MINERAÇÃO DE DADOS TEXTUAIS UTILIZANDO TÉCNICAS DE CLUSTERING PARA O IDIOMA PORTUGUÊS

Maria Célia Santos Lopes

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

_____

Prof. Nelson Francisco Favilla Ebecken, D.Sc

_____

Prof. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof. Elton Fernandes, D.Sc.

_____

Prof. Alexandre Gonçalves Evsukoff, Dr.

_____

Prof. Eduardo Raul Hruschka, D.Sc.

_____

Prof. Geraldo Martins Tavares, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2004

LOPES, MARIA CÉLIA SANTOS

Mineração de Dados Textuais Utilizando Técnicas de Clustering para o Idioma Português [Rio de Janeiro] 2004

XI, 180 p. 29,7 cm (COPPE/UFRJ, D.SC., Engenharia Civil, 2004)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Text mining

2. Clustering de Dados Textuais

3. Representação Visual de Resultados

I. COPPE/UFRJ II. Título (série)

A Gerson e Matheus

A meus pais

AGRADECIMENTOS

A meu marido Gerson pelo apoio e ajuda e, principalmente, pela paciência em todos os momentos ao longo da realização desta tese.

A meus pais pela constante presença e apoio incondicionais sem os quais se tornaria bem difícil alcançar este objetivo.

Ao meu orientador prof. Nelson Francisco Favilla Ebecken pelo apoio e incentivo no desenvolvimento da tese.

Ao CNPQ pelo suporte financeiro que viabilizou a realização desta tese.

Ao Laboratório do Núcleo de Transferência de Tecnologia – NTT, pela infra-estrutura, suporte administrativo e logístico.

Gostaria de agradecer também a Rodrigo Bessa pelo trabalho caprichoso das figuras que ilustraram esta tese e a todos os demais amigos do NTT, Estela Estrella e Guilherme Saad Terra, pelo apoio e incentivo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D. Sc.)

MINERAÇÃO DE DADOS TEXTUAIS UTILIZANDO TÉCNICAS DE CLUSTERING PARA O IDIOMA PORTUGUÊS

Maria Célia Santos Lopes

Outubro/2004

Orientador: Nelson Francisco Favilla Ebecken

Programa: Engenharia Civil

Esta tese se concentra no desenvolvimento de uma solução que realize o clustering de documentos cujos conteúdos se apresentam no idioma Português. O presente trabalho é composto de três módulos distintos, sendo que cada módulo gera a entrada para o módulo seguinte. O primeiro módulo realiza o pré-processamento de dados textuais fazendo as considerações necessárias para o tratamento de dados textuais em Português. O segundo módulo é o módulo de clustering de dados, que disponibiliza diferentes métodos, é alimentado pela etapa de preparação dos dados, e gera a saída de acordo com o método escolhido. A visualização de resultados é considerada de grande auxílio para a interpretação e utilização dos resultados fornecidos por um processo de clustering. Dessa forma, o terceiro módulo é o módulo de visualização que disponibiliza uma forma facilmente interpretável de visualização de resultados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D. Sc.)

TEXT DATA MINING USING CLUSTERING TECHNIQUES FOR PORTUGUESE LANGUAGE

Maria Célia Santos Lopes

October/2004

Advisor: Nelson Francisco Favilla Ebecken

Department: Civil Engineering

This thesis focuses on the development of a document clustering solution which deals with document contents in Portuguese. The current work presents three distinct modules, each module generating the input to the next one. The first module presents text data preprocessing, performing all considerations needed for textual data treatment in Portuguese. The second module is the clustering module which disposes different methods, it is fed with data preparation phase output, and displays results according to the method chosen. The visualization of results is considered of great help in order to use and interpret results coming from a clustering process. Thus, the third module is the visualization one, which disposes an easily interpretable way of visualizing results.

Índice Geral

Índice de Figuras

Capítulo 1
Apresentação

1 Introdução

Todos os tipos de textos que compõem o dia a dia de empresas e pessoas são produzidos e armazenados em meios eletrônicos. Inúmeras novas páginas contendo textos são lançadas diariamente na Web. Outros tipos de documentos como relatórios de acompanhamento, atas de reuniões, históricos pessoais, etc. são periodicamente gerados e atualizados. Entretanto, até pouco tempo atrás, essas informações em formato de textos não eram usadas para significar algum tipo de vantagem competitiva, ou mesmo como suporte à tomada de decisões, ou ainda como indicador de sucesso ou fracasso. Com o advento do *text mining*, a extração de informações em textos passou a ser possível e o imenso e crescente mundo dos textos está começando a ser explorado.

Em virtude desse crescimento contínuo do volume de dados eletrônicos disponíveis, técnicas de extração de conhecimento automáticas tornam-se cada vez mais necessárias para valorizar a gigantesca quantidade de dados armazenada nos sistemas de informação. Além disso, como as técnicas desenvolvidas para data mining foram desenvolvidas para dados estruturados, técnicas específicas para *text mining* tem sido desenvolvidas para processar uma parte importante da informação disponível que pode ser encontrada na forma de dados não-estruturados.

As aplicações de *text mining* podem fornecer uma nova dimensão das informações disponíveis nas empresas, sendo utilizadas no acompanhamento da gerência de projetos a partir de relatórios de *status*, documentação de projeto e comunicações com o cliente. Uma outra aplicação é o desenvolvimento do planejamento de marketing baseado em detalhes e planos passados, opções de anúncios e pesquisas de marketing. Aplicações não tão pretensiosas já se encontram implantadas atualmente, como a categorização automática de mensagens de correio eletrônico em bancos de investimento; e a extração automática de resumos a partir de documentos pesquisados, realizada por alguns mecanismos de busca na Web.

O *Clustering* de documentos tem sido estudado intensivamente por causa de sua aplicabilidade em áreas tais como information retrieval [61], web mining [60, 27], e análise topológica. Um outro catalisador para o desenvolvimento de um algoritmo de *clustering* de documentos eficiente é a quantidade gigantesca de dados não-estrutrados na internet. A maior parte dessa informação se encontra em formato de textos, por exemplo, emails, notícias, páginas web, relatórios, etc. Organizá-los em uma estrutura

lógica é uma tarefa desafiadora. *Clustering* é empregado, mais recentemente, para percorrer (browsing) coleções de documentos [62] ou organizar os resultados de uma consulta retornados por um mecanismo de busca [63]. Ele também pode servir como um passo de pré-processamento para outros algoritmos de mineração de dados como classificação de documentos [8]. Um objetivo ambicioso do *clustering* de documentos é gerar automaticamente *clusters* de documentos organizados hierarquicamente [5] da mesma forma que a hierarquia de assuntos Yahoo!, por exemplo. A utilização de métodos de *clustering* em vários contextos, pelas mais diferentes disciplinas, refletem sua grande utilidade na exploração de conhecimento sobre dados.

1.1 Motivação

A grande maioria de textos técnicos, científicos, notícias, etc. disponibilizados na web se encontram na língua inglesa. Embora não ocorram com a mesma assiduidade que no idioma inglês, as páginas e documentos disponíveis em português estão se tornando mais e mais freqüentes. No entanto, para tratar os conteúdos desses arquivos, é necessário que pelo menos uma parte da abordagem esteja ligada ao idioma. Além do fato de os dados se apresentarem em formato de texto, o que já exige um trabalho adicional, a língua em que os mesmos se encontram influencia a análise em vários momentos. O objetivo desta tese é a disponibilização de uma solução capaz de agrupar textos em português por similaridade de conteúdo e que permita a visualização da saída de uma forma que facilite a interpretação dos resultados. A parte inicial do trabalho de preparação de dados é dependente do idioma e envolve alguns passos – preparação de listas de *stopwords*, algoritmo de *stemming*, *case folding*, dicionário – que exigem uma infinidade de considerações para a identificação e aglutinação de termos na massa de dados. Ainda assim, quando esta etapa está concluída, é comum que os arquivos em formato texto sejam representados por uma quantidade muito grande de termos. Com o objetivo de promover uma diminuição dessa quantidade de termos, é freqüente a utilização de critérios de redução da representação destes arquivos. O conjunto de termos resultante pode, então, ser convertido para uma forma mais estruturada de dados, passando, por exemplo, à forma de tabelas ou listas invertidas que são facilmente manuseáveis.

A etapa subsequente diz respeito à definição do problema e à forma de abordagem e solução do mesmo. Neste ponto, a expectativa do usuário será traduzida nas atividades que devem ser desempenhadas para que o resultado desejado seja alcançado. Nesta tese, são apresentadas várias tarefas comumente realizadas em dados textuais e tipos de aplicações que as utilizam. O objetivo desta tese em termos de tarefa é o *clustering* (aprendizado não-supervisionado) de dados textuais. Esta abordagem difere das demais por considerar o idioma português e suas particularidades no tratamento dos dados.

Quando o processo de *clustering* é terminado, produz-se uma saída identificando os *clusters* ou agrupamentos encontrados pelo método utilizado. O resultado apresentado, no entanto, nem sempre é facilmente entendido. No caso dos textos, em que cada termo considerado é convertido para um atributo ou dimensão, a visualização da saída pode ser confusa e pouco esclarecedora. Nesta tese, apresenta-se uma forma de visualização que permite ao usuário ver toda a estrutura e organização dos *clusters* encontrados.

Tanto o tratamento de documentos em Português como a visualização detalhada de resultados provenientes de um processo de *clustering* de dados textuais constituem os pontos de maior contribuição científica do presente trabalho.

1.2 Linhas Gerais da Tese

Esta tese é composta de uma primeira parte que mostra uma seqüência teórica de tratamento de dados textuais desde a formação das bases de textos até a interpretação dos resultados, apresentando diferentes aplicações dos dados textuais. A segunda parte desta tese apresenta os módulos que incluem o pré-processamento de dados em Português; o processo de *clustering* propriamente dito, que é disponibilizado em métodos diferentes possibilitando a comparação dos resultados; e a visualização dos resultados produzidos.

1.2.1 Estudo das Etapas do Processo de *Text Mining*

Como introdução à vasta gama de aplicações as quais os dados textuais podem servir, apresenta-se na primeira parte desta tese um estudo descrevendo as etapas do processo de *text mining*, mostrando conceitos que serviram como base, as diferentes

metodologias que podem ser empregadas, aplicações possíveis e interpretação dos resultados.

1.2.2 Ambiente de *Clustering* de Documentos

Em vários momentos do processo de *text mining* o idioma é um diferenciador no tratamento dos dados textuais. A maior parte das ferramentas disponíveis no mercado atualmente não possui abordagem para o idioma Português e dependem de contratação de serviços especializados. Esta tese dedica um tratamento especial ao idioma desenvolvendo toda a preparação de dados – *stop words*, *stemming*, dicionário – para o Português. De todas as possíveis aplicações disponíveis para textos que serão apresentadas na parte de estudo da tese, será abordado o *clustering* de documentos. Será apresentada, também, uma forma elucidativa de visualização dos resultados provenientes do processo de *clustering* da etapa anterior. O objetivo deste trabalho é, então, o desenvolvimento de uma solução de *clustering* de dados textuais em português, realizando todos os passos de preparação de dados, *clustering* de documentos e visualização de resultados.

A segunda parte desta tese enfocará, portanto, a tarefa de *clustering* de um modo mais detalhado e descreverá a solução proposta para *clustering* de documentos em Português.

Capítulo 2

Fundamentação Teórica

2 Etapas do Processo de Text Mining

Text mining, também conhecido como *Text data mining* [7] ou *Knowledge discovery from textual databases* [6], refere-se ao processo de extrair padrões interessantes e não-triviais ou conhecimento a partir de documentos em textos não-estruturados. *Text mining* pode também ser definido como um conjunto de técnicas e processos que se prestam a descobrir conhecimento inovador nos textos. Esta nova tecnologia está sendo empregada atualmente em projetos de diversas áreas, por exemplo, para descobrir fatos na genética e na pesquisa de proteínas.

Nesta parte da tese será apresentado o esquema básico de um processo de *text mining* com os passos que podem compor cada etapa. Esse esquema é mostrado na figura 2.1.

Primeiramente, será mostrado como o pré-processamento dos dados é realizado, de forma a preparar o conjunto de dados textuais para as fases posteriores de execução das tarefas de processamento dos dados e análise de resultados obtidos. Em seguida, será apresentado o conjunto de tarefas que podem ser realizadas a partir dos textos, mostrando em que situações podem ser utilizadas e o que se espera como saída. Depois, apresentam-se formas de avaliação da qualidade dos resultados advindos das etapas anteriores, para que os mesmos possam ser efetivamente empregados. Algumas ferramentas disponíveis atualmente serão também comentadas.

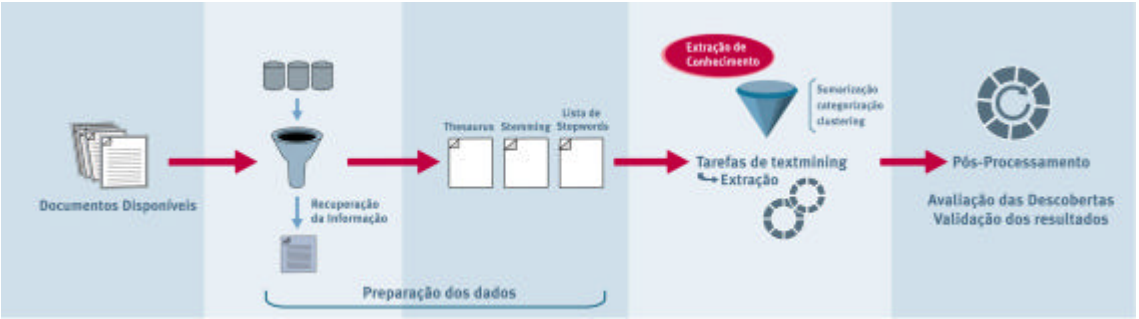


Figura 2.1 Etapas do Text mining

2.1 Preparação dos Dados

O manuseio de arquivos texto apresenta alguns desafios. O primeiro a ser citado envolve o próprio formato dos textos com nenhuma ou pouca estruturação, o que dificulta a utilização imediata de várias técnicas de mineração de dados conhecidas. Um outro desafio diz respeito ao tamanho dos arquivos em formato texto, comumente da ordem de milhares de palavras ou termos. Além disso, muitas dessas palavras são repetidas, expressam o mesmo significado ou são de significado irrelevante. As situações mencionadas acima, assim como outras encontradas quando se lida com dados textuais, devem ser trabalhadas e resolvidas para viabilizar o uso de arquivos texto em primeira instância e em segunda aumentar a eficiência de atividades executadas a posteriori.

A preparação dos textos é a primeira etapa do processo de descoberta de conhecimento em textos. Esta etapa envolve a seleção das bases de textos que constituirão os dados de interesse e o trabalho inicial para tentar selecionar o núcleo que melhor expressa o conteúdo dos textos, ou seja, toda a informação que não refletir nenhuma idéia considerada importante poderá ser desprezada.

Além de promover uma redução dimensional, esta etapa tenta identificar similaridades em função da morfologia ou do significado dos termos, de modo a aglomerar suas contribuições.

2.1.1 Recuperação de Informação

A área de Recuperação de Informação (RI ou *Information Retrieval*) desenvolveu modelos para a representação de grandes coleções de textos que identificam documentos sobre tópicos específicos. Os documentos recuperados são colocados em ordem e apresentados ao usuário. A figura 2.2 apresenta um modelo geral para RI como descrito em [9].

Embora esse seja um campo vasto, neste trabalho o interesse em RI se restringe à representação e identificação de documentos sobre conjuntos de assuntos específicos.

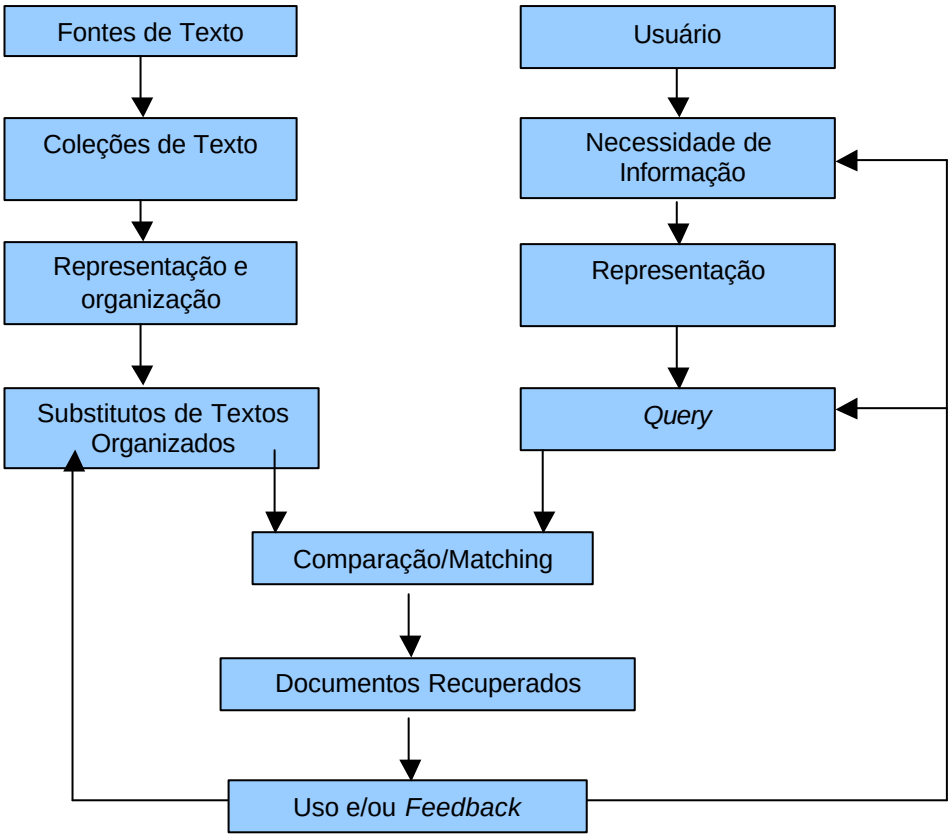


Figura 2.2 Modelo geral de RI

Neste modelo, um usuário com alguma necessidade de informação apresenta uma *query* ou consulta ao sistema RI . A consulta é a representação da necessidade de informação do usuário em uma linguagem entendida pelo sistema. Esta representação é considerada como uma aproximação, devido à dificuldade de representação da necessidade de informação. A consulta é, então, comparada aos documentos, que estão organizados em substitutos dos textos (isto é, palavras-chave, títulos, resumos). Substitutos de textos podem ser vistos como uma representação estruturada dos dados textuais não-estruturados. Assim, eles fornecem uma alternativa de representação dos documentos originais, uma vez que eles tomam bem menos tempo para serem examinados e ao mesmo tempo codificam deixas semânticas suficientes para serem usados em comparação às consultas ao invés dos documentos originais. Como resultado da comparação, um conjunto de documentos seria selecionado e apresentado ao usuário. O usuário pode usar estes documentos ou fornecer ao sistema um retorno (*feedback)* que resulta em modificações na consulta (*feedback* de relevância), na necessidade de

informação original ou nos substitutos [9]. O processo de interação continua até que o usuário esteja satisfeito ou até que o usuário deixe o sistema.

Pode-se considerar que a RI seja o primeiro passo de um processo de mineração de textos. Um sistema RI atua como se fosse um filtro sobre um conjunto de documentos, retornando ao usuário o resultado de um problema ou consulta particular.
Esta seção dá uma visão geral de métodos usados nos sistemas de RI. Na literatura, métodos de RI são categorizados em dois grupos: tradicional e moderno. Uma descrição de alguns métodos de cada categoria é mostrada a seguir.

2.1.1.1 Métodos Tradicionais

São os métodos mais conhecidos e que são comumente utilizados para Recuperação de Informação.

2.1.1.1.1 Coincidência de Cadeias de Palavras

O usuário especifica sua necessidade de informação através de uma cadeia de palavras. Um documento satisfaria a necessidade de informação de um usuário se a cadeia especificada pelo usuário existir no documento. Este método é uma das primeiras e mais simples abordagens. Este método sofre de três problemas [102]:

- Homonímia: o significado de uma palavra depende do contexto no qual ela aparece;
- Sinonímia: palavras tendo o mesmo significado;
- Tempo de resposta ruim.

2.1.1.1.2 Modelo Booleano

Na recuperação Booleana um documento é representado por um conjunto de termos-índice. Uma consulta ou *query* consiste de um conjunto de termos-índice combinados com operadores de Boole. Este método é uma modificação do método 2.1.1.1.1 onde o usuário pode combinar palavras usando operadores booleanos tais como AND, OR e NOT. O modelo é binário, isto é, a freqüência de um termo não tem efeito. Nesse modelo, a semântica da consulta é bem definida – cada documento, ou corresponde à expressão booleana, ou não. Em razão desta semântica descomplicada e do cálculo

direto dos resultados utilizando operações de conjuntos, o modelo booleano é amplamente usado em ferramentas de busca comerciais.

Este método dá ao usuário uma ferramenta para expressar melhor sua necessidade de informação mas ao mesmo tempo requer alguma habilidade por parte do usuário. Neste método, o usuário tem que estar bem familiarizado com as primitivas booleanas especialmente em casos de *queries* ou consultas complexas. Vários mecanismos de busca são baseados neste método – Alta Vista, Lycos e Excite são alguns exemplos [10]. Em [11] encontram-se mais detalhes sobre este modelo.

Os problemas relacionados ao modelo booleano são, no entanto, bem conhecidos:

- a formulação de uma consulta adequada, isto é, a seleção dos termos para a consulta é difícil, especialmente se o domínio não é bem conhecido.
- O tamanho da saída não pode ser controlado. O conjunto resultante tanto pode conter nenhum como milhares de itens. Além disso, sem um grau de comparação parcial, não se pode saber o que foi deixado de fora da definição da consulta.
- Uma vez que não há um grau de comparação, não é possível ordenar os resultados de acordo com a relevância.

2.1.1.1.3 Modelo de Espaço Vetorial

O Modelo de Espaço Vetorial (VSM – Vectorial Space Model) se presta a resolver problemas de representação de documentos utilizando representação geométrica. Esse modelo é utilizado, também, em casos em que é preciso encontrar documentos que atendam a um critério, e a solução desse problema decorre naturalmente do esquema de representação dos documentos.

Documentos são representados como pontos (ou vetores) em um espaço Euclidiano t dimensional onde cada dimensão corresponde a uma palavra (termo) do vocabulário [12]. Cada palavra tem um peso associado para descrever sua significância, que pode ser sua freqüência em um documento, ou uma função dela. A similaridade entre dois documentos é definida ou como a distância entre os pontos ou como o ângulo entre os vetores, desconsiderando o comprimento do documento. O comprimento é desconsiderado para levar em conta documentos de tamanhos diferentes. Assim, cada documento é normalizado de forma que fique com comprimento unitário.

Por causa de sua simplicidade, o modelo do espaço vetorial (VSM) e suas variantes são uma forma bastante comum de representar documentos textuais na mineração de coleções de documentos. Uma explicação para isso é que operações de vetores podem ser executadas muito rapidamente e existem algoritmos padronizados eficientes para realizar a seleção do modelo, a redução da dimensão e visualização de espaços de vetores. Em parte por estas razões, o modelo de espaço vetorial e suas variações tem persistido em avaliações de qualidade, no campo da recuperação de informação [13].

Um problema óbvio com o modelo de espaço vetorial é a dimensionalidade alta: o número de palavras diferentes em uma coleção de documentos facilmente atinge centenas de milhares. Outro problema conhecido é composto por variações de estilo de escrita, erros de grafia, etc.

Além disso, quaisquer duas palavras são consideradas por definição não-relacionadas. Entretanto, é difícil obter uma informação exata das relações semânticas apenas a partir das informações textuais, automaticamente.

Se fosse possível basear um modelo em alguns tipos de variáveis latentes (ou dimensões conceituais) ao invés de palavras, uma representação consideravelmente mais concisa, provavelmente seria obtida [14].

a. Atribuição de Pesos (*Weighting*)

Neste modelo, cada documento é representado como um vetor cujas dimensões são os termos presentes na coleção de documentos inicial a ser minerada. Cada coordenada do vetor é um termo e tem um valor numérico que representa sua relevância para o documento. Normalmente, valores maiores implicam em relevâncias maiores. Este processo de associar valores numéricos a coordenadas de vetor é referenciado como atribuição de pesos ou *weighting*. Formalmente falando, *weighting* é o processo de dar ênfase aos termos mais importantes. Existem várias medidas de atribuição de pesos (*weighting*) entre as quais podemos citar três mais populares que são: Binária, TF e TF*IDF. O esquema binário usa os valores 1 e 0 para revelar se um termo existe em um documento ou não, respectivamente. A frequência do termo (Term Frequency - TF) conta as ocorrências de um termo em um documento e usa este contador como uma medida numérica. Normalmente, as medidas são normalizadas para valores no intervalo [0,1]. Isto é feito independentemente para cada documento, dividindo-se cada medida de coordenada pela medida de coordenada mais alta do documento considerado. Este

procedimento ajuda a resolver problemas asssociados com o tamanho (comprimento) do documento. Sem a normalização, um termo pode ter uma medida maior num certo vetor-documento simplesmente porque o documento correspondente é muito grande. O terceiro esquema é o TF*IDF (Term Frequency – Inverse Document Frequency) onde se multiplica a medida de coordenada oriunda de um esquema TF por seu peso global. A medida total de um termo se torna a combinação de sua medida local (medida TF) e global (medida IDF). A medida IDF para o termo t é definida como $log(N/N_t)$ onde N é o número total de documentos e N_t é o número total de documentos contendo t. A IDF aumenta conforme a singularidade do termo entre os documentos aumenta – isto é conforme sua existência diminui – dando assim ao termo um peso maior. Termos que ocorrem muito num certo documento (contagem local alta) e termos que ocorrem em poucos documentos (contagem global alta) são ditos como tendo alto poder de decisão e recebem pesos altos. O peso de um termo em um documento é uma combinação de suas contagens local e global. Pelas mesmas razões previamente expostas, a normalização é, também, usualmente usada neste processo. Para normalizar medidas baseadas no esquema TF*IDF, a normalização do cosseno [124] é usada. Ela é calculada como a seguir:

$$Nt_k d_j = \frac{TF * IDF(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} (TF * IDF(t_s, d_j))(TF * IDF(t_s, d_j))}} \qquad \text{Equação 2.1}$$

onde t_k e d_j são o termo e o documento em consideração, respectivamente, $TF*IDF(t_s,d_j)$ é a medida da coordenada de t_s em d_j, e $|T|$ é o número total de termos no espaço de termos.

b. Medida do Cosseno

O Modelo do Espaço Vetorial ([15]e [16]) usa as estatísticas dos termos para comparar as necessidades de informação e os documentos. Cada documento é representado como um vetor de n dimensões onde cada dimensão é um termo proveniente do conjunto de termos usados para identificar o conteúdo de todos os documentos e consultas. A cada termo é então dado um peso dependendo da sua importância em revelar o conteúdo de seu documento associado ou consulta. Freqüências de termos podem ser binárias (simples e eficientes) ou números (mais exatas). Um esquema típico usado para associar

pesos a termos é o TF*IDF explicado previamente. Para comparar um documento dado com alguma consulta ou mesmo com outro documento (ambos representados no formato de vetor), o cosseno do ângulo [10] entre dois vetores é medido como:

$$Cos(I, D) = I.D / (\|I\| \|D\|) \qquad \text{Equação 2.2}$$

onde *I* é ou o vetor de consulta , *D* é um vetor documento, *I.D* é o produto escalar de *I* e *D*, e ||*I*|| é a raiz quadrada do produto escalar do vetor *I* por ele mesmo.

Este método tem as vantagens de requerer uma intervenção de usuário mínima e ser robusto; entretanto, ele ainda sofre de efeitos de homonímia e sinonímia nos termos.

2.1.1.1.4 Uso do Thesaurus

Para resolver problemas de vocabulário (homonímia e sinonímia), alguns sistemas integraram um thesaurus no seu processo. Um thesaurus é um conjunto de termos (palavras ou frases) com relações entre elas. A finalidade do thesaurus é aplicar ou substituições palavra a palavra – substituir todos os termos que tem relação no thesaurus (isto é, tem o mesmo significado) pelo mesmo termo – ou substituições de hierarquias de conceitos tais como substituições de generalização [18] onde todos os termos são generalizados para os termos adequados de mais alto nível de acordo com a hierarquia de conceitos descritas no thesaurus. Embora o uso do thesaurus tenha ajudado com problemas de sinonímia, este método ainda sofre de homonímia. O Thesaurus será melhor explicado na seção 2.2.4.

2.1.1.1.5 Método do arquivo de assinatura

Neste método, cada documento é representado por um cadeia de *bits* de tamanho fixo, que é referida como a assinatura do documento. Para conseguir esta assinatura, uma técnica de *hashing* e uma técnica de superimposição de código são aplicadas nas palavras dos documentos [17]. As assinaturas de todos os documentos são armazenadas seqüencialmente em um arquivo – o arquivo de assinaturas – que é muito menor que o conjunto original de documentos. Comparações entre consultas e documentos são feitas através do arquivo de assinatura. Este método tem as vantagens de ser simples e de ser capaz de tolerar erros de tipos e grafia. Alguns algoritmos usando estes métodos são apresentados em [20] e [21]. Veja em [19] o capítulo 9.5 para mais detalhes.

2.1.1.1.6 Inversão

Cada documento é representado por um conjunto de palavras chave que descrevem seu conteúdo. Estas palavras-chave são armazenadas (normalmente em ordem alfabética) em um arquivo-índice. Um outro arquivo – arquivo de endereçamento – é criado para guardar o conjunto de apontadores de cada palavra-chave no arquivo-índice e o conjunto de ponteiros nos arquivos de endereçamentos são utilizados para comparar com as consultas. Este método sofre de sobrecarga de armazenamento (até 300% do tamanho original do arquivo [22]) e o custo de manter o índice atualizado num ambiente dinâmico – assim, é mais adequado para sistemas RI . Por outro lado, é fácil de implementar, rápido e suporta sinônimos. Este método tem sido adotado em vários sistemas comerciais. Confira [16] para detalhes.

2.1.1.2 Métodos Modernos

Os métodos descritos até este ponto usam apenas uma porção limitada da informação associada a um documento. Em alguns casos, isto não é suficiente. Os métodos apresentados a seguir são considerados alguns dos mais recentes empregados na filtragem de informação.

2.1.1.2.1 Indexação Semântica Latente (LSI)

O modelo de recuperação de informação com indexação semântica latente é construído sobre a pesquisa anterior em recuperação de informação e, usando a decomposição de valor singular (DVS) [64] para reduzir as dimensões do espaço termo-documento, tenta resolver problemas de sinonímia e polissemia (uma palavra que representa mais de um significado) que são o grande problema dos sistemas de recuperação de informação automáticos. LSI representa explicitamente termos e documentos em um espaço rico e de dimensionalidade alta, permitindo que relacionamentos semânticos subentendidos (“latentes”) entre termos e documentos sejam explorados durante a procura. LSI depende dos termos que constituem um documento para sugerir o conteúdo semântico do documento. Entretanto, o modelo LSI vê os termos em um documento como indicadores não muito confiáveis de conceitos contidos no documento. Ele assume que a variabilidade de escolha das palavras torna parcialmente obscura a estrutura semântica do documento. Pela redução da dimensionalidade do espaço termo-documento, os

relacionamentos semânticos subentendidos entre documentos são revelados, e muito do "ruído" (diferenças no uso das palavras, termos que não ajudam a distinguir documentos, etc.) é eliminado. LSI estatisticamente analisa os padrões de uso das palavras pela coleção de documentos inteira, colocando documentos com padrões de uso de palavras similares próximos uns dos outros no espaço termo-documento e permitindo que documentos semanticamente relacionados estejam próximos uns aos outros, mesmo que eles possam não compartilhar termos [65].

LSI difere de tentativas anteriores de usar modelos de espaço reduzido para recuperação de informação de várias maneiras. A mais notada é que LSI representa documentos em um espaço de dimensionalidade alta. No exemplo apresentado em [66] foram usadas apenas sete dimensões para representar o espaço semântico. Em segundo, ambos termos e documentos são explicitamente representados no mesmo espaço. Em terceiro, diferentemente de Borko e Bernick [67], nenhuma tentativa de interpretar o significado de cada dimensão é feita. Cada dimensão é simplesmente assumida para representar um ou mais relacionamentos semânticos no espaço termo-documento. Finalmente, por causa dos limites impostos, na maior parte, por exigências computacionais de abordagens de recuperação de informação do tipo espaço-vetor, tentativas anteriores se concentraram em coleções de documentos relativamente pequenas. LSI é capaz de representar e manipular grandes conjuntos de dados, tornando-se viável para aplicações do mundo real [68].

Comparada a outras técnicas de recuperação de informação, LSI tem um desempenho surpreendemente bom. Em um teste, Dumais [65] reportou que LSI fornece 30% a mais de documentos relacionados que as técnicas de recuperação padrão baseadas em palavras quando procurando na coleção MED. Em cinco coleções de documentos aproximadamente, o mesmo estudo indicou que LSI teve um desempenho 20% melhor que técnicas de recuperação léxicas. Além disso, LSI é totalmente automática e fácil de usar, não requerendo expressões complexas ou sintaxe para representar a query. Uma vez que termos e documentos são representados explicitamente no espaço, *feedback* de relevância [68] pode ser integrado com o modelo LSI sem problemas, fornecendo um desempenho geral ainda melhor.

2.1.1.2.2 Método Conexionista

Este método usa uma rede neural onde cada nó representa uma palavra-chave do conjunto de documentos. Para efetuar uma consulta num conjunto de documentos, os níveis de atividade dos nós de entrada são passados para um nível específico. Esta atividade é então propagada para outra(s) camada(s) eventualmente chegando até a camada mais externa na qual são apontados os documentos que atendem. No caso da rede ser treinada adequadamente, este método mostra bons resultados revelando sua habilidade para manusear semântica (extrair radicais de palavras, e sinônimos) adequadamente. Uma descrição mais detalhada é apresentada em [23].

2.1.2 Filtragem de Informação (*Information Filtering*) ou FI

Sistemas de Filtragem de Informação lidam com grandes seqüências de documentos novos, normalmente distribuídos a partir de fontes remotas. Algumas vezes são referidos como roteadores de documentos (*document routing*). O sistema mantém os perfis de usuários que descrevem seus interesses a longo prazo. O perfil deve descrever o que o usuário gosta ou não. Os documentos novos que não casam com o perfil do usuário são removidos das seqüências que chegam. Como resultado, o usuário só vê o que é deixado na sequência depois que os documentos descasados foram removidos – um filtro de e-mails por exemplo remove "lixo" dos e-mails. A figura 2.3 mostra um modelo geral para FI [9].

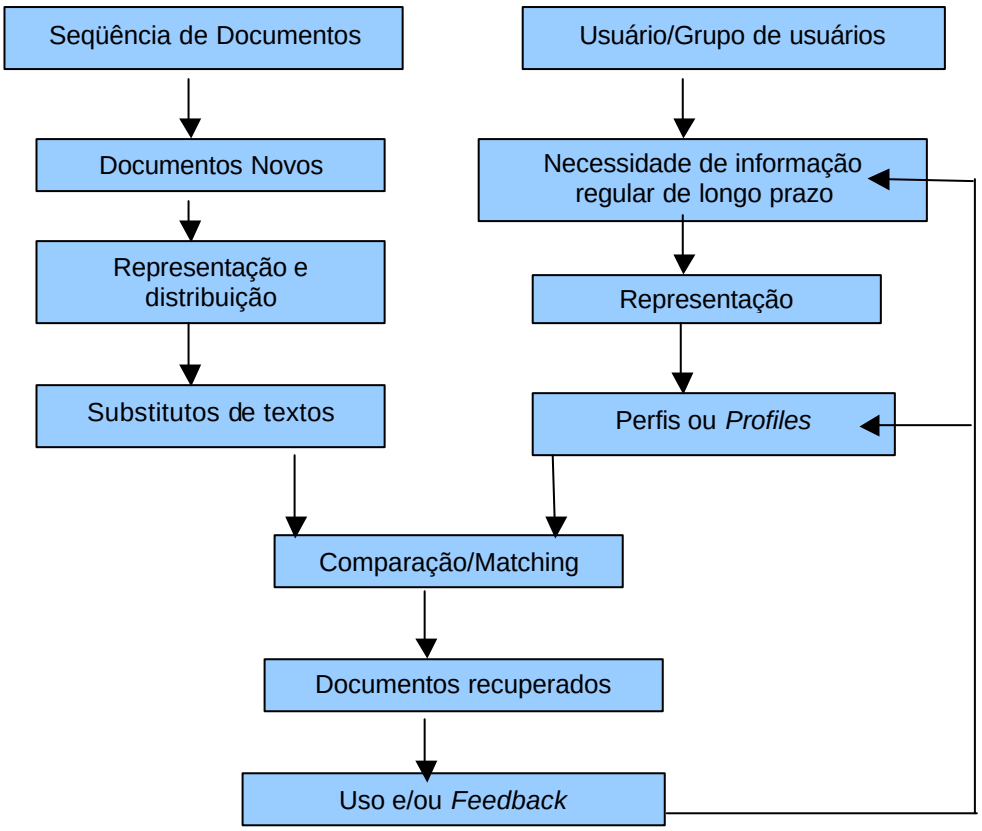


Figura 2.3 Modelo geral de FI

O primeiro passo no uso de um sistema FI é criar um perfil ou profile. Um perfil representa as necessidades de informação de um usuário ou grupo de usuários, que se percebe estável por um longo período de tempo. Sempre que um novo documento é recebido na seqüência de dados, o sistema o representa com o substituto do texto e o compara com cada perfil armazenado no sistema. Se o documento coincide com o perfil, ele será roteado para o usuário correspondente. O usuário pode então usar os documentos recebidos e/ou fornecer um retorno (*feedback*). O retorno fornecido pode levar a modificações no perfil e/ou necessidade de informação.

Os métodos descritos para RI são também utilizados para FI. A maior parte desses métodos foi inicialmente desenvolvida visando RI . Com o advento da FI, os métodos de RI foram adaptados para se adequar às necessidades da FI. Além disso, mais pesquisa tem surgido nesta área e isto resultou no desenvolvimento de mais métodos novos, que são correntemente usados em ambas as áreas.

2.1.3 Recuperação de Informação e Filtragem de Informação:

Recuperação de Informação (RI) e Filtragem de Informação (FI) [9, 10, 17] são dois processos que tem objetivos subentendidos equivalentes. Eles lidam com o problema de procura de informação. Dada alguma necessidade de informação apresentada pelo usuário de uma forma adequada, eles tentam devolver um conjunto de documentos que satisfaçam a necessidade.

A necessidade de informação é representada via consultas ou *queries* em sistemas de RI e via perfis ou *profiles* em sistemas FI. Nosso estudo destes processos focará apenas em objetos como documentos que podem ser semi-estruturados, por exemplo e-mails, ou não-estruturados – embora isto possa ser aplicado, por exemplo, a objetos de multimídia também. Na literatura, RI é visto como um antecessor de FI. A razão para isto é que RI é mais antigo e FI baseia vários de seus fundamentos em RI . Muito da pesquisa feita para RI tem sido usada e/ou adaptada para se adequar a FI. Apesar do fato que estes dois processos são muito similares do ponto de vista de fundamentos, existem várias diferenças entre os dois que os tornam dois tópicos separados. Uma comparação entre estes tópicos é delineada a seguir:

- Sistemas RI são desenvolvidos para tolerar algumas inadequações na representação da consulta ou *query* da necessidade da informação enquanto sistemas FI assumem que perfis ou *profiles* são acurados.
- Sistemas RI são normalmente usados uma vez por um usuário único (por um usuário de uma só *query* ou consulta [9]) enquanto sistemas FI são repetidamente usados pelo mesmo usuário com algum *profile*.
- Sistemas RI são desenvolvidos para servir aos usuários com necessidade de curto prazo enquanto sistemas FI servem a usuários cujas necessidades são relativamente estáticas por um longo período de tempo.
- Sistemas RI normalmente operam em coleções estáticas de documentos enquanto sistemas FI lidam com dados dinâmicos de seqüências de documentos.
- O primeiro objetivo de RI é coletar e organizar (ordenar de acordo com a importância) um conjunto de documentos que coincide com uma dada consulta, enquanto o objetivo primário da FI é distribuir os novos documentos recebidos aos usuários com perfis coincidentes.

- Em FI, o fato de um documento chegar no tempo (ou ocasião) adequado é de grande importância enquanto em RI isto não é tão importante.

A comparação acima entre FI e RI realça as principais diferenças entre os dois em relação a objetivos, uso, usuários e os tipos de dados que eles operam (estático vs. dinâmico).

De forma a prover aos usuários com a informação requisitada, tanto sistemas RI quanto FI devem representar a necessidade de informação consulta ou *query* e perfil ou *profile* respectivamente) e o conjunto de documentos de uma maneira adequada à comparação e combinação. Algumas representações correntemente usadas são: a representação vetorial, redes neurais e redes semânticas [10]. O mecanismo de comparação usado para combinar a necessidade de informação e de documentos depende da técnica de representação utilizada. Para melhorar a acurácia dos sistemas RI e FI, um mecanismo de *feedback* de relevância é utilizado. Uma vez que é apresentado ao usuário um conjunto de documentos retornado por um sistema RI ou FI, as medidas de precisão ou *recall* são calculadas. Precisão é o percentual de documentos recuperados que são relevantes à consulta ou ao perfil. Ela é calculada como o número de itens relevantes recuperados divididos pelo número total de itens recuperados. O *recall* mede o percentual de itens relevantes recuperados em relação ao número total de itens relevantes no banco de dados. A grosso modo, estas duas medidas têm uma relação inversa. Conforme o número de resultados retornados aumenta, a probabilidade de retornar respostas erradas também aumenta.

2.2 Análise dos Dados

O objetivo principal da análise dos dados é facilitar a identificação de similaridades de significado entre as palavras, apesar de suas variações morfológicas. Essa situação ocorre, por exemplo, pela variação de um mesmo termo assumindo diferentes sufixos, que é o caso do *stemming*. Uma outra situação contemplada pela análise dos dados é o caso de palavras sinônimas: apesar de serem morfologicamente diferentes expressam a mesma idéia. As diferentes implementações não necessariamente consideram todos os passos apresentados a seguir para a análise de dados.

2.2.1 Case Folding

Case Folding é o processo de converter todos os caracteres de um documento no mesmo tipo de letra – ou todas maiúsculas ou minúsculas. Isso tem a vantagem de acelerar comparações no processo de indexação.

2.2.2 Stop words

Um dos primeiros passos no processo de preparação dos dados é a identificação do que pode ser desconsiderado nos passos posteriores do processamento dos dados. É a tentativa de retirar tudo que não constitui conhecimento nos textos. Nesta etapa, uma lista contendo palavras a serem descartadas é formada. Este conjunto de palavras é chamado de *Stop words* (conhecido ainda como *Stoplist*).

Stop words são palavras que não tem conteúdo semântico significante no contexto em que ela existe e são palavras consideradas não relevantes na análise de textos. Normalmente isso acontece por se tratarem de palavras auxiliares ou conectivas (e, para, a, eles) e que não fornecem nenhuma informação discriminativa na expressão do conteúdo dos textos. Na construção de uma lista de *stop words* incluem-se palavras como preposições, pronomes, artigos e outras classes de palavras auxiliares.

Stop words também podem ser palavras que apresentam uma incidência muito alta em uma coleção de documentos. *Stop words* geralmente não são incluídas como termos indexados.

2.2.3 Stemming

O processo de *stemming* é realizado considerando cada palavra isoladamente e tentando reduzí-la a sua provável palavra raiz. Isto tem a vantagem de eliminar sufixos, indicando formas verbais e/ou plurais; entretanto, algoritmos de *stemming* empregam linguística e são dependentes do idioma.

Os algoritmos de *stemming* correntes não costumam usar informações do contexto para determinar o sentido correto de cada palavra, e realmente essa abordagem parece não ajudar muito. Casos em que o contexto melhora o processo de *stemming* não são muito freqüentes, e a maioria das palavras pode ser considerada como apresentando um

significado único. Os erros resultantes de uma análise de sentido imprecisa das palavras, em geral, não compensam os ganhos que possam ser obtidos pelo aumento de precisão do processo de *stemming*.

Existem, porém, outros tipos de erros que devem ser observados e controlados durante a execução do *stemming*. Os erros mais comuns associados ao processo de *stemming* podem ser divididos em dois grupos:

- *Overstemming*: acontece quando a cadeia de caracteres removida não era um sufixo, mas parte do *stem*. Isto pode resultar na conflação de termos não-relacionados.
- *Understemming*: acontece quando um sufixo não é removido. Isto geralmente causa uma falha na conflação de palavras relacionadas.

Alguns métodos de *stemming* são apresentados a seguir, com o objetivo de mostrar as diferentes abordagens utilizadas pelos algoritmos existentes. Estes métodos foram desenvolvidos para a língua inglesa, embora sejam encontradas adaptações de alguns deles para diversos idiomas.

2.2.3.1 Método do *Stemmer* S

Um método de *stemming* simples é o *stemmer* S [24], no qual apenas uns poucos finais de palavras da língua inglesa são removidos: “ies”, “es”, e “s” (com exceções). Embora o *stemmer* S não descubra muitas variações, alguns sistemas práticos o usam pois ele é conservador e raramente surpreende o usuário negativamente.

2.2.3.2 Método de Porter

O processo de *stemming* de Porter [25] consiste da identificação das diferentes inflexões referentes à mesma palavra e sua substituição por um mesmo *stem*. A idéia é conseguir agregar a importância de um termo pela identificação de suas possíveis variações.

Termos com um *stem* comum usualmente têm significados similares, por exemplo:

CONSIDERAR
CONSIDERADO
CONSIDERAÇÃO
CONSIDERAÇÕES

Um outro ponto importante é o aumento de desempenho de um sistema quando ocorre a substituição de grupos de termos por seu *stem*. Isto acontece por causa da remoção dos diferentes sufixos, -AR, -ADO, -AÇÃO, -AÇÕES, deixando apenas o radical comum CONSIDER.
O algoritmo de Porter remove 60 sufixos diferentes em uma abordagem multi-fásica. Cada fase remove sucessivamente sufixos e promove alguma transformação no *stem*.

2.2.3.3 Método de Lovins

O método de Lovins [26] é um método de um único passo, sensível ao contexto e que usa um algoritmo da combinação mais longa para remover cerca de 250 sufixos diferentes. Este método remove no máximo um sufixo por palavra, retirando o sufixo mais longo conectado à palavra. O algoritmo foi desenvolvido a partir de um conjunto exemplo de palavras na língua inglesa, usadas para formar a lista de regras de Lovins. Vários sufixos no entanto não foram contemplados por esta lista. Mesmo assim, é o mais agressivo dos três algoritmos de *stemming*.

2.2.4 Uso do Dicionário ou *Thesaurus*

Um dicionário pode ser definido como um vocabulário controlado que representa sinônimos, hierarquias e relacionamentos associativos entre termos para ajudar os usuários a encontrar a informação de que eles precisam. Embora isto pareça complexo, é apenas uma questão de se entender para que um thesaurus é desenvolvido. O valor do thesaurus vem justamente dos problemas inerentes à procura e indexação da linguagem natural. Usuários diferentes definem a mesma query usando termos diferentes. Para resolver este problema, um thesaurus mapeia termos variantes –sinônimos, abreviações, acrônimos, e ortografias alternativas – para um termo preferido único para cada conceito. Para processos de indexação de documentos, o thesaurus informa que termos

índices devem ser usados para descrever cada conceito. Isto reforça a consistência da indexação.

Um thesaurus pode também representar a riqueza dos relacionamentos associativos e hierárquicos [1]. Usuários podem expressar a necessidade de informação com um nível de especificidade mais restrito ou mais amplo que o usado pelo indexador para descrever os documentos. O mapeamento de relacionamentos hierárquicos endereçam este problema.

2.2.4.1 Termos Compostos

Além de considerar termos simples, existem Thesaurus que consideram a utilização de termos compostos nos casos de palavras que aparecem sempre juntas. O uso de descritores consistindo de mais de uma palavra são aceitáveis, no entanto, somente se o termo composto expressa um conceito único expresso pela associação dos termos considerados. São palavras que quando se reúnem apresentam um significado diferente que cada uma delas tem separadamente.

2.2.4.2 Relacionamentos entre termos

Relacionamentos como os mostrados a seguir podem ser encontrados em um thesaurus:

- Relacionamento de Equivalência
- Relacionamento de Hierarquia
- Relacionamento de Associação

Relacionamento	Indicador
Equivalência	Sinônimos
Hierarquia	Termo Amplo
	Termo Restrito
Associação	Termo Relacionado

Tabela 2.1. Tabela de Relacionamentos de um Thesaurus

O relacionamento hierárquico é a primeira característica que distingue um thesaurus de uma lista de termos não estruturada, como um glossário. Ele é baseado em graus ou níveis de super ou subordenação. O descritor superordenado representa uma classe ou um todo; o descritor subordenado refere-se aos membros ou partes de uma classe. No thesaurus, relacionamentos hierárquicos são expressados pelas seguintes notações:

Termo Amplo (Broader Term) = rótulo para o descritor superordenado
Termo Restrito (Narrower Term) = rótulo para o descritor subordenado

O relacionamento hierárquico cobre três situações logicamente diferentes e mutuamente exclusivas: a) o relacionamento genérico, b) o relacionamento modelo, c) o relacionamento todo-parte. Cada descritor subordenado deve se referir ao mesmo tipo de conceito que o seu descritor superordenado, isto é, ambos os termos amplo e restrito devem representar um objeto, uma ação, uma propriedade, etc.

O relacionamento genérico identifica a ligação entre a classe e seus membros ou espécies. Neste tipo de relacionamento a seguinte afirmação sempre pode ser aplicada: "[termo restrito] é um [termo amplo]".

O relacionamento modelo identifica a ligação entre uma categoria genérica de coisas ou eventos, expressos por um nome comum, e um modelo individual daquela categoria, freqüentemente um nome próprio.

O relacionamento todo-parte cobre situações nas quais um conceito é incluído por herança em outro, independentemente do contexto, de forma que os descritores podem ser organizados em hierarquias lógicas, com o todo sendo tratado como um termo amplo. Alguns exemplos seriam orgãos do corpo, sistemas ou pontos geográficos.

2.2.5 Transformação dos Dados

Uma das formas mais comuns de utilização dos dados provenientes dos textos é a sua conversão em tabelas. Este formato permite a aplicação de diversas técnicas desenvolvidas para dados estruturados.

2.2.5.1 Conversão em Tabelas

Após a eliminação da lista de *stopwords*, utilização do *Thesaurus* e execução do processo de *Stemming*, obtém-se um conjunto de dados reduzido em relação ao original, formado pelos termos restantes e que podem, então, ser analisados. Este tipo de representação é conhecido como *bag of words* e pode ser facilmente convertido em tabelas. Embora, esse processo de conversão seja praticamente imediato, a dimensionalidade do conjunto resultante de atributos é freqüentemente bastante alta, pois cada termo que não foi eliminado ou aglutinado, é transformado em um atributo.

Uma coleção de documentos é representada da seguinte forma: cada célula expressa a relação termo-documento do tipo $term_k$-d_j. Esta relação é dada por um peso a_{jk},como mostrado na tabela 2.2.

	$Term_1$	...	...	$Term_k$
d_1	a_{11}	...	...	a_{1k}
...	...	...	...	...
d_j	a_{j1}	...	...	a_{jk}

Tabela 2.2. Tabela de conversão dos termos em atributos

2.3 Processamento dos Dados – Tarefas de Text Mining

A maior parte dos problemas com o manuseio de textos está relacionada à busca da representação adequada, ou um modelo para os dados disponíveis usando os recursos existentes com um tempo limitado, de forma que o desempenho subseqüente do modelo atenda aos critérios de qualidade e eficiência.

Na etapa de processamento dos dados, os objetivos do processo de *text mining* devem ser definidos para que as tarefas cabíveis possam ser executadas. Existem várias tarefas que podem fazer parte do processo de extração de conhecimento em textos.

Cada tipo de tarefa extrai um tipo diferente de informação dos textos. O Processo de Agrupamento ou *Clustering* torna explícito o relacionamento entre documentos,

enquanto a categorização identifica os tópicos-chave de um documento. A extração de características é usada quando se precisa conhecer pessoas, lugares, organizações e objetos mencionados no texto. A sumarização estende o princípio de extração de características concentrando-se mais em sentenças inteiras que em nomes ou frases. A Indexação temática é útil quando se quer ser capaz de trabalhar preferencialmente com tópicos que com palavras-chave. Algumas tarefas utilizadas usualmente pelo *Text mining* são descritas a seguir.

2.3.1 Indexação

A indexação permite que se procure eficientemente em textos por documentos relevantes a uma query sem precisar examinar os documentos inteiros. Nesta forma, a indexação de textos é similar às indexações de bancos de dados convencionais, que permitem que se evite escanear tabelas inteiras para a recuperação eficiente de linhas de dados. Os tipos mais comuns de indexação são a indexação do texto completo e a indexação temática. Existem ainda a indexação tradicional, a indexação por tags, a indexação semântica latente e a indexação por listas invertidas.

2.3.1.1 Indexação do Texto Completo

A indexação do texto completo ocorre automaticamente em várias ferramentas de análise de textos quando os documentos são carregados. Índices normalmente guardam informação sobre a localização dos termos dentro do texto, de forma que operadores de proximidade, assim como operadores booleanos possam ser utilizados em queries no texto completo. Os operadores mais comuns em queries de texto são:

- Operadores Booleanos: and, or, not
- Operadores de Proximidade: near, within

A indexação suporta operadores booleanos, permitindo que estas operações sejam executadas nos índices sem a procura no documento completo. Assim, os operadores podem ser utilizados de forma rápida e eficiente mesmo para grandes coleções de textos. Uma vez que os índices mantém informações sobre a posição das palavras dentro de um texto, operadores de proximidade podem também fazer uso desses índices.

2.3.1.2 Indexação Temática

A indexação temática depende do uso do dicionário. O Thesaurus é um conjunto de termos que define um vocabulário e é montado usando relacionamentos. Ele fornece uma estrutura hierárquica que permite às ferramentas de *text mining* encontrar rapidamente generalizações assim com especializações de termos específicos como mostrado na figura 2.4.

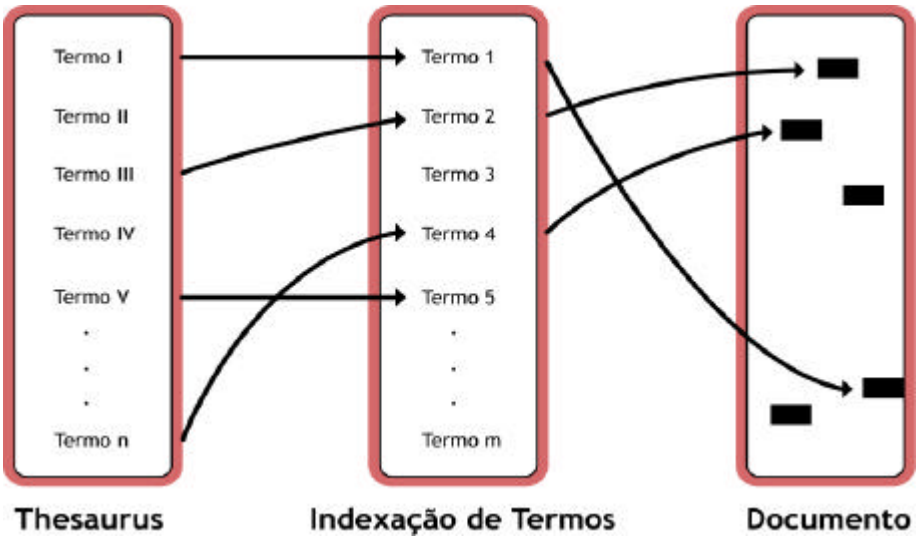


Figura 2.4 Utilização do thesaurus na indexação temática

A estrutura de thesaurus mais comumente utilizada para indexação temática consiste de quatro componentes principais:

- Thesaurus
- Termo indexador
- Termo preferido
- Termo não-preferido

Termos indexadores ou são uma palavra simples ou um termo composto representando um conceito no thesaurus. Termos preferidos são os termos usados quando se indexa conceitos. Termos preferidos são organizados hierarquicamente. Os termos não-

preferidos são ligados à estrutura hierárquica por sua referência ao termo preferido. Termos preferidos estão relacionados uns aos outros por relações que definem a hierarquia.

2.3.1.3 Indexação Semântica Latente

Procuras regulares por palavras-chave aproximam uma coleção de documentos da seguinte forma: um documento contém uma palavra dada ou não, sem meio termo. Cria-se um conjunto de resultados olhando em cada documento procurando por certas palavras e frases, descartando quaisquer documentos que não os contenha, e ordenando o resto baseado em algum sistema de ordenação. Cada documento é julgado sozinho pelo algoritmo de procura – não existe interdependência de qualquer tipo entre documentos, que são avaliados apenas por seus conteúdos.

Relacionamentos entre palavras podem ser deduzidos de seus padrões de ocorrências nos documentos. Esta noção é utilizada em um método chamado Indexação Semântica Latente (LSI) [54] que aplica a decomposição de valor singular (DVS) à matriz documento-por-palavra para obter uma projeção de ambos, documentos e palavras, num espaço referenciado como o espaço latente. A indexação semântica latente adiciona um passo importante ao processo de indexação de documentos. Além de registrar que palavras-chave um documento contém, o método examina a coleção de documentos como um todo, para ver que outros documentos contém algumas daquelas mesmas palavras. LSI considera documentos que tem as mesmas palavras em comum como sendo semanticamente próximos, aqueles com poucas palavras em comum como sendo semanticamente distantes. Este método simples se correlaciona surpreendentemente bem com a forma com que o ser humano, olhando para o conteúdo, poderia classificar uma coleção de documentos. Embora o algoritmo LSI não entenda nada sobre o que as palavras significam, os padrões que ele percebe podem fazer com que ele pareça espantosamente inteligente. Quando se procura uma base de dados indexada-LSI, o mecanismo de busca verifica os valores de similaridade que ele calculou para cada palavra do conteúdo, e retorna os documentos que ele "pensa" que melhor se adequam à consulta. Dois documentos podem ser semanticamente muito próximos mesmo que eles não compartilhem uma palavra específica, porque LSI não requer uma comparação para retornar resultados úteis. Em situações em que uma procura simples de palavra falhará

se não houver uma correspondência perfeita, LSI freqüentemente retornará documentos relevantes que não contém a palavra indicada.

Os algoritmos originais LSI tem uma complexidade computacional alta, $O(N^3)$, o que é problemático para o uso com grandes conjuntos de dados. A complexidade computacional do LSI é conhecida como $J(nld)$, onde n é o número de documentos, l é o número médio de palavras diferentes em cada documento, e d é a dimensionalidade resultante.

2.3.1.4 Indexação por Tags

Na indexação por tags, algumas partes do texto são selecionadas automaticamente para fazer parte do índice. Para a indexação por tags, normalmente adota-se o uso de gramáticas *parsers* e expressões regulares para a definição e reconhecimento das *tags*. As palavras-chave são extraídas com base nestas *tags*.

2.3.1.5 Indexação por Listas ou Arquivos Invertidos

Uma outra técnica bem conhecida e utilizada para indexar arquivos em forma de texto são as listas ou arquivos invertidos [28,29]. Um arquivo invertido contém, para cada termo que aparece no banco de dados, uma lista contendo os números dos documentos contendo aquele termo. Para processar uma consulta ou query, um vocabulário é usado para mapear cada termo da query para o endereço da lista invertida; as listas invertidas são lidas a partir do disco; e as listas são mescladas, considerando a interseção dos conjuntos de números de documentos em operações AND, a união em operações OR, e o complemento em operações NOT. As listas invertidas de maneira geral apresentam bom desempenho para procuras de palavras-chave únicas (logarítmico em relação ao tamanho do banco de dados, o que realmente significa poucos acessos a disco por procura) mas seu desempenho rapidamente degrada quando o tamanho da consulta aumenta. As consultas podem conter vários termos indexados.

Algumas melhorias foram acrescentadas ao método das listas invertidas. São as chamadas listas invertidas comprimidas [31,32,33,34] que aumentaram o desempenho significativamente em termos de requisitos de armazenagem em relação às listas originais.

2.3.2 Extração de Informação

A Extração de Informação (EI) é uma área de pesquisa que conjuga *text mining* e processamento de linguagem natural (PLN). EI é o processo de extrair informação pré-definida sobre objetos e relacionamentos entre eles a partir de documentos textuais. Normalmente esta informação é armazenada em modelos. Em geral, os dois objetivos da EI são:

- dividir um documento em partes relevantes e irrelevantes, e
- preencher modelos pré-definidos com informação extraída.

2.3.2.1 Aplicações de Extração de Informação

Tarefas de EI simples, como extrair nomes próprios e de companhias a partir de textos, podem ser executadas com alta precisão, diferentemente de tarefas mais complexas, como determinar a sequência de eventos a partir de um documento. Em tais tarefas complexas, geralmente, sistemas de EI são definidos em um domínio muito restrito e transformar o sistema de um domínio para outro exige muito trabalho e requer suporte de especialistas. Sistemas de EI escaneiam uma coleção de documentos de forma a transformá-la em pedaços bem menores de informações relevantes extraídas que são fáceis de serem apreendidas.
Algumas aplicações de EI comuns incluem:

- Preencher modelos (*templates*) com informação extraída. Esses modelos são então guardados em ambientes estruturados tais como bancos de dados para recuperação de informação rápida mais tarde. Referir-se a [40] para alguns exemplos de sistemas de peenchimento de modelos.
- Responder questões pré-definidas como “Onde fica o Líbano?” Esta é uma variação de preenchimento do modelo mas ainda é uma área não muito avançada, de forma que não se pode responder questões tais como “Que país teve a inflação mais baixa em 1999?”; entretanto, alguns sistemas poderiam ser capazes de apontar ao usuário alguns documentos que discutam a questão considerada. O sistema BORIS (1983) é

um sistema EI pergunta-resposta que tenta entender pequenos documentos específicos de um domínio e responder as questões sobre eles.

- A sumarização de documentos em extrações que são resumos feitos por máquina são totalmente diferentes de resumos feitos pelo homem. Normalmente, extrai-se um grupo de sentenças altamente relevantes e as apresenta como um resumo. Alguns sistemas para esta finalidade são descritos em [40].
- Extrair os termos mais importantes a partir de um conjunto de documentos de forma que estes termos extraídos possam ser utilizados para a indexação eficiente de documentos como o oposto a usar todos os termos dos documentos para indexá-los. Isto é especialmente útil em sistemas RI e FI.

2.3.2.2 Construindo um sistema EI

Duas abordagens básicas para a construção de sistemas EI podem ser citadas: abordagem da engenharia do conhecimento e abordagem do treinamento automático.

- **Abordagem da engenharia do conhecimento:** engenharia do conhecimento é o processo de construir sistemas usando a abordagem de tentativa e erro. Um engenheiro de conhecimento é responsável por construir um sistema e modificá-lo depois de consultar um especialista que tenha conhecimento sobre o domínio onde o sistema é construído. Isto é normalmente um processo interativo onde o engenheiro de conhecimento extrai um conjunto de regras que são executadas sobre um conjunto de dados de treinamento. A saída é então examinada e as regras que compõem o sistema são modificadas de acordo. Esta abordagem é referida como baseada no conhecimento humano (human-based approach) e tem a vantagem de incorporar os mais bem sucedidos sistemas EI desenvolvidos até o presente estudo. Entretanto, ele apresenta problemas como:
  - tedioso ciclo testar-e-depurar,
  - intervenção direta do trabalho e das habilidades humanas,
  - dependência dos recursos lingüísticos.

  Resumindo, a abordagem da engenharia de conhecimento requer gramáticas construídas à mão e padrões no domínio para serem descobertos pelos especialistas pela inspeção da saída de algumas amostras de treinamento. Veja [35] para mais detalhes.

- **Abordagem do treinamento automático:** Uma pessoa que tenha bom conhecimento do domínio e das tarefas de EI requeridas faz as anotações no conjunto de documentos sob consideração para a informação a ser extraída. Por exemplo, os documentos usados por um sistema EI para reconhecimento de nomes tem todos os nomes próprios relevantes no domínio anotados. Depois da anotação, alguns algoritmos são executados no conjunto de documentos considerados para retirar informação que possa ser empregada por um sistema EI quando aplicado a novas coleções. Esta informação pode ser considerada como uma gramática "feita à mão" na abordagem da engenharia de conhecimento. Claramente, esta abordagem tem a vantagem de requerer muito menos intervenção humana. Não são necessárias regras pré-definidas; apenas informação extraída a partir do conjunto de treinamento é usada. Veja [35] para mais detalhes.

2.3.2.3 Componentes de um sistema EI

Embora sistemas EI variem em arquitetura, a maior parte deles compartilham uma base comum de componentes. Estes componentes são resumidos na figura 2.5 e descritos suscintamente a seguir.

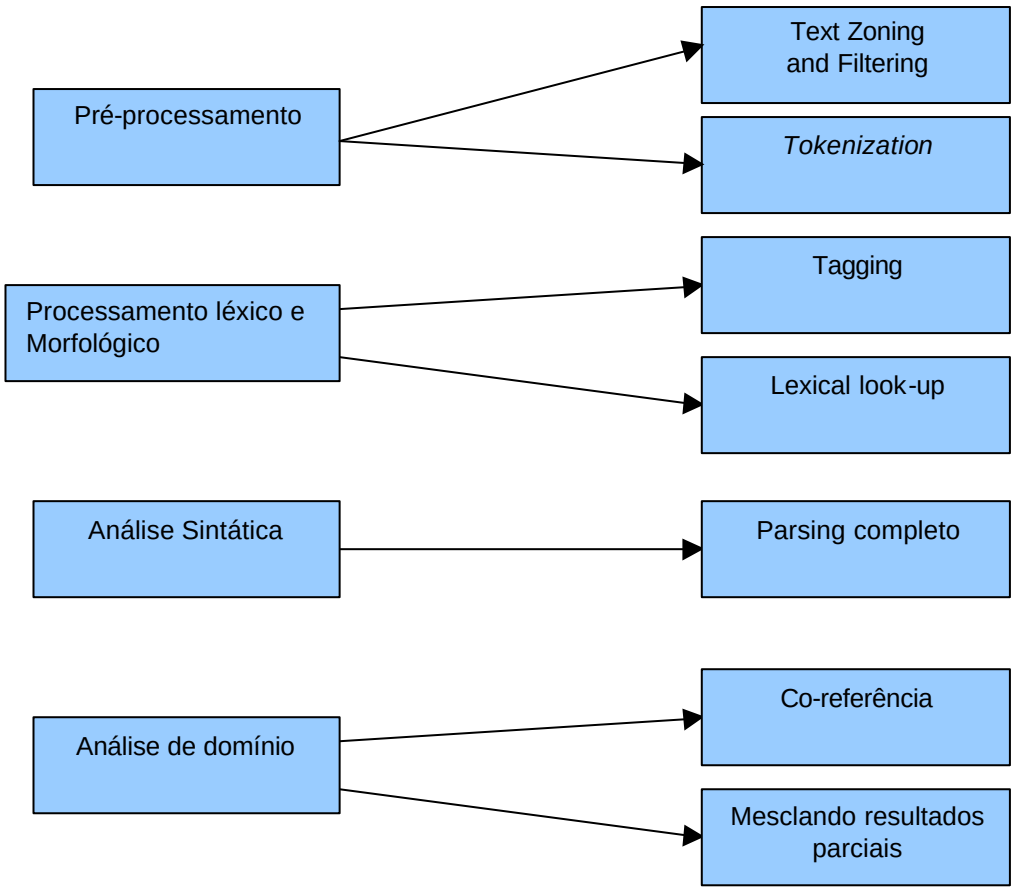


Figura 2.5 Componentes Principais em um sistema EI

O processo envolve *tokenization*, zoneamento de textos, e filtragem de textos. *Tokenization* é a parte onde se divide o texto em sentenças, parágrafos, palavras, etc. Zoneamento de textos é o processo de análise de texto (parsing), separando em segmentos de regiões formatadas e não formatadas. A filtragem de textos é o processo de ignorar informação não relevante. Usualmente, técnicas estatísticas são empregadas para decidir sobre a relevância de uma informação fornecida dentro de regiões do texto. Processamento Léxico e Morfológico incluem *derivação/lookup*, *part of speech tagging* e *semantic tagging*. *Lexical lookup* é o processo de construir um léxico para identificar o que exatamente se está procurando. Geralmente, léxicos incluem termos específicos de todos os domínios e incorporam uma parte envolvendo o idioma, mas não devem ser muito grandes. *Part of speech tagging* é usada para encontrar unidades frasais, mas pode não ser necessária para todas as aplicações de EI. *Semantic Tagging* ou *word sense tagging* é o processo de encontrar unidades frasais de nomes, como pessoas ou corporações que podem ser de interesse. Por exemplo, pode-se usar letras maiúsculas como indicativos para nomes próprios. Depois de dividir o texto em regiões

(usualmente sentenças) e definir um léxico, a análise semântica é feita. Isto inclui análise ou *parsing*, onde sentenças são divididas em unidades presentes no léxico através de *parse trees*. Só se realiza essa análise ou parsing em sentenças que poderiam conter informação valiosa. Em alguns sistemas, *parsing* envolve somente termos dependentes de domínio onde se tenta localizar somente padrões que estão dentro da sentença. A análise do domínio envolve co-referência e fusão dos resultados parciais. Co-referência é simplesmente o processo de unificar todas as unidades que referenciam a mesma entidade semântica. Entidades relevantes poderiam ser referenciadas de múltiplas formas, várias vezes. Por exemplo, "International Business Machines", "IBM", e "Big Blue" são todas referências à mesma entidade. Isto mostra o caminho a um nível mais alto de fusão, fundindo resultados parciais, onde toda a informação pertencendo à mesma entidade de informação é fundida para produzir uma saída parcial. Veja [35] para mais detalhes.

2.3.3 Extração de Características

Métodos de extração de características são usualmente decompostos em dois passos distintos [36]:

- a extração de termos pode ocorrer nas bases da informação lingüística estruturada.
- a seleção de termos pode ocorrer nas bases de alguma métrica estatística, como a freqüência, informação mútua, coeficiente Φ^2 , etc.

2.3.3.1 Informação Linguística definido importância

Neste primeiro passo, a extração de características tenta identificar nomes, e em alguns casos pode até determinar se o nome é uma pessoa, um lugar, uma empresa, ou outro objeto. Algoritmos de extração de características podem usar dicionários ou thesaurus para identificar alguns termos, e padrões lingüísticos para detectar outros. O nome de uma empresa, por exemplo, pode não estar no dicionário, mas um programa de extração de características poderia ser capaz de determinar que ele é um nome e provavelmente um termo significativo.

A extração de termos pela utilização de padrões linguísticos pode ser realizada a partir de padrões morfo-sintáticos pré-definidos, como por exemplo:

- 'Nome Prep Nome': cargo de gerente, pista de dança, etc.
- 'Nome Adjetivo': água gelada, taxa mensal, etc.

Pode-se empregar ainda algoritmos de reconhecimento de padrão que costumam utilizar uma técnica chamada Hidden Markov Models (HMM) [37]. Esses modelos podem ser treinados para detectar padrões numa seqüência de objetos. Por exemplo, um nome, seguido por um verbo é, freqüentemente seguido por outro nome. Treinados com exemplos suficientes, os algoritmos HMM podem aprender um grande número de padrões com uma exatidão bem alta.

2.3.3.2 Métrica Definidoras de Importância

Alguns métodos são mostrados a seguir, cada um deles empregando um critério de limite para importância do termo de forma a alcançar o grau desejado de eliminação de termos a partir do vocabulário total do documento.

2.3.3.2.1 Freqüência de Documentos - Document Frequency (DF)

Freqüência de documentos é o número de documentos no qual um termo ocorre. A idéia deste método é o cálculo da freqüência de cada termo e a remoção do espaço das características daqueles termos cuja freqüência de documentos é inferior a um limite determinado. A suposição básica é a de que termos raros ou são não-informativos para predizer a categoria ou não influenciam o desempenho global. Em qualquer dos casos, a remoção de termos raros reduz a dimensionalidade do espaço de características.
Freqüência de documentos é a técnica mais simples de redução de termos. Ela é facilmente escalável para conjuntos bem maiores de textos com uma complexidade computacional aproximadamente linear em relação ao número de documentos.

2.3.3.2.2 Ganho de Informação

Ganho de Informação é freqüentemente empregado como um critério de importância do termo no campo do aprendizado de máquina [38]. Ele mede o número de partes de informação obtidas para predição da categoria, pela presença ou ausência de um termo

em um documento. Dado um conjunto de documentos, o ganho de informação é calculado para cada termo, e os termos cujos ganhos de informação são menores que um determinado limite são retirados do espaço das características. Este cálculo inclui as estimativas das probabilidades condicionais de uma categoria dado um termo, e o cálculo da entropia na definição. A estimativa da probabilidade tem uma complexidade de tempo de *O(N)* e uma complexidade de espaço de *O (V N)* onde *N* é o número de documentos e *V* é o tamanho do vocabulário. O cálculo da entropia tem uma complexidade de tempo de *O (V m)* onde *m* é o dimensionalidade do espaço das categorias.

2.3.3.2.3 Informação Mútua

Informação mútua é um critério comumente usado em modelagem estatística da linguagem em associações de palavras e aplicações correlatas [39]. Considerando-se uma tabela de contingências de um termo *t* e uma categoria *c*, *A* é o número de vezes em que *t* e *c* co-ocorrem, *B* é o número de vezes em que *t* ocorre sem *c*, *C* é o número de vezes em que *c* ocorre sem *t*, e *N* é o número total de documentos, então o critério de informação mútua entre *t* e *c* é definido como:

$$I(t,c) = \log \frac{P_r(t \wedge c)}{P_r(t) \times P_r(c)} \qquad \text{Equação 2.3}$$

e é estimada usando-se:

$$I(t,c) \approx \frac{\log A \times N}{(A+C)(A+B)} \qquad \text{Equação 2.4}$$

I(t,c) tem naturalmente o valor de zero se *t* e *c* são independentes. Para medir a importância de um termo em uma seleção de características global, combinam-se as pontuações específicas da categoria de um termo em duas formas alternativas:

$$I_{avg}(t) = \sum_{i=1}^{m} P_r(c_i) I(t, c_i) \qquad \text{Equação 2.5}$$

$$I_{\max}(t) = \max_{i=1}^{m} \{I(t, c_i)\}$$ Equação 2.6

O cálculo da Informação Mútua tem uma complexidade de *O(Vm)*, similarmente ao cálculo do Ganho de informação.

Uma deficiência da Informação Mútua é que a pontuação é fortemente influenciada pelas probabilidades marginais dos termos, como pode ser visto nesta forma equivalente:

$$I(t, c) = \log P_r(t|c) - \log P_r(t)$$ Equação 2.7

Para termos com uma probabilidade igual a $P_r(t|c)$, poucos termos vão ter uma pontuação maior que termos comuns. Desta forma, não se pode comparar pontuações entre termos de freqüência muito diferente.

Os filtros de seleção atuais consistem de combinações de diferentes métricas associadas cujos limites são definidos experimentalmente.

2.3.3.2.4 Estatística X^2

A estatística X^2 mede a falta de dependência entre t e c e pode ser comparada à distribuição X^2 com um grau de liberdade para julgar extremos. Usando uma tabela de contingências de um termo t e uma categoria c, onde A é o número de vezes que t e c co-ocorrem, B é o número de vezes que t ocorre sem c, e C é o número de vezes que c ocorre sem t, D é o número de vezes que nem c nem t ocorrem, e N é o número total de documentos, a medida de importância é definida por:

$$c^2(t, c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)}$$ Equação 2.8

A estatística X^2 tem naturalmente um valor igual a zero se t e c são independentes. Calcula-se para cada categoria a estatística X^2 entre cada termo no conjunto de

documentos e aquela categoria, e então combina-se as pontuações específicas da categoria para cada termo na medida:

$$c^{2}_{avg}(t) = \sum_{i=1}^{m} P_r(c_i)\, c^{2}(t, c_i) \qquad \text{Equação 2.9}$$

2.3.4 Sumarização

É o processo da redução da quantidade de texto em um documento, porém mantendo seus significados-chave [41].

2.3.4.1 Sumarização por Abstração

Este tipo de sumarização tenta trabalhar da mesma forma em que o ser humano resume, ou seja, pela determinação do significado e posterior escrita em um pequeno documento, enfatizando as idéias principais do texto original. Pesquisadores de Inteligência Artificial fizeram algumas incursões nesta área, mas a criação de resumos por abstração não é uma opção prática hoje.

2.3.4.2 Sumarização por Extração

A Sumarização indireta não usa padrões como aqueles encontrados na extração de características. Ao invés disso, sentenças inteiras ou parágrafos são copiados do documento original numa tentativa de construir um texto menor que ainda conserve as idéias-chave do documento original. Isto é chamado de sumarização por extração.
A sumarização por extração é baseada na medida da importância relativa das palavras em um documento. Palavras como as *stopwords*, mesmo ocorrendo freqüentemente, não ajudam na distinção do significados dos textos e são ignoradas. Outras palavras são analisadas para identificar seus *stems*, e a freqüência destes *stems* são um indicativo de sua importância, uma vez que todas as variações de uma palavra são substituídas pelo mesmo *stem* e computadas como se fossem a mesma. Enquanto um contador simples de *stems* de palavras pode trabalhar bem em alguns casos, outras pistas estão freqüentemente disponíveis e são utilizadas para melhorar o processo.

2.3.4.2.1 Extração Automática de Resumos

A extração automática de resumos, é tipicamente baseada na extração de sentenças importantes a partir dos textos.

Pesquisadores tendem a concentrar seus esforços na extração de sentenças mais importantes, levando em conta alguns quesitos como:

- a posição de uma sentença dentro do documento ou parágrafo
- a presença de palavras-chave e expressões tais como 'importante', 'definitivamente', 'em particular' (todas positivas), e 'talvez', 'por exemplo' (todas negativas)
- a presença de construções indicativas tais como ' A finalidade dessa pesquisa é' e 'Nossa investigação mostrou que'
- o número de links semânticos entre uma sentença e seus vizinhos

Os métodos acima de extração de resumos são freqüentemente referenciados como procedimentos de extração.

2.3.5 Categorização

A categorização, em *Text mining*, visa identificar os tópicos principais em um documento e associar este documento a uma ou mais categorias pré-definidas [42]. Existem duas maneiras de criar as categorias. A primeira é a criação de um thesaurus para definir o conjunto de termos específicos para cada domínio e a relação entre eles [43]. A estrutura do thesaurus, com as relações pertinentes estão apresentadas na seção 2.2.4. As categorias podem assim ser criadas, baseando-se na freqüência das palavras específicas de cada domínio que estão no texto. Uma outra forma seria treinar uma ferramenta de categorização com um conjunto de documentos amostrais, como mostrado na figura 2.6. Um conjunto de exemplos representando cada categoria é apresentado à ferramenta que, então, analisa estatisticamente modelos lingüísticos, tais como afinidades léxicas e freqüências de palavras, para produzir uma assinatura estatística para cada categoria. O categorizador aplica as assinaturas estatísticas a documentos para encontrar os candidatos mais parecidos. A maior vantagem desta abordagem é que o processo trabalhoso de montar um thesaurus é evitado.

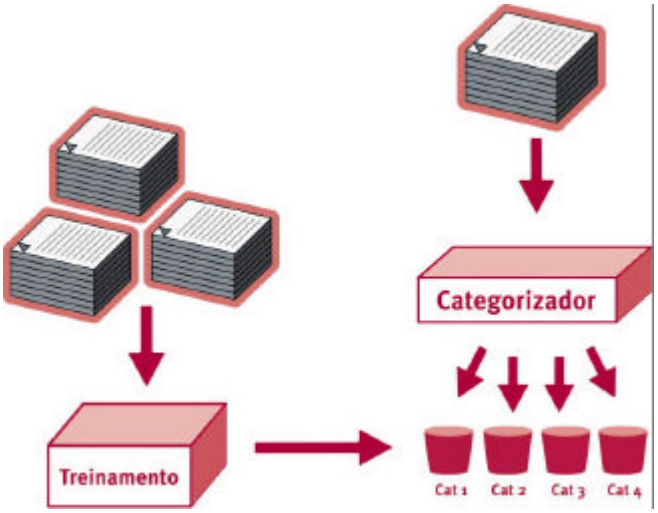


Figura 2.6 Processo de Categorização

Os vários algoritmos de categorização de documentos que têm sido desenvolvidos dividem-se em duas categorias gerais. A primeira categoria contém algoritmos de aprendizado de máquina [45] tais como árvores de decisão, conjuntos de regras, classificadores baseados em exemplos, classificadores probabilísticos, *support vector machines*, etc., que tem sido ou usados diretamente, ou adaptados para uso no contexto de dados em forma de documentos. A segunda categoria contém algoritmos de categorização especializados desenvolvidos a partir da área de recuperação de informação. Exemplos de tais algoritmos incluem *feedback* de relevância, classificadores lineares, classificadores de conjuntos de exemplos genéricos, etc.

As categorias são escolhidas para corresponder aos tópicos ou temas dos documentos. O principal objetivo da categorização de textos é a organização automática. Alguns sistemas de categorização (ou categorizadores) retornam uma única categoria para cada documento enquanto outros retornam categorias múltiplas. Em ambos os casos, um categorizador pode retornar nenhuma categoria ou algumas categorias com confiabilidade muito baixa; nesses casos, o documento é normalmente associado a uma categoria rotulada como "desconhecida" para posterior classificação manual.

O início da categorização de textos basicamente foi guiada pela engenharia do conhecimento. Dado um conjunto de categorias pré-definidas, um conjunto de regras é definido manualmente para cada categoria por especialistas. Essas regras especificam condições que um documento deve satisfazer para pertencer à categoria correspondente. Nos anos 90, o aprendizado de máquina começou a ficar popular e assumir o processo

de categorização. A categorização por aprendizado de máquina provou ser tão acurada como a categorização dirigida por especialistas, ao mesmo tempo ela é mais rápida e não requer especialistas. Neste estudo, está sendo considerada apenas a categorização por aprendizado de máquina. Veja [46] para uma descrição completa sobre categorização de textos.

2.3.5.1 Categorização de rótulo simples vs categorização multi-rótulo

A categorização de rótulo simples associa cada documento a uma e apenas uma categoria; por outro lado, a categorização multi-rótulo retorna um conjunto de k – onde k é pré-definida – categorias às quais um documento pode pertencer. No primeiro caso, têm-se categorias não superpostas. Várias aplicações usam um caso especial do primeiro caso chamado caso binário onde se tem duas categorias complementares; um documento pode pertencer a apenas uma dessas categorias [46].

2.3.5.2 Categorização por pivotamento de categoria vs pivotamento de documento

A categorização por pivotamento de categoria (CPC) encontra todos os documentos que podem ser arquivados sob uma categoria específica. Por outro lado, a categorização por pivotamento de documento (CPD) encontra todas as categorias sob as quais um certo documento pode ser arquivado. CPD é usado em aplicações onde os documentos vem como uma sequência (isto é, não disponível ao mesmo tempo). Um exemplo é a filtragem de e-mail onde um novo e-mail recebido é associado a uma categoria específica. CPC é usado quando novas categorias podem ser adicionadas dinamicamente porque um conjunto de documentos não parece ser classificado sob qualquer das categorias dadas. CPD é o mais popular.

2.3.5.3 Categorização rígida (*hard*) vs ordenação

Dado um documento d para ser classificado, a categorização por ordenação retorna uma lista ordenada de categorias – contendo todas as categorias – de tal forma que categorias com maiores probabilidades de conter d são colocadas no topo da lista. Essa lista poderia servir como uma forma de adicionar conhecimento do especialista na classificação porque apenas categorias no topo da lista serão consideradas. Uma variação disso poderia ser quando é dada uma categoria c e o sistema retorna uma lista

ordenada contendo todos os documentos de tal forma que documentos no topo da lista tem maiores probabilidades de pertencer a *c*. O primeiro é o caso referido como *ranking* de categoria e o último como *ranking* de documentos. A categorização por ranking é referida como categorização semi automática e é usada em aplicações onde a exatidão do classificador é muito crítica e os resultados de um sistema totalmente automatizado comprovaram ser significativamente menos acurados que aqueles com *expertise* humano.

A categorização rígida é o processo de retornar uma única categoria, dado um documento. A categorização rígida é totalmente automatizada e é mais comum.

2.3.5.4 Aplicações em Categorização de Texto

Aplicações em Categorização de Texto (CT) datam dos anos 60. A Categorização de Texto se expandiu de outras áreas como Recuperação de Informação e Filtragem de Informação até que se tornou uma área de pesquisa própria nos anos 80. A seguir, um resumo das maiores áreas de aplicação em CT é apresentado:

- **Indexação Automática para Sistemas RI :** Cada documento que é dado é representado por um conjunto de palavras e/ou frases que descrevem seu conteúdo. Os termos selecionados são retirados de um grupo de termos chamado de dicionário de controle. A criação de tal dicionário é feita utilizando conhecimento humano e é muito cara. As aplicações geralmente definem dois números U e L de tal forma que a cada documento é associado um número de termos entre U e L – isto é, $L < n < U$. A categorização por pivotamento de documentos (explicada anteriormente) é muito comum em tais aplicações. Veja [47] para um exemplo de tais aplicações.

- **Organização de Documentos:** A aplicação de Categorização de Texto envolvendo organização de documentos tipicamente associa um documento a uma ou mais categorias com o objetivo de organizar esses documentos sendo eles de uso pessoal ou corporativo. Por exemplo, jornais recebem um número de anúncios todo dia e se beneficiariam muito com um sistema automático que pudesse associar esses anúncios às suas categorias correspondentes (Imóveis,

Autos, etc.). Uma finalidade importante para a organização de documentos é facilitar o processo de busca. Veja [48] para exemplos.

- **Filtragem de Textos:** Aplicações de Categorização de Texto são usadas em um nível mais alto no campo da Filtragem de Informação onde uma sequência de documentos pode ser direcionada para um usuário específico ou filtrada, tomando como base os perfis dos usuários que expressam os interesses dos usuários. A fonte dos documentos é referida como produtora e o usuário como consumidor. Para cada consumidor, o sistema bloqueia a entrega de qualquer dos documentos da sequência em que o consumidor não esteja interessado. O sistema usa o perfil do usuário para categorizar o documento como interessante ou não para o usuário. O categorizador binário de rótulo simples é usado para categorização de textos – as duas categorias são: relevante e irrelevante. O sistema poderia ser melhorado também no sentido de classificar todos os documentos direcionados, isto é, nem bloqueados nem filtrados – em um conjunto de categorias pré-definidas como a organização de documentos. Um exemplo poderia ser um filtro de *e-mails* que bloqueasse mensagens indesejadas e organizasse as recebidas nas categorias definidas pelo usuário como mostra [49]. Em qualquer caso, o filtro pode estar situado no lado do produtor onde ele roteia documentos para todos os consumidores não-bloqueados, ou ele pode estar situado no lado do consumidor onde ele bloqueia documentos indesejados.

- ***Word Sense Disambiguation* (WSD):** é o processo pelo qual um sistema é capaz de encontrar o sentido de uma palavra ambígua, ou seja que tenha mais de um significado, baseada em sua ocorrência no texto. Por exemplo, a palavra "classe" poderia significar um lugar onde os estudantes estudam, ou uma categoria, dado um texto contendo esta palavra, uma aplicação desse tipo deve retornar qual dos significados é pretendido pelo texto em questão. O Word Sense Desambiguation é usado em processamento de linguagem natural (PLN) e indexação de documentos por sentido de palavras. É considerada uma tarefa CT por pivotamento de documentos de rótulo simples. Em [50] este assunto é tratado mais extensivamente.

- **Categorização Hierárquica de Páginas Web:** A categorização hierárquica de páginas Web classifica páginas da Web ou websites em um conjunto hierárquico de categorias pré-definidas residentes nos portais. Ao invés de usar mecanismos de busca para encontrar documentos através da internet, pode-se seguir um caminho mais fácil pela hierarquia de categorias de documentos para procurar pelo alvo apenas na categoria especificada. CPC é usada para permitir adição e/ou exclusão automática de documentos novos ou indesejados respectivamente. Veja [51] para um exemplo.

2.3.6 Construção de Classificadores de Texto

Independentente da aplicação, um classificador de texto pode ser ansioso ou preguiçoso. No primeiro caso, um conjunto de documentos pré-classificados é usado para construir o classificador. Este conjunto de dados é dividido randomicamente em conjunto de treinamento, T_R, e conjunto de teste, T_S. As duas partições não precisam ser iguais. O T_R é usado para construir ou treinar o classificador e então o T_S é usado para testar a precisão do classificador. Este processo é repetido até que um classificador aceitável é desenvolvido. Por outro lado, um classificador preguiçoso não constrói um classificador a frente no tempo. Sempre que uma nova amostra não rotulada n é recebida, o classificador encontra as k amostras rotuladas mais similares, onde k é pré-definido, para decidir sobre o rótulo de n. Vários algoritmos que se adequam às duas categorias acima têm sido desenvolvidos.

2.3.6.1 Classificadores de Árvore de Decisão

Classificadores de Árvore de Decisão (CAD) tentam superar a dificuldade de interpretar os classificadores probabilísticos que são quantitativos pelo uso de representações simbólicas. Um Classificador de Árvore de Decisão é basicamente uma árvore com nós internos representando termos, bordas representando testes no peso que um termo deve ter, e nós-folha representando categorias. O classificador classifica um documento percorrendo um caminho para o nó-folha apropriado – isto é, classe ou categoria.

Referir-se ao capítulo 3 em [38] para maiores detalhes. Alguns exemplos de tal classificador incluem ID3, C4.5 e C5.

2.3.6.2 Classificadores de Regra de Decisão

Classificadores de Regra de Decisão criam uma regra disjuntiva normal (CRD) para cada categoria c_i. A premissa da regra é composta de letras significando a presença ou a ausência de uma palavra-chave no documento d_j, enquanto a cláusula denota que a decisão de classificar d_j para c_i leva a premissa a ser satisfeita por d_i. Um aprendiz de regra do tipo CRD que tem sido aplicado à Classificação de Texto é o CHARADE [53].

2.3.6.3 Classificadores de Regressão

Um exemplo de um algoritmo derivado de regressão é o *linear least squares fit* (LLSF) onde cada documento d_j é associado com um vetor de entrada $I(d_j)$ de T termos ponderados e um vetor de saída $O(d_i)$ de C pesos (um para cada classe). A classificação aqui é feita produzindo um vetor de saída de uma amostra dado seu vetor de entrada. Uma matriz M, $C \acute{} T$ é construída para realizar esta tarefa de forma que $M(d_j)=O(d_j)$. De acordo com [54], LLSF é um dos classificadores mais conhecidos; entretanto, ele sofre do alto custo de computar a matriz M.

2.3.6.4 Classificadores Rocchio

Cada categoria é representada por um perfil (um documento protótipo) que é criado usando o método Rocchio [46] que dá pesos apropriados aos termos no espaço de termos de tal forma que o vetor de perfil resultante é o mais útil para discriminar a classe correspondente. Este perfil é então usado para classificação de outros documentos. Um exemplo interessante é mostrado em [55].

2.3.6.5 Redes Neurais

Uma rede neural (RN) tem um conjunto de nós dividido em camadas. A primeira camada é a camada de entrada, seguida de zero ou mais camadas intermediárias, seguida de uma camada de saída. Cada nó recebe um peso de entrada e produz uma saída. Os nós de entrada são usados para representar todos os termos e os termos de

saída representam o conjunto de categorias. Para classificar um documento d_j, os pesos do termo representando d_j são alimentados nos nós de entrada. Isto será propagado pela rede até que finalmente o resultado seja recebido pelos nós de saída. Em [56] uma rede neural não-linear é apresentada.

2.3.6.6 Classificadores baseados em exemplos

Classificadores baseados em exemplos são classificadores preguiçosos. Eles não constróem um classificador a frente no tempo mas usam as amostras rotuladas para classificar novas amostras. Um classificador deste tipo é o k-Vizinho mais Próximo (KNN) [57] que, dado um novo exemplo d_j, tenta encontrar k documentos que são mais similares a d_j entre o conjunto de amostras dado. Então um processo – tal como o *plurality voting* – é executado pelos vizinhos selecionados para dar a d_j o rótulo mais apropriado.

2.3.6.7 Classificadores Probabilísticos

Classificadores Probabilísticos representam a probabilidade de um documento d_x pertencer à categoria c_y por $P(c_y|d_x)$ e a computa pela seguinte fórmula:

$$P(c_y|d_x) = P(c_y)P(d_x|c_y)/P(d_x) \qquad \text{Equação 2.10}$$

onde $P(c_y)$ é a probabilidade que um documento radomicamente selecionado pertença a categoria c_y, $P(d_x)$ é a probabilidade que um documento radomicamente selecionado tenha d_x como uma representação vetorial e $P(d_x|c_y)$ é a probabilidade que a categoria c_y contenha o documento d_x. O cálculo de $P(d_x|c_y)$ é facilitado pelo uso da suposição de independência ou suposição Naïve Bayes que afirma que quaisquer duas coordenadas de um vetor-documento são estatisticamente independentes. Assim $P(d_x|c_y)$ é calculada como:

$$P(d_x|c_y) = \prod_{k=1}^{T} P(W_{zx}|c_y) \qquad \text{Equação 2.11}$$

onde W_{zx} é o peso do z-ésimo termo do vetor d_x. Em [52] é apresentada uma discussão detalhada sobre o assunto.

2.3.6.8 Classificadores baseados em *Support Vector Machines* (SVM)

Support Vector Machines foi primeiramente aplicado em Classificação de Textos no final dos anos 90 [58]. SVM divide o espaço de termos em hiperplanos ou superfícies separando as amostras de treinamento positivas das negativas – algumas vezes estas superfícies são referidas como superfícies de decisão. Então a superfície que provê a maior separação (a maior margem possível entre as amostras positivas e negativas) é selecionada.

2.3.6.9 Comitê de Classificadores

Um Comitê de Classificadores é um tipo de classificador que engloba um número de outros classificadores (da lista descrita até agora), os utiliza para fazer a classificação, compara suas saídas e então seleciona o resultado mais apropriado (usando *plurality voting* por exemplo).

2.3.7 *Clustering* de Documentos

Clustering é uma técnica útil quando se quer agrupar documentos similares. O processo de *clustering* básico funciona da seguinte forma: primeiro, uma descrição simplificada do documento é criada para cada texto que vai sendo adicionado aos *clusters*. A descrição é normalmente um vetor de características, ou uma lista de temas dominantes ou palavras-chave e uma medida de importância relativa de cada tema ou palavra-chave no documento [59].

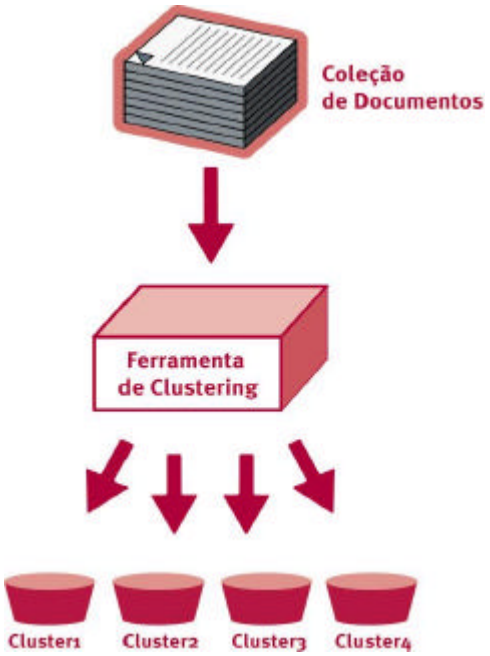


Figura 2.7 Processo Básico de *Clustering*

O próximo passo é determinar a proximidade de dois documentos baseada em seus vetores de características. Quando se faz o *clustering* de dados numéricos, normalmente isto é feito assumindo que os pesos de um vetor de características definem um ponto no espaço e encontrando a distância entre esses dois pontos. Em documentos, além das métricas de distâncias convencionais já conhecidas, existem as medidas de similaridades conceituais, que são usadas para medir a distância numa hierarquia de assunto. Medidas de similaridades conceituais são geralmente baseadas numa função de distância entre tópicos na hierarquia de assuntos e os pesos desses tópicos nos documentos. Uma vez que a distância entre documentos é calculada, seja pelas medidas de similaridade padrão ou conceituais, eles podem ser agrupados de várias formas diferentes. A figura 2.7 mostra o esquema básico da atuação de uma ferramenta de *clustering* sobre um conjunto de documentos, separando-os em subgrupos de documentos chamados *clusters*.

2.3.7.1 Técnicas de *Clustering*

Nesta seção, será apresentada uma breve descrição das técnicas de *clustering* de documentos mais populares que são *clustering* hierárquico, *clustering* K-means e *clustering* SOM.

2.3.7.1.1 *Clustering* Hierárquico

Técnicas de *Clustering* Hierárquico produzem uma hierarquia de partições com uma simples partição incluindo todos os documentos num extremo, e *clusters* unitários cada um composto de um documento individual no outro extremo. A árvore descrevendo a hierarquia de *clusters* é chamada de dendrograma. Cada *cluster* ao longo da hierarquia é visto como uma combinação de dois *clusters* a partir do próximo nível mais alto ou mais baixo, dependendo se a abordagem é divisiva ou aglomerativa, respectivamente.

a) Aglomerativa: O *Clustering* aglomerativo começa com o conjunto de todos os *clusters* cada um incluindo um documento como a raiz da árvore e então combina os pares de *clusters* mais similares ou mais próximos juntos a cada nível da árvore até que ela forme um único *cluster* contendo todos os documentos no nível de folha. Isto requer a definição de uma métrica – distância ou similaridade – para ser efetuada.
   A maior parte dos pacotes estatísticos utilizam métodos aglomerativos, dentre os quais os mais conhecidos são:

   (1) *single linkage*: o modelo aglomerativo básico é o *single linkage*, também chamado vizinho mais próximo. Ele pode começar a partir da matriz de similaridades entre objetos (matriz de distâncias ou matriz de semelhanças). Ele ordena pares de objetos segundo sua similaridade de maneira descendente e forma *clusters* hierarquicamente, começando dos pares mais similares. A cada passo aglomera-se uma observação adicional para formar um *cluster*. Assim, o primeiro *cluster* é aquele com duas observações que apresentam a menor (mínima) distância entre si. Uma terceira observação que tenha a próxima menor distância, é adicionada ao *cluster* de duas observações para criar um *cluster* de três observações ou um novo *cluster* com duas observações é formado. O algoritmo continua até que todas as observações estejam em um *cluster*. A distância entre quaisquer dois *clusters* é a menor distância de qualquer ponto em um *cluster* para qualquer ponto em outro *cluster*. Dois *clusters* são unificados em qualquer estágio pelo *link* simples mais curto ou forte entre eles.

No *single linkage*, para juntar um objeto a um *cluster* já existente, ele deve ter uma similaridade ao menos igual à considerada para um dos objetos já incluídos no *cluster*. Dessa forma, um elemento será agrupado com um *cluster* no nível correspondente à similaridadde máxima entre ele e os elementos no *cluster*.

Devido à sua natureza, o single linkage apresenta uma propriedade que é chamada encadeamento (*chaining*). Como resultado, ele pode ser inadequado em vários casos. O encadeamento significa que os objetos juntam-se a um *cluster* facilmente se existem objetos intermediários que façam a ponte entre o *cluster* e os outros objetos. Uma vez que o encadeamento aparece onde existem pontos intermediários, *single linkage* é uma boa forma de detectá-los. Por outro lado, o encadeamento faz com que o *single linkage* seja propenso a ruídos nos dados. Quanto maior o *cluster* é, mais fácil é unir mais e mais elementos, porque a probabilidade de encontrar objetos intermediários aumenta. Assim, *single linkage* é conhecido como aquele que contrai o espaço das relações entre objetos na proximidade dos *clusters*.

O *single linkage* foi desenvolvido por Florek [70,71] e mais tarde reinventado por McQuitty [72] e Sneath [73].

(2) *complete linkage*: Este é semelhante ao *single linkage*, exceto que é baseado na máxima distância, e não na distância mínima. No *complete linkage*, dois *clusters* se fundem dependendo do par de objetos mais distante entre eles. Em outras palavras, A distância máxima entre quaisquer dois indivíduos em um *cluster* representa a menor esfera (diâmetro mínimo) que pode englobar o *cluster* [74]. Em outras palavras, um objeto se junta a um *cluster* quando a sua similaridade em relação a todos os elementos daquele *cluster* é igual ou maior que o nível considerado. Em um *cluster* que está crescendo, novos objetos tem pouca chance de se juntar, porque eles devem estar próximos a todos os objetos no *cluster*, então eles se juntam posteriormente quando o nível de similaridade é considerado baixo. Assim, quando um *cluster* cresce, ele se move para longe de outros objetos. Complete linkage é conhecido como aquele que expande o espaço na proximidade dos *clusters*. Este

algoritmo pode ser adequado quando se quer inspecionar descontinuidades claras.

(3) *average linkage*: Nesse caso é usada a distância ponderada de amostras de um *cluster* para amostras de outros *clusters*. O par de *clusters* com a menor distância ponderada é unido. Isto mantém uma "densidade" dos *clusters* relativamente constante o que significa que regiões com uma densidade alta de pontos podem ter um *cluster* maior que aquelas com uma densidade menor.

(4) método de *Ward*: Este método é distinto de todos os outros métodos porque ele usa uma abordagem de análise de variância para avaliar as distâncias entre *clusters*. Em resumo, este método tenta minimizar a Soma dos Quadrados (SS) de quaisquer dois *clusters* (hipotéticos) que podem ser formados a cada passo. Referir-se a [75] para detalhes concernentes a este método. Em geral, este método é considerado um método muito eficiente, porém, ele tende a criar um número exagerado de *clusters* pequenos, porque quanto mais dispersas estiverem as observações, maior a soma dos quadrados fará a distância.

(5) método do Centróide: A distância euclidiana é medida entre centróides de dois *clusters*.

Todos esses métodos variam em relação à definição de distância.

b) Divisiva: O *Clustering* divisivo trabalha de forma oposta. Ele começa com um único *cluster* contendo todos os documentos no nível da raiz e então separa um *cluster* em dois a cada nível da árvore até que ele forme um conjunto de *clusters*, no nível de folha, cada uma contendo um e apenas um *cluster*.

2.3.7.1.2 *Clustering* K-means

Dado um número fixo de k, o *clustering* K-means cria um conjunto de k *clusters* e distribui o conjunto de documentos dados entre esses *clusters* usando a similaridade entre os vetores-documento e os centróides dos *clusters*. Um centróide é o vetor médio de todos os vetores-documento no respectivo *cluster*. Cada vez que se adiciona um documento em um *cluster*, o centróide daquele *cluster* é recalculado. Note que, quase sempre, um centróide não corresponde a um documento. A similaridade entre um documento *d* e um centróide *c* é calculada como o somatório de todos os vetores-documento no *cluster* dividido pelo número de vetores-documento.

Em geral, acredita-se que os métodos de *clustering* hierárquicos produzem *clusters* de melhor qualidade que aqueles produzidos pelo K-means (ou uma de suas variantes), o qual tem uma complexidade linear ao número de documentos, mas produz *clusters* de menor qualidade [2].

2.3.7.1.3 *Clustering* de Palavras

Métodos de *Clustering* podem ser usados para reduzir o número de dados pelo agrupamento de itens similares [77]. Na representação de documentos, métodos de *clustering* podem ser aplicados para agrupar palavras similares, e então representar documentos mais em função de *clusters* de palavras que em palavras individuais. Um *clustering* bem adequado para a representação de documentos deve reduzir a variação da forma, ao mesmo tempo deve perder tão pouca informação quanto possível considerando o conteúdo semântico, especialmente os tópicos discutidos em um documento.

Um apanhado de vários métodos para coletar informação de palavras com a finalidade de realizar um *clustering* dessas palavras automaticamente é apresentado em [78]. Em idiomas com restrições estritas na ordem das palavras, como o Inglês, a distribuição das palavras no contexto imediato de uma palavra contém quantidades consideráveis de informação considerando a categoria sintática de uma informação, e mais ainda dentro das categorias sintáticas, informação sobre a categoria semântica [79].

2.3.7.2 Avaliação da Qualidade do *Cluster*

Vários esquemas tem sido desenvolvidos para avaliar a qualidade dos *clusters* produzida por um técnica de *clustering*. O esquema de Entropia é baseado na idéia de que a melhor entropia é obtida quando cada *cluster* contém apenas um documento [2]. Primeiro, calcula-se a distribuição de classes dos documentos. Para cada *cluster* produzido por uma certa técnica, calcula-se a probabilidade de um documento em um *cluster* j pertencer a uma classe i, p_{ij}. Então a entropia para cada *cluster* j é calculada usando a seguinte fórmula:

$$Entropiaj = -\sum p_{ij} \log\left(p_{ij}\right) \qquad \text{Equação 2.12}$$

Assim, a entropia de um conjunto de *clusters*, s, é calculada como

$$Entropias = \sum_{j=1}^{|s|} \frac{n_j * Entropia\ j}{n} \qquad \text{Equação 2.13}$$

Onde $|s|$ é o número de documentos em s, n_j é o tamanho do *cluster j*, e *n* é o número total de documentos em todos os *clusters*.

Um outro esquema é baseado em RI é a medida *F-measure*. Aqui, considera-se cada *cluster* como sendo o resultado de uma *query* ou consulta e cada classe como sendo o resultado que se supõe como retorno a uma *query*. Calcula-se o *recall* e a precisão (veja a seção RI e FI) como:

$$Recall(i,j) = n_{ij}/n_i \qquad \text{Equação 2.14}$$

$$Precisão(i,j) = n_{ij}/n_j \qquad \text{Equação 2.15}$$

Onde n_{ij} é número de documentos que existem numa classe *i* e no *cluster j* ao mesmo tempo, n_j é o tamanho do *cluster j* e n_i é o tamanho da classe *i*. A medida *F* geral é computada como a média de todas as *F-measures*. Existem outros esquemas na literatura; em [19] capítulo 3 podem ser encontrados mais detalhes.

2.3.7.3 Aplicações em *Clustering* de Texto

Clustering tem sido empregado em um número de aplicações baseadas em texto tais como Categorização de Textos [80] e Recuperação de Informação [81]. Ele é também usado na varredura de uma coleção de documentos [82], organizando documentos

retornados por um mecanismo de busca em resposta a uma *query* [63], automaticamente gerando *clusters* hierárquicos de documentos [83]. *Clustering* é o agrupamento de representações de documentos similares em partições onde os documentos nas mesmas partições são mais similares uns aos outros que a qualquer outro documento em qualquer outra partição. *Clusters* são usualmente mutuamente exclusivos e coletivamente exaustivos. *Clustering* é útil para desenvolvimento de taxonomias como por exemplo o site Yahoo! e o Open Directory (dmoz.org/) que embora sejam construídos manualmente, podem ser muito assistidos por um processo de *clustering* preliminar de grandes quantidades de amostras de documentos Web. *Clustering* pode também ajudar a procuras rápidas baseadas em similaridade. Dado um *clustering* de um corpus pré-computado, a procura por documentos similares a um documento-consulta d_c pode ser eficientemente limitado a um pequeno número de *clusters* que são mais similares a d_c, rapidamente eliminado um grande número de documentos que se poderia seguramente supor que tenham uma colocação baixa.

Para concluir, é interessante ressaltar que as tarefas a serem realizadas nesta etapa estão relacionadas em grande parte aos objetivos a serem alcançados pela análise dos textos. Algumas tarefas atuam na melhoria de desempenho de tarefas subsequentes como, por exemplo, a extração de características que atua reduzindo o espaço das características para que tarefas como a de *clustering* ou categorização sejam mais eficientes ou até mesmo possíveis.

2.4 Pós-Processamento dos Dados

O pós-processamento dos dados consiste da fase de validação das descobertas efetuadas pela etapa de processamento dos dados e da visualização dos resultados encontrados. Métricas de avaliação de resultados, ferramentas de visualização, e conhecimento de especialistas ajudam a consolidar os resultados.

2.4.1 Métricas de Avaliação de Resultados

Essas métricas de avaliação de desempenho de um sistema foram adotadas da área de Recuperação de Informação e são baseadas na noção de relevância: se um documento atender à necessidade de informação do usuário, ele é considerado relevante à solicitação do usuário. A qualidade de recuperação de um sistema RI pode ser medida para uma coleção de textos, um conjunto de solicitações e seus respectivos documentos relevantes. As medidas de avaliação básicas são as seguintes:

2.4.1.1 Precisão

Avalia o quanto o modelo acerta:

$$\text{Precisão} = \frac{\text{número de itens relevantes recuperados}}{\text{número total de itens recuperados}}$$

2.4.1.2 Recall

Avalia o quanto o modelo contabiliza:

$$\textit{Recall} = \frac{\text{número de itens relevantes recuperados}}{\text{número de itens relevantes na coleção}}$$

Um modelo pode ser avaliado, pela representação prévia de dados não-vistos, ou pela medida do quanto alguma tarefa específica, como predição ou classificação pode ser realizada com o modelo.

2.4.2 Ferramentas de Visualização

Nos últimos anos, sistemas gráficos com telas coloridas se tornaram equipamentos comuns, além de ferramentas de programação e tecnologia para construir aplicações altamente gráficas e interativas. Ao mesmo tempo, pesquisas em visualização de dados e métodos de análise de dados exploratória despontaram, fornecendo métodos e equipamentos capazes de ilustrar propriedades e relacionamentos de conjuntos de dados graficamente complexos [84].

Informação quantitativa tem sido apresentada usando meios gráficos [85]. A informação pode ser traduzida visualmente usando-se uma combinação de pontos, linhas, símbolos, palavras, cores e intensidade de sombreamento. Em particular, o uso de gráficos pode ajudar a entender o sentido de grandes e complexos conjuntos de dados que não poderiam ser manuseados de outra maneira. Se a proximidade espacial é utilizada para traduzir similaridade de itens (usando *scatter plots*), informações referentes a *clusters*, padrões e outliers são passadas por simples observação visual, e um resumo da informação é automaticamente percebido pelo observador.

2.4.2.1 Visualizações 2-D

Allan [86,87] desenvolveu uma visualização para mostrar o relacionamento entre documentos e partes de documentos. Ele transformou os documentos em arrays de forma oval e os conectou quando sua similaridade era grande o suficiente. O objetivo imediato de Alan era encontrar grupos de documentos relevantes, mas para encontrar padrões não-usuais de relacionamentos entre documentos.

O sistema Vibe [88] é uma visualização 2-D que mostra como os documentos são relacionados uns aos outros em termos das dimensões selecionadas pelo usuário. Os documentos sendo percorridos (*browsed*) são colocados no centro de um círculo. O usuário pode colocar qualquer número de termos ao longo da borda do círculo, onde eles formam “poços de gravidade” que atraem documentos dependendo da significância desses termos naquele documento. O usuário pode modificar a localização dos termos e ajustar seus pesos para melhor entender os relacionamentos entre os documentos.

2.4.2.2 Visualizações 3-D

Estações gráficas potentes e o apelo visual dos gráficos tridimensionais têm encorajado esforços para apresentar relacionamentos entre documentos no espaço tridimensional.

O sistema Liberworld [89] inclui uma implementação do sistema Vibe descrito acima, mas apresentado no espaço 3D. O usuário ainda deve selecionar os termos, mas agora eles são colocados na superfície de uma esfera ao invés de na borda de um círculo.

A dimensão adicional deveria permitir que o usuário visse a separação mais prontamente.

O sistema descrito em [90] é similar em relação à abordagem ao sistema Bead [91] uma vez que ambos usam formas de "*spring embedding*" para colocar objetos de várias dimensões no espaço 3D. A pesquisa de Bead não investigou a questão de enfatizar a separação de documentos relevantes e não-relevantes.

2.4.3 Conhecimento de Especialistas

A participação de especialistas é importante ao longo de todo o processo de extração de conhecimento em texto acompanhando a análise: ajudando a resolver situações de conflito, indicando caminhos, complementando informações. A partir de um determinado ponto do processamento dos dados, torna-se fundamental a colaboração de especialistas para o progresso da pesquisa.

Durante a fase de validação de resultados, o que foi descoberto durante o processo de extração de conhecimento como resultado final deve novamente ser verificado pelos especialistas.

Existem, porém, alguns problemas relativos à associação do conhecimento humano a sistemas inteligentes:

- diferenças entre especialistas são típicas - comumente especialistas apresentam discordâncias em seus pontos de vista;
- o mesmo especialista em tempos diferentes, pode dar uma resposta diferente à mesma pergunta.

2.5 Ferramentas Comerciais

Nesta seção, serão comentadas ferramentas comerciais para text mining. Uma descrição sucinta de cada uma delas é apresentada. Ao final da seção, uma tabela resumindo características dessas ferramentas de text mining será mostrada.

2.5.1 Intelligent Miner for Text

Intelligent Miner for Text [104] (IBM) oferece um mecanismo de busca, ferramentas de acesso à Web, e um conjunto de ferramentas para analisar textos.

Intelligent Miner realiza recuperação de textos completa usando Mecanismo de busca IBM e disponibiliza, também, ferramentas de acesso Web, tais como NetQuestion Solution e IBM Web Crawler.

As ferramentas de análise de textos (Text Analysis Tools) realizam extração de características, clustering, categorização e sumarização. As ferramentas de análise podem extrair informação-chave de documentos, organizar os documentos por assunto, e encontrar temas predominantes em uma coleção de documentos. A versão 2.3, disponível para avaliação neste estudo, não possui interface para usuário, dificultando sua utilização. Os comandos são executados no ambiente Windows através de linhas de comando em uma janela de *Command Prompt*.

2.5.2 Temis

A ferramenta Temis [105] é composta de módulos independentes que são descritos suscintamente a seguir.

2.5.2.1 Insight Discoverer Extractor

O servidor *Insight Discoverer Extractor* usa informação lingüística e semântica para extrair conhecimento a partir de documentos. Ele identifica conceitos e relações entre conceitos, e é dirigido por regras de extração de conhecimento especializadas chamadas *Skill Cartridges*.

O servidor *Insight Discoverer Extractor* é o cérebro usado para desenvolver aplicações que desempenham a extração de informação em vários idiomas e em grandes coleções

de documentos. Os idiomas contemplados pela ferramenta são: Português, Inglês, Alemão, Francês, Italiano , Espanhol e Holandês.
O servidor *Insight Discoverer Extractor* pode extrair os mesmos conceitos em várias Línguas com os *Skill Cartridges* apropriados.

2.5.2.2 Insight Discoverer Categorizer

O servidor *Insight Discoverer Categorizer* usa um algoritmo matemático para associar documentos a categorias. O categorizador lê e aprende a partir de um conjunto de documentos que já estão colocados em uma certa categoria. Então, quando alimentado com um novo conjunto de documentos, ele categoriza comparando-os aos pré-categorizados. O categorizador trabalha nos modos "supervisionado" e "não-supervisionado". O modo "supervisionado" sugere uma ou mais categorias para cada documento. O modo "não-supervisionado" associa automaticamente os documentos às categorias mais relevantes.
O servidor *Insight Discoverer Categorizer* baseia sua categorização em um vetor semântico descrevendo cada documento para sua categorização. Ele pode ser usado para produzir vetores semânticos a partir de documentos.

2.5.2.3 Insight Discoverer Clusterer

O *Insight Discoverer Clusterer* é uma solução para estruturar informação não-estruturada. Ele classifica e reagrupa documentos em classes coerentes, baseando-se em suas similaridades semânticas. O *Clusterer* executa este processo automaticamente em tempo real. Ele cria uma árvore de conceitos pertinente para uma dada coleção de documentos. O processo de *Clustering* dessa ferramenta é iterativo e cria *subclusters* para cada *cluster*. O *clusterer* oferece uma excelente visibilidade em grandes conjuntos de documentos e em domínios complexos. O usuário pode navegar dentro dos *clusters* para ter um bom entendimento da base de dados. O módulo de *clustering* é implemtentado cono um servidor Java RMI com uma API pública. Ele utiliza uma variação do conhecido algoritmo *K-means*. Nessa ferramenta, uma sessão de *clustering* é organizada em três passos:

- Pesos e filtragem de palavras: o servidor calculará o peso para cada palavra de acordo com sua freqüência dentro do documento, e o número de documentos

com essa palavra. Essa pesagem valoriza palavras que não são muito freqüentes e remove palavras muito pouco freqüentes para evitar ruído.

- Cálculo de clusters: os documentos são organizados em clusters por um algoritmo desenvolvido pela própria Temis. É possível definir um limite mínimo de similaridade. Documentos que não combinam com um cluster nesse valor mínimo não serão afetados, mas serão mantidos em um estado inalterado.
- Descrição estatística nos clusters: o servidor armazena os resultados do clustering numa aplicação interativa, pronta para ser navegada pelo usuário. Ele fornece um título para cada cluster, baseado nas palavras mais representativas e a lista dos melhores documentos de cada cluster, sorteados por similaridade decrescente.

2.5.3 Online Miner

O servidor *Online Miner* recupera documentos e os estrutura em formato XML. Ele também extrai, organiza e monitora características importantes nos documentos. Por fim, ele organiza documentos em *clusters* coerentes de forma a facilitar a navegação dentro da coleção de documentos.

O servidor Online Miner é desenvolvido para ser amigável:

- Ele dirige e coordena servidores especializados (*Insight Discoverer Extractor*, *Insight Discoverer Clusterer*) e pode interfacear com os mecanismos de busca mais comuns.
- A única aplicação requerida para o usuário final interagir com o *Online Miner* é um navegador de web.
- Cada *Skill Cartridge* cobre um domínio vertical e uma função de negócios.

2.5.4 TextSmart

TextSmart [106] (SPSS) trabalha com dados no formato de perguntas e respostas, porque foi desenvolvido para pesquisa de opinião. Assim, para que os dados possam ser manuseados pelo programa, é necessária a formulação de perguntas. O programa tenta,

na realidade, encontrar relações entre as respostas referentes a um mesmo tema, o motivo da pergunta.

Ele executa alguns passos de pré-processamento: stopwords (adaptável para qualquer idioma), stemming (para idioma inglês), dicionário (adaptável para qualquer idioma). É uma ferramenta de categorização automática baseada em clustering de termos.

TextSmart gera uma lista contendo o ranqueamento geral dos termos, ou seja, o número de vezes que cada termo aparece em todas as respostas.

2.5.4.1 Categorização dos termos baseada em clustering

A categorização automática é um processo de três passos descritos a seguir: Primeiro, o programa cria uma matriz de similaridades a partir dos termos. Ele combina cada termo com cada outro termo na lista e checa para certificar-se qual a freqüência que cada par ocorre em uma resposta (co-ocorrência). O algoritmo constrói uma tabela de contingências 2x2 para cada par de termos como resposta. O algoritmo usa essa informação para computar uma medida binária, a medida de similaridade de *Jaccard*, para cada par de termos.

O conjunto todo forma uma grande matriz de similaridades: cada coeficiente representa a "distância" entre um par de termos. A matriz de similaridades é agrupada hierarquicamente e coloca os agrupamentos em um número específico de categorias. O algoritmo usado para criar as categorias é uma variação de *clustering* hierárquico. Esse algoritmo tenta produzir agrupamentos cujas maiores distâncias entre quaisquer dois membros seja a menor possível. Ele tende a produzir agrupamentos compactos.

2.5.5 Smart Discovery

Smart Discovery [107] (*Inxight*) extrai metadados a partir de documentos e combina esses metadados com interfaces de procura familiares e opções de navegação intuitivas. Dessa forma, o Smart Discovery permite que usuários encontrem e empreguem rapidamente a informação de forma relevante e precisa.

Smart Discovery permite:

- classificar dados não-estruturados em tópicos organizados e taxonomias de eventos, fornecendo uma alternativa para as meras procuras por palavra-chave pela fontes de dados, independente da localização, língua ou formato.

- extrair as entidades mais relevantes contidas nos dados em formato de textos – pessoas, lugares, companhias, datas e eventos – e interagir com elas para localizar a informação mais relevante às suas necessidades.
- identificar os relacionamentos e ligações entre pessoas, organizações, e outras entidades dentro dos conjuntos de dados em formato de textos.
- recuperar resultados de busca relevantes, mesmo com consultas de uma palavra, e interagir com resultados da busca.

2.5.6 VizServer

VizServer [108] (*Inxight*) fornece métodos para explorar grandes coleções de informação. O *Vizserver* possui visualizadores potentes que permitem aos usuários procurar e localizar informações e padrões nos conjuntos de dados relacionais, tabulares, hierárquicos, etc.

VizServer permite aos usuários localizar a informação específica que se está procurando 62,5% mais rápido que os métodos de navegação padrões. Essa capacidade aumenta significativamente a produtividade e permite decisões de negócios mais ágeis.

2.5.7 Knowledge Discovery System

Knowledge Discovery System [109] software baseia-se na idéia de "lugares" virtuais onde usuários podem organizar informação, serviços. Ele fornece também ferramentas para satisfazer necessidades particulares, mas ao mesmo tempo mantém e atualiza informações em um contexto mais geral.

Para sustentar esse conceito, foram definidos dois componentes distintos de gerenciamento de conhecimento: agregação de conhecimento e descoberta de informação. Os dois princípios que regem a agregação de conhecimento são: (1) Indivíduos e grupos necessitam de lugares virtuais para trabalhar, tomar decisões, e agir. (2) Lugares virtuais devem incluir aplicações, serviços colaborativos e serviços pessoais.

Indivíduos ou grupos definem os requisitos para esses lugares virtuais, mas indivíduos nem sempre sabem o que não sabem. Ás vezes, apenas conversar com um especialista é a melhor forma de aprender. A descoberta de informação é uma forma de fornecer

acesso a toda a informação que é relevante em um ambiente corporativo que não tem conhecimento prévio de sua existência.

2.5.8 ThemeScape

ThemeScape [110] (Cartia) é um software de visualização de textos que lê automaticamente um grande número de documentos, reconhecendo o conteúdo da informação e organizando coleções de textos por tópico num mapa que é mostrado. Esse mapa pode ser aproximado para aumentar o nível de detalhamento. Nele se pode marcar documentos importantes e enviá-los para outras pessoas. *Themescape* apresenta uma nova forma de organizar informação que não utiliza nem diretórios de arquivos, nem árvores hierárquicas. Ao invés disso, cada documento é representado como um pequeno ponto no mapa topológico. O conceito é simples: quanto mais similares dois documentos são, mais próximos eles aparecem. O mapa apresenta picos onde existe uma alta concentração de documentos sobre o mesmo tópico, e a distância entre esses picos mostra o quanto os tópicos estão relacionados. Essa interface visual torna possível para o usuário saber rapidamente que informação está disponível, que tópicos dominam a coleção, e quais tópicos se interrelacionam.

Uma vez que os documentos são representados como pontos no espaço é possível mostrar milhares de documentos de uma só vez e utilizar a facilidade de aproximação (*zoom*) para revelar mais detalhes. Em qualquer região do mapa, basta clicar com o *mouse* que uma lista de documentos e conteúdos são abertos automaticamente na tela. Quando se aponta para qualquer documento, um pequeno resumo do texto é mostrado.

2.5.9 Data Junction

Data Junction [111] é uma ferramenta de transformação compreensiva que elimina completamente a necessidade de gerar, compilar e fazer o *link* de programas de forma a extrair e transformar dados. O ambiente contém um mecanismo que conecta os dados em seu ambiente nativo e executa transformações na memória. *Data Junction* possui uma interface que permite a criação de rotinas de transformação complexas.

O projeto da ferramenta baseia-se em três telas intuitivas. A primeira tela permite ao usuário definir e conectar às estruturas dos dados fonte. A segunda tela permite ao usuário de finir e conectar com a estrutura dos dados alvo. Ambas as telas de fonte e alvo permitem ao usuário ver e percorrer os dados e estruturas de dados uma vez que as conexões tenham sido estabelecidas. A terceira tela é onde o usuário define o relacionamento entre fonte e alvo. A tela mapa disponibiliza:

- Mapeamento visual: A funcionalidade de "*drag and drop*" permite ao usuário comparar e integrar campos dos dados fonte com estruturas dos dados alvo, arrastando-as para seus lugares.
- Manipulação de dados: o relacionamento entre os campos fonte e alvo é tratado internamente como uma expressão lógica (isto é, campo-alvo=campo-fonte). Pode-se escolher a habilidade de manipular conteúdos dos dados diretamente ou refinando essa expressão. Como um adicional para usuários, *Data Junction* encapsulou um conjunto conhecido de funções de manipulação de dados em um construtor de expressões robusto.
- Filtragem e Sorteio: Existem duas formas de se operar a filtragem no *Data Junction*. Usuários podem filtrar registros fonte para um subconjunto do atual definindo parâmetros de amostragrem promovendo a obtenção de uma amostra da entrada, ou o construtor de expressões pode ser empregado para especificar critérios de extração lógica. Em adição à filtragem, *Data Junction* oferece habilidades de inspeção de dados.

2.5.10 TextAnalyst

TextAnalyst [112] é uma ferramenta para análise semântica, navegação, e procura em textos. Quando o *TextAnalyst* analisa um texto, ele determina que conceitos – palavras ou combinações de palavras – são mais importantes no contexto do texto investigado. Cada conceito é rotulado como um nó e a ele é associado um peso semântico numérico, que é a medida da probabilidade para que este conceito seja importante no texto estudado. Simultaneamente, *TextAnalyst* determina os pesos das relações entre conceitos individuais no texto e conceitos ligados às sentenças no texto original onde esses conceitos foram encontrados.

A estrutura resultante, chamada rede semântica, é um conjunto dos conceitos mais significativos destilados a partir dos textos analisados, junto com os relacionamentos semânticos entre esses conceitos no texto. A rede semântica é um gráfico que representa toda a informação mais importante a partir da investigação dos textos de uma maneira concisa.

2.5.11 Technology Watch

Technology Watch [113] (IBM/Synthema) é uma solução baseada em um processo qie consiste de dois passos: análise léxica e *clustering*.

A análise léxica é subdividida em análise semântica e gramatical. Ela reduz ambigüidades torna a freqüência das palavras descrevendo cada documento significativa. Assim, foram desenvolvidos léxicos e dicionários para analisar todas as frases, classificar palavras eliminando a ambiguidade e detectando seus sinônimos. É realizado também um processo de *stemming*.

O Processo de clustering utiliza um *tagging* morfo-sintático. Uma vez que os documentos tenham sido estruturados lingüisticamente, eles são agrupados utilizando o método de Análise de Dados Relacionais (Relational Data Analysis - RDA) [114].

2.5.12 Outras Ferramentas

Existem algumas outras aplicações como o *dt-search* [115] que pode buscar instantaneamente por gigabytes de textos em um microcomputador, rede, Internet ou Intranet. Ele pode também servir para pesquisa em publicações como busca instantânea de texto, de grandes coleções de textos a sites da Web ou CD/DVDs.

Existe, também, *WizDoc* [116], que é um mecanismo de busca baseado em conceito que indexa em torno de cem páginas por minuto. Ele suporta consultas em linguagem natural, retornando ao usuário páginas que se relacionam com o assunto. *WizDoc* também possui um método de busca por *strings*.

É capaz de dividir os documentos em tópicos, disponibilizando os resultados. *WizDoc* possui um corretor ortográfico e uma comparação fuzzy. Ele possui, também, *thesaurus* para sinônimos.

Existe, ainda, o *cMap* [117] que é a implementação variada de um tipo de rede neural conhecido como Self Organizing Map (SOM) [103]. *CMap* é uma implementação customizada que possui algumas otimizações em relação ao algoritmo de SOM básico. *CMap* descobre clusters e categorias dentros dos conjuntos de documentos para o qual a estrutura interna não é conhecida. Essas categorias são então mostradas em um gráfico do tipo tridimensional, para permitir que o usuário explore e percorra o conjunto de dados. Essas otimizações melhoraram o tempo de execução consideravelmente.

Uma outra ferramenta é o *SemioMap* [118]. Esse programa permite extrair relações importantes, mostrando como as frases interagem nos documentos e descobrindo significados que não estão aparentes.

O *DR-LINK* [119] é um sistema de consulta em linguagem natural que permite ao usuário fazer uma pergunta em inglês comum e receber como retorno uma lista de documentos relevantes. *DR-LINK* fornece alerta automático, visualização de dados e *feedback* de relevância e classificação de documentos.

As ferramentas descritas nesta parte da tese representam algumas das mais conhecidas ferramentas comerciais existentes no mercado. O objetivo desta seção é dar uma noção da variedade de abordagens existentes nas aplicações de *text mining*.

2.5.13 Tabela Comparativa

Uma tabela comparativa das ferramentas comerciais apresentadas na seção anterior é mostrada a seguir. Essa tabela resume suas principais características de implementação. Os campos em branco significam que a característica em questão não se manifesta na ferramenta avaliada.

Produto/ Aplicação	**Companhia/ Organização**	**Funções de Refino de Texto**	**Tipo de Abordagem**	**Funções de Destilação de Conhecimento**
iMiner	IBM	Recuperação de Info, sumarização	Baseada em Documento	*Clustering*, categorização
Smart Discovery	Inxight	Recuperação de Info, sumarização	Baseada em Documento	Classificação
dt-Search	DT-Search		Baseada em Documento	Indexação Procura por Thesaurus
TextSmart	SPSS Corporation		Baseada em Documento	Categorizaçao baseada em *clustering*
ThemeScape	Cartia		Baseada em Documento	*Clustering*, visualização
cMap	Canis		Baseada em Documento	Histogramas de termos *Clustering*,visualização
Technology Watch	IBM/ Synthema		Baseada em Documento	*Clustering*, visualização
VizServer	Inxight		Baseada em Documento	*Hyperbolic tree* Visualização
Temis	Temis Group	Recuperação de Info	Baseada em Conceito	*Clustering*, Extração, categorização
SemioMap	Entrieva		Baseada em Conceito	Visualização
Knowledge Discovery System	IBM	Recuperação de Info	Baseada em Conceito	
DR_LINK *CINDOR CHESS*	TextWise	Recuperação e Extração de Info	Baseada em Conceito	
WizDoc	WizSoft		Baseada em Conceito	Indexação
Content Extractor	Data Junction	Extração de Info	Baseada em Conceito	
TextAnalyst	Megaputer	Recuperação de Info, sumarização	Baseada em Conceito rede semântica	Classificação

Tabela 2.3. Ferramentas de *Text mining*

Capítulo 3

Descrição do Sistema

3 Descrição do Sistema

Este capítulo visa descrever todos os passos da composição da solução para *clustering* de dados em Português. Serão mostradas as linguagens utilizadas, algoritmos e bibliotecas adotadas para a composição dos módulos integrantes da solução disponibilizada.

3.1 Linguagem Perl

A linguagem Perl se originou e se disseminou em ambiente UNIX. Hoje em dia, porém, existem versões para ambiente windows. O nome é a abreviatura de "*Practical Extraction and Report Language*" (Linguagem prática de extração e relatório). É considerada uma linguagem recomendada para desenvolvimento de *scripts* para servidores web.

A linguagem perl foi escolhida para o desenvolvimento dos programas nesta tese por se tratar de uma linguagem eficiente e de fácil manutenção na programação de uma ampla faixa de tarefas. Dentre as facilidades oferecidas pela linguagem Perl estão as relacionadas à manipulação de arquivos de texto, motivo principal da escolha desta linguagem.

A Perl é bastante sucinta, ou seja, poucas linhas de código podem descrever várias tarefas, sem significar linhas de código muito difíceis de ler ou complicadas para escrever.

3.1.1 Compilador e Interpretador

Normalmente, para executar um programa escrito em qualquer linguagem, é necessário um compilador ou um interpretador. Os dois tem suas vantagens:

- Um compilador gera um arquivo executável a partir das linhas de código programadas. O arquivo executável gerado pode então ser executado inúmeras vezes, copiado para outros computadores, e assim por diante, mesmo sem o código-fonte do programa. Durante a compilação do programa, os compiladores

tendem a executar otimizações elaboradas no código do programa de forma que o código final seja executado mais eficientemente.

- Um interpretador processa o código do programa linha a linha e executa as tarefas solicitadas pelo código neste mesmo instante. Depois que o programa foi escrito ele pode ser executado imediatamente, sem a necessidade de um estágio separado de compilação.

Existem vantagens e desvantagens nos dois métodos. O código compilado leva mais tempo para ser preparado, mas depois ele é rapidamente executado e o código fonte permanece secreto. O código interpretado fica pronto rapidamente para a execução, mas não é tão rápido quanto o código compilado.

A linguagem Perl é um compilador que tenta ser um interpretador. A Perl compila um código de programa em código executável antes de executá-lo, portanto existe um estágio de otimização e o código executável trabalha mais rapidamente. Entretanto, ele não grava esse código em um arquivo executável separado. Em vez disso, armazena-o na memória e depois, executa-o.

Desta forma, a Perl combina o ciclo de desenvolvimento rápido de uma linguagem interpretada com a execução eficiente de um código compilado. Existem, no entanto, as desvantagens correspondentes: a necessidade de compilar o programa cada vez que ele é executado significa uma inicialização mais lenta que a de uma linguagem puramente compilada e requer que os desenvolvedores distribuam o código-fonte para os usuários.
No entanto, atualmente existem programas capazes de converter um script Perl em um programa executável, como o Perl2Exe.

Outra característica importante da linguagem Perl é que além de portável (pode ser executada em diversas plataformas) permite a integração com módulos compilados, como por exemplo DLLs.
Esta característica pode ser explorada para aumentar o desempenho do sistema, implementado-se partes críticas, ou com muito cálculos matemáticos em C ou C++ por exemplo.

3.1.2 Conversor Script – Executável

O Perl2Exe [99] é um programa para a conversão de scripts Perl em arquivos executáveis. Ele permite que se crie programas independentes em Perl que não precisam do interpretador Perl, ou seja, geram-se arquivos executáveis que não precisam disponibilizar o código-fonte. O Perl2Exe pode gerar executáveis para servidores Windows e Unix.

3.2 Case Folding

Para facilitar a identificação correta de palavras, o primeiro passo do processo de pré-processamento dos dados textuais é a conversão para o mesmo tipo de caracter: maiúsculo ou minúsculo. No nosso caso, o tipo de caracter adotado foi o minúsculo.

3.3 Stemmers

A implementação de Porter [25,92] foi comparada com o algoritmo adotado, o *StemmerPortuguese* [93], no artigo [94]. Nesse artigo, o algoritmo adotado apresenta desempenho superior em praticamente todas as classes sintáticas.
A implementação do algoritmo de Porter para o Português [92] foi adaptada para Perl para utilização nesta tese, juntamente com o *StemmerPortuguese*. Empregou-se, no entanto, uma versão em Perl e não a original disponibilizada em C.

3.3.1 Método de Porter

O algoritmo de Porter se encontra disponível em várias páginas da internet. Este algoritmo é considerado como referência no emprego do processo de *stemming* para a língua inglesa. Uma variação deste algoritmo foi desenvolvida para o Português [92] e é usada como base neste trabalho para a implementação em Perl. A seguir, apresentam-se alguns conceitos básicos utilizados no algoritmo de Porter implementado.
Este algoritmo faz uso das definições de regiões R1 e R2. Estas regiões são definidas como a seguir:

- *R1* é a região depois da primeira não vogal seguindo a vogal, ou é a região vazia no fim da palavra se não existe tal não-vogal.
- *R2* é a região depois da primeira não-vogal seguindo a vogal em R1 ou é a região vazia no fim da palavra se não existe tal não-vogal.

A definição de não-vogal varia em cada idioma. Em italiano, por exemplo, o *i* entre duas outras vogais não é uma vogal. O que deve ser considerado vogal deve ficar bem explícito para a versão de *stemmer* em cada idioma.

A seguir, exemplos de R1 e R2 são mostrados para algumas palavras em português

B a i l a r i n a

|<----------->| R1

|<-------->| R2

A letra *l* é a primeira não-vogal seguindo a vogal em bailarina, assim R1 é **arina.** Em R1, r é a primeira não vogal seguindo a vogal. Assim R2 é **ina**.

B a i l e

|<->| R1

>|< R2

Novamente, a letra *l* é a primeira não-vogal seguindo a vogal, assim R1 é apenas a última letra, e. R1 não contém nenhuma não-vogal, assim R2 é a região vazia ao final da palavra.

B ó i a

>|< R1

>|< R2

Em bóia R1 e R2 são ambos nulos.

Outros exemplos:

A m i g á v e l

|<---------->| R1

|<--->| R2

I n d e p e n d e n t e

|<---------->| R1

|<---------->| R2

3.3.1.1 O algoritmo de *Stemming* de Porter

Outras considerações devem ser feitas para contornar problemas inerentes à Lingua Portuguesa. Um destes problemas está relacionado à acentuação. As letras em Português incluem as seguintes formas acentuadas, ***á é í ó ú â ê ô ç ã õ ü***

As seguintes letras são vogais: ***á é í ó ú â ê ô***

E as duas formas de vogais anasaladas, ***ã õ***
devem ser tratadas como uma vogal seguida de uma consoante. ***ã*** e ***õ*** são portanto substituídas por ***a~*** e ***o~*** na palavra, onde**~** é um caracter separado a ser tratado como uma consoante. E assim - *R*2 (veja a nota em *R*1 e *R*2) e *RV* tem a mesma definição que no *stemmer* para Espanhol. RV é definido como a seguir:

Se a segunda letra é uma consoante, RV é a região depois da próxima vogal a seguir, ou se as primeiras duas letras são vogais, RV é a região depois da próxima consoante, e de outra forma (caso consoante-vogal) RV é a região depois da terceira letra. Mas RV é o fim da palavra se essas posições não puderem ser encontradas.

Por exemplo:

m a c h o	o l i v a	t r a b a l h o	á u r e o
|.....|	|….|	|……..|	|....|

Sempre faça Passo 1.

Descrição do Passo 1 – Remoção de sufixo padrão

Procure pelo mais longo entre os seguintes sufixos, e execute a ação indicada.

eza ezas ico ica icos icas ismo ismos ável ível ista istas oso osa osos osas amento amentos imento imentos adora ador ança~o adoras adores aço~es

apague se em *R*2

logía logías

substitua com ***log*** se em *R*2

ência ências

substitua com ***ente*** se em *R*2

amente

apague se em *R*1

se precedido por ***iv***, apague se em *R*2 (e se precedido por ***at***, apague se em *R*2), de outra forma,

se precedido por ***os***, ***ic*** ou ***ad***, apague se em *R*2

mente

apague se em *R*2

se precedido por ***avel*** or ***ível***, apague se em *R*2

idade idades

apague se em *R*2

se precedido por ***abil***, ***ic*** or ***iv***, apague se em *R*2

iva ivo ivas ivos

apague se em *R*2

se precedido por ***at***, apague se em *R*2

ira iras

substitua com ***ir*** se em *RV* e precedido por ***e***

Faça passo 2 se nenhum final foi removido pelo passo 1.

Descrição do Passo 2 – Sufixos de Verbos

Procure pelo mais longo entre os seguintes sufixos em *RV*, e se encontrado, apague.

ada ida ia aria eria iria ará ara erá era irá ava asse esse isse aste este iste ei arei erei irei am iam ariam eriam iriam aram eram iram avam em arem erem irem assem essem issem ado ido ando endo indo ara~o era~o ira~o ar er ir as adas idas ias arias erias irias arás aras erás eras irás avas es ardes erdes irdes ares eres ires asses esses isses astes estes istes is ais eis íeis aríeis eríeis iríeis áreis areis éreis ereis íreis ireis ásseis ésseis ísseis áveis ados idos ámos amos íamos aríamos eríamos iríamos áramos éramos íramos ávamos emos aremos eremos iremos ássemos êssemos íssemos imos armos ermos irmos eu iu ou ira iras

Se o último passo a ser obedecido – ou passo 1 ou 2 – alteraram palavra, faça passo 3

Descrição do Passo 3

Apague sufixo ***i*** se em *RV* e precedido por ***c***

Alternativamente, se nem passo 1 ou 2 alteraram a palavra, faça passo 4

Descrição do Passo 4 – Sufixo residual

Se a palavra termina com um dos sufixos : ***os a i o á í ó***

em *RV*, apague-o

Sempre faça passo 5

Descrição do Passo 5

Se a palavra termina com: ***e*** ***é*** ***ê***

em *RV*, apague, e se precedida por ***gu*** (ou ***ci***) com o ***u*** (ou ***i***) em *RV*, apague o ***u*** (ou ***i***).

Ou se a palavra termina com ***ç*** remova a cedilha

E finalmente:

Mude ***a~***, ***o~*** de volta para ***ã***, ***õ***

Durante a implementação do sistema, de forma a melhorar os resultados obtidos, foram realizadas as seguintes modificações no algoritmo de Porter original:

- RV = R1
- sufixos ordenados pelo comprimento

Esta modificações foram propostas a partir dos resultados obtidos com o conjunto de palavras de teste.

3.3.2 *Stemmer* Portuguese

O algoritmo adotado para o *stemming* de dados textuais em Português [93] é o *Stemmer Portuguese*, também conhecido por RSLP (Removedor de Sufixos para a Língua Portuguesa), que é apresentado a seguir. Implementado originalmente em linguagem C, o algoritmo é composto de oito passos que devem ser executados seguindo uma ordem definida. O fluxograma mostrado na figura 3.1 apresenta a seqüência que estes passos devem obedecer.

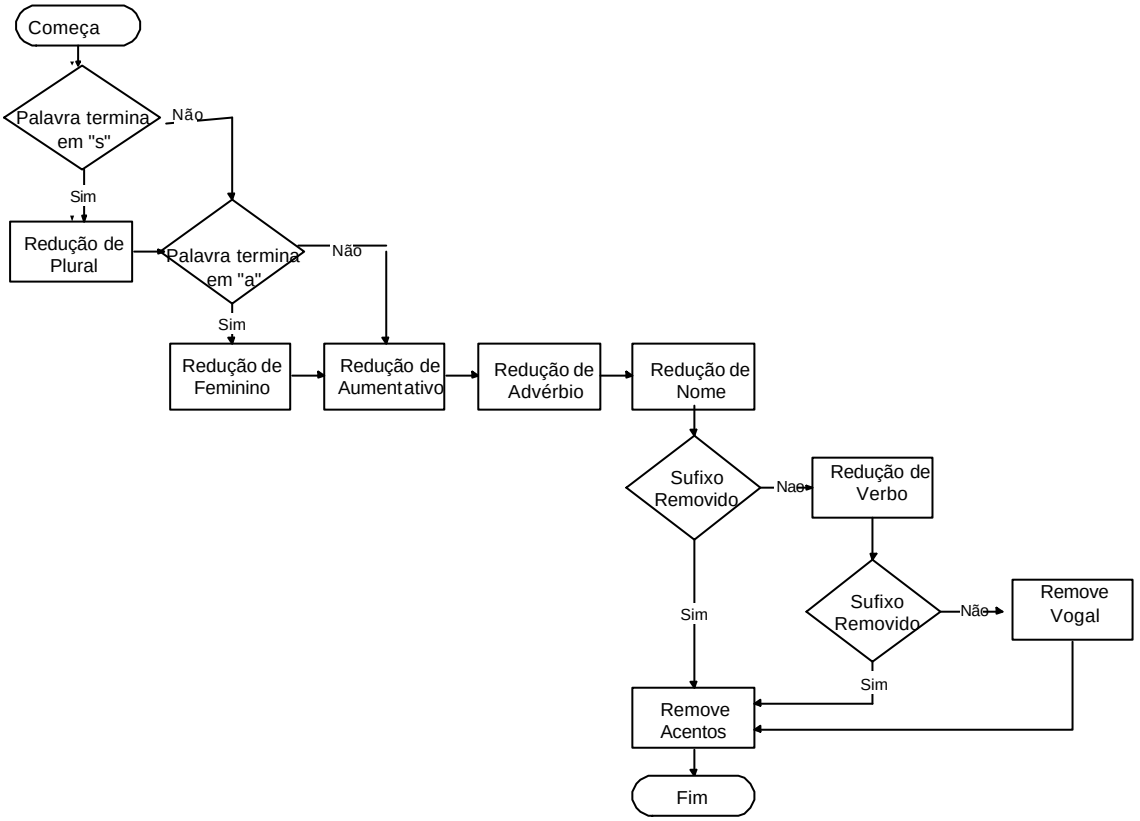


Figura 3.1 Seqüência de passos para o algoritmo de *stemming* RSLP

Em cada passo está contido um conjunto de regras. Porém, nem todas as regras são aplicadas. A cada passo as regras são examinadas e apenas uma regra é aplicada. O sufixo mais longo possível é sempre removido primeiro por causa da ordem das regras dentro de um passo, desse modo o sufixo plural –es deveria ser testado antes do sufixo –s. No momento, esse *stemmer* para o idioma Português contém 199 regras, que estão apresentadas no Apêndice B. Basicamente, cada regra mostra:

- O sufixo a ser removido;

- O comprimento mínimo do *stem*: isto serve para evitar remover um sufixo quando o *stem* é muito curto. Esta medida varia para cada sufixo, e os valores são definidos pela observação de listas de palavras terminando com o dado sufixo. Embora não exista suporte lingüístico para esse procedimento, ele reduz erros de *overstemming*.
- Um sufixo de reposição a ser anexado ao *stem*, se aplicável;
- Uma lista de exceções: para aproximadamente, todas as regras definidas, existiram exceções. Assim, uma lista de exceções foi adicionada para cada regra. Tais listas foram construídas com a ajuda de um vocabulário de 32000 palavras em Português disponível em [95]. Testes com o *stemmer* mostraram que listas de exceção reduzem erros de *overstemming* em 5%.

3.3.2.1 Algoritmo de *Stemming* - *StemmerPortuguese*

Note que os *stems* gerados não precisam ter necessariamente um significado, uma vez que eles são usados para indexar um banco de dados de documentos e não são apresentados ao usuário.

Descrição do Passo 1 – Redução de Plural

Com raras exceções, as formas de pural em Português terminam em –s. Entretanto, nem todas as palavras que terminam em –s denotam plural, por exemplo lápis. Esse passo consiste basicamente em remover o "s" final das palavras que não estão listadas como exceções. Mesmo assim algumas vezes umas pequenas modificações são necessárias, por exemplo, palavras terminadas em –ns deveriam ter este sufixo substituído por "m" como em bons → bom.

Descrição do Passo 2 – Redução de Feminimo

Todos os nomes e adjetivos em Português tem um gênero. Este passo consiste em transformar formas femininas em seus correspondentes masculinos. Apenas palavras terminando em –a são testadas neste passo mas nem todas elas são convertidas, apenas aquelas terminando nos sufixos mais comuns, por exemplo, chinesa → chinês

Descrição do Passo 3 – Redução de Advérbio

Este passo é o mais curto de todos, já que só existe um sufixo que denota advérbios –mente. Novamente, nem todas as palavras que terminam com –mente são advérbios, assim, uma lista de exceção é necessária.

Descrição do Passo 4 – Redução de Aumentativo/ Diminutivo

Nomes e adjetivos em Português apresentam muito mais variações que o equivalente em Inglês. As palavras tem formas aumentativas, diminutivas e superlativas, por exemplo, casinha, onde –inha é o sufixo que indica um diminutivo. Esses casos são tratados nesse passo. De acordo com [96] existem 38 desses sufixos, entretanto alguns deles são obsoletos. De modo a evitar o *overstemming*, o algoritmo de *stemming* adotado utiliza apenas os mais comuns que ainda são de uso comum.

Descrição do Passo 5 – Redução de Sufixo de Nome

Esse passo testa palavras em relação a 61 finais de nomes (e adjetivos). Se um sufixo é removido aqui, os passos 6 e 7 não são executados.

Descrição do Passo 6 – Redução de Sufixo de Verbo

O idioma Português é rico em termos de formas verbais, enquanto os verbos regulares em inglês tem apenas quatro variações (por exemplo, talk, talks, taked, talking), os verbos regulares em Português apresentam aproximadamente 50 formas diferentes [97]. Cada uma tem seu sufixo específico. Os verbos podem variar de acordo com o tempo, pessoa, número e modo. As estruturas das formas verbais podem ser representadas como: raiz+vogal temática+tempo+pessoa, por exemplo, and+a+ra+m. Formas verbais são reduzidas à sua raiz.

Descrição do Passo 7 – Remoção de Vogal

Esta tarefa consiste em remover a última vogal (“a”, “e”, “o”) das palavras que não tem sofreram *stemming* pelos passos 5 e 6, por exemplo a palavra menino, não sofreria

qualquer modificação proveniente dos passos anteriores, portanto este passo removerá seu –o final, de forma que ela possa ser reunida às outras variações tais como menina, meninice, meninão, menininho, que também serão convertidas ao *stem* menin.

Descrição do Passo 8 – Remoção de Acentos

Remover os acentos é necessário porque existem casos em que algumas formas de variações da palavra são acentuadas e outras não, como em psicólogo e psicologia, depois desse passo ambas as formas seriam unidas para psicolog. É importante que este passo seja feito nesse ponto e não no início do algoritmo porque a presença de acentos é significante para algumas regras, por exemplo, óis → ol transformando sóis em sol. Se ao invés disso a regra fosse ois → ol, ela cometeria erros como fazer o *stemming* de dois para dol.

3.3.2.2 Dificuldades no *Stemming* Português

O *stemming* do idioma Português é muito mais problemático que o *stemming* da língua Inglesa devido a sua morfologia mais complexa. Abaixo discute-se algumas dificuldades particularmente importantes que foram consideradas na implementação original do StemmerPortuguese conforme descritas a seguir:

- Lidando com exceções: Como mencionado anteriormente, uma das maiores dificuldades em construir esse algoritmo para o Português é que para quase toda regra formulada existem exceções, por exemplo "ão" é um sufixo comumente usado para denotar aumentativo, entretanto, nem todas as palavras terminadas em "ão" são aumentativos. Por essa razão, diferentemente do *stemmer* de Porter, houve a necessidade de listas de exceção. Se tivesse sido escolhido não usar essas listas, o *stemmer* cometeria erros de *overstemming* se a regra fosse mantida e *understemming* se a regra fosse abandonada.

- Homografia: Existem vários casos de homografia na Língua Portuguesa, muitos envolvendo verbos conjugados, por exemplo, casais que pode ser o plural de casal, ou a segunda pessoa do plural do presente do verbo "casar". Esse algoritmo não tem informação sobre categorias de palavras, assim os diferentes

sentidos dessas palavras não podem ser distinguidos. Para esse caso específico, o *stemmer* assume o primeiro significado e executa o *stemming* para sua forma singular casal, devido ao fato de que a segunda pessoa do plural ser quase obsoleta no Português moderno.

- Verbos irregulares: A versão corrente do *stemmer* não trata verbos irregulares, mas isso não parece afetar seriamente os resultados pelos testes efetuados em [93]. Os testes mostraram que menos de 1% dos erros ocorrem por causa desta razão.

- Mudanças para a raiz morfológica: Existem casos em que o processo de inflexão modifica a raiz da palavra. Os casos em que a mudança obedece regras ortográficas (por exemplo, ns $\rightarrow$ m) são tratadas com sucesso. Quanto aos demais casos, ainda não estão sendo tratados por este *stemmer*, por exemplo emitir e emissão, que são semanticamente relacionados, mas não estão sendo reduzidos ao mesmo *stem*, já que o primeiro é reduzido para emit e o segundo para emis.

- *Stemming* de Nomes Próprios: Nomes próprios não deveriam sofrer *stemming*, o problema está na dificuldade de reconhecê-los. Uma lista de nomes próprios não pode ser considerada uma solução ideal por duas razões principais: existem infinitas possibilidades e alguns nomes próprios também são compartilhados por nomes de coisas, por exemplo Pereira é um sobrenome comum em Português mas ele também significa "pé-de-pêra". Como no *stemming* de Porter, a atual versão desta implementação executa o *stemming* em nomes próprios.

3.4 Análise de Clusters

A análise de *Clusters* é uma importante atividade humana e normalmente forma a base de aprendizado e conhecimento. Um exemplo poderia ser o de uma criança que aprende a distinguir entre plantas e animais, ou ainda entre peixes e aves, melhorando esquemas de *clustering* subconscientes. Basicamente, o esquema é aprendido por observação das propriedades ou características (por exemplo, a presença de asas ou nadadeiras) dos objetos. Este tipo de propriedade binária é fácil de medir, mas algumas propriedades podem ser mais difíceis de mensurar. Um exemplo seria o de propriedades que são expressas em valores numéricos, como a altura de uma pessoa.

Clustering é aplicado a uma grande variedade de problemas em diversas áreas de pesquisa. No campo dos negócios *clustering* pode ser usado para ajudar pessoas da área de marketing a descobrir grupos distintos em suas bases e caracterizar grupos de consumidores baseando-se em padrões de compra. No campo da medicina, *clustering* de doenças, sintomas de doenças e curas para doenças, comumente levam a taxonomias interessantes. No campo da biologia, pode ser usado para categorizar genes com função similar e obter taxonomias de animais e plantas. No campo da psiquiatria, *clustering* tem ajudado médicos no correto diagnóstico de *clusters* de sintomas como paranóia, mania de perseguição, etc. Na arqueologia, pesquisadores freqüentemente aplicam técnicas analíticas de *cluster* para estabelecer taxonomias de ferramentas de pedra, objetos funerários, etc.

Clustering é um processo de particionar um conjunto de objetos de dados em um conjunto de subclasses significativas, chamadas *clusters*. Formalmente, consideremos uma coleção de *n* objetos decritos por um conjunto de *p* atributos. O processo de *clustering* visa obter uma divisão útil de *n* objetos em um número de *clusters*. Um *cluster* é uma œoleção de objetos de dados que são similares uns aos outros, baseados em seus valores de atributos, e assim podem ser tratados coletivamente como um grupo. *Clustering* é útil no entendimento da distribuição de um conjunto de dados.

Um algoritmo de *cluster* tenta encontrar grupos naturais nos dados baseados em similaridade de atributos. A seguir apresenta-se alguns requisitos típicos de *clustering* em mineração de dados [44]:

- Escalabilidade: Vários algoritmos de *clustering* podem trabalhar bem em pequenos conjuntos de dados; entretanto alguns deles podem falhar em manusear grandes conjuntos de dados. Uma solução imediata para este problema é executar o *clustering* em um subconjunto (ou amostra) de um grande conjunto de dados dado, mas isto pode levar a resultados tendenciosos. Uma outra solução é a utilização de algoritmos paralelos e distribuídos.

- Dimensionalidade Alta: Um banco de dados pode conter muitas dimensões ou atributos. A maior parte dos algoritmos de *clustering* trabalha bem em dados de dimensionalidade baixa, mas podem falhar em realizar o *clustering* de objetos em um espaço de dimensionalidade alta, especialmente quando os objetos de dados estão muito esparsos ou espalhados. Em conjunto de dados de dimensionalidade alta, normalmente não existem *clusters* naturais no espaço dimensional inteiro, apenas no subespaço formado por um conjunto de dimensões correlacionadas. Localizar *clusters* no subespaço pode ser desafiador. Um exemplo típico é o *clustering* de documentos que é também foco desta tese. Vários algoritmos de *clustering* simplesmente constróem uma nova dimensão para cada palavra distinta no conjunto de documentos. Devido aos grandes corpus em Português, o espaço usualmente contém acima de dez mil dimensões, o que reduz enormemente o desempenho do algoritmo. Este problema é também proximamente relacionado aos assuntos escalabilidade e eficiência.

- Forma arbitrária dos *clusters*: Vários algoritmos desempenham *clustering* baseados na medida de distância Euclidiana ou Manhattan. Algoritmos usando estes tipos de medida de distância tendem a encontrar *clusters* esféricos com densidade e tamanho similar. Esta limitação frequentemente degrada a exatidão. Essas duas métricas, em geral, não são adequadas para *clustering* de documentos.

- Insensibilidade à ordem dos dados de entrada: Alguns algoritmos de *clustering* são muito sensíveis à ordem dos dados de entrada. As soluções de *clustering* produzidas a partir do mesmo conjunto de objetos de dados pode ser completamente diferente dependendo de diferentes ordens dos dados de entrada.

Em outras palavras, a qualidade das soluções de *clustering* produzidas pelo mesmo conjunto de objetos de dados pode ser completamente diferente dependendo das diferentes ordens dos dados de entrada. Em outras palavras, a qualidade das soluções de *clustering* pode variar substancialmente e se tornar imprevisíveis.

- Manuseio de dados com ruído: *outliers* ou dados errôneos são um problema comum em bancos de dados. Um algoritmo de *clustering* robusto deveria minimizar o impacto deste ruído; de outro modo pode levar a uma acurácia de *clustering* pobre.

- Conhecimento prévio do domínio: Vários algoritmos de *clustering* requerem que o usuário especifique alguns parâmetros de entrada como o número de clusters k, o número de iterações, métricas, etc. Para determinar valores razoáveis desses parâmteros de entrada, algum conhecimento prévio do domínio é muitas vezes necessário. Entretanto, eles são difíceis de estimar em alguns casos, especialmente para conjuntos de dados contendo objetos de dimensionalidade alta. A acurácia do *clustering* pode degradar drasticamente se um algoritmo de *clustering* é muito sensível a esses parâmetros de entrada. Isto não apenas incomoda os usuários, mas também torna a qualidade do *clustering* difícil de controlar.

3.4.1 *Clustering* Hierárquico de Documentos

Clustering de Documentos é a organização automática de documentos em *clusters* ou grupos de modo que documentos dentro de um *cluster* tem similaridade alta comparando-se uns aos outros, mas são muito dissimilares a documentos em outros *clusters*. Em outras palavras, o agrupamento é baseado no princípio de maximização da similaridade intra-*cluster* e minimização da similaridade inter-*cluster*. O maior desafio do *clustering* é identificar eficientemente grupos significativos que são concisamente anotados.

Clustering é também chamado de aprendizado não-supervisionado porque aprende-se por "observação" e não por "exemplos".

Ao invés de produzir uma simples lista de *clusters*, o *clustering* hierárquico de documentos organiza os *clusters* em uma hierarquia ou árvore que facilita o *browsing*. O relacionamento pai-filho entre os nós da árvore pode ser visto como tópicos ou sub-tópicos numa hierarquia de assuntos.

O conceito de *clustering* hierárquico e a incapacidade dos métodos de *clustering* padrão formulam o objetivo desta pesquisa: Fornecer um método de *clustering* acurado, eficiente e escalável, que resolva problemas do *clustering* de documentos. A hierarquia de *clusters* resultante deveria facilitar o *browsing* e ser adequado para outros processamentos de outros algoritmos de mineração de dados.

Muitos algoritmos de *clustering* de documentos empregam vários passos de pré-processamento, incluindo a remoção de *stop words* e *stemming* no conjunto de documentos. Para fins de *clustering*, não faz diferença se os *stems* gerados são palavras genuínas ou não. *Stemming* não apenas aglomera variantes de um termo em uma forma única de representação, mas também reduz o número de termos distintos necessários para representar um conjunto de documentos. Um número menor de termos distintos resulta em economia de memória, espaço e tempo de processamento.
Cada documento é representado por um vetor de freqüências dos itens remanescentes dentro do documento. Esses vetores-documento formam um modelo vetorial no qual todas as operações de *clustering* são executadas. Existem tipicamente milhares de itens remanescentes depois da remoção de *stop words* e *stemming*. Em outras palavras, o espaço vetorial ainda teria uma dimensionalidade muito alta [76].

Como um passo extra do pré-processamento, vários algoritmos de *clustering* de documentos substituem a frequência de termos atual de um item por uma frequência balanceada, isto é, freqüência do termo (*term frequency*), - freqüência inversa do documento (*inverse document frequency*) – TF-IDF – no vetor documento. A idéia é que se um item é muito comum por todos os diferentes documentos, então ele teria um baixo poder discriminativo, e vice-versa [61]. Experimentos mostram que o peso TF-IDF aumenta a acurácia do *clustering* em todos os algoritmos testados.

$$w_{i,d} = TF_{i,d} \times \log(n/DF_i)$$

onde:

$TF_{i,d}$ = freqüência de um termo *i* em um documento *j*

n = número total de documentos

DF_i =número de documentos que contém o termo *i*

Assim como outros algoritmos de *clustering* de documentos, o método adotado emprega a remoção de *stopwords*, *stemming*, modelo vetorial, e TF-IDF.

Para *clustering* de documentos similares juntos, a maior parte dos algoritmos de *clustering* requer uma medida de similaridade entre dois documentos d_1 e d_2. Várias medidas possíveis são propostas na literatura, mas a mais comum é a medida cosseno [2], definida abaixo:

$$similaridade(d_1, d_2) = \cos seno(d_1, d_2) = \frac{(d_1 \bullet d_2)}{\|d_1\| \cdot \|d_2\|}$$

onde $\bullet$ representa o vetor produto escalar e $\| \ \|$ representa o comprimento de um vetor.

3.5 C-Clustering Library

Os algoritmos de *clustering* disponibilizados são provenientes da integração da *C-Clustering Library* [98] com o sistema desenvolvido. C-*Clustering Library* é um conjunto de rotinas que reunidas implementam os algoritmos de *clustering* que são mais comumente usados. Os algoritmos de *clustering* são:

- *Clustering* Hierárquico (*centroid- linkage, single-linkage, complete-linkage,* e *average-linkage*);
- *Clustering K-means*;
- *Self-Organizing Maps*;

Esta biblioteca foi escrita em ANSI C e, dessa forma, pode ser facilmente ligada a outros programas C/C++. Esta biblioteca é particularmente útil quando chamada de uma linguagem script tal como Perl [120], Python [121], ou Ruby [122]. A interface com a linguagem Perl é feita utilizando um wrapper disponível.

3.5.1 Manuseio dos Dados

A entrada para as funções de distância contém dois arrays e dois índices de linha ou de coluna, ao invés de dois vetores de dados. Isto permite calcular a distância entre dois casos (vetores) ou entre atributos na matriz de similaridades formada.

3.5.1.1 Pesos

Para a maior parte das funções de distâncias disponíveis na C-*Clustering Library*, um vetor peso pode ser aplicado. O vetor peso contém pesos para os elementos do vetor de dados. Se o peso para o elemento I é wi, então aquele elemento é tratado como ele ocorresse wi vezes nos dados. O peso não tem que ser um número inteiro. Para a correlação de Spearman e t de Kendall, discutida a seguir, os pesos não tem um significado bem definido e dessa forma não são implementados.

3.5.1.2 Valores Ausentes

É comum encontrar dentre os dados que serão levados à análise, valores de dados não-preenchidos. Neste caso, esse valores são indicados e não são levados em consideração para o cálculo das funções de distância.

3.5.2 Funções de Distância

Para o *clustering* de dados em grupos de itens similares, deve-se primeiro definir o que significa similar. Na C-*Clustering Library*, oito funções de distância estão disponíveis para medir a similaridade ou a distância entre os vetores de dados:

- Coeficiente de correlação de Pearson;

- Valor absoluto do coeficiente de correlação de Pearson;
- Correlação de Pearson descentralizada (equivalente ao cosseno do ângulo entre dois vetores de dados);
- Correlação de Pearson descentralizada absoluta (equivalente ao cosseno do menor ângulo entre dois vetores de dados);
- Correlação de Spearman;
- t de Kendall;
- Distância Euclidiana;
- Distância Euclidiana harmonicamente somada;
- Distância City-block.

As primeiras seis distâncias estão relacionadas ao coeficiente de correlação, enquanto as três outras estão relacionadas à distância Euclidiana.
Quando se lida com documentos, normalmente não se utiliza distância Euclidiana, pois, tradicionalmente não fornecem bons resultados. É comum a utilização do cosseno.

3.5.2.1 Coeficiente de Correlação de Pearson

O coeficiente de correlação de Pearson é definido como:

$$r = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

no qual $\bar{x}, \bar{y}$ são a média das amostras de x e y respectivamente, e s_x, s_y são o desvio padrão da amostra de x e y. O coeficiente de correlação de Pearson é uma medida para o quão bem uma linha reta pode ser adequado para um gráfico de dispersão (*scatterplot*) de x e y. Se todos os pontos no gráfico de dispersão repousam sobre um linha reta, o coeficiente de correlação de Pearson é ou +1 ou –1, dependendo se a inclinação da linha é positiva ou negativa. Se o coeficiente de correlação de Pearson é igual a zero, não existe correlação linear entre x e y.
A distância de Pearson é então definida como:

$$d_p \equiv 1 - r.$$

Uma vez que o coeficiente de correlação encontra-se entre –1 e 1, a distância de Pearson encontra-se entre 0 e 2.

Note que a correlação de Pearson automaticamente centraliza os dados pela subtração da média, e os normaliza pela divisão pelo desvio padrão. Essa normalização é útil em várias situações, porém existem casos em que a magnitude dos atributos precisa ser preservada.

3.5.2.2 Correlação de Pearson Absoluta

Tomando-se o valor absoluto da correlação de Pearson, encontra-se um número entre zero e um. Se o valor absoluto é um, todos os pontos no scatter plot repousam sobre uma linha reta ou com inclinação positiva ou negativa. Se o valor absoluto é igual a zero, não existe correlação entre *x* e *y*.

A distância é definida usualmente como:

$$d_A \equiv 1 - |r|,$$

onde r é o coeficiente de correlação de Pearson. Uma vez que o valor do coeficiente de correlação de Pearson encontra-se entre 0 e 1, a distância correspondente encontra-se entre 0 e 1 também.

3.5.2.3 Correlação descentralizada – Cosseno do ângulo

Em alguns casos pode ser preferível usar correlação descentralizada ao invés de coeficiente de correlação de Pearson convencional. A correlação descentralizada é definida como:

$$r_U = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i}{s_x^{(0)}}\right)\left(\frac{y_i}{s_y^{(0)}}\right),$$

onde

$$s_x^{(0)} = \sqrt{\frac{i}{n}\sum_{i=1}^{n} x_i^2}$$

$$s_y^{(0)} = \sqrt{\frac{i}{n}\sum_{i=1}^{n} y_i^2}$$

Esta é a mesma expressão que para o coeficiente de correlação de Pearson convencional, exceto que a média das amostras $\overline{x}, \overline{y}$ são definidas como iguais a zero. A correlação descentralizada pode ser apropriada se existe um estado de referência zero.

A distância correspondente ao coeficiente de correlação descentralizado é definido como

$$d_U \equiv 1 - r_U ,$$

onde r_U é a correlação descentralizada. Uma vez que o coeficiente de correlação descentralizado encontra-se entre -1 e 1, a distância correspondente encontra-se entre 0 e 2.

A correlação descentralizada é igual ao cosseno do ângulo de dois vetores de dados no espaço n-dimensional, e frequentemente é referido como tal. Deste ponto de vista, faz mais sentido definir a distância como o arco cosseno do coeficiente de correlação descentralizado.

3.5.2.4 Correlação descentralizada absoluta

Assim como a correlação de Pearson, pode-se definir a medida de distância usando o valor absoluto da correlação descentralizada:

$$d_{AU} \equiv 1 - |r_U|,$$

onde r_U é o coeficiente de correlação descentralizado. Uma vez que o coeficiente de correlação descentralizado encontra-se entre 0 e 1, a distância correspondente encontra-se entre 0 e 1, também.

Geometricamente, o valor absoluto da correlação descentralizada é igual ao cosseno entre as linhas de sustentação dos dois vetors de dados – isto é, o ângulo sem levar a direção dos vetores em consideração.

3.5.2.5 Correlação de Spearman

A correlação de Spearman [123] é um exemplo de uma medida de similaridade não-paramétrica. Ela é útil porque ela é mais robusta contra dados errôneos (*outliers*) que a correlação de Pearson.
Para calcular a correlação de Pearson, substitui-se cada valor dos dados pela sua faixa se os dados fossem ordenados em cada vetor por seu valor. Calcula-se então a correlação de Pearson entre os dois vetores de faixa ao invés de vetores de dados.

Se a correlação de faixa de Spearman é usada, os pesos não podem ser adequadamente aplicados aos dados, especialmente se os pesos não são necessariamente números inteiros. O cálculo da correlação de faixa de Spearman na C-*Clustering Library* portanto não leva em consideração nenhum peso. Como no caso da correlação de Pearson, pode-se definir uma medida de distância correspondendo à correlação de faixa de Spearman como

$$d_S \equiv 1 - r_S,$$

onde r_S é a correlação de faixa de Spearman.

3.5.2.6 t de Kendall

O t de Kendall é um outro exemplo de medida de similaridade não paramétrica. Ela é similar à correlação de faixa de Spearman, mas ao invés das próprias faixas, apenas as faixas relativas são usadas para calcular o t. Como na correlação de faixa de Spearman, os pesos são ignorados no cálculo. Pode-se definir uma medida de distância correspondente ao t de Kendall como:

$$d_K \equiv 1 - t.$$

Uma vez que o t de Kendall é definido de tal forma que se encontra entre –1 e 1, a distância correspondente estará entre 0 e 2.

3.5.2.7 Distância Euclidiana

A distância Euclidiana é definida como:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Nessa fórmula, os parâmetros x_i e y_i são subtraídos diretamente um do outro. Deveria-se dessa forma ter certeza que esses parâmetros estão normalizados adequadamente quando se usa a distância Euclidiana.

Diferentemente das funções de distância baseadas em correlação, a distância Euclidiana leva em consideração a magnitude das diferenças dos valores dos dados. Dessa forma, ela preserva mais informação sobre os dados e pode ser preferível.

Entretanto, embora seja uma medida comum para dados convencionais, esta métrica e suas variações, como a Distância Euclidiana Harmonicamente Somada não são consideradas propícias para dados textuais, apresentando resultados pobres quando comparadas a outras métricas.

3.5.2.8 Distância Euclidiana Harmonicamente Somada

Distância Euclidiana Harmonicamente Somada é a variação da distância Euclidiana, onde os termos para as diferentes dimensões são inversamente somados (similar à média harmônica):

$$d = \left[\frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{x_i - y_i} \right)^2 \right]^{-1}$$

A distância Euclidiana harmonicamente somada é mais robusta contra dados errôneos quando comparada à distância Euclidiana. Note que a distância Euclidiana harmonicamente somada não é uma métrica. Por exemplo considere:

$$\underline{u} = (1,0);$$
$$\underline{v} = (0,1);$$
$$\underline{w} = (1,1).$$

Isto fornece $d(u,v) = 1$ enquanto $d(u,w) + d(w,v) = 0$

3.5.2.9 Distância City-block

A distância City-block, também conhecida como Manhattan distance, é relacionada à distância Euclidiana. Assim como a distância Euclidiana corresponde ao comprimento do caminho mais curto entre dois pontos, a distância city-block é a soma das distâncias ao longo de cada dimensão:

$$d = \sum_{i=1}^{n} |x_i - y_i|.$$

Isto é igual à distância que se deveria caminhar entre dois pontos em uma cidade, onde se tem que caminhar por quadras da cidade. A distância city-block é uma métrica que como a distância Euclidiana, possui parâmetros que são subtraídos uns dos outros diretamente, e deve-se ter o cuidado de que os parâmetros sejam normalizados.

3.5.2.10 Calculando a distância entre *clusters*

Nos métodos de *clustering* hierárquicos, a matriz de distâncias é primeiramente calculada, e em passos sucessivos do algoritmo a nova matriz de distâncias é calculada a partir da martiz de distâncias anterior. Em alguns casos, entretanto, gostaria-se de calcular a distância entre *clusters* diretamente, dados seus membros.

A distância entre dois *clusters* pode ser definida de várias maneiras. A distância entre os meios aritméticos de dois *clusters* é usada no *clustering* de pares centroid-linkage e no *clustering* K-means. Para o último, a distância entre as médias de dois *clusters* pode ser usada alternativamente. A distância de pares mais curta entre elementos de dois *clusters*

é usada no *clustering* de pares single-linkage. No *clustering* de pares average-linking, a distância entre dois *clusters* é definida como a média sobre as distâncias de pares.

3.5.2.11 Matriz de Distâncias

Na etapa de *clustering*, um passo considerado básico é o cálculo da matriz de distâncias. Essa matriz contém todas as distâncias entre os itens que estão sendo agrupados. Uma vez que as funções de distância são simétricas, a matriz de distâncias também é simétrica. Além do mais, os elementos da diagonal são zero, assim como a distância de um item para ele mesmo é zero. A matriz de distâncias pode, desse modo, ser armazenado em um array, com o número de colunas em cada linha igual ao número da linha (zero-offset). A distância entre os itens *i* e *j* é armazenada no localização [*i*] [*j*] se *j* < *i*, em [*j*] [*i*] se *j* > *i*, enquanto ela é zero se *j*=*i*. Note que a primeira linha da matriz de distâncias é vazia. Ela é incluída para conveniência computacional, uma vez que incluir uma linha vazia requer um armazenamento mínimo.

3.5.3 Algoritmos Particionais

Algoritmos particionais dividem itens em K *clusters* de tal forma que a soma das distâncias dos itens aos centros de seus *clusters* é mínima. O número de *clusters* K é especificado pelo usuário. Dos algoritmos particionais disponíveis na C-*Clustering Library*, foram utilizados:

- *Clustering* K-means
- *Clustering* K-medians

Esses algoritmos diferem em como o centro do *cluster* é definido. No *clustering* K-means, o *cluster* é definido como o vetor médio dos dados ponderado por todos os itens no *cluster*. Ao invés de vetor médio, no *clustering* K-medians a média é calculada para cada dimensão do vetor de dados. O algoritmo de *clustering* é adequado para casos no quais a matriz de distâncias é conhecida, mas a matriz de dados originais não está disponível.

O algoritmo maximização de expectativa (Expectation-Maximixation - EM) é comumente usado para encontrar o particionamento em k grupos. O primeiro passo no

algoritmo EM é criar K *clusters* e associar randomicamente itens a eles. A seguinte iteração é então feita:

- Calcula-se o centróide de cada *cluster*;
- Para cada item, determina-se qual centróide de *cluster* é o mais próximo;
- Reassocie o item àquele *cluster*.

A iteração é terminada se não há mais reassociações de itens acontecendo.

Como a associação inicial dos itens aos *clusters* é feita randomicamente, usualmente uma solução diferente de *clustering* é encontrada cada vez que o algoritmo EM é executado. Para encontrar a solução ótima de *clustering*, o algoritmo K-means é repetido várias vezes, cada vez começando de um *clustering* randômico inicial diferente. A soma das distâncias dos itens para os centros dos *clusters* é guardada para cada execução, e a solução com o menor valor dessa soma será retornada como a solução de *clustering* geral.

A freqüência com que o algoritmo de EM deve ser executado depende do número de itens sendo agrupados. Como regra, pode-se considerar a freqüência com que a solução ótima foi encontrada. Esse número é retornado pelos algoritmos particionais como implementado na biblioteca. Se a solução ótima foi encontrada muita vezes, é improvável que soluções melhores que aquela encontrada existam. Entretanto, se a solução ótima foi encontrada apenas uma vez, pode existir outras soluções com uma soma de distância intra-*cluster* menor.

3.5.3.1 Inicialização

O algoritmo K-means pode ser inicializado pela associação randômica de itens aos *clusters*. Um cuidado especial seria assegurar-se que nenhum *cluster* vazio é produzido. Isto é feito primeiro escolhendo-se K itens randomicamente e associando-se cada um deles a um *cluster* diferente. Os itens remanescentes são então randomicamente associados aos *clusters*. Cada *cluster* é assim garantido de conter ao menos um item.

3.5.3.2 Encontrando o centróide do *cluster*

O centróide de um *cluster* pode ser definido de várias maneiras. Para o *clustering* K-means, o centróide de um *cluster* é definido como o vetor médio de todos os itens num

cluster para cada dimensão separadamente. Para robustez contra dados errôneos, no *clustering* K-medians a média para cada dimensão é usada ao invés do vetor médio. A C-*Clustering Library* fornece rotinas para calcular o vetor médio do *cluster* e a média do *cluster*.

3.5.3.2.1 Encontrando o vetor médio do *cluster*

A rotina calcula o centróide dos *clusters* calculando-se o vetor médio para cada dimensão separadamente para todos os objetos de um *cluster*. Valores ausentes não são incluídos no cálculo do vetor médio. Se o meio do *cluster* tiver um valor ausente, será armazenado no array cmask. Se para o *cluster i* os valores de dados para a dimensão *j* estão ausentes para todos os itens, então cmask [*i*] [*j*] (ou cmask [*j*] [i] se transpose = =1) é definido como zero. De outro modo, é definido como 1.

3.5.3.2.2 Encontrando a média do *cluster*

A rotina calcula os centróides dos *clusters* calculando a média para cada dimensão separadamente por todos os itens num *cluster*. Valores de dados ausentes não são incluídos no cálculo da média. Se a média do *cluster* tiver um valor ausente, será armazenado no array cmask. Se para o *cluster i* os valores de dados para a dimensão *j* estão ausentes para todos os itens, então *cmask [i] [j]* (ou *cmask [j] [i]* se *transpose = =1*) é definido como zero. De outro modo, é definido como 1. Calcular a média pode tomar mais tempo que calcular o meio.

3.5.3.3 Algoritmo EM

O algoritmo EM como implementado na C-*Clustering Library* primeiro radomicamente associa itens aos *clusters*, seguindo por iteração para encontrar uma solução de *clustering* com a menor soma de distâncias intra-*cluster*. Durante a iteração, primeiro encontra-se os centróides de todos os *clusters*, onde os centróides são definidos em termos do vetor médio ou da média. As distâncias de cada objeto para os centros dos *clusters* são calculadas, e determina-se para cada objeto que *cluster* está mais perto. Reassocia-se, então, os itens para os seus *clusters* mais próximos e recalcula-se os centros dos *clusters*.

Todos os itens são primeiro associados antes de recalcular os centróides dos *clusters*. Isto tem duas consequências:

- Se não-checados, *clusters* podem se tornar vazios se todos os itens são reassociados. Para o *clustering* K-means e K-medians, a rotina EM mantém o registro do número de itens em cada *cluster* todas as vezes, e proíbe um item de ser reassociado a um *cluster* diferente se isto puder fazer com que o *cluster* corrente se torne vazio.

- Em princípio, a ordem no qual itens são reassociados a *clusters* não importam. Entretanto, uma vez que se force um item a ficar em um *cluster* se ele é o último item remanescente, para *clustering* K-means e K-medians precisa-se randomizar a ordem de qualquer maneira para assegurar que nem sempre os mesmos itens sejam forçados a ficar em um *cluster*.

O algoritmo EM termina quando não acontecem mais reassociações. Nota-se, entretanto, que para alguns conjuntos de associações iniciais de *cluster*, o algoritmo EM falha em convergir devido à mesma solução de *clustering* reaparecendo periodicamente depois de um pequeno número de passos iterativos. No algoritmo EM como implementado na C-*Clustering Library*, a ocorrência de tais soluções periódicas é checada. Depois de um determinado número de passos de iteração, o resultado corrente do *clustering* é guardado como uma referência. Comparando o resultado do *cluster* depois de cada passo de iteração subsequente com o estado de referência, pode-se determinar se um resultado de *clustering* previamente encontrado é achado. Em tal caso, a iteração é parada. Se depois de um dado número de iterações o estado de referência não tiver ainda sido encontrado, a solução de *clustering* corrente é guardada para ser usada como novo estado de referência. Inicialmente, dez passos de iteração são executados antes de salvar novamente o estado de referência. O número de passos iterativos é dobrado cada vez, para assegurar que o comportamento periódico com períodos mais longos possa ser detectado.

3.5.3.4 Encontrando a solução ótima

K-means e K-medians

A solução ótima é encontrada, executando o algoritmo EM repetidamente e salvando a melhor solução de *clustering* que foi retornada dessa rotina. Isto pode ser feito automaticamente chamando a rotina k*cluster*. A rotina para calcular o centróide do *cluster* e a função de distância são selecionados baseados nos argumentos passados no k*cluster*.

O algoritmo EM é então executado repetidamente, salvando a melhor solução de *clustering* que foi retornada por essas rotinas. Além disso, k*cluster* conta com que freqüência o algoritmo EM encontrou esta solução. Se ela foi encontrada várias vezes, pode-se assumir que não existem soluções possíveis com uma soma de distâncias intra-*cluster*. Se, entretanto, a solução foi encontrada apenas uma vez, pode ser que exista uma solução melhor.

3.5.4 Clustering Hierárquico

3.5.4.1 Métodos de *clustering* hierárquicos

Métodos de *Clustering* Hierárquicos são essencialmente diferentes do método de *clustering* K-means. Nos métodos de *clustering* hierárquicos os dados são arranjados segundo uma estrutura de árvore. Um dos passos básicos do processo de *clustering* é o cálculo da matriz de distâncias, especificando todas as distâncias entre os itens a serem agrpupados. A seguir, cria-se um nó pela união de itens ou nós baseados na distância entre eles, até que todos os itens pretençam ao mesmo nó. Pode-se criar uma estrutura de árvore traçando novamente que itens e nós foram unidos. Diferentemente do algoritmo EM, que é usado no *clustering K-means*, o processo completo de *clustering* hierárquico é determinístico.

Existem muitas variações de *clustering* hierárquico, que diferem em como a distâncias entre os sub-nós é definida em termos de seus membros. Na C-*Clustering Library* *single*, *maximum*, *average*, e *centroid linkage* estão disponíveis.

- No *clustering* de pares *single-linkage*, a distância entre dois nós é definida como a distância mais curta dentre as distâncias de pares entre os membros dos dois nós.

- No *clustering* de pares *maximum-linkage*, alternativamente conhecida como *clustering* de pares *complete-linkage*, a distância entre dois nós é definida como a distância mais longa dentre as distâncias de pares entre os membros dos dois nós.
- No *clustering* de pares *average-linkage*, a distância entre dois nós é definida como a média de todas as distâncias de pares entre os elementos dos dois nós.
- No *clustering* de pares *centroid-linkage*, a distância entre dois nós é definida como a distância entre seus centróides. Os centróides são calculados tomando-se o vetor médio de todos os elementos de um *cluster*. Como a distância de cada novo nó formado para os nós existentes e itens precisam ser calculados a cada passo, o tempo computacional de *clustering* de pares *centroid-linkage* pode ser significativamente mais longo que para outros métodos de *clustering* hierárquicos. *Clustering* de pares *centroid-linkage* é algumas vezes referido como *clustering* de pares average-linkage.

Para *clustering* de pares *single-*, *complete-*, e *average-linkage*, a distância entre dois nós pode ser encontrada diretamente a partir das distâncias entre itens individuais. Por essa razão, o algoritmo de *clustering* não precisa dos dados diretamente, uma vez que a matriz de distâncias seja conhecida. Para *clustering* de pares *centroid-linkage*, entretanto, os centróides dos novos sub-nós formados só podem ser calculados a partir dos dados originais e não a partir da matriz de distâncias.

3.5.4.2 Podando a árvore de *clustering* hierárquico

A estrutura de árvore gerada pela rotina de *clustering* hierárquico pode ser melhor analisada pela divisão dos dados em *n clusters*, onde *n* é um inteiro positivo menor ou igual ao número de elementos que foram agrupados. Isso pode ser alcançado ignorando os *n-1* eventos ligados no topo da estrtura de árvore, resultando em *n* sub-nós separados. Os elementos em cada sub-nó são então associados ao mesmo *cluster*. A rotina cuttree determina para qual *cluster* cada elemento é associado, baseado no resultado do *clustering* hierárquico armazenado na estrutura de árvore.

3.5.5 SOM

Self-Organizing Maps (SOM) [103] foi inventada por Kohonen e foi criada usando técnicas de redes neurais artificiais. Um conjunto de vetores é entrada para um mapa

(*map*) consistindo de unidades. Associado com cada unidade está um vetor peso, inicialmente consistindo de valores randômicos. As unidades respondem mais ou menos ao vetor de entrada de acordo com a correlação entre o vetor de entrada e o vetor peso da unidade. À unidade com a resposta mais alta permite-se "aprender", assim como algumas unidades da vizinhança. A vizinhança decresce em tamanho durante o período de treinamento. O aprendizado é feito ajustando-se os pesos das unidades em uma pequena quantidade para refletir melhor o vetor de entrada.

O resultado do treinamento é que um modelo de organização emerge no mapa. Unidades diferentes aprendem a responder a vetores diferentes do conjunto de entrada, e unidades próximas tenderão a responder a vetores de entrada que se assemelham. Quando o treinamento é encerrado, o conjunto de vetores de entrada é aplicado ao mapa mais uma vez, marcando para cada vetor de entrada a unidade (*cluster*) que responde de maneira mais forte (a mais similar) àquele vetor. Assim, SOM organiza vetores de entrada (itens) em unidades (*clusters)* que estão localizados em alguma topologia. Usualmente uma topologia retangular é escolhida. Os *clusters* gerados são tais que *clusters* vizinhos na topologia são mais similares uns aos outros que *clusters* longe uns dos outros na topologia.

Se documentos estão sendo agrupados, então o número de elementos em cada vetor de dados é igual ao número de termos na matriz de *clustering*.

SOM é então gerada tomando os documentos um de cada vez, e encontrando que *cluster* na topologia tem o vetor de dados mais próximo. O vetor de dados daquele *cluster*, assim como aqueles de *clusters* vizinhos, são ajustados usando o vetor de dados do documento em consideração.

O ajuste é dado por:

$$\Delta \underline{x}_{term} = t \cdot \left(\underline{x}_{doc} - \underline{x}_{term} \right)$$

O parâmetro t é um parâmetro que decresce a cada passo da iteração. É usada uma função linear simples do passo de iteração:

$$t = t_{init} \cdot \left(1 - \frac{i}{n} \right),$$

no qual t_{init} é o valor inicial de t como especificado pelo usuário i é o número do passo de iteração corrente, e n é o número total de passos de iteração a ser executado. Enquanto mudanças são feitas rapidamente no início da iteração, ao final da iteração apenas pequenas mudanças são feitas.

Todas os *clusters* dentro de um raio R são ajustados para o documento em consideração. Esse raio decresce conforme o cálculo progride com:

$$R = R_{\max} \cdot \left(1 - \frac{i}{n}\right),$$

no qual o raio máximo é definido como:

$$R_{\max} = \sqrt{N_x^2 + N_y^2},$$

onde (N_x, N_y), são as dimensões do retângulo definindo a topologia.

3.6 O Processo de *Clustering*

O *clustering* de textos em português é realizada em quatro etapas distintas.

- pré-processamento
- geração da matriz de *clustering*
- *clustering* propriamente dito
- visualização

A primeira etapa é conhecida como pré-processamento e corresponde às atividades preparação dos textos originais para o *clustering* e totalização dos termos.
Nesta etapa são realizadas diversas atividades, dentre elas a retirada de palavras, termos e símbolos indesejados, como por exemplo, pontuação e números. A atividade seguinte é a substituição dos sinônimos, seguida de *stemming* dos termos restantes. A etapa final é a totalização e sumarização de todos os termos presentes na lista de arquivos.
Para a atividade de *stemming* o sistema oferece como alternativas os algorítmos RSLP, Porter e nenhum.

Na segunda etapa, o sistema produz a matriz de *clustering* com os valores de peso a serem adotados para cada termo no correspondente arquivo.
Nessa matriz, as linhas correpondem aos arquivos (documentos) a serem agrupados e as colunas aos termos encontrados na lista de arquivos pré-processados.
As opções disponíveis para o cálculo do peso na matriz de *clustering* são TF (*Term Frequency*), IDF (*Inverse Document Frequency*) e TF-IDF (*Term Frequency * Inverse Document Frequency*).
Ainda como opção, o usuário pode definir o número de termos, dentre os mais freqüentes que serão utilizados na geração da matriz de *clustering*, bem como alterar a lista de termos a considerar.
Estas opções possibilitam ao usuário um maior controle sobre o processo de *clustering* subseqüente e pode ser repetida inúmeras vez sem que seja necessária uma nova realização da etapa de pré-processamento.

A terceira etapa corresponde ao *clustering* dos textos a partir da matriz de *clustering* produzida na etapa anterior.

Como opções são oferecidos os algoritmos de *clustering* hierárquico, k-means e SOM (*Self-Organizing Maps*).
Para o *cluster* hierárquico, o sistema ainda oferece as opções de *Centroid Linkage*, *Single Linkage*, *Complete Linkage* e *Average Linkage*. Para o algoritmo *K-Means* as opções para a distância são Coeficiente de correlação de Pearson, Correlação de Pearson descentralizada, Correlação de Pearson descentralizada absoluta, Valor absoluto do coeficiente de correlação de Pearson, Correlação de Spearman, t de Kendall, Distância Euclidiana, Distância Euclidiana harmonicamente somada e Distância City-block, todas descritas anteriormente.

A quarta etapa corresponde a visualização dos resultados. A técnica utilizada aqui é baseada na geração de gráficos de árvores hierárquicas do tipo dendrograma. Estes gráficos apresentam na lateral a árvore de relações dos arquivos e uma matriz central com cores que correspondem ao peso/ocorrência de cada termo no arquivo.
Este tipo de gráfico possibilita uma análise imediata e eficiente do processo de *clustering* e da relação entre os documentos, entre termos e entre termos e documentos.

3.6.1 Módulos do Sistema

Nessa seção são mostrados os módulos que constituem o sistema. São 5 os módulos principais como citados a seguir

- Pré-processamento
- Geração da Matriz de *Clustering*
- *Clustering*
- Visualização
- Interface

.
O primeiro módulo é o de pré-processamento e envolve os passos:

Pré-Processamento:

- *stoplists*
- *stopwords*

- sinônimos
- *stemming*:
    - RLSP
    - Porter

O módulo seguinte é o módulo gerador da matriz de *clustering* e envolve os seguintes passos:

Matriz de *Clustering*:

- lista de termos
- peso:
    - TF
    - IDF
    - TFIDF

Uma vez produzida a matriz de *clustering*, selecionados termos e respectivos pesos, passa-se a etapa seguinte que é a do processo de *clustering* propriamente dito. Pode-se optar por uma das três opções abaixo:

***Clustering*:**

- Hierárquico
- K-Means
- SOM

Ao final do processo de *clustering*, é gerado um arquivo do tipo .cdt que alimenta o módulo seguinte de visualização. Os métodos particionais também geram um arquivo do tipo amostra/*cluster* que permite avaliar o índice de acerto do método. O módulo de visualização transforma o arquivo fornecido pelo módulo de *clustering* em uma visualização do tipo dendrograma.

Visualização

- Gráfico em Árvore do tipo dendrograma

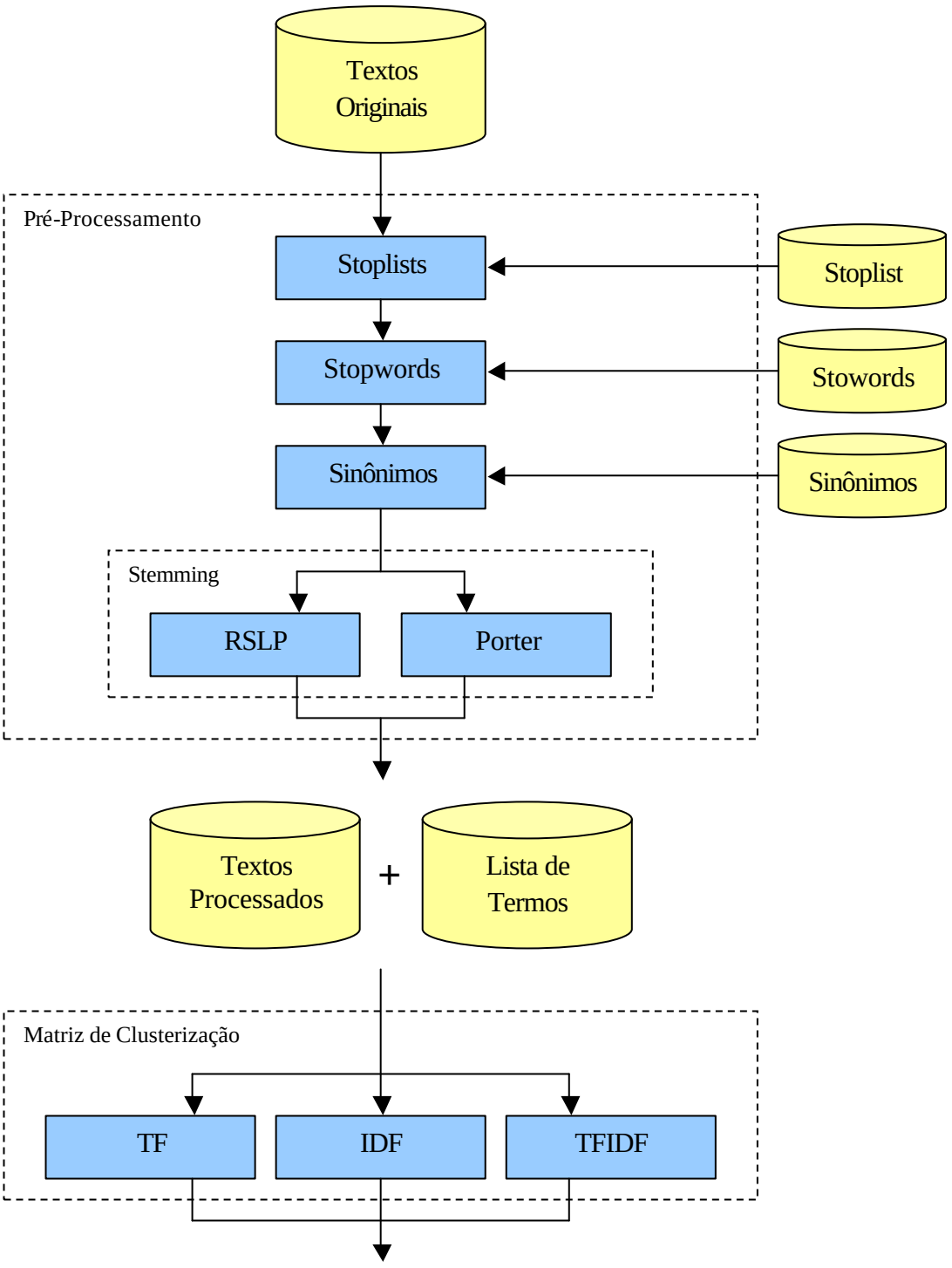
O último módulo é o de interface que envolve os passos mostrados a seguir:

Interface

- preparação dos dados
- acionamento dos módulos

3.6.2 Diagrama do Sistema

A figura 3.2 mostra um diagrama em blocos dos módulos do sistema, representando todas as etapas envolvidas desde o pré-processamento até a visualização dos resultados.



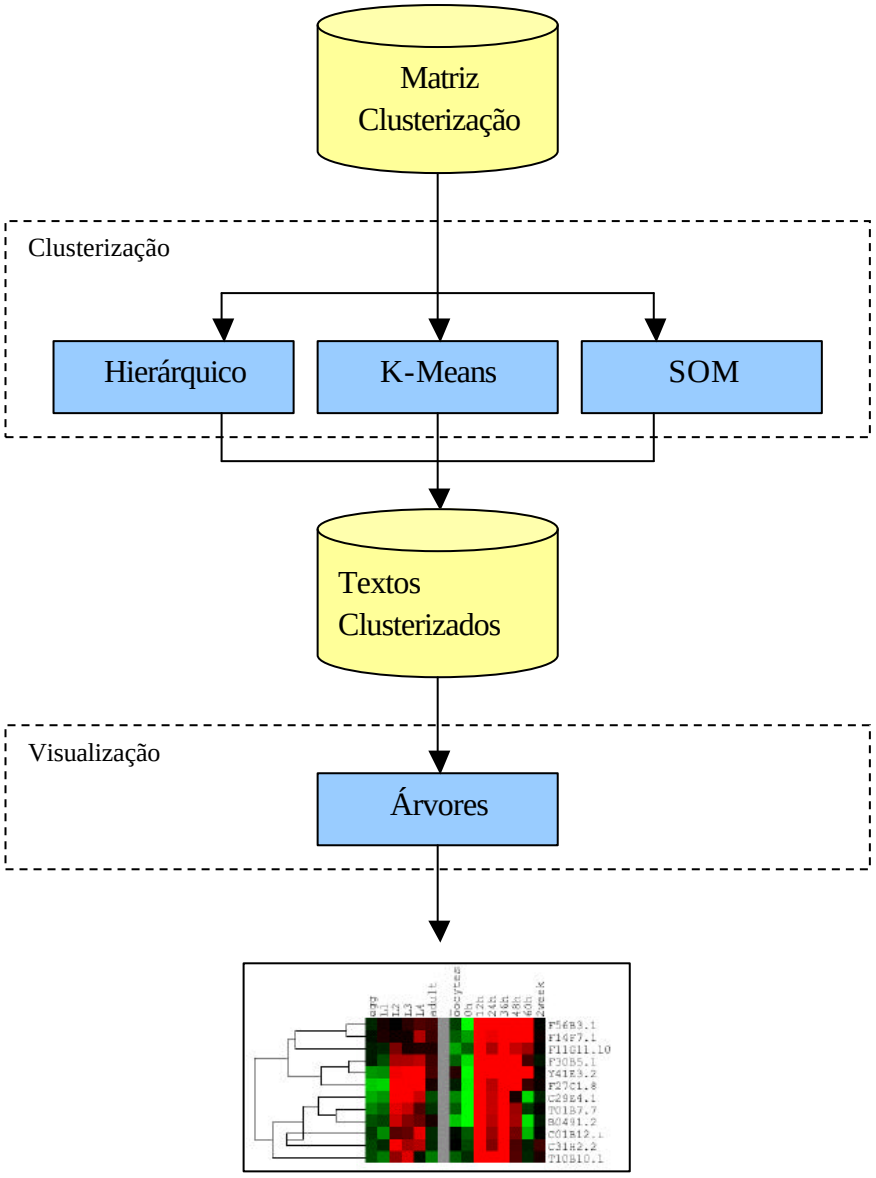


Figura 3.2 Diagrama em blocos do sistema

3.6.3 *Interface* com o Usuário

Os diversos módulos do sistema foram desenvolvidos e implementados para funcionarem independentemente uns dos outros.
De forma a facilitar o processo de *clustering* como um todo, foi dsenvolvido um módulo de interface que tem por objetivos, facilitar a preparação dos dados e o acionamento do diversos módulos de processamento de forma integrada.
O módulo de Inteface foi desenvolvido utilizando tecnologia Perl/TK e funciona perfeitamente integrado aos demais módulos do sistema além de garantir a portabilidade do conjunto.

O módulo de Interface oferece ao usuário um conjunto de diálogo gráfico para definição dos parâmetros e opções de execução, ferramentas para edição e seleção dos arquivos integradas e recursos de preenchimento automático de campos comumente usados (arquivo de configuração).

Mostra-se na figura 3.3, a tela de entrada do módulo de *clustering* com as opções disponíveis para o usuário.

Figura 3.3 Tela de entrada da interface de usuário do módulo de *clustering*

A tela da figura 3.3 mostra a interface do módulo de *clustering* que permite escolher o número de termos a considerar na análise, utilizar o *stemmer* de Porter, RSLP ou não utilizar e ainda o peso associado aos termos que podem ser TF, IDF, ou TF-IDF(mais utilizado).

A interface mostrada na figura 3.4 permite que sejam definidas algumas configurações como os arquivos de stopwords e sinônimos. Essa interface permite também editar os arquivos texto, para incluir ou retirar termos da análise ou modificar sinônimos.

Figura 3.4 Tela do módulo de configuração

O módulo de configuração atende tanto ao módulo de *clustering* quanto ao módulo de categorização assistida.

3.6.3.1 Perl/TK

Há muitos módulos diferentes disponíveis que estendem as funcionalidades do Perl. O módulo TK possibilita facilmente adicionar uma interface gráfica aos scripts Perl e ainda utilizar todos os recursos do Perl. Ao invés de exigir um comando digitado com algumas opções ou a entrada de dados em forma de linha de comando, o programa pode ser acionado a partir de um ícone, ou de um comando simples e a partir daí a interface manipula tudo: um conjunto de diálogos possibilita, de um modo fácil, a definição dos parâmetros e opções de execução.

Usando os módulos incluídos na distribuição do TK, pode-se criar janelas com botões, listas, textos, e outros tipos de *widgets* para ajudar o usúario na navegação dentro de uma aplicação.

Perl/TK utiliza os recursos de programação orientada a objeto disponíveis no Perl5 e foi convertido do TCL/TK para o uso com o Perl.

As grandes vantagens do Perl/TK são portabilidade e facilidade de implementação. O programa clássico do tipo "Hello, World" que em C facilmente chega a 100 linhas de código tanto no MS-Windows como em X-Windows pode ser feito com aproximadamente 5 linhas utilizando TK.

3.6.4 Fluxo de Trabalho

- Definição dos arquivos a serem agrupados
- Indicação dos arquivos com *stoplists*, *stopwords* e sinônimos
- Indicação do *stemming* a ser utilizado
- Indicação do número de termos a considerar
- Indicação do peso a ser utilizado
- Indicação do algoritmo de *cluster* e opção

De uma forma geral a intervenção do especialista no processo de *clustering* de textos em português pode ser realizada pela análise da lista dos termos presentes nos arquivos. Tal especialista poderá verificar quais desses termos realmente podem ter importância para a classificação do conjunto de arquivos em questão e através de alterações nesta lista obter um resultado mais adequado. Tais termos poderão ser apenas descartados para lista de termos a serem utilizados na geração da matriz de *clustering* ou transferidos para a lista de sinônimos ou mesmo *stopwords*.

3.7 O Processo de Categorização Assistida

A Categorização Assistida de textos em português é realizada em duas etapas distintas:

- Geração das Categorias
- Categorização por Similaridade

Na primeira etapa, cada arquivo da lista de arquivos da categoria é pré-processado e depois aglutinado em um único arquivo representando a categoria.

Nesta etapa, são realizadas diversas atividades, dentre elas a retiradas de palavras, termos e símbolos indesejados, como por exemplo, pontuação e números. A atividade seguinte é a substituição dos sinônimos, seguida do *stemming* dos termos restantes. Para

a atividade de *stemming* o sistema oferece como alternativas os algoritmos RSLP, Porter e nenhum.

Esta etapa também é conhecida como etapa de treinamento.

A etapa seguinte corresponde a categorização dos textos propriamente dita. Nessa etapa, cada arquivo é inicialmente pré-processado e depois categorizado segundo o algoritmo de similaridade selecionado. O sistema oferece como alternativas para os índices de similaridade as medidas de similaridade do Cosseno (*Cosine Similarity*), similaridade de *Jaccard* (*Jaccard Similarity)* e similaridade do Cosseno Ponderado (*Weighted Cosine Similarity*).

Para computar a similaridade do Cosseno (*Cosine Similarity*) entre dois documentos *D* e *E*, considera-se *Ds* e *Es* como sendo o conjunto de termos ocorrendo em *D* e *E* respectivamente. Define-se *T* como a união de *Ds* e *Es*, e considera-se t_i como o i-ésimo elemento de *T*.

Assim os vetores-termo de D e E são

Dv=(nD(t1),nD(t2),...,nD(tN))

Ev=(nE(t1),nE(t2),...,nE(tN))

Onde $nD(t_i)$ é o número de ocorrências do termo t_i em *D*, e $nE(t_i)$ o mesmo para E.

Agora, pode-se finalmente definir a similaridade de cosseno *CS*:

*CS = (Dv,Ev) / (Norm(Dv)*Norm(Ev))*

Aqui (... , ...) é o produto escalar e utiliza-se a norma Euclideana (raiz quadrada da soma dos quadrados).

Jaccard Similarity ou similaridade de Jaccard é definida como a seguir: dados dois documentos, *D* e *E*, faça *Ds* e *Es* ser o conjunto de termos que ocorrem em *D* e *E*, respectivamente. Defina *S* como a interseção de *Ds* e *Es* e *T* como sua união. Assim, a similaridade de Jaccard é o número de elementos de *S* dividido pelo número de elementos de *T*.

Para computar a similaridade do Cosseno Ponderado (weighted cosine similarity) entre dois documentos D e E, pela definição da similaridade de cosseno, os vetores termo de D e E são:

*Dv=(nD(t1)*w1,nD(t2)*w2,...,nD(tN)*wN)*

*Ev=(nE(t1)*w1,nE(t2)*w2,...,nE(tN)*wN)*

Os pesos são valores reais não negativos; cada termo tem associado um peso. Para alcançar a generalidade, pesos podem ser definidos como uma função. Por exemplo, um modo comum de definir pesos é o IDF (inverse document frequency)

No processo de categorização, o menor índice de similaridade define a categoria a qual o arquivo deverá pertencer.

3.7.1 Interface com o usuário

A figura 3.4 mostra a tela de entrada do módulo de categorização assistida com as opções disponíveis para o usuário.

Figura 3.5 Tela de entrada da interface de usuário do módulo de categorização

A tela da figura 3.5 mostra a interface do módulo de categorização que permite escolher utilizar o *stemmer* de Porter, RSLP ou não utilizar e o tipo de medida de similaridade: *cosine similarity* ou similaridade do cosseno, *weighted cosine similarity* ou similaridade do cosseno ponderado e *Jaccard similarity* ou similaridade de Jaccard .

3.7.2 Fluxo de Trabalho

- Definição dos arquivos que compõem as categorias
- Definição dos arquivos que deverão ser categorizados
- Indicação do arquivos com *stoplists*, *stopwords* e sinônimos
- Indicação do *stemming* a ser utilizado
- Indicação da similaridade a ser utilizada

De uma forma geral, a intervenção do especialista no processo de categorização de textos em português pode ser realizada pela análise da lista dos termos presentes nos arquivos. Tal especialista poderá verificar quais destes termos realmente podem ter importância para a categorização do conjunto de arquivos em questão e através de alterações nesta lista obter um resultado mais adequado.

3.8 Módulo de Visualização

Java Treeview é um programa complexo consistindo de aproximadamente 42.000 linhas de código. Ele foi inicialmente desenvolvido para fornecer um conjunto de vistas e funcionalidades com as quais usuários poderiam construir facilmente aplicações e testar idéias novas. Ao longo do tempo, o Java TreeView evoluiu para uma aplicação única que pode carregar arquivos de vários tipos diferentes. A principal vantagem de ser implementado em Java é ser independente de plataforma, facilitando a portabilidade. Nesta tese, o Java TreeView é utilizado como o módulo de visualização de todas as saídas geradas pelos métodos executados no processo de clustering da solução proposta. A figura 3.6 mostra o ambiente disponibilizado pelo Java TreeView.

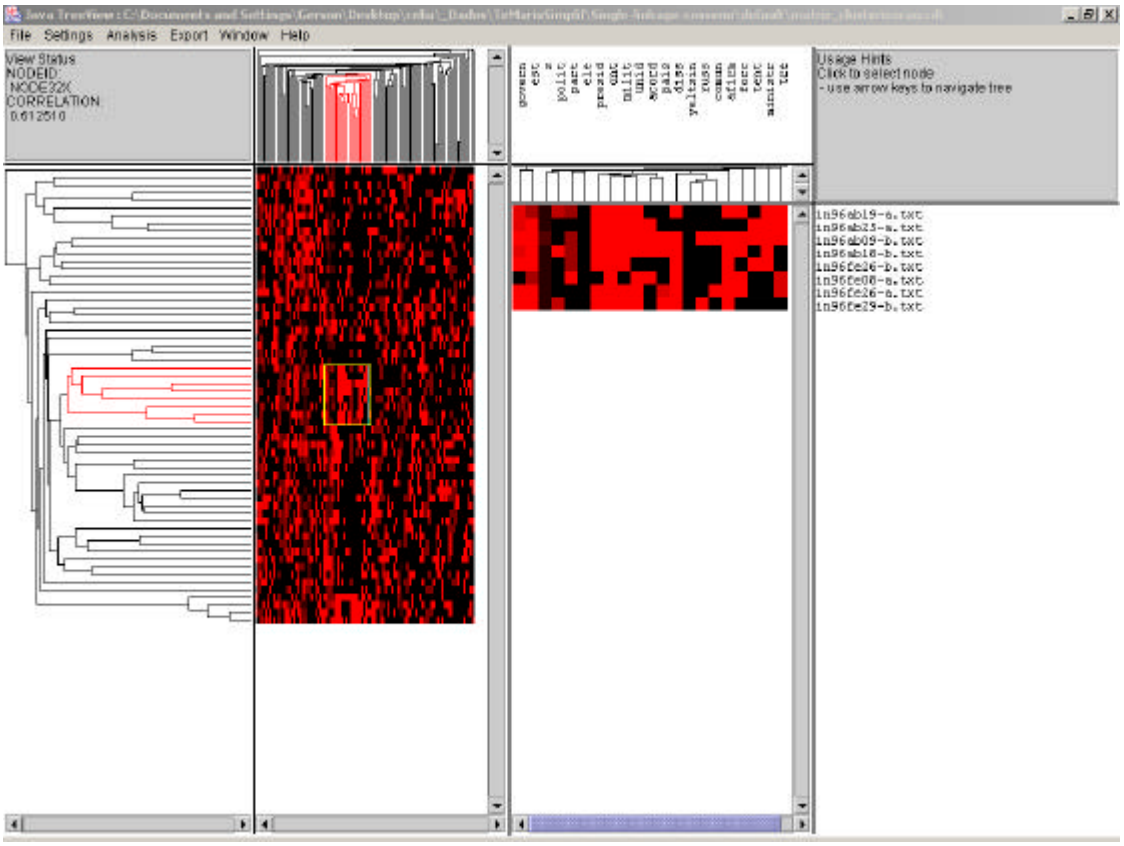


Figura 3.6 Módulo de Visualização – Java TreeView

Pode-se observar à esquerda da figura o dendrograma referente aos documentos e na parte superior o dendrograma referente aos termos. No centro da figura, encontra-se uma matriz de cores principal em que o preto corresponde ao zero e as tonalidades da cor escolhida (no caso o vermelho), representam as expressões dos termos nos

documentos, retirados diretamente da matriz de clustering após o clustering realizado. O retângulo assinalado no centro da matriz de cores denota o cruzamento de um cluster de termos com um cluster de documentos. Essa visualização hierárquica permite o entendimento de toda a organização de documentos e termos e das relações entre eles, em todos os níveis dos dendrogramas.

A idéia de uma visualização hierárquica mais detalhada foi motivada pela necessidade de refino em um estudo relacionado a um banco de dados de atendimento à clientes em que a ferramenta utilizada, embora realizasse o clustering hierárquico, não disponibilizava o dendrograma, dificultando o entendimento dos resultados apresentados. Além disso, não era possível ter controle sobre alterações promovidas nos termos, que se rearrumavam sem que se pudesse entender o porquê. O tipo de visualização disponibilizado pelo Java TreeView resolve as deficiências de visualização encontradas anteriormente.

Capítulo 4

Estudos de Casos

4 Estudos de Casos

A seguir serão apresentados os estudos de casos conduzidos para avaliação da qualidade dos resultados fornecidos pela ferramenta de *clustering* para Português. Serão utilizados conjuntos de textos (corpus) que possuem classificação prévia, porém, essa classificação será apenas utilizada para verificar a exatidão conseguida pelo processo de *clustering*. Dessa forma, a classificação prévia só será utilizada no pós-processamento dos resultados. No caso de tarefas de *clustering*, isso se torna de grande ajuda para que haja uma forma de avaliação segura da qualidade dos resultados obtidos.

4.1 Bases de Textos

Para fins dos estudos de caso que serão apresentados neste capítulo serão descritos a seguir os conjuntos de dados textuais utilizados, com suas características, fontes, assuntos, etc.

4.1.1 Corpus Conhecimentos Gerais

Este corpus foi montado a partir de um site na internet contendo diferentes seções para pesquisa escolar. O site Conhecimentos Gerais disponibiliza 17 assuntos diferentes, cada um contendo um conjunto de textos. Os assuntos e textos foram transportados para uma base local para serem utilizados na pesquisa.
Os assuntos disponíveis no corpus são mostrados a seguir.

Número de classes	Assunto	Número de textos
1	Artes Plásticas**	19
2	Astronomia	12
3	Biologia	7
4	Cinema	14
5	Cultura Popular	7
6	Ecologia	15
7	Física**	13
8	Geografia**	22
9	História do Brasil	53
10	Literatura	16
11	Matemática	5
12	Medicina	9
13	Música**	17
14	Química	10
15	Religião**	19
16	Teatro**	22
17	Tecnologia	9

Tabela 4.1. Características do corpus de conhecimentos gerais

As seções assinaladas com duplo asterisco indicam as seis classes que foram utilizadas primeiramente nos estudos de caso.

4.1.2 Corpus TeMario

Este corpus foi criado no âmbito do projeto EXPLOSA do NILC. Consiste em 100 textos jornalísticos, acompanhados dos respectivos sumários manuais e extratos ideais (gerados automaticamente).

O TeMário é composto de 100 textos jornalísticos coletados, totalizando 61.412 palavras, cujas origens e distribuições por assunto são apresentadas a seguir:

- 60 textos constam do jornal *on-line* Folha de São Paulo e estão distribuídos igualmente nas seções:
    - Especial,
    - Mundo e
    - Opinião;
- os 40 textos restantes foram publicados no Jornal do Brasil, também *on-line*, e estão também uniformemente distribuídos nas seções:
    - Internacional e
    - Política.

A Tabela 4.1 sintetiza esses dados, mostrando também o número de palavras por seção e o número médio de palavras por texto de cada seção.

Jornais	**Seções**	**Número de textos**	**Número de palavras**	**Média de palavras/texto**
Folha de São Paulo	Especial**	20	12.340	617
	Mundo	20	13.739	686
	Opinião**	20	10.438	521
Jornal do Brasil	Internacional**	20	12.098	604
	Política	20	12.797	639
	Total	**100**	**61.412**	639
	Média		**12.282**	613

Tabela 4.2. Características do corpus de textos-fonte

As seções assinaladas com duplo asterisco indicam as três classes que foram utilizadas primeiramente nos estudos de caso.

4.1.3 Corpus CETENFolha

O CETENFolha (**C**orpus de **E**xtratos de **T**extos **E**letrônicos **N**ILC/**Folha** de S. Paulo) é um corpus criado pelo projeto Processamento computacional do Português (projeto que deu origem à Linguateca) com base nos textos do jornal Folha de São Paulo que fazem parte do corpus NILC/São Carlos.

O corpus inclui o texto da Folha de S. Paulo do ano de 1994 (as 365 edições), incluindo cadernos não-diários.

O CETENFolha está dividido em extratos, classificados por semestre e caderno do jornal do qual provêm. Cada extrato está dividido em parágrafos e frases, e os títulos e os autores dos artigos estão assinalados. Embora os textos tenham sido identificados, a indicação fornecida, que é o caderno de origem da notícia, nem sempre corresponde ao assunto contido no texto.

Esse estudo utilizou 12396 extratos de 3 cadernos diferentes: Agrofolha, Informática, Folhateen, conforme mostra a tabela 4.2.

Seções	**Número de textos**
Agrofolha	3721
Informática	5231
Folhateen	3444
Total	**12396**

Tabela 4.2. Características dos extratos do corpus CETENFolha

4.2 Cluto

Cluto [100, 101] é um pacote para *clustering* de conjuntos de dados tanto de dimensões baixas como altas e para a análise de características dos vários *clusters*. Ele fornece três classes diferentes de algoritmos de *clustering* que operam ou diretamente no espaço de características dos objetos ou no espaço de similaridades de objetos. Esses algoritmos são baseados em paradigmas particionais, aglomerativos e de particionamento de grafos. Uma característica chave em quase todos os algoritmos de *clustering* do Cluto é que eles tratam o problema de *clustering* como um processo de otimização que procura maximizar ou minimizar um função de critério particular, definida localmente ou globalmente por todo o espaço de soluções. Os métodos de *clustering* do Cluto usados nessa tese são descritos suscintamente a seguir.

Não existe uma técnica de clustering que seja universalmente aplicada para descobrir a variedade de estruturas presentes num conjunto de dados multi-dimensionais [4]. Dessa forma, o Cluto foi utilizado para servir como termo de comparação com os resultados de *clustering* obtidos com o sistema desenvolvido.

4.2.1 Repeated Bisections

Nesse método, a solução de *clustering* desejada da forma K é computada executando uma seqüência de k-1 *repeated bisections*. Nessa abordagem, a matriz é primeiramente agrupada em dois grupos, e então um desses grupos é selecionado e mais uma vez dividido em dois. Esse processo continua até que o número de *clusters* desejado é encontrado. Durante esse passo, o *cluster* é dividido em dois (*bisected*) de forma que de forma que a solução de *clustering* de dois cursos otimiza uma função de critério de *clustering* particular, que no caso do Cluto é escolhida através do parâmetro crfun. Essa abordagem garante que a função de critério é otimizada localmente, dentro de cada bi-seção, mas em geral não é globalmente otimizada.

4.2.2 Direct

Nesse método, a solução de *clustering* é computada encontrando simultaneamente todos os *k clusters*. Em geral, computar um *clustering k-way* é mais lento que um *clustering* do tipo *repeated bisections*. Em termos de qualidade , para valores razoavelmente pequenos de k (usualmente menos que 10-20), a abordagem direta leva a *clusters* melhores que aquelas obtidas por meio do repeated bisections. Entretanto, conforme k cresce, a abordagemdo *repeated bisections* tende a ser melhor que o *clustering* direto.

4.2.3 Graph

Nesse método, a solução de *clustering* é computada modelando-se primeiramente os objetos usando o gráfico do vizinho mais próximo (cada objeto se torna um vértice, e cada objeto é conectado aos outros objetos mais similares), e então divide-se o grafo em *k-clusters* ussando um algoritmo de particionamento de grafos *min-cut*.

4.2.4 Agglo

Nesse método, a solução de *clustering* é computada usando o paradigma aglomerativo, cujo objetivo é otimizar localmente (minimizar ou maximizar) uma função de critério de *clustering* particular (que é selecionada usando o parâmentro *crfun*). A solução é obtida parando o processo de aglomeração quando restam *k clusters*. As opções de método aglomerativos disponíveis na ferramenta Cluto são: *single-linkage, complete-linkage, group average.*

4.3 Efeito do *Stemming* no Processo de *Clustering*

O estudo de caso visa determinar a influência do uso do *stemming* em relação à tarefa de *clustering* propriamente dita, executada na etapa seguinte ao pré-processamento. Como a utilização do *stemming* visa agregar as variações de uma mesma palavra de forma a somar contribuições, espera-se que o *stemming* consiga melhorar a qualidade dos dados fornecidos à etapa de clustering. A taxa de acerto dos diferentes métodos de *clustering* executados para as situações sem e com o uso de stemming serão avaliadas e comparadas.

Primeiramente, o *stemming* não foi realizado, num segundo momento foi aplicado o algoritmo de *Stemmer* de Porter, versão Português, e num terceiro momento foi aplicado o algoritmo *Stemmer Portuguese*, também chamado de RSLP. Dessa forma, as tabelas a seguir apresentam os resultados obtidos para os diferentes métodos de *clustering* para três situações distintas de *stemming*:

- Sem *stemmer*: Os dados são pré-processados utilizando apenas *stopwords*, dicionário, sem a utilização de nenhum *stemmer*.

- *Stemmer* Porter: Além dos passos de pré-processamento mencionados acima, utiliza-se o *stemmer* de Porter.

- *Stemmer Portuguese*: Somados aos demais passos do pré-processamento, utiliza-se o *stemmer Portuguese* no lugar do *stemmer* de Porter.

Foram utilizados, também, diferentes conjuntos de dados textuais. Além da variação na utilização do *stemming*, foi efetuada também uma variação na lista de termos produzida após o pré-processamento para as três situações descritas acima, gerando assim o conjunto de resultados chamado de "*default*" para a lista utilizada da forma como foi gerada; e o conjunto chamado "1modif" para a lista de termos após a primeira modificação, retirando-se alguns termos.

O peso utilizado para os termos em todos os experimentos apresentados é o TF-IDF. Selecionou-se como base 100 termos para a geração da matriz de *clustering* dessa parte do estudo.

4.3.1 Corpus Conhecimentos Gerais

A tabela 4.2 a seguir mostra os resultados obtidos para o conjunto de dados "Conhecimentos Gerais" para 6 classes primeiramente e depois utilizando mais onze classes. As seis classes iniciais usadas foram: Artes Plásticas, Física, Geografia, Música, Religião, Teatro.

Assim, além de avaliar o efeito do *stemming*, pode-se perceber a mudança ocasionada por um número maior de classes (inclusive superpostas).

	Métodos de Clustering Utilizados	Conhecimentos Gerais 6 classes		Conhecimentos Gerais 17 classes	
		default	1modif	default	1modif
Sem Stemmer	K-Means	0.7589	0.7857	0.4126	0.4535
	Direct	0.7679	0.7857	0.4684	0.5056
	Repeated Bisection	0.7589	0.7768	0.4684	0.4684
	Graph	0.7589	0.7679	0.4387	0.5576
	Agglo - Single Linkage	0.2143	0.3125	0.2305	0.2193
	Complete Linkage	0.5804	0.6071	0.4164	0.4126
	Group Average	0.4464	0.6161	0.4907	0.5502
	SOM	0.7321	0.7946	0.4498	0.4535
Stemmer Porter	K-Means	0.8143	0.8385	0.4684	0.4981
	Direct	0.8214	0.8385	0.5204	0.5762
	Repeated Bisection	0.8214	0.8214	0.5130	0.5279
	Graph	0.8197	0.8214	0.5279	0.5428
	Agglo - Single Linkage	0.2321	0.3304	0.2230	0.2305
	Complete Linkage	0.6994	0.7143	0.4535	0.4684
	Group Average	0.5982	0.6939	0.5613	0.5688
	SOM	0.8304	0.8304	0.4721	0.4796
Stemmer Portuguese	K-Means	0.9018	0.9196	0.5316	0.5428
	Direct	0.9107	0.9196	0.5390	0.5762
	Repeated Bisection	0.9107	0.9196	0.5056	0.5465
	Graph	0.9196	0.9107	0.5353	0.5725
	Agglo - Single Linkage	0.4732	0.4732	0.2565	0.2305
	Complete Linkage	0.7143	0.8482	0.4796	0.5316
	Group Average	0.7054	0.6786	0.5911	0.6059
	SOM	0.8929	0.9196	0.4684	0.4833

Tabela 4.3. Resultados da taxa de acerto para "Conhecimentos Gerais"

4.3.1.1 Sem *Stemmer*

A primeira parte da tabela mostra os resultados de vários métodos de *clustering*, sem a utilização de qualquer *stemmer*. Pode-se observar que para ambos os conjuntos de dados, a não utilização de *stemmer* gerou os piores resultados da tabela.

Corpus Conhecimentos Gerais – 6 classes

Para esse conjunto de dados, alguns métodos particionais ainda assim apresentaram um índice de acerto de 78,57% e o SOM conseguiu 79,46% ambos depois da retirada de termos da lista, correspondente à primeira modificação nos dados.

Corpus Conhecimentos Gerais – 17 classes

Para esse conjunto de dados, a precisão dos métodos de *clustering* caíram consideravelmente. As melhores marcas foram conseguidas pelos métodos *graph* partioning based com 55,76% de acerto e o método hierárquico group average com 55,02%, ambos depois da retirada de termos da lista, correspondente à primeira modificação nos dados.

4.3.1.2 *Stemmer* de Porter

A utilização do *stemmer* de Porter melhorou os resultados para todos os métodos executados, quando comparados aos resultados sem o *stemmer*. No entanto, não foi um aumento muito expressivo no índice de acerto, como se poderia supor.

Corpus Conhecimentos Gerais – 6 classes

As melhores taxas conseguidas foram para os métodos *K-means*, e para o método *Direct* ambos após a 1ª modificação nos dados. Conseguindo um aumento no índice de acerto de 6,72% em relação à melhor exatidão conseguida sem uso de *stemming*.

Corpus Conhecimentos Gerais – 17 classes

As melhores taxas de acerto conseguidas foram pelos métodos direct com 57,62% de acerto e o método hierárquico group average com 56,88%, ambos depois da primeira modificação nos dados. Esses resultados, refletem um aumento de 3,33% no índice de acerto em relação à situação sem *stemmer*.

4.3.1.3 Stemmer Portuguese

A utilização do *Stemmer* Portuguese ou RLSP foi indubitavelmente o que forneceu os melhores resultados em termos de *clustering*, superando em exatidão o *clustering* sem a utilização do *stemmer* e com o *stemmer* de Porter.

Corpus Conhecimentos Gerais – 6 classes

Os resultados alcançados com a utilização do *stemmer Portuguese* superaram em qualidade todos os resultados anteriores. Mais ainda, com o emprego desse *stemmer* alcançou-se uma taxa de acerto igual a 91,96% em vários métodos de *clustering*, o que pode ser considerada para todos os fins uma excelente taxa de acerto. Esses resultados, refletem um aumento de 9,67% no índice de acerto em relação à melhor situação utilizando *stemmer* de Porter e 17,04% em relação à melhor situação sem utilização de *stemmer*.

Corpus Conhecimentos Gerais – 17 classes

A utilização do *stemmer Portuguese* também promoveu a melhor taxa de acerto para esse conjunto de dados. Os melhores resultados foram conseguidos com a utilização do método hierárquico group average e iguais a 59,11% sem modificação e 60,59% com modificação. Embora as melhorias alcançadas não sejam tão significativas como para o conjunto de 6 classes, podemos notar um aumento no índice de acerto de 5,16% em relação aos resultados atingidos com a utilização do *stemmer* de Porter e 8,66% sem a utilização de nenhum *stemmer*.

4.3.1.4 Resultados obtidos com a Ferramenta Temis

Como uma forma de avaliar se os resultados obtidos pelo módulo de *clustering* da solução proposta seriam bons, mesmo se comparados a uma ferramenta comercial, utlilizou-se o módulo de *clustering* da Ferramenta Temis – *Insight Discoverer Clusterer*. O conjunto de dados utilizado para a comparação de resultados foi o Corpus Conhecimentos Gerais, com seis classes. Esse conjunto de dados foi o escohido por corresponder àquele com a maior taxa de acerto para os métodos disponibilizados no módulo de *clustering* do sistema.

A taxa de acerto conseguida para esse conjunto de dados, na ferramenta Temis foi de 83,92%, contra 90,18% conseguido com o método *K-means* e 89,29% para o método *SOM*, ambos para dados "default" e 91,96% conseguidos com os métodos *K-means* e *SOM*, ambos para dados "1modif".

A taxa de acerto conseguida pela ferramenta Temis se equipara aos melhores resultados utilizando o Stemmer de Porter.

É preciso ressaltar, no entanto, que a ferramenta já possui um tratamento para dados em Português incluso e o único trabalho de preparação de dados necessário é a conversão de dados para o formato XML, caso estes ainda não estejam nesse formato.

4.3.2 Corpus TeMario

A tabela 4.3 a seguir mostra os resultados obtidos para o conjunto de dados "TeMario" para 3 classes primeiramente e depois utilizando mais duas classes. As três classes inicias usadas foram: Especial, Internacional e Opinião. Como mencionado anteriormente, essa variação no conjunto de dados visa entender comportamentos associados ao aumento e/ou deterioriação nos dados.

	Métodos de Clustering Utilizados	TeMario 3 classes		TeMario 5 classes	
		default	1modif	default	1modif
Sem Stemmer	K-Means	0.5667	0.6000	0.4300	0.4600
	Direct	0.5333	0.5833	0.3900	0.4600
	Repeated Bisection	0.5667	0.6000	0.4200	0.4500
	Graph	0.4167	0.4833	0.4200	0.3700
	Agglo - Single Linkage	0.3500	0.4000	0.2500	0.2700
	Complete Linkage	0.4600	0.4833	0.3500	0.3900
	Group Average	0.4500	0.4500	0.3700	0.3800
	SOM	0.5833	0.6000	0.4300	0.4500
Stemmer Porter	K-Means	0.6333	0.6667	0.4600	0.5000
	Direct	0.6167	0.6500	0.4800	0.5000
	Repeated Bisection	0.6333	0.6500	0.4600	0.4800
	Graph	0.4833	0.5500	0.4600	0.4600
	Agglo - Single Linkage	0.3667	0.3667	0.2800	0.3600
	Complete Linkage	0.5333	0.4500	0.4000	0.4400
	Group Average	0.4000	0.4500	0.4000	0.4400
	SOM	0.6167	0.6667	0.4600	0.5000
Stemmer Portuguese	K-Means	0.8000	0.8167	0.5000	0.5500
	Direct	0.7500	0.7667	0.4900	0.4800
	Repeated Bisection	0.6833	0.6500	0.5000	0.5500
	Graph	0.6667	0.6667	0.5000	0.5400
	Agglo - Single Linkage	0.4333	0.4600	0.3000	0.4800
	Complete Linkage	0.5167	0.6167	0.4400	0.4800
	Group Average	0.4500	0.4500	0.4500	0.5000
	SOM	0.7500	0.7667	0.5100	0.5500

Tabela 4.4. Resultados da variação da taxa de acerto para "TeMario"

4.3.2.1 Sem *Stemmer*

A primeira parte da tabela mostra os resultados obtidos sem a utilização de qualquer *stemmer*. A comparação dos vários métodos é mostrada. Pode-se observar que também para esse caso, a não utilização de *stemmer* gerou os piores resultados da tabela para ambos os casos.

Corpus TeMario – 3 classes

Para esse conjunto de dados, houve um empate entre dois métodos particionais K-means Repeated Bisection e o SOM apresentando um índice de acerto de 60,00%, todos após a retirada de termos da lista, correspondente à primeira modificação nos dados.

Corpus TeMario – 5 classes

Para esse conjunto de dados, a precisão dos métodos de *clustering* é um pouco inferior. Os maiores valores alcançados em precisão foram o K-means e o Direct ambos particionais e iguais a 46% ambos depois da retirada de termos da lista, correspondente à primeira modificação nos dados.

4.3.2.2 *Stemmer* de Porter

A utilização do *stemmer* de Porter conseguiu resultados um pouco melhores para todos os métodos, em relação aos resultados obtidos sem o uso do *stemmer*.

Corpus TeMario – 3 classes

As melhores taxas conseguidas foram para os métodos K-means, e para o método SOM, e iguais a 66,67%, ambas obtidas após a 1ª modificação nos dados. Esses resultados revelam um aumento no índice de acerto de 11,11% em relação à melhor exatidão conseguida sem uso de *stemming*.

Corpus TeMario – 5 classes

As melhores taxas de acerto foram conseguidas pelos métodos K-means, Direct com 50,00% de acerto e o método SOM também com 50,00%, todos depois da primeira modificação nos dados. Esses resultados, refletem um aumento de 8,69% no índice de acerto em relação à situação sem *stemmer*.

4.3.2.3 *Stemmer Portuguese*

A utilização do RSLP forneceu novamente os melhores resultados em termos de *clustering*. Assim como no conjunto de dados "Conhecimentos Gerais", o uso desse *stemmer* conseguiu taxas de acerto melhores em todas as situações mostradas na tabela quando comparadas às situações – sem *stemmer* e *stemmer* de Porter.

Corpus TeMario – 3 classes

Os resultados obtidos para essa situação demonstraram que a utilização do processo de *stemming* pode levar a ganhos consideráveis como nesse caso, em que a taxa de acerto para a melhor situação chegou a 81,67% contra 60,00% da taxa de acerto sem uso do *stemming* e 66,67% para o *stemmer* de Porter. Isso significa um expressivo aumento no índice de acerto de 36,11% em relação à primeira situação e 22,49% em relação à segunda.

Corpus TeMario – 5 classes

Embora os aumentos de precisão conseguidos para esse conjunto de dados não tenham sido tão grandes, ainda assim observa-se uma melhoria considerável. Os melhores resultados foram conseguidos com a utilização do K-means e do SOM e iguais a 55,00%, ambos após a modifcação nos dados. Embora as melhorias alcançadas não sejam tão significativas como para o conjunto "TeMario" com 3 classes, pode-se notar um aumento no índice de acerto de 10,00% em relação aos resultados atingidos com a utilização do *stemmer* de Porter e 19,56% sem a utilização de nenhum *stemmer*.
A taxa de acerto encontrada dada a comparação entre a classificação prévia e os *clusters* criados pela ferramenta foi de 50%. Note que essa baixa taxa de acerto pode ser devido ao fato de que cada seção do jornal deveria corresponder a um assunto ou tópico, o que não é a realidade.

Observa-se para os conjuntos de dados analisados, que tanto para situações de taxa de acerto alta como baixa, ocorre um acerto progressivo. As taxas de acerto para dados sem

stemming são as mais baixas, passando para valores intermediários com relação aos dados com *stemming* de Porter, e por fim chegando aos valores mais altos da tabela, que dizem respeito aos dados que sofreram o processo de *stemming* RSLP. Esses resultados condizem com a idéia de que a aglutinação das contribuições de termos similares poderia ajudar a descrever melhor coleções de textos. A interpretação mais plausível desses resultados seria que dados sem *stemming* enfraquecem a força de atributos que podem auxiliar a tarefa de *clustering*, comprometendo a exatidão conseguida. O *stemmer* de Porter é uma implementação que consegue ganhos em relação a situação sem *stemmer*, porém, não consegue se igualar à implementação RSLP. Observando um conjunto de dados que sofreu os dois tipos de *stemmers* (Porter e RSLP) comprova-se esse fato. Encontra-se variações da mesma palavra que ao longo do processo de *stemming* acabaram resultando em radicais diferentes. Isso acontece com mais freqüência para a implementação de Porter que para a RSLP.

Isso reforça a idéia de que o pré-processamento dos dados pode resultar na melhoria de qualidade da etapa de realização das tarefas de text mining. Outro passos de pré-processamento, como o dicionário, que não foram enfatizados nessa tese, poderiam, se trabalhados adequadamente, fornecer ganhos aos processos subseqüentes.

4.4 Variação do Número de Termos - Dados *Default*

Alguns conjuntos de dados utilizados nessa tese não apresentaram uma taxa de acerto muito alta. A variedade de métodos usados ajudou a encontrar agrupamentos um pouco mais precisos. No entanto, esperava-se de conseguir ainda melhores resultados. Dado o fato que o *clustering* é realizado baseado nos termos selecionados dentro da ordenação de termos segundo o peso escolhido, a hipótese de que se mais termos fossem considerados poderia melhorar o desempenho do *clustering* foi seguida. Escolheu-se os dados *"default"*, ou seja, como fornecidos pela saída do pré-processamento e variou-se o número de termos de 50 até 1000. Em alguns casos foi necessário variar até mais para entender a resposta da taxa de acerto em função da variação do número de termos. Como o melhor índice de acerto para todos os conjuntos de dados utilizados nesse estudo foi usando o *stemmer Portuguese*, essa parte do estudo utilizou esse *stemmer* apenas.

4.4.1 Corpus Conhecimentos Gerais – 6 classes

O *clustering* de dados proporcionou um bom acerto para esse conjunto de dados, porém, a variação da quantidade de termos levada em consideração poderia indicar se o número de 100 termos utilizado seria o melhor. Para esse conjunto de termos a variação foi feita de 50 até 2000 termos. O gráfico mostra a taxa de acerto em 50 igual a 89,28%, subindo em 100 e se mantendo em 200 igual a 90,17%, atingindo o máximo em 500 igual a 91,07% e então passando a cair.

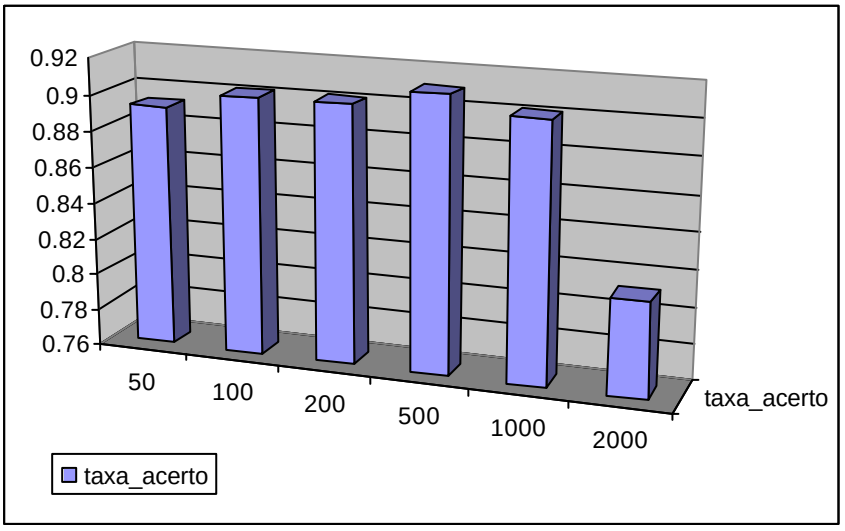


Gráfico 4.1. Taxa de acerto para Corpus Conhecimentos Gerais com 6 classes

A forma aproximada dada pelos topos das barras se repete ao longo do estudo para os diferentes conjuntos de dados.

no. termos	taxa_acerto
50	0.8928
100	0.9018
200	0.9018
500	0.9107
1000	0.9018
2000	0.8125
3000	0.6875

Tabela 4.4. Taxa de acerto em função do número de termos 6 classes

Embora a exatidão do processo de *clustering* seja a meta principal, não se pode esquecer que uma quantidade muito grande de termos levada em consideração torna o processo de *clustering* mais lento. Assim, o que se procura é a quantidade mínima de termos que forneça uma boa exatidão. O caso mostrado no gráfico 4.1 ilustra bem essa situação. Nesse caso, a melhoria da taxa de acerto entre o pior e melhor caso é de 1.7857%. Porém, sobe-se de 50 termos (na pior situação) para 500 termos (na melhor situação), ou seja 10 vezes a quantidade de termos.

4.4.2 Corpus Conhecimentos Gerais – 17 classes

Composto de um número maior de classes e de documentos que o conjunto "conhecimentos gerais" anterior e, portanto, contendo um número maior de termos (11534 contra 6857), tem-se uma indicação que o corpus precisará de uma quantidade maior de termos para agrupar bem os dados. O gráfico 4.2 mostra os resultados obtidos pela variação do número de termos para esse conjunto de dados, que nesse caso foi feita de 50 até 5000 termos.

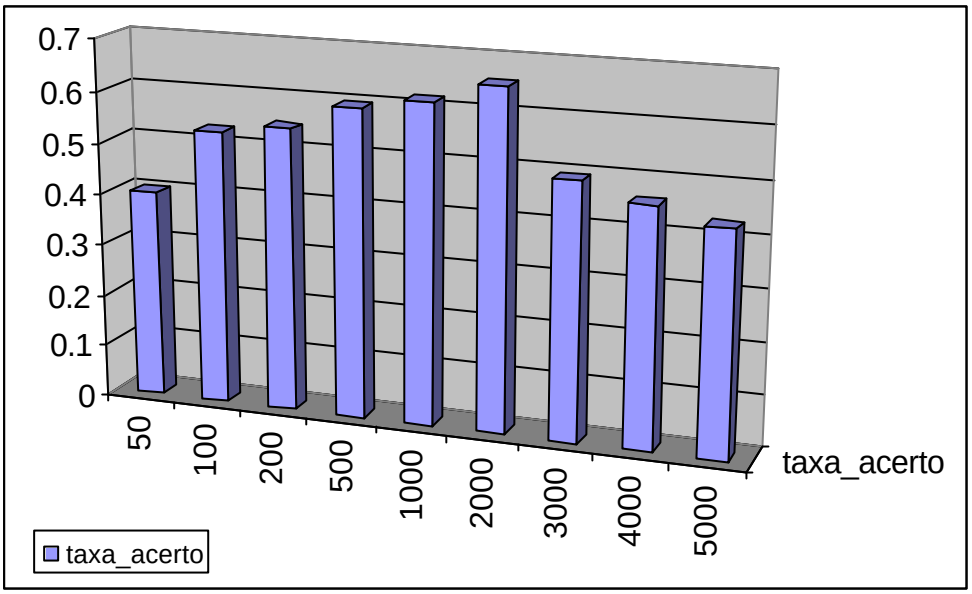


Gráfico 4.2. Taxa de acerto para Corpus Conhecimentos Gerais com 17 classes

Para esse caso, a taxa de acerto apresenta uma variação significativa em função da quantidade de termos. Na pior situação, também para o número de termos igual a 50, a taxa de acerto é igual a 40,14% e na melhor situação, para o número de termos igual a 2000, tem um acerto de 65,42%, ou seja, superior a 15% de melhoria. No entanto, a quantidade de termos para a melhor situação é 40 vezes maior que na pior situação, demandando muito mais tempo de processamento.

no. termos	taxa_acerto
50	0.4015
100	0.5316
200	0.5465
500	0.5948
1000	0.6171
2000	0.6543
3000	0.4944
4000	0.4609
5000	0.4312

Tabela 4.5. Taxa de acerto em função do número de termos 17 classes

4.4.3 Corpus TeMario – 3 classes

Essa coleção contém 60 arquivos em 3 classes diferentes. De forma contrária ao conjunto de dados anterior, cuja variedade de termos implicou na solicitação de um número maior de termos para a obtenção de mellhores resultados, essa coleção de

apenas 60 arquivos atingiu sua taxa de acerto máxima para o número de termos igual a 100. A partir daí, o aumento do número de termos não acarreta aumento no acerto.

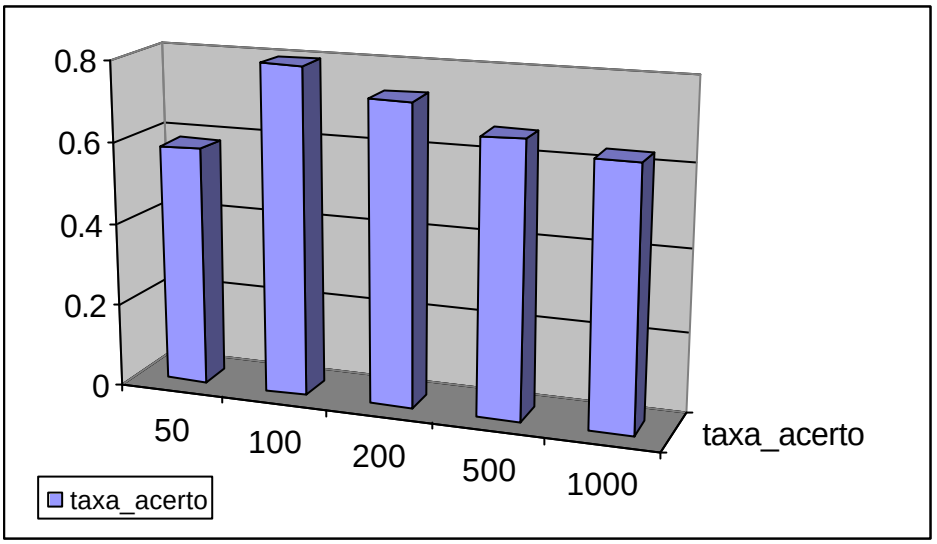


Gráfico 4.3. Taxa de acerto para Corpus TeMario com 3 classes

A variação da taxa de acerto neste caso é bastante acentuada, mesmo para uma variação não muito significativa no número de termos, partindo da pior situação com 50 termos e uma taxa de acerto de 58,33%, e atingindo a melhor situação para 100 termos com 80,00% . Essa variação significa um aumento de 37,15% na taxa de acerto, apenas pela inclusão de mais 50 termos na análise .

no. termos	taxa_acerto
50	0.5833
100	0.8000
200	0.7333
500	0.6667
1000	0.6333

Tabela 4.6. Taxa de acerto em função do número de termos 3 classes

4.4.4 Corpus TeMario – 5 classes

Esse conjunto formado por 5 classes, duas a mais que o anterior. A variação de termos nesse caso será bem interessante pelo fato das classes não descreverem exatamente um único assunto. Classes como “mundo” e “internacional” apresentam termos em comum, o que dificulta o processo de *clustering*. Nota-se que o aumento no número de termos

considerados contribui de forma significativa para a melhoria na taxa de acerto, porém, a melhor situação apresenta um índice de acerto de 50%, o que não pode ser considerado um acerto muito alto.

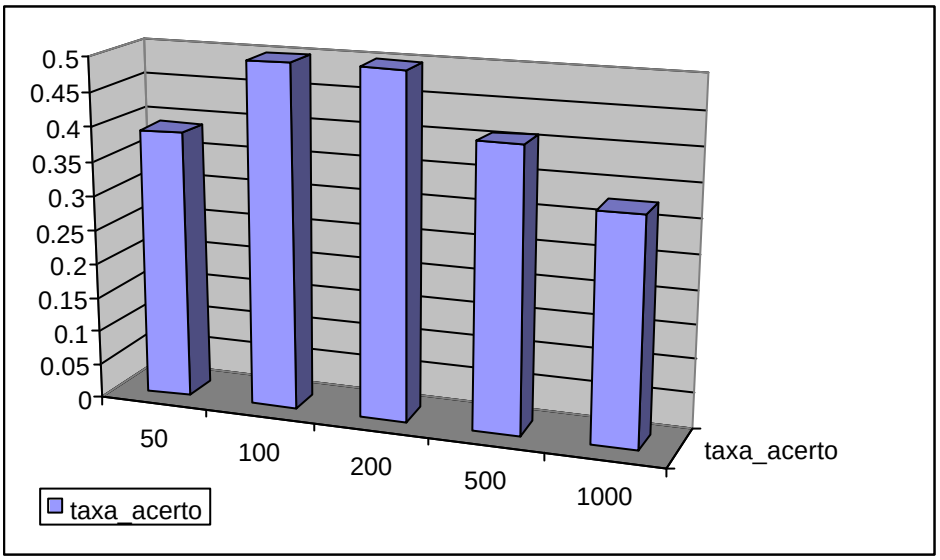


Gráfico 4.4. Taxa de acerto para Corpus TeMario com 5 classes

Nesse caso específico, os documentos que representam as classes não possuem atributos (termos) em comum suficientes para serem identificados pelo processo de *clustering* como dos mesmos *clusters*. A melhoria só seria conseguida por uma classificação manual e posterior utilização da ferramenta de *clustering*. A seguir, apresenta-se o gráfico da taxa de acerto em função do número de termos para esse conjunto de dados.

no. termos	Taxa_Acerto
50	0.39
100	0.50
200	0.50
500	0.41
1000	0.33

Tabela 4.7. Taxa de acerto em função do número de termos 5 classes

4.4.5 Corpus CETENFolha

Esse conjunto de documentos é bem maior que os anteriores possuindo 12396 arquivos de tópicos diferentes do Jornal Folha de São Paulo. Um dos objetivos de escolher um conjunto de dados maior é testar a solução para uma quantidade maior de dados. A

variação do número de termos para esse conjunto de dados foi efetuada de 100 até 2000 termos. No caso dos 2000 termos considerados, o sistema gerou uma matriz de 2000 colunas por 12396 linhas de dados. Mesmo para essa quantidade de dados o sistema se mostrou estável.

A melhora na taxa de acerto alcançada para esse conjunto de dados não foi tão expressiva quanto as dos demais conjuntos apresentados anteriormente. Uma das classes, a Folhateen, teve um índice de acerto pequeno em relação a classificação original, demonstrando uma dificuldade em se estabelecer um vocabulário que defina esta classe.

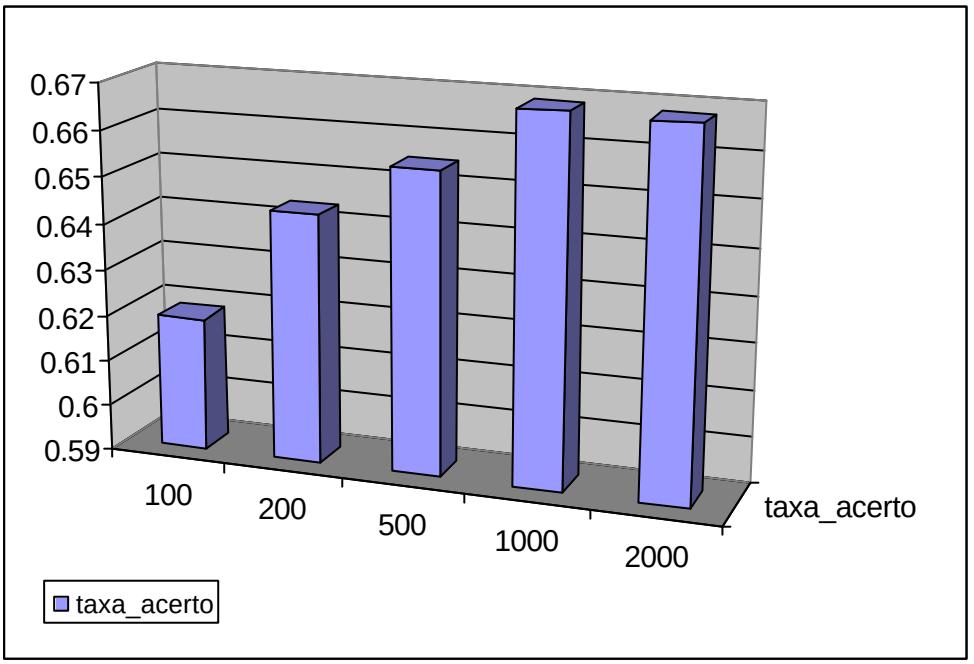


Gráfico 4.5. Taxa de acerto para Corpus CETENFolha com 3 classes

Pode-se observar na tabela 4.8, no entanto, que há uma variação para melhor de 8,08% entre a pior taxa de acerto para um número de termos igual a 100 e a melhor taxa de acerto conseguida para um número de termos igual a 1000. Esse aumento na taxa de acerto, no entanto, é provocado por aumento do número de termos de dez vezes, implicando em um aumento considerável de processamento para o sistema.

no. termos	taxa_acerto
100	0.6187
200	0.6437
500	0.6549
1000	0.6688
2000	0.6681

Tabela 4.8. Taxa de acerto em função do número de termos 3 classes

A contribuição dos termos é limitada por um critério de qualidade. Diferentemente do que se possa imaginar, adicionando cada vez mais termos à análise não a torna cada vez mais rica, fornecendo resultados cada vez mais precisos. Esta situação está relacionada ao problema de selecionar atributos relevantes ao processo de clustering:

- atributos correlacionados/dependentes
- explosão combinatória

Os resultados obtidos com os conjuntos de dados utilizados mostraram que a contribuição dos termos vai aumentando, atinge um máximo de acerto e depois tende a cair. O número de termos que fornece esse máximo varia de coleção para coleção.

4.5 Visualização dos Resultados

A visualização é uma parte importante do processo de *clustering*. Considerando que o *clustering* de dados é um tipo de aprendizado não supervisionado por definição, ou seja, não existe uma classificação inicial dos dados, a visualização pode ajudar a entender resultados. No caso dos textos, em que cada termo considerado é convertido para um atributo ou dimensão, a visualização da saída pode ser confusa e difícil de entender. Nesta tese, apresenta-se uma forma de visualização que permite ao usuário entender a organização de documentos e termos, permitindo ao usuário identificar, por exemplo, grupos de termos relacionados a grupos específicos de documentos, termos que aparecem indiferentemente em todos os *clusters*, etc.

Todos os métodos de *clustering* do sistema fornecem um tipo de saída que pode ser transformada em uma visualização. O tipo de visualização disponibilizada varia com o tipo de método executado. Os métodos hierárquicos geram dendrogramas que podem ser só de documentos, só de termos e de documentos e termos. Junto com o dendrograma, é gerada para todos os métodos uma matriz de cores indicando que termos ocorrem em que documentos. Todas as imagens produzidas nessa seção consideraram a hierarquia de documentos e termos. A ferramenta de visualização fornece para cada nó do dendrograma selecionado a correlação do conjunto de documentos abaixo dele, além de identificar cada um dos documentos. O método de visualização adotado não possui a escala de medida das alturas no dendrograma, dificultando o corte para contagens de *clusters*. No entanto, ele fornece toda a estrutura organizacional de documentos para qualquer coleção permitindo uma visualização completa do conjunto de documentos. Além disso, ele fornece também a estrutura de organização de termos associados a esses documentos, o que permite identificar a ocorrência de palavras que ainda podem ser retiradas da análise. É o caso de palavras muito genéricas, por exemplo, a palavra “computador” no caso de se tratar do assunto informática, palavras que não ajudam na definição de contextos, por exemplo, a palavra “dia” ou “mês”. É possível também marcar *clusters* de palavras e cruzar o conjunto selecionado com o de documentos.

Os métodos particionais geram a visualização da matriz de cores dividida no número de *clusters* definido em K. Dessa forma, identifica-se perfeitamente que documentos pertencem a que *clusters*.

O objetivo dessa parte da tese é mostrar como a visualização pode ser útil na interpretação dos resultados.

Para a disponibilização das visualizações, usar-se-á a convenção de vermelho para as visualizações provenientes dos dados recebidos pelo módulo de *clustering* sem qualquer mudança e azul para as visualizações produzidas após a primeira modificação nos dados.

4.5.1 Corpus Conhecimentos Gerais – 6 classes

Esse corpus foi montado a partir de um *site* de pesquisa na internet. Foram elaboradas primeiramente 6 classes sobre temas diferentes para servir como ponto de partida. Obviamente as classes foram omitidas e o conjunto como um todo foi fornecido à ferramenta para a execução dos diferentes métodos de *clustering*. As classes escolhidas para compor esse corpus foram: Artes Plásticas, Física, Geografia, Música, Religião e Teatro. Os arquivos correspondentes a essas classes totalizaram 112 arquivos.

K-means

A visualização apresentada na figura orresponde ao método K-means para dados pré-processados utlizando o *stemming* RSLP. O K-means, como os demais métodos particionais, necessitam da definição a priori do número de *clusters* e foram executados com o K igual a 6, de forma a tentar reproduzir *clusters* assemelhados às classes originais.

As visualizações obtidas para esse conjunto de dados são mostradas nas figuras 4.1 (a) e (b), para dados inalterados e com alterações respectivamente.

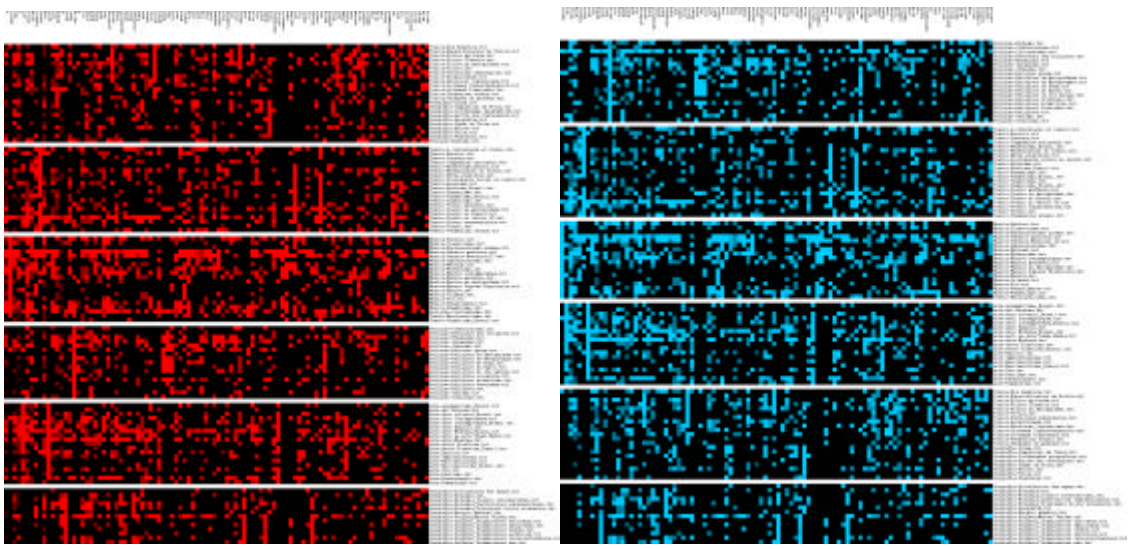


Figura 4.1 (a)K-means sem mudanças nos dados (b) com mudanças

A visualização conforme aparece mostrada nas figuras 4.1 (a) e (b) correspondem às saídas do módulo de visualização quando exportadas para o formato PNG. Nessas

figuras, observa-se os documentos na lateral direita da visualização e os termos na parte superior. Dessa forma, é possível identificar toda a distribuição de termos nos documentos da coleção. No ambiente de visualização, é possível marcar partes da matriz de cores de forma a estudar apenas um subconjunto de documentos ou de termos.

Graph – Cluto

A visualização fornecida pelo Cluto, é gerada diretamente do processo de *clustering* e não permite qualquer interação, nem configuração. A saída do Cluto mostrada a seguir na figura 4.2 foi gerada para o método Graph.

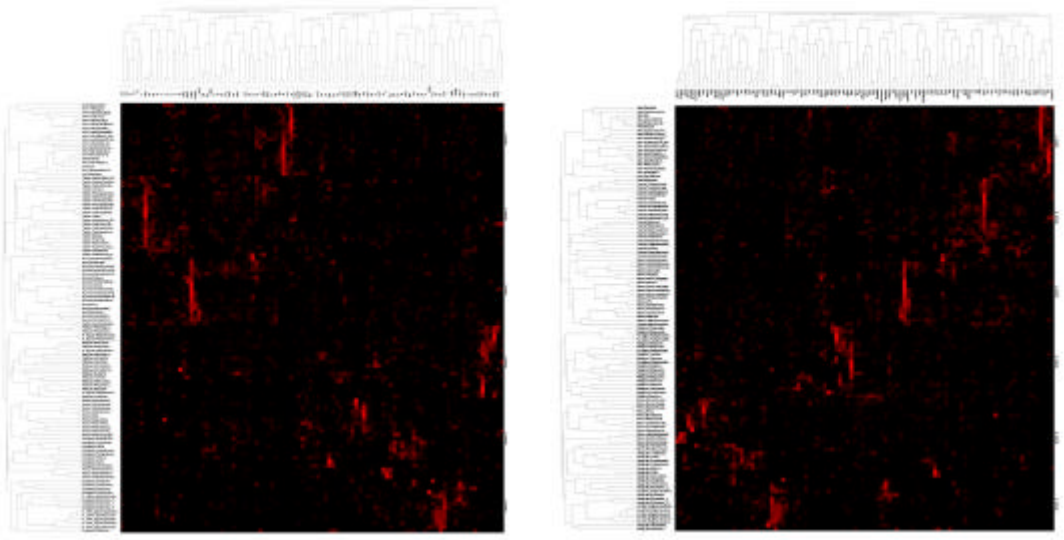


Figura 4.2 (a)Graph sem mudanças nos dados (b) com mudanças

Clustering Hierárquico - *C-Clustering-Library*

Os resultados produzidos para a seleção *cluster* hierárquico usando single-linkage são mostradas a seguir nas figuras 4.3. Embora o visualizador não gere uma saída do tipo "amostra n – *cluster* x" que relaciona as amostras a *clusters*, ela mostra toda a hierarquia dos documentos e dos termos.

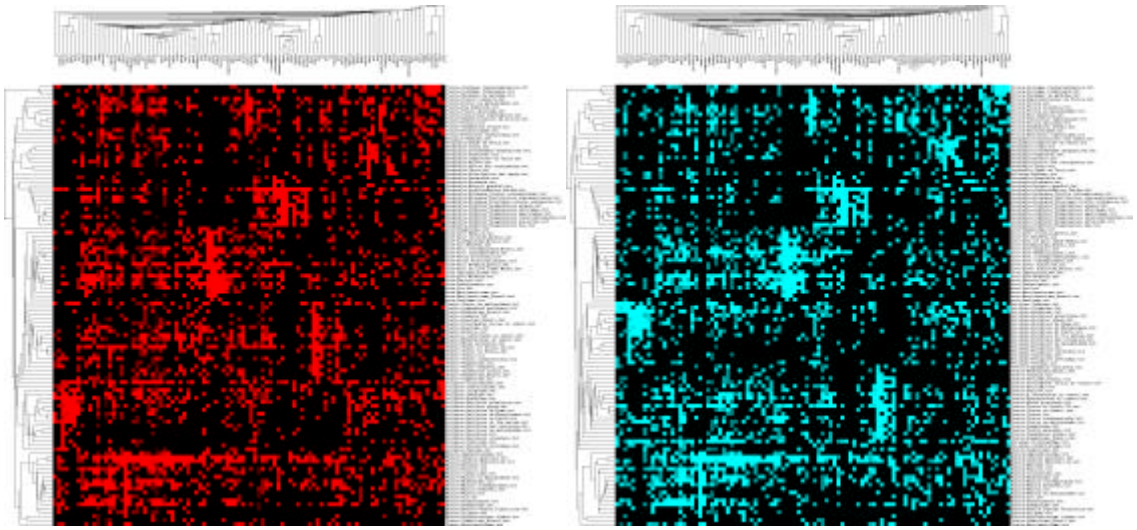


Figura 4.3 (a)Hierárquico sem mudanças nos dados (b) com mudanças

A figura 4.4 mostra como é possível identificar grupos de termos que se encontram muito ligados a determinados grupos de documentos. No exemplo assinalado na figura 4.4(a), mostra-se um *cluster* ligado à "Geografia", muito ligado a um grupo de termos bem específico e detalhado na figura.

Existe, no entanto um outro grupo ligado à Geografia mostrado na figura 4.4 (b) que não ficou dentro do mesmo *cluster* anterior e que praticamente não tem ligação com os termos "fortes" do *cluster* Geografia da figura 4.4(a).

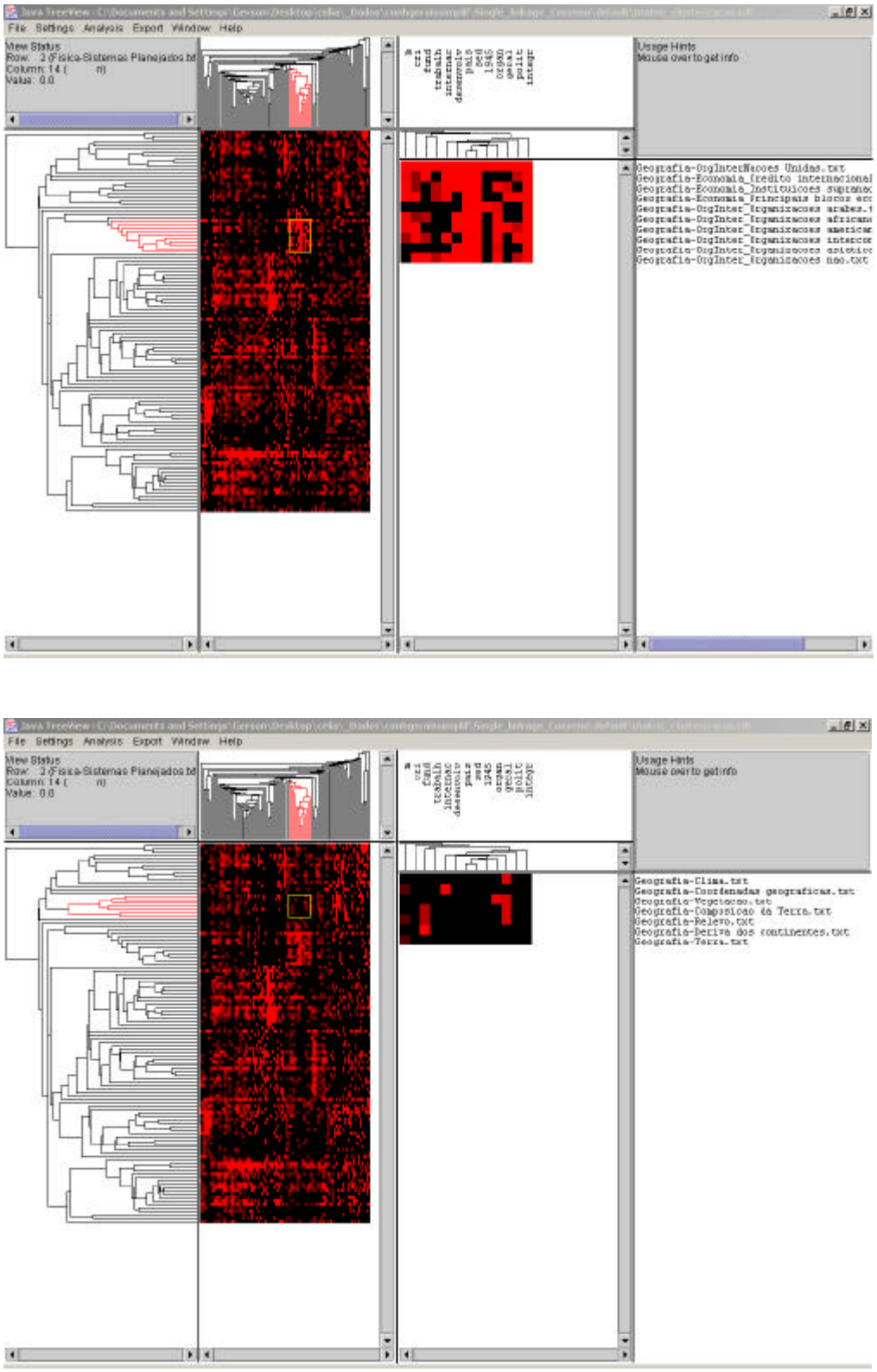


Figura 4.4 (a)termos ocorrem em documentos “Geografia” (b) termos não ocorrem

4.5.2 Corpus Conhecimentos Gerais – 17 classes

Esse corpus foi montado a partir do mesmo *site* de pesquisa, mas agora utiliza-se 17 classes sobre temas diferentes. Algumas visualizações produzidas para esse conjunto de dados serão apresentadas e comentadas.

K-means – C-Clustering Library

As visualizações produzidas para dados sem alterações e com alterações são mostradas na figura 4.5. Nota-se que acontece uma re-arrumação dos vetores documento. No entanto, examinando com mais calma, repara-se que os *clusters* em si não mudaram muito de composição, apenas de lugar.

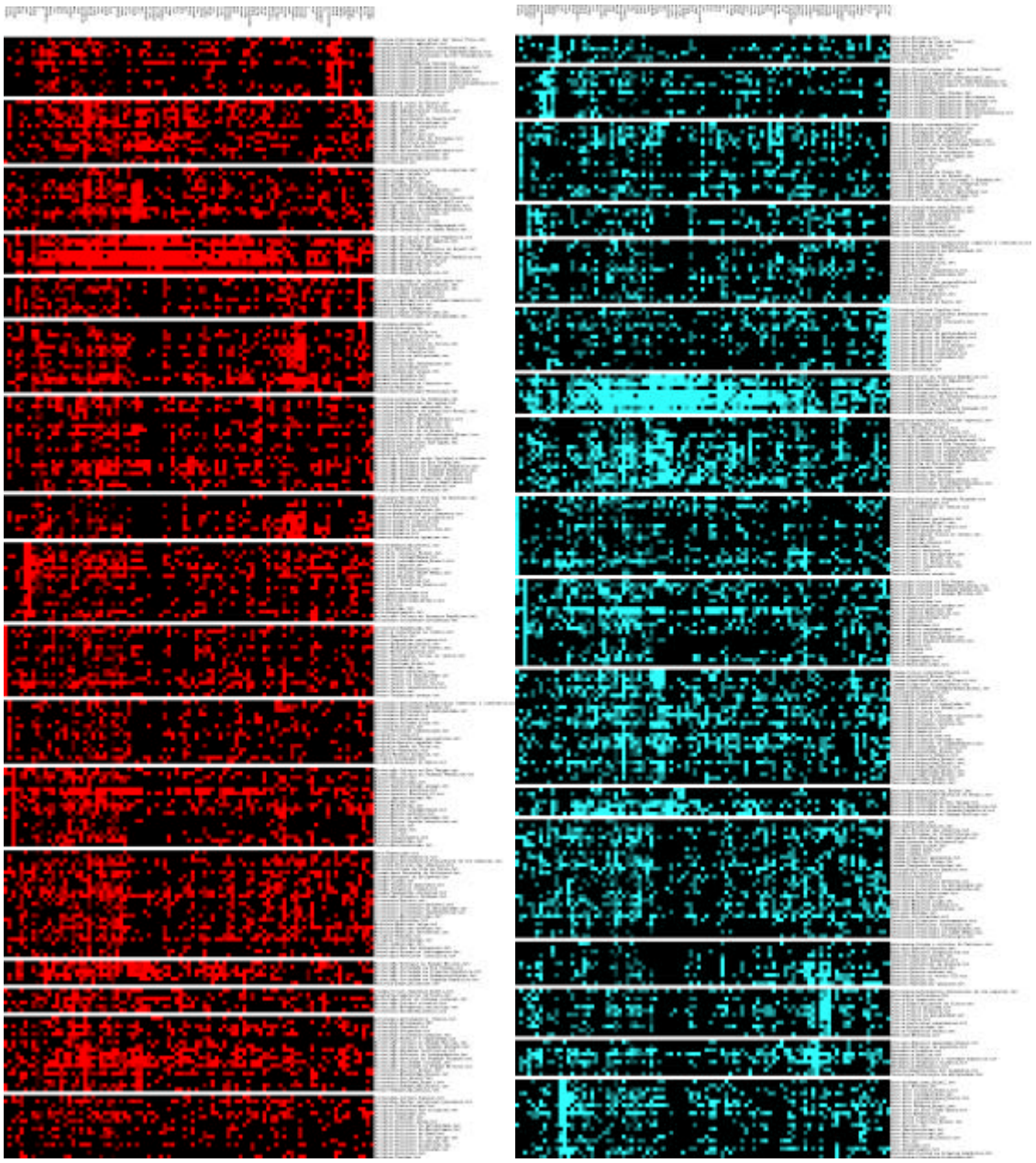


Figura 4.5 (a)K-means sem mudanças nos dados (b) com mudanças

Clustering Hierárquico – *C-Clustering Library*

A primeira vez que o programa foi executado para esses dados, o resultado foi produzido para a lista dos 100 termos mais freqüentes, exatamente como ela foi gerada pelo script, sem retirar mais nenhum termo. A figura 4.6(a) reflete o resultado produzido.

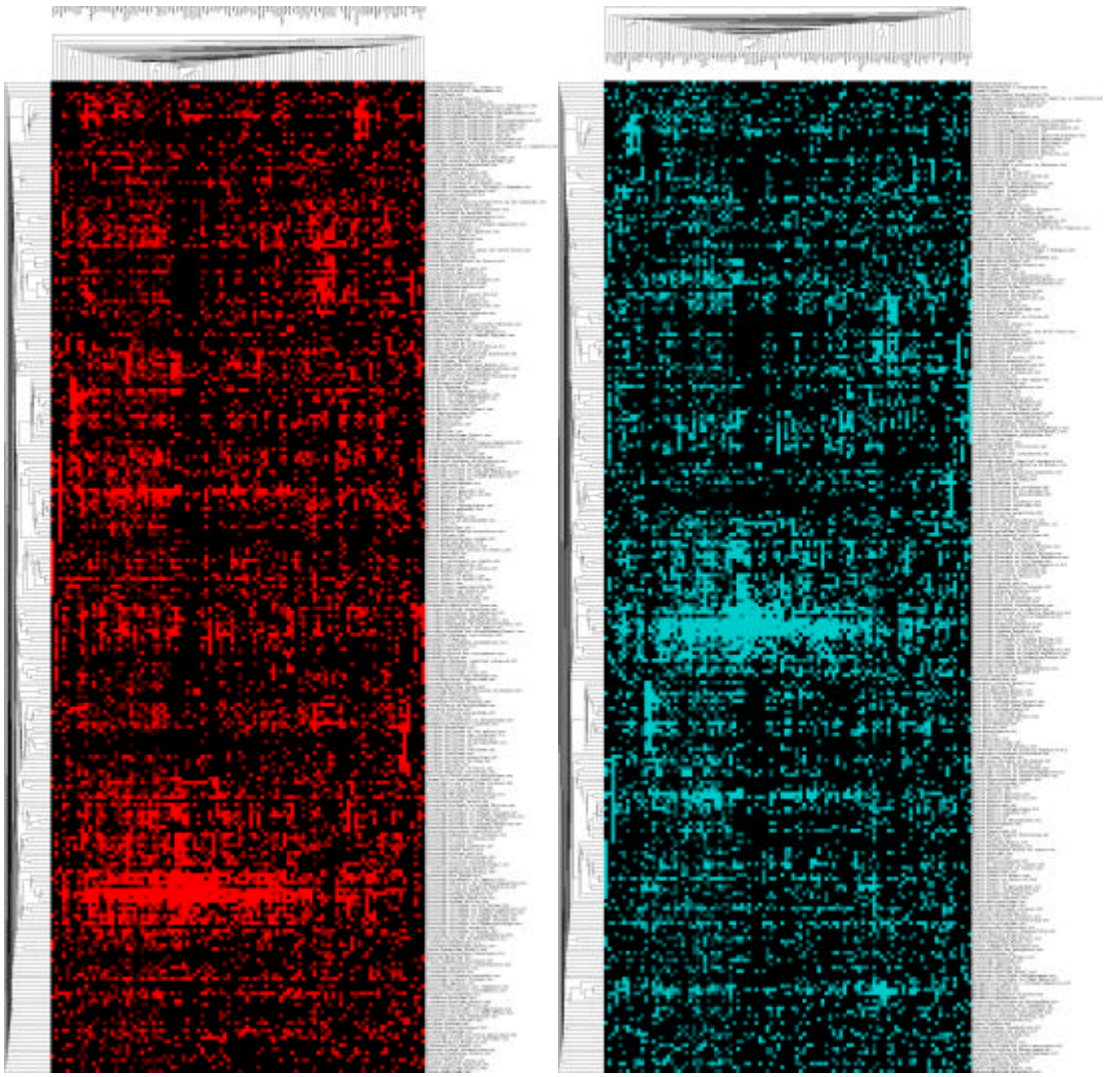


Figura 4.6 (a)Hierárquico sem mudanças nos dados (b) com mudanças

Para esse conjunto de dados notamos que os *clusters* aparecem com uma certa separação uns dos outros, denotando um possível bom *clustering* de dados. No entanto, ao observar os termos sendo considerados para a análise, nota-se a presença de termos como verbos ser e estar, palavras como não, etc. que poderiam ser retiradas das análise por se tratarem de palavras genéricas, sem poder discriminativo. Operando essa retirada, gera-se a nova visualização da distribuição dos dados, pelos novos *clusters* gerados que é mostrada na figura 4.6(b) .

4.5.3 Corpus TeMario

As visualizações produzidas para esse conjunto de dados também foram realizadas para dados com *stemming* RSLP. Foram considerados os 100 termos mais freqüentes da coleção.

K-means

As visualizações produzidas pelas execuções do método *K-means* são apresentadas na figura 4.7. A primeira visualização foi produzida com os dados tal qual foram fornecidos pelo módulo de pré-processamento (a) e depois alguns termos foram retirados (b). Nas figuras, já aparece assinalado um conjunto de apenas quatro documentos que formam um *cluster*. Essa situação será comentada a seguir.

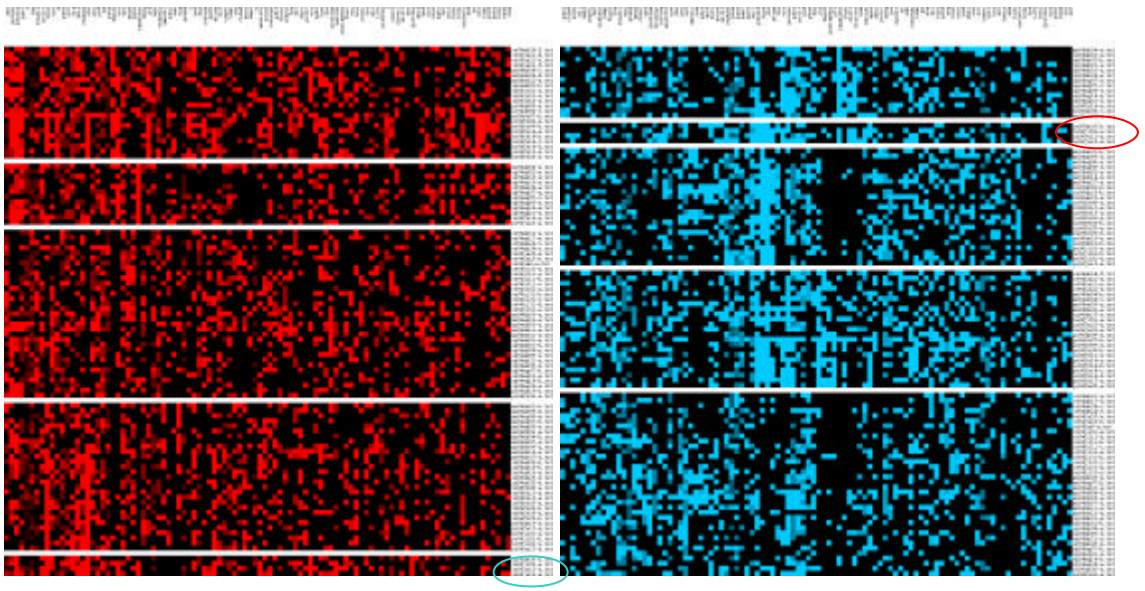


Figura 4.7 (a) Método K-means simples (b) Método com retirada de termos

Clustering Hierárquico

A visualização mostrada a seguir se relaciona com o método de *clustering* hierárquico *single-linkage*. Para esse conjunto de dados algumas considerações são pertinentes. Os grupos de notícias, embora identifiquem perfeitamente seções do Jornal Folha de São Paulo e Jornal do Brasil, não restringem assuntos, assim "política" pode tratar de

política no mundo e se aproximar de uma notícia da sessão “mundo” ou “internacional” em termos de conteúdo. Assim, durante o processo de *clustering* percebemos a rearrumação de vários documentos, alguns se aproximando de documentos tidos como de outras seções. Alguns documentos, no entanto se apresentam sozinhos e bem separados como é o caso do pequeno *cluster* assinalado nas figuras 4.7 e 4.9. Nesse caso, quatro documentos aparecem sempre juntos, apresentam distâncias próximas e são bem destacados dentro da hierarquia de *clusters*. São os documentos “in96ab26-b.txt”, “in96jul3-a.txt”, “in96jl02-a.txt”, “in96jul9-a.txt”. Recorrendo aos textos nota-se o assunto “Rússia” como tema principal associado a eleições, política, etc. como mostra a figura 4.2. Dessa forma, pôde-se verificar que um *cluster* com medidas próximas e bem separado trata de um mesmo assunto.

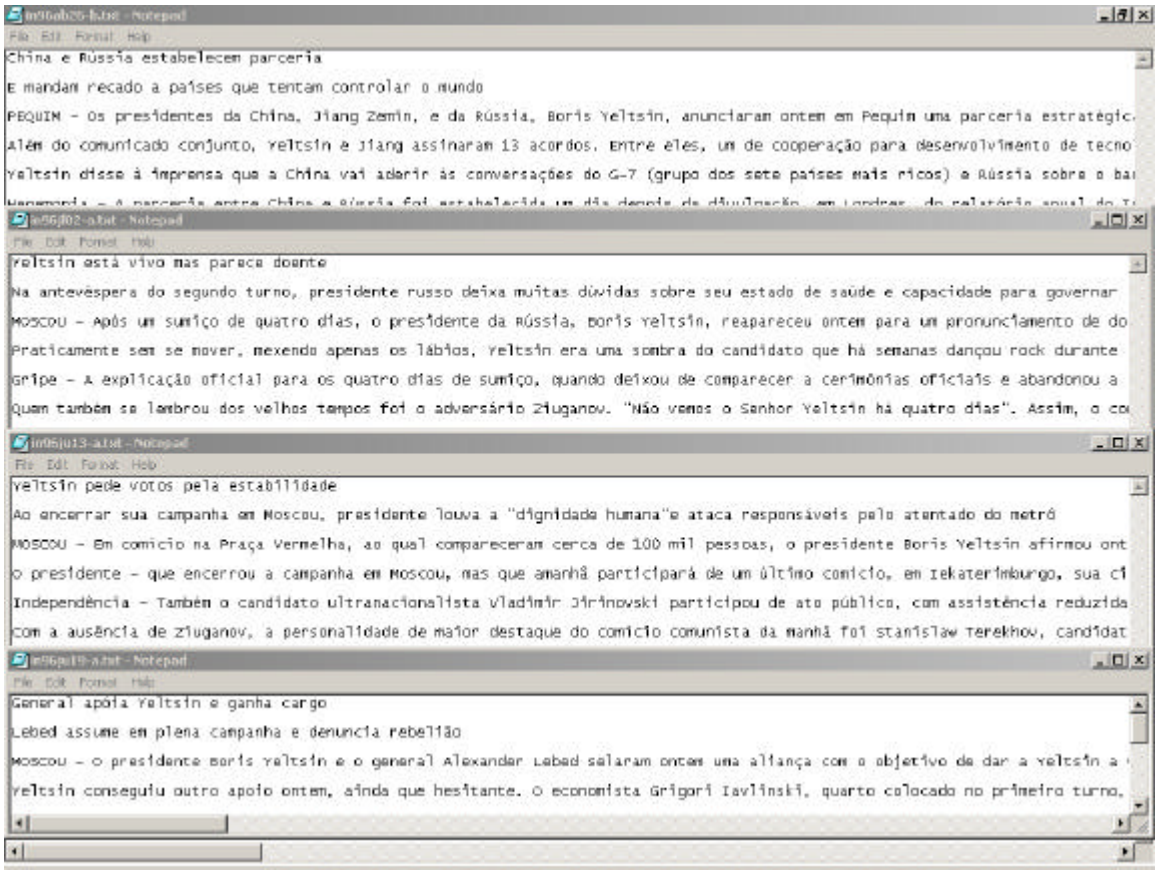


Figura 4.8 Documentos próximos e destacados dos demais na hierarquia de *clusters*

A visualização dos resultados produzidos para esse conjunto de dados, tanto a produzida pelo K-means como pelo método hierárquico, não fornecem imediatamente ao olhar a

identificação de *clusters* expressivos bem separados. Somente com a utilização de métricas é possível identificá-los.

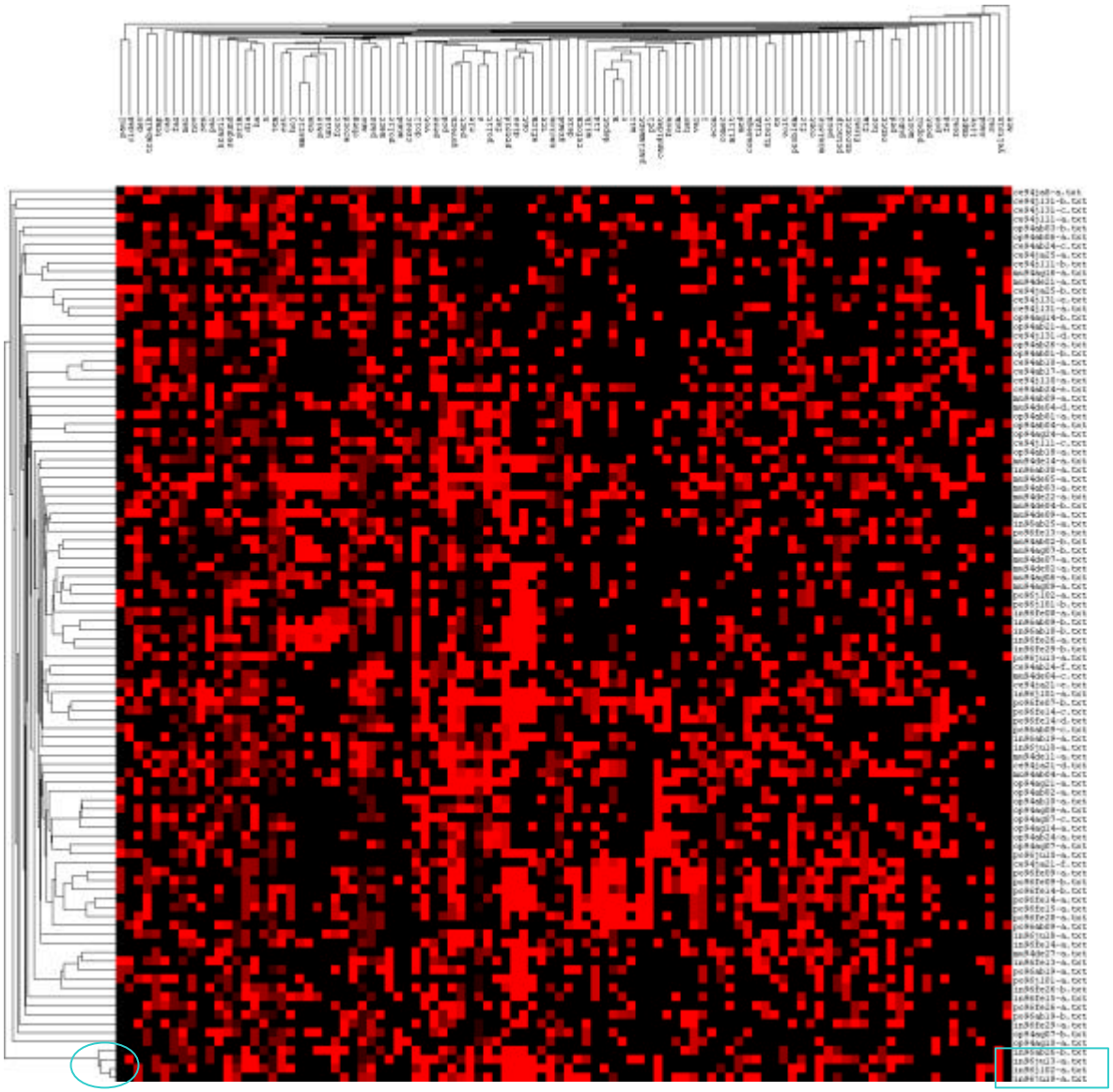


Figura 4.9 Visualização Hierárquica com dados inalterados

Após a retirada de termos a visualização produzida pela ferramenta é apresentada na figura 4.10. Não houve grandes alterações na arrumação dos vetores, o que indica que os termos retirados não exerciam grande influência na análise.

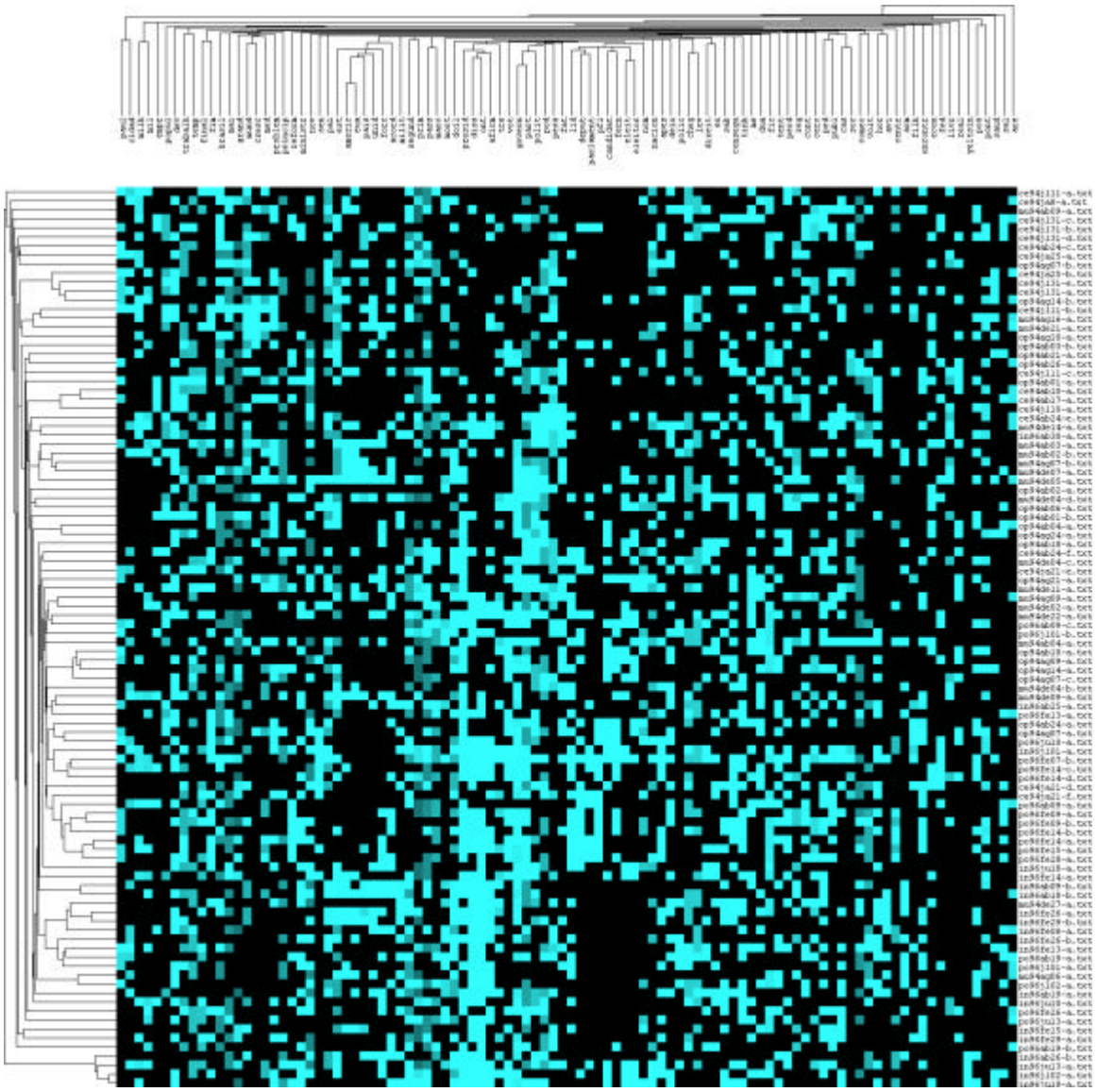


Figura 4.10 Visualização Hierárquica após a retirada de termos

4.6 Resultados da Categorização Assistida

Como uma forma complementar ao *clustering* de documentos, também conhecido como categorização baseada em *clustering*, implementou-se um módulo de categorização assistida. Essa categorização é do tipo explicado anteriormente em que se separa um conjunto de documentos representativo de cada assunto, treina-se a ferramenta e posteriormente apresentam-se novos documentos à ferramenta para que sejam categorizados.

Corpus Conhecimentos Gerais – 6 classes

Pela exatidão alcançada pela categorização baseada em *clustering*, pode-se inferir que os conjuntos de dados referentes às 6 classes sejam suficientes para treinar a ferramenta de categorização, categorizando corretamente as novas amostras. Separou-se de cada conjunto amostras aleatórias e treinou-se a ferramenta com as demais amostras. Em seguida apresentou-se as amostras (retiradas do conjunto original) a categorizar à ferramenta. Para esse conjunto de dados, os resultados foram:

- para *CosineSimilarity*: 100%
- para *JaccardSimilarity*: 100%
- para *WeightedCosineSimilarity:* 100%

Corpus TeMario – 5 classes

Esse corpus contém 100 arquivos no total, divididos em 5 classes iniciais, 20 documentos em cada uma. O categorizador gera um resumo contendo o número de frases e termos para cada categoria processada.

Numa segunda etapa, as amostras não-vistas são apresentadas ao categorizador já treinado que associa a cada uma delas a categoria que considera mais adequada. Para esse conjunto de dados os resultados foram:

- para *CosineSimilarity*: 80%
- para *JaccardSimilarity*: 80%
- para *WeightedCosineSimilarity:* 80%

A categorização errou basicamente atribuindo documentos da categoria “mundo” à categoria “internacional”.

Capítulo 5

Considerações Finais

5 Considerações Finais

No presente trabalho foi apresentada a fundamentação teórica para o problema de *clustering* de documentos para a Língua Portuguesa e proposto um conjunto de ferramentas para a sua solução. Foi descrito, também, o processo de *clustering* adotado, os algoritmos implementados, juntamente com sua aplicação e avaliação nos casos estudados.
Foi apresentado também, um método de avaliação e análise de resultados baseado em gráficos do tipo dendrograma associado à matriz de cores.

Como derivação da solução apresentada para o problema de clustering de textos em português foi também apresentada uma solução para categorização de textos, a partir de categorias pré-definidas, por similaridade, no módulo de categorização assistida.

A proposta de solução apresentada é uma proposta que utiliza diferentes bibliotecas e visualizadores do tipo gratuito e de fonte aberta (*open-source*), com uma visualização que permite estudo e melhor entendimento dos dados textuais a custo zero.

Embora essa solução tenha sido desenvolvida com o objetivo de processar dados no idioma Português, a sua utilização para outros idiomas é conseguida sem grande esforço, já que toda a parte de pré-processamento funciona ou com arquivos externos, do tipo .txt, como no caso do dicionário e *stopwords*, ou chamando rotinas como é o caso do *stemming*. Do mesmo modo como foi utilizado dois *stemmers* distintos para a Língua Portuguesa, poderia-se ter utilizado um *stemmer* para outro idioma, sem nenhuma complicação.

5.1 Conclusões

A precisão conseguida tanto no processo de *clustering* como no processo de categorização está intimamente ligada à qualidade dos dados. Se os dados não possuem atributos em comum, é muito difícil mesmo para diferentes métodos detectar semelhanças entre os documentos. No entanto, alguns problemas podem ser diminuídos e até contornados pela utilização técnicas auxiliares às tarefas principais, como foi

mostrado nesta tese. Com o emprego do *stemming*, melhora-se a qualidade do processo de *clustering* em todas as situações estudadas.
Uma outra situação interessante é que cada coleção de documentos precisa de um número diferente de termos para ser descrita. Esse número de termos conforme foi verificado tem uma influência no índice de acerto. Isso significa dizer que se um número menor ou maior de termos que o número ótimo for utilizado a taxa de acerto pode ser bastante comprometida.

A solução implementada atendeu plenamente aos objetivos propostos. Os testes e avaliações apresentados mostraram a eficiência e aplicabilidade da solução proposta para o problema de *clustering* de textos para o Idioma Português.

A visualização dos resultados se mostrou bastante informativa e de grande valia para a interpretação dos mesmos.

5.2 Trabalhos Futuros

O sistema originado nesta tese foi implementado de forma que a adição de filtros (parsers) para o processamento de arquivos nos formatos HTML, DOC, XML e PDF é imediata e não exige nenhuma modificação nos atuais módulos.
De forma semelhante, o sistema pode ser adaptado para processar documentos diretamente de uma base de dados relacional sem nenhuma modificação apenas pela adição de extensões.
A vantagem do uso de gerenciadores mais sofisticados é a possibilidade do uso de bases distribuídas e o acesso otimizado aos documentos.
Outra extensão prevista é a adaptação do sistema a ambientes distribuídos paralelos. Isso pode ser obtido com pequenas modificações no sistema, já que os módulos são independendentes, com operações que podem ser aplicadas a diversos documentos simultaneamente.
Através do estudo de caso foi possível perceber a influência do número de termos na acurácia do *clustering* de documentos. Um estudo complementar mais aprofundado poderia determinar a relação do número de termos, número de *clusters*, número de

arquivos e quem sabe até do tamanho do arquivos, de forma a maximizar a eficiência do sistema.

5.3 Perspectivas Futuras

A riqueza dos textos, a complexidade dos problemas relativos à linguagem, e o esparsamento dos dados, juntos, impõem sérios desafios para a modelagem. Mesmo ao coletar dados a nível de palavras individuais aparecem problemas como, por exemplo, a dificuldade de obtenção de uma amostra verdadeiramente representativa de todas as palavras. Essa situação pode ocorrer por causa de amostras super-adequadas, generalizações e concatenações incorretas, etc. Em alguns casos, atividades humanas podem ser utilizadas para codificar algumas informações relevantes tais como regras gramaticais, inflexões de palavras, etc.
Existe, ainda, o bem conhecido problema de inteligência artificial: como extrair conhecimento de especialistas da área e converter em um formato utilizável pela máquina.

Mesmo considerando esses problemas ainda existentes nesta área para a implantação de sistemas mais complexos para o tratamento de textos, sabe-se que novas soluções para estes problemas não tão novos estão surgindo.

Esta área de pesquisa, no entanto, tende a crescer rapidamente devido principalmente a enorme quantidade de textos produzidas pela Web, e pelo grande interesse que essa disponibilidade acarreta de modo geral. Pode-se encontrar textos dos mais diversos assuntos disponíveis em qualquer tempo pela internet.

Além disto, as empresas redescobriram suas informações já armazenadas em textos e estão utilizando estas informações como uma vantagem competitiva em relação aos seus concorrentes. O *text mining* apresentou os principais recursos para que a inteligência competitiva pudesse ser efetivada na prática [3].

Referências Bibliográficas

6 Referências Bibliográficas

[1] Sullivan, D., *Document Warehousing and Text mining*, 1ª edition, John Wiley & Sons, New York, 2001.

[2] M. Steinbach, G. Karypis, and V. Kumar. "A comparison of document clustering techniques". *In KDD Workshop on Text mining*, 2000. http://citerserr.nj.nec.com/steinback00comparison.html

[3] Zanasi, A., "Web Mining through the Online Analyst", *Data Mining II*, N.Ebecken & C.A.Brebbia, 2000.

[4] Jain, A.K., Murty, M.N., Flynn, P.J., "Data Clustering: A Review". In ACM Computing Surveys, vol 31, pg 264-323, 1999.

[5] Koller, D. and Sahami, M., "Hierarchically classifying documents using very few words". In D. Fisher, editor, *Proceedings of (ICML) 97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, Morgan Kaufmann Publishers, San Francisco, US, 1997.

[6] Feldman, R., Dagan, I., "Knowledge discovery in textual databases (KDT)". *In proceedings of the First International Conference on Knowledge Discovery snd Data Mining (KDD-95)*, Montreal, Canada, August 20-21, AAAI Press, 112-117.

[7] Hearst, M.A., "Text Data Mining: Issues, techniques, and the relationship to information access", *Presentation notes for UW/MS workshop on data mining*, July 1997.

[8] Griffiths, A., Robinson, L. A., and Willett, P., "Hierarchical agglomerative clustering methods for automatic document classification". *Journal of Documentation*, 40(3):175–205, September 1984.

[9] Belkin, N.J. and Croft, W. B. "Information filtering and information retrieval: Two sides of the same coin?", *Communications of the ACM*, 35(12):29-38, 1992.

[10] O' Riordan, C. and Sorensen, H., "Information Filtering and Retrieval: An Overview", citeseer.nj.nec.com/483228.html

[11] Eagan, A., and Bender, L., "Spiders and worms and crawlers, oh my: Searching on the world wide web". *In Proceedings of the Conference sponsored by the Librarians Associations of the University of California*,1996.

[12] Salton, G., *Automatic Text Processing*, Addison-Wesley, 1989.

[13] Baeza-Yates, R., Ribeiro-Neto, B., *Modern Information Retrieval*", Addison Wesley Longman, 1999.

[14] Gärdenfors, P., *Conceptual Spaces*, MIT Press, 2000.

[15] Salton, G., Wong, A., Yang, C., "A vector space model for automatic indexing", *Communications of the ACM*, vol. 18, pp. 163-620, 1975.

[16] Salton, G., and McGill, M., *Introduction to Modern Information Retrieval*, McGraw Hill International, 1983.

[17] Faloutsos, C., Oard, D., *A Survey of Information Retrieval and Filtering Methods*, Technical Report, Information Filtering Project, University of Maryland, College Park, MD, http://citeseer.nj.nec.com/faloutsos96survey.html, 1996.

[18] Miller, G., "Wordnet: An online lexical database", *International Journal of Lexicography*, 3(4):235-312, 1996.

[19] Han J., Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2001.

[20] Files, J.R., Huskey, H.D., "An information retrieval system based on superimposed coding", *In Proceedings AFIPS FJCC*, 35:423-432, 1969.

[21] Harrison, M.C., "Implementation of the substring test by hashing", *CACM*, 14(12):777-779, December 1971.

[22] Haskin, R.L., "Special-purpose processors for text retrieval", *Database Engineering*, 4(1):16-29, September 1981.

[23] Scholtes, J. C., "Neural nets and their relevance for information retrieval", *ITLI Prepublication CL-91-02*, University of Amsterdam, Institute for Language, Logic and Information, Department of Computational Linguistics, October 1991.

[24] Harman, D., "How effective is suffixing?", *Journal of the American Society for Information Science*, Volume 42, Number 1, pages 7-15, 1991.

[25] Porter, M., "An algorithm for suffixing stripping", Program, Volume 14, Number 3, pages 130-137, 1980.

[26] Lovins, J. B., "Development of a stemming algorithm", *Mechanical Translation and Computacional Linguistics*, Volume 11, Number 1-2, pages 22-31, 1968.

[27] Han, E. H., Boley, B., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J., "Webace: a web agent for document categorization and exploration". *In Proceedings of the second international conference on Autonomous agents*, pages 408–415. ACM Press, 1998.

[28] Elmasre, R., Navathe, S.B., *Fundamentals of Database Systems*, Benjamim/Cummins, USA, 1989.

[29] Fox, E.A., Harman, D.K., Baeza-Yates, and Lee, W.C., "Inverted Files", In Frakes and Baeza-Yates [30], chapter 3, pag. 28–43.

[30] Frakes, W.B., Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, 1992.

[31] Bell, T.C., Moffat, A., Nevill-Manning, C.G., Witten, I.H., and Zobel, J., "Data Compression in full-text retrieval systems, *Journal of American Society for Information Science*, 44(9):508-531, October, 1993.

[32] Bookstein, A., Klein, S.T., Raita, T., "Model based concordance compression", In J. A Storer and M. Cohn, editors, *Proc. IEEE Data Compression Conference*, pages, 82-91, Snowbird, Utah, IEEE Computer Society Press, Los Alamitos, California, March 1992.

[33] Choueka, Y., Fraenkel, A.S., Klein, S.T., "Compression of concordance in full text retrieval systems. *In Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 597-612, Grenoble, France, June 1988, ACM Press , New York, 1988.

[34] Fraenkel, A.S., Klein, S.T., "Novel compression of sparse bit-strings – Preliminary Report". *In A. Apostolico and Z. Galil, editors, Combinatorial Algorithms on Words*, Volume 12, NATO ASI Series F, pages 169-183, Berlin, Springer-Verlag, 1985.

[35] Appelt, D. E. and Israel, D. J., "Introduction to Information Extraction Technology". *In Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.

[36] Daille, B., "Study and Implementation of combined Techniques for Automatic Extraction of Terminology". *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994.

[37] Leek, T. R., *Information Extraction Using Hidden Markov Models*, Master of Science of Computer Science Thesis, University of California, San Diego, 1997.

[38] Mitchell, T., *Machine Learning*, MacGrall Hill, New York, 1996.

[39] Church, K.W., and Hanks, P., "Word Association Norms, mutual information and Lexicography". *In Proceedings of ACL*, 27, pages 76-83, Vancouver, Canada, 1989.

[40] Zechner, K., "A literature survey on information extraction and text summarization", *Term paper*, Carnegie Mellon University, 1997. http://www.contrib.andrew.cmu.edu/ ~zechner/klaus.html

[41] Hahn, U. and Mani, I., "The challenges of Automatic Summarization". *IEEE Computer*, Vol.33, No.11, November, 2000.

[42] Yang, Y., Pedersen, J.P., "A Comparative Study on Feature Selection in Text Categorization". *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pp. 412-420, 1997.

[43] Chidanand, A., Damerau, F., and Weiss, S.M., "Automated Learning of Decision Rules for Text Categorization", *ACM Transaction on Information Systems*, Vol.12, No.3, July 1994.

[44] Han, J and Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, 2001.

[45] Dumais, S., Platt, J., Heckerman, D., and Sahami, M., "Inductive learning algorithms and representations for text categorization". *In Proceedings of the 1998 ACM 7th international conference on information and knowledge management*, pages 148, 155, 1998.

[46] Sebastiani, F., *Machine learning in automated text categorisation: a survey*, Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, C.N.R., Pisa, IT, 1999. http://citeseer.nj.nec.com/sebastiani99machine.html

[47] Tzeras, K., and Hartmann, S., "Automatic indexing based on Bayesian inference networks". *In Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, Pittsburg, PA, 22-34, 1993.

[48] Larkey, L.S. , "A patent search and classification system". *In Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, 90-95,Berkley, CA, 1999.

[49] Androutsopoulos, I., Koutsias, J. Chandrinos, K. V. and Spyropoulos, "An experimental comparison between Bayesian and keyword and based anti-spam filtering with personal email messages". *In Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, 160-167, C.P., Athens, Greece 2000.

[50] Roth, D., "Learning to resolve natural language ambiguities: A unified approach". *In Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, 806-813, Madison, WI, 1998.

[51] Dumais, S. T. and Chen, H. 2000. "Hierarchical classifications of web content". *In Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, 256-263, Athens, Greece, 2000.

[52] Lewis, D.D 1998. "Naïve (Bayes) at forty: The independence assumption in information retrieval". *In Proceedings of ECML-98, 10th European Conference on Machine Learning*, 4-15, Chemnitz, Germany, 1998.

[53] Moulinier, I. And Ganascia, J. 1996. "Applying an existing machine learning algorithm to text categorization". *In connectionist, statistical and symbolic approaches to learning for Natural Language Processing*, S. Wermter, E. Riloff, and G. Schaler, eds. Springer Verley, Heidelberg, Germany, 343-354.

[54] Yang, Y. and Liu, X., "A re-examination of text categorization methods". *In Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, , 42-39, Berkley, CA, 1999.

[55] Cohen, W.W. and Singer, Y. 1999. "Context-sensitive learning methods for text categorization", *ACM Trans, Inform. Syst*., 17, 2 , 141-173.

[56] Lam, S.L. and Lee, D.L. 1999. "Feature reduction for neural network based test categorization". *In Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Applications*, 195-202, Hsinchu, Twaiwan, 1999.

[57] Yang, Y., "Expert network: effective and efficient learning from human decisions in text categorization and retrieval". *In Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*), 13-22, 1994, Dublin, Ireland, 1994.

[58] Joachims, T., "Text Retrieval with support vector machine: learning with many relevant features". *In Proceedings of ECML-98, 10th European Conference on Machine Learning* , 137-142, Chemnitz, Germany, 1998.

[59] Vasileios, H., Gravano, L., and Maganti, A., "An Investigation of Linguistic Features and Clusters Algorithms for Topical Document Clustering". *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[60] Kosala and Blockeel. "Web mining research: A survey", *SIGKDD Explorations: Newsletter of the Special Interest Group SIG on Knowledge Discovery & Data Mining*, 2, 2000.

[61] Rijsbergen C. J. van, *Information Retrieval*, Dept. of Computer Science, University of Glasgow, Butterworth, London, 2 edition, 1979.

[62] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W., "Scatter/gather: A cluster-based approach to browsing large document collections". *In Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[63] Zamir, O., Etzioni, O., Madani, O., and Karp, R. M., "Fast and intuitive clustering of web documents", *In KDD'97*, pages 287–290, 1997.

[64] Golub, G. and Van Loan, C., *Matrix Computations*, Johns-Hopkins, Baltimore, Maryland, second edition, 1989.

[65] Dumais, S., "Improving the retrieval of information from external sources", *Behavior Research Methods, Instruments, & Computers*, 23(2):229--236, 1991.

[66] Koll, M., “WEIRD: An approach to concept-based information retrieval”, *SIGIR Forum*, 13:32--50, 1979.

[67] Borko, H. and Bernick, M., “Automatic document classification”. *In Journal of the Association for Computing Machinery*, 10:151--162, 1963.

[68] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R., “Indexing by latent semantic analysis”. In *Journal of the American Society for Information Science*, 41(6):391--407, 1990.

[69] Salton, G. and Buckley, C., “Improving retrieval performance by relevance feedback”. *Journal of the American Society for Information Science*, 41(4):288--297, 1990.

[70] Florek, K., Lukaszewicz, J., Perkal, J., and Zubrzycki, S. (1951a), "Sur la Liaison et la Division des Points d'un Ensemble Fini," *Colloquium Mathematicae*, 2, 282 -285.

[71] Florek, K., Lukaszewicz, J., Perkal, J., and Zubrzycki, S. (1951b), "Taksonomia Wroclawska," *Przeglad Antropol.*, 17, 193 -211.

[72] McQuitty, L.L. (1957), "Elementary Linkage Analysis for Isolating Orthogonal and Oblique Types and Typal Relevancies," *Educational and Psychological Measurement*, 17, 207 -229.

[73] Sneath, P.H.A. (1957), "The Application of Computers to Taxonomy," *Journal of General Microbiology*, 17, 201 -226.

[74] Hair, J.F., Anderson, R.E., and Tatham, R.L., *Multivariate Data Analysis with Readings*, 2nd Edition, MacMillan Publishing Co., New York, PP 303.

[75] Ward, J.H., "Hierarchical Grouping to Optimize an Objective Function". *Journal of the American Statistical Association*, 58, 236 –244, 1963.

[76] Ross, K., Srivastava, D., Fast Computation of sparse datacubes. In Proceedings of 23th International Conference on Very Large Databases (VLDB97), pages 116-125, Athens, Greece, Morgan Kaufmann, August 1997.

[77] Jain, A.K. and Dubes, R.C., *Algorithms for Clustering Data*, Prentice Hall, 1988.

[78] Charniak, E., *Statistical Language Learning*, MIT Press, 1993.

[79] Finch, S. and Chater, N., “Unsupervised Methods for Finding Linguistic Categories”. *In Artificial Neural Networks*, 2, pp.II-1365-1368, North Holland, 1992.

[80] Buckley, C. and Lewit, A. F., “Optimizations of inverted vector searches”, *SIGIR ’85*, pp. 97-110, 1985.

[81] Kowalski, G., *Information Retrieval Systems – Theory and Implementation*, Kluwer Academic Publishers, 1997.

[82] Cutting, D.R., Karger, D.R., Pedersen, J.O., and Tukey, J. W., "Scatter/Gather: A cluster-based approach to browsing large document collections". *SIGIR '92*, 318-329, 1992.

[83] Koller, D. and Sahami, M., "Hierarchically Classifying Document using very few words". *In Proceedings of the 14th International Conference on Machine Learning (ML)*, Nashville, Tenessee, 170-178, July 1997.

[84] Vesanto, J., Himberg, J., Alhoniemi, E., and Parhankangas, J., *Som Toolbox para Matlab 5*, Report A57, Helsinki University of Technology, Neural Network Research Centre, Espoo, Finland, April, 2000.

[85] Wise, J.A., "The Ecological Approach to Text Visualization", *Journal of the American Society for Information Science*, vol.50, no.13, pp. 1224-1233, 1999.

[86] Allan, J., *Automatic Hypertext Construction*, PHD thesis, Cornell University, January, 1995. Also technical report TR95-1484.

[87] Allan, J., "Building hypertext using information retrieval", *Information Processing and Management*, 33(2):145-149, 1997.

[88] Dubin, D., "Document analysis for visualization". *In Proceedings of ACM SIGIR*, pages 199-204, July 1995.

[89] Hemmje, M., Kunkel, C., and, Willet, A., "Lyberworld – a visualization user interface supporting fulltext retrieval". *In Proceedings of ACM SIGIR*, pages 254–259, July, 1994.

[90] Allan, J., Leouski, A.V., Swan, R.C., *Interactive Cluster Visualization for Information Retrieval*, Technical Report IR-116, University of Massachusetts, Amherst, 1997.

[91] Chalmers, M., and Chitson, P., "Bead: Explorations in information visualization". *In Proceedings of ACM SIGIR*, pages 330-337, June 1992.

[92] http://snowball.tartarus.org/portuguese/stemmer.html, acessado em agosto de 2004.

[93] Moreira Orengo, V. and Huyck, C.R. "A Stemming Algorithm for The Portuguese Language". *In Proceedings of the SPIRE Conference*, Laguna de San Raphael, Chile, November 13-15, 2001.

[94] Chaves, M. S., Um estudo e apreciação sobre algoritmos de stemming. *In: IX JORNADAS IBEROAMERICANAS DE INFORMÁTICA*, agosto de 2003, Cartagena de Indias, Colômbia.

[95] http://snowball.tartarus.org/portuguese/voc.txt, acessado em agosto de 2004.

[96] Cunha, C. And Lindley-Cintra, L. Nova Gramática do Português Contemporâneo. Ed. Nova Fronteira, Rio de Janeiro, 1985, 719p.

[97] Macambira, J.R., A Estrutura Morfo-Sintática do Português. Ed. Pioneira, São Paulo, 1999.

[98] M.J.L. de Hoon, S. Imoto, J. Nolan, and S. Miyano: "Open Source Clustering Software", Bioinformatics, 2004, in press

[99] Perl2exe Users Manual, Indigo Star Softwares, Ontario, Canada, 29 de junho de 2004, http://www.indigostar.com

[100] Zhao, Y., Karypis, G., "Criterion Functions for Document Clustering: Experiments and Analysis", Technical Report TR#1--40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.

[101] Zhao, Y., Karypis, G., "Evaluation of Hierarchical Clustering Algorithms for Document Datasets", pages 515-524, ACM Press, 2002.

[102] Croft. W.B. and Das, R. "Experiments with query acquisition and use in document retrieval systems". In Proceedings of the ACM SIGIR, Conference on Research and Development in Information Retrieval (1990), 349-368.

[103] Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 3rd edition, 1989.

[104] Wohl, A.D., "Intelligent Text Mining Creates Business Intelligence". In: IBM Business Intelligence Solutions CD, CD_ROM, EUA, 1998.

[105] Zanasi, A., "Text Mining: the new competitive intelligence frontier". In VST2001 Barcelona Conference Proceedings – IRIT, 2001.

[106] http://www.spss.com/textsmart, acessado em agosto de 2004.

[107] Inxight SmartDiscovery, "The new Standard for Interprise Information Retrieval", Inxight Product White Paper, Inxight Software, April, 2003.

[108] Inxight SmartDiscovery, "From documents to Information: A New Model for Informartion Retrieval", Inxight Product White Paper, Inxight Software, November, 2004.

[109] Pohs, W., Pinder, G., Dougherty, C., White, M., "The Lotus Knowledge Discovery System: Tools and Experiences", IBM Systems Journal, Volume 40, Number 4, 2001.

[110] Dodge, M., "NewsMaps: Topographic Mapping of Information", 2000. http://mappa.mundi.net/maps/maps_015/-23k

[111] http://fqspl.com.pl/cambio.htm, acessado em setembro de 2004.

[112] Ananyan, S., “Foreward to Document Warehousing and Text Mining Book”, Megaputer Intelligence, Boomington, IN, January, 2001.

[113] Neri, F., “Mining Information in Large Textual Data Sets”, Synthema Solutions, Pisa, Italy, 1996.

[114] Hartigan, J.A., *Clustering Algorithms*, Jonh Wiley & Sons, New York, 1975.

[115] Thede, E., “Where the *@#! Is That? The Art of Text Query”, PC AI Magazine, vol. 14.1, Jan/Feb 2000.

[116] http://www.wizsoft.com, acessado em setembro de 2004.

[117] http://www.canis.uiuc.edu/projects/interspace/technical/canis-report-0006.html, acessado em setembro de 2004.

[118] http://www.entrieva.com/entrieva/products/semiomap.asp?Hdr=semiomap, acessado em agosto de 2004.

[119] http://www.textwise.com/government/index.html, acessado em agosto de 2004.

[120] http://www.perl.org, acessado em setembro de 2004.

[121] http://www.python.org, acessado em março de 2004.

[122] http://www.ruby.org, acessado em março de 2004.

[123] Sachs, L., *Applied Statistics*, 2nd. Edition, Springer-Verlag, New York, 1984.

[124] Salton, G., Wong, A. Yang, C.S., “A vector Space Model for Information Retrieval”. Comunications of the ACM, 18(11):613-620, November, 1975.

Apêndices

Apêndice A

Cálculo da Distância entre *Clusters*

Prototype

double *cluster*distance (int *nrows*, int *ncolumns*, double** *data*, int** *mask*, double *weight*[], int *n1*, int *n2*, int *index1*[], int *index2*[], char *dist*, char *method*, int *transpose*);
returns the distance between two *clusters*.

Arguments

• int *nrows*;
The number of rows in the *data* matrix, equal to the number of genes in the gene expression experiment.
• int *ncolumns*;
The number of columns in the *data* matrix, equal to the number of microarrays in the gene expression experiment.
• double** *data*;
The data array containing the gene expression data. Genes are stored row-wise, while microarrays are stored column-wise. Dimension: [*nrows*][*ncolumns*].
• int** *mask*;
This array shows which elements in the *data* array, if any, are missing. If *mask*[i][j]==0, then *data*[i][j] is missing. Dimension: [*nrows*][*ncolumns*].
• double *weight*[];
The weights that are used to calculate the distance. Dimension: [*ncolumns*] if *transpose*== 0; [*nrows*] if *transpose*==1.
• int *n1*;
The number of elements in the first *cluster*.
• int *n2*;
The number of elements in the second *cluster*.
• int *index1*[];
Contains the indeces of the elements belonging to the first *cluster*. Dimension: [*n1*].
• int *index2*[];
Contains the indeces of the elements belonging to the second *cluster*. Dimension: [*n2*].
• char *dist*;
Specifies which distance measure is used. See Chapter 2 [Distance functions], page 2.
• char *method*;
Specifies how the distance between *clusters* is defined:
‘a’ Distance between the two *cluster* centroids (arithmetic mean);
‘m’ Distance between the two *cluster* centroids (median);
‘s’ Shortest pairwise distance between elements in the two *clusters*;
‘x’ Longest pairwise distance between elements in the two *clusters*;
‘v’ Average over the pairwise distances between elements in the two *clusters*.
• int *transpose*;
If *transpose*==0, the distances between rows in the data matrix are calculated. Otherwise, the distances between columns are calculated.

Matriz de Distâncias

Prototype

double** distancematrix (int *nrows*, int *ncolumns*, double** *data*, int** *mask*, double *weight*[], char *dist*, int *transpose*);

returns the distance matrix stored as a ragged array. If insu±cient memory is available to store the distance matrix, a NULL pointer is returned, and all memory allocated thus far is freed.

Arguments

• int *nrows*;
The number of rows in the data matrix, equal to the number of genes in the gene expression experiment.
• int *ncolumns*;
The number of columns in the data matrix, equal to the number of microarrays in the gene expression experiment.
• double** *data*;
The data array containing the gene expression data. Genes are stored row-wise, while microarrays are stored column-wise. Dimension: [*nrows*][*ncolumns*].
• int** *mask*;
This array shows which elements in the *data* array, if any, are missing. If *mask*[i][j]==0, then *data*[i][j] is missing. Dimension: [*nrows*][*ncolumns*].
• double *weight*[];
The weights that are used to calculate the distance. Dimension: [*ncolumns*] if *transpose*== 0; [*nrows*] if *transpose*==1.
• char *dist*;
Specifies which distance measure is used. See Chapter 2 [Distance functions], page 2.
• int *transpose*;
If *transpose*==0, the distances between the rows in the data matrix are calculated. Otherwise, the distances between the columns are calculated.

Inicialiação do Algoritmo K-means

Prototype

void randomassign (int *nclusters*, int *nelements*, int *clusterid*[]);

Arguments

• int *nclusters*;
The number of *clusters*.
• int *nelements*;
The number of elements (genes or microarrays) to be *cluster*ed.
• int *clusterid*[];
The *cluster* number to which each element was assigned. Space for this array should be allocated before calling randomassign. Dimension: [*nelements*].

Encontrando o Meio do *Cluster*

Prototype

void get*cluster*mean (int *nclusters*, int *nrows*, int *ncolumns*, double** *data*, int** *mask*, int *clusterid*[], double** *cdata*, int** *cmask*, int *transpose*);

Arguments

• int *nclusters*;
The number of *clusters*.
• int *nrows*;
The number of rows in the data matrix, equal to the number of genes in the gene expression experiment.
• int *ncolumns*;
The number of columns in the data matrix, equal to the number of microarrays in the gene expression experiment.
• double** *data*;
The data array containing the gene expression data. Genes are stored row-wise, while microarrays are stored column-wise. Dimension: [*nrows*][*ncolumns*].
• int** *mask*;

This array shows which elements in the *data* array, if any, are missing. If *mask*[i][j]==0, then *data*[i][j] is missing. Dimension: [*nrows*][*ncolumns*].
• int *clusterid*[];
The *cluster* number to which each item belongs. Each element in this array should be between 0 and *nclusters*-1 inclusive. Dimension: [*nrows*] if *transpose*==0, or [*ncolumns*] if *transpose*==1.
• double** *cdata*;
This matrix stores the centroid information. Space for this matrix should be allocated before calling get*cluster*mean. Dimension: [*nclusters*][*ncolumns*] if *transpose*== 0 (row-wise *clustering*), or [*nrows*][*nclusters*] if *transpose*==1 (columnwise *clustering*).
• int** *cmask*;
This matrix stores which values in *cdata* are missing. If *cmask*[i][j]==0, then *cdata*[i][j] is missing. Space for *cmask* should be allocated before calling get*cluster*mean. Dimension: [*nclusters*][*ncolumns*] if *transpose*==0 (row-wise *clustering*), or [*nrows*][*nclusters*] if *transpose*==1 (column-wise *clustering*).
• int *transpose*;
This flag indicates whether row-wise (gene) or column-wise (microarray) *clustering* is being performed. If *transpose*==0, rows (genes) are being *cluster*ed. Otherwise, columns (microarrays) are being *cluster*ed.

Encontrando a média do *Cluster*

Prototype
void get*cluster*median (int *nclusters*, int *nrows*, int *ncolumns*, double** *data*, int** *mask*, int *clusterid*[], double** *cdata*, int** *cmask*, int *transpose*);

Arguments
• int *nclusters*;
The number of *clusters*.
• int *nrows*;
The number of rows in the data matrix, equal to the number of genes in the gene expression experiment.
• int *ncolumns*;
The number of columns in the data matrix, equal to the number of microarrays in the gene expression experiment.
• double** *data*;
The data array containing the gene expression data. Genes are stored row-wise, while microarrays are stored column-wise. Dimension: [*nrows*][*ncolumns*].
• int** *mask*;
This array shows which elements in the *data* array, if any, are missing. If *mask*[i][j]==0, then *data*[i][j] is missing. Dimension: [*nrows*][*ncolumns*].
• int *clusterid*[];
The *cluster* number to which each item belongs. Each element in this array should be between 0 and *nclusters*-1 inclusive. Dimension: [*nrows*] if *transpose*==0, or [*ncolumns*] if *transpose*==1.
• double** *cdata*;
This matrix stores the centroid information. Space for this matrix should be allocated before calling get*cluster*median. Dimension: [*nclusters*][*ncolumns*] if *transpose*== 0 (row-wise *clustering*), or [*nrows*][*nclusters*] if *transpose*==1 (columnwise *clustering*).
• int** *cmask*;
This matrix stores which values in *cdata* are missing. If *cmask*[i][j]==0, then *cdata*[i][j] is missing. Space for *cmask* should be allocated before calling get*cluster*median. Dimension: [*nclusters*][*ncolumns*] if *transpose*==0 (row-wise *clustering*), or [*nrows*][*nclusters*] if *transpose*==1 (column-wise *clustering*).
• int *transpose*;
This flag indicates whether row-wise (gene) or column-wise (microarray) *clustering* is being performed. If *transpose*==0, rows (genes) are being *cluster*ed. Otherwise, columns (microarrays) are being *cluster*ed.

Encontrando o Medóide do *Cluster*

Prototype

void get*cluster*medoid(int *nclusters*, int *nelements*, double** *distance*, int *clusterid*[], int *centroids*[], double *errors*[]);

Arguments

• int *nclusters*;
The number of *clusters*.
• int *nelements*;
The total number of elements that are being *cluster*ed.
• double** *distmatrix*;
The distance matrix. The distance matrix is symmetric and has zeros on the diagonal. To save space, the distance matrix is stored as a ragged array. Dimension: [*nelements*][] as a ragged array. The number of columns in each row is equal to the row number (starting from zero). Accordingly, the first row always has zero columns.
• int *clusterid*[];
The *cluster* number to which each element belongs. Dimension: [*nelements*].
• int *centroid*[];
For each *cluster*, the element of the item that was determined to be its centroid. Dimension: [*nclusters*].
• int *errors*[];
For each *cluster*, the sum of distances between the items belonging to the *cluster* and the *cluster* centroid. Dimension: [*nclusters*].

Encontrando a solução Ótima – K-means e K-Medians

Prototype

void k*cluster* (int *nclusters*, int *nrows*, int *ncolumns*, double** *data*, int** *mask*, double *weight*[], int *transpose*, int *npass*, char *method*, char *dist*, int *clusterid*[], double** *cdata*, double* *error*, int* *ifound*);

Arguments

• int *nclusters*;
The number of *clusters k*.
• int *nrows*;
The number of rows in the data matrix, equal to the number of genes in the gene expression experiment.
• int *ncolumns*;
The number of columns in the data matrix, equal to the number of microarrays in the gene expression experiment.
• double** *data*;
The data array containing the gene expression data. Genes are stored row-wise, while microarrays are stored column-wise. Dimension: [*nrows*][*ncolumns*].
• int** *mask*;
This array shows which elements in the *data* array, if any, are missing. If *mask*[i][j]==0, then *data*[i][j] is missing. Dimension: [*nrows*][*ncolumns*].
• double *weight*[];
The weights that are used to calculate the distance. Dimension: [*ncolumns*] if *transpose*== 0; [*nrows*] if *transpose*==1.
• int *transpose*;
This flag indicates whether row-wise (gene) or column-wise (microarray) *clustering* is being performed. If *transpose*==0, rows (genes) are being *cluster*ed. Otherwise, columns (microarrays) are being *cluster*ed.
• int *npass*;
The number of times the EM algorithm should be run. If *npass* > 0, each run of the EM algorithm uses a diferent (random) initial *clustering*. If *npass* == 0, then the EM algorithm is run with an initial *clustering* specified by *clusterid*. For *npass* > 0,

items are reassigned to *clusters* in a randomized order. Since the *cluster* centroids are recalculated only once after all items have been considered for reassignment, the order of item reassignment is relevant only when the last item in a *cluster* is about to be reassigned to a di®erent *cluster*. To prevent *clusters* from becoming empty, such reassignments are not allowed; which items are reassigned may therefore depend on the order in which items are considered. For *npass*==0, the EM algorithm is run only once, without randomizing the order in which items are reassigned to *clusters*, using the initial *clustering* as specified by *clusterid*. The order in which items are reassigned is identical to the order in which items are given in the data matrix.

• char *method*;

Specifies whether the arithmetic mean (*method*=='a') or the median (*method*=='m') should be used to calculate the *cluster* center.

• char *dist*;

Specifies which distance function should be used. The character should correspond to one of the distance functions that are available in the C *Clustering Library*. See Chapter 2 [Distance functions], page 2.

• int *clusterid*[];

This array will be used to store the *cluster* number to which each item was assigned by the *clustering* algorithm. Space for *cluster*id should be allocated before calling k*cluster*. If *npass*==0, then the contents of *clusterid* on input is used as the initial assignment of items to *clusters*; on output, *clusterid* contains the optimal *clustering* solution found by the EM algorithm. Dimension: [*nrows*] if *transpose*==0, or [*ncolumns*] if *transpose*==1.

• double** *cdata*;

This matrix stores the centroid information. Space for *cdata* should be allocated before

calling k*cluster*. Dimension: [*nclusters*][*ncolumns*] if *transpose*==0 (row-wise *clustering*), or [*nrows*][*nclusters*] if *transpose*==1 (column-wise *clustering*).

• double* *error*;

The sum of distances of the items to their *cluster* center after *k*-means *clustering*, which can be used as a criterion to compare *clustering* solutions produced in di®erent calls to k*cluster*.

• int* *ifound*;

Returns how often the optimal *clustering* solution was found.

K-Medoids

Prototype

Arguments

• int *nclusters*;

The number of *clusters* to be found.

• int *nelements*;

The number of elements to be *cluster*ed.

• double** *distmatrix*;

The distance matrix. The distance matrix is symmetric and has zeros on the diagonal. To save space, the distance matrix is stored as a ragged array. Dimension: [*nelements*][] as a ragged array. The number of columns in each row is equal to the row number (starting from zero). Accordingly, the first row always has zero columns.

• int *npass*;

The number of times the EM algorithm should be run. If *npass* > 0, each run of the EM algorithm uses a di®erent (random) initial *clustering*. If *npass* == 0, then the EM algorithm is run with an initial *clustering* specified by *clusterid*. The order in which items are reassigned is identical to the order in which items are given in the distance matrix.

• int *clusterid*[];

This array will be used to store the *cluster* number to which each item was assigned by the *clustering* algorithm. Space for *cluster*id should be allocated before calling k*cluster*. On input, if *npass*==0, then *clusterid* contains the initial *clustering* assignment from which the *clustering* algorithm starts; all numbers in *clusterid* should be

between 0 and *nelements*-1 inclusive. If *npass*!=0, *clusterid* is ignored on input. On output, *clusterid* contains the number of the *cluster* to which each item was assigned in the optimal *clustering* solution. On output, the number of a *cluster* is defined as the item number of the centroid of the *cluster*. Dimension: [*nelements*].
Chapter 5: Partitioning algorithms 19
• double* *error*;
The sum of distances of the items to their *cluster* center after *k*-means *clustering*, which can be used as a criterion to compare *clustering* solutions produced in di®erent calls to kmedoids.
• int* *ifound*;
Returns how often the optimal *clustering* solution was found.

Podando a Árvore de *Clustering* Hierárquico

Prototype
void cuttree (int *nelements*, int *tree*[][2], int *nclusters*, int *clusterid*[]);
Arguments
• int *nelements*;
The number of elements whose *clustering* results are stored in the *tree* hierarchical *clustering* result.
• int *tree*[][2];
The hierarchical *clustering* solution. Each row in the matrix describes one linking event. The two columns contain the numbers of the nodes that were joined. The original elements are numbered {0, . . . , *nelements*-1}, nodes are numbered {-1, . . . , -(*nelements*-1)}. Note that the number of nodes is one less than the number of elements. The cuttree routine performs some error checking of the structure of the *tree* argument in order to avoid segmentation faults. However, errors in the structure of *tree* that would not result in segmentation faults will in general not be detected. Dimension: [*nelements*-1][2].
• int *nclusters*;
The desired number of *clusters*. The number of *clusters* should be positive, and less than or equal to *nelements*.
• int *clusterid*[];
The *cluster* number to which each element is assigned. Memory space for *clusterid* should be allocated before cuttree is called. Dimension: [*nelements*].

Apêndice B

Regras de Remoção de Sufixos

Para um melhor entendimento das tabelas de regras é apresentado um exemplo de regra comentado :

> “inho”, 3, “ ”, {“caminho”, “carinho”, “cominho”, “golfinho”, “padrinho”, “sobrinho”, “vizinho”}

Quando “inho” é um sufixo que denota um diminutivo, 3 é o tamanho mínimo para o *stem*, o que evita que palavras como “linho”sofram o *stem* e palavras entre colchetes são as exceções para esta regra, isto é, elas terminam com o sufixo, mas não só diminutivos. Todas as outras palavras que terminam em –inho e que são mais longas que 6 caracteres sofrerão *stemming*. Não existe sufixo de reposição para essa regra.

A seguir são mostradas as tabelas contendo as regras para o *stemming* em Português. A lista de exceções não é apresentada aqui.

Regras de Redução de Plural			
Sufixo a Remover	**Tamanho Mínimo Stem**	**Substituição**	**Exemplo**
"ns"	1	"m"	bons → bom
"ões"	3	"ão"	balões → balão
"ães"	1	"ão"	capitães → capitão
"ais"	1	"al"	normais → normal
"éis"	2	"el"	papéis → papel
"eis"	2	"el"	amáveis → amável
"óis"	2	"ol"	lençóis → lençol
"is"	2	"il"	barris → barril
"les"	3	"l"	males → mal
"res"	3	"r"	mares → mar
"s"	2		casas → casa

Regras de Redução de Feminino			
Sufixo a Remover	**Tamanho Mínimo Stem**	**Substituição**	**Exemplo**
"ona"	3	"ão”	chefona → chefão
“ã”	2	"ão”	vilã → vilão
"ora"	3	"or"	professora → professor
"na"	4	"no”	americana → americano
"inha"	3	"inho"	chilena → chileno

"esa"	3	"ês"	sozinho → sozinha
"osa"	3	"oso"	inglesa → inglês
"íaca"	3	"íaco"	famosa → famoso
"ica"	3	"ico"	maníaca → maníaco
"ada"	2	"ado"	prática → prático
"ida"	3	"ido"	mantida → mantido
"ída"	3	"ido"	cansada → cansado
"ima"	3	"imo"	prima → primo
"iva"	3	"ivo"	passiva → passivo
"eira"	3	"eiro"	primeira → primeiro

Adverb Reduction Rule			
Sufixo a Remover	**Tamanho Mínimo Stem**	**Substituição**	**Exemplo**
"mente"	4		felizmente → feliz

Augmentative/Dimunutive Reduction Rules			
Sufixo a Remover	**Tamanho Mínimo Stem**	**Substituição**	**Exemplo**
"díssimo"	5		cansadíssimo → cansad
"abilíssimo"	5		amabilíssimo → ama
"íssimo"	3		fortíssimo → fort
"ésimo"	3		
"érrimo"	4		chiquérrimo → chiqu
"zinho"	2		pezinho → pe
"quinho"	4	"c"	maluquinho → maluc
"uinho"	4		amiguinho → amig
"adinho"	3		cansadinho →cansad
"inho"	3		carrinho → carr
"alhão"	4		grandalhão→ grand
"uça"	4		dentuça → dent
"aço"	4		ricaço→ ric
"adão"	4		casadão→ cans
"ázio"	3		corpázio→ corp
"arraz"	4		pratarraz→ prat
"arra"	3		bocarra → boc
"zão"	2		calorzão→ calor
"ão"	3		meninão → menin

Noun Reduction Rules							
Sufixo a Remover	**Tam Min**	**Subs**	**Exemplo**	**Sufixo a Remover**	**Tam Min**	**Subs**	**Exemplo**
"encialista"	4		existencialista → exist	"izaç"	5		concretizaç→concret
"alista"	5		minimalista → minim	"aç"	3		alegaç→aleg
"agem"	3		contagem → cont	"iç"	3		aboliç→abol
"iamento"	4		genrenciamento → gerenc	"ário"	3		anedotário→anedot
"amento"	3		monitoramento →monit	"ério"	6		ministério→minist
"imento"	3		nascimento → nasc	"ês"	4		chinês → chin
"alizado"	4		comercializado→comerci	"eza"	3		beleza→ bel
"atizado"	4		traumatizado → traum	"ez"	4		rigidez→rigid

"izado"	5		alfabetizado→alfabet	"esco"	4		parentesco→parent
"ativo"	4		associativo → associ	"ante"	2		ocupante→ocup
"tivo"	4		contraceptivo→contracep	"ástico"	4		bombástico→bomb
"ivo"	4		esportivo→esport	"ático"	3		problemático→problem
"ado"	2		abalado → abal	"ico"	4		polêmico→polêm
"ido"	3		impedido → imped	"ividade"	5		produtividade→produt
"ador"	3		ralador → ral	"idade"	5		profundidade→profund
"edor"	3		entendedor→entend	"oria"	4		aposentadoria→aposentad
"idor"	4		cumpridor→cumpr	"encial"	5		existencial→exist
"atória"	5		obrigatória→obrig	"ista"	4		artista→art
"or"	2		produtor→produt	"quice"	4	"c"	maluquice→maluc
"abilidade"	5		comparabilidade→compar	"ice"	4		chatice→ chat
"icionista"	4		abolicionista→abol	"íaco"	3		demoníaco→demon
"cionista"	5		intervencionista→interven	"ente"	4		decorrente→decorr
"ional"	4		profissional→profiss	"inal"	3		criminal→crim
"ência"	3		referência→refer	"ano"	4		americano→ americ
"ância"	4		repugnância→repugn	"ável"	2		amável→ am
"edouro"	3		abatedouro→abat	"ível"	5		combustível→combust
"queiro"	3	"c"	fofoqueiro→fofoc	"ura"	4		cobertura→cobert
"eiro"	3		brasileiro→brasil	"ual"	3		consensual→consens
"oso"	3		gostoso→gost	"ial"	3		mundial→mund
"alizaç"	5		comercializaç→comerci	"al"	4		experimental→experiment
"ismo"	3		consumismo →consum				

Verb Reduction Rules

Sufixo a Remover	Min Size	Subs	Exemplo	Sufixo a Remover	Min Size	Subs	Exemplo
"aríamo"	2		cantaríamo → cant	"eria"	3		beberia→ beb
"ássemo"	2		cantássemo → cant	"ermo"	3		bebermo → beb
"eríamo"	2		beberíamo → beb	"esse"	3		bebesse→ beb
"êssemo"	2		bebêssemo → beb	"este"	3		bebeste→ beb
"iríamo"	3		partiríamo → part	"íamo"	3		bebíamo → beb
"íssemo"	3		partíssemo →part	"iram"	3		partiram→ part
"áramo"	2		cantáramo → cant	"íram"	3		concluíram→ conclu
"árei"	2		cantárei→ cant	"irde"	2		partirde→ part
"aremo"	2		cantaremo → cant	"irei"	3		partírei→ part
"ariam"	2		cantariam→ cant	"irem"	3		partirem→ part
"aríei"	2		cantaríei→ cant	"iria"	3		partiria→ part
"ássei"	2		cantássei→ cant	"irmo"	3		partirmo → part
"assem"	2		cantassem→ cant	"isse"	3		partisse→ part
"ávamo"	2		cantávamo → cant	"iste"	4		partiste→ part
"êramo"	3		bebêramo → beb	"amo"	2		cantamo → cant
"eremo"	3		beberemo → beb	"ara"	2		cantara→ cant
"eriam"	3		beberiam→ beb	"ará"	2		cantará→ cant
"eríei"	3		beberíei→ beb	"are"	2		cantare→ cant
"êssei"	3		bebêssei→ beb	"ava"	2		cantava→ cant
"essem"	3		bebessem→ beb	"emo"	2		cantemo → cant
"íramo"	3		partiríamo → part	"era"	3		bebera→ beb
"iremo"	3		partiremo → part	"erá"	3		beberá→ beb
"iriam"	3		partiriam→ part	"ere"	3		bebere→ beb
"iríei"	3		partiríei→ part	"iam"	3		bebiam→ beb
"íssei"	3		partíssei→ part	"íei"	3		bebíei→ beb
"issem"	3		partissem→ part	"imo"	3		partimo → part
"ando"	2		cantando→ cant	"ira"	3		partira→ part
"endo"	3		bebendo→ beb	"irá"	3		partirá→ part
"indo"	3		partindo→ part	"ire"	3		partire→ part

"ondo"	3	propondo→ prop	"omo"	3	compomo → comp
"aram"	2	cantaram→ cant	"ai"	2	cantai→ cant
"arde"	2	cantarde → cant	"am"	2	cantam→ cant
"arei"	2	cantarei→ cant	"ear"	4	barbear→barb
"arem"	2	cantarem→ cant	"ar"	2	cantar→ cant
"aria"	2	cantaria→ cant	"uei"	3	cheguei → cheg
"armo"	2	cantarmo → cant	"ei"	3	cantei→ cant
"asse"	2	cantasse → cant	"em"	2	cantem→ cant
"aste"	2	cantaste→ cant	"er"	2	beber→ beb
"avam"	2	cantavam→ cant	"eu"	3	bebeu→ beb
"ávei"	2	cantávei→ cant	"ia"	3	bebia→ beb
"eram"	3	beberam→ beb	"ir"	3	partir→ part
"erde"	3	beberde→ beb	"iu"	3	partiu→ part
"erei"	3	beberei→ beb	"ou",	3	chegou→ cheg
"êrei"	3	bebêrei→ beb	"i"	3	Bebi→ beb
"erem"	3	beberem→ beb			

Vowel Removal Rules			
Sufixo a Remover	**Tamanho Mínimo Stem**	**Substituição**	**Exemplo**
"a"	3		menina → menin
“e”	3		grande → grand
"o"	3		menino → menin