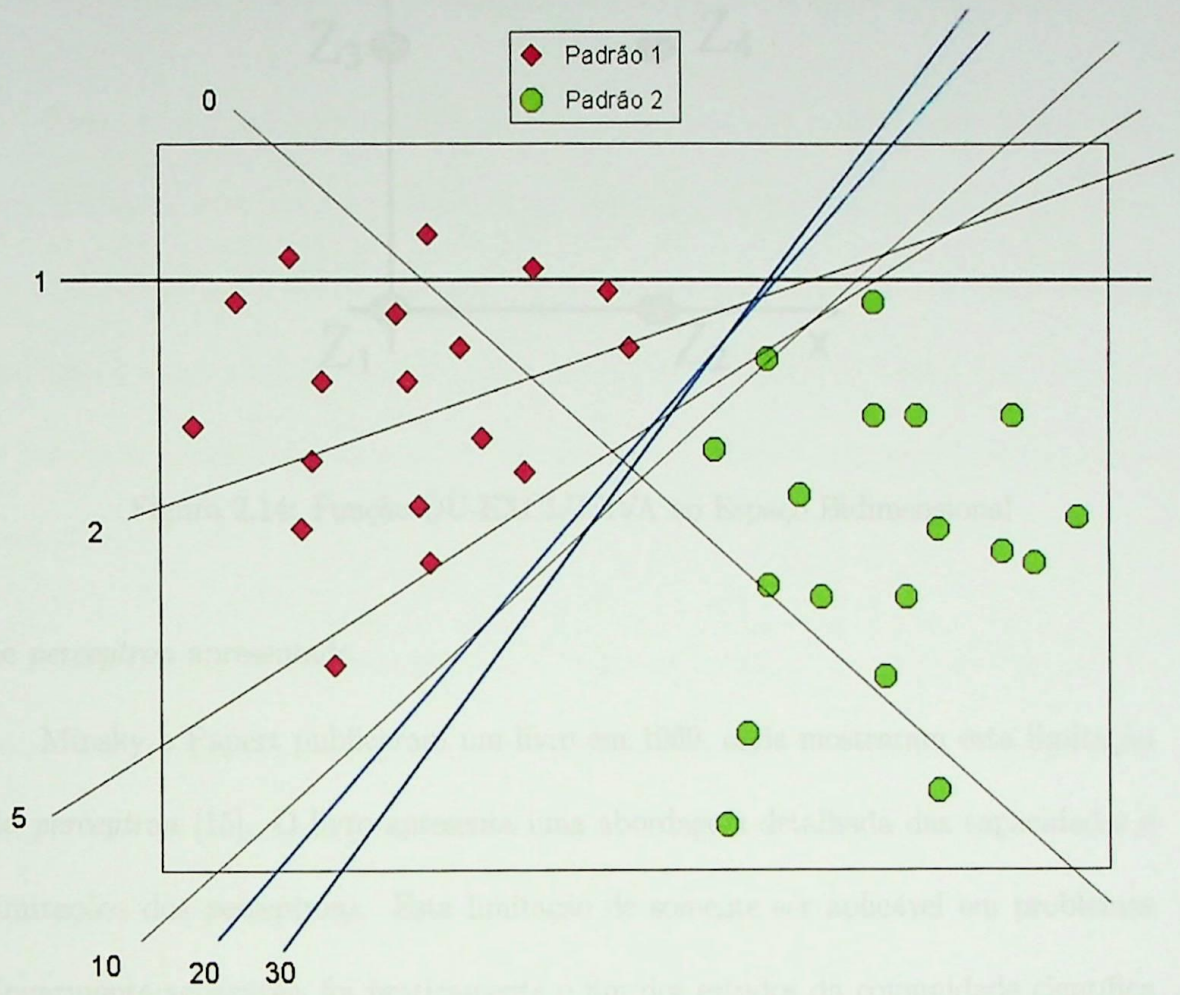

Figura 2.12: Evolução na Classificação de Padrões

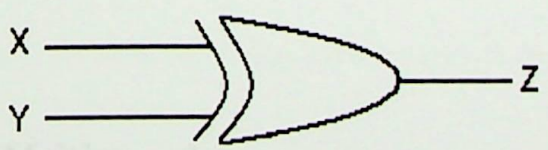

Figura 2.13: Símbolo da Função Lógica OU-EXCLUSIVO

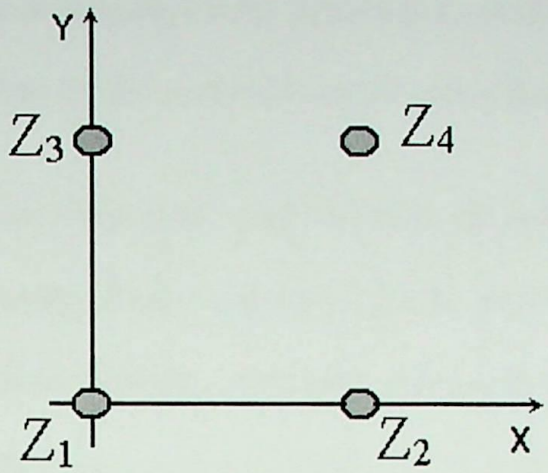

Figura 2.14: Função OU-EXCLUSIVA no Espaço Bidimensional

de *perceptron* apresentado.

Minsky e Papert publicaram um livro em 1969, onde mostraram esta limitação do *perceptron* [15]. O livro apresenta uma abordagem detalhada das capacidades e limitações dos *perceptrons*. Esta limitação de somente ser aplicável em problemas linearmente separáveis foi praticamente o fim dos estudos da comunidade científica na área de redes neurais [6]. Somente em 1986, quase vinte anos depois, houve um avanço significativo na área, com Rumelhart e McClelland [16], e o modelo do *Perceptron* Multicamadas (ou *Perceptron* de Múltiplas Camadas), que será descrito na próxima seção.

2.3.5 *Perceptron* Multicamadas

Uma solução para a limitação dos *perceptrons* em problemas linearmente separáveis foi então desenvolvida. Primeiramente, ao modelo inicial, foram adicionadas camadas de *perceptrons*, com ligações entre eles e as camadas. As unidades *perceptron*

continuavam a realizar os mesmos cálculos internamente, ou seja, a soma ponderada de suas entradas, a ativar ou não a saída, de acordo com a função de ativação.

Para as funções linear e sigmoidal, o valor da saída será praticamente 1 se a soma ponderada exceder bastante o limite e, da mesma forma, praticamente zero se a soma for bem menor que o limite. Contudo, caso a soma ponderada e o limite sejam valores próximos, a saída oscilará entre os dois extremos, 0 e 1. Deste modo, fica resolvido o problema da camada anterior em não passar informação mais significativa para a camada seguinte.

Na prática, a função escolhida é não-linear sigmoidal, já que camadas de *perceptrons* com funções lineares podem ser reduzidas a uma única camada apenas. Isto ocorre porque cada camada realizaria cálculos puramente lineares que podem ser reduzidos em apenas uma operação, de acordo com Beale e Jackson [6].

Desta forma, foi criado o modelo de múltiplas camadas, figura 2.15. Este modelo é basicamente composto de três camadas: a entrada, a intermediária (ou oculta) e a saída. Os *perceptrons* das camadas intermediárias e de saída são como os *perceptrons* descritos, mas com função de ativação sigmoidal. A camada de entrada tem o propósito de distribuir os valores recebidos para a camada seguinte, não realizando nenhuma operação.

Com a mudança da função de ativação, é necessário alterar a regra de aprendizado, descrita na próxima seção. Esta rede agora é capaz de aprender a reconhecer padrões mais complexos [6].

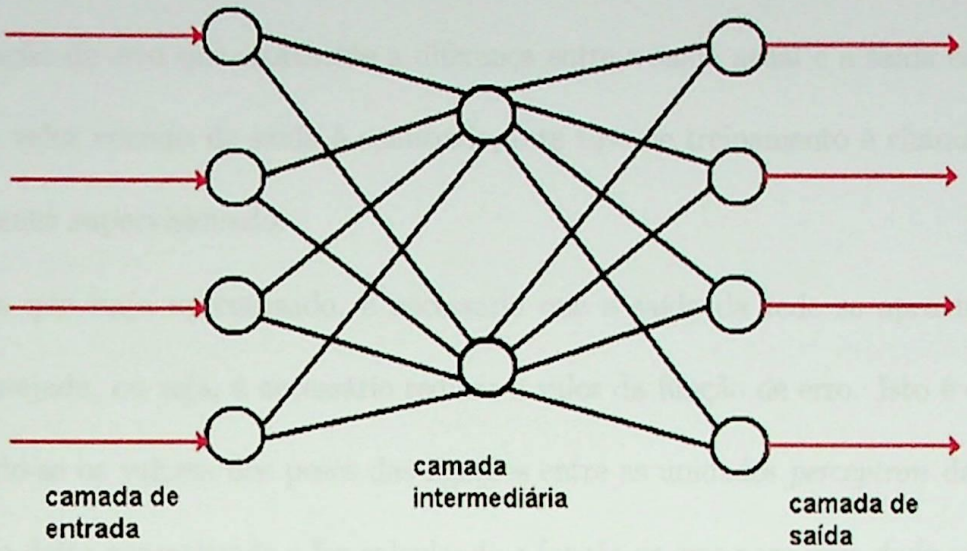

Figura 2.15: Novo Modelo Multicamadas

2.3.6 A Nova Regra de Aprendizado

A nova regra de aprendizado é conhecida como "*generalised delta rule*" ou "*back-propagation rule*" e foi sugerida em 1986 por Rumelhart, McClelland e Williams. A operação da rede é similar ao modelo *perceptron*, na qual é mostrado um padrão à rede que então apresenta a saída. A comparação da saída obtida com a desejada propicia o ajuste dos pesos para que, na iteração seguinte, a saída seja ainda mais precisa. A regra de aprendizado utilizada para o modelo de *perceptron* não é mais adequada para o modelo de múltiplas camadas. Todavia, como visto anteriormente, o uso da função sigmoidal possibilita que informação suficiente seja transportada para as camadas posteriores, permitindo que as unidades sejam ajustadas com o fim de reduzir o erro na próxima iteração da rede [6].

A nova regra de aprendizado é mais complexa. Quando uma entrada é apresentada

a uma rede não treinada, uma saída aleatória é produzida. É necessário então definir uma função de erro que represente a diferença entre a saída atual e a saída correta. Como o valor correto de saída é conhecido, este tipo de treinamento é chamado de treinamento supervisionado.

Para que haja aprendizado, é necessário que a saída da rede se aproxime da saída desejada, ou seja, é necessário reduzir o valor da função de erro. Isto é obtido ajustando-se os valores dos pesos das ligações entre as unidades *perceptron* da rede. A função delta generalizada o faz calculando a função de erro para uma dada entrada e retro-propagando o erro de uma camada para a anterior. Portanto, cada unidade da rede tem seus pesos ajustados para a minimização do erro.

Para as unidades na saída, ajustar o peso é relativamente simples, contudo para as unidades intermediárias, o ajuste não é tão óbvio. Pode-se pensar que as unidades intermediárias ligadas à saída com maior erro precisam de pesos ajustados em valores maiores, enquanto que as intermediárias ligadas às saídas com erro menor quase não precisem de ajustes. De fato, matematicamente é mostrado que os pesos para um nó em particular deve ser ajustado na proporção direta ao erro das unidades às quais é ligado; assim retro-propagar o erro pela rede permite ajustar os pesos entre as camadas corretamente, permitindo que o erro seja reduzido e que a rede aprenda [6].

2.3.7 Algoritmo MLP

O algoritmo para treinamento de uma rede neural artificial multicamadas, ou *Multilayer Perceptron* (MLP), com retro-propagação de erro é mostrado nesta seção. Ele exige que as unidades tenham funções limitadoras não lineares continuamente

diferenciáveis. Assim, é assumido o uso da função sigmoidal,

$$f(t) = \frac{1}{1 + e^{-kt}} \tag{2.6}$$

que apresenta derivada simples. O algoritmo MLP é descrito abaixo:

Passo 1 - inicializar os pesos e limites das sinapses e neurônios, respectivamente

Definir $w_i(t), (0 \leq i \leq n)$, como o peso da entrada i no tempo t e θ como o limite para a unidade de saída. Permitir que w_0 seja $-\theta$ e x_0 seja sempre 1. Permitir também que os valores $w_i(0)$ e $x_i(0)$ sejam valores aleatórios pequenos, e assim inicializar os pesos e limites.

Passo 2 - Apresentar as entradas e a saída desejada

Apresentar a entrada $X_p = \{x_0, x_1, ..., x_n\}$ correspondente à saída $T_p = \{t_0, t_1, ..., t_m\}$, onde n é o número de unidades de entrada e m é o número de unidades de saída. Para a comparação de padrões, X_p e T_p representam a entrada e saída comparada. Para classificação, T_p é ajustado para zero, exceto um elemento ajustado em 1, que corresponde à classe a qual X_p pertence. O erro para P é dado por:

$$E_p = \frac{1}{2} \sum \left(t_{pj} - o_{pj}\right)^2 . \tag{2.7}$$

Passo 3 - Calcular a saída atual

Para cada camada, $y_{pj} = f\left[\sum_{i=0}^{n} w_i(t) x_i(t)\right]$ e passa a ser a entrada para a próxima camada. A camada final apresenta os valores o_{pj}.

Passo 4 - Adaptação dos pesos

Inicia-se na camada de saída e retorna para a camada imediatamente anterior:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} o_{pj} \tag{2.8}$$

onde w_{ij} representa os pesos da unidade i para a unidade j, η é o ganho e δ_{pj} é um termo de erro para a entrada p na unidade j:

$$o_{pj} = f_i \left(\sum_i w_{ij} o_{pi} \right) \tag{2.9}$$

Para as unidades de saída:

$$\delta_{pj} = k o_{pj} (1 - o_{pj})(t_{pj} - o_{pj}) \tag{2.10}$$

Para as unidades intermediárias:

$$\delta_{pj} = k o_{pj} (1 - o_{pj}) \sum_k \delta_{pk} w_{jk} \tag{2.11}$$

onde a soma é para as k unidades na camada posterior à camada da unidade j.

Repetir os passos de 2 a 4 até que o erro seja minimizado ao nível desejado.

2.3.8 O problema do OU-EXCLUSIVO

Conforme mostrado anteriormente, o Modelo do *Perceptron* é incapaz de resolver o problema da operação XOR. Todavia, o *Perceptron* de Multicamadas é capaz de solucionar problemas não-linearmente separáveis, como o caso da operação citada. Pode-se simular tal operação utilizando o *Perceptron* Multicamadas com apenas um *perceptron* na camada intermediária, e utilizar a tabela verdade 2.2 como conjunto de treinamento. Por sua característica não-linear, a função de ativação será a função sigmóide.

A tabela 2.3 mostra o conjunto de treinamento utilizado - no lugar do 0 é utilizado 0,1; e o 1 é substituído por 0,9.

Exemplo	Entrada 1	Entrada 2	Saída Desejada
0	0,1	0,1	0,1
1	0,1	0,9	0,9
2	0,9	0,1	0,9
3	0,9	0,9	0,1

Tabela 2.3: Conjunto de Treinamento XOR

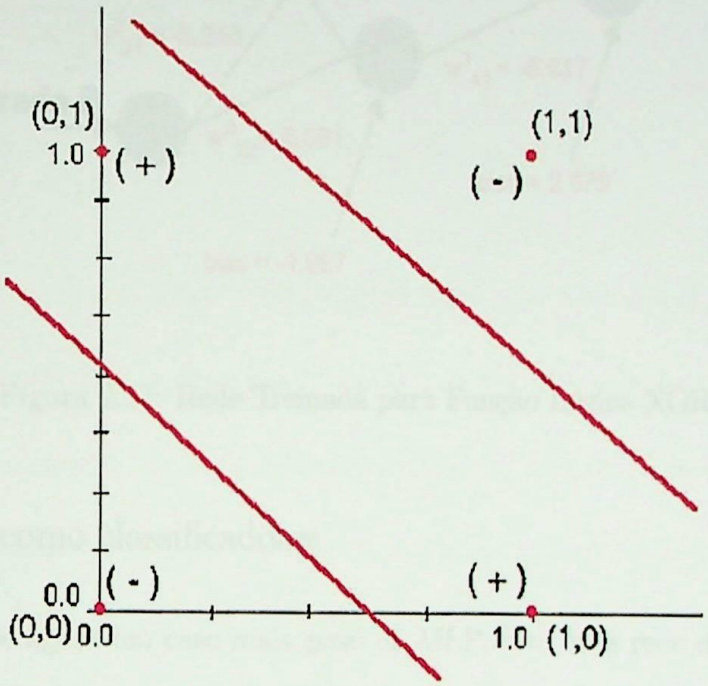

Figura 2.16: Função Lógica XOR - Impossibilidade de Separação Linear

Observando o gráfico espaço de decisão (figura 2.16), nota-se que não é possível separar as áreas de classificação utilizando-se somente retas. A rede dada pela figura 2.17 representa o estado final após o treinamento.

Com esta arquitetura simples, a rede responde corretamente a problemas não-linearmente separáveis. Porém, para casos mais complexos, surge a necessidade de uma modelagem específica, onde podem ser utilizados mais *perceptrons* e mais camadas, a fim de que se alcance o resultado esperado.

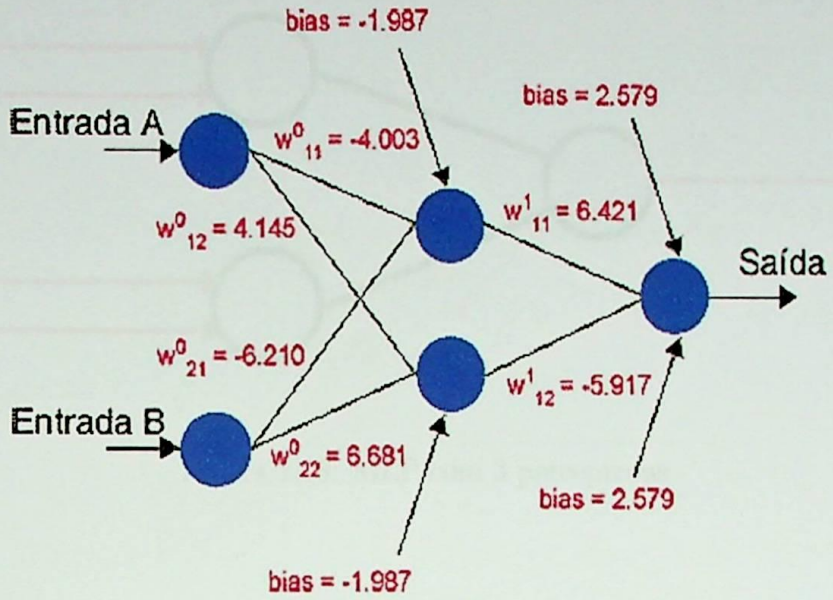

Figura 2.17: Rede Treinada para Função Lógica XOR

2.3.9 MLP como classificadores

Considera-se agora um caso mais geral de MLP, com uma rede de três unidades *perceptrons* (figura 2.18). Se a unidade intermediária tem uma função limitadora, que só é ativada quando ambas as unidades na primeira camada estão ligadas, tem-se a operação "E" lógica.

Como as unidades da primeira camada definem uma linha no espaço de estado, a segunda produzirá uma classificação baseada na combinação destas linhas. Se uma unidade é arranjada para responder com 1, caso a entrada esteja abaixo da linha de decisão, e a outra responder com 1, se a mesma entrada está além desta linha de decisão, então a segunda camada produz uma solução como ilustra a figura 2.19.

Mais de duas unidades podem ser usadas na primeira camada, produzindo uma combinação com mais de duas linhas de decisão. As regiões produzidas são con-

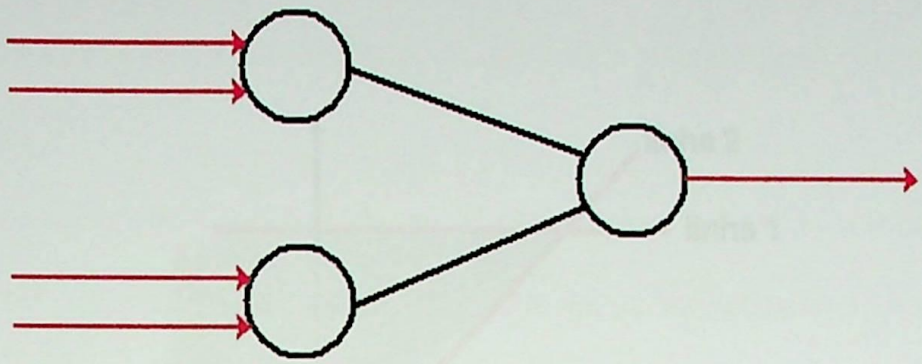

Figura 2.18: MLP com 3 perceptrons

hecidas como regiões convexas. Estas regiões podem se interceptar, superpor-se ou permanecer completamente separadas entre si, resultando nas mais variadas formas.

Assim, três camadas de unidades *perceptrons* (ou quatro, considerando-se a camada de entrada, que não realiza nenhuma operação) podem arbitrariamente formar regiões convexas complexas, capazes de separar quaisquer tipos de classes [6]. A complexidade das formas está limitada ao número de *perceptrons* na rede, isto significa que não seria necessário mais do que três camadas na rede para qualquer nível de complexidade do problema abordado (Teorema de Kolmogorov [6]).

2.3.10 Generalização

O último assunto a ser tratado na introdução sobre redes neurais é a generalização. É uma das características mais importantes e essenciais das redes neurais: a capacidade da mesma em classificar corretamente padrões que nunca foram apresentados. A rede de múltiplas camadas generaliza detectando características significativas do padrão de entrada. Estas características são representadas nos pesos das unidades

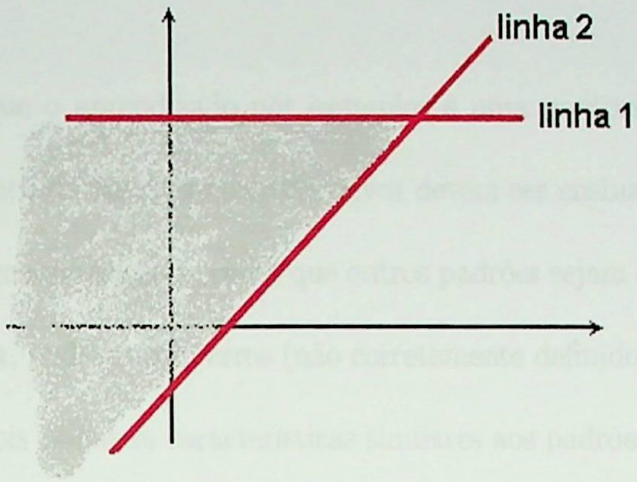

Figura 2.19: Região de Decisão: combinação de 2 perceptrons

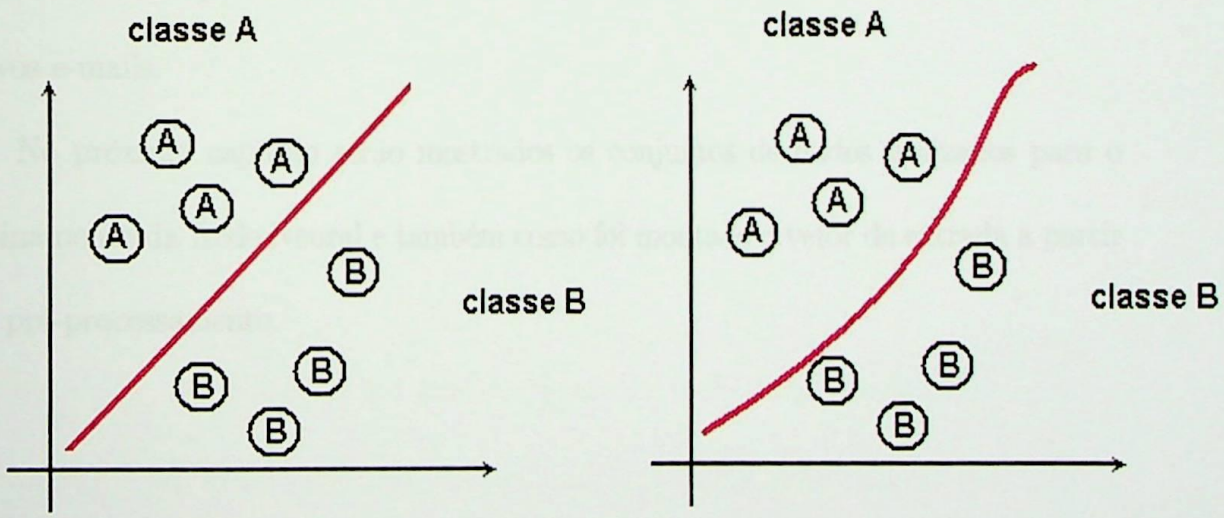

Figura 2.20: Perceptrons como Classificadores

internas da rede. Desta forma, um padrão desconhecido é classificado como outros padrões já apresentados à rede neural, já que ambos compartilham características similares.

Isto significa que o aprendizado por exemplos é uma proposição viável, já que somente um conjunto de padrões representativos devem ser ensinados à rede neural. A propriedade de generalização permite que outros padrões sejam classificados corretamente [6], ou seja, padrões com erros (não corretamente definidos) também podem ser classificados, pois possuem características similares aos padrões puros.

É a característica da generalização que permite que redes neurais tenham mais sucesso no mundo real que outros tipos de sistemas inteligentes. Esta capacidade permitiu escolher Redes Neurais para o problema da classificação de e-mails. Não existe uma função matemática clara que separe os e-mails corretamente. As redes neurais podem detectar padrões diferentes através do treinamento e classificar corretamente novos e-mails.

No próximo capítulo serão mostrados os conjuntos de dados utilizados para o treinamento da Rede Neural e também como foi montado o vetor de entrada a partir do pré-processamento.

CAPÍTULO 3

CONJUNTOS DE DADOS E DISTRIBUIÇÕES UTILIZADAS

No processo de classificação, os e-mails são apresentados como padrões de entrada à rede neural. Para uma rede neural com um número fixo de padrões de entrada n, um e-mail é transformado em um vetor de entrada v_n de dimensão n

$$v_n = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ . \\ . \\ . \\ a_n \end{pmatrix} \tag{3.1}$$

onde $a_i (i = 1...n)$ é o valor que descreve a ocorrência de uma palavra i.

Neste capítulo serão descritos os conjuntos de dados utilizados na rede neural. Primeiramente, será mostrado como foram obtidos os e-mails a partir de bases de dados e as comparações entre os diferentes tipos de bases usadas pela comunidade científica. Após a escolha da base de dados, será apresentado os métodos de obtenção do vetor de entrada para a rede neural.

3.1 Conjuntos de Dados

Para treinamento de uma rede neural, em qualquer aplicação, é necessário um bom conjunto de dados para a fase de treinamento. Este conjunto deve ser representativo para o problema proposto. A divisão entre padrões positivos (*spams*) e padrões negativos (*hams*) também é considerado, como será visto adiante.

3.1.1 Bases de Dados Públicas

Existem bases de dados de e-mails disponíveis na Internet para serem utilizados em testes. Algumas delas serão descritas, assim como suas vantagens e desvantagens.

3.1.1.1 Ling-Spam Corpus

A base de dados Ling-Spam [18] foi produzida pelo grupo de Ion Androutsopoulos e utilizada em seus artigos, assim como em trabalhos de outros autores. O Ling-spam é uma mistura de *spams* com mensagens legítimas enviadas para uma lista de lingüística. Esta base de dados consiste de 2893 mensagens, entre as quais:

- 2412 mensagens são da área de lingüística. Somente a mensagem em si foi aproveitada. O autor retirou o texto adicionado pelo servidor da lista de lingüística;
- 481 são *spams*, recebidos pelo autor. Os anexos, *tags* HTML e *spams* duplicados recebidos não foram incluídos.

Nota-se que esta base contém aproximadamente 17% de *spams*, o que o autor considerou como a porcentagem de *spams* que ele recebia. Quanto aos tópicos das

mensagens, eles não são tão específicos como poderia se esperar, já que contêm além de assuntos da área, anúncios de empregos, disponibilidades de novos softwares e outras discussões [19].

3.1.1.2 PU1 e PU123A Corpus

Estes conjuntos consistem de apenas 1099 mensagens [20], entre as quais 481 são marcadas como *spam* e 618 como legítimas, ou seja, aproximadamente 44% de *spams*. Esta base de dados foi produzida pelo mesmo grupo do Ling-Spam. Os cabeçalhos, anexos e *tags* HTML foram removidos, deixando apenas o sujeito e o corpo da mensagem. Por questões de privacidade, cada palavra foi mapeada em um número único no PU1, que pode vir em quatro versões diferentes: com ou sem *stemming* e com ou sem s*top word-removal*. Já a versão PU123 apresenta as palavras originais, mas sem *stemming* e s*top-word removal* [18].

Stemming é uma técnica de transformar palavras nas suas formas morfológicas originais. Neste caso, por exemplo, conjugações diferentes de um verbo poderiam ser transformadas no verbo original, por exemplo, "testado" e "testei" virariam "testar". Para que o sistema funcione na prática é necessário um bom algoritmo lingüístico com uma boa base de dados para cada linguagem que se deseja aplicar.

Stop-word removal é outra técnica usada para remover palavras que ocorrem freqüentemente. Para esta remoção, é necessário existir uma lista previamente criada. Nota-se, neste caso, que é necessário escolher bem quais palavras entrarão nesta lista. Além disso, deve-se considerar também uma base de dados para cada linguagem que se deseja aplicar.

A vantagem de se utilizar estas duas técnicas é a de que haveria uma redução na dimensão do espaço característico, com conseqüente melhoria no processo de aprendizagem dos filtros, que poderiam fazer previsões mais acuradas. Neste caso, haveria uma redução no problema dos dados serem muitos esparsos, de acordo com Zhang *et al.* [20]. Entretanto, Androutsopoulos *et al.* [17] investigou o uso destas técnicas no classificador Bayesiano usando o conjunto PU1. Os resultados mostraram que, na maioria das vezes, utilizando as técnicas *stemming e stop-word removal,* não houve melhora estatística significativa comparado à bases de dados onde não foram usadas essas técnicas. Desta forma, devido à complexidade necessária para aplicá-las, tais técnicas não foram utilizadas neste trabalho.

3.1.1.3 SpamAssassin Corpus

O conjunto de mensagens SpamAssassin Corpus (SA) [22] é uma seleção criada especialmente para auxiliar nos testes de sistemas de filtros anti-spam. Apresenta as seguintes características:

- Todos cabeçalhos foram introduzidos completamente. Apenas algumas mensagens tiveram o *hostname* alterado[1] para outro válido, apenas por questões de privacidades. Na maioria dos casos, os cabeçalhos utilizados foram os originais;

- Todas as mensagens recebidas pelo autor foram enviadas com o conhecimento de que seriam disponibilizadas para o domínio público. Foram também adicionadas mensagens do tipo *newsletter* de servidores públicos ao conjunto;

[1] A segunda parte que compõe um endereço de e-mail. Ex.: nome@servidor.com.br -> o hostname é *servidor.com.br.*

- Existem 1897 *spams* e 4150 *hams* no conjunto, ou seja, aproximadamente 31% de *spams*.

3.1.2 Escolha da Base de Dados

O conjunto Ling-Spam foi compilado de diferentes fontes: as mensagens são originadas de listas de discussão livres de *spams*, e os tópicos destas listas são específicos. Os *spams* foram recolhidos de uma caixa de e-mails pessoal. Desta forma, observa-se que a distribuição de e-mails não é a mesma de um usuário normal, o que pode fazer com que o conjunto Ling-Spam seja facilmente filtrado [20]. A mesma informação vale para os conjuntos PU1 e PU123A. Já o conjunto SA é uma compilação de mensagens pessoais de diversos assuntos e fontes, junto com os *spams* recebidos nos mesmos endereços de e-mails. Assim sua distribuição é mais próxima ao de um usuário comum.

O SpamAssassin possui os e-mails no formato original, sem retirada de *tags* HTML, anexos ou cabeçalhos. Isto é essencial para a proposta deste trabalho. Os conjuntos Ling-Spam e PU1 e PU123A tiveram os anexos e *tags* HTML removidos, o que desqualifica o conjunto.

Para este trabalho, foi escolhida a base SpamAssassin, que também apresenta o maior conjunto de dados, importante para um treinamento representativo, teste e avaliação da rede neural. Além disso, como os e-mails estão no formato original, o sistema desenvolvido para o SpamAssassin pode ser facilmente utilizado em filtros de e-mails em várias aplicações práticas.

3.2 Processamento da Base de Dados

Agora que foi escolhida a base de dados SpamAssasin, será mostrado o processamento antes da geração do vetor característico para a rede neural.

O software desenvolvido em Java realiza a última fase antes da entrada da rede neural, após a etapa realizada na seção 2.2.4. Nesta fase é feita a preparação dos vetores de entrada necessários para treinamento da rede neural. E o processamento é repetido na geração dos vetores para os conjuntos de teste e de validação.

3.2.1 Pré-processamento para Geração do Vetor Característico

Após o processo descrito na seção 2.2.4, todos os e-mails limpos são gravados em diretórios. Cada e-mail continua gravado em arquivos separados. A primeira classe usada nesta etapa é chamada de `GeraEstruturaDados`. Ela é responsável por gerenciar outras classes na geração de estruturas de dados que permitirão os cálculos estatísticos nas etapas seguintes do processamento. Um diretório e uma váriável (indicando se o conjunto é de *spams* ou *hams*) formam os parâmetros de entrada da classe. Após esta entrada, um comando é enviado para o processamento do diretório. Diferentes diretórios podem ser passados como objeto da classe. No final, é gerado um arquivo que serve de entrada para a classe `Estatisticas`, que será explicada mais adiante.

Cada arquivo de e-mail é representado por um objeto da classe `NoEmails`. As seguintes informações são retiradas de cada arquivo e guardadas neste objeto:

- Classificação do e-mail: *spam* ou *ham*;

- Lista *hash*[2] com todas as palavras do e-mail e o número de ocorrências de cada palavra.

Por necessidade de cálculos estatísticos, a classe `ListaEmails` armazena todos os e-mails na classe `NoEmails`. Esta classe também armazena os dados em uma lista *hash*, usando o par (nome do e-mail, `NoEmails`).

Uma outra classe armazena todas as palavras encontradas em todos os e-mails, chamada `NoPalavras`. Para armazenar a lista de todas as palavras de todos os e-mails, foi criada a classe `ListaPalavras`, que utiliza uma lista *hash* no formato (palavra, `NoPalavras`).

Todas as classes descritas acima servem de suporte para gerar as estruturas de dados utilizadas nos cálculos estatísticos para se obter o Vetor Característico. Estas estruturas de dados são gravadas em um arquivo por `GeraEstruturaDados`. Na seção 3.3 serão explicados os métodos estatísticos usados no projeto.

3.2.2 Separação da Base de Dados

Para treinamento da rede neural, foi feita a separação da base de dados. O corpo de mensagens SpamAssassin foi então dividido nos três subconjuntos:

- Conjunto de Treinamento (60% do total de mensagens);
- Conjunto de Validação (20% do total de mensagens);
- Conjunto de Teste (20% do total de mensagens).

[2]Uma tabela *hash* é basicamente um dicionário onde, dado um objeto chave, pode-se ter o valor do objeto. Tabelas *hash* usam uma função hash para encontrar a chave do código que representa uma entrada na tabela. Chaves iguais devem sempre retornar o mesmo código *hash*, permitindo, dessa maneira, que você encontre a entrada associada na tabela *hash*.

Cada conjunto recebeu a mesma proporção de *spams* e de *hams*. Este particionamento é normalmente recomendado. O conjunto de treinamento, como o nome já diz, é usado para o treinamento da rede neural (propagação do erro, adaptação dos pesos, etc.). O conjunto de validação é usado durante o treinamento para testar a capacidade da rede em generalizar, já que, se a rede for submetida ao mesmo conjunto exaustivamente, pode tornar-se altamente especializada nesta fase (e não classificar corretamente novos tipos de dados). O conjunto de teste é usado após o treinamento da rede para medir o desempenho final da rede [23].

3.3 Seleção de Características

A seleção de características é uma parte importante na classificação de textos e, portanto, nos filtros anti-spam em particular, de acordo com Goetschi [23]. Uma característica é uma palavra, e o número total de característica corresponde ao conjunto de todas as palavras que ocorrem no conjunto de treinamento. Este conjunto normalmente é muito grande (uma dimensão por cada palavra diferente). Desta forma, a maior dificuldade na classificação de textos é a alta dimensão do espaço característico.

A maioria das técnicas, incluindo as redes neurais, não é capaz de lidar com um conjunto tão grande, porque o processamento exigido, em termos computacionais, é muito alto e, em conseqüência, a classificação é ineficiente [24].

A seleção de características é o método utilizado para reduzir a dimensão do espaço característico, selecionando as características (ou palavras, no caso deste projeto) que são mais informativas na classificação a ser feita. É altamente desejável que se faça a seleção sem sacrificar a precisão da separação em duas classes distintas.

Também é desejável que a seleção seja feita de forma automática, isto é, sem que nenhuma definição ou construção de característica seja necessária. Existem diversos algoritmos para realizar essa tarefa, entre os quais:

- *Document Frequency Thresholding* (DF);
- χ^2 *statistic* (qui-quadrado);
- *Mutual Information* (MI);
- *Information Gain* (IG);
- *Term Strength* (TS).

Neste trabalho foram escolhidos os três primeiros algoritmos para a seleção de características. A escolha foi feita com base nos trabalhos de Yang e Pedersen [24] e Androutsopoulos *et al.* [17]. Os três métodos mostraram bons resultados. Nota-se, em especial, no trabalho de Yang e Pedersen, que o DF, apesar de ser o mais simples dos métodos (consumindo menores recursos computacionais), apresentou resultados muito bons. O método χ^2 também apresentou resultados bons, apesar de ser computacionalmente mais exigente. O MI foi escolhido por estar presente em alguns trabalhos, como o de Androutsopoulos [17]. Nas próximas seções serão apresentados os métodos empregados.

3.3.1 *Document Frequency Thresholding* (DF)

Document Frequency (DF) mede o grau de ocorrência de um termo w em um conjunto C. Foi calculado o DF para cada palavra encontrada no conjunto de dados conforme a equação 3.2.

$$DF(w) = \frac{N\left[w \in \{spam, ham\}\right]}{T} \tag{3.2}$$

onde $N\,[w \in \{spam, ham\}]$ é o número de ocorrências da palavra w nas classes $\{spam, ham\}$ e T, o número total de palavras nas classes mostradas.

Para gerar o vetor característico, foram escolhidas as palavras com valores de DF mais altos, considerando-se que palavras com baixa freqüência de ocorrência não são significativas para a predição de categorias ou não afetam o desempenho geral da rede [24]. Em ambos os casos, a remoção dos termos raros reduz a dimensão do espaço característico. Melhorias na precisão da categorização também são possíveis se alguns dos termos raros removidos são ruídos (ou seja, informação inválida).

DF é a técnica mais simples para a redução do vocabulário. Ela facilmente reduz conjuntos extremamente grandes, em uma complexidade computacional aproximadamente linear ao número de mensagens no conjunto de dados. Entretanto, DF só é normalmente considerado nas abordagens iniciais, não é o principal critério de seleção de características. Outrossim, esta técnica não é usada na redução drástica de termos de um conjunto, porque se assume, em Teoria da Informação, que termos com baixo DF são considerados como de alta informação, já que ocorrem poucas vezes e podem ser significativos para a divisão em diferentes classes [24].

3.3.2 χ^2 *statistic* (qui-quadrado)

A distribuição χ^2 mede o grau de independência entre um elemento e e um conjunto S [26]. Se w é uma característica, e C um conjunto com duas classes - *spam* e *ham* - a distribuição chi-quadrado de uma característica é dada por:

$$\chi^2(w) = P(spam) \cdot \chi^2(w, spam) + P(ham) \cdot \chi^2(w, ham) \tag{3.3}$$

onde $P(spam)$ e $P(ham)$ são as probabilidades de ocorrência de e-mails *spam* e *ham*, respectivamente. A distribuição chi-quadrado para uma caraterística w e uma classe c é dada por:

$$\chi^2(w,c) = \frac{N \times (kn - ml)^2}{(k+m) \times (l+n) \times (k+l) \times (m+n)} \tag{3.4}$$

onde k é o número de e-mails, dentro da classe c, que contém a característica w; l é o número de e-mails, dentro da classe $\bar{c}$, que contém a característica w; m é o número de e-mails, dentro da classe c, que não contém a característica w; n é o número de e-mails, dentro da classe $\bar{c}$, que não contém a característica w; e N é o número total de e-mails dentro da classe c.

As características com os valores mais altos de χ^2 foram escolhidas. Cada uma dessas características é uma entrada na MLP.

3.3.3 *Mutual Information* (MI)

Mutual Information, ou Informação Mútua, é um método básico usado em Teoria da Informação [26]. Se w é uma característica, a Informação Mútua da característica w é dada por:

$$MI(w) = \sum_{f=\{w,\overline{w}\}} \sum_{c=\{spam,ham\}} P(f,c) \log_2 \frac{P(f,c)}{P(f) \cdot P(c)} \tag{3.5}$$

onde $P(f,c) = P(f \cap c) = P(c) \cdot P(f \mid c)$ é a probabilidade de f e c ocorrerem simultaneamente. As características com os valores mais altos de MI são então selecionadas. Cada característica selecionada é uma entrada na MLP.

3.4 Vetor Característico

Nesta seção será explicado como foram aplicados os três tipos de estatísticas na base de dados, bem como a criação do vetor característico.

3.4.1 Classe Estatísticas

Com os dados gerados pela classe `GeraEstruturaDados`, é possível realizar os cálculos DF, χ^2 e MI. A implementação das equações destes métodos foi feita na classe Estatísticas.

Para cada palavra, é calculado um valor em cada método estatístico. Assim, é gerado um arquivo com a palavra e os resultados obtidos. A partir deste arquivo, é possível gerar o vetor característico.

3.4.2 Vetor Característico

A partir do arquivo com os dados estatísticos para todas as palavras, foi possível gerar o vetor característico para os três métodos. Por exemplo, para obtenção do vetor característico DF, uma ordenação DF foi usada. Assim, as palavras com mais alto DF são separadas e guardadas em um segundo arquivo. O mesmo é feito para χ^2 e MI. Isso foi necessário, já que a quantidade total de palavras do conjunto de dados foi de 56652.

A escolha das palavras para cada um dos métodos obedeceu o seguinte:

- DF - as palavras com maior número de ocorrências foram escolhidas, ou seja, maior DF, em ordem decrescente. Neste trabalho o DF usado foi a soma do

valor de DF de *spams* com o DF de *hams* para cada palavra;

- χ^2- as palavras com o valor χ^2 mais alto foram escolhidas, isto é, a soma do valor de χ^2 *spams* com o χ^2 *hams*, em ordem decrescente;

- MI - as palavras com o valor MI mais alto foram escolhidas também em ordem decrescente.

O número de padrões do conjunto foi escolhido de acordo com o número de entradas da rede neural, conforme será visto na seção 4.2.2.

3.4.3 Criação do Vetor Característico

Após a escolha das palavras para cada método de seleção de características, é necessário gerar o vetor característico para cada e-mail. Nessa fase, é determinada a gama de valores que cada elemento do vetor deve possuir. Cada um desses elementos corresponde a um nó de entrada da Rede Neural Artificial. Uma mensagem é representada por um vetor $X = (x_1, x_2, ..., x_n)$, onde n é o tamanho do vetor característico e $x_i; (1 \leq i \leq n)$, denota o valor da palavra na posição i do vetor característico para referida mensagem.

Na literatura existem algumas formas comumente utilizadas para compor o vetor característico, entre as quais podem ser citadas:

- *Freqüência do Termo* (*Term Frequency* - TF), igual ao número de vezes que a palavra ocorre no e-mail;

- *Peso binário* (*Binary weighting*), se uma palavra i ocorre pelo menos uma vez no e-mail, a_i é 1, se não ocorre, a_i é 0;

- *Peso normal* (*Normal weighting*), considerando-se as palavras do e-mail cujo vetor característico está sendo gerado, é escolhida a palavra com o maior número de ocorrências. Este número é escolhido como referência superior, ou seja, o valor máximo +1/2. E o limite inferior, -1/2, é atribuido às palavras que não ocorreram no e-mail. A partir disso, é feita a normalização do vetor característico.

O software desenvolvido em Java (figura 3.1) permite a escolha do diretório com arquivos de e-mails a serem processados, o número de elementos desejados para o vetor característico e o arquivo com as palavras selecionadas por um dos métodos estatísticos, gerando um arquivo de saída com os vetores para cada e-mail da base de dados. No apêndice D é mostrado um e-mail pré-processado e os vetores característicos gerados a partir dele. No próximo capítulo será explicado o treinamento realizado e os resultados obtidos.

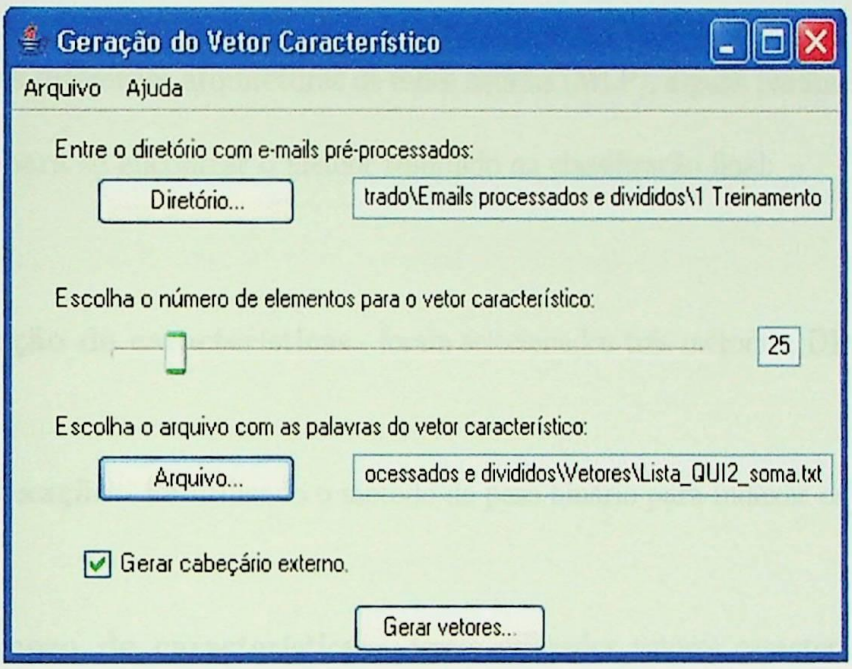

Figura 3.1: Programa em Java - Geração do Vetor Característico

CAPÍTULO 4

RESULTADOS OBTIDOS

À rede neural, foi aplicada a base de e-mails *SpamAssassin*[1], conforme descrito em 3.1.2.

Para as diferentes arquiteturas de redes neurais (MLP), alguns parâmetros foram avaliados para se encontrar o melhor resultado na classificação final:

- **Seleção de características** - foram selecionados três métodos, DF, χ^2 e MI;
- **Indexação** - foi utilizado o método de peso binário para indexar os e-mails;
- **Número de características** - foram utilizados vetores característicos com tamanhos diferentes para entrada da rede.

Nota-se que a combinação dos parâmetros acima permite obter algumas formas de configuração para a rede, aumentando ainda mais as possibilidades. Além disso, neste capítulo é mostrado o efetivo treinamento da rede neural, as arquiteturas testadas, a importância da pré-filtragem e os resultados obtidos, bem como a forma usada para a medida de desempenho da filtragem relizada.

[1]Obtidas no Site do autor [22], com numerações 20030228 e 20050311.

4.1 Medidas de Desempenho

A fim de se comparar os resultados obtidos para cada cada modificação de parâmetros do filtro anti-spam, foi necessário utilizar algumas medidas de desempenho. Em problemas de classificação, as medidas mais utilizadas são a *precisão* e o *recall [23]*. Foram usadas as letras S e H para *spam* e *ham* respectivamente, $n_{S\to S}$ e $n_{H\to H}$ como o número de *spams* e *hams* corretamente classificados, $n_{H\to S}$ como o número de mensagens legítimas erroneamente classificadas em *spam* (falso positivo) e $n_{S\to H}$ como o número de *spams* classificados erroneamente em mensagens legítimas (falso negativo).

Precisão pode ser definida como o número de mensagens de um conjunto de testes corretamente classificadas em uma categoria dividido pelo número total de mensagens classificadas (correta ou incorretamente). Assim sendo, define-se a precisão de *spam* (SP) e de *ham* (HP) como:

$$SP = \frac{n_{S\to S}}{n_{S\to S} + n_{H\to S}} \tag{4.1}$$

$$HP = \frac{n_{H\to H}}{n_{H\to H} + n_{S\to H}} \tag{4.2}$$

Recall pode ser definido como o número de mensagens de um conjunto de testes corretamente classificadas em uma categoria dividido pelo número total de mensagens que são realmente da categoria. Desta forma, define-se *recall* de *spam* (SP) e de *ham* (HP) como sendo:

$$SR = \frac{n_{S\to S}}{n_{S\to S} + n_{S\to H}} \tag{4.3}$$

$$HR = \frac{n_{H\to H}}{n_{H\to H} + n_{H\to S}} \tag{4.4}$$

Pode-se medir também a exatidão (Acc), que indica a proporção de classificações corretas, e o erro (Err), que indica a proporção de classificações incorretas:

$$Err = \frac{n_{S \to H} + n_{H \to S}}{N_S + N_H} \tag{4.5}$$

$$Acc = 1 - Err \tag{4.6}$$

onde N_S e N_H é o número total de *spams* e *hams* respectivamente.

Quando se usa um filtro anti-spam, é importante notar que classificar erroneamente um e-mail legítimo como *spam* é muito mais grave que deixar um *spam* passar como e-mail legítimo, de acordo com Zhang *et al.* [20] e Androutsopoulos [17]. Deixar um *spam* passar pelo filtro geralmente não causa prejuízo grave ao usuário, enquanto que bloquear um e-mail legítimo (ou mesmo marcá-lo como *spam*) pode ser algo sério, já que o usuário deixará de lê-lo ou só o verá quando acessar a pasta de *spam* (o que normalmente não é feito, nem desejado). Os valores de precisão e *recall* mostrados anteriormente não dizem nada sobre o desempenho do filtro quando um falso positivo ou um falso negativo apresentam pesos diferentes.

Dessa forma, para introduzir um peso maior para um falso positivo do que para um falso negativo, criou-se uma medida de exatidão ponderada (*Weighted Accuracy - WAcc*). O *WAcc* foi introduzido por Androutsopoulos [17] e tem sido utilizado em diversos testes de filtro anti-spam. A exatidão ponderada é dada por:

$$WAcc = \frac{\lambda n_{H \to H} + n_{S \to S}}{\lambda N_H + N_S} \tag{4.7}$$

Da mesma forma, o erro ponderado é dado por:

$$WErr = 1 - Acc = \frac{\lambda n_{H \to S} + n_{S \to H}}{\lambda N_H + N_S} \tag{4.8}$$

O WAcc trata uma mensagem legítima como se ela fosse λ mensagens: quando um falso positivo ocorre, é contado como λ erros; e quando é classificado corretamente, são contados λ acertos. Nota-se que, quanto maior λ, maior a penalização de um falso positivo.

Androutsopoulos também introduziu três valores diferentes para λ: 1, 9 e 999. Quando λ é um, *spams* e *hams* têm pesos iguais, enquanto que, para λ igual a 9, um falso positivo é penalizado nove vezes mais que um falso negativo. Já para λ igual a 999, classificar erroneamente um e-mail legítimo é tão ruim como deixar passar 999 *spams* pelo filtro. Esse último caso é aplicável somente em sistemas onde mensagens marcadas como *spams* são apagadas diretamente, sem que o usuário saiba. Na prática, usar um valor tão alto como 999 faz com que $WAcc$ seja tão alto que possa ser mal interpretado [20]. Para evitar este problema, o mesmo autor faz uma comparação com uma base mais simples: onde nenhum filtro está presente - assim mensagens legítimas nunca são bloqueadas e *spams* não são classificados. A versão de base do erro e da exatidão são definidas como:

$$WAcc^b = \frac{\lambda N_H}{\lambda N_H + N_S} \tag{4.9}$$

$$WErr^b = \frac{\lambda N_S}{\lambda N_H + N_S} \tag{4.10}$$

Para finalizar, Androutsopoulos apresenta também o *total cost ratio* (TCR) como o parâmetro final de medida de desempenho dos filtros anti-spam:

$$TCR = \frac{WErr^b}{WErr} \tag{4.11}$$

Dessa forma, quanto maior TCR, melhor é o filtro, e TCR deve ser sempre maior que um, ou seja, $WErr$ será menor que o valor de base $WErr^b$. Se TCR for menor que um significa que o sistema é pior do que o valor de base - o filtro está prejudicando o desempenho - e seria melhor não utilizá-lo.

Agora que já foram expostas as diversas formas de medida de desempenho, serão mostrados como os dados foram utilizados, assim como os experimentos realizados.

4.2 Dados utilizados e preparação dos experimentos

Conforme discutido na seção 3.2.2, a base de dados SpamAssassin foi dividida em três subconjuntos:

- Treinamento (60%) - este conjunto continha 2484 *hams* e 2484 *spams*: utilizado para treinar a rede neural;

- Testes (20%) - este conjunto continha 832 *hams* e 832 *spams*: utilizado como teste da rede para verificar o treinamento da mesma;

- Validação (20%) - este conjunto continha 826 *hams* e 826 *spams*: utilizado para verificar a capacidade de generalização da rede, além de medir seu desempenho final.

4.2.1 Configuração dos Experimentos

Após a divisão dos dados, foram gerados os vetores de entrada da rede neural para diversos casos. A primeira variável foi o tamanho do vetor característico. Foram testados vetores de dimensões 6, 12 e 25. Isto significa que a rede neural deve ter essas dimensões de entrada para cada caso.

Para cada dimensão do vetor característico, foram gerados os casos de testes, variando-se o método de seleção de características, DF, χ^2 e MI. Finalmente, para cada método de seleção de característica, foram gerados vetores utilizando indexação por peso binário.

4.2.2 Arquitetura da Rede Neural

A rede neural (MLP) foi avaliada de acordo com os experimentos realizados. Ela foi configurada para ter 6, 12 e 25 unidades de entrada. Diferentes arquiteturas, incluindo de três até vinte unidades na camada intermediária (ou oculta) foram utilizadas. A camada de saída continha unidades lineares para evitar *flat spots*, de acordo com Fahlman *[31]*.

A ativação de cada *perceptron* da camada oculta foi atribuída à função sigmoidal. O treinamento foi feito baseado em épocas. Ao final de cada época, a taxa de aprendizado e o momento foram modificados, e o erro total calculado.

O treinamento foi realizado com validação cruzada, isto é, ele é interrompido sempre que o erro total aumenta no conjunto de testes. A taxa de treinamento é sempre reduzida em 50% quando o erro total aumenta, e aumentada de 2% quando

o erro diminui. O momento é desativado até o final do treinamento se o erro total aumenta. Na saída da rede foram utilizadas duas unidades em todos os experimentos:

- (0 1) quando os padrões negativos são apresentados à entrada da rede, isto é, um e-mail legítimo ou *ham*;
- (1 0) quando os padrões positivos, ou seja, um *spam* é apresentado à entrada da rede.

Para inicialização dos pesos, foi utilizado um valor aleatório entre [-0,5; 0,5].

4.3 Resultados Experimentais

Nesta seção serão mostrados os resultados dos experimentos realizados, bem como a análise dos resultados obtidos para o conjunto de validação.

4.3.1 MLP com 6 nós de entrada

Foram realizados três experimentos, com 6 nós de entrada, usando DF, χ^2 e MI. Os resultados obtidos estão apresentados nas tabelas 4.1, 4.2 e 4.3, onde SP: *spam precision*; HP: *ham precision*; SR: *spam recall*; HR: *ham recall*; WAcc: *weighted accuracy*; TCR: *total cost ratio.*

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
87,05%	90,72%	91,16%	86,44%	1	88,80%	50,00%	4,47
				9	86,91%	90,00%	0,76
				999	86,45%	99,90%	0,01

Tabela 4.1: Experimento 1 - Rede com 6 entradas, DF binário

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
93,76%	96,38%	96,49%	93,58%	1	95,04%	50,00%	10,07
				9	93,87%	90,00%	1,63
				999	93,59%	99,90%	0,02

Tabela 4.2: Experimento 2 - Rede com 6 entradas, qui quadrado binário

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
94,50%	97,62%	97,70%	94,31%	1	96,00%	50,00%	12,52
				9	94,65%	90,00%	1,87
				999	94,31%	99,90%	0,02

Tabela 4.3: Experimento 3 - Rede com 6 entradas, MI binário

No primeiro experimento, a rede classificou incorretamente 112 *hams* e 73 *spams*. No segundo, pode-se observar uma melhora, já que a rede classificou incorretamente 53 e 29 padrões de *ham* e *spam*, respectivamente. Isso indica de que o método de seleção de características utilizado é importante na classificação. No terceiro experimento, a rede classificou incorretamente 47 *hams* e 19 *spams*. Observando o valor TCR, que indica o desempenho final da rede, nota-se que a mesma só é viável para o DF com 6 entradas se classificar um *spam* incorretamente tiver o mesmo peso que classificar um e-mail legítimo ($\boldsymbol{\lambda}$=1). Já para os métodos χ^2 e MI podem ser usados tanto $\boldsymbol{\lambda}$=1 quanto $\boldsymbol{\lambda}$=9, indicando uma melhoria.

4.3.2 MLP com 12 nós de entrada

Neste caso, aumentou-se o número de nós de entrada da MLP para doze unidades e foram realizados testes com DF, χ^2 e MI. Os resultados do desempenho da rede são apresentados nas tabelas 4.4, 4.5 e 4.6, onde SP: *spam precision*; HP: *ham precision*; SR: *spam recall*; HR: *ham recall*; WAcc: *weighted accuracy*; TCR: *total cost ratio*.

Nota-se uma melhora significativa em relação à rede com apenas 6 *perceptrons* de

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
91,57%	95,80%	96,00%	91,16%	1	93,58%	50,00%	7,79
				9	91,65%	90,00%	1,20
				999	91,17%	99,90%	0,01

Tabela 4.4: Experimento 4 - Rede com 12 entradas, DF binário

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
97,36%	98,17%	98,18%	97,34%	1	97,76%	50,00%	22,32
				9	97,42%	90,00%	3,88
				999	97,34%	99,90%	0,04

Tabela 4.5: Experimento 5 - Rede com 12 entradas, qui quadrado binário

entrada. Observa-se também que os resultados obtidos utilizando χ^2 e MI são bem melhores que a distribuição DF. No quarto experimento, a rede classificou incorretamente 73 e-mails legítimos e 33 *spams*, no quinto experimento os erros foram de 22 *hams* e 15 *spams* e, no sexto, foram classificados incorretamente 27 *hams* e 12 *spams*.

4.3.3 MLP com 25 nós de entrada

Finalmente, aumentou-se o número de nós de entrada da MLP para vinte e cinco unidades e foram realizados testes com DF, χ^2 e MI. Os resultados do desempenho da rede desse caso estão apresentados nas tabelas 4.7, 4.8 e 4.9, onde SP: *spam precision*; HP: *ham precision*; SR: *spam recall*; HR: *ham recall*; WAcc: *weighted accuracy*; TCR: *total cost ratio*.

No sétimo experimento, a MLP classificou incorretamente 23 padrões de *ham* e

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
96,79%	98,52%	98,55%	96,73%	1	97,64%	50,00%	21,18
				9	96,91%	90,00%	3,24
				999	96,73%	99,90%	0,03

Tabela 4.6: Experimento 6 - Rede com 12 entradas, MI binário

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
97,14%	94,69%	94,55%	97,22%	1	95,88%	50,00%	12,15
				9	96,95%	90,00%	3,28
				999	97,21%	99,90%	0,04

Tabela 4.7: Experimento 7 - Rede com 25 entradas, DF binário

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
100%	99,16%	99,15%	100%	1	99,58%	50,00%	118
				9	99,92%	90,00%	118
				999	100%	99,90%	118

Tabela 4.8: Experimento 8 - Rede com 25 entradas, qui quadrado binário

45 padrões de *spam*. No oitavo experimento, a rede classificou corretamente todos os padrões de *ham* e incorretamente apenas 7 padrões de *spam*. Por fim, no oitavo experimento a rede classificou corretamente todos os padrões de *ham* e *spam*.

Os resultados obtidos são muito promissores. A porcentagem de classificações corretas é bem alta, principalmente no caso MI com 25 entradas, onde não houve nenhuma classificação incorreta. Isto mostra que a MLP é capaz de generalizar para diferentes tipos de padrões de *hams* e *spams*.

4.4 Análise dos Resultados

Observando os resultados, nota-se que a rede neural foi capaz de classificar corretamente boa parte dos *spams*, principalmente em um dos testes, onde não houve classificação errada de nenhum e-mail legítimo e *spam*, que é um resultado excep-

SP	HP	SR	HR	λ	WAcc	WAcc^b	TCR
100%	100%	100%	100%	1	100%	50,00%	∞
				9	100%	90,00%	∞
				999	100%	99,90%	∞

Tabela 4.9: Experimento 9 - Rede com 25 entradas, MI binário

cional. Nesse caso, o TCR da rede é infinito, indicando um caso de filtro ideal, conforme foi mostrado na seção 4.1.

A tabela 4.10 apresenta um resumo dos resultados obtidos. Os valores de classificação correta de *ham* e *spam* desta tabela correspondem, respectivamente, aos valores de *Ham Recall* (HR) e *Spam Recall* (SR) das tabelas anteriores.

Exp.	N° entradas	Seleção caract.	Classific. correta ham (%)	Classific. correta spam (%)
1	6	DF	86,44	91,16
2	6	χ^2	93,58	96,49
3	6	MI	94,31	97,70
4	12	DF	91,16	96,00
5	12	χ^2	97,34	98,18
6	12	MI	96,73	98,55
7	25	DF	97,22	94,55
8	25	χ^2	100,00	99,15
9	25	MI	100,00	100,00

Tabela 4.10: Resultados dos experimentos

Para um melhor entendimento da qualidade destes resultados, é feita uma comparação com resultados reportados por um artigo de Chuan, Xianliang, Mengshu e Xu [28], onde foram realizados experimentos com três diferentes modelos de filtro anti-spam com a base de dados *Spam Assassin Public Corpus*. O primeiro utilizou um classificador *Naïve Bayesian* (NBC), o segundo utilizou uma rede de *perceptrons* com múltiplas camadas (MLP), enquanto que o terceiro utilizou uma *learning vector quantization* (LVQ), um modelo mais complexo de redes neurais. Nos dois últimos testes eles utilizaram 100 entradas na rede neural.

No artigo, eles reportaram que o NBC atingiu uma taxa de 86,48% de classificações corretas de padrões *spam*, enquanto que os modelos MLP e LVQ atingiram

91,26% e 93,58%, respectivamente. Estes resultados são pobres quando comparados com os mostrados na tabela 4.10. Os autores deste trabalho removeram todos os anexos e informação HTML dos e-mails da base SpamAssassin. Desse modo, pode-se observar a importância dessas partes no filtro anti-spam, validando o trabalho do pré-filtro.

Uma das razões para as taxas de classificações corretas apresentadas pela rede se deve aos dois métodos de seleção de características utilizados: o χ^2, que apresentou resultados muito bons em diversas referências, como em Goetschi [23], Yang e Pedersen [24] e Zhang *et al.* [20]; e o MI, cujo alto desempenho foi contrário ao encontrado por Yang e Pedersen [24]. O MI foi aplicado em alguns trabalhos como Androutsopoulos *et al.* [17], Chuan *et al.* [28] e Özgür, Güngör e Gürgen [5]. Nesse trabalho, as redes testadas apresentaram um número menor de padrões de entradas (ao contrário dos demais citados neste parágrafo). Isso é importante porque, com uma rede menor, o tempo de treinamento se torna menor e a tarefa de otimização mais rápida. O tempo de classificação é extremamente crítico para um filtro, pois, em um servidor, podem passar por minuto centenas de e-mails, de acordo com Gomes *et al.* [30].

Comparando o desempenho do filtro anti-spam, variando o número de atributos (no caso da medida de desempenho para o filtro onde não há peso negativo ao classificar erroneamente um e-mail legítimo, figura 4.1), nota-se que para o caso de DF há uma pequena melhora conforme o aumento do número de atributos, mas que não é suficiente para se chegar perto do ganho com χ^2 e MI. Nesse caso, as medidas de desempenho indicam que todos os métodos de seleção de características poderiam ser usados sem problemas, já que apresentam sempre TCR>1.

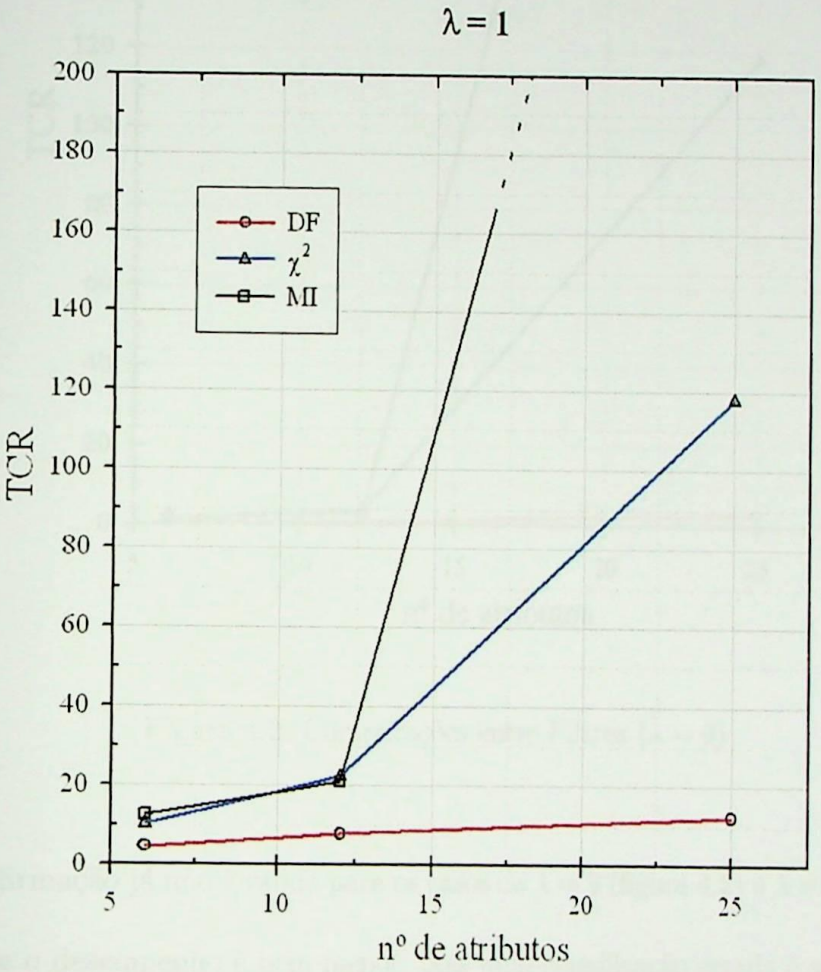

Figura 4.1: Comparações com Filtros *Baseline* ($\lambda = 1$)

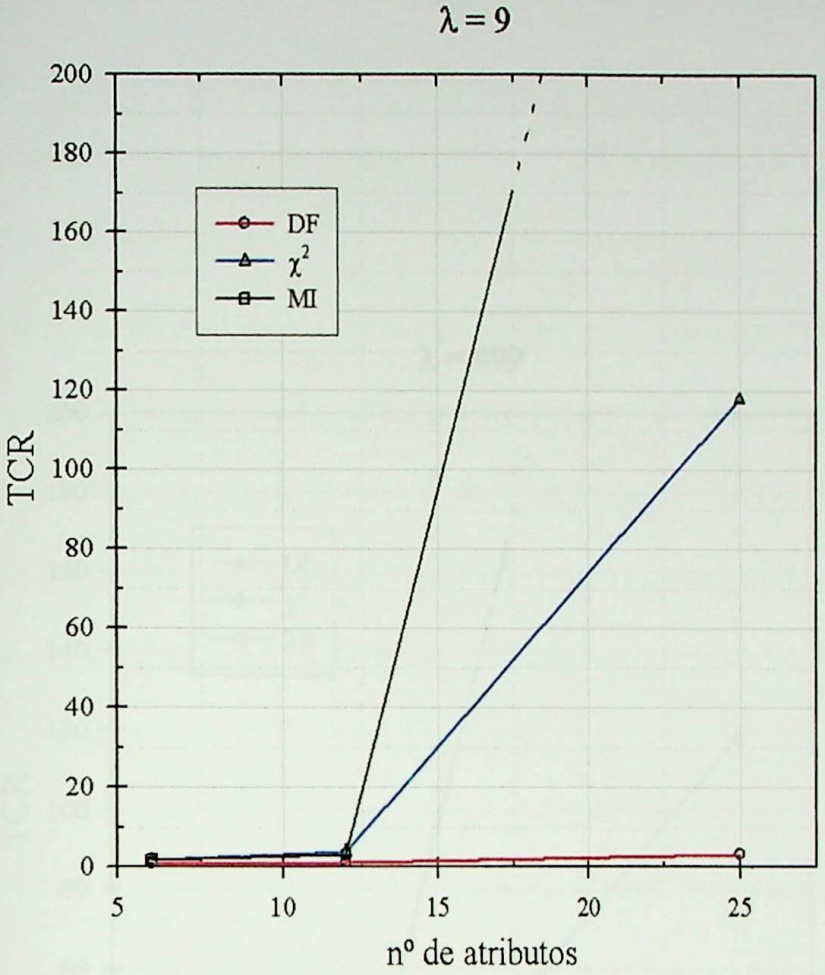

Figura 4.2: Comparações entre Filtros ($\lambda = 9$)

Tal afirmação já não é válida para os casos de $\lambda = 9$ (figura 4.2) e $\lambda = 999$ (figura 4.3), onde o desempenho é bem menor, pois uma classificação errada é severamente punida na avaliação de desempenho. Para $\lambda = 9$, usando DF, o filtro ainda é viável, caso a rede contenha 12 ou 25 entradas e, usando χ^2 e MI, a rede é viável em todos os casos. Contudo, no caso de $\lambda = 999$, o sistema anti-spam só seria viável para os casos onde se utiliza χ^2 e MI com 25 entradas.

Uma observação importante a ser feita é que em nenhuma referência do presente trabalho, onde foram feitas as avaliações de desempenho apresentadas, foi encontrado

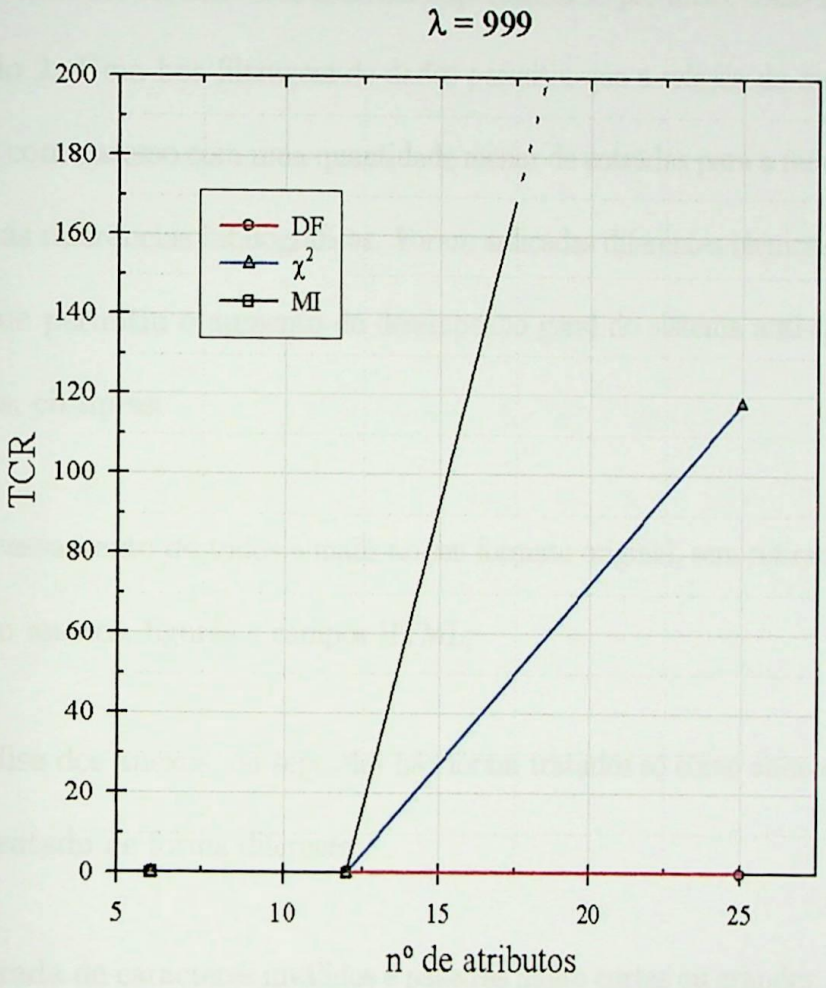

Figura 4.3: Comparações entre Filtros ($\lambda = 999$)

um valor de TCR tão alto quanto aquele obtido por MI em uma rede neural com 25 entradas. Isso indica a qualidade do trabalho e comprova, mais uma vez, a fundamental importância da pré-filtragem realizada, além da correta seleção de características e treinamento da rede neural.

Neste trabalho foi atribuído extrema importância ao pré-filtro, como foi explicado no capítulo 2. Uma boa filtragem de dados permitiu que a seleção de características fosse feita com sucesso com uma quantidade menor de entradas para a rede que aquela adotada nas referências bibliográficas. Foram aplicadas diferentes técnicas para o pré-filtro, o que permitiu o aumento do desempenho geral do sistema anti-spam. Entre as técnicas, citam-se:

- Processamento de todos e-mails no seu formato original, sem retirada de partes como anexos, figuras e campos HTML;

- Análise dos anexos, ou seja, eles não foram tratados só como anexos, cada tipo foi tratado de forma diferente;

- Retirada de caracteres inválidos e palavras muito curtas ou grandes, permitindo uma melhora na seleção de características;

- Processamento complexo dos textos HTML com diferentes algoritmos para extrair o máximo de informação relevante para o sistema, isso também permitiu uma melhora na seleção de características e ganhos no sistema como um todo;

- Utilização de diversos algoritmos para detecção de padrões típicos de *spammers*.

4.5 Distribuição do Software

Conforme exposto anteriormente, o software anti-spam foi desenvolvido em sua maior parte em Java: tanto o sistema de pré-filtragem como a seleção de característica e montagem do vetor característico.

O projeto foi desenvolvido usando orientação a objetos, o que garante modularidade, flexibilidade e possibilidade de reutilização do software. Todos os métodos desenvolvidos foram bem documentados.

Existe a possibilidade de que o software seja distribuído, primeiramente no meio acadêmico, com intuito de novas melhorias e adaptações, funcionando na prática para o controle de *spams*. Após tais testes acadêmicos, há também a possibilidade do pacote ser distribuído na Internet como uma ferramenta aberta (*open-source*). Qualquer desenvolvedor poderia criar novos módulos de integração ao sistema, fazendo com que seu desempenho aumente, conforme as técnicas dos *spammers* se alteram. Desse modo, o sistema ficaria sempre atualizado, garantindo o seu bom desempenho por tempo indeterminado.

CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposto o desenvolvimento de um filtro anti-spam com uma rede neural artificial como classificador. As redes neurais artificiais são usadas em diversos trabalhos de classificação de texto, por isso a escolha para o método de classificação.

Os *spams*, que são e-mails indesejados recebidos por qualquer pessoa com uma conta de e-mail na Internet, normalmente contêm informações de publicidade que não foram requisitadas. Os *spammers* (enviadores de *spams*), podem conseguir os endereços de e-mails de diversas formas: comprar listas de e-mails que certas empresas criam, rastrear a Internet na busca de e-mails em páginas de *sites*, colocar nomes aleatórios antes de domínios conhecidos (ex.: nome_qualquer@yahoo.com).

Os primeiros filtros anti-spam eram bastante simples, pois os *spams* continham informações óbvias, por exemplo, no assunto da mensagem. O usuário simplesmente colocava um filtro em seu gerenciador de e-mails. Com o desenvolvimento de novas formas de *spam*, a tarefa tornou-se mais difícil. Assim, surgiram novas técnicas que exigiam filtros cada vez mais complexos e com melhor capacidade de detecção e adaptação.

Analisando-se diversas propostas de filtro anti-spam atualmente em uso, observou-se que as mesmas não apresentaram resultados tão bons e taxas de falsos positivos (classificar e-mails legítimos como *spams*) desejáveis para o usuário final.

A proposta foi, então, o desenvolvimento de um sistema anti-spam diferente dos desenvolvidos até o momento. Esse daria ênfase maior ao pré-filtro, pois os atuais sistemas anti-spam se preocupam mais com melhoria da classificação. Com o pré-filtro, o sistema tornou-se capaz de selecionar melhor as características e o classificador mostrou um desempenho muito melhor, pois o uso de técnicas de categorização simplifica o trabalho da MLP. Em conseqüência, o conjunto de dados se tornou mais simples e uniforme, além de ter partes desnecessárias removidas.

Assim, foram estudadas algumas técnicas de camuflagem reportadas até o momento e desenvolvidos algoritmos que permitiram detectar as tentativas de enganar os filtros que não seriam notadas por nenhum sistema classificador atual, inclusive as redes neurais. Muitas informações, antes ignoradas por diversos trabalhos, foram aproveitadas, mesmo padrões ainda desconhecidos foram tratados pelo sistema.

Para que estas técnicas de pré-filtragem pudessem ser aplicadas com sucesso, foi necessário uma base de dados com e-mails em seu formato original. Também, para realizar o treinamento da rede neural, foi necessário obter uma base de dados grande, tanto com e-mails legítimos, quanto *spams*. Assim sendo, foi escolhida uma base reconhecida na comunidade científica e por empresas desenvolvedoras de filtros anti-spam: a base pública *SpamAssassin*. Essa base de dados possui todas as características desejáveis: além de ter uma quantidade razoável de e-mails *spams* e *hams*, mantém os e-mails em sua forma original, sem retirar nenhuma informação. Em outras bases de dados existentes, aplicadas em trabalhos de diversos artigos, como a base LingSpam [18], os campos HTML e os anexos foram removidos.

Após a filtragem dos dados, foi efetuada a separação dos dados em três sub-

conjuntos: treinamento, teste e validação. O conjunto de treinamento e o de teste foram usados durante a fase de treinamento da rede neural. O conjunto de validação, para validar a rede neural, ou seja, fazer medidas de desempenho e avaliação final do sistema.

As redes neurais têm como entrada um vetor, chamado de vetor característico. Esse vetor contém informações, as mais relevantes possíveis, para aprendizado eficiente. Para esse propósito, foi realizado a seleção estatística, a fim de se calcular quais características comporiam o vetor característico. Para a classificação textual, as características foram as palavras originais dos e-mails e também as palavras inseridas pelo pré-filtro, de acordo com algoritmos. Na literatura existem diversas formas de se realizar a seleção. No presente trabalho foram usadas as técnicas DF, χ^2 e MI. Para todas as palavras encontradas foram efetuados os cálculos. Dessa forma, as palavras que compuseram o vetor característico foram determinadas. Para cada técnica, um vetor diferente foi gerado.

Determinadas as caracteríticas, cada e-mail foi transformado em um vetor. Se uma dada palavra do e-mail aparece no vetor característico, a posição do vetor é incrementada. Caso a palavra não exista, nada é feito. Se uma palavra no vetor característico não foi encontrada no e-mail, é colocado zero em sua posição no vetor. Para cada e-mail foram gerados três vetores diferentes: um que indica a quantidade de vezes que a palavra está presente (indexação por freqüência do termo), outro, indicando se a palavra ocorreu ou não, independente do número de vezes (indexação binária) e, finalmente, o que é normalizado pela palavra com maior número de ocorrências em um dado e-mail na base de dados (indexação normal).

Finalmente, realizou-se o treinamento da rede neural MLP. Foram testadas diversas arquiteturas MLP, com número de *perceptrons* variando na camada oculta, além da variação da taxa de aprendizado e momento. Assim, obteve-se a rede com melhor desempenho para o sistema. O número de entradas foi alterado na busca da melhor rede, ou seja, melhor relação custo/benefício - já que uma rede com número de entradas muito grande apresentaria um tempo de treinamento muito alto, além da demora na classificação de novos padrões.

Após a obtenção dos dados finais da rede neural, foram realizadas as medidas de desempenho da rede. Os dados encontrados foram muito promissores, principalmente no caso da MLP com apenas 25 entradas, usando MI e indexação binária. Nesse caso, não houve nenhuma classificação incorreta de e-mails legítimos e *spams*, no conjunto de validação que continha 1652 padrões de e-mails. Este resultado é excepcional e não foi encontrado em nenhuma referência tal qualidade de classificação. Os resultados desse trabalho foram apresentados na *Lecture Notes in Computer Science* [25]. Este resultado excelente valida todo o trabalho desenvolvido no pré-filtro, indicando que é uma parte importante do sistema e não pode ser ignorado.

Cabe observar que é desejável que sempre haja atualizações no sistema, buscando adaptá-lo a novos padrões de *spams* que possam surgir. Um sistema rígido (que não permite mudanças) logo tornar-se-ia ultrapassado. O pacote desenvolvido permite que novas técnicas de pré-filtragem sejam integradas facilmente ao sistema, propiciando adaptação rápida às mudanças ocorridas.

A base de dados *SpamAssassin,* utilizada no treinamento da rede neural, é constituída de e-mails somente em inglês. Desta forma, as palavras específicas (que não

são resultantes da detecção de padrões e das *tags* HTML) presentes nos vetores característicos estão em inglês. A princípio, isto pode não ser um problema grave, já que a grande maioria dos *spams* recebidos estão em inglês (de 90 a 95%, de acordo com Vauhini Vara [34] - o que confere com o maior originador de *spams*, os Estados Unidos [35]). Entretanto, estas estatísticas podem mudar com o passar do tempo. Portanto, o sistema deverá ser capaz de detectar corretamente os padrões de *spam* recebidos. As palavras que estão no vetor característico que não são dependentes do idioma (resultantes da detecção de padrões e das *tags* HTML) podem ajudar na detecção correta de *spams* em outras linguagens, mas a precisão do filtro diminuiria e o número de falsos positivos e falsos negativos poderia aumentar a um nível não desejável. Por esta razão, na próxima seção serão mostradas duas soluções que poderão ser utilizadas para contornar este problema.

5.1 Trabalhos Futuros

Durante o trabalho e após a conclusão do mesmo, foram encontrados alguns pontos que poderiam ser modificados ou melhor trabalhados, proporcionando algumas melhorias no sistema.

A rede neural utilizada no trabalho foi a MLP com uma camada intermediária. Poderiam ser testadas outras arquiteturas, entre as quais, podem-se citar:

- Alteração no número de camadas intermediárias e *perceptrons* nessas camadas;
- Testar vetores característicos com outros tamanhos;
- Utilização de outros tipos de redes neurais, como as redes de Kohonen. As

redes de Kohonen usam uma arquitetura diferente das MLP. O treinamento é não-supervisionado e a rede aprende novos padrões através da reorganizaçao dos *perceptrons* vizinhos em torno do vencedor (aquele que apresenta um resultado melhor para o padrão de entrada). Esta região se especializa em um determinado tipo de entrada, como por exemplo, *spams*, e regiões diferentes classificam padrões diferentes.

Poderiam ser testados outros tipos de classificadores, como os sistemas baseados em lógica *Fuzzy* com aprendizagem, utilizados como classificadores em diversas áreas, como no trabalho de Nozaki, Ishibuchi e Tanaka [36] e no trabalho na área de lingüística de Alcala *et al.* [37]. Alguns autores apresentaram classificadores baseados em regras com aprendizado, como Golding e Rosenbloom [38]. Em outros trabalhos foram criados sistemas mistos, baseados em sistemas *fuzzy* com regras e também sistemas *fuzzy* com redes neurais, como os apresendados por Cordón, del Jesus e Herrera [39] e por Castellano e Fanelli [40], respectivamente. Com o uso de sistemas mistos, isto é, que utilizam diferentes técnicas de inteligência artificial, existe a possibilidade de aproveitar as melhores características de cada técnica, podendo serem obtidos melhores resultados do que os obtidos por cada técnica isoladamente, de acordo com Medsker [41]. Desta forma, são interessantes de serem analisados e comparados com o presente trabalho.

Nesse trabalho foram testados três métodos de seleção de características, o DF, χ^2 e MI . Existem outros métodos, como o *Information Gain* (IG), que também pode ser testado, e que apresentou bons resultados em alguns trabalhos, como os de Yang

e Pedersen, [24], Zhang *et al.* [20] e de Sakkis e Androutsopoulos *et al.* [19]. O IG mede o número de bits de informação obtidos para a predição de categorias através do conhecimento da presença ou ausência de um termo em uma mensagem. Já para gerar o vetor característico, foi utilizado a técnica de indexação binária. A indexação normal também poderia ser aplicada para comparação.

A base de dados SpamAssassin foi a única testada. Dentre as encontradas em diversas referências e na Internet, esta foi a única julgada ideal para o trabalho, pois continha os e-mails originais, ao contrário da LingSpam, PU1 e PU123A. O sistema poderia ser testado para validação em outras bases de e-mails, caso estas estejam disponíveis e contenha os e-mails sem alteração.

O pré-filtro foi a parte mais explorada no presente trabalho. O sistema foi todo desenvolvido em Java, utilizando Orientação a Objetos, além de estar todo bem documentado. Dessa forma, adicionar uma nova funcionalidade ao pré-filtro não exige grandes adaptações ao sistema. Devido à forma modular com que foi desenvolvido, adicionar novos algoritmos de detecção de novos padrões e tentativas de camuflagem de *spams* é mais simples.

No trabalho foram criados diversos algoritmos, como aparece na seção 2.2.3. Poderiam ser adicionados outros algoritmos para detectar novos padrões à medida em que estes forem surgindo. Dentre algumas melhorias, podem-se citar:

- Algoritmo de detecção de texto em imagens. Alguns *spams* mais recentes apresentam um texto considerado legítimo pelos filtros, mas contêm uma imagem que é exibida antes do texto, que nada mais é do que a imagem do texto com o

spam. Desta forma, os filtros podem classificar o e-mail como válido. Este tipo de algoritmo de detecção não foi implementado no trabalho pois adicionaria um tempo computacional alto. Tal implementação talvez seja viável em um futuro próximo, quando o tempo de reconhecimento de texto em imagens não seja um fator impactante no desempenho geral do sistema. Uma idéia razoável no momento é estudar estas imagens visando a extração de alguma informação relevante, que não exija cálculos computacionais intensivos, como a cor de fundo da imagem e tamanho. Estes valores seriam salvos como informação e poderiam ser utilizados na seleção de características (como criar um campo específico no vetor característico);

- Aperfeiçoar o algoritmo de detecção de palavras separadas por caracteres inválidos, como espaços, pontos, traços, etc. Os *spammers* podem separar uma palavra de diversas formas e, mesmo assim, fazer com que ela continue legível para o ser humano, mas consegue passar pelo filtro sem ser detectada;

- Criar um algoritmo que analise outros tipos de cabeçalhos dos e-mails. No presente trabalho apenas o assunto foi avaliado pelo pré-filtro. Zhang, Zhu e Yao [20] constataram que classificadores usando vetor característico com apenas o cabeçalho, atingiram resultados comparáveis àqueles com apenas a mensagem principal. Algumas entradas na rede neural poderiam ser dedicadas às características extraídas apenas do cabeçalho.

Os algoritmos poderiam ser integrados na forma de *plugins*, ou seja, módulos criados por outros desenvolvedores, sendo adicionados ao sistema através de uma

interface gráfica específica. Vários módulos seriam agregados, assim como os atuais poderiam ser aperfeiçoados ou removidos, de acordo com a necessidade. Uma interface gráfica completa poderia ser desenvolvida para o controle de todo o sistema. A ordem de execução dos *plugins* também seria alterada, de acordo com as informações tratadas.

Lee, Hui e Fong [27] testaram uma forma diferente de compor o vetor característico: 55 entradas foram obtidas na seleção de dados, porém eles usaram entradas dedicadas a outras informações, como a posição de ocorrência de certas informações específicas e o número de ocorrência de certos termos. O trabalho não foi de classificação de *spams*, mas algumas entradas específicas da rede neural poderiam representar informações especiais, como a presença de imagens indicativas de *spam* e dados do cabeçalho.

O tipo de decisão adotado neste trabalho foi binário, ou seja, o e-mail é considerado *spam* ou legítimo, não existindo meio termo. Poderia ser explorada outra forma de classificação final, onde a decisão é linear. Isto é possível porque a saída da rede neural é linear - é dada uma pontuação ao e-mail na entrada que varia entre dois extremos indicando a classificação do mesmo. Neste trabalho usou-se uma faixa indicando um e-mail *spam*, e outra, *ham*. Pode-se, por exemplo, adicionar uma terceira faixa, onde a classificação não foi precisa ao indicar se o e-mail é um *spam* - ela seria intermediária as duas faixas fixas adotadas atualmente. As mensagens poderiam ser então salvas em uma terceira pasta do gerenciador de e-mail do usuário. A princípio pode parecer uma má solução, entretanto, considerando-se o alto custo de um falso positivo, verificar esta pasta pode não ser considerado grave. O usuário poderia modificar as faixas que deseja para cada pasta, ajustando melhor o filtro, de acordo com

suas preferências.

Existe ainda possibilidade de distribuir o software, inicialmente no meio acadêmico, para uso no controle de *spams*. A princípio, o software poderia ser integrado em um servidor para testes, onde seriam feitos os ajustes iniciais junto com o servidor de recebimento de e-mails POP3[1]. Posteriormente, de acordo com o desempenho obtido, o software seria integrado à servidores de grande tráfego de e-mails. Nessa fase, o sistema deverá ter um desempenho confiável o suficiente para não acrescentar atrasos no serviço de e-mails. A forma mais indicada para o tratamento dos e-mails classificados como *spams*, em um primeiro momento, seria adicionar um texto, como "[SPAM]", no assunto do e-mail. Além disso, seria acrescentado um filtro em cada gerenciador de e-mail para que estes fossem movidos para uma caixa de *spams*. Logo que o sistema fosse validado na prática, caso o usuário deseje, estes *spams* poderiam ser apagados automaticamente.

Após os testes no meio acadêmico - dependendo dos resultados obtidos - o sistema poderá ser distribuído na Internet sob licença GNU[2]. O desenvolvimento na Internet se dá de forma rápida, já que pessoas ou organizações de diversos lugares e conhecimento poderiam participar. Assim, novos *plugins* para o pré-filtro poderiam ser desenvolvidos, o que daria uma maior flexibilidade ao sistema desenvolvido.

Foram levantadas duas soluções para o caso da linguagem dos *spams*, citado na primeira parte deste capítulo. A primeira consiste na utilização de bases de dados diferentes para cada idioma desejado. Seria então necessário o treinamento de

[1]POP3 - Servidor responsável por entregar os e-mails para os usuários.

[2]Licença GNU disponível em http://www.gnu.org/licenses/. É usada para distribuição de software livre.

uma rede neural diferente para cada idioma. A precisão na classificação poderia ser aumentada sensivelmente, já que cada classificador teria nas suas entradas as palavras específicas corretamente escolhidas. Porém, existem algumas desvantagens nesta solução:

- Necessidade de bases de dados de *spams* diferentes para cada linguagem adotada;
- Seria necessário identificar o idioma de cada e-mail que é enviado ao sistema, o que não é uma tarefa trivial.

A segunda solução seria adotar uma base de dados com *spams* de todas as linguagens desejadas. Desta forma, não seriam necessárias grandes adaptações ao sistema já desenvolvido. Contudo, com o aumento do número de palavras diferentes, provenientes dos demais idiomas adicionados, o número de entradas da rede neural deverá ser aumentado para que a precisão de classificação não seja prejudicada. Ambas as soluções poderiam ser testadas e, a partir das medidas de desempenhos, a melhor poderia ser escolhida para ser adotada em aplicações práticas.

BIBLIOGRAFIA

[1] Fawcett, T., "“In vivo” spam filtering: A challenge problem for KDD", ACM SIGKDD Explorations 5 (2003) 140-148

[2] Página da Internet, SpamCop Statistics, Ironport Systems, Inc. Disponível em: *http://www.spamcop.net/spamgraph.shtml?spammonth.* Acesso em 08.10.2006.

[3] Cournane, A., Hunt, R., "An analysis of the tools used for the generation and prevention of spam", Computers & Security 23 (2004) 154-166

[4] Evett, D., Página da Internet, Spam Statistics 2006. Disponível em: *http://spam-filter-review.toptenreviews.com/spam-statistics.html.* Acesso em 08.10.2006.

[5] Özgür, L., Güngör, T., Gürgen, F., "Adaptive anti-spam filtering for agglutinative languages: a special case for Turkish", Pattern Recognition Letters 25 (2004) 1819-1831

[6] Beale, R., Jackson, T., "Neural Computing: An Introduction", IOP Publishing Ltd., Bristol, UK (1990)

[7] Bishop, C.M., "Neural Networks for Pattern Recognition", Oxford University Press (1995)

[8] Gomes, L.H., Cazita, C., Almeida, J.M., Almeida, V., Meira Junior, W., "Characterizing a spam trafic", Proceedings of the Internet Measurement Conference, ACM SIGCOMM (2004)

[9] Pfleeger, S.L., Bloom, G., "Canning spam: Proposed solutions to unwanted e-mail", IEEE Security & Privacy 3 (2005) 40-47

[10] Zorkadis, V., Karras, D.A., Panayotou, M., "Eficient information theoretic strategies for classifier combination, feature extraction and performance evaluation in improving false positives and false negatives for spam e-mail filtering", Neural Networks 18 (2005) 799-807

[11] Haykin, S., "Neural Networks: A Comprehensive Foundation", 2 edn, Prentice-Hall, Inc (1999)

[12] McCulloch, W. S., Pitts, W. "A Logical Cauculos of the Ideas Immanent in Nervus Activity", Bulletin of Mathematical Biophysics, vol 5, (1943) 115-133.

[13] Rosenblatt, R., "Principles of Neurodynamics", Spartan Books, New York (1959)

[14] Rosenblatt, R., "The perceptron: A perceiving and recognizing automaton", Report 85-460-1, Project PARA, Cornell Aeronautical Laboratory, Nova Iorque (1957)

[15] Minsky, M., Papert, S., "Perceptrons", MIT Press, Cambridge (1969)

[16] Rumelhart, D.E., Hinton, G.E., McClelland, J.L., "A general framework for parallel distributed processing", Rumelhart, D.E., McClelland, J.L., the PDP Research Group, eds.: Parallel Distributed Processing. Volume 1. The MIT Press, Cambridge, MA (1986) 45-76

[17] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D., "An evaluation of naive Bayesian anti-spam Filtering", Proceedings of the Workshop on Machine Learning in the New Information Age (2000) 9-17.

[18] Página da Internet, Internet Content Filtering Group, National Centre for Scientific Research. Disponível em: *http://www.iit.demokritos.gr/skel/i-config/*. Acesso em 09.08.2006.

[19] Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C.D., Stamatopoulos, P., "A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists", Information Retrieval, v. 6, n. 1. (2003) 49-73.

[20] Zhang, L., Zhu, J., Yao, T., "An Evaluation of Statistical Spam Filtering Techniques". ACM Transactions on Asian Language Information Processing 3 (2004) 243-269

[21] Carreras, X., Marquez, L. "Boosting Trees for Clause Splitting", Proceedings of CoNLL-2001, Toulouse, France (2001) 73-75.

[22] Página da Internet, The Apache SpamAssassin Project, The Apache Software Foundation. Disponível em: *http://spamassassin.apache.org/publiccorpus/*. Acesso em 09.08.2006.

[23] Goetschi, R. "Spam-Filtering Using Artificial Neural Networks", Semester Thesis, Computer Science Department, Berne University of Applied Sciences (2004)

[24] Yang, Y., Pedersen, J.O., "A Comparative Study on Feature Selection in Text Categorization", Proceedings of the International Conference on Machine Learning (1997)

[25] Carpinteiro, O. A. S., Lima, I., Assis, J. M. C., Souza, A. C. Z., Moreira, E. M., Pinheiro, C. A. M, "A neural model in anti-spam systems", Lecture Notes in Computer Science, v. 4132, p. 847-855, 2006.

[26] Papoulis, A., Pillai, S.U., "Probability, Random Variables, and Stochastic Processes", 4 edn, McGraw-Hill (2001)

[27] Lee, P.Y., Hui, S.C., Fong, A.M., "Neural Networks for Web Content Filtering", IEEE Intelligent Systems (2002) 48-57.

[28] Chuan, Z., Xianliang, L., Mengshu, H., Xu, Z., "A LVQ-based neural network antispam e-mail approach", ACM SIGOPS Operating Systems Review 39 (2005) 34-39

[29] Drucker, H., Wu, D., Vapnik, V.N., "Support vector machines for spam categorization", IEEE Transactions on Neural networks 10 (1999) 1048-1054

[30] Gomes, L. H., Cazita, C., Almeida, J. M., Almeida, V., Meira Jr., W., "Characterizing a Spam Traffic", Internet Measurement Conference (2004)

[31] Fahlman, S.E., "An empirical study of learning speed in back-propagation networks", Technical Report CMU-CS-88-162, School of Computer Science - Carnegie Mellon University, Pittsburgh, PA (1988)

[32] Página da Internet, IBM Research IBM/EPFL Blue Brain Project, IBM®. Disponível em: *http://domino.research.ibm.com/comm/pr.nsf/pages/rsc.bluegene_cognitive.html.* Acesso em 26.10.2006.

[33] Hidalgo, J.G., "Evaluating cost-sensitive unsolicited bulk e-mail categorization", Proceedings of SAC-02, 17th ACM Symposium on Applied Computing, 615–620 (2002)

[34] Página da Internet, Vara, V., "Rise in Non-English Spam Challenges Junk Mail Filters". Disponível em: *http://www.barracudanetworks.com/ns/news_and_events/news.php?nid=112.* Acesso em 14.01.2007.

[35] Página da Internet, Spamhaus Statistics, Worst Countries, "The Top 10". Disponível em: *http://www.spamhaus.org/statistics/countries.lasso.* Acesso em 14.01.2007.

[36] Nozaki, K., Ishibuchi, H., Tanaka, H., "Adaptive Fuzzy Rule-Based Classification Systems", IEEE Transactions on Fuzzy Systems, Vol.4, No.3, pp. 238-250 (1996)

[37] Alcala, R., Casillas, J., Cordón, O., Herrera, F., Zwir, I., "Techniques for Learning and Tuning Fuzzy Rule-Based Systems for Linguistic Modeling and Their Application", C.T. Leondes (ed.), Knowledge Engineering. Systems, Techniques and Applications, Academic Press (1999)

[38] Golding, A. R., Rosenbloom, P. S., "Improving rule-based systems through case-based reasoning", Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA, pp. 22–27 (1991)

[39] Cordón, O., del Jesus, M.J., Herrera, F., "A proposal on reasoning methods in fuzzy rule-based classification systems," International Journal of Approximate Reasoning, Vol. 20 (1), pp. 21-45 (1999)

[40] Castellano, G., Fanelli, A. M., "A self-organizing neural fuzzy inference network", Proceedings of IEEE Int. Joint Conference on Neural Networks (IJCNN2000), Como, Italy, 5, pp. 14-19 (2000)

[41] Medsker, L. R., "Hybrid Intelligent Systems", Department of Computer Science and Information Systems, The American University, Kluwer Academic Publishers (1995)

APÊNDICES

A Exemplos de Ofuscação dos Spams

Alguns exemplos de técnicas utilizadas pelos *spammers* são apresentadas nesta seção do apêndice.

Tags HTML inválidos

O texto abaixo não contém nenhuma *tag* HTML válida, e é utilizado para enganar os filtros anti-spam:

> < O ministro da Presidência boliviano, Juan Ramón Quintana, fez votos para que a Petrobras compreenda que já não está "no reino das anomalias, que já não está vivendo diante de governos marionetes". O alto funcionário fez esta afirmação faltando só doze horas para vencer o prazo que a companhia petrolífera e outras sete companhias, entre elas a hispano-argentina Repsol YPF, têm para assinar seus novos contratos de operação na Bolívia. >

Uso de imagens com texto

A figura 5.1 mostra um exemplo de um *spam* dentro de uma imagem. Note que o texto exibido é uma imagem:

De: Herman Bean <ojm@tship.com>
Para: [redacted]
Enviadas: Quinta-feira, 26 de Outubro de 2006 10:57:27
Assunto: patient

YOU'VE SEEN IT BEFORE YOU SAY? YOU AIN'T SEEN NOTHING YET!

Trade Date: THURSDAY, OCTOBER 26, 2006
Company: L INTL COMPUTERS INC (Other OTC:LITL.PK)
Symbol: LITL
Wed Close: $1.09
5-day Target: $10
Recommendation: Strong Buy

LITL GETS LISTED ON THE FRANKFURT STOCK EXCHANGE!

MAJOR PR CAMPAIGNS ARE ABOUT TO BEGIN FOR NOVEMBER! THIS THING WILL GO THROUGH THE ROOF! WATCH LITL LIKE A HAWK STARTING THURSDAY, OCTOBER 26!

Figura 5.1: Exemplo de Spam com HTML em branco

Textos invisíveis

Os textos exemplificados abaixo, quando inseridos em um e-mail HTML (casos *a* e *c*), ou no cabeçalho (caso *b*), não serão exibidos ao usuário pelo gerenciador de e-mail. Contudo, um filtro anti-spam comum pode processá-los, fazendo com que o *spam* seja classificado erroneamente como *ham*:

> (a) 2034892423084 2039840348 unanimously repetition 209384093 20938403948 20384034 modem (noun) a device or program that enables a computer to transmit data 94032948
>
> (b) X-Mime-Key: search words: interface external telephone momem cooper modulate cdma gsm signal voice fax second protocol bps bits per second
>
> (c) circumstances lot out chance oh. modern ought wonder cousin.

Note que no caso *c* o texto é exibido. Entretanto, como a fonte utilizada é branca e o fundo também o é, o usuário não será capaz de ver o texto. As figuras 5.2 e 5.3

mostram um *spam* com esta técnica. Na primeira imagem, o texto não está visível, entretanto, na segunda, imagem o texto foi selecionado o texto foi selecionado e pode ser observado.

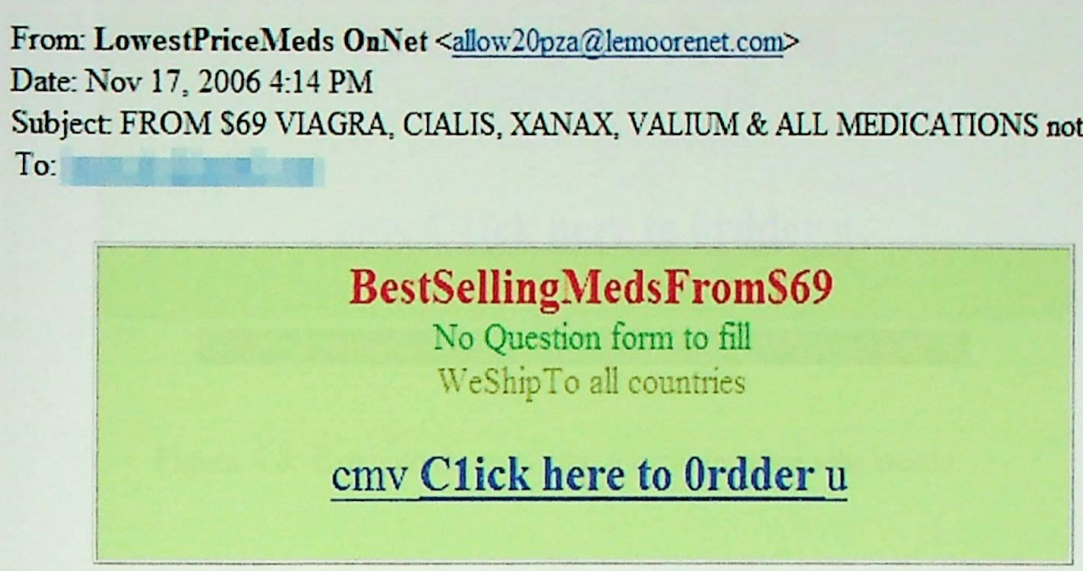

Figura 5.2: Exemplo de *spam* com texto invisível

Comentários HTML

Os dois exemplos abaixo ilustram outra técnica de esconder palavras dos *spammers*, enganando filtros comuns:

(a) <HTML><BODY>**via**<!— xe64 —>**gra**</BODY></HTML>

(b) <HTML><BODY>**GAN****NHE DIN**<!— xe64 —>**HEIRO !**</BODY></HTML>

No caso *a*, a palavra "viagra" é exibida ao usuário, e no caso *b*, "GANHE DINHEIRO !".

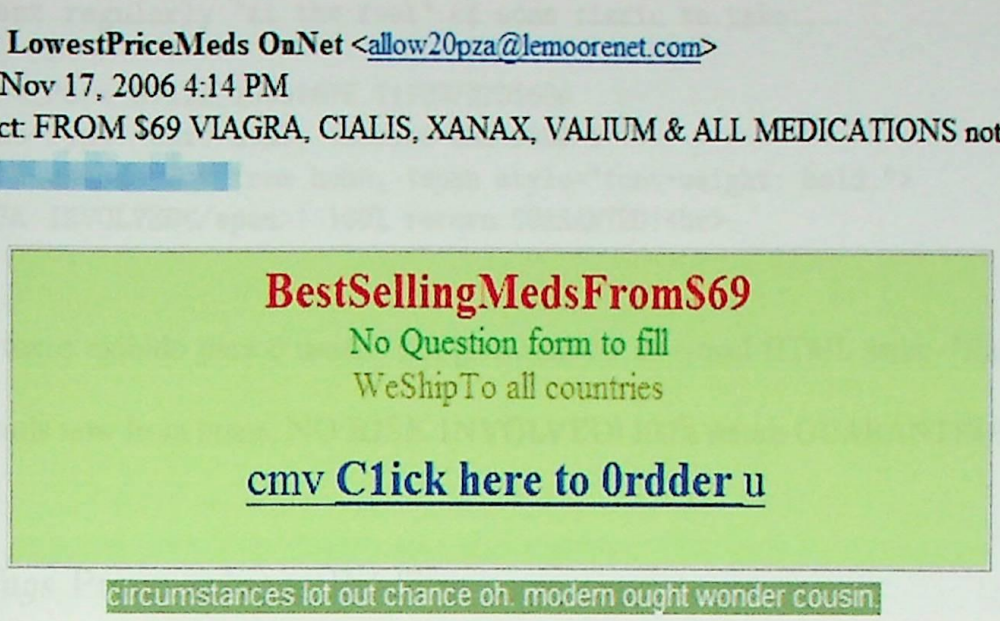

Figura 5.3: Exemplo de *spam* com o texto invisível selecionado

Texto redundante

Com o uso de e-mails MIME, mensagens com formatação (HTML) normalmente são enviados com texto redundante plano idêntico ao HTML, apenas com a formatação removida - por questão de compatibilidade para os gerenciadores de e-mails mais antigos. Os *spammers* utilizam esta característica para enviar um texto "legítimo" na parte de texto, e o *spam* é enviado na parte HTML, conforme o exemplo abaixo:

```
From: -----------------
To: -------------------
Subject: actress pageant
MIME-Version: 1.0
Content-Type: multipart/alternative;
\qquad boundary="----=_Part_89123_15091675.1162072701639"

------=_Part_89123_15091675.1162072701639
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
We are in the stage of weakness and a state of paucity.
Call me crazy, but here's what I find creepy: people
```

```
who meet regularly "at the feet" of some cleric to take...

------=_Part_89123_15091675.1162072701639
Content-Type: text/html; charset=ISO-8859-1
Earn thousands now from home, <span style="font-weight: bold;">
NO RISK INVOLVED</span>! 100% return GUARANTED!<br>
```

O texto exibido para o usuário nos gerenciadores de e-mail HTML seria: "Earn thousands now from home, **NO RISK INVOLVED**! 100% return GUARANTED!".

B *Tags* Processados no Pré-filtro

As *tags* utilizadas no trabalho e suas respectivas categorias são listadas abaixo:

- **Categoria 1** - *tag* totalmente descartado: *comment tag(!—), marquee, style, title*;
- **Categoria 2** - *tag* com atribuitos removidos: *abbr, acronym, area, b, bdo, body, br, caption, cite, code, col, colgroup, dd, del, dfn, div, dl, dt, em, fieldset, frame, frameset, h1, h2, h3, h4, h5, h6, head, hr, html, i, iframe, ins, kbd, label, legend, li, link, meta, noframes, noscript, object, ol, optgroup, option, p, param, pre, q, samp, script, select, small, spanstrong, sub, sup, table, tbody, td, textarea, tfoot, th, thead, tr, tt, u, var*;
- **Categoria 3** - *tag* processado integralmente: *a, base, blockquote, button, font, form, img, input, map.*

C Exemplos de Processamento do Sistema

Exemplo de *ham*

Abaixo segue um *ham* da base de dados SpamAssassin, retirado do arquivo "20030228 easy ham.tar.bz2", cujo número é 485. É um exemplo de e-mail legítimo que usa texto sem formatação. Foram removidos os campos não utilizados do cabeçalho para facilitar o entendimento:

```
From: harley@argote.ch (Robert Harley)
To: fork@spamassassin.taint.org
Subject: Re: Selling Wedded Bliss (was Re: Ouch...)
Message-Id: <20020907135257.7E5CBC44D@argote.ch>
Sender: fork-admin@xent.com
Date: Sat,  7 Sep 2002 15:52:57 +0200 (CEST)

 CDale URLed thusly:
>http://www.news.harvard.edu/gazette/2000/10.19/01_monogamy.html
 >The assumption that females of all species tend to be less
promiscuous than males simply does not fit the facts, Hrdy contended.
 Well, DUH!!!
 It is perfectly obvious that (heterosexual) promiscuity is exactly,
precisely identical between males and females.
 Of course the shapes of the distributions may differ.
 R
```

Após o pré-filtro, foi obtido o seguinte:

```
From: Robert Harley <harley@argote.ch>
To: fork@spamassassin.taint.org
Subject: re: selling wedded bliss (was re: ouch...)
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

cdale urled thusly:
!_LINK
 the assumption that females of all species tend to be less
promiscuous than males simply does not fit the facts,
hrdy contended.
 well, duh
 it is perfectly obvious that (heterosexual) promiscuity is exactly,
precisely identical between males and females.
 of course the shapes of the distributions may differ.
 r
```

Observa-se que houveram algumas modificações na mensagem: todo o texto, inclusive o assunto, foi transformado em caracteres minúsculos; o *link* foi substituído pela seqüência "!_LINK"; os campos "From", "To" e "Subject" do cabeçalho foram mantidos originais, os demais foram introduzidos automaticamente pelo software de

tratamento de e-mails JavaMail do Java - isto não influencia na classificação, já que somente o assunto é usado.

Exemplo de *spam*

O e-mail abaixo é um *spam* da base SpamAssassin retirado do arquivo "20050311 spam 2.tar.bz2", cujo número é 005. Da mesma forma, os campos do cabeçalho não utilizados foram removidos. Este exemplo utiliza formatação HTML:

```
Message-Id: <00003a9c22bc$000065e7$000067d7@ecis.com>
To: <Undisclosed Recipients@netnoteinc.com>
From: tmarain@ecis.com
Subject: Fire The Creep You Call Your Boss!! 26583
Date: Sun, 27 May 2001 12:39:01 -1600
MIME-Version: 1.0
X-Priority: 3
X-Msmail-Priority: Normal
<FONT face=3D"MS Sans Serif">
<FONT size=3D2> <HTML><FONT BACK=3D"#ffffff" style=3D"BACKGROUND-COLOR: #=
ffffff" SIZE=3D2 PTSIZE=3D10><BR><BR>
 FOLLOW ME TO FINANCIAL FREEDOM!!<BR><BR>
<BR><BR>
I Am looking for people with good work ethic and extrordinary desire<BR><B=
R>
 to earn at least $10,000 per month working from home!<BR><BR>
<BR><BR>
NO SPECIAL SKILLS OR EXPERIENCE REQUIRED We will give you all the <BR><BR>
training and personal support you will need to ensure your success!<BR><BR=
>
<BR><BR>
This LEGITIMATE HOME-BASED INCOME OPPORTUNITY can put you back in<BR><BR>
 control of your time,your finances,and your life!<BR><BR>
<BR><BR>
If you've tried other opportunities in the past that have failed to <BR><B=
R>
 live up their promises,<BR><BR>
<BR><BR>
 THIS IS DIFFERENT THEN ANYTHING ELSE YOU'VE SEEN!<BR><BR>
<BR><BR>
 THIS IS NOT A GET RICH QUICK SCHEME!<BR><BR>
```

```
<BR><BR>
YOUR FINANCIAL PAST DOES NOT HAVE TO BE YOUR FINANCIAL FUTURE!<BR><BR>
<BR><BR>
 CALL ONLY IF YOU ARE SERIOUS!<BR><BR>
<BR><BR>
 1-800-345-9708 <BR><BR>
<BR><BR>
 DONT GO TO SLEEP WITHOUT LISTENING TO THIS!<BR><BR>
<BR><BR>
"ALL our dreams can come true- if we have the courage to persue them"<BR><=
BR>
 -Walt Disney<BR><BR>
<BR><BR>
Please Leave Your Name And Number And Best Time To Call<BR><BR>
<BR><BR>
 DO NOT RESPOND BY EMAIL<BR><BR>
<BR><BR>
This is not a SPAM. You are receiving this because<BR><BR>
you are on a list of email addresses that I have bought.<BR><BR>
And you have opted to receive information about<BR><BR>
business opportunities. If you did not opt in to receive<BR><BR>
information on business opportunities then please accept <BR><BR>
our apology. To be REMOVED from this list simply reply <BR><BR>
with REMOVE as the subject. And you will NEVER receive <BR><BR>
another email from me.</FONT></HTML><BR>
<BR>
</FONT></FONT>
```

Após o pré-filtro, foi obtido o seguinte:

```
Message-ID: <795840.1123250184718.JavaMail.jmcassis@gres02>
From: tmarain@ecis.com
Subject: fire the creep you call your boss    !_NUMERO_SUBJECT
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

!_in_font !_in_face !_in_sans !_in_serif"
!_in_font !_in_size  !_in_font !_in_back !_in_style !_in_#ffffff"
!_in_size !_in_ptsize

           follow me to financial freedom

i am looking for people with good work ethic and extrordinary desire
```

```
 to earn at least !_MONEY per month working from home
no special skills or experience required we will give you all the
training and personal support you will need to ensure your success
this legitimate home-based income opportunity can put you back in
                control of your time,your finances,and your life

if you've tried other opportunities in the past that have failed to
                live up their promises,
     this is different then anything else you've seen
       this is not a get rich quick scheme
your financial past does not have to be your financial future
                call only if you are serious
                  1-800-345-9708
    dont go to sleep without listening to this
"all our dreams can come true- if we have the courage to persue them"
                    -walt disney
please leave your name and number and best time to call
                do not respond by email
this is not a spam. you are receiving this because
you are on a list of email addresses that i have bought.
and you have opted to receive information about
business opportunities. if you did not opt in to receive
information on business opportunities then please accept
our apology. to be removed from this list simply reply
with remove as the subject. and you will never receive
another email from me.
```

Algumas modificações podem ser observadas na pré-filtragem:

- Substituição de todos *tags* HTML por palavras específicas - que iniciam com "!_in_";
- Conversão de todos os caracteres em minúsculos;
- Quantidades monetárias foram convertidas em "!_MONEY";
- O número no final do assunto (*subject*), "26583", foi convertido em "!_NUMERO_SUBJECT".

D Exemplos de Vetores Característicos

Exemplo 1

O e-mail abaixo é um *spam* da base SpamAssassin (número 315 do arquivo "20050311 spam 2.tar.bz2") após a pré-filtragem:

```
Message-ID: <31020178.1123250192453.JavaMail.jmcassis@gres02>
From: Debt Relief <DebtRelief@mail5.domainset>
To: yyyy@netnoteinc.com
Subject: too many credit card bills - this fixes that
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

!_in_img !_in_height !_in_src !_in_width
!_in_img !_in_align !_in_height !_in_hspace !_in_src !_in_width
!_in_font !_in_color !_in_face !_in_helvetica !_in_sans-serif"
!_in_size we can help you
...

!_in_font !_in_face !_in_helvetica !_in_sans-serif" !_in_size
!_in_img !_in_height !_in_src !_in_width reduce your
 monthly payment up to 60%.
!_in_img !_in_height !_in_src !_in_width lower your credit
 card interest rates.
!_in_img !_in_height !_in_src !_in_width stop late or over
 the limit fees.
!_in_img !_in_height !_in_src !_in_width combine your
 bills into one low simple payment.
!_in_img !_in_height !_in_src !_in_width provide a
 structured payment plan.
!_in_img !_in_height !_in_src !_in_width bring your account to
 a current status.
!_in_img !_in_height !_in_src !_in_width private handling of
 your accounts.
!_in_img !_in_height !_in_src !_in_width put a debt specialist
on your side. !_in_img !_in_height !_in_src !_in_width
!_in_href !_in_img !_in_border !_in_height !_in_src !_in_width
 !_in_font !_in_color !_in_size
!_in_font !_in_face !_in_helvetica !_in_sans-serif" not a loan,
no credit check, no property needed.
!_in_font !_in_size !_in_color this email is not sent unsolicited.
```

```
you are receiving it because you requested receive this email
by opting-in with our marketing partner. you will receive notices
of exciting offers, products, and other options however, we are
committed to only sending to those people that desire these offers.
if you do not wish to receive such offers !_in_href !_in_font
!_in_size !_in_color !_in_blue click here. or paste the following
into any browser: !_LINK s to remove your email name from our list.
you may contact our company by mail at 1323 s.e. 17th street,
suite number 345, ft. lauderdale, fl 33316
!_in_img !_in_src !_in_serialno
```

A tabela 5.1 mostra os vetores característicos, com indexação binária e normal, obtidos para o e-mail acima, usando o método MI. Cabe observar que o número de ocorrências também pode ser usado como vetor de entrada, conforme mostrado na seção 3.4.3 (método Freqüência do Termo).

n_i	Palavra	nº ocorrências	MI binário	MI normal
1	!_in_font	6	1	0,857
2	!_in_color	4	1	0,571
3	!_in_size	5	1	0,714
4	!_in_face	3	1	0,429
5	you	6	1	0,857
6	your	7	1	1
7	!_in_sans-serif	1	1	0,143
8	!_MONEY	0	0	0
9	!_LINK	1	1	0,143
10	click	1	1	0,143
11	the	2	1	0,286
12	money	0	0	0
13	free	0	0	0
14	!_in_blockquote	0	0	0
15	business	0	0	0
16	wrote	0	0	0
17	our	3	1	0,429
18	!_in_helvetica	3	1	0,429
19	!_BIGTEXT	0	0	0
20	receive	3	1	0,429
21	!_in_arial	0	0	0
22	email	3	1	0,429
23	please	0	0	0
25	but	0	0	0
25	order	0	0	0

Tabela 5.1: Exemplos de Vetores Característicos