UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Filtragem de Informações no Ambiente do Direto

por

RICARDO BALINSKI

Trabalho de Conclusão submetido à avaliação,
como requisito parcial para a obtenção
do grau de Mestre em Informática

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, maio de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Balinski, Ricardo

Filtragem de Informações no Ambiente do Direto / por Ricardo Balinski. – Porto Alegre: PPGC da UFRGS, 2002.
87 f.: il.

Trabalho de conclusão (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 2002. Orientador: Geyer, Cláudio Fernando Resin.

1. Sistemas de recuperação de informações. 2. Classificação de textos. 3. Filtragem de informação. 4. Filtro de mensagens. 5. Sieve. 6. Direto. 7. Canais de informação. 8. Modelo do espaço vetorial. I. Geyer, Cláudio Fernando Resin. II. Título.

Sumário

Lista de Abreviaturas

API	Application Program Interface
CART	Classification and Regression Trees
CSS	Cascading Style Sheets
DHTML	Dynamic HTML
DOM	Document Object Model
FTP	File Transfer Protocol
GNU	GNU's Not Unix
HTML	Hypertext Markup Language
HTTP	Hipertext Transfer Protocol
ICAP	Internet Content Adaptation Protocol
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol
IPCE	InterProcess Communication Environment
JDBC	Java Database Connectivity
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LSI	Latent Semantic Index
MIME	Multipurpose Internet Mail Extensions
MS	Microsoft
POP	Post Office Protocol
PROCERGS	Companhia de Processamento de Dados do Estado do Rio Grande do Sul
RFC	Request for Comments
RTF	Rich Text Format
SAX	Simple API for XML
SGDB	Sistema Gerenciador de Banco de Dados
SGML	Standard Generalized Markup Language
SMART	System for the Mechanical Analysis and Retrieval of Text
SMS	Short Messages Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SRI	Sistemas de Recuperação de Informações
SSL	Secure Sockets Layer
TFC	Term Frequency Component
TFIDF	Term Frequency and Inverse Document Frequency
UFRGS	Universidade Federal do Rio Grande do Sul
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

Lista de Figuras

Lista de Tabelas

Resumo

Os Sistemas de Recuperação de Informações (SRI) computadorizados são sistemas capazes de armazenar, recuperar e manter informações, visando minimizar o esforço humano na realização de tais atividades. A classificação de textos é um subdomínio dos sistemas de recuperação de informações que tem como objetivo classificar um texto em uma ou mais categorias existentes. Pode ser utilizada na classificação de mensagens, notícias e documentos, na filtragem de informações, na sumarização de textos, além de auxiliar profissionais na execução destas tarefas.

A filtragem automatizada das mensagens de correio eletrônico é uma forma de organizar o trabalho do usuário. O volume de informações divulgadas através deste serviço torna fundamental um sistema de filtros para melhor uso do serviço. Sieve é uma proposta para padrão de linguagens de filtro de mensagens.

O Direto é um software de correio, agenda e catálogo corporativos que visa atender todo Governo do Estado do Rio Grande do Sul. Foi desenvolvido na PROCERGS, Companhia de Processamento de Dados do Estado do Rio Grande do Sul, utilizando a linguagem Java e utiliza os serviços de IMAP, SMTP, LDAP e SGBD. Está disponível com licença de software livre.

O objetivo deste trabalho é aplicar técnicas de filtragem no Direto. O trabalho apresenta uma solução para filtrar as mensagens de correio do Direto utilizando Sieve. Também é especificado um serviço de canais de informação que visa divulgar informações de forma eficiente no Estado. Este serviço possui vários canais, cada um destinado a divulgar informações de determinado domínio. O usuário assina os canais que desejar e pode criar filtros para melhor refinamento das informações que deseja receber. Os filtros utilizam técnicas de classificação de textos no processo de filtragem.

Palavras-chave: sistemas de recuperação de informações, classificação de textos, filtragem de informação, filtro de mensagens, sieve, direto, canais de informação, modelo do espaço vetorial.

TITLE: "INFORMATION FILTERING IN THE DIRETO ENVIRONMENT"

Abstract

A Computerized Information Retrieval Systems (IRS) can store, retrieve and maintain information. Its purpose is to minimize the human effort needed in such activities. Text categorization is a subdomain of IRS research that consists of identifying text as an instance of one or more pre-determined categories. Text categorization can be used to divide messages, news and other documents in categories, to filter information, and to make text summarization. These tasks can be done automatically or as a support to professionals.

The e-mail service releases a high amount of information and it is important the creation of a filtering system to improve the use of this service. Automatic e-mail filtering can aid users to organize its messages and thus to be more productive. Sieve is proposed to be a standard mail filtering language.

Direto is a corporative e-mail, calendar and directory system whose target is to serve all the Rio Grande do Sul State. It was developed at PROCERGS using the Java language. It uses IMAP, SMTP, LDAP and DBMS services. It is available by the GPL free software license.

This work aims to apply filtering techniques to Direto. This work presents a solution using Sieve language. This work also specifies an information channel service which releases information in an effective way. This service is composed of information channels, each one releasing information of a specific area. The user select a channel with information he wants to receive and subscribes to it. The filters use text categorization techniques in the filtering process.

Keywords: information retrieval systems, text categorization, information filtering, message filtering, sieve, direto, information channels, vector space model.

1 Introdução

Com a popularização da Internet e de outras tecnologias foi criado um ambiente onde muitas informações trafegam e estão disponíveis a todos os usuários. O serviço de correio eletrônico está amplamente difundido entre os usuários da Internet, sendo o serviço que possui maior utilização. É uma forma de comunicação rápida que facilita a distribuição de informações.

Juntamente com a popularização e o crescimento do uso do serviço de correio eletrônico, também tem aumentado as diversas finalidades para as quais o serviço é utilizado. O correio hoje é utilizado para mensagens pessoais, comerciais, listas de discussão, correntes e propagandas. Esta grande quantidade de informações têm inundado o usuário de informações. Cada vez tem se gasto mais e mais tempo filtrando informações. A quantidade de dados irrelevantes e às vezes indesejados tem aumentado a necessidade por sistemas inteligentes que automatizem a seleção das informações recebidas. Um serviço de filtro de mensagens de correio eletrônico auxilia no aumento de produtividade para o usuário através da organização de mensagens recebidas em pastas, diferentes prioridades e também com a rejeição de mensagens indesejadas.

O Direto é uma ferramenta de *groupware* desenvolvida pela PROCERGS, que inclui os serviços de correio eletrônico, agenda e catálogo corporativos. O Direto se destina a integrar todos os órgãos do governo estadual. Outras organizações também podem se beneficiar do produto, pois o mesmo está licenciado como software livre.

O governo estadual, devido a sua natureza, é uma grande organização distribuída que gera muitas informações. A divulgação destas informações internamente é um processo complexo, pois existe dificuldade em localizar o público alvo para determinada informação, e também a divulgação da informação de forma generalizada, para todos os usuários ocasionaria um significativo aumento das informações recebidas por cada usuário, o que diminuiria sensivelmente a sua produtividade.

Este trabalho propõe a aplicação de técnicas de filtragem para o sistema Direto. É proposto um sistema para filtro de mensagens de correio eletrônico visando aumentar a produtividade do serviço de correio eletrônico. Também é proposto um serviço de canais de informação. Este serviço tem como finalidade a divulgação de informação através de canais para o usuário, e está baseado nas técnicas dos sistemas de recuperação de informações.

No capítulo 2 é apresentado o Direto. O capítulo inicia com o histórico do correio eletrônico no Estado do Rio Grande do Sul. Depois discorre sobre o projeto do Direto, incluindo seus requisitos iniciais, o framework utilizado e a arquitetura utilizada. Também são descritos todas as tecnologias e serviços utilizados. A licença do produto também é apresentada, pois o fato de ser software livre possui elevada importância para o desenvolvimento do projeto.

No capítulo 3 são apresentados os Sistemas de Recuperação de Informações baseados principalmente no Modelo do Espaço Vetorial. São apresentadas as etapas para classificação de texto, pré-processamento, indexação, e redução da dimensão, as quais realizam a sumarização de um documento e mais a etapa da classificação de fato do documento em alguma categoria. São apresentados métodos para cada etapa do

processo. Também são abordados o *clustering* de um conjunto de documentos e as métricas para avaliação de desempenho de um Sistema de Recuperação de Informações.

No capítulo 4 é apresentado Sieve, uma linguagem para filtragem de mensagens de correio eletrônico que se propõe como padrão Internet. São apresentados os requisitos, a definição, os comandos, os tipo de dados, as estruturas de controle, os tipos de testes da linguagem. O capítulo termina com um exemplo de um script Sieve.

No capítulo 5 é apresentada uma proposta para um sistema de filtros de mensagens de correio eletrônico para o Direto utilizando Sieve. É explicado o motivo da opção pelo uso de Sieve. São apresentados os teste, as regras, e as ações possíveis sobre as mensagens. O capítulo finaliza com uma comparação dos filtros propostos com outro sistema de regras para mensagens e com a comparação de filtragem no servidor versus filtragem no cliente.

No capítulo 6 é apresentado o serviço de canais de informação para o Direto. É feita uma introdução sobre a necessidade deste serviço, depois o serviço é descrito em detalhes. São detalhados todos os algoritmos, estruturas de dados, e possíveis ações utilizadas no serviço.

2 Direto

Este capítulo apresenta o software de correio, agenda e catálogo corporativo Direto. Um breve histórico de correio eletrônico no Estado é apresentado. O histórico é importante, pois leva ao momento da criação do Direto. A cultura de correio já existente, juntamente com as tecnologias disponíveis influenciou a definição da solução, auxiliando e influenciando na definição dos requisitos.

Os serviços que fornecem a infra-estrutura para o Direto, IMAP, SMTP, LDAP e SGBD, juntamente com as tecnologias utilizadas no desenvolvimento do software, Java, XML, XSL, entre outras são descritos. A partir da infra-estrutura e das tecnologias disponíveis foi montada a arquitetura do software. A arquitetura possui componentes de controle, apresentação, negócio e dados.

Dificuldades financeiras para implantar uma solução para 200 mil usuários direcionou para o uso de software livre. A experiência da PROCERGS utilizando software livre, incluindo o sistema operacional Linux [LIN 2001] e o servidor Web Apache [APA 2001], com sucesso influenciou na decisão. Além de ser utilizado no Estado, o Direto está disponível com licença de software livre para outras organizações interessadas utilizarem o produto sem custo de licenciamento.

2.1 Histórico de correio eletrônico no Estado

A PROCERGS atua desde 28 de dezembro de 1972 como órgão executor da política de informática do Estado. Desde 1991 a PROCERGS vem trabalhando para suprir o Estado com uma ferramenta adequada para comunicação entre os órgãos. Nesta época, especificamente no segundo semestre de 1991, foi realizado um projeto de avaliação das ferramentas de automação de escritório existentes no mercado, considerando os seguintes aspectos: rodar no ambiente mainframe para aproveitar o parque instalado, sem maior investimento em infra-estrutura, e o produto deveria conter os serviços de mensagens, agenda e quadro de avisos. Foi escolhido o Memo da Eurosoft. O Memo foi implantado em janeiro de 1992 e, no momento do desenvolvimento deste trabalho, está instalado em 68 clientes da PROCERGS contando com mais de 13.000 usuários.

Diante das novas tendências tecnológicas no mercado da informática, no segundo semestre de 1996, a PROCERGS implantou o Correio Eletrônico do Lotus Notes.

Em 1999 o Estado contava com duas soluções de Correio Eletrônico, o Memo e o Lotus Notes. Existia a necessidade de uma ferramenta padrão de comunicação para o Estado, que atingisse inicialmente 50.000 usuários (atualmente a previsão é de que 200.000 servidores do Estado utilizem a solução). Os pontos críticos que motivaram a demanda por uma nova solução foram:

- Falta de uma ferramenta padrão de Correio Eletrônico para o Governo do Estado que facilitasse a comunicação interna e externa;
- Convivência de duas ferramentas de Correio Eletrônico adotadas pelo Estado (Memo e Notes), uma com interface caractere e outra com interface gráfica;

- Ferramentas de Correio Eletrônico existentes estavam sub-implementadas, pois alguns recursos do Notes como replicação, trabalho off-line e acesso remoto não foram implementados por problemas técnicos de segurança. No Memo, alguns recursos também não foram implementados;
- Falta de conhecimento dos usuários das ferramentas utilizadas;
- Correio Eletrônico Memo é visto como ferramenta com obsolescência tecnológica pelo tipo de interface;
- Montagem de uma infra-estrutura de hardware e software para o Estado depende da disponibilidade de recursos financeiros elevados;
- Familiaridade dos usuários do Estado com correio Internet e ferramentas de Correio Eletrônico como Exchange, Outlook, e Eudora;
- Falta de integração entre as agendas do Memo e do Notes.

A PROCERGS, após avaliar alternativas, incluindo ferramentas de *groupware* do mercado e a alternativa de desenvolver sua própria solução, decidiu por desenvolver sua própria solução.

2.2 Projeto Direto

O Direto [DIR 2000] é uma ferramenta de *groupware*, que inclui os serviços de correio eletrônico, agenda e catálogo corporativos. O Direto se destina a integrar todos órgãos do governo estadual. Também se baseia em software livre, o que não somente diminuiu o custo do projeto, mas viabilizou o projeto financeiramente. O fato de possuir licença de software livre possibilita a outras organizações interessadas utilizarem o produto sem nenhum custo.

O Direto possui interface Web para ser acessado a partir de um *browser*. A interface Web possui várias vantagens em relação a um cliente específico instalado na máquina do usuário. O sistema pode ser acessado de qualquer ponto da Web sem necessidade de instalação de um cliente ou de determinada configuração específica. O cliente é independente de sistema operacional, basta um *browser* Web. Os usuários já possuem experiência com este tipo de interface. E a criptografia dos dados pode ser garantida pelo protocolo SSL, suportado pelos principais *browsers*. O único inconveniente é que nesta solução todo processamento fica centralizado no servidor e o usuário precisa estar on-line para acessar seus dados.

A solução teve como principais requisitos tecnológicos os seguintes itens:

- Seguir protocolos padrão Internet.
- Independência de plataforma no cliente e no servidor.
- Prover independência de Gerenciador de Banco de Dados.
- Prover independência de Servidor Web.
- Ser modular, ou seja, permitir que novas funcionalidades pudessem ser adicionadas no futuro.
- Possuir custo baixo.

A solução, materializada através do produto Direto, baseou-se totalmente em protocolos padrão Internet. O Direto está estruturado de forma modular utilizando os protocolos vCalendar para agenda, SMTP e IMAP para correio e LDAP para catálogo.

2.3 Framework do Direto

O Direto é uma solução que utiliza vários serviços integrados. Os serviços são implementados por produtos de software livre, e podem ser substituídos sem que isso afete a solução, pois todos produtos implementam protocolos padrão Internet.

Como a solução utiliza protocolos abertos, o acesso aos dados pode também ser disponibilizado por clientes de fornecedores que implementem esses protocolos.

Outra consideração importante é a portabilidade da solução proposta. Qualquer uma das peças envolvidas pode ser substituída sem alterar a estrutura desenvolvida. Abaixo são descritos os serviços utilizados no Direto.

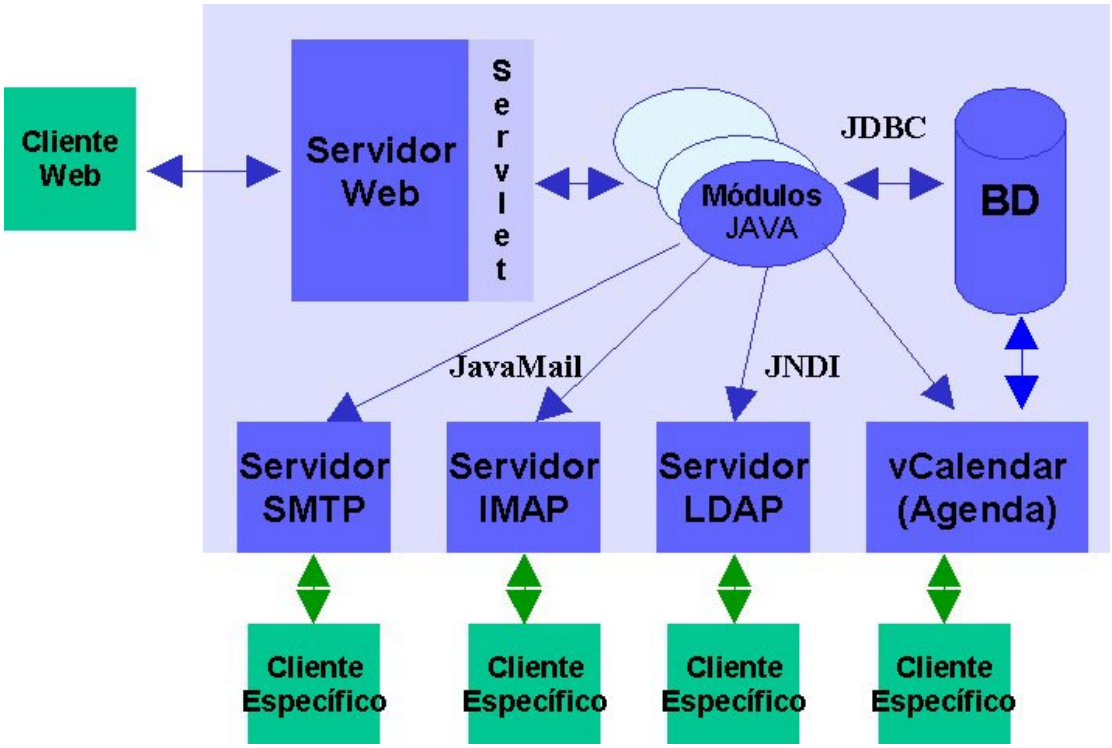

FIGURA 2.1 – Framework do Direto.

2.3.1 Serviço de entrega de mensagens - SMTP (RFC 821)

O protocolo SMTP (*Simple Mail Transfer Protocol*) está definido na RFC 821 [POS 82]. O objetivo deste protocolo é o de transferir mensagens de forma confiável e

eficiente. O SMTP é independente de um subsistema particular de transmissão e requer somente um canal de dados confiável. Uma das principais características do SMTP é a sua capacidade de transmitir mensagem entre ambientes diversos de serviços de transporte. Um serviço de transporte provê um ambiente de comunicação interprocessos (IPCE). Um IPCE pode cobrir uma rede, várias redes ou um subconjunto de redes. É importante perceber que os sistemas de transportes (ou IPCEs) não são uma relação de um-para-um com as redes. Um processo pode se comunicar diretamente com qualquer outro através de qualquer IPCE conhecido mutuamente. E-mail é uma aplicação de comunicação interprocesso. Mais especificamente, um e-mail pode ser transmitido entre hosts com diferentes sistemas de transportes por um host com ambos sistemas de transportes.

O protocolo SMTP é, realmente, um protocolo simples e eficiente e os protocolos "abertos" de acesso a mensagens como IMAP e POP (abordados na seção seguinte) não duplicam sua função. Na verdade, trabalham em conjunto. O protocolo SMTP pela sua eficiência e simplicidade se torna uma opção extremamente vantajosa no que diz respeito a serviço de roteamento e entrega de mensagens. O suporte a esse protocolo já é realidade em vários produtos de mercado, como Notes e Exchange, principalmente em função da compatibilidade com o mundo Internet.

O projeto do SMTP é baseado no seguinte modelo de comunicação: como resultado de uma requisição de e-mail de um usuário qualquer, o SMTP-origem estabelece um canal de transmissão de "duas-mãos" com um SMTP-destino. O SMTP-destino pode ser tanto o servidor final do mail como um intermediário. Comandos de requisição SMTP são enviados do SMTP-origem e o SMTP-destino envia as respostas. Uma vez que o canal é estabelecido, o SMTP-origem envia um comando MAIL, indicando o remetente da mensagem. Se o SMTP-destino pode aceitar a mensagem, responde com OK. O SMTP-origem envia, então, um comando RCPT identificando um destinatário para a mensagem. Se o SMTP-destino pode aceitar uma mensagem para aquele destinatário, responde com OK; caso contrário, responde com uma notificação de rejeição daquele destinatário (mas não encerra a transação de e-mail). Um SMTP-origem pode negociar vários destinatários. Quando o destinatário for negociado, o SMTP-origem envia os dados da mensagem, terminando com uma seqüência especial. Se o SMTP-destino processa a mensagem corretamente, envia um OK. O diálogo SMTP é propositadamente lock-step, um de cada vez.

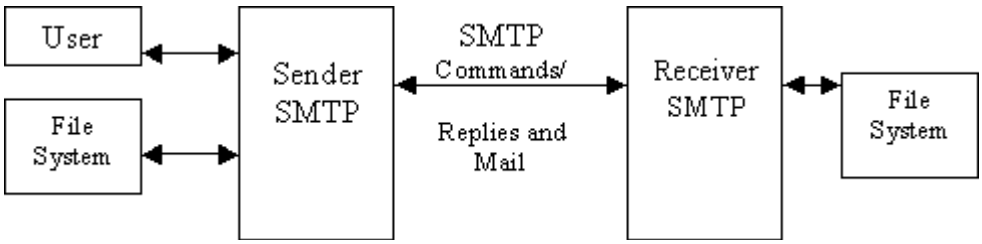

FIGURA 2.2 - Modelo de uso de SMTP.

2.3.2 Serviço de acesso a mensagens

Como não seria conveniente enviar as mensagens diretamente ao computador do destinatário, uma vez que isso pressupõe, no mínimo, que o computador esteja sempre disponível, o processo de armazenamento de mensagens é feito em um servidor. E, se as mensagens precisam chegar ao computador do destinatário em um dado momento, torna-se necessário a existência de um protocolo de acesso a essas mensagens. O Direto suporta tanto POP3 como IMAP4.

2.3.2.1 POP3 (RFC 1725)

Dos três protocolos de acesso a mensagens da Internet, o POP (*Post Office Protocol*) é o mais antigo, sua primeira RFC data de outubro de 1984. A revisão atual desse protocolo é a POP3 (especificada na RFC 1725) [MYE 94]. O protocolo POP foi projetado especificamente para o acesso off-line de mensagens, ou seja, o servidor de mensagens é apenas um depositário temporário das mensagens, devendo a manipulação das mensagens e pastas ficar sob responsabilidade do cliente (cliente POP).

2.3.2.2 IMAP4 (RFC 1730)

O protocolo IMAP (*Internet Message Access Protocol*) – versão 4 [CRI 96] permite que um cliente acesse e manipule mensagens em um servidor. Permite a manipulação de pastas de mensagens ("mailboxes"), de uma maneira funcionalmente equivalente a manipulação de pastas locais. IMAP4 também provê a capacidade de um possível cliente off-line ressincronizar suas mensagens com o servidor. O protocolo prevê funções como: criação, exclusão e renomeação de pastas; checagem de novas mensagens; remoção permanente de mensagens; pesquisa de atributos ou texto de mensagens.

2.3.2.3 Comparação POP3 x IMAP4

O protocolo IMAP é um superset das capacidades do protocolo POP. O IMAP suporta totalmente o processo off-line tão bem quanto o protocolo POP, e ainda, suporta o processo de trabalho desconectado (ressincronização de conteúdo cliente/servidor). A seguir, são enumeradas características similares e diferenças entre os protocolos [GRA 95].

2.3.2.3.1 Características similares

- Somente manipulam acesso a mensagens, com relação ao serviço de enviar mensagens, ambos delegam ao SMTP;
- Consideram um servidor sempre disponível e compartilhado;
- Permitem acesso a uma mensagem de diversas plataformas de cliente;
- Permitem acesso a uma mensagem de qualquer ponto da rede;
- Suporte total ao trabalho no formato off-line;
- Várias implementações disponíveis livremente na Internet;

- Várias implementações de clientes em diversas plataformas;
- Disponibilidade de implementações comerciais;
- São protocolos abertos, definidos por RFCs Internet;
- São protocolos nativos Internet, não é necessário "gateway" para Internet;

2.3.2.3.2 Vantagens do protocolo IMAP sobre POP3

- Manipulação de pastas remotas;
- Suporte a múltiplas pastas;
- Otimização da performance no processo on-line de acesso;
- Pesquisa e seleção realizadas pelo servidor minimizando a transferência de dados pela rede;
- Visão única das mensagens pelo usuário, não importando o cliente ou a máquina de acesso;
- Disponibilização de diversos *flags* de mensagem;

2.3.3 Serviço de diretório

A localização de pessoas e recursos é um fator de muita importância em um ambiente de comunicação distribuída. Para implementar esse serviço se torna necessário um servidor de diretórios com informações sobre usuários e recursos organizados de maneira hierárquica.

A seguir, é apresentada uma breve descrição do protocolo LDAP que tem se tornado um padrão de fato e de direito no mundo Internet. Os principais fornecedores de ferramentas de correio, como Netscape e Exchange, já suportam esse protocolo. O LDAP surgiu como uma implementação simplificada do protocolo X.500.

2.3.3.1 LDAP (Lightweight directory access protocol)

Inicialmente desenvolvido na Universidade de Michigan, LDAP [YEO 95] é hoje um padrão para serviços de diretório sobre TCP/IP. Um ou mais servidores LDAP constituem uma árvore de diretório. Um cliente LDAP ao submeter uma pesquisa a um servidor LDAP, recebe uma resposta a sua requisição ou um caminho a um outro servidor LDAP que pode possuir as respostas ao cliente.

Um servidor LDAP é um servidor especializado para diretórios. A diferença de um servidor de diretório para um banco de dados genérico pode ser evidenciada pelo padrão de uso. Um diretório possui informações que são extensivamente pesquisadas, mas muito pouco atualizadas. Servidores LDAP são projetados para esse perfil de uso, muitas consultas e poucas alterações, enquanto um banco de dados relacional preocupa-se com a manutenção de forma geral dos dados.

LDAP usa o conceito de hierarquia orientada a objetos de entradas, onde os dados são representados como objetos e estão organizados de forma hierárquica em uma estrutura de árvore. Outros serviços Internet também utilizam o conceito de hierarquia, como é o caso DNS (*Domain Name Service*) para a resolução de nomes para endereços IP.

LDAP é mais do que um protocolo, existem quatro conceitos importantes relacionados: o padrão aberto LDAP, a API, o formato de texto LDIF e as definições de classe de objetos.

- **Padrão aberto LDAP:** As RFCs 1777 e 1778 definem o protocolo que permite a clientes de diferentes fornecedores, em qualquer plataforma, acessar dados de um servidor LDAP. Muitos fornecedores anunciaram suporte ao protocolo LDAP. A Universidade de Michigan tem o código fonte original (slapd), assim como outras ferramentas disponíveis para download.

- **API:** Um dos fatores que contribuíram para o sucesso do LDAP é a capacidade dos desenvolvedores acessarem as informações de um servidor LDAP sem ter que escrever e depurar muito código. Uma API (*Application Programming Interface*) bem projetada permite esse acesso.

- **LDIF:** Outro pedaço importante da arquitetura LDAP é o formato LDIF. Esse formato é usado para exportar e importar dados de/para servidores LDAP. Esse formato permite que se escreva scripts que convertam os dados de uma base qualquer para o formato LDAP de maneira fácil.

- **Classes de Objetos:** Uma entrada no LDAP contém mais do que os atributos que formam um nome distinto de um objeto. Uma entrada também contém uma indicação da classe de objeto (ou subclasse) a qual o objeto pertence. Todos os objetos na mesma classe têm uma ou mais características similares. Classes e objetos podem ser arbitrariamente subdivididos em subclasses e assim por diante, indefinidamente. Usar classes e objetos simplifica o processo de dar nome e organizar objetos. Existem regras para formar os nomes dos objetos numa classe ou subclasse. Uma subclasse herda regras da classe superior imediatamente acima, que a contém. As regras especificam que atributos são obrigatórios ou opcionais na entrada de uma classe de objetos. As regras também especificam que classes e objetos podem estar subordinados a uma dada entrada. Por exemplo, pode-se ter uma classe objeto "país" que seria superior aos objetos da classe "estado" ou da classe "organização" no nível imediatamente inferior da árvore. Este segundo tipo de regra é chamado de regra estrutural. Regras estruturais também especificam que atributos associados com uma dada classe devem aparecer no nome distinto relativo da entrada de nível imediatamente inferior. Regras estruturais são armazenadas em entradas junto com a classe e o nome do objeto.

2.3.4 Serviço de Agenda

O uso de sistemas de agendamento e calendário tem crescido significantemente. Aplicações intra-organização e inter-organizações demandam um rápido processo de agendamento e calendário, com particular ênfase nos negócios do Estado, onde a agilidade de agendamento e calendário pode acrescentar uma proporcional agilidade ao

Estado. No entanto, no que tange a padrões abertos, o crescimento dessa demanda não foi acompanhado de perto pelos padrões Internet. Atualmente os sistemas de agendamento são proprietários e não interoperáveis. No entanto, existem hoje vários esforços da comunidade Internet em padronizar esse processo. Os seguintes padrões e protocolos estão sendo definidos: "*Calendaring Data Interchange Standard*", "*Calendaring Interoperability Protocol*" e "*Calendar Access Protocol*". Nas seções seguintes, são brevemente abordados esses trabalhos.

2.3.4.1 vCalendar

O primeiro padrão de fato a surgir foi o vCalendar [VCA 2001] da Versit (consórcio de empresas). Esse padrão define o formato da informação de agendamento e calendário, como, por exemplo, assunto de uma reunião, lista de convidados e data. Esse formato de dados permitirá, por exemplo, a troca de dados de um sistema de agendamento para outro na forma de um arquivo no formato vCalendar.

2.3.4.2 iCalendar (RFC 2445)

O formato iCalendar [DAW 98] é um esforço do IETF e procura revisar o padrão vCalendar para torná-lo um padrão Internet de direito. O "*Internet Calendaring and Scheduling Core Object Specification*" ou iCalendar permite a captura e troca de informações entre aplicações de agendamento. O formato é definido como um tipo MIME, isso permitirá que um objeto seja trocado entre sistemas por diversos transportes, incluindo SMTP, HTTP, um sistema de arquivos, protocolos de iteração desktop como drag-and-drop, entre muitos outros.

2.3.4.3 ICAP

Esse protocolo permite que diferentes clientes e diferentes servidores de vários fornecedores possam operar, analogamente aos padrões de acesso a mensagem como POP3 e IMAP4 para comunicação cliente-servidor de mensagens. O IETF está trabalhando nesse protocolo e existe uma proposta de padronização para o ICAP.

2.3.5 Java

A linguagem Java [SUN 95], criada pelo grupo liderado por James Gosling na Sun Microsystems, é uma linguagem computacional completa, independente de plataforma e com uma série de facilidades para a integração com a Internet.

Fatores que motivaram o uso da linguagem Java incluem:

- Multi-plataforma: O compilador Java compila o código Java em "*bytecodes*". Estes *bytecodes* são então interpretados por uma "Máquina Virtual" Java que é escrita para a arquitetura de processador em que o programa irá rodar.
- Suporte aos protocolos de rede: Suporte nativo para HTTP, FTP, MIME, e Sockets

- Orientada a objetos: facilita a reutilização de código.
- Grande variedade de APIs: Diminuem o tempo de desenvolvimento
- Facilidade de distribuir processamento e trabalhar com objetos distribuídos

2.4 Arquitetura do Direto

O Direto está dividido em componentes de acordo com as suas responsabilidades. Isto facilita a alteração e atualização de determinado componente e também sua distribuição. A figura 2.3 apresenta a arquitetura do Direto.

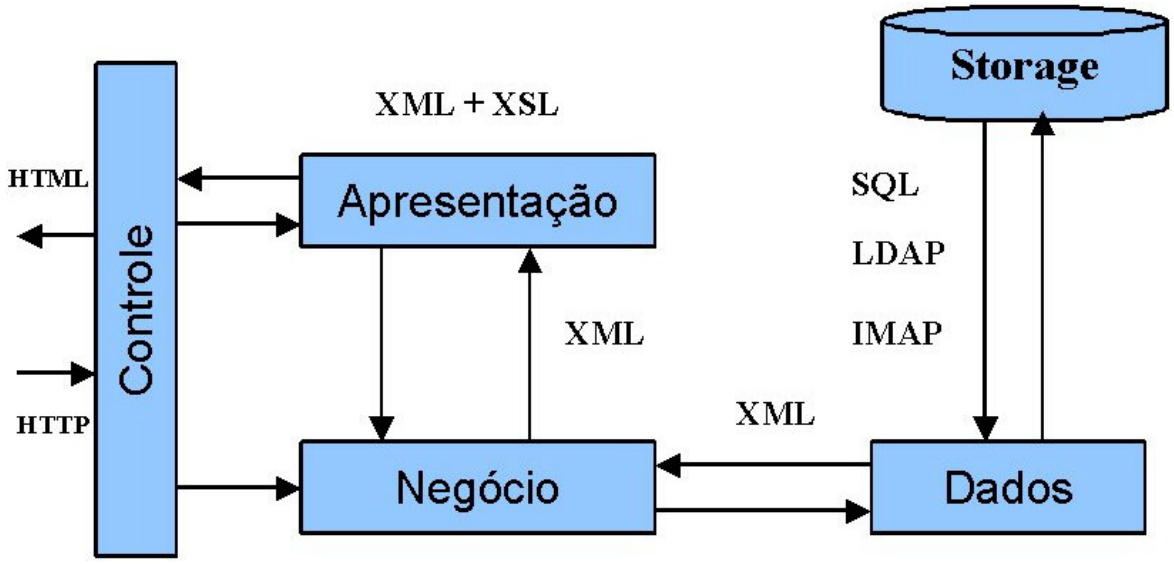

FIGURA 2.3 – Arquitetura do Direto.

2.4.1 Controle

O componente de controle é quem recebe as requisições dos usuários e possui como responsabilidades autenticação de usuários, controle de sessão e *cache* de sessões de usuários.

O componente de controle é implementado por um *servlet* que recebe todas requisições *http*, autentica usuários, controla a sessão e mantém em *cache* os objetos do usuário.

A autenticação de usuários pode ser feita contra uma base de dados LDAP, IMAP, POP ou banco de dados relacional. Isto depende de um parâmetro em um arquivo de configuração. A instalação padrão do Direto utiliza o LDAP para armazenar as informações do usuário e, por conseqüência, autentica os usuários contra a base de dados LDAP.

Após a autenticação existe a necessidade de controle da sessão do usuário. O controle de sessão serve para identificar que uma determinada requisição pertence a determinado

usuário autenticado no sistema. O controle de sessão do Direto utiliza a classe *Session* do pacote *servlet* [SUN 2001d] da linguagem Java para a implementação. Esta classe, por sua vez, utiliza *cookies* [KRI 97] para implementar o controle de sessão. O Direto também define o período que um usuário pode permanecer sem interagir com o sistema até que sua sessão seja expirada.

A *cache* de objetos do usuário mantém em memória todos objetos utilizados pelos usuários com sessões ativas. Cada entrada da *cache* está relacionada com uma sessão ativa de usuário. Os objetos dos usuários precisam ficar em *cache* para que exista uma referência a eles, e eles fiquem acessíveis a partir de outras requisições de usuários e não sejam eliminados pelo *garbage colector*. A importância de não se perder a referência a um objeto é que se perder a referência, o objeto fica inacessível, é destruído pelo *garbage colector* e é necessário criar uma nova instância do mesmo, e o estado do anterior é perdido.

2.4.2 Negócio

Os componentes de negócio são constituídos de classes Java que se comunicam com outros serviços para realizar determinada tarefa. A comunicação com os serviços é feita utilizando-se APIs da linguagem Java. As principais APIs utilizadas são:

- **JavaMail** [SUN 2001a]: é utilizada para a comunicação com os serviços de correio, acesso a caixa postal e serviço de transporte. O serviço de transporte utilizado é o SMTP, e a API é utilizada no envio de mensagens de correio eletrônico. No acesso a caixa postal, a API é utilizada para verificar mensagens novas, listar o cabeçalho de mensagens de determinada pasta, ler mensagens e excluir mensagens. Para o acesso à caixa postal é necessário adicionar uma outra API, um *provider* específico para cada protocolo de acesso (POP3 e IMAP4).

- **JNDI** [SUN 2001b]: é utilizada para o acesso ao LDAP. A API JNDI é utilizada para consultas, inclusões e exclusões sobre a base LDAP do Direto.

- **JDBC** [SUN 2001c]: é utilizada para o acesso ao banco de dados. Trabalha em conjunto com o driver do banco de dados. É utilizada principalmente no módulo de agenda em consultas, inclusões e exclusões de registros.

Os componentes de negócio possuem complexidades distintas dependendo de cada módulo do sistema. O módulo de agenda é o mais complexo, visto que, não existe um servidor de compromissos de agenda. A agenda do Direto mapeou o padrão vCalendar para uma base de dados relacional para simular o servidor de agenda. Diversos métodos complexos, como, por exemplo, verificar a ocorrência de um compromisso repetitivo em determinado dia, tiveram que ser implementados na lógica de negócios por não existir um servidor de compromissos.

O Direto segue protocolos abertos e possibilita que cada um de seus serviços seja acessado por clientes específicos além da própria interface do produto. Os serviços de correio e catálogo podem facilmente ser acessados e utilizados por outros clientes de outros fornecedores. Para garantir a mesma facilidade para o serviço de agenda, o

Direto implementa a funcionalidade de exportar e importar a agenda em formato vCalendar. Isto permite ao usuário exportar sua agenda, importar em algum outro cliente, como, por exemplo, uma agenda em um palm, ou qualquer outro software de agenda que importe arquivo no formato vCalendar, consultar e alterar os compromissos neste software e depois exportar a agenda no formato vCalendar e importar no Direto.

O resultado de cada chamada às regras de negócio é um documento XML contendo o resultado da tarefa realizada. O XML é gerado utilizando-se a API SAX [MEG 2000], e é enviado para o componente de apresentação.

2.4.3 Dados

O Direto utiliza diversas fontes de dados para armazenar os dados do usuário. O motivo disso é que diferentes tipos de serviços necessitam de diferentes tipos de sistemas de armazenamento dependendo da natureza do serviço. Os dados do Direto estão armazenados nas seguintes bases de dados:

- Banco de Dados Relacional: Armazena todos os dados de agenda. O uso de um banco de dados relacional para a agenda deve-se a não existência de um banco de compromissos de agenda. A natureza das operações sobre os dados da agenda, com muitas consultas de leituras e muitas escritas, contribuiu para a escolha de uma base de dados relacional.
- IMAP: Todas as mensagens de correio eletrônico estão armazenadas no servidor IMAP. O motivo disso é o IMAP ser padrão para acesso e armazenamento de mensagens de correio eletrônico com algumas vantagens ao padrão POP3.
- LDAP: O uso de LDAP vem crescendo muito ultimamente. Possui como características leitura rápida e escrita demorada, sendo ideal para dados que recebem muitas consultas e poucas atualizações. O LDAP armazena todos dados relativos ao usuário, seus contatos, assim como suas preferências sobre o sistema Direto.

2.4.4 Apresentação

A camada de apresentação é responsável por receber os dados da camada de negócios e formatar os dados para serem enviados para o *browser* do usuário. Os dados são recebidos em formato XML e são transformados para HTML utilizando-se um template XSL.

A estrutura de apresentação tem como objetivo isolar as regras de negócio da apresentação. Isso facilita qualquer alteração na interface com o usuário, isola as regras de negócio do código de apresentação, e também facilita a especialização da equipe de desenvolvimento.

O componente de negócio envia o resultado em XML para a camada de apresentação, que lê um arquivo XSL, específico para cada tarefa, sobre este arquivo XSL é feito um *parser* gerando uma estrutura de árvore, e é feita uma transformação XSLT sobre as duas árvores gerando uma terceira árvore XML que contém a saída a ser apresentada para o usuário. Esta árvore é serializada, ou seja, a partir da árvore é gerada uma *stream*

para ser enviada ao usuário. Esta *stream* é uma string XML. A string XML pode ser tanto HTML, como alguma a outra linguagem derivada do XML como WML, conforme apresentado na figura 2.4.

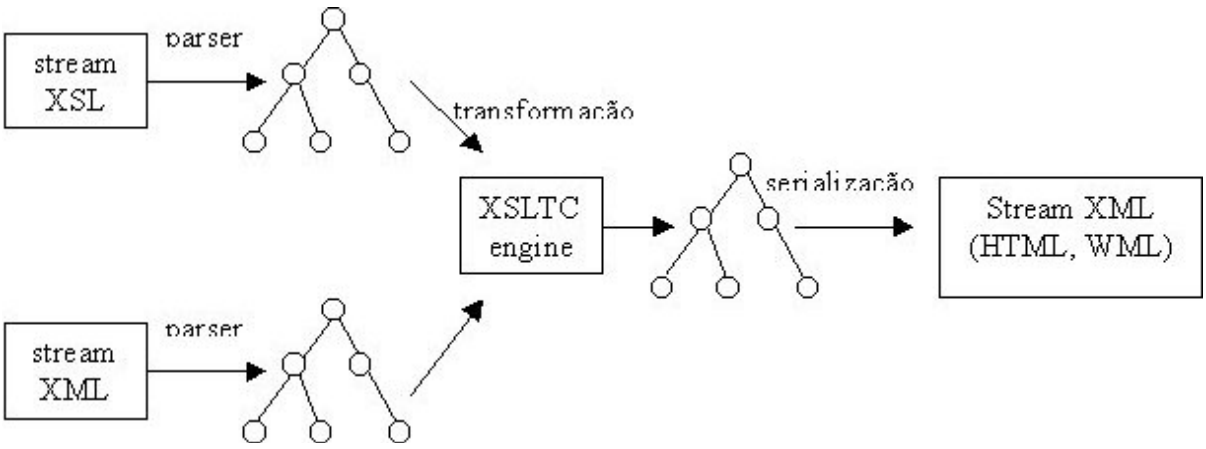

FIGURA 2.4 – Transformação XSLT

Este processo foi otimizado visando melhorar a performance. O arquivo XSL é lido apenas uma vez, depois é feito *parser* deste arquivo gerando uma árvore DOM (Document Object Model). O objeto resultante é armazenado em *cache*, e toda vez que é necessária alguma transformação XSLT com algum XSL, o mesmo é buscado na *cache*. No lado dos dados XML, a camada de negócio não retorna um *stream* XML para a camada de apresentação, o XML é gerado a partir de eventos SAX o que elimina a necessidade de *parser* sobre a *stream* XML, a árvore DOM é gerada diretamente a partir de eventos SAX.

Apesar das otimizações, o processo de geração do XML de saída é lento. Uma solução possível seria o uso de JSP para a geração com a interface com o usuário. JSP atende aos requisitos de separar a lógica de negócios da apresentação e também pode ajudar na especialização da equipe de trabalho. E ainda possui a vantagem de ser menos complexo e de mais fácil aprendizado do que XSL. Esta solução não foi utilizada porque a especificação de JSP somente foi publicada após o início do desenvolvimento do produto.

2.4.5 Tecnologias utilizadas

2.4.5.1 Servlet

Devido à importância da geração de conteúdo dinâmico para o desenvolvimento da Web, foi natural que a Sun propusesse extensões para java em seu domínio. Da mesma forma que a Sun introduziu *applets* como pequenas aplicações baseadas em Java para adicionar funcionalidade interativa aos navegadores da Web, em 1996 a Sun introduziu *servlets* [SUN 2001d] como pequenas aplicações baseadas em Java para adicionar funcionalidade dinâmica a servidores Web.

A solução *Servlet*, além de prover independência de fornecedor, tem como principais características:

- Eficiência: O código de inicialização de um *servlet* é executado somente na primeira vez em que o servidor Web o carrega. Nas próximas solicitações, serão realizadas chamadas de métodos do servlet, sem implicação de criação de um novo processo.
- Persistência: Os *servlets* podem manter estado entre requisições. Quando um *servlet* é carregado, ele permanece na memória do servidor enquanto estiver recebendo requisições.
- Robustez: Como os *servlets* são desenvolvidos com a linguagem Java, herdam sua característica de robustez, consistência e manipulação eficiente de erros. Além disso, Java provê um eficiente método de "*garbage colector*" que previne problemas com utilização de memória.
- Adaptação natural a Internet: Ainda, outra característica bastante importante é a disponibilidade de uma grande quantidade de classes. Essas classes incluem suporte a rede, suporte arquivos, acesso a banco de dados, suporte aos protocolos IMAP, LDAP, SMTP, etc.
- Extensibilidade: Por ser Java uma linguagem orientada a objetos, os *servlets* possuem as características de herança e polimorfismo, permitindo uma possível reutilização de código.

2.4.5.2 JavaServer Pages (JSP)

A especificação JavaServer Pages 1.0 [SUN 2001] foi lançada em junho de 1999, garantindo assim uma plataforma estável e bem definida, a partir da qual os fornecedores puderam desenvolver produtos. Diversos produtos de terceiros estão disponíveis para adicionar funcionalidade JSP e *servlet* a servidores Web.

JSP é uma tecnologia baseada em Java que simplifica o processo de desenvolvimento de sites dinâmicos. JSP é composto de tags que são incluídas junto ao código HTML para serem executadas durante uma requisição. O código JSP é compilado para *servlets* Java, isto garante melhor desempenho do que linguagens de scripts interpretados.

2.4.5.3 XML

XML (*eXtensible Markup Language*) [XML 2001] é um padrão para publicação, combinação e intercâmbio de documentos multimídia, desenvolvido pelo consórcio W3C (*World Wide Web Consortium*). Assim como outras linguagens de marcação, XML lida com instruções embutidas no corpo de documentos chamadas tags, que permitem a descrição de dados. XML tem como base linguagens mais antigas como SGML e HTML, sendo atualmente empregada na representação de estruturas de dados estruturados e semi-estruturados e seu intercâmbio na Web.

2.4.5.4 XSL

Um documento XSL [XSL 2001] é um documento XML que consegue transformar um documento XML em outros formatos de documentos (HTML, TeX, PostScript, RTF), além de poder utilizar alguns elementos de estilo disponíveis em CSS. A linguagem

XSL pode ser utilizada para acrescentar aspectos de apresentação aos elementos de um documento XML. Desta forma, é possível criar múltiplas representações da mesma informação a partir de vários documentos XSL diferentes aplicados a um único documento XML.

Um documento XSL pode conter uma série de regras, denominadas templates. Os templates do documento XSL são aplicados ao documento XML e o resultado obtido é o conteúdo do documento XML com o estilo de apresentação aplicado e organizado como o documento XSL especifica.

2.5 Licença do Produto

O Direto além de utilizar software de distribuição livre, também é disponibilizado como um software livre. O Direto possui licença GPL [FRE 2001a] e possui distribuição e licença gratuita, a partir do site do produto (http://www.direto.org.br) mantido pela PROCERGS, seguindo as regras de software livre.

2.5.1 Software Livre

O conceito de Software Livre surgiu em 1984 com o projeto GNU, que visava o desenvolvimento de um sistema operacional completo, similar ao Unix, e livre, com o código fonte aberto para ser utilizado por qualquer usuário, podendo ser copiado, modificado e distribuído livremente, tanto o original, como as modificações implementadas.

Um software é considerado livre [FRE 2001] quando a sua licença permite as seguintes liberdades:

- Executar o software, com qualquer propósito.
- Modificar o software e adaptá-lo às suas necessidades (para esta liberdade ser efetiva na prática, deve-se ter acesso ao código fonte).
- Redistribuir cópias, tanto grátis como com taxas.
- Distribuir versões modificadas do software, de tal modo que a comunidade possa beneficiar-se com as suas melhorias.

2.5.2 GPL

GPL é um método legal de disponibilizar um software como software livre e exigir que todas as versões modificadas e extendidas do programa também sejam software livre.

O modo mais simples de tornar um programa em software livre é coloca-lo sob o domínio público, sem copyright. Isto permite às pessoas compartilharem o programa e suas melhorias, se elas assim o desejarem. Mas isto também permite a pessoas não-cooperativas converterem o programa em software proprietário. Elas podem fazer modificações, muitas ou poucas, e distribuir o resultado como um produto proprietário. As pessoas que recebem o programa nesta versão modificada não têm as liberdades que o autor original deu a elas; o intermediário as retirou.

GPL diz que qualquer um que redistribui o software, com ou sem modificações, tem que passar adiante as liberdades de fazer novas cópias e modifica-las.

Para disponibilizar um software como *GPL*, primeiro afirma-se que ele está sob copyright; depois se adiciona o termo de distribuição, que é um instrumento legal que concede a todos o direito de usar, modificar, e redistribuir o código-fonte do programa ou qualquer outro programa derivado dele, mas somente se o termo de distribuição do mesmo permanecer inalterado. Assim, o código e as liberdades se tornam legalmente inseparáveis.

A licença apropriada é incluída em vários manuais e em cada distribuição de código-fonte.

O fato de ser software livre facilita a contribuição para o desenvolvimento e aperfeiçoamento do produto tanto através de trabalhos acadêmicos quanto da contribuição da comunidade de software livre assim como de outras empresas que adotarem o Direto como ferramenta de *groupware*.

A utilização de software livre não somente foi um dos motivos do sucesso do produto, mas foi o fator que viabilizou o projeto. O fato de utilizar software livre faz com que o Direto alavanque vários projetos que não seriam viáveis financeiramente se não utilizassem software livre.

3 Sistemas de Recuperação de Informação

Este capítulo apresenta sistemas de recuperação de informações. O capítulo está mais focado na classificação de textos, uma sub-área de SRI que objetiva classificar um texto em uma ou mais categorias existentes, utilizando o modelo do espaço vetorial. O modelo do espaço vetorial está amplamente difundido na categoria e tem como característica transformar um texto em vetor, onde cada eixo significa um termo significativo do texto.

A classificação de textos possui as etapas de pré-processamento, indexação, redução da dimensão e classificação. Para cada etapa existe mais de um método disponível que pode ser utilizado dependendo do contexto e da aplicação a que se destinam. A classificação de texto possui algumas métricas para avaliar o desempenho de sistemas baseado nos resultados obtidos a partir de uma classificação.

Também é apresentada a técnica de *clustering*, que se destina a agrupar documentos similares dentro de uma coleção. A partir deste agrupamento podem ser criadas categorias de documentos ou criado grupos de termos com significados semelhantes.

3.1 Introdução

Sistemas de Recuperação de Informações (SRI) [FAL 95] são sistemas capazes de armazenar, recuperar e manter informações e visam minimizar o esforço humano necessário para este fim. No contexto de SRI, as informações podem ser textos, imagens, áudio, vídeo e outros objetos multimídia.

A diferença entre um SRI e um Sistema de Gerenciamento de Banco de Dados (SGBD) é que para cada consulta realizada no SGBD existe apenas uma resposta correta, enquanto que para uma consulta em um SRI existem várias respostas corretas. Isso se deve porque uma consulta em um SGBD trabalha sobre uma base de dados bem estruturada, e a consulta é bastante direta utilizando a teoria de conjuntos. Em um SRI, a base de pesquisa é formada por documentos não estruturados e a consulta geralmente utiliza modelos probabilísticos. Enquanto a resposta de uma consulta a um SGBD é um conjunto com a resposta correta, a resposta para uma consulta em um SRI é a lista de documentos ordenados pelo grau de semelhança com a consulta.

3.2 Modelo do Espaço Vetorial

As técnicas apresentadas neste trabalho seguem o Modelo do Espaço Vetorial, desenvolvido por Gerald Salton [SAL 75]. Este modelo foi desenvolvido para ser utilizado em um sistema de recuperação de informações chamado SMART [SAL 83a].

Neste modelo cada documento é representado por um vetor de termos, e cada termo possui um peso associado que indica o grau de importância do mesmo no documento. Portanto, cada documento possui um vetor associado que é constituído por elementos formados pelo termo e mais o seu peso.

Neste vetor podem ser representados todos os termos do dicionário e não somente aqueles presentes no documento. Os termos que não estão presentes no documento recebem peso zero, e os outros recebem um peso referente ao seu grau de importância no documento.

O peso de cada termo pode ser calculado de diversas formas, como apresentado nas seções seguintes. A maior parte dos métodos se baseia na freqüência do termo no documento para determinar seu peso.

Cada elemento do vetor é considerado uma coordenada dimensional. Assim, os documentos podem ser colocados em um espaço euclidiano de n dimensões (onde n é o número de termos) e a posição do documento em cada dimensão é dada pelo peso do termo associado a aquela dimensão.

A distância entre dois documentos indica o grau de similaridade entre ambos, ou seja, documentos com os mesmos termos estão localizados em uma mesma região do espaço e, em teoria, possuem conteúdos similares.

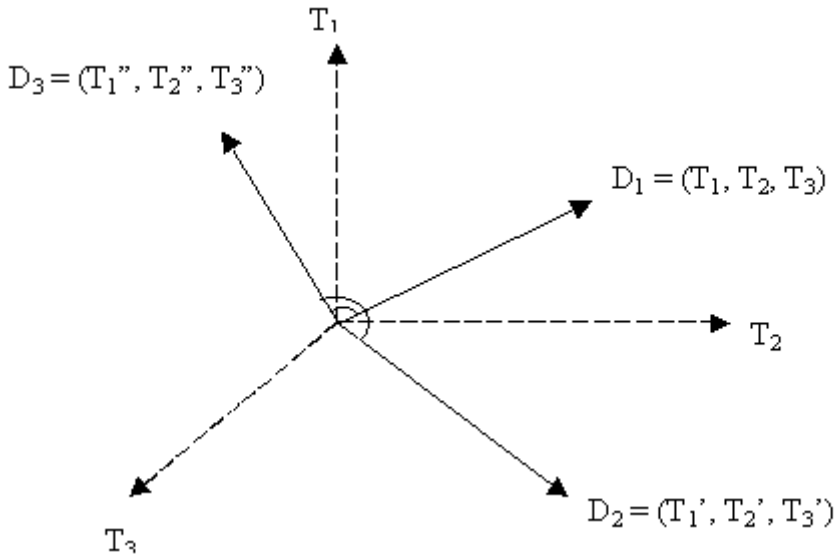

FIGURA 3.1 – Representação de documentos no Modelo do Espaço Vetorial

3.3 Pré-processamento

O primeiro passo na classificação de texto é transformar documentos, os quais geralmente são *strings* de caracteres, em uma forma de dados que sirva de entrada para o algoritmo de classificação. A transformação do texto geralmente se dá da seguinte forma:

- Conversão do formato do texto
- Remoção das *stopwords*
- Conversão dos termos em radicais

A conversão do formato do texto realiza a transformação do texto original, que pode estar no formato HTML, RTF, ou algum outro, para o formato contendo apenas texto. Isto é importante para padronizar a entrada do texto. Documentos podem vir de diferentes origens e pode estar em diferentes formatos, mas os métodos pressupõem que a entrada é somente texto, sem a formatação de apresentação.

As *stopwords* são termos freqüentes em um texto e que não carregam nenhuma informação (ex.: pronomes, preposições, conjunções, artigos, advérbios, etc) relevante, as *stopwords* não fornecem nenhuma contribuição na identificação do conteúdo do texto. A remoção das *stopwords* tem como objetivo eliminar termos que não são representativos ao documento.

A conversão dos termos em radicais consiste na remoção do sufixo dos termos para gerar apenas os radicais de acordo com as regras gramaticais da linguagem. Isto é feito para agrupar termos que possuem o mesmo significado conceitual, como classificar, classificado, classificação e classificando.

A conversão dos termos em radicais auxilia muito o processo de filtragem e classificação de documentos, tanto na redução do número de termos diferentes existentes no vetor que representa um documento (minimizando um dos maiores problemas da classificação que é o de trabalhar com enorme escala de termos), assim como também na melhor definição de peso para os termos, pois se um termo aparece n vezes em um documento e o plural do mesmo termo ocorre mais m vezes, sem a radicalização existiriam duas entradas no vetor, mas utilizando a radicalização o termo só apareceria uma vez no vetor e com $n+m$ ocorrências.

Um dos algoritmos mais citados na literatura é o *Porter Stemming Algorithm* [POR 80]. Este é um algoritmo simples e muito eficiente para a radicalização de termos. O algoritmo é executado em cinco passos, sendo cada passo realiza uma transformação sobre o termo alvo. Cada passo é formado por um conjunto de regras do tipo: Se a termo t possui mais do que s sílabas e termina com o sufixo *SUFIX*, o sufixo *SUFIX* é substituído por *SUF*.

O *Porter Stemming Algorithm* funciona para a conversão de termos em radicais apenas para termos da língua inglesa. Para a conversão de termos da língua portuguesa é necessário utilizar alguma outra técnica adaptada.

Após o pré-processamento, o texto está transformado em um conjunto de termos que são os termos que realmente podem identificar a categoria do texto.

3.4 Indexação

No Modelo do Espaço Vetorial os documentos são representados por vetores de termos. Uma coleção de documentos pode ser representada por uma matriz A de termos por documento, onde cada entrada representa a ocorrência do termo em um documento, isto é, $A = (a_{ik})$, onde a_{ik} é o peso do termo i no documento k. Como todos termos não aparecem em cada documento, a matriz A é geralmente esparsa. O número de linhas, M, da matriz corresponde ao número de termos no dicionário, podendo ser muito grande. Uma das principais característica, ou dificuldades da classificação de textos é a grande dimensão espacial.

Existem várias formas de se determinar o peso a_{ik} de um termo i em um documento k, mas todas são baseadas em duas observações sobre o texto:

- Quanto mais vezes um termo ocorre em um documento, mais relevante ele é para o tópico do documento.
- Quanto mais vezes ele ocorre dentre todos os documentos de uma coleção, menos importante ele é para diferenciar os documentos.

A seguir são apresentados diferentes métodos [SAL 88] de determinar o peso de um termo baseado no seu número de ocorrência no documento, na freqüência de ocorrência no documento, e também levando em consideração o conjunto de documentos.

TABELA 3.1 – Definição das variáveis utilizadas

i	Termo em questão
k	Documento em questão
N	Número de documentos na coleção
M	Número de termos na coleção após o pré-processamento
n_i	Número total de ocorrências do termo i na coleção
f_{ik}	Freqüência do termo i no documento k
a_{ik}	Peso do termo i no documento k

3.4.1 Boolean Weighting

Este é o método mais simples para determinar o peso de um termo. Define o peso igual a 1 se o termo ocorre no texto e 0 no caso contrário.

$$a_{ik} = \begin{cases} 1 \text{ se } f_{ik} > 0 \\ 0 \text{ caso contrário} \end{cases}$$

3.4.2 Word Frequency Weighting

Outro método simples que associa o peso do termo no documento à sua freqüência no documento. Este método está baseado na premissa de que a freqüência do termo no documento fornece informação útil sobre a importância deste termo.

$$a_{ik} = f_{ik}$$

3.4.3 tf x idf Weighting

Os dois métodos anteriores não levavam em consideração a freqüência do termo sobre todos os documentos na coleção. Um método bem conhecido para computar o peso de termos é o *tfxidf weighting*, o qual associa o peso para um termo i em um documento k

na proporção da freqüência do termo no documento, e na proporção inversa do número de documentos na coleção em que o termo aparece ao menos uma vez.

$$a_{ik} = f_{ik} * \log\left(\frac{N}{n_i}\right)$$

3.4.4 tfc Weighting

O método *tfxidf* não leva em consideração que documentos podem ser de diferentes tamanhos. O método *tfc weighting* é similar ao *tfxidf* exceto pelo fato de que a normalização do tamanho é usada como parte da fórmula para cálculo do peso.

$$a_{ik} = \frac{f_{ik} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{j=1}^{M}\left[f_{jk} \times \log\left(\frac{N}{n_j}\right)\right]^2}}$$

3.4.5 ltc Weighting

Este é um método um pouco diferente, que utiliza o logaritmo da freqüência do termo ao invés de a freqüência do termo, reduzindo assim os efeitos da enorme diferença entre as freqüências.

$$a_{ik} = \frac{\log\left(f_{ik} + 1.0\right) \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{j=1}^{M}\left[\log\left(f_{jk} + 1.0\right) \times \log\left(\frac{N}{n_j}\right)\right]^2}}$$

3.5 Redução da Dimensão

Um problema central na classificação de texto é a grande dimensão da matriz de termos. Existe uma dimensão para cada termo encontrado na coleção de documentos. Técnicas de classificação tradicionais não conseguem trabalhar com um conjunto tão grande, pois a quantidade de processamento envolvida é muito grande, e os resultados se tornariam pouco confiáveis devido à falta de dados para treinar o suficiente. Devido a estes fatos existe a necessidade de redução do conjunto de termos, a qual é chamada de redução da dimensão [RIZ 2000]. A maioria das técnicas de redução pode ser dividida em duas categorias: seleção de características e re-parametrização.

Métodos de seleção de características visam remover termos não informativos dos documentos e construir um conjunto de termos ou radicais que facilite a identificação da categoria a qual ele pertence, de forma a melhorar a eficácia da classificação e reduzir a complexidade computacional através da redução do número de termos. A premissa básica na qual grande parte dos métodos se baseia é que se um termo ocorre com grande freqüência em diversas categorias ele é insignificante para todas categorias, e se um termo ocorre com rara freqüência, ele é insignificante.

Re-parametrização é o processo de construir novas características como combinação ou transformação de características originais.

A seguir são apresentados métodos para redução da dimensão baseados na seleção de características.

3.5.1 Document frequency Thresholding

A freqüência de documento para um termo é o número de documentos em que ele ocorre. Em *Document Frequency Thresholding* [YAN 97] é computada a freqüência de documento para cada termo dentro do conjunto de treinamento de documentos, e são removidos aqueles termos que possuem freqüência de documento abaixo do valor pré-determinado. A premissa básica é que termos raros podem ser tanto termos não informativos para classificação de texto, como podem não possuir influência na performance global.

3.5.2 Ganho de Informação

O método ganho de informação [YAN 97] mede o número de bits de informação obtidos pela previsão de classificação conhecendo a presença ou ausência de um termo em um documento. O grau de representatividade de um documento em uma categoria é calculado com e sem o termo em questão. Se a diferença entre as representatividades do documento com e sem o termo for abaixo de determinado valor, o termo é desconsiderado do conjunto de termos do documento.

3.5.3 Qui-Quadrado

O método qui-quadrado [YAN 97]mede o nível de independência entre um termo *i* e uma categoria *c*. Considerando que *A* é o número de vezes em que *i* e *c* aparecem em conjunto, *B* é o número de vezes em que *i* aparece sem a presença de *c*, *C* é o número de vezes que *c* aparece sem *i*, *D* é o número de vezes que nem *c* nem *i* aparecem. É calculado da seguinte forma:

$$x^2(w_i, c_j) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)}$$

3.6 Métodos

Classificação de textos é a tarefa de atribuir automaticamente um texto a uma ou mais categorias pré-definidas. Enquanto que mais e mais informações textuais estão disponíveis on-line, a busca efetiva de informação relevante é difícil sem a devida indexação e sumarização do conteúdo de documentos. A aplicação de classificação de textos resolve este problema. Um crescente número de métodos estatísticos de classificação e técnicas de aprendizado de máquina tem sido aplicado à classificação de textos nos últimos anos. Os métodos de classificação são os procedimentos que efetivamente classificam o documento em determinada classe [WIV 2000].

A maior parte das pesquisas em classificação de texto tem sido destinada a resolver problemas binários, onde um documento é classificado como relevante ou como irrelevante com relação a um tópico pré-definido. Contudo, existem várias origens de dados textuais, como *Internet Newsgroup*, correio eletrônico e bibliotecas digitais, os quais são compostos de diferentes tópicos o que acaba gerando um problema de classificação multi-classe.

Em problemas multi-classe acontece o caso em que documentos são comuns a mais do que um tópico. Por exemplo, um artigo de *news* pode ser relevante para vários grupos. Contudo, poucos métodos têm sido desenvolvidos para classificação de texto multi-classe, onde um documento pode pertencer a mais de uma classe.

Uma abordagem comum para problemas multi-classe é quebrar o problema, considerando uma classificação para cada classe. É feita uma classificação para cada classe existente e os resultados são combinados em uma única decisão. O resultado final é uma lista de possíveis classes para o documento com o grau de pertinência do documento para cada classe.

3.6.1 Similaridade *Cosine*

No modelo do espaço vetorial, um documento é representado por um vetor de termos e seus respectivos pesos normalizados. O peso de um termo é calculado utilizando-se diversas técnicas apresentadas, as quais podem levar em consideração o número de ocorrências de determinado termo em um documento, o número total de termos em um documento, a freqüência de um termo em um documento e no conjunto de documentos de uma categoria ou de diversas categorias. A medida de similaridade *cosine* [SAL 75] calcula o cosseno do ângulo entre vetores de termos. São criados vetores para representar as categorias. Para o vetor de um documento é calculada a similaridade com todas as categorias, a categoria que possuir maior similaridade, ou que possuir similaridade maior que determinado limiar, recebe o documento.

Fórmula para similaridade entre dois documentos:

$$COSINE(D_i, D_j) = \frac{\sum_{k=1}^{t} (TERMO_{ik} \times TERMO_{jk})}{\sqrt{\sum_{k=1}^{t} (TERMO_{ik})^2 \times \sum_{k=1}^{t} (TERMO_{jk})^2}}$$

3.6.2 Rocchio

Rocchio [JOA 97] é um método clássico para filtragem ou roteamento de documentos em recuperação de informações. Neste método, um vetor protótipo é construído para cada categoria de classificação. O vetor de cada categoria é computado calculando-se a média dos vetores dos documentos da categoria, ou também dos documentos de treinamento, que incluem documentos que identificam a categoria em questão, e recebem um peso positivo no cálculo. Documentos de outras categorias também podem entrar no cálculo, estes, porém, recebem um peso negativo para o cálculo do vetor modelo da categoria.

Durante a classificação de determinado documento, é medida a distância entre o cosseno do vetor da categoria e o cosseno do vetor do documento. Caso a distância seja menor que determinado valor pré-estabelecido, o documento é inserido na categoria. Este método oferece flexibilidade para a escolha do algoritmo que calcula a distância entre os vetores, isto é, sua similaridade.

3.6.3 Naive Bayes

A classificação de *Naive Bayes* [BIL 99] é feita utilizando dados de treinamento para estimar a probabilidade do documento pertencer a cada classe. São utilizados os termos do documento com seus respectivos pesos para realizar a classificação. Para cada termo do documento, é calculada a probabilidade de o mesmo pertencer à categoria. É feita uma soma das probabilidades levando em consideração o peso dos termos e depois esta soma é normalizada. Se o resultado for maior que determinado coeficiente, o documento é incluído na categoria. Este modelo está baseado na independência dos termos, ou seja, assumem que os termos são independentes entre si.

3.6.4 Modelo Booleano

No modelo booleano [FRA 92] os documentos são considerados conjunto de termos. As expressões booleanas são utilizadas para realizar as operações sobre os conjuntos. Em uma busca, por exemplo, é feita uma consulta utilizando os operadores (*and*, *or* e *not*), e os documentos que satisfazerem esta consulta são retornados. Os documentos podem ser ordenados pelo grau de interseção, onde o mais alto é aquele que contém todos os termos especificados na consulta do usuário e o mais baixo o que contém somente uma.

Um problema do modelo booleano é o de que ele não identifica a relevância de um termo em um documento, ou seja, se um termo está presente em um documento ele é considerado muito importante (não há valores intermediários). Porém, o simples fato de um termo aparecer em um documento não significa que ele seja significativamente

importante para o documento. Existem termos que são utilizados para o encadeamento de idéias e frases (como as conjunções) que não são relevantes para a busca e informações. Há outros termos, ao contrário, que facilitam a compreensão das idéias básicas do texto e que são extremamente importantes (termos em títulos e resumos, por exemplo). O modelo booleano não leva em conta nenhum destes casos, considerando relevantes documentos que possuam os mesmos termos (mesmo que estes tenham importâncias diferentes em cada documento).

3.6.5 Vizinho mais próximo

Para classificar um vetor de um documento desconhecido, o método do vizinho próximo [YAN 97] calcula a similaridade do documento em questão com todos os documentos do conjunto. Depois os *k* documentos com a maior similaridade são ordenados, juntamente com a categoria a qual pertencem e seu grau de similaridade. É calculada a similaridade média com cada categoria, e a categoria com a maior média de similaridade recebe o novo documento.

3.6.6 Árvores de Decisão

Nesta classe o vetor do documento é comparado com uma árvore de decisão para determinar se o documento é relevante ou não. A árvore de decisão é construída a partir de exemplos de treinamento. Um dos métodos mais populares é o CART [BRE 84.

O CART constrói uma árvore de decisão dividindo o conjunto de vetores de treinamento em cada nodo de acordo com a função de um único elemento do vetor. A primeira tarefa é descobrir qual dos elementos do vetor realiza a melhor divisão, isto é, qual consegue particionar o conjunto em subconjuntos homogêneos. Para escolher o melhor divisor, cada termo é levado em consideração e é testado como possível divisor. É medido o grau de semelhança entre os documentos de cada conjunto e também o grau de diferença entre os conjuntos. O termo que conseguir dividir os documentos em dois conjuntos, com a maior homogeneidade interna e o maior grau de diferença entre conjuntos, é o escolhido.

O processo é repetido até que mais nenhum conjunto possa ser particionado, ou seja, não pode ser encontrada mais nenhuma divisão que consiga dividir algum conjunto em outros, com diferenças significativas. Neste ponto é atribuída uma categoria para cada folha da árvore. Isto não garante que cada documento do conjunto, quando avaliado pela árvore de decisão, pertença à categoria do resultado. Existe um taxa de erro nesta classificação. A taxa de erro de uma folha é definida como a probabilidade de um documento de treinamento que chegue nesta folha ser classificado erroneamente. A taxa de erro da árvore é calculada a partir das taxas de erro das folhas.

Mesmo que seja baixa a taxa de erro da árvore após a mesma ser finalizada para os documentos de treinamento, provavelmente esta não é a árvore ideal para classificar novos documentos. Isto porque a árvore foi criada objetivando os documentos de treinamento e não novos documentos. Para garantir menor taxa de erro, a árvore é podada. O objetivo da poda é remover desvios que forneçam a menor contribuição para a classificação geral.

São selecionadas subárvores a partir da árvore original, e cada subárvore é submetida a uma bateria de classificação de novos documentos, novo conjunto de treinamento. A árvore que realizar a tarefa de classificação com a menor taxa de erro é declarada vencedora.

3.7 Clustering

O *clustering* (ou agrupamento) é um método de descoberta de conhecimento utilizado para identificar co-relacionamentos e associações entre objetos, facilitando assim a identificação de classes [BOL 98]. No caso de documentos, o *clustering* identifica os documentos com assuntos similares e aloca-os em um *cluster*, gerando *clusters* de documentos similares. Isto é extremamente útil quando não se tem idéia dos assuntos tratados em cada documento e se deseja separá-los por assunto. A figura 3.2 representa um conjunto de documentos com *clusters* de documentos similares.

Esta técnica geralmente é utilizada antes do processo de classificação, facilitando a definição de classes, pois o especialista pode analisar os co-relacionamentos entre os elementos de uma coleção de documentos e identificar a melhor distribuição de classes para os documentos em questão. Isso significa que não há necessidade de se ter conhecimento prévio sobre os assuntos dos documentos ou do contexto dos documentos. Os assuntos e as classes dos documentos são descobertos automaticamente pelo processo de agrupamento. Também pode ser utilizado como auxílio na divisão de uma categoria em outras categorias.

Na área de descoberta de conhecimento em textos, o *clustering* é comumente utilizado no processo de descoberta de associações entre termos, facilitando o desenvolvimento de dicionários e *thesaurus*. *Thesaurus* significa um conjunto de termos associados que possuem o mesmo significado. Os dicionários podem ser utilizados em ferramentas de busca ou editoração de documentos, expandindo consultas ou padronizando o vocabulário dos documentos em edição.

Alguns requisitos são levados em consideração sobre os métodos de *clustering*. O método deve possuir estabilidade sobre o crescimento do conjunto de documentos, isto é, o particionamento não deve mudar drasticamente com a inserção de novos documentos. Pequenos erros na descrição de documentos devem significar pequenas modificações no particionamento. O método deve ser independente do ordenamento inicial dos documentos.

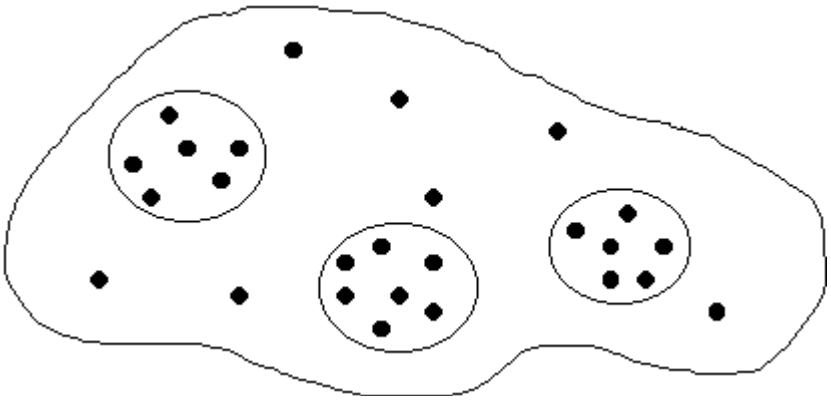

FIGURA 3.2 – Conjunto de documentos em clusters

Os métodos de *clustering* estão separados em duas classes: métodos baseados na matriz de similaridade e métodos iterativos.

3.7.1 Métodos baseados na matriz de similaridade

Estes métodos geralmente são da ordem de complexidade $\geq (n^2)$, onde n é o número de documentos e aplicam técnicas de teoria dos grafos. Uma função de similaridade de documentos é utilizada para medir a similaridade entre os documentos. Esta função mede a similaridade entre cada par de documentos. Neste método, os documentos são comparados entre si, e os de maior similaridade são agrupados no mesmo *cluster*.

3.7.2 Métodos iterativos

Esta classe consiste de métodos que possuem complexidade $< (n^2)$. Estes métodos baseiam-se diretamente na descrição do documento e não necessitam calcular matriz de similaridade entre os documentos. A classificação final depende da ordem em que os objetos são processados e o resultado de erros na descrição de documentos ocasiona modificações imprevisíveis no *clustering*. Os algoritmos são baseados em heurísticas e eles necessitam de um número de parâmetros determinados empiricamente, tais como:

- O número de *clusters* desejados;
- O tamanho máximo e o mínimo de cada *cluster*;
- Uma medida de similaridade mínima entre o documento e o *cluster*, abaixo da qual um documento não será incluído no *cluster*;
- A possibilidade de um documento estar presente em mais de um *cluster*.

Muitos métodos existem na literatura. O mais simples e mais rápido é o "*simple pass*" [YAT 99]. Cada documento é processado uma única vez e é atribuído a uma ou mais categorias, ou é criada uma nova categoria.

3.8 Avaliação do desempenho

Um item importante na classificação de textos é como medir o desempenho de um método de classificação. Várias medidas têm sido utilizadas, cada uma objetivando avaliar algum aspecto do desempenho de classificação de um sistema.

A eficiência de um SRI pode ser avaliada principalmente por duas grandezas [LEW 91], o *recall* e a *precision* de suas respostas. *Recall* refere-se ao número de documentos relevantes retornados sobre o total de documentos relevantes do universo. A *precision* refere-se ao número de documentos relevantes encontrados sobre o total de documentos retornados. Quanto maior a participação de documentos relevantes na resposta, maior a *precision* do SRI.

Na figura 3.3 é mostrada a relação entre *recall* e *precision*. No caso a, todos os documentos relevantes são retornados, e somente eles. No caso b, o sistema foi impreciso e não abrangente, retornaram muitos documentos irrelevantes e nem todos relevantes foram retornados. No caso c, todos os documentos relevantes foram retornados, mas documentos irrelevantes também foram retornados, o sistema foi abrangente e impreciso. No caso d, todos os documentos relevantes não foram retornados, mas todos retornados foram relevantes, o sistema foi preciso e não abrangente.

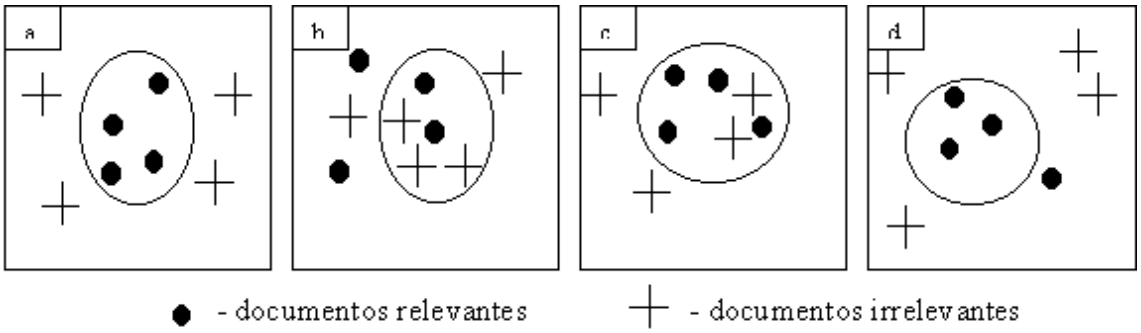

FIGURA 3.3 – Relação entre recall e precision

3.8.1 Parâmetros de desempenho

Na avaliação do desempenho de um método de classificação, quatro itens são de interesse:

- *a* - número de documentos corretamente classificados para esta categoria.
- *b* - número de documentos incorretamente classificados para esta categoria.
- *c* - número de documentos incorretamente rejeitados desta categoria.
- *d* - número de documentos corretamente rejeitados desta categoria.

Para estas quantidades são definidas as seguintes medidas de desempenho:

3.8.1.1 Recall

Avalia a habilidade do sistema em recuperar os documentos relevantes para o usuário. Ele mede a quantidade de itens relevantes, dentre os existentes na base de dados, que foram recuperados.

$$recall = \frac{a}{(a+c)}$$

3.8.1.2 Precision

Avalia a eficiência do sistema em deixar documentos irrelevantes a uma consulta fora do conjunto de resultado.

$$precision = \frac{a}{(a+b)}$$

3.8.1.3 Fallout

Avalia a taxa de retorno de documentos irrelevantes em uma consulta. É calculado dividindo o número de documentos irrelevantes retornados em uma consulta por todos os documentos irrelevantes do sistema para aquela consulta.

$$fallout = \frac{b}{(b+d)}$$

3.8.1.4 Accuracy

Avalia a quantidade de documentos que foram corretamente rejeitados ou corretamente selecionados. É calculado dividindo o total de documentos corretamente classificados pelo número total de documentos avaliados.

$$accuracy = \frac{(a+d)}{(a+b+c+d)}$$

3.8.1.5 Error

Avalia o grau de erro do sistema, considerando o número de documentos erroneamente selecionados e o número de documentos erroneamente rejeitados. Possui valor complementar ao da *accuracy*.

$$error = \frac{(b+c)}{(a+b+c+d)}$$

3.8.2 Tarefa de Múltiplas Classificações Binárias

Uma aproximação para o problema de classificar um texto entre várias categorias é quebrar a tarefa em problemas disjuntos de classificação binária. Para cada categoria é feita a verificação se o documento pertence à mesma ou não.

3.8.2.1 Micro e Macro Média

Para avaliar a média de desempenho entre categorias, existem dois métodos convencionais: micro-média e macro-média. Os valores de desempenho pela macro-média são determinados primeiramente computando as médias por categoria e, então, calculada a média de todas as categorias para definir a média do sistema. Os valores de desempenho da micro-média são determinados utilizando os valores *a, b, c,* e *d* para todas as categorias, e a partir destes totais é calculado o desempenho do sistema. Existe uma importante distinção entre os dois tipos de cálculo. Micro-média atribui um peso igual para cada documento, enquanto que macro-média atribui um peso igual para cada categoria de classificação.

3.8.2.2 *Break-Even Point*

Geralmente um sistema exibe uma dualidade entre *recall* e *precision*. Obter um alto *recall* significa sacrificar a *precision* e vice-versa. Se o *recall* e a *precision* estão configurados para se obter um valor igual, então este ponto é chamado de *break-even point* do sistema. Este ponto é comumente utilizado na avaliação da classificação de textos.

3.8.3 Classificação Multi-Classe e Multi-Label

Para medir o desempenho de um classificador que produz uma lista ordenada de categorias para cada teste com um documento, com um valor para cada categoria candidata, um método chamado *interpolated 11-point average precision* pode ser utilizado. Neste método, o *recall* para um documento específico é definido por:

$$recall = \frac{\text{Nº de categorias encontradas que estão corretas}}{\text{Nº total de categorias a que o documento pertence}}$$

Para cada um de 11 valores 0.0, ..., 1.0 desta fração, o sistema decide o quanto da lista ordenada que ele precisa seguir (isto é, o tamanho do numerador) para alcançar o *recall* especificado. A precisão então é computada para este número de categorias por:

$$precision = \frac{\text{Nº de categorias encontradas que estão corretas}}{\text{Nº total de categorias encontradas}}$$

A partir dos 11 valores de *precision* é calculada a média, que é a medida de desempenho por documento. Para um conjunto de teste de documentos, o valor médio de *precision* de documentos individuais é calculado para obter a média global de desempenho do sistema sobre o conjunto de documentos.

A implementação de um sistema de filtragem de textos é apresentada no capítulo 6. Esta implementação segue as etapas de classificação apresentadas neste capítulo, implementando um método escolhido para cada etapa. Nos capítulos 4 e 5 é apresentada uma solução para filtragem de mensagens de correio eletrônico.

4 Sieve

Este capítulo apresenta a linguagem Sieve [SIE 2001], uma linguagem padrão para filtragem de mensagens de correio eletrônico. A especificação define uma linguagem bastante poderosa, mas com algumas restrições visando um sistema de filtragem seguro. A linguagem está especificada em uma RFC. A implementação da linguagem fica a critério de cada fornecedor, o qual deve implementar um conjunto mínimo de funcionalidade e pode acrescentar extensões, algumas destas restritas a determinados ambientes. Sieve pode ser implementada tanto em servidores como também em clientes de correio eletrônico, sendo independente dos protocolos de correio utilizados.

A linguagem considera uma mensagem definida na RFC 822 [CRO 82] para aplicar as regras de filtragem, as quais se baseiam no cabeçalho da mensagem e no seu tamanho para definir a ação a ser executada. O capítulo apresenta os possíveis testes, as estruturas de controle, os comandos e os tipos de dados definidos pela especificação, além de um exemplo de um script.

4.1 Introdução

Existem várias razões para se utilizar um sistema de filtro de mensagens. A quantidade de mensagens recebidas pelos usuários de correio eletrônico tem aumentado muito devido ao grande aumento do uso de correio eletrônico para receber vários tipos de informações, como extrato bancário, listas de discussão, piadas, e diversos tipos de mensagens indesejadas.

Sieve é uma linguagem para filtro de mensagens de correio eletrônico que foi desenvolvida com o propósito de ser uma sintaxe padrão Internet para filtros de mensagens. A linguagem tem sido desenvolvida desde 1993. Em 1995, na Universidade Carnegie Mellon, através do projeto *Cyrus*, [CYR 2001] foi feita a primeira tentativa de especificar formalmente a linguagem. Desde então, através de discussões abertas e com o apoio do IETF (*Internet Engineering Task Force*), a evolução de Sieve se tornou um esforço de vários fornecedores.

Em janeiro de 2001, Sieve foi proposto como um padrão Internet, descrito na *RFC 3028 - Sieve: A Mail Filtering Language* [SHO 2001]. Existem múltiplas implementações de Sieve, rodando tanto em clientes como em servidores de e-mail.

A linguagem possui restrições de forma a permitir um sistema de filtragem seguro no servidor. A intenção é que seja impossível aos usuários fazer alguma coisa mais complexa e danosa do que escrever filtros para mensagens. A linguagem não fornece maneiras de codificar loops, escrever funções e utilizar variáveis.

A execução de scripts Sieve é feita, por definição, no momento da entrega final. Sieve define como entrega final o momento em que a mensagem é movida para a caixa postal permanente acessível pelo usuário.

Sieve reconhece as partes de uma mensagem de e-mail padrão Internet: cabeçalhos, endereços, e partes MIME distintas. Sieve não avalia o corpo de mensagens. Futuras extensões ou revisões poderão fazê-lo.

4.2 Requisitos para Sieve

A partir da experiência e de contribuições de administradores, usuários finais, e desenvolvedores de produtos clientes e servidores de e-mail na especificação de Sieve, surgiram os seguintes requisitos:

- Para correio eletrônico Internet, a linguagem deve ser especificada considerando mensagens no formato especificado pela RFC 822;
- A linguagem deve ser independente do local onde será utilizada, podendo ser tanto utilizada em clientes como em servidores;
- Deve ser utilizável por usuários de vários níveis, incluindo desde usuários finais até administradores de sistemas;
- Deve permitir mecanismo de extensões para novas funcionalidades e possível compatibilidade com funções de filtro mais avançadas.

Também existem algumas características que são desejáveis em uma linguagem padronizada de filtragem:

- Sua sintaxe e estrutura devem ser simples para a construção a partir de uma interface gráfica, e preferencialmente de relativa facilidade de leitura pelos usuários;
- A sintaxe da linguagem deve ser segura, ou seja, no caso de Sieve, não possuir loops, chamadas a funções, e variáveis. O fluxo de execução deve ser um script linearmente interpretado;
- A linguagem não deve ser muito complexa. A experiência de desenvolver novos padrões Internet sugere iniciar com uma solução simples.

4.3 Definição da linguagem

A linguagem consiste de um conjunto de comandos. Cada comando consiste de um conjunto de *tokens* delimitados por espaço. Os *tokens* são considerados não sensíveis a caixa. O identificador do comando é o primeiro *token* e é seguido por zero ou mais *tokens* de argumentos. Argumentos podem ser dados, literais, *tags*, blocos de comandos, ou comandos de testes.

4.3.1 Espaço em branco

São utilizados para separar *tokens*. São constituídos por *tabs*, *newlines* e espaço em branco. A quantidade de espaços em branco não é significativa.

4.3.2 Comentários

Comentários são equivalentes a espaço em branco, sendo eliminados do processamento. Comentários podem existir em duas formas, com o uso do caractere "#", e entre os caracteres "/*" e "*/", no primeiro caso tudo o que vier na mesma linha após o caractere "#" é considerado comentário.

Exemplo:

```
if size :over 100K { # isto é um comentário
                discard;
             }
if size :over 100K { /* isto é um comentário
                isto ainda é um comentário */ discard /*
isto é um comentário
                */ ;
             }
```

4.3.3 Dados literais

Dados literais significam dados que não são executados. São utilizados como argumentos de comandos. Dados literais são limitados a *strings* e números.

4.3.4 Testes

Testes são dados como argumentos para comandos de forma a controlar suas ações. Os testes são representados pelas estruturas de controle *if / elseif / else* para decidir qual bloco de código será executado.

Dentro das estruturas de teste são utilizados operadores. Os operadores lógicos "*e"* e "*ou"* são representados por "*allof*" e "*anyof*", respectivamente. O operador "*allof*" representa o "*e*" lógico. O resultado é verdadeiro se todas as condições forem verdadeiras. O operador "*anyof*" representa o "*ou*" lógico. O resultado é verdadeiro se uma ou mais condições forem verdadeiras.

O exemplo a seguir testa se existem os cabeçalhos "*From*" e "*Date*" ou se "*From*" é igual a "*exemplo@exemplo.com.br*". Caso alguma dessas condições seja satisfeita, a ação "*discard*" é executada.

```
if anyof (not exists ["From", "Date"],
          header :contains "From" "exemplo@exemplo.com.br")
     {
          discard;
     }
```

4.3.5 Argumentos

Para especificar uma ação a ser tomada, os comandos necessitam de argumentos. Existem três tipos de argumentos: posicional, *tagged* e opcional.

Os argumentos posicionais são passados para um comando, o qual diferencia seus significados baseado na ordem em que aparecem. Quando um comando utiliza argumentos posicionais eles devem ser fornecidos e devem estar na ordem pré-estabelecida.

Os argumentos *tagged* são utilizados em comandos que iniciam com ":" seguidos por um *tag* nomeando o argumento, tal como "*: contains*". Este argumento significa que zero ou mais dos próximos *tokens* possuem um significado em particular dependendo do argumento.

Os argumentos opcionais são exatamente como os argumentos *tagged*, com a diferença de que eles são opcionais. No caso de não serem informados, um valor padrão é aplicado.

4.3.6 Comparadores de string

Quando se compara uma string com outra existem várias formas de realizar esta comparação. Elas estão baseadas em três tipos de comparação: busca exata, busca por *substring* e busca utilizando coringas. Os tipos de comparação são utilizados como argumentos *tagged* para testar qual tipo de comparação será utilizada.

A comparação por *substring* está representada pelo tipo "*:contains*". Neste tipo de comparação são retornadas *strings* que contém a *substring* especificada para a comparação. O valor nulo "" está contido em todas as *strings* e uma busca por este valor retornará todas *strings* do texto.

A comparação por valor exato está representada pelo tipo "*:is*". Esta comparação retorna apenas as *strings* que são exatamente iguais à string procurada.

A comparação utilizando coringa está representada pelo tipo "*:matches*". Os coringas são representados pelos caracteres "*" e "?". O caractere "*" representa zero ou mais caracteres na *string* de busca. O caractere "?" representa um caractere na *string* de busca. Se for necessário realizar uma busca pelos caracteres "*" ou "?" dentro de uma *string*, devem ser utilizadas as seqüências "*" e "\\?", respectivamente, na string de busca.

Na comparação de endereços de e-mail existem outros três argumentos. O argumento "*:localpart*" significa que a comparação deve ser feita com a parte local do endereço, ou seja, o que vem antes do "@". O argumento " *:domain*" significa que a comparação é sobre o domínio do endereço, ou seja, a parte após o "@". O argumento " *:all*" significa que a comparação leva em consideração todo o endereço.

4.3.7 Blocos de comandos

Blocos são conjuntos de comandos que estão contidos entre chaves. Blocos são fornecidos a comandos de forma que os comandos possam implementar comandos de controle.

4.3.8 Comandos

Scripts Sieve são seqüências de comandos. Os comandos podem possuir argumentos *tagged* ou posicionais. Existem três tipos de comandos, comandos de teste, comandos de ação e comandos de controle.

O mais simples tipo é o comando de ação. O comando de ação é um identificador seguido por zero ou mais argumentos e terminado por ";". Não utiliza testes ou blocos como argumento.

O comando de controle possui como argumento um teste, e dependendo da avaliação do teste executa um bloco de comandos.

O comando de teste é utilizado como parte do comando de controle. O teste é utilizado para verificar quando o bloco de comandos fornecido é executado.

4.3.9 Considerações

Algumas ações não podem ser utilizadas em conjunto com outras porque o resultado seria absurdo. Algumas decisões foram tomadas visando minimizar os erros tanto de lógica quanto de funcionamento dos scripts. Estas decisões são mostradas a seguir.

4.3.9.1 "keep" implícito

Para prevenir erros em scripts que possam causar perdas de mensagens para usuários, em Sieve considera-se que em scripts que possam ser finalizados sem uma ação tomada, a ação a ser tomada é o "*keep*". Isto significa que se uma mensagem não é arquivada, redirecionada, descartada ou rejeitada, Sieve assume que ela será mantida no INBOX do usuário.

Exemplo:

```
If size :over 500k {discard; }
```

Se a mensagem tiver 500k ou menos, a ação tomada será o "*keep*".

4.3.9.2 Unicidade de mensagem em caixa postal

Implementações não devem entregar uma mensagem para a mesma pasta mais de uma vez, mesmo se o script explicitamente o tenta fazer. Isto significa que se o script

contiver uma seqüência de comandos para armazenar a mensagem em uma pasta, após já ter armazenado a mesma mensagem na mesma ou em outra pasta, este armazenamento não ocorrerá.

4.3.9.3 Limite no número de ações

Implementações podem limitar o número de ações executadas sobre uma mesma mensagem. Também pode haver restrições sobre quais ações que podem ser executadas em conjunto sobre uma mesma mensagem.

4.3.9.4 Extensões e funcionalidades opcionais

Devido à diferença de características de vários sistemas de e-mail, várias funcionalidades da especificação são opcionais. Antes de estas extensões serem executadas, elas necessitam ser declaradas através do comando "*require*".

4.3.9.5 Erros

Em linguagens de programação existem erros em tempo de compilação e erros em tempo de execução. Erros em tempo de compilação podem ser detectados através de uma avaliação sintática. Os erros de execução só podem ser verificados em tempo de execução e podem ocorrer devido a falhas transientes como, por exemplo, disco cheio. As implementações são livres para definir quais tipos de erro serão verificados e em qual momento. A única exigência é que as implementações devem avisar ao usuário sobre a ocorrência de um erro, informando o erro ocorrido e quais ações foram realizadas. A implementação deve realizar um "*keep*" implícito da mensagem, se possível.

4.4 Comandos de controle

Estruturas de controle são necessárias para permitir ações condicionais e múltiplas.

4.4.1 Estrutura de controle If

A estrutura do "*if*" é constituída de três partes, "*if*", "*elseif*", e "*else*".

Sintaxe: if <test1: test> <block1: block>
Sintaxe: elseif <test2: test> <block2: block>
Sintaxe: else <block1: block>

Exemplo:

```
require "fileinto";
             if header :contains "from" "spam" {
                discard;
             } elseif header :contains ["subject"]["$$$"] {
```

```
    discard;
} else {
    fileinto "INBOX";
}
```

4.4.2 Estrutura de controle Require

O comando *require* indica que o script utiliza certa extensão. Algumas extensões necessitam de declaração para serem utilizadas. Várias extensões podem ser declaradas em um único comando *require*. As extensões não fazem parte do *"core"* da linguagem, pois podem ser dependentes da plataforma e arquitetura do sistema de correio. Podem ser utilizadas para utilizar alguma funcionalidade específica do ambiente utilizado.

Sintaxe: require <capabilities: string-list>

4.4.3 Estrutura de controle stop

Sintaxe: stop

O *stop* finaliza todo o processamento do script. Se nenhuma ação tiver sido executada sobre a mensagem, então a ação "*keep*" é executada sobre a mensagem.

4.5 Comandos de ação

Existem cinco comandos de ação que podem ser executados sobre a mensagem: *reject, fileinto, redirect, keep, e discard*. As implementações de Sieve devem suportar as três primeiras e podem suportar as duas últimas.

4.5.1 Reject

A ação *reject* recusa a entrega da mensagem. A mensagem é retornada ao remetente com uma justificativa. O destinatário não recebe a mesma.

Sintaxe: reject <reason: string>

Exemplo:

```
if header :contains "from" "maisum@spam.com.br" {
                reject "Eu não estou recebendo as suas
mensagens de spam. Por favor não insista.";
            }
```

4.5.2 Fileinto

A ação *fileinto* entrega a mensagem para uma pasta específica. As implementações podem suportar esta ação, mas em alguns ambientes isto pode não ser possível. Para o uso desta ação é necessário declarar o seu uso através do *require*.

Sintaxe: fileinto <folder: string>

Exemplo:

```
require "fileinto";
             if header :contains ["from"] "chefe" {
                fileinto "INBOX.importantes";
             }
```

4.5.3 Redirect

A ação *redirect* é utilizada para enviar a mensagem para outro usuário a partir de um endereço especificado. Este comando não altera em nada o corpo e o cabeçalho da mensagem, mas pode incluir outros cabeçalhos. As implementações podem incluir soluções para evitar loops de mensagens através do uso deste comando. Uma alternativa é a implementação incluir algum cabeçalho de controle indicando que já foi filtrada pelo servidor na caixa de determinado usuário.

Sintaxe: redirect <address: string>

Exemplo:

```
redirect "redirect@exemplo.com.br";
```

4.5.4 Keep

A ação *keep* é a ação que é tomada quando nenhuma outra é executada. Ela simplesmente entrega a mensagem para a caixa postal padrão do usuário.

Sintaxe: keep

Exemplo:

```
if size :under 1M { keep; } else { discard; }
```

Observe que o script acima é idêntico ao script abaixo.

Exemplo:

```
if not size :under 1M { discard; }
```

4.5.5 Discard

A ação *discard* não faz nada. Ela não entrega a mensagem para o usuário, não redireciona a mensagem para nenhum endereço e também não retorna nada para quem enviou. O comando *discard* faz com que a mensagem seja descartada, e nem o destinatário e nem o remetente saibam que isto ocorreu.

Sintaxe: discard

Exemplo:

```
if header :contains ["from"] ["chato@exemplo.com.br"] {
                discard;
            }
```

4.6 Comandos de Teste

Testes são utilizados em condições para decidir qual parte da condição será executada.

Os testes obrigatórios que as implementações devem suportar são os seguintes: "*address*", "*exists*", "*false*", "*header*", "*not*", "*size*", e "*true*". A especificação define o teste "*envelope*" como opcional.

4.6.1 Address

O teste *address* verifica endereços Internet nas estruturas do cabeçalho da mensagem que contém endereços. Toda implementação deve verificar pelo menos nos seguintes cabeçalhos: *From*, *To*, *Cc*, *Bcc*, *Sender*, *Resent-From*, *Resent-To*.

Sintaxe: address [ADDRESS PART] [COMPARATOR] [MATCH-TYPE] <header-list: string-list> <key-list:string-list>

Exemplo:

```
if address :domain :is ["from"] "mycompany.com.br" {
                keep; }
```

4.6.2 Allof

O teste *allof* executa a função do operador lógico *AND*.

Sintaxe: allof <tests: test-list>

Exemplos:

```
allof (false, false)  =>   false
allof (false, true)   =>   false
```

```
allof (true,  true)   =>   true
```

4.6.3 Anyof

O teste *anyof* executa a função do operador lógico *OR*.

Sintaxe: anyof <tests: test-list>

Exemplo:

```
anyof (false, false)  =>   false
anyof (false, true)   =>   true
anyof (true,  true)   =>   true
```

4.6.4 Envelope

O teste *envelope* retorna verdadeiro quando parte do envelope SMTP [RFC 821] especificado, isto é, algum cabeçalho SMTP, for igual ao termo informado. Para o uso deste teste é necessário declarar o seu uso.

Sintaxe: envelope [COMPARATOR] [ADDRESS-PART] [MATCH-TYPE] <envelope-part: string-list> <key-list: string-list>

Exemplo:

```
require "envelope";
             if envelope :all :is "from"
"chato@exemplo.com.br" {
                discard;
             }
```

4.6.5 Exists

O teste *exists* verifica a existência de campos, os quais são passados como argumento, no cabeçalho da mensagem.

Sintaxe: exists <header-names: string-list>

Exemplo:

```
if not exists ["From","Date"] {
                discard;
             }
```

4.6.6 False

O teste *false* sempre retorna falso.

Sintaxe: false

4.6.7 Header

O teste *header* avalia se qualquer cabeçalho da mensagem é semelhante à chave informada. Se determinado cabeçalho existe na mensagem, ele contém o valor nulo “”. Se o cabeçalho não existe na mensagem, o mesmo não contém o valor nulo “”.

Sintaxe: header [COMPARATOR] [MATCH-TYPE] <header-names: string-list> <key-list: string-list>

Exemplo:

```
Supondo que a mensagem contenha no cabeçalho:
X-Direto-Empresa: PROCERGS
header :is ["X-Direto-Empresa"] [""]          => false
header :contains ["X-Direto-Empresa"] [""]    => true
```

4.6.8 Not

O teste *not* recebe algum outro teste como entrada e inverte o seu resultado.

Sintaxe: not <test>

4.6.9 Size

O teste “*size*” verifica o tamanho da mensagem e compara o tamanho da mensagem com algum valor especificado.

Sintaxe: size <”:over” / “:under”> <limit: number>

4.6.10 True

O teste “*true*” sempre avalia para verdadeiro.

Sintaxe: true

4.7 Exemplo de um script Sieve

Esta seção apresenta um script Sieve. Este script demonstra várias funcionalidades de Sieve. Todas as ações estão comentadas no próprio script.

```
# Declaração de features opcionais ou extensões utilizadas
pelo script
require ["fileinto", "reject"];

# Rejeita mensagens grandes
if size :over 1M
{
     reject text:
     Por favor não me envie mensagens grandes.
     Obrigado.
     ;
     stop;
}
# Move mensagens de lista de discussão para pasta
específica
if header :is "Sender" "listas@egroups.com"
{
     fileinto "listas";  # move para a pasta "listas"
}

# Mantém todas mensagens de colegas de trabalho
elsif address :domain :is ["From", "To"]
"procergs.rs.gov.br"
{
     keep;     # mensagens são entregues na caixa de
entrada
}

# Tenta filtrar mensagens que são spam
elsif anyof (not address :all :contains
     ["To", "Cc", "Bcc"] "spammer@spam.com",
     header :matches "subject"
     ["*make*money*fast*", "*free*course*"])
{
     fileinto "spam";   # move para a pasta "spam"
}
else
{
# Move as outras mensagens para a pasta “pessoais”
     fileinto "pessoais";
}
```

5 Filtro de mensagens para o Direto

Este capítulo apresenta uma solução para o problema de filtragem de mensagens no Direto. A solução se baseia na linguagem Sieve. É apresentado todo o processo de filtragem de uma mensagem, que se inicia com a definição de regras por parte do usuário. O usuário define as suas regras de filtragem a partir de uma interface Web, a qual restringe testes e ações que o script criado pelo usuário pode realizar. O usuário pode alterar as suas regras e ordem de execução sempre que julgar necessário.

Então, o script é armazenado no servidor para ser executado sempre que uma nova mensagem chega para o usuário. O capítulo analisa as diferenças entre o processo de filtragem realizado no servidor e o processo de filtragem realizado no cliente de correio eletrônico, e realiza uma comparação do sistema de filtragem do Direto com outro sistema de filtros.

5.1 Introdução

Uma mensagem de correio eletrônico, definida em [CRO 82], possui uma estrutura bem definida. A mensagem possui um corpo, que geralmente contém um texto não estruturado que pode ser em HTML ou *text plain*. Além do corpo também pode conter arquivos anexados. Também possui vários cabeçalhos identificando, entre outras informações, o remetente, o assunto, a data de envio e a prioridade da mensagem. Além destes cabeçalhos existem outros cabeçalhos padronizados. A mensagem também pode conter cabeçalhos específicos, não padronizados, incluídos pelo agente utilizado para o envio da mensagem. Pode-se perceber que a mensagem possui vários campos, além do texto, que podem auxiliar na filtragem da mensagem.

A proposta inicial deste trabalho era o desenvolvimento de filtros de mensagens para o Direto. Porém, com o decorrer do mesmo, o foco foi sendo alterado para o desenvolvimento de canais de informação para o Direto. Vários argumentos ajudaram nesta alteração de rumo. Alguns deles são:

- Em janeiro de 2001 foi disponibilizada a especificação da linguagem Sieve. A linguagem tem o objetivo de se tornar a linguagem padrão para a escrita de filtros para mensagens de correio;

- O atual servidor IMAP do Direto, o *Cyrus IMAP*, já implementa Sieve, então não havia necessidade de aplicar esforços para desenvolver uma solução já existente;

- O Direto é codificado em Java e o principal gargalo do sistema é o uso da CPU pelos processos Java. A implementação de uma solução de filtro de mensagens escrita em Java acentuaria o problema de desempenho.

A solução Sieve, apesar de não disponibilizar filtragem baseada no conteúdo da mensagem, atende a necessidade da grande maioria dos usuários de correio. No caso de mensagens de correio, a filtragem de mensagens pelos cabeçalhos da mensagem é satisfatória, pois tem resultado exato. A filtragem de texto por conteúdo possui resultado

probabilístico, dependendo da técnica utilizada. Como as mensagens de correio eletrônico são enviadas diretamente para o endereço do usuário, existe grande probabilidade de todas serem relevantes, com exceção dos *spams*. Além disso, Sieve tem a intenção de implementar filtragem baseada no texto da mensagem em versões futuras.

5.2 Regras para filtro de mensagens

O primeiro passo na implementação de filtros de mensagens utilizando Sieve no Direto foi a seleção das regras a serem implementadas. As regras são compostas de condições a serem testadas e de ações a serem executadas. O filtro é composto por um conjunto de regras definidas pelo usuário.

No Direto, os usuários têm a possibilidade de criar, editar e listar as regras. As regras são ordenadas e a primeira condição atingida é a que terá a sua ação executada.

As condições a serem avaliadas estão baseadas em cabeçalhos da mensagem. Os seguintes cabeçalhos são avaliados:

- To
- Cc
- Subject
- From

Além destas condições, o tamanho da mensagem também é avaliado, apesar do tamanho da mensagem não ser muito importante no desempenho de uso do Direto. O desempenho do Direto quanto ao tamanho da mensagem é indiferente no processo de listagem das mensagens, durante a exibição da mensagem é relevante quando o texto da mensagem é muito grande. Porém, dificilmente uma mensagem possui um texto no corpo com tamanho grande. As mensagens grandes geralmente são constituídas de arquivos anexados. No Direto, o único inconveniente de mensagens grandes é para fazer o download dos anexos, pois na visualização da mensagem é apenas indicado o nome dos arquivos anexos com links para o usuário fazer o download. Apesar disso, o tamanho da mensagem é importante, pois auxilia no controle da aplicação da política de uso do sistema, auxiliando a racionalizar o uso dos recursos, não aceitando mensagens grandes que poderão estourar a quota de espaço em disco de um usuário. As mensagens do Direto podem ser acessadas por qualquer software cliente de correio que implemente o seu protocolo de acesso às mensagens, no caso o protocolo IMAP. Quando se tem uma conexão lenta com o servidor IMAP e se deseja baixar todas as mensagens a partir de algum cliente de correio, mensagens com tamanho grande levarão um grande período para concluir a operação.

Além das condições avaliadas, outro item importante é o comparador utilizado na comparação da condição com o valor. Sieve suporta os comparadores de *string* ":contains", ":is", e ":matches". Estes cabeçalhos são avaliados pelo comparador ":contains". O comparador ":is" retorna verdadeiro se as *strings* são exatamente iguais, o comparador ":matches" retorna verdadeiro se a *string* especificada é uma *substring* da *string* sendo comparada, o comparador ":contains" faz uma comparação de *strings*

aceitando caracteres coringas. O comparador utilizado nos filtros do Direto é o ":matches", pois é abrangente o suficiente, aceita buscas por *substrings*, e possui fácil utilização pelo usuário por não utilizar coringas.

Após o script avaliar positivamente uma condição, uma ação será executada. As ações a serem executadas são as seguintes:

- *Fileinto*: serve para armazenar a mensagem em uma pasta específica, que não o *inbox* do usuário. Separa as mensagens recebidas em pastas. Bastante útil no uso com listas de discussão, e quando se deseja separar as mensagens recebidas em pastas com um critério bem definido;

- *Reject*: recusa a entrega da mensagem. A mensagem é retornada ao remetente com uma justificativa do não recebimento pelo destinatário. Pode ser utilizado em regras para mensagens de *spams*. A ação *discard* possui quase a mesma funcionalidade do *reject*, com a diferença que *discard* não envia mensagem ao remetente. A ação *discard* não será habilitada, pois seu uso é perigoso. Pode ser utilizada de forma indevida por usuários inexperientes ocasionando perda de mensagens.

- *Redirect*: redireciona a mensagem para outro endereço eletrônico sem deixar cópia na caixa postal do usuário.

- *Keep*: não realiza nenhuma ação sobre a mensagem. Ela é entregue normalmente na caixa postal do usuário.

Outra funcionalidade incorporada ao Direto a partir do Sieve é o aviso de férias "vacation". Esta ação é uma extensão Sieve [SHO 2000]. Quando o usuário habilita o aviso de férias, ele especifica uma mensagem que será enviada a todos que lhe enviarem mensagens. As mensagens recebidas são entregues normalmente em sua caixa postal. Esta regra pode ser combinada com outras. A diferença é que esta ação não necessita de uma condição para ser executada

5.3 Definição da interface com o usuário

Em um sistema Web a interface é um dos fatores críticos. Em comparação com os tradicionais sistemas cliente-servidor, os quais possuem um cliente "gordo" com capacidade de processamento, os sistemas Web possuem limitações de processamento no lado do cliente, pois rodam sobre um *browser*. Em sistemas Web a interface precisa ser bem planejada, pois além de ficar mais atraente e amigável com o usuário, precisa contornar limitações tecnológicas do *browser*.

As telas de criação do filtro para o Direto são criadas utilizando DHTML. Dessa forma, após o usuário selecionar uma condição, os outros componentes da tela serão alterados para serem coerentes com as opções anteriores. Por exemplo, após o usuário selecionar a ação rejeitar, aparecerá o componente para informar o motivo da recusa. Mas se o usuário selecionasse a ação mover, apareceria a lista de pastas do mesmo para que fosse possível selecionar a pasta destino.

Além da tela de criação dos filtros, existe a tela que lista todas as regras do usuário na ordem em que as mesmas são aplicadas. Esta tela possui as funcionalidades de excluir, editar, incluir e alterar a ordem em que as regras são executadas.

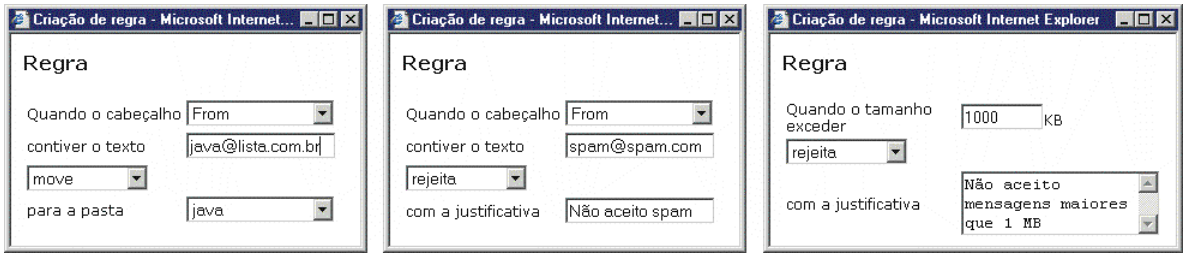

FIGURA 5.1 – Tela de criação de regras

Outra responsabilidade delegada para a interface é a geração das regras na sintaxe de Sieve a partir das opções selecionadas pelo usuário. Esta geração é feita utilizando-se a linguagem JavaScript. Após o usuário criar a regra através da interação com a interface do sistema, a interface se encarrega de criar o script Sieve relativo à regra criada e enviar a regra ao servidor. A interface também possui a responsabilidade de receber um script Sieve e apresentá-lo em formato amigável ao usuário. Este formato consiste na separação do script em um conjunto de regras, e para cada regra exibir uma descrição informando a finalidade da regra ou então montar a interface para a edição da regra.

5.4 Local de armazenamento dos scripts

Após o script Sieve chegar ao servidor, ele precisa ser armazenado em alguma estrutura. Durante o desenvolvimento deste trabalho, o serviço Sieve do servidor IMAP do Direto suportava apenas o armazenamento dos scripts em arquivos localizados em diretório específico. Durante a chegada de uma mensagem na caixa postal do usuário, o serviço automaticamente procura a existência do script no diretório definido.

No ambiente do Direto, os scripts além de serem armazenados em um arquivo no servidor, também serão armazenados no LDAP, apesar de o armazenamento do script em dois locais distintos representar redundância. O objetivo do armazenamento no LDAP é facilitar a consulta do script. O administrador, ou mesmo o usuário, possui facilidades para realizar consultas ao LDAP, porém não possui as mesmas facilidades para leitura de arquivos no servidor.

No Direto existe forte integração do LDAP com o serviço de correio, incluindo o IMAP e o SMTP. As contas IMAP, por exemplo, são autenticadas no LDAP. Esta integração do LDAP com serviços de correio não é exclusividade do Direto. Com o crescimento do uso de LDAP, vários serviços estão sendo utilizados de forma integrada ao mesmo. Já existe a definição de uma estrutura para armazenamento de scripts Sieve em LDAP [WAL 2001], e certamente os serviços Sieve em pouco tempo estarão integrados ao LDAP, de onde buscarão os scripts a serem executados.

Vários outros trabalhos estão sendo desenvolvidos sobre Sieve. Dentre eles pode-se citar alguns que estão com status de Internet Drafts:

- Protocolo para gerenciamento de scripts Sieve remotamente [MAR 2000]: Permitirá que, a partir de qualquer cliente de correio eletrônico que implementar tal protocolo, seja possível ao usuário gerenciar seus scripts Sieve localizados no servidor;

- Uma extensão para fornecer notificações instantâneas quando chegar nova mensagem [MAR 2000a]: Atualmente para ser notificado da chegada de nova mensagem é necessário verificar de tempos em tempos por novas mensagens. Com esta extensão será possível que a chegada de nova mensagem dispare um evento de notificação ao usuário. A notificação poderá se dar de várias maneiras, incluindo SMS (serviço de mensagens curtas via celular), mensagens instantâneas via ICQ, por exemplo. A extensão de notificação poderá ser combinada com os filtros, selecionando para notificação imediata, por exemplo, apenas as mensagens enviadas pelo chefe;

- Uma extensão para o uso de expressões regulares na comparação de *strings* [MUR 2001]: Esta extensão aumentará o poder de avaliação e comparação de *strings* nos scripts Sieve;

- Uma extensão para a avaliação de sub-partes de um endereço de correio eletrônico [MUR 2000];

- Uma extensão para flags IMAP [MEL 2000]. Esta extensão possibilitará alterar flags das mensagens recebidas. Isto possibilitará setar ou remover flags das mensagens, como alterar a prioridade de determinadas mensagens, ou setar flags específicos para a aplicação.

Com o avanço destes trabalhos, o sistema de filtros do Direto será alterado a fim de se adaptar às novidades. A criação de um protocolo para gerenciamento remoto de scripts ocasionará, pelo menos, a reformulação da interface dos scripts para que ela contemple todas as ações e condições de script Sieve, pois o script do usuário poderá ser alterado a partir de várias origens.

5.5 Comparação do sistema de filtros do Direto com o Outlook Express

Esta seção realiza a comparação de filtros de mensagens de correio eletrônico disponibilizado pelo MS Outlook Express 6.0 [MIC 2001] com os filtros de uma implementação de Sieve rodando no servidor.

A tabela 5.1 apresenta a comparação entre os dois sistemas de filtros.

TABELA 5.1 – Comparação Sieve x Outlook Express

	MS Outlook Express	Sieve
Executa filtro imediatamente quando mensagem chega no servidor	não	sim
Avalia cabeçalhos To, Cc, Subject	sim	sim
Avalia prioridade	sim	sim
Avalia tamanho	sim	sim
Avalia todos cabeçalhos	não	sim
Avalia corpo da mensagem	sim	não
Verifica se existem anexos	sim	não
Verifica existência de determinado cabeçalho	não	sim
Aceita o uso de coringas na comparação de *strings*	não	sim
Move ou copia para pasta	sim	sim
Rejeita mensagem com justificativa	não	sim
Exclui mensagem	sim	sim
Realça com cor ou sinaliza	sim	não
Marca como lida	sim	não
Responde com mensagem	sim	sim
Redireciona para outro endereço	sim	sim

Os dois sistemas de filtros possuem funcionalidades semelhantes. Avaliam os principais cabeçalhos das mensagens e realizam as principais ações desejadas pelo usuário, como mover para pasta e excluir, entre outras.

Uma implementação da linguagem Sieve pode ser desenvolvida para executar no cliente ou no servidor. Nesta comparação foi considerado Sieve rodando no servidor. O fato de rodar no servidor significa uma grande vantagem. Em ações de mover para pasta, excluir e alterar prioridade, o fato de rodar no servidor ou no cliente não faz diferença para os usuários. Porém, em ações como redirecionar para outro endereço, rejeitar com justificativa, responder com mensagem, e outras ações que geram envio de mensagens, o fato de rodar no servidor significa uma grande vantagem, pois a mensagem gerada é enviada no mesmo instante em que a mensagem avaliada chegou na caixa postal do usuário. Esta diferença, além de significar respostas mais rápidas, também é responsável pela possibilidade de algumas funcionalidades serem viáveis, como é o caso do aviso de férias. Quando uma mensagem chega na caixa postal do usuário, pode ser executado um script Sieve com a ação de aviso de férias. O remetente da mensagem logo receberá o aviso de que o destinatário está fora, ou de férias e só poderá ler a mensagem quando retornar. Esta funcionalidade, bastante útil, não pode ser utilizada quando o serviço de filtragens roda no cliente.

O fato do sistema de filtragem rodar no cliente possibilita algumas ações que não são possíveis quando rodam no servidor. É o caso de sinalizar ou realçar a mensagem com determinada cor. Quando o sistema de filtros roda no cliente, é possível controlar melhor a forma com que a mensagem será apresentada. Nos sistemas que rodam no servidor não é possível controlar a forma de apresentação das mensagens. A única possibilidade é setar flags padronizados no cabeçalho da mensagem, pois não se tem

idéia do cliente que o usuário utiliza, pois o mesmo pode utilizar qualquer cliente. Ainda resta a possibilidade de criar um flag customizado no cabeçalho da mensagem para um cliente específico, no caso o Direto exibir a mensagem com algum destaque.

Uma grande vantagem dos filtros do MS Outlook Express é que os mesmos podem verificar a existência ou não de termos no corpo da mensagem. Apesar da verificação de termos no corpo da mensagem não fazer parte da maioria das regras criadas pelos usuários, esta verificação pode ser utilizada para tentar identificar *spams*, mesmo sabendo-se que a técnica de utilizar lógica booleana para classificação de texto não é a mais eficiente.

A comparação de *strings* em Sieve é mais rica, pois além de fazer a comparação utilizando a *string* inteira e *substrings*, também pode utilizar coringas.

A solução inicial dos filtros para o Direto será aperfeiçoada juntamente com o desenvolvimento de Sieve. Novas funcionalidades serão incorporadas, a interface e a estrutura de armazenamento serão alteradas visando a adaptação com as novas versões.

A solução utilizando Sieve também segue a filosofia do sistema Direto, que é utilizar protocolos padrões onde possível, com o objetivo de possuir independência de fornecedor e de plataforma. A solução possibilita que usuários que utilizem qualquer cliente de correio se beneficiem dos filtros, pois a filtragem é feita no servidor.

Caso Sieve venha a se firmar como linguagem padrão para filtro de mensagens, o protocolo de gerenciamento remoto de scripts Sieve terá papel fundamental, pois garantirá a possibilidade do gerenciamento dos scripts do usuário a partir de qualquer cliente de correio que implementar o protocolo. Este protocolo exigirá que a interface de filtros do Direto aceite qualquer script Sieve válido, pois os scripts poderão ser incluídos e editados por qualquer cliente.

Como complemento do sistema de filtro de mensagens do Direto existe um trabalho de criação de um assistente de feedback para visualizar as estatísticas das preferências e escolher a configuração dos parâmetros do serviço de filtragem de e-mails do Direto. Este assistente irá sugerir a criação de novas regras de filtragem ao usuário baseado nas ações do usuário sobre as suas mensagens. Este é um trabalho de mestrado, atualmente em desenvolvimento na UFRGS por Luis Cesar de Mello.

6 Canais de informação para o Direto

Este capítulo apresenta o serviço de canais de informação para o Direto. Este serviço visa a distribuição de informações no domínio do Direto no Estado do Rio Grande do Sul. As informações são distribuídas de forma eficiente, atingindo todos e somente os interessados na informação. O serviço é composto de vários canais, cada um visando divulgar informações de determinado assunto.

Este serviço utiliza filtros definidos pelo usuário, baseados no conteúdo do texto, implementados utilizando o modelo do espaço vetorial. Para obter resultados mais precisos e eficientes na filtragem, o sistema estimula, através de facilidades, o usuário a informar mais termos na definição do seu filtro.

O capítulo apresenta todo o fluxo de um texto no sistema e todos os processos que são aplicados ao texto. Os algoritmos e as estruturas de dados utilizados são descritos.

6.1 Introdução / motivação

É sabido que o Governo do Estado do Rio Grande do Sul devido a sua natureza e grandeza é um grande gerador de informações. Estas informações são das mais diversas naturezas, podendo ser leis, acontecimentos, eventos, artigos, e das mais diversas áreas, incluindo saúde, educação, e transporte, entre outras.

A divulgação e distribuição destas informações possui um custo elevado. Existe dificuldade de se atingir todo o público de interessados em determinada informação e o inverso também é verdadeiro, também existe dificuldade para se obter todas as informações geradas e divulgadas pelo Estado sobre determinado tópico. Por exemplo, um evento de educação a distância promovido por determinada secretaria teria como público alvo possivelmente educadores e informatas, entre outros. Porém, divulgar para todos educadores e informatas seria um processo de elevado custo, e por outro lado, nem todos educadores e informatas se interessariam por este evento. Neste caso teríamos divulgado para um público muito maior do que os interessados e ainda correríamos o risco de não atingir todos interessados. Por outro lado, este tipo de divulgação faria com que o público geral recebesse muita informação indesejada. Pode-se perceber que este tipo de divulgação não é eficiente.

O Direto é um projeto que visa integrar todo o Estado em nível de correio, agenda e catálogo, e deverá ser utilizado por todas as secretarias do Estado. Ele fornece a infraestrutura necessária para que Secretarias troquem mensagens, localizem pessoas e façam agendamento entre si rapidamente e com baixo custo.

A adição de um serviço de canais de informação ao Direto cria uma teia de divulgação e distribuição de informações em todo Estado. Através de um canal, a informação é distribuída para quem realmente possui interesse na mesma. Esta divulgação possui um custo baixíssimo, pois utiliza uma infraestrutura já existente para o Direto. É um canal muito eficiente para divulgação de informações, conseguindo atingir quem realmente possui interesse na informação. Também evita que usuários recebam informações

indesejadas. A distribuição e divulgação de informações é feita de maneira muito eficiente e também com baixíssimo custo.

Estes canais de informação também podem ser utilizados em algum subdomínio do Estado, como, por exemplo, uma Secretaria. Isto possibilita às Secretarias melhorem as suas comunicações internas.

Outra conseqüência da criação de canais de informações é o início do processo de criação de perfis dos usuários do sistema, pois cada usuário fornece suas áreas de interesse através da inscrição em canais de informação. Isso possibilita a identificação de assuntos de preferência do usuário. Porém o problema de perfis de usuários não está no escopo deste trabalho.

6.2 Descrição dos Canais de Informação

Os canais de informação pretendem distribuir informações por tópicos, de forma eficiente e rápida, para todos usuários que tiverem interesse naquela informação. Usuários que não tiverem interesse, não assinando o canal, ou definindo alguma regra de filtragem, não deverão receber a informação.

A essência dos canais de informação é que eles são utilizados de maneira unidirecional para divulgação e distribuição de informações, ou seja, que determinada informação siga o fluxo do canal para encontrar o interessado na mesma, e não para que alguém interessado em determinada informação consulte todos os assinantes do canal sobre a sua dúvida. Porém, não existe nenhuma restrição que impeça que o serviço seja utilizado como lista de discussão. O objetivo dos canais de informação não é a discussão de temas, como funciona uma lista de discussão, mas sim a divulgação e distribuição de informações.

Cada canal distribuirá informação de determinado assunto. Inicialmente os assuntos serão poucos e abrangentes, e de acordo com o interesse serão refinados e divididos em outros canais até atingirem uma granularidade mínima. Para a divisão de um canal pode-se utilizar a técnica de *clustering* sobre os documentos de seu histórico para verificar possíveis subtópicos dentro do canal. Após a identificação de subtópicos, o canal poderá ser dividido em dois ou mais canais.

Os canais podem ser diferenciados entre si não apenas com relação à temática das informações divulgadas, mas também com relação a algumas particularidades, dependendo do objetivo do canal e do domínio de usuários atingidos. Algumas opções particulares a cada canal são as seguintes:

- Guardar histórico: o histórico do canal serve de base para aplicação do processo de *clustering* a fim de auxiliar o administrador na divisão do canal. O histórico também pode ser pesquisado para localizar informações já divulgadas. Porém, este histórico pode ocupar uma grande área em disco;

- Moderador do canal: cada canal pode possuir zero ou vários moderadores. Em um canal com grande fluxo de informações o trabalho de moderação pode ser dividido em vários moderadores com o objetivo de deixar o canal mais eficiente

com relação ao tempo de resposta. Caso o canal não possua nenhum moderador, as informações são enviadas diretamente aos assinantes;

- Com ou sem filtro para o usuário: pode se restringir a criação de filtros para determinados canais, isto garante que todos assinantes receberão todas informações enviadas pelo canal, ou também pode se possibilitar ao usuário criar um filtro para restringir as informações recebidas pelo canal;

- Organizações autorizadas ou domínio de assinantes do canal: esta opção serve para definir um canal que não abranja todo o domínio de usuários do Direto. Pode ser utilizado para um canal de informação interno de uma secretaria, cujas informações apenas sejam relevantes para os usuários daquela secretaria;

- Usuários cadastrados: lista de todos usuários cadastrados no canal.

No sistema de canais de informações as tarefas são executadas por diferentes tipos de usuários, os quais possuem poderes diferenciados para realizarem diferentes tarefas. Um usuário pode pertencer a várias categorias simultaneamente. Os tipos de usuários dos canais de informação são:

- Administrador: cria os canais, adiciona moderadores aos canais, define o domínio dos usuários que poderão utilizar o canal. Quebra canal em mais de um através de *clustering* se houver histórico;

- Administrador Global: possui os mesmos poderes do Administrador local, é o responsável pela criação de administradores;

- Moderador do canal: aprova ou rejeita a mensagem. Quando rejeita a mensagem informa uma justificativa para ser retornada ao usuário que a enviou. Também define o horário de envio da mensagem;

- Assinante: lista os canais que estão disponíveis para ele. Se tiver interesse pode inscrever-se no canal e também cancelar a sua assinatura. Também cria filtro para determinado canal. Na criação ou edição do filtro pode adicionar um documento texto ao seu filtro e depois editar palavras-chave e os respectivos pesos, e eliminar termos do filtro a fim de deixar o seu filtro mais refinado. Pode postar mensagens nos canais em que está inscrito.

O fluxo de um documento desde a sua inserção no canal até o seu recebimento, figura 6.1, possui três principais etapas: a inclusão do texto no canal, a aprovação pelo moderador do canal e a entrega na caixa postal do assinante. Estas etapas são detalhadas a seguir.

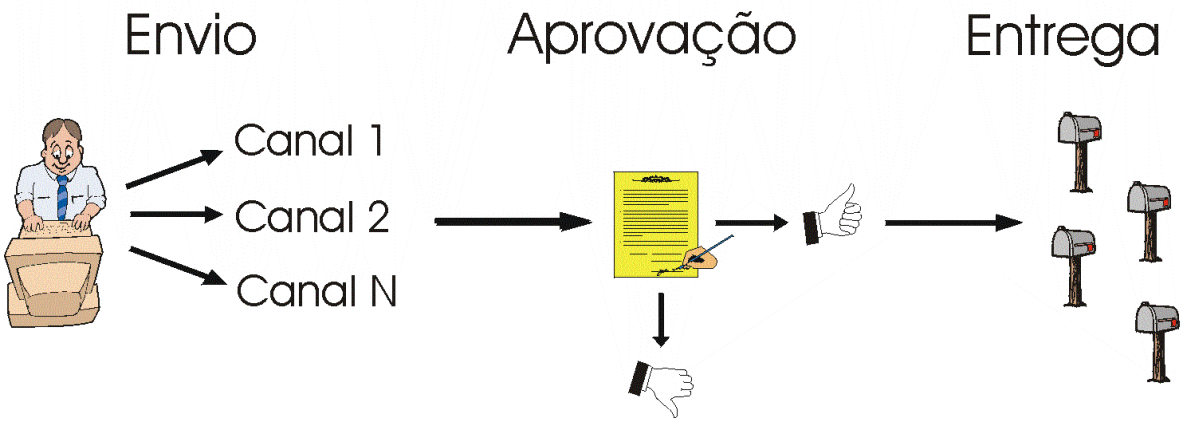

FIGURA 6.1 – Fluxo de um texto no canal

6.2.1 Inclusão do texto

Após o usuário obter e preparar a informação que ele deseja divulgar, o mesmo escolhe um ou mais canais onde a informação será divulgada. Assim como acontece em *newsgroups*, uma determinada informação pode interessar a mais de um canal, por exemplo, a divulgação de um curso de Java interessa tanto o canal de treinamento como o canal da linguagem Java. Por isso a opção de escolher vários canais.

Após o usuário selecionar os canais e informar o texto, o sistema realiza a sumarização do texto, ou seja, gera um vetor contendo os termos que melhor representam o documento e os seus respectivos pesos. Este processo está descrito na seção 6.3 (Descrição da criação de um vetor de texto).

O sumário do texto é retornado ao usuário que está divulgando a informação, recebendo o mesmo uma tela com os termos resultantes da sumarização do texto. Então, é possível editar esta lista, incluindo novos termos, removendo outros e alterando o peso de cada um deles. Esta interação com o usuário é importante, pois faz com que o mesmo valorize mais os termos mais representativos do documento e melhore a semântica da sumarização. Esta etapa, apesar de ser cansativa por ser uma tarefa a mais a ser realizada, também é interessante para quem está divulgando o texto, pois uma melhor sumarização do texto significa uma probabilidade maior de o mesmo ser recebido por quem realmente tem interesse no assunto abordado.

No sumário do texto também são inseridos metadados identificando a origem do texto. No Direto todo usuário está associado a alguma organização. Os metadados inseridos contêm palavras-chave identificando a área de atuação da organização a que o usuário pertence. Estes metadados podem valorizar o texto que aborde um assunto que seja da atividade fim da organização que gerou a informação.

Após a inclusão do texto e a sumarização do mesmo, o texto é enviado ao moderador do canal, caso o canal seja moderado. Caso contrário o texto é enviado diretamente aos assinantes do canal.

6.2.2 Aprovação do texto

Os moderadores de cada canal, lembrando que cada canal pode ter mais de um moderador, recebem um texto e têm a opção de aprovar ou rejeitar o mesmo. Caso seja rejeitado, o texto retorna ao usuário com alguma justificativa da reprovação. Caso seja aprovado, o texto é enviado aos assinantes do canal e armazenado na base de conhecimento do canal. Na base de conhecimento é armazenado o texto juntamente com seu sumário, identificação do remetente e do moderador, além de data e hora.

A figura de moderador do canal possui um papel bastante importante no sistema. Itens importantes a serem considerados são que a base de usuários do canal é muito vasta, constituída pelos funcionários do Estado que utilizam o serviço durante o seu expediente de trabalho e que as informações serão divulgadas por um canal que utiliza a ferramenta de comunicação oficial do Estado. Então, é importante um moderador que filtre textos fora do escopo do canal. O moderador evitará que *spams* e mensagens *off-topic* sejam divulgadas no canal.

É importante a definição de uma política de uso dos canais de informação. Esta política de uso regulamentará o uso dos canais e a maneira pela qual os mesmos serão utilizados. É baseado neste documento que o moderador do canal se apoiará para a aprovação ou rejeição de determinado texto.

Outro papel do moderador é definir o horário em que o texto será enviado aos usuários. Esta preocupação é valida, pois um texto pode gerar até duzentas mil mensagens, e vários textos simultâneos, em horário de pico podem comprometer o desempenho do sistema como um todo. Por isso, textos com menor urgência serão enviados em horário de menor utilização do sistema.

6.2.3 Entrega do texto

A entrega de uma mensagem de informação para os usuários do canal é feita primeiramente para os usuários que não possuem filtro. Este processo é baseado em uma lista de distribuição para os usuários sem filtro do canal. Então, para cada usuário que possui filtro para o canal, é verificado se a similaridade entre o filtro do usuário e o documento é maior que a especificada pelo usuário. Ou seja, o sumário do texto é comparado com o filtro do usuário para o canal específico. Cada usuário pode possuir um filtro específico para cada canal que o mesmo assinar. Se o texto estiver com a similaridade desejada pelo usuário com relação ao seu filtro, o texto é entregue na sua caixa postal, caso contrário é descartado. Este processo está ilustrado na figura 6.2.

A medida de similaridade do texto com o filtro do usuário é feita através do método *cosine* [SAL 75]. Este método está descrito na seção 3.6.1.

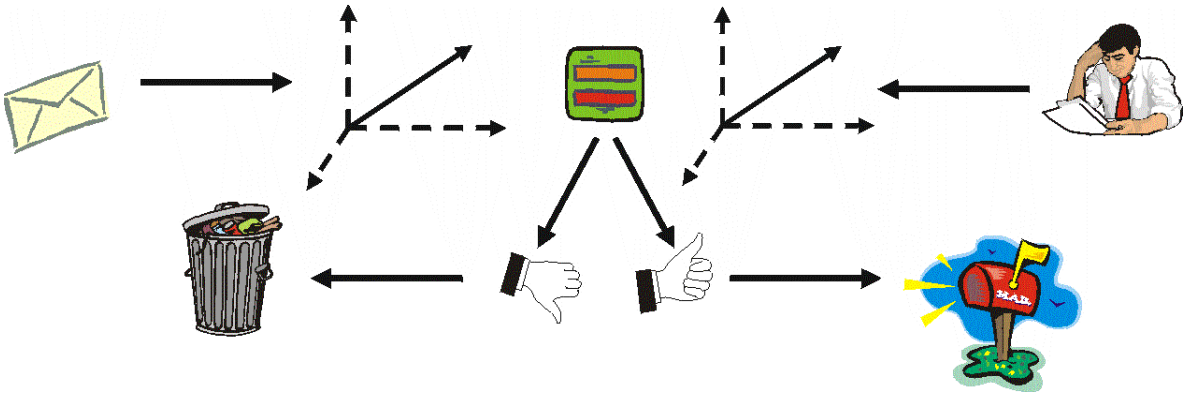

FIGURA 6.2 – Processo de recebimento de mensagens.

6.2.4 Criação do filtro pelo usuário

Na especificação dos seus filtros, os usuários geralmente tendem a colocar dois ou três termos para representar o assunto da informação que eles desejam receber. No sistema de canais de informações, que está baseado no Modelo do Espaço Vetorial, poucos termos dificilmente representam bem um texto ou uma área do conhecimento. Então, deve-se estimular o usuário a colocar um número maior de termos quando estiver definindo o seu filtro.

A forma tradicional da criação do filtro pelo usuário é disponibilizar ao mesmo uma forma de informar termos associados aos seus pesos. Neste caso o usuário inclui termos com seus respectivos pesos, edita a lista de termos alterando pesos e incluindo e removendo termos para formar o seu filtro.

Uma maneira de estimular o usuário a informar vários termos é incluir uma funcionalidade onde o usuário possa adicionar um texto ao seu filtro. O usuário seleciona um texto cujo conteúdo lhe interessa e adiciona este texto ao seu filtro. O usuário possui a possibilidade de adicionar tantos textos quanto for o seu interesse.

Sobre o texto adicionado pelo usuário é aplicada a sumarização, e os termos representativos com seus respectivos pesos são adicionados à lista de termos do filtro do usuário. Dessa forma, o filtro é refinado. E a funcionalidade de editar a lista de termos do filtro está disponível a qualquer hora, permitindo ao usuário editar o seu filtro, alterando os termos de acordo com seu interesse.

Esta solução gera filtros mais eficientes e representativos de acordo com a área de interesse do usuário.

6.3 Descrição da criação de um vetor de texto

Este tópico desenvolve a sumarização de um texto, ou seja, a geração de um vetor de termos associados a um peso, a partir do texto. Este processo se divide em duas etapas. A primeira é o pré-processamento, que prepara os termos do texto para indexação. O pré-processamento é composto dos processos de remoção de *stopwords*, transformação de termos em radicais, substituição de termos por sinônimos e inserção dos termos no

vetor, juntamente com seu número de ocorrências. A segunda etapa é composta pela indexação dos termos, que consiste em atribuir um peso a cada termo do vetor baseado em alguns critérios.

Quando um usuário recebe uma mensagem do canal, e a mesma é comparada com o seu filtro, o processo de criação do vetor é um pouco diferente. Em uma parte da mensagem já vem o sumário do texto, e o vetor é criado a partir da leitura deste sumário. Esta abordagem evita que para a mesma mensagem seja feita uma geração de vetor para cada usuário. O processo de geração de um vetor consome grande quantidade de processamento, e uma forma de economizar em processamento é gerar o sumário de um texto uma única vez. Nesta geração, o sumário é incluído em uma parte da mensagem, tendo como inconveniente o aumento do tamanho das mensagens, que pode quase duplicar. Mas entre as alternativas de consumir muita CPU ou ocupar maior área em disco e tráfego de rede, optou-se pela segunda, uma vez que no ambiente do Direto o recurso mais escasso é o processamento, e com relação à ocupação em disco, a parte da mensagem que contém o sumário pode ser removida durante a comparação com o filtro, não gerando consumo maior do recurso de armazenamento.

A seguir são apresentadas as etapas de criação do vetor a partir de um texto.

6.3.1 Remoção de stopwords

As *stopwords* são termos freqüentes em um texto e que não carregam nenhuma informação relevante, as mesmas não fornecem nenhuma contribuição na identificação do conteúdo do texto. Uma lista de *stopwords* é exibida no Anexo 1.

Para uso nos canais de informação, as *stopwords* são armazenadas em uma estrutura de mapa. A estrutura de dados *mapa* armazena pares (chave, valor). A busca pela chave é rápida, pois internamente é feito um *hash* sobre as chaves para indicar a posição que ocupam na estrutura de dados. A chave geralmente é uma *string* e o valor pode ser qualquer tipo de objeto. No caso dos canais de informação, a chave é a própria *stopword*, e o valor contém a categoria gramatical do termo juntamente com uma justificativa para o termo ser considerado *stopword*.

Na implementação, para cada *token* do texto é feita uma verificação se o mesmo consta no mapa de *stopwords*. Caso o termo conste, ele é considerado uma *stopword* e eliminado da sumarização do documento, caso contrário ele é um potencial identificador do texto.

Texto com *stopwords*	*Stopwords* riscadas	Texto sem *stopwords*
O número de habitantes do planeta Terra é de aproximadamente 6 bilhões. A taxa de crescimento vegetativo	~~O~~ número ~~de~~ habitantes ~~do~~ planeta Terra é ~~de~~ ~~aproximadamente~~ 6 bilhões. ~~A~~ taxa ~~de~~ crescimento vegetativo	número habitantes planeta Terra é 6 bilhões taxa crescimento vegetativo

FIGURA 6.3 – Processo de remoção de stopwords.

6.3.2 Transformação de termos em radicais

A conversão dos termos em radicais consiste na remoção do sufixo dos termos para gerar apenas os radicais de acordo com as regras gramaticais da linguagem. Isto é feito para agrupar termos que possuem o mesmo radical, mas diferentes sufixos.

A conversão dos termos em radicais auxilia muito o processo de filtragem e classificação de documentos, reduzindo o número de termos diferentes existentes no vetor que representa um documento, minimizando um dos maiores problemas da classificação que é o de trabalhar com enorme escala de termos, isto também melhora a definição do peso para os termos, pois se um termo aparece *n* vezes em um documento e outra flexão do mesmo termo ocorre mais *m* vezes, sem a radicalização existiriam duas entradas no vetor, mas utilizando a radicalização o termo só aparece uma vez no vetor e com *n+m* ocorrências.

Fica em aberto a especificação de um algoritmo de conversão de termos em português para seus radicais, pois não foi encontrado nenhum algoritmo de transformação de termos em radicais para a língua portuguesa. Futuramente poderá ser agregada uma função que realiza a transformação de termos em radicais sem a necessidade da reformulação do serviço.

6.3.3 Substituição de termos por sinônimos

A substituição de termos por sinônimos é uma etapa muito importante no processo de sumarização de um texto. Além de diminuir a dimensão do vetor do documento, o que reduz o processamento, também consegue determinar com maior precisão o conjunto dos termos com mesmos significados conceituais mais representativos do documento, uma vez que traduz um conjunto de termos para um único sinônimo. Estes termos com o mesmo significado são agrupados em um único termo que os representa, evitando a criação de mais de uma dimensão com o mesmo significado. O mesmo processo de substituição de termos por sinônimo é aplicado tanto nos termos que geram o filtro como também nos termos que geram o sumário de um documento. Isto garante que se o assunto do documento a ser filtrado e do filtro for semelhante e os dois utilizarem termos diferentes para desenvolver o mesmo assunto, eles serão considerados similares.

A função de substituição de termos por sinônimos recebe como entrada um termo e retorna um termo sinônimo que pode ser o mesmo termo de entrada.

Todos os termos do dicionário da língua portuguesa ficam armazenados em uma estrutura de mapa, e para cada termo armazenado na chave da estrutura está associado um outro termo que é o valor. Este valor é o sinônimo que será utilizado na geração do vetor.

6.3.4 Inclusão do termo no vetor sumário do documento

A estrutura de armazenamento dos termos durante o pré-procesamento é um mapa. Após o termo passar pelas etapas anteriores de remoção de *stopwords*, substituição por sinônimos e transformação em radical, ele é inserido no mapa.

Antes de inserir o termo, é verificado se o mesmo já consta no mapa. Se não constar, o mesmo é incluído com valor igual a um. Caso o mesmo já exista no mapa, o seu valor é incrementado.

O contador de termos do texto também é incrementado a cada inclusão de termos no mapa. Este contador é utilizado no método que define o peso dos termos no referido documento.

6.3.5 Algoritmo criaVetor

O algoritmo *criaVetor* recebe um texto como entrada e retorna um mapa contendo os termos mais representativos do documento e o peso de cada um destes termos no documento. O algoritmo realiza as seguintes tarefas:

Pré-Processamento

- Remoção de *stopwords*
- Transformação de termos em radicais
- Substituição de termos por sinônimos
- Inserção dos termos no vetor com seu número de ocorrências

Indexação

- Aplicação de método para gerar pesos de termos: utiliza o método *word frequency weighting,* que associa ao termo a sua freqüência no documento

```
CriaVetor(String texto)

numTermos = 0
para cada termo do texto
     se ehStopWord(termo)
          loop
     termo = radical(termo)
     termo = sinônimo(termo)
     insereMapaAux(termo)
     numTermos++
fim-para
para cada termo do mapaAux
     valor = mapaAux.getValor(termo)
     valor = valor / numTermos
     mapa.put(termo,valor)
fim-para
retorna mapa
```

6.4 Comparação de vetores

A comparação de vetores é feita utilizando o método de similaridade *cosine*. Este método assume que um documento pode ser representado por termos com seus respectivos pesos mapeados para um espaço cartesiano, onde cada eixo, ou dimensão, é representado por um termo. Um vetor de documento é criado ligando os pontos, pesos dos termos, em cada dimensão. A similaridade entre dois documentos é proporcional ao cosseno do ângulo formado entre eles.

$$COSINE(D_i, D_j) = \frac{\sum_{k=1}^{t} (TERMO_{ik} \times TERMO_{jk})}{\sqrt{\sum_{k=1}^{t} (TERMO_{ik})^2 \times \sum_{k=1}^{t} (TERMO_{jk})^2}}$$

O algoritmo *similaridadeVetor* recebe como parâmetro o mapa contendo o vetor do documento e o mapa do vetor do filtro, e retorna a similaridade entre ambos.

```
similaridadeVetor(Mapa filtro, Mapa documento)

acumNumerador = 0
acumDenominadorF = 0
acumDenominadorD = 0
para cada termo tf do filtro
                              vf = filtro.getValor (tf)
     vd = documento.getValor (tf)
     se vd = 0
          loop
     acumNumerador = acumNumerador  + (vd * vf)
     acumDenominadorF = acumDenominadorF +  (vf * vf)
     acumDenominadorD = acumDenominadorD +  (vd * vd)
fim-para
acumDenominador = acumDenominadorF * acumDenominadorD
acumDenominador = raizQuadrada(acumDenominador)
similaridade = acumNumerador / acumDenominador
retorna similaridade
```

6.5 Divisão de um canal

Quando o número de mensagens de um canal começa a atingir um patamar muito elevado, ou quando as informações divulgadas pelo canal começam a abordar diferentes tópicos dentro do domínio do canal, o canal pode ser dividido em mais de um.

As vantagens de se dividir os canais são muitas. Fica mais atrativo para o usuário assinar um canal que aborda um tema específico de seu interesse, do que assinar um canal genérico e ter que especificar um filtro para selecionar os textos de seu interesse.

Além disso, a informação é enviada para menos usuários mais especializados. A divisão de um canal de informática em canal de vírus, hardware e software aumenta a especialização do canal e faz com que determinada informação chegue a um número menor de usuários, usuários estes com maior probabilidade de possuir real interesse na informação. Isto diminui a importância e necessidade dos filtros.

O sistema fica com melhor desempenho, ou seja, com maior *precision* e menor *fallout*. Uma das preocupações é evitar o envio de informações a usuários que não possuem interesse na mesma, pois isto possui um custo tanto operacional como também consome o tempo do assinante.

A divisão do canal é feita pelo administrador do canal. O administrador pode obter subsídios para auxiliá-lo na definição das novas categorias a serem criadas. Ele pode se basear tanto na sua lembrança das informações enviadas pelo canal, como também pode solicitar uma ajuda ao sistema do canal para criar novas categorias.

A ajuda solicitada ao sistema do canal gera a aplicação de métodos de *clustering* [BOL 98] sobre os documentos do canal. O processo de *clustering* tem como objetivo encontrar conjuntos de documentos similares que possam gerar novos canais.

A especialização do canal informática, por exemplo, com o auxílio de *clustering*, poderia gerar os canais de hardware, software e vírus, como exibido na figura 6.4. A figura 6.4 representa todos os documentos do universo do canal informática. Após a aplicação de um método de *clustering* são encontrados grupos de documentos similares, os quais estão localizados próximos uns aos outros. Estes três clusters encontrados são fortes candidatos a tornarem-se canais.

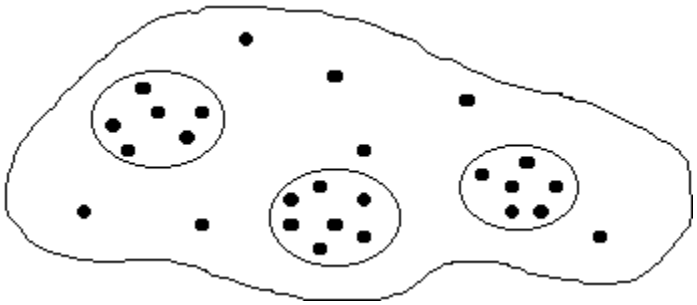

FIGURA 6.4 – Documentos do canal informática após o clustering

Existem vários algoritmos para *clustering*, os quais se baseiam em diferentes abordagens e possuem diferentes complexidades. Um dos métodos mais simples é o *Hierarchical Agglomeration Clustering* [KAR 99]. Este método inicia considerando cada documento do canal como um cluster. A cada ciclo, os dois clusters mais próximos são unificados em um cluster. A distância entre dois clusters é medida através da aplicação do método *cosine* sobre os vetores sumários dos clusters. A cada ciclo o número de clusters é diminuído em uma unidade e os ciclos são repetidos até que o número de clusters desejados seja atingido, ou então até que a maior similaridade entre dois clusters esteja abaixo de um determinado valor.

Este algoritmo possui alta complexidade. A complexidade do método pode ser melhorada se a cada ciclo exista mais de uma unificação de clusters. Os critérios podem se basear na unificação de clusters que possuam similaridade maior que determinado valor.

```
enquanto (numClusters > desejado) & (maisSimilar > simUser)
     para cluster ci de 1 a N faça
          para cluster cj de i a N faça
               se (similaridade(ci, cj) > maisSimilar)
                    maisSimilar = similaridade(ci, cj)
                    clusterSimilar = (ci, cj)
               fim-se
          fim-para
     fim-para
     unifica(clusterSimilar)
fim-enquanto
```

Após realizar o *clustering* do canal, o administrador recebe o sumário dos clusters resultantes, juntamente com o número de documentos no cluster. Estas informações auxiliam o administrador a decidir sobre os canais que serão criados, pois ele pode analisar o conjunto dos termos do sumário de cada cluster para identificar tópicos. Este processo de *clustering* serve como um sistema de apoio à decisão do administrador durante a especialização de canais.

7 Considerações finais

A popularização e o crescimento do uso do serviço de correio eletrônico, juntamente com a diversificação do seu uso, aumentou a importância deste serviço para seus usuários. Atualmente recebe-se mensagens pessoais, comerciais, listas de discussão, correntes, propagandas, informação sobre andamento de determinados processos, informações bancárias, entre outras. Este grande volume de mensagens dificulta a tarefa de leitura das mesmas pelo usuário. O uso de técnicas de filtragem juntamente com o serviço de correio eletrônico ajuda o usuário a organizar as suas mensagens recebidas aumentando a produtividade de seu uso. Algumas considerações sobre o sistema de filtragem são:

- O serviço de filtragem localizado no servidor significa que o usuário só precisa configurar uma vez seus filtros e que eles ficarão ativos no servidor até que o usuário altere as suas configurações. Já o serviço de filtragem localizado no cliente significa que o usuário necessita configurar seus filtros em cada cliente que utilizar e também que os filtros apenas estarão ativos quando o cliente estiver ativo e conectado com o servidor onde se localiza a caixa postal do usuário.

- O fato de realizar a filtragem no servidor é bastante significativo. O sistema Direto possibilita que as mensagens de correio sejam acessadas por qualquer cliente de correio IMAP, e o fato de a filtragem ser realizada no servidor significa que os filtros funcionarão independentemente do cliente de correio utilizado. Além disso, algumas funcionalidades que necessitam de retorno rápido do sistema de filtragem, tais como aviso de férias e rejeição / encaminhamento de mensagens, se tornam possíveis. Estas funcionalidades necessitam que o sistema de filtragem esteja ativo no momento da chegada da mensagem na caixa postal do usuário para que seja enviada alguma mensagem de resposta ou encaminhamento de mensagem instantaneamente. Fornecem serviços de grande valor ao usuário.

- Por outro lado, o fato do sistema de filtragem rodar no cliente possibilita algumas ações que não são possíveis quando rodam no servidor. É o caso de sinalizar ou realçar a mensagem com determinada cor. Quando o sistema de filtros roda no cliente, é possível controlar melhor a forma com que a mensagem será apresentada, pois existe total integração entre o sistema de filtragem e o cliente de correio.

- O serviço de filtragem realizado no servidor, além de possuir vantagens sobre o serviço de filtragem realizado no cliente, também possui independência do cliente de correio eletrônico utilizado para acesso à caixa postal. Como conseqüência, a caixa postal do usuário possui um comportamento único, independentemente do cliente utilizado para o acesso.

- A opção por Sieve como linguagem de filtragem de mensagem segue a filosofia de desenvolvimento do Direto. Sieve pretende ser a linguagem padrão para linguagens de filtro de mensagens de correio eletrônico Internet, e o Direto pretende, sempre que possível, utilizar padrões.

- Existem muitas facilidades para o uso de padrões. Entre elas: independência do cliente utilizado. Tanto o Direto como outro cliente que siga os padrões podem ser utilizados para realizar tarefas. Maior disponibilidade de APIs. Java disponibiliza uma ampla biblioteca de APIs, apesar de ainda não existir alguma API que trabalhe com Sieve, mas certamente se a linguagem se firmar como padrão surgirão novas APIs.

- Portanto, a adoção de Sieve como linguagem para filtragem de mensagens está em perfeita sintonia com a opção de se utilizar padrões onde for possível.

O serviço de canais de informação possibilita a divulgação de informações de forma eficiente. Uma das motivações para o seu uso é o domínio de usuários que utilizará o Direto. Serão todas as organizações do Governo do Estado do Rio Grande do Sul. O Direto, além de integrar a comunicação eletrônica dentro e entre as organizações, possibilitará, através dos canais de informação, que as informações possam ser divulgadas rapidamente e com baixíssimo custo. As considerações a respeito deste serviço são:

- Os canais de informação, além de adicionarem um novo serviço ao Direto, também possuem as suas peculiaridades. O filtro do usuário para os canais de informação realiza filtragem baseada no conteúdo do texto, para tanto utiliza o Modelo do Espaço Vetorial.

- O uso do Modelo do Espaço Vetorial está consagrado na classificação de textos. A filtragem de informações pode ser encarada como classificação de informações com as suas peculiaridades. Enquanto a classificação de informações classifica um texto em várias categorias, a filtragem de informações classifica o texto em apenas uma categoria. Se o texto passa pelo filtro ele pertence à categoria, caso contrário ele não pertence à categoria.

- Na classificação de informações, o texto é comparado com a categoria de textos, enquanto que na filtragem de informações o texto é comparado com o filtro do usuário. Usuários tendem a criar filtros com poucos termos. Um filtro com poucos termos funciona bem em sistemas de classificação baseados no modelo *booleano*. Porém, em comparações utilizando o modelo do espaço vetorial o número de termos do filtro é significativo para a comparação de dois textos. A solução de criar uma funcionalidade de adicionar um texto ao filtro do usuário estimula o usuário a criar um filtro bastante significativo aos seus interesses.

Ainda existem muitas melhorias a serem desenvolvidas e implementadas nos sistemas de classificação. Estas melhorias visam otimizar o desempenho e tempo computacional como também memória consumida e precisão da classificação do sistema.

7.1 Trabalhos Futuros

No desenvolvimento deste trabalho foram surgindo novos temas de trabalhos relacionado com o assunto abordado. Estes trabalhos visam complementar ou aperfeiçoar o mesmo. São eles:

- Mineração de perfis de usuários. Esta atividade consistiria em realizar uma mineração sobre os canais assinados pelos usuários. O resultado seria inferências do tipo: 70% dos usuários que assinam o canal X e o canal Y também assinam o canal Z.

- A classe implementada e os algoritmos desenvolvidos podem ser aproveitados em outras aplicações de classificação de texto, tal como uma aplicação que extrai notícias da web e classifica em diversas categorias.

- O serviço de canais de informação recebe apenas documentos no formato texto como entrada. Uma ferramenta que extraísse o texto a partir de arquivos em diversos formatos, como .pdf, .doc, .rtf, .html entre outros, aumentaria a gama de possíveis documentos a serem enviados pelo serviço de canais de informação.

- No modelo do espaço vetorial, o processo de transformação dos termos em radicais resulta em uma maior representatividade e precisão dos termos do vetor gerado. Um algoritmo eficiente para transformação de termos da língua portuguesa em radicais geraria vetores de termos mais significativos em relação ao texto original.

- Outra melhoria possível no serviço de canais de informação está no processo de postagem de algum texto. Uma alteração é partir do sistema a sugestão do canal em que o texto será postado. Para tanto, é necessário gerar um vetor representativo de cada canal a partir de seu histórico.

Anexo Lista de *stopwords*

Advérbios de afirmação

- sim
- deveras
- efetivamente
- certamente
- realmente
- deverás

Advérbios de dúvida

- talvez
- quica
- porventura
- acaso
- possivelmente
- provavelmente

Advérbios de intensidade

- mais
- menos
- muito
- pouco
- tao
- tanto
- assaz
- bastante
- bem
- demais
- quanto
- quao
- quase

Advérbios de lugar

- ali
- aqui

Advérbios de modo

- mal
- corretamente
- calmamente
- assim
- bem
- debalde
- depressa
- devagar
- melhor
- pior
- fielmente
- levemente

Advérbios de negação

- nao
- nunca
- jamais
- absolutamente
- não

Advérbios de ordem

- primeiramente
- ultimamente
- depois

Advérbios de tempo

- hoje
- amanha
- sempre
- logo
- agora
- depois
- ainda
- anteontem
- antes
- breve
- cedo
- entao
- ja
- jamais
- nunca

- ontem
- outrora
- tarde
- amanhã
- então
- já

Artigos definidos

- a
- as
- o
- os

Artigos indefinidos

- uns
- umas
- uma
- um

Artigos

- o
- os
- a
- as
- um
- uns
- uma
- umas
- ao
- do
- no
- pelo
- da
- na
- pela
- aos
- dos
- nos
- pelos
- das
- nas
- pelas
- num
- numa
- nuns
- numas
- dum
- duma
- duns
- dumas

Consoantes

- b
- c
- d
- f
- g
- h
- j
- k
- l
- m
- n
- p
- q
- r
- s
- t
- v
- x
- y
- z
- w

Conjunções

- que
- mas
- porém
- porem
- ou
- e

Interjeições

- ah
- oh
- oba
- opa
- avante
- coragem
- eia
- vamos
- bis
- bem
- bravo
- viva
- oxala
- ai
- ui

- chi
- ih
- ue
- puxa
- hum
- hem
- alo
- o
- ola
- psiu
- psit
- silencio
- alto
- basta
- uh

Numerais cardinais

- um
- dois
- tres
- quatro
- cinco
- seis
- sete
- oito
- nove
- dez
- onze
- doze
- treze
- quatorze
- quinze
- dezesseis
- dezessete
- dezoito
- dezenove
- vinte
- trinta
- quarenta
- cinquenta
- sessenta
- setenta
- oitenta
- noventa
- cem
- duzentos
- trezentos
- quatrocentos
- quinhentos
- seiscentos
- setecentos
- oitocentos
- novecentos
- mil
- milhao
- bilhao

Numerais ordinais

- primeiro
- segundo
- terceiro
- quarto
- quinto
- sexto
- setimo
- oitavo
- nono
- decimo
- vigesimo
- trigesimo
- quadragesimo
- quinquagesimo
- sexagesimo
- septuagesimo
- octogesimo
- nonagesimo
- centesimo
- ducentesimo
- trecentesimo
- quadrigentesimo
- quingentesimo
- seiscentesimo
- sexcentesimo
- septingentesimo
- octingentesimo
- nogentesimo
- milesimo
- milesimos
- milionesimo
- milionesimos
- bilionesimo
- bilionesimos
- primeira
- segunda
- terceira
- quarta
- quinta
- sexta
- setima
- oitava
- decima

Preposições

- a
- ante
- até
- após
- com
- contra
- de
- desde
- por
- para
- perante
- sem
- sobre
- sob
- trás
- tras
- ate
- apos

- pára
- ao
- à
- aos
- às
- do
- da
- dos
- das
- no
- na
- nos
- nas
- pelo
- pela
- pelos
- pelas
- dum
- duma
- duns
- dumas
- num
- numa
- nuns
- numas

Pronomes de tratamento

- você
- voce
- senhor
- senhora
- vossa
- senhoria
- sua
- senhorita

Pronomes demonstrativos

- este
- esse
- aquele
- estes
- esses
- aqueles
- esta
- essa
- aquela
- estas
- essas
- aquelas
- isto
- isso
- aquilo
- tal
- tais

Pronomes indefinidos

- alguém
- ninguém
- algo
- nada
- tudo
- cada
- qualquer
- algum
- outro
- nenhum
- muito
- pouco
- mais
- menos
- varios
- alguns
- nenhuns
- todo
- todos
- outros
- muitos
- poucos
- certo
- certos
- vario
- tanto
- tantos
- quanto
- quantos
- quaisquer
- alguma
- algumas
- nenhuma
- nenhumas
- toda
- todas
- outra
- outras
- muita
- muitas
- pouca
- poucas
- certa
- certas
- varia
- varias
- tanta
- tantas
- quanta
- quantas
- outrem

Pronomes interrogativos

- que
- quem
- qual

- quanto
- quantos
- quais

Pronomes pessoais oblíquos átonos

- me
- te
- o
- a
- lhe
- se
- nos
- vos
- os
- as
- lhes

Pronomes pessoais oblíquos tônicos

- mim
- ti
- si
- comigo
- contigo
- conosco
- convosco
- consigo
- nos
- vos

Pronomes pessoais reto

- eu
- tu
- ele
- ela
- eles
- elas
- nós
- vós

Pronomes possessivos

- meu
- teu
- seu
- meus
- teus
- seus
- nosso
- vosso
- nossos
- vossos

Pronomes relativos

- qual
- quais
- que
- quem
- onde
- como
- cujo
- quanto
- aonde
- cuja

Vogais

- a
- e
- i
- o
- u

Bibliografia

[AAS 99] AAS K.; EIKVIL L. **Text categorisation:** a survey. [S.l.]: Norwegian Computing Center, 1999. Disponível em: <http://citeseer.nj.nec.com/aas99text.html>. Acesso em: 25 out. 2001.

[APA 2001] APACHE SOFTWARE FOUNDATION. **Apache HTTPD Project.** Disponível em <http://httpd.apache.org>. Acesso em: 15 nov. 2001.

[BIL 99] BILLSUS, D.; PAZZANI, M. J. A Hybrid User Model for News Story Classification. In: INTERNATIONAL CONFERENCE ON USER MODELING, 1999, Banff, Canada. **Proceedings...** Disponível em: <http://citeseer.nj.nec.com/billsus99hybrid.html>. Acesso em: 20 jan. 2000.

[BOL 99] BOLEY, D.; GINI, M.; GROSS, R.; HAN, E.; HASTINGS, K. **Partitioning-Based Clustering for Web Document Categorization.** [S.l.]: Decision Support Systems, 1999. Disponível em: <http://citeseer.nj.nec.com/9105.html>. Acesso em: 20 jan. 2000.

[BOR 97] BORGES, Clairmont. **Serviços Básicos para Facilitar o Consumo de Informação num Cenário de Rede**. 1997. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BRE 84] BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and Regression Trees**. Wadsworth, Belmont, Canada: [s.n.], 1984

[CRI 96] CRISPIN, M. **Internet Message Access Protocol,** v.4, RFC 2060. **N**ov. 1996. Disponível em: <http://www.ietf.org/rfc/rfc2060.txt>. Acesso em: 12 ago. 2000.

[CRO 82] CROCKER, D. **Standard for the Format of ARPA Internet Text Messages**, RFC 822. Aug. 1982. Disponível em: <http://www.ietf.org/rfc/rfc822.txt>. Acesso em: 12 ago. 2000.

[CYR 2001] CYRUS Project. Disponível em: <http://asg.web.cmu.edu/cyrus/>. Acesso em: 15 nov. 2001.

[DAW 98] DAWSON, F.; STERNESON, D. **Internet Calendaring and Scheduling Core Specification** (iCalendar), RFC 2445. Nov. 1998. Disponível em: <http://www.imc.org/draft-ietf-calsch-ical-main>. Acesso em: 11 nov. 2001.

[DIR 2000] DIRETO: Software de Correio, Catálogo e Agenda Corporativo. Disponível em: <http://www.direto.org.br>. Acesso em: 10 abr. 2002.

[FAL 95] FALOUTSOS, C.; OARD, D. W. **A Survey of Information Retrieval and Filtering Methods.** College Park, MD: University of Maryland. Aug. 1995. Disponível em: <http://citeseer.nj.nec.com/faloutsos96survey.html>. Acesso em 15 jan. 2001.

[FRA 92] FRAKES, W. B.; YATES, R. B. **Information Retrieval, Data Structures & Algorithms.** Englewood Cliffs, NJ: Prentice Hall, 1992.

[FRE 2001] FREE SOFTWARE FOUNDATION. **Software Categories**. Disponível em: <http://www.fsf.org/philosophy/categories.html>. Acesso em: 11 nov. 2001.

[FRE 2001a] FREE SOFTWARE FOUNDATION. **GNU General Public License**. Disponível em: <http://www.fsf.org/copyleft/gpl.html>. Acesso em: 11 nov. 2001.

[GRA 95] GRAY, T. **Message Access Paradigms and Protocols.** Sept. 1995. Disponível em: <ftp://cac.washigton.edu/mail/imap.vs.pop>. Acesso em: 11 nov. 2001.

[INT 2001] INTERNET MAIL CONSORTIUM. Disponível em: <http://www.imc.org>. Acesso em: 11 nov. 2001.

[JOA 97] JOACHIMS, T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 1997. **Proceedings...** Germany: [s.n.], 1997.

[KAR 99] KARYPIS, G.; HAN, E.; KUMAR, V. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. **Computer.** Special Issue on Data Analysis and Mining, 1999. Disponível em: <ftp://ftp.cs.umn.edu/dept/users/kumar/chameleon.ps>. Acesso em: 11 nov. 2001.

[KRI 97] KRISTOL, D.; MONTULLI, L. **HTTP State Management Mechanism**, RFC 2109. Feb. 1997. Disponível em: <http://www.ietf.org/rfc/rfc2109.txt>. Acesso em: 11 nov. 2001.

[LEW 91] LEWIS, D. D. Evaluating Text Categorization. In: THE SPEECH AND NATURAL LANGUAGE WORKSHOP, 1991. **Proceedings...** San Mateo: Morgan-Kaufmann, 1991.

[LIN 2001] LINUX. **The Linux Homepage.** Disponível em: <http://www.linux.org>. Acesso em: 15 nov. 2001.

[MAR 2000] MARTIN, Tim. **A Protocol for Remotely Managing Sieve Scripts**, Internet Draft. Dec. 2000. Disponível em: <http://search.ietf.org/internet-drafts/draft-martin-managesieve-03.txt>. Acesso em: 18 out. 2001.

[MAR 2000a] MARTIN, Tim. **Sieve - An Extension for Providing Instant Notifications,** Internet Draft. Dec. 2000. Disponível em: <http://search.ietf.org/internet-drafts/draft-martin-sieve-notify-01.txt>. Acesso em: 18 out. 2001.

[MEL 2000] MELNIKOV, Alexey. **Sieve - IMAP Flag Extension,** Internet Draft. Oct. 2000. Disponível em: <http://www.imap.org/papers/docs/sieve-flag.html>. Acesso em: 18 out. 2001.

[MEG 2000] MEGGINSON, David. **Simple API for XML**. May 2000. Disponível em: <http://www.megginson.com/SAX/>. Acesso em: 15 nov. 2001.

[MIC 2001] MICROSOFT CORPORATION. **Outlook Express.** Disponível em: <http://www.microsoft.com/windows/oe/>. Acesso em: 15 nov. 2001.

[MUR 2000] MURCHISON, Ken. **Sieve - Subaddress Extension**, Internet Draft. Sept. 2000. Disponível em: <http://search.ietf.org/internet-drafts/draft-murchison-sieve-subaddress-03.txt>. Acesso em: 15 nov. 2001.

[MUR 2001] MURCHISON, Ken. **Sieve - Regular Expression Extension,** Internet Draft. Jan. 2001. Disponível em: <http://search.ietf.org/internet-drafts/draft-murchison-sieve-regex-04.txt>. Acesso em: 15 nov. 2001.

[MYE 94] MYERS, J.; ROSE, M. **Post Office Protocol,** v.3, RFC 1725. Nov. 1994. Disponível em: <http://www.ietf.org/rfc/rfc1725.txt>. Acesso em: 15 nov. 2001.

[POR 80] PORTER, M. F. An algorithm for suffix stripping. **Program**, [S.l.], v.14, n. 3, p. 130-137, 1980. Disponível em: <http://telemat.die.unifi.it/book/2001/wchange/download/stem_porter.html>. Acesso em: 31 maio 2001.

[POS 82] POSTEL, Jonathan B. **Simple Mail Transfer Protocol,** RFC 821. Aug. 1982. Disponível em: <http://www.ietf.org/rfc/rfc821.txt>. Acesso em: 15 nov. 2001.

[RIZ 2000] RIZZI, C. B. **Categorização de Texto por Rede Neural:** Estudo de Caso. 2000. 212p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[SAL 75] SALTON, G.; WONG, A.;YANG C.S. A Vector Space Model for Automatic Indexing. **Communications of the ACM**, New York, v.18, 1975.

[SAL 83] SALTON G.; MCGILL M. J. **Introduction to Modern Information Retrieval**. New York: McGraw-Hill, 1983.

[SAL 83a] SALTON G.; MCGILL M. J. **The SMART and SIRE Experimental Retrieval Systems.** New York: McGraw-Hill, 1983.

[SAL 88] SALTON, G.; BUCKLEY, C. Term weighting approaches in automatic text retrieval. **Information Processing and Management**, Oxford, v. 24, n. 5, p. 513-523, 1983.

[SHO 2000] SHOWALTER, Tim. **Sieve Vacation Extension,** Internet Draft. Aug. 2000. Disponível em: <http://search.ietf.org/internet-drafts/draft-showalter-sieve-vacation-04.txt>. Acesso em: 11 nov. 2001.

[SHO 2001] SHOWALTER, T. **Sieve: A Mail Filtering Language,** RFC 3028. Jan. 2001. Disponível em: <http://www.ietf.org/rfc/rfc3028.txt>. Acesso em: 11 nov. 2001.

[SIE 2001] SIEVE: A Mail Filtering Language. Disponível em: <http://www.cyrusoft.com/sieve>. Acesso em: 12 jun. 2001.

[SUN 95] SUN MICROSYSTEMS. **Java Programming Language**. 1995. Disponível em: <http://java.sun.com>. Acesso em: 11 nov. 2001.

[SUN 2001] SUN MICROSYSTEMS. **JavaServer Pages**. Disponível em: <http://java.sun.com/products/jsp/>. Acesso em: 11 nov. 2001.

[SUN 2001a] SUN MICROSYSTEMS. **Javamail API**. v.1.2. Disponível em: <http://java.sun.com/products/javamail/index.html>. Acesso em 11 nov. 2001.

[SUN 2001b] SUN MICROSYSTEMS. **JNDI API**. Disponível em: <http://java.sun.com/products/jndi/>. Acesso em: 11 nov. 2001.

[SUN 2001c] SUN MICROSYSTEMS. **JDBC API**. Disponível em: <http://java.sun.com/products/jdbc>. Acesso em: 11 nov. 2001.

[SUN 2001d] SUN MICROSYSTEMS. **Servlets API**. v.2.2. Disponível em: <http://java.sun.com/products/servlet>. Acesso em: 11 nov. 2001.

[VCA 2001] VCALENDAR. Disponível em: <http://www.imc.org/pdi/>. Acesso em: 11 nov. 2001.

[WAL 2001] WALL, M. **The Sieve Language and a General Model for Delivery and Interoperable Filtering in Internet Mail.** Pittsburgh, Feb. 2001. Disponível em: <http://www.cyrusoft.com/sieve>. Acesso em: 11 nov. 2001.

[WIV 2000] WIVES, L. K. **Tecnologias de Descoberta de Conhecimento em Textos Aplicadas à Inteligência Competitiva**. 2000. Exame de Qualificação (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[XML 2001] XML: Extensible Markup Language. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 11 nov. 2001.

[XSL 2001] XSL Transformations (XSLT). Disponível em: <http://www.w3.org/TR/xslt>. Acesso em: 11 nov. 2001.

[YAN 97] YANG, Y.; PEDERSEN, J. O. A Comparative Study on Feature Selection in Text Categorization. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 14., 1997. **Proceedings...** San Francisco: Morgan Kaufmann Publishers, 1997.

[YAT 99] YATES, R. B.; RIBEIRO NETO, B. **Modern Information Retrieval.** [S. l.]: Addison Wesley Longman,1999.

[YEO 95] YEONG, W.; HOWES, T.; KILLE, S. **Lightweight Directory Access Protocol,** RFC 1777. May 1995. Disponível em: <http://www.ietf.org/rfc/rfc1777.txt>. Acesso em: 11 nov. 2001.