3
Etapas da Metodologia de Mineração de Textos

Neste capítulo são analisadas e discutidas as etapas da metodologia para Mineração de Textos que está sob análise, conforme inicialmente proposto por Aranha em [29]. Em seu trabalho, Aranha sugere o que seria um modelo completo para aquisição de conhecimento a partir de textos, porém com aprofundamento na sub-etapa de Pré-processamento. O objetivo deste capítulo é apresentar de forma clara e detalhada todos os passos e técnicas consideradas até o presente momento como "estado da arte", contemplando **todas** as etapas da metodologia em estudo, conforme ilustrado na Figura 3[2].

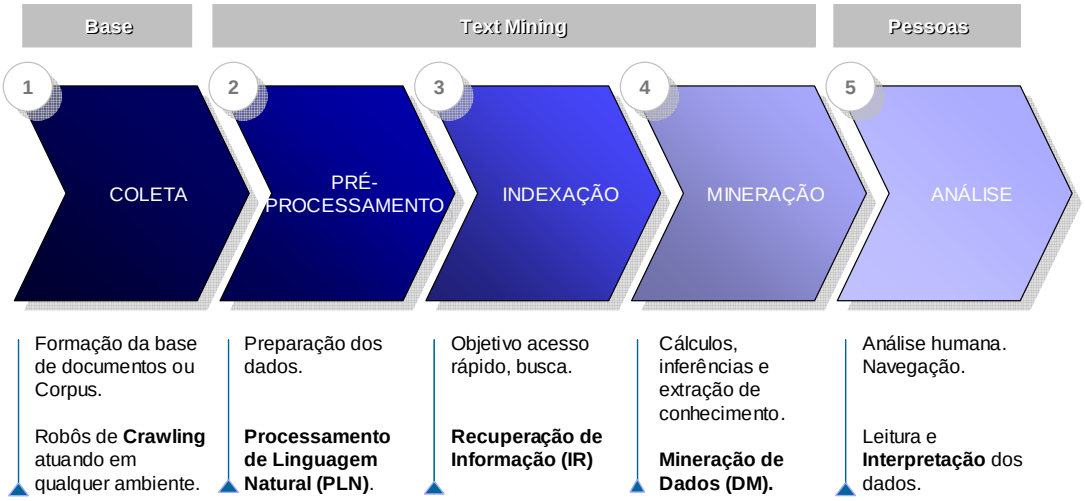

Figura 3 – Diagrama que ilustra a metodologia de Mineração de Textos com o "encadeamento" de técnicas proposta por Aranha.

Coleta é a etapa inicial e tem como objetivo formar uma base de dados textual, conhecida na literatura como C*orpus* ou *Corpora*. Pode se dar de várias maneiras, porém todas necessitam de grande esforço, a fim de se conseguir material de qualidade e que sirva de matéria-prima para a aquisição de conhecimento.

[2] A Figura 3 não sofreu nenhuma modificação em relação àquela apresentada inicialmente por Aranha em sua Tese de Doutorado.

Pré-processamento é a etapa executada imediatamente após a Coleta e tem como objetivo prover alguma formatação e representação da massa textual. É bastante onerosa, com a aplicação de diversos algoritmos que consomem boa parte do tempo do processo de extração de conhecimento.

Indexação é o processo que organiza todos os termos adquiridos a partir de fontes de dados, facilitando o seu acesso e recuperação. Uma boa estrutura de índices garante rapidez e agilidade ao processo, tal como funciona o índice de um livro.

Após terem sido obtidas uma estrutura para os dados e uma forma de prover rápido acesso, a etapa de **Mineração** propriamente dita é responsável pelo desenvolvimento de **cálculos**, inferências e algoritmos e que tem como objetivo a extração de conhecimento, descoberta de padrões e comportamentos que possam surpreender.

Finalmente, a **Análise** é a última etapa é deve ser executada por pessoas que, normalmente, estão interessadas no conhecimento extraído e que devem tomar algum tipo de decisão apoiada no processo de Mineração de Texto.

A seguir, são mencionados detalhes específicos de cada etapa, com a menção de algoritmos, implementações, problemas e soluções propostas na literatura.

3.1. Coleta

Entende-se por Coleta o processo de busca e recuperação de dados e este tem como finalidade formar a base textual da qual se pretende extrair algum tipo de conhecimento. Coletar dados é uma tarefa bastante custosa aonde existem diversos desafios, a começar pela descoberta da localização das fontes de dados.

Basicamente, são três os principais ambientes de localização das fontes: pastas de arquivos encontradas no disco rígido de usuários, tabelas de diversos bancos de dados e a Internet. As particularidades destes três ambientes serão relatadas a seguir.

As pastas de arquivos são talvez a forma mais natural de se armazenar documentos na forma digital. Coletar documentos no disco rígido de um computador é algo que exige bastante cautela, pois é necessário fazer a distinção

entre arquivos textuais produzidos por pessoas e arquivos binários e de configuração, normalmente interpretados apenas pela máquina. Algumas iniciativas podem ajudar no gerenciamento de documentos eletrônicos, como é o caso dos sistemas de GED (gerenciamento eletrônico de documentos) para grandes empresas e dos sistemas de busca local como o Google Desktop, Yahoo! Desktop e Ask Jeeves Desktop. Alguns sistemas operacionais também incluíram esta funcionalidade em suas novas versões, como é o caso do Windows Vista™, desenvolvido e comercializado pela Microsoft.

A obtenção de massa textual a partir de tabelas de banco de dados se dá, principalmente, através do conteúdo de colunas do tipo *string*, que nada mais são do que campos de armazenamento de texto livre, sem nenhuma restrição a não ser a quantidade máxima de caracteres suportada por registro. Como os dados podem estar espalhados por diversas tabelas de diversos bancos de dados, a construção de um *Data Warehouse* [2][3] é vista como um ótimo ponto de partida para a obtenção de dados, uma vez que a teoria de DW prega a integração de diversas fontes. Ainda neste tipo de sistema, supõe-se que seja realizada uma etapa de pré-processamento, aonde um dos objetivos é prover uma "limpeza dos dados", garantindo qualidade no que se está disponibilizando.

O terceiro ambiente de localização de fontes de dados é a Internet. Neste, a heterogeneidade é o desafio predominante, aonde é encontrada uma infinidade de tipos de página, como notícias de revistas, *bloggers*, anúncios, documentos, artigos técnicos e planilhas. Para a realização da coleta neste ambiente, é comum a utilização de ferramentas de apoio, como Motores de Busca Baseados em Robô (*Robotic Internet Search Engines*) e Diretórios de Assunto (*Subject Directo*ries) [30].

Crawler ou *Webcrawler* é o nome dado aos robôs especializados em navegar na Internet, de forma autônoma e exploratória, com o objetivo de realizar a coleta automática de documentos. Para tanto, é necessário que as páginas HTML sejam interpretadas de forma correta, com a identificação de *hiperlinks*, seguido de visitação, conforme o ser humano realiza. Outra função importante é saber gerenciar bem seu caminho de percurso, que tem a forma de um grafo, de modo a impedir que o robô visite várias vezes a mesma página ou entre em ciclos eternos.

Há uma forma alternativa de utilização de *crawlers*, que faz uma coleta segmentada, fazendo a varredura de forma orientada. Os robôs que se utilizam

desta forma variante são chamados de *crawlers* focados. Um *crawler* deste tipo é altamente efetivo na construção de coleções de documentos de qualidade sobre tópicos específicos e oriundos da *web*, usando simples computadores "caseiros" [31]. São considerados mais "inteligentes" que os *crawlers* normais, pois fazem uso de algoritmos que identificam documentos similares, agilizando a busca e dispensando o uso de grandes recursos de *hardware*.

Finalmente, após a apresentação dos três ambientes, é necessário destacar que em ambientes de pesquisa e desenvolvimento de Mineração de Textos, é comum a utilização de *corpus* conhecidos, previamente coletados e preparados, com a diversidade suficiente para que sejam empregados em trabalhos de pesquisa e de natureza científica. A Tabela 3 apresenta um resumo dos principais *corpus* utilizado como *benchmark* pela comunidade de pesquisadores em geral.

Tabela 3 – Resumo das principais coleções de texto usadas pela comunidade científica

Corpus	**Definição**
Reuters 21578 Corpus	Coletânea de 21.578 notícias publicadas pela agência de notícias *Reuters* no ano de 1987. Desenvolvida em 1996 por Lewis e disponível no formato SGML[3].
Reuters Corpus RCV1	*Corpus* também desenvolvido pela agência *Reuters*, com 880 mil arquivos em inglês, contendo notícias publicadas entre 20/08/1996 e 19/08/1997. Atualmente é mantido pela agência americana NIST[4] e disponível no formato XML.
Movie Review Data Set	Coleção de críticas pessoais sobre filmes coletadas do IMDb[5]. Ao todo são duas mil avaliações, divididas igualmente em dois grupos: o de comentários "favoráveis", ou seja, o autor assistiu e recomenda o filme; e o de comentários "desfavoráveis", com avaliações negativas e suas respectivas justificativas.
The Brown Corpus	*The Brown Corpus of Standard American English* foi um dos primeiros *corpus* criados para o propósito de processamento de textos. É Constituído de textos de diversos gêneros, com aproximadamente um milhão de palavras.
Penn Tree Bank	Coleção de notícias extraídas do *Wall Street Journal*, com etiquetas morfossintáticas manualmente identificadas.
CETENFolha	Coleção compilada pelo NILC[6], com cerca de 24 milhões de palavras em português, criada com base nos textos publicados no Jornal Folha de São Paulo.

[3] SGML é uma metalinguagem para definição de linguagens de marcação para documentos.
[4] *National Institute of Science and Technology* (http://www.nist.gov)
[5] *The Internet Movie Database* (http://www.imdb.com)
[6] Núcleo Interinstitucional de Lingüística Computacional (http://www.nilc.icmsc.sc.usp.br)

3.2. Pré-processamento

Pré-processamento é a etapa realizada imediatamente após a Coleta, com o objetivo de se obter alguma estrutura para a massa textual. Pré-processar textos é, por muitas vezes, o processo mais oneroso da metodologia de MT, uma vez que não existe uma única técnica que possa ser aplicada para a obtenção de uma representação satisfatória em todos os domínios. Assim sendo, para se chegar à representação adequada, pode ser necessária a realização de muitos experimentos empíricos [32].

O principal objetivo de se pré-processar textos é aumentar a qualidade inicial dos dados, aonde diversas técnicas podem ser aplicadas e até mesmo combinadas, num processo similar ao mecanismo de *pipeline*, aonde a saída de determinado programa é entrada para outro, similar a uma estrutura de "dutos interconectados".

Normalmente, o produto final do pré-processamento é uma estrutura do tipo atributo-valor, conforme verificado na Tabela 4. As linhas fazem alusão a cada um dos documentos da coleção, enquanto que as colunas fazem referência aos atributos, presentes ou não, em cada um dos documentos. A intersecção entre atributos e documentos é marcada pelo peso dado a determinado atributo em determinado documento (por exemplo, pode-se utilizar a freqüência de aparição do atributo no documento). Esta estrutura precisa ser significante, representativa e que reflita fielmente a diversidade original dos dados. De posse desta estrutura, é possível a execução da etapa de Mineração, precedida ou não da de Indexação.

Tabela 4 – Representação atributo-valor obtida à partir da etapa de Pré-processamento

	Atrib1	**...**	**AtribN**
Doc1	V11	...	V1N
...	....	....	....
Docm	Vm1	...	Vmn

3.2.1.
Tokenization (Atomização)

Tokenization é o primeiro passo da etapa de pré-processamento e sua execução tem como finalidade extrair **unidades mínimas de texto** a partir de um texto livre. Cada unidade é chamada de ***token*** e que, na grande maioria das vezes, corresponde a uma palavra do texto, podendo também estar relacionado a mais de uma palavra, símbolo ou caractere de pontuação. É um termo bastante utilizado ao longo desta dissertação, mesmo nos momentos em que "palavra" pode parecer ter o mesmo sentido. De fato, muitas vezes um *token* representa uma e apenas uma palavra no texto, conforme já mencionado. Entretanto, preferiu-se manter o termo técnico verificado na literatura. Como exemplo, a frase "Zico foi o maior jogador da história!" possui oito *tokens*, conforme mostra o exemplo abaixo:

"Zico foi o maior jogador da história!"
[Zico] [foi] [o] [maior] [jogador] [da] [história] [!]

O caractere que sempre é descartado na geração de *tokens* é o "espaço", como pode ser observado na transformação acima. Existem diversas estratégias para a obtenção dos *tokens* de um texto. A "quebra" de um texto em seus delimitadores é uma estratégia simples e que apresenta bons resultados quando se possui uma grande massa de dados. Por exemplo, pode-se quebrar um texto nos seguintes delimitadores, além do espaço: () <>!-?.;'- "|.

Entretanto, a tarefa de identificação de *tokens*, que é relativamente simples para o ser humano, pode ser bastante complexa de ser executada por um computador. Este fato é atribuído ao grande número de papéis que os delimitadores podem assumir. Por exemplo, o "ponto" pode ser usado para marcar o fim de uma sentença, mas também é usado em abreviações e números. Outro exemplo é o travessão, que pode indicar o início de uma citação no texto ou, quando entre dígitos, indicar um número de telefone (ex., (21) 2235-7553) ou uma operação de subtração, também entre números.

É possível acrescentar mais informações à *Tokenization*, com a adição de dicionários e regras de formação. Em [33] é apresentado um subsistema que faz uso de tais artefatos, unindo funções e camadas, de forma similar a uma "linha de montagem", como ilustrado na Figura 4.

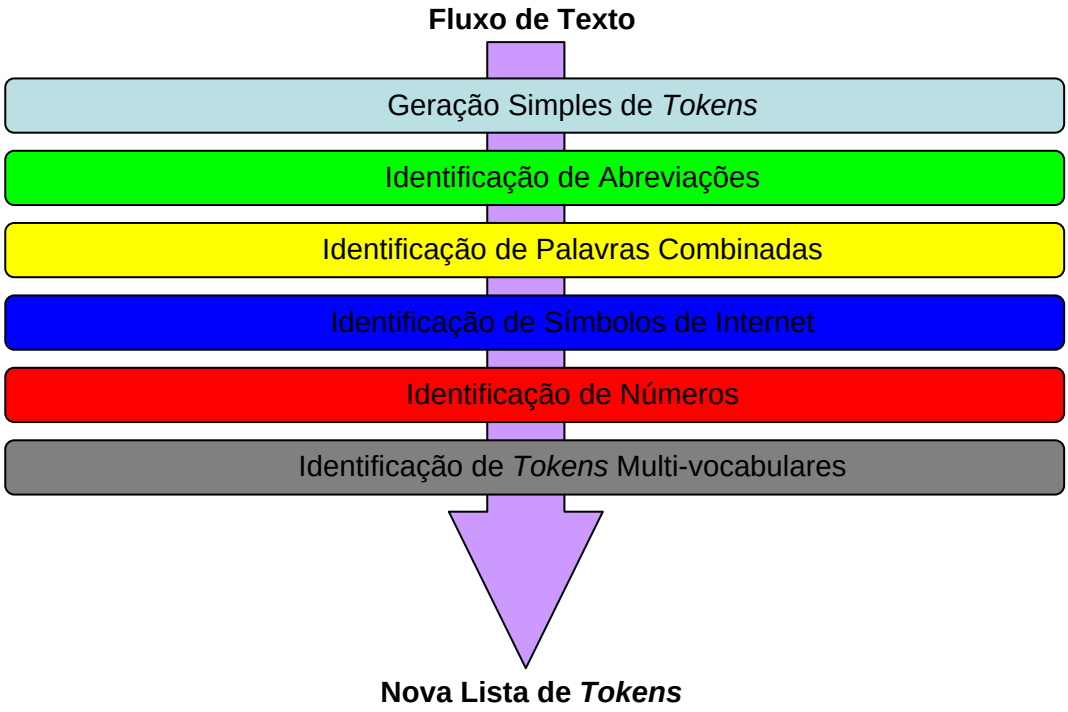

Figura 4 – "Linha de montagem" de um procedimento de *Tokenization*.

A "linha de montagem" começa com a apresentação de um fluxo de texto ao subsistema. Entende-se por fluxo de texto toda e qualquer seqüência de caracteres. Em **Geração Simples,** são identificados os *tokens* preliminares com base em uma lista de delimitadores e o espaço em branco. Em seguida, é feita a **Identificação de Abreviações**, com a ajuda de dicionários pré-estabelecidos. Existem diferentes tipos de abreviação, com uma ou mais palavras ou com ou sem utilização de pontuação. Em **Identificação de Palavras Combinadas**, palavras que foram separadas por determinados caracteres como "&" e "-" são unidas, formando um único *token*. Estas palavras podem ou não estar separadas por espaços, como em "AT&T" e "AT & T". Em **Identificação de Símbolos de Internet,** é observada a existência de endereços de e-mail, endereços de *sites* (URLs) e endereços IP. Em **Identificação de Números**, são identificadas toda e qualquer forma de apresentação de número, incluindo também medidas e valores. Por último, a **Identificação de *Tokens* Multi-vocabulares** observa a aparição de palavras que precisam estar unidas em único *token*, com o objetivo de manter o sentido original encontrado no texto. A Tabela 5 apresenta, passo a passo, a exemplificação do resultado da execução desse subsistema.

Tabela 5 – Exemplificação do resultado da execução de um subsistema de *Tokenization* que baseia-se em dicionários pré-estabelecidos e regras de formação.

Descrição	Tokens
Primeiramente, o fluxo de texto é apresentando ao subsistema de T*okenization.*	A Casa & Lar se mudou. Agora, atendemos na Av. Dom Casmurro, nº 200. Você também pode comprar pelo nosso endereço na Internet: http://www.casaelar.com.br. Comprando por lá, você tem até R$ 100,00 de desconto em tubos e caixas d'água!
Em seguida são gerados os *tokens* preliminares, com a observação de espaços em branco e delimitadores.	[A][Casa][&][Lar][se][mudou][.][Agora][,] [atendemos] [na] [Av][.] [Dom] [Casmurro][,] [n][º] [200][.] [Você] [também][pode] [comprar] [pelo] [nosso] [endereço] [na] [Internet][:] [http][:][/][/][www][.][casaelar][.][com][.][br][.] [Comprando] [por] [lá][,] [você] [tem] [até] [R$] [100][,][00] [de] [desconto][em][tubos][e][caixas][d][‘][água][!]
Logo após, são identificadas as abreviações (em verde), as palavras combinadas (em amarelo) e os símbolos de Internet (em azul).	[A][Casa&Lar][se][mudou][.][Agora][,][atendemos] [na] [Av.] [Dom] [Casmurro][,] [nº] [200][.] [Você] [também][pode] [comprar] [pelo][nosso] [endereço] [na][Internet][:] [http://www.casaelar.com.br][.][Comprando][por] [lá][,] [você] [tem] [até] [R$] [100][,][00] [de] [desconto][em][tubos][e][caixas][d][‘][água][!]
Por último são identificados os números e os *tokens* multi-vocabulares. Esta é a lista final que será retornada pelo subsistema *Tokenization* de acordo com o exemplo de entrada.	[A][Casa&Lar][se][mudou][.][Agora][,] [atendemos] [na] [Av. Dom Casmurro][,] [nº 200][.] [Você] [também][pode] [comprar] [pelo] [nosso] [endereço] [na] [Internet][:] [http://www.casaelar.com.br] [.] [Comprando] [por] [lá][,] [você] [tem] [até] [R$ 100,00] [de] [desconto][em][tubos][e][caixas d‘água][!]

3.2.2.
Correção Ortográfica

Erros ortográficos são comuns quando se trabalha com grandes massas de dados, especialmente se geradas a partir de digitação manual por seres humanos. O trabalho de identificação automática destes erros vem sendo objeto de estudo por cientistas no mundo inteiro.

Em 1965, o cientista russo Vladimir Levenshtein apresentou um algoritmo que define a distância de edição entre dois *strings* (seqüência de caracteres). Este algoritmo ficou conhecido por **Distância de Levenshtein** ou, simplesmente, **Distância de Edição** e baseia-se no número mínimo de operações necessárias para transformar um *string* em outro. É a partir desta idéia que muitos corretores automáticos de ortografia se baseiam para detectar um erro e sugerir sua possível correção, através dos candidatos que possuem as menores distâncias.

As operações que transformam uma seqüência de caracteres em outra pode ser de: **inserção** (inserção de um novo caractere no *string* "destino"), **eliminação** (eliminação de um caractere no *string* "origem") e **substituição** (substitui um caractere do *string* "origem", com o objetivo de transformar no *string* "destino"). O Exemplo abaixo exibe os passos necessários para transformar "casas" em "massa", definindo a distância de edição em 3 (três).

1.	casas → masas	(substituição de 'c' por 'm')
2.	masas→ mass	(eliminação de 'a')
3.	mass → massa	(inserção de 'a')

Outra abordagem que apresenta bons resultados é a técnica de indexação por n-gramas de letras. Uma n-grama de letras é uma seqüência de *n* letras de uma dada palavra. Para exemplificar, a palavra "maleta" pode ser dividida em quatro 3-gramas, também conhecido como trigramas: "mal", "ale", "let" e "eta". A idéia é que os erros ortográficos mais comuns só afetam poucos constituintes de n-grama, então, podemos buscar pela palavra correta através daqueles que compartilham a maior parte dos n-gramas com a palavra errada [29].

A idéia de indexar as n-gramas de uma palavra segue a mesma da etapa de Indexação, a qual será explicada no decorrer desta dissertação. O objetivo é

manter uma lista de n-gramas “apontando” para as palavras que o contém. Quando a palavra é procurada, os n-gramas são processados e procurados no índice. A palavra que apresentar o maior número de n-gramas associados será a de maior relevância, indicando um possível candidato para correção [29].

Existem diversas abordagens para o problema de correção ortográfica. Para um estudo mais detalhado sobre as diversas técnicas, vide [34].

3.2.3. Redução do Léxico

Um dos grandes desafios da Mineração de Textos, senão o maior, e o que a faz ser muito mais complexa que a Mineração de Dados, é o elevado número de dimensões existentes, se considerarmos que cada *token* em um texto é mapeado para uma dimensão. Logicamente, um texto simples possui algumas centenas de palavras, ao passo que uma tabela com dados estruturados, com um pouco mais de algumas dezenas de colunas já é considerada uma estrutura de grande porte.

Existem inúmeras soluções para a redução de um léxico, com o objetivo de se obter apenas os *tokens* que realmente são importantes e traduzem a essência de um texto. A seguir, são explicadas as principais abordagens encontradas na literatura.

3.2.3.1. Seleção de Características

Seleção de Características é o procedimento que define, através da aplicação de algoritmos e métricas, o subconjunto mais discriminante de um conjunto inicial de características, reduzindo-se então o espaço inicial.

Este procedimento é bastante útil, em particular em Mineração de Textos, onde o número de dimensões de um léxico (cada *token* representa uma dimensão) é demasiadamente alto, conforme já mencionado. Como benefícios dessa redução está o aumento da performance das Tarefas de Mineração e a diminuição do tempo de execução dos algoritmos correspondentes.

Uma das preocupações que se deve ter quando da aplicação de algoritmos de Seleção de Características é a de que a Redução do Léxico não afete de maneira drástica o sistema, com perda mínima da informação.

A seguir, são apresentadas as principais métricas baseadas puramente em estatística e que definem o quão importante determinado *token* é para o léxico.

- **Freqüência de Documentos:** Esta métrica utiliza um critério bastante simples e intuitivo que é computar o número de documentos no qual determinado termo aparece e remover aqueles cuja freqüência está abaixo de um limiar predefinido. A suposição é que termos raros não são significativos para a discriminação entre classes de texto ou que, pelo menos, esta eliminação não impacte no desempenho global. Esta é a técnica mais simples de redução, visto que a complexidade se mantém constante em relação ao número de documentos.

- **Ganho de Informação:** Define a importância de determinado termo para a discriminação entre classes de documentos previamente conhecidos, verificando o quanto cada termo está correlacionado com cada classe. Esta métrica é baseada em outra, definida no âmbito da Teoria da Informação, chamada de Entropia. A Entropia mede a quantidade de informação de um atributo, caracterizando a impureza de um conjunto de exemplos. Dado um conjunto *S* de exemplos, uma categorização em *C* categorias *c1,c2,...,cn,* e a probabilidade de determinado exemplo pertencer à determinada categoria, probabilidade esta denotada por p_i, a Entropia *E(S)* é definida como:

$$E(S) = \sum_{i=1}^{n} - p_i \lg(p_i)$$

O ganho de informação para um *token T* de um conjunto de documentos *D* permite verificar a diminuição da entropia quando utilizamos *T* como parte do léxico e, conseqüentemente, parte influente na discriminação entre classes [35]. Seja P(T) o conjunto dos valores que *T* pode assumir; seja x um elemento deste conjunto e seja S_x o subconjunto de S formado pelos dados em que T=x; a

entropia que se obtém ao particionar *S* em função de *T* é dada por duas equações:

$$E(T) = \sum_{x \in P(T)} \frac{|S_x|}{|S|} Entropia(S_x)$$

$$Ganho(S,T) = Entropia(S) - E(T)$$

- **Informação Mútua:** Informação Mútua é um critério normalmente usado em modelagem estatística da linguagem em associações de palavras correlatas [36]. Se for considerada uma tabela de contingências de um termo *t* e uma categoria *c*, *A* é o número de vezes que *t* e *c* co-ocorrem, *B* é o número de vezes que t ocorre sem c, C é o número de vezes que c ocorrem sem t, e N é o número total de documentos, então o critério de informação mútua entre *t* e *c* é definido como:

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

E é estimado usando:

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

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:

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

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

- **Estatística χ^2**: Esta métrica mede a falta de independência entre *t* e *c* e pode ser comparada à distribuição χ^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*, 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:

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

A estatística χ^2 tem naturalmente um valor de zero se *t* e *c* são independentes. Calcula-se para cada categoria a estatística χ^2 entre cada termo no conjunto de documentos e àquela categoria, e então são combinadas as pontuações específicas da categoria pra cada termo através de:

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} P_r(c_i)\chi^2(t,c_i)$$

3.2.3.2. Remoção de *Stopwords*

Em um documento, existem muitos *tokens* que não possuem nenhum valor semântico, sendo úteis apenas para o entendimento e compreensão geral do texto. Estes *tokens* são palavras classificadas como *stopwords* e correspondem ao que é chamado de *stoplist* de um sistema de Mineração de Textos.

Uma lista de *stopwords* é constituída pelas palavras de maior aparição em uma massa textual e, normalmente, correspondem aos **artigos**, **preposições, pontuação**, **conjunções** e **pronomes** de uma língua. A identificação e remoção desta classe de palavras reduz de forma considerável o tamanho final do léxico, tendo como conseqüência benéfica o aumento de desempenho do sistema como um todo.

A *stoplist* pode ser definida manualmente, por um especialista no domínio do assunto, ou de forma automática, através da freqüência de aparição das palavras no léxico. Um percentual *K* das palavras de maior aparição define a lista de remoções. A Tabela 6 ilustra uma pequena *stoplist* definida manualmente e a identificação e descarte de *tokens*. Já a Tabela 7 apresenta uma *stoplist* obtida automaticamente a partir de um sistema pronto, com a lista das 100 palavras de maior aparição.

Tabela 6 – Identificação e Remoção de *Stopwords* (os *tokens* descartados estão tachados)

Stoplist				**Texto**
A	seu	De	Um	~~[A]~~[Casa&Lar]~~[se]~~[mudou]~~[.]~~[Agora]~~[,]~~
O	deve	do	uma	[atendemos] ~~[na]~~ [Av. Dom Casmurro]~~[,]~~ [nº
pelo	sua	da	sobre	200]~~[.]~~ [Você] ~~[também]~~[pode]—[comprar]
por	nosso	também	são	~~[pelo]~~ [nosso] [endereço] ~~[na]~~ [Internet]~~[:]~~
em	nossa	se	cada	[http://www.casaelar.com.br] ~~[.]~~
na	.	comigo	isso	[Comprando] ~~[por]~~ ~~[lá]~~~~[,]~~ [você] [tem] [até]
no	!	pela		[R$ 100,00] ~~[de]~~
como	;	?		[desconto]~~[em]~~[tubos]~~[e]~~[caixas d'água]~~[!]~~
lá	,	só		

Tabela 7 – *Stoplist* obtida automaticamente a partir de um sistema de Mineração de Texto pronto.

A	Como	e	Foram	Mil	ou	que	só
à	contra	é	governo	milhões	país	quem	sobre
ainda	da	ela	grande	muito	para	r	sua
ano	das	ele	há	mundo	paulo	rio	também
anos	de	em	hoje	na	pela	são	tem
ao	depois	entre	isso	não	pelo	se	ter
aos	deve	era	já	nas	pessoas	segundo	todos
apenas	dia	está	local	no	pode	sem	três
as	disse	estado	maior	nos	por	ser	um
às	diz	estão	mais	o	porque	será	uma
até	do	eu	mas	ontem	presidente	seu	us
brasil	dois	foi	mesmo	os	quando	seus	vai
com	dos	folha			quero		

3.2.3.3. Normalização

Normalização é a técnica de Redução de Léxico que se baseia no agrupamento de *tokens* que compartilham de um mesmo padrão. Existem diversas abordagens de agrupamento, que vão desde a identificação morfológica do *token* até o reconhecimento de sinônimos e conceitos similares.

Em geral, a aplicação de técnicas de Normalização introduz uma melhora significativa nos sistemas de Mineração de Texto. Esta melhora varia de acordo com o escopo, o tamanho da massa textual e o que se pretende obter como saída do sistema (normalmente definido pela natureza da Tarefa de Mineração, capítulo 4). Sistemas de Classificação são os que mais se beneficiam deste tipo de redução do léxico, uma vez que utilizam a estatística como base teórica central.

De acordo com a forma de agrupamento das realizações das palavras, os processos de normalização podem ser de vários tipos. Os principais são:

- ***Stemming****:* O processo de *stemming* concentra-se na redução de cada palavra do léxico, até que seja obtida sua respectiva raiz. Desta maneira, tem-se como principal benefício a eliminação de sufixos que indicam variação na forma da palavra, como plural e tempos verbais. Os algoritmos em geral não se preocupam com o uso do contexto no qual a palavra se encontra, e esta abordagem parece não ajudar muito. Casos em que o contexto ajuda no processo de *stemming* não são freqüentes, e a maioria das palavras pode ser considerada como apresentando um significado único [37]. A seguir, uma lista dos principais métodos encontrados na literatura:

  - **Método do *Stemmer* S**: Método simples que foca apenas em algumas poucas terminações de palavras do inglês. Os principais sufixos a serem removidos são: *ies*, *es* e *s* (com exceções). Embora este método não introduza muito impacto nos léxicos, é bastante utilizado por seu caráter conservador e que raramente surpreende o usuário.

- **Método de *Porter*:** Este método se concentra na identificação das diferentes formas e inflexões referentes à mesma palavra e sua substituição por um radical comum. Por exemplo, as seguintes palavras compartilham do mesmo radical: CORRER, CORRIDA, CORRIDO e CORRIDÃO. Neste caso, a aplicação do Método de *Porter* mapearia todas estas palavras para seu radical comum "CORR". É importante ressaltar que este método remove 60 sufixos diferentes em uma abordagem multifásica.

- **Método de *Lovins*:** Método de passo único. É Sensível ao contexto e abrange uma gama maior de sufixo (250 ao todo). Baseia-se numa lista de regras, chamada de regras de Lovins, e que, num passo único, faz a remoção de, no máximo, um único sufixo por palavra. Apesar de não incluir vários sufixos em sua abordagem, é o mais agressivo dos algoritmos apresentados.

- ***Lemmatization**:* Substitui as diversas formas de representação da palavra pela forma primitiva. As formas "livro", "livros" e "livraria" apontam todas para a palavra "livro". Este método tem a vantagem de manter uma estrutura que preserva o sentido das palavras, ao contrário dos métodos de *stemming.*

- **Identificação de Sinônimos, Hierarquias e Relacionamentos Associativos:** A identificação destes em uma coleção textual auxilia na redução do léxico, de modo a se concentrar basicamente no significado das palavras. Normalmente, é realizado com base em um dicionário chamado de *thesaurus.* Um *thesaurus* é definido como um repositório de mapeamentos entre termos variantes – sinônimos, abreviações, acrônimos e ortografias alternativas – para um termo preferido único para cada conceito.

3.2.3.4.
Expressões Multi-vocabulares

O processo de detecção e associação de uma seqüência de *tokens* em um único *token* é chamado de criação de Expressões Multi-vocabulares**,** e pode ser bastante útil em Mineração de Textos, tanto para a redução do léxico quanto para o resultado final das tarefas de Mineração.

Na literatura lingüística temos algumas classes de expressões multi-vocabulares, que são: colocações, expressões e expressões idiomáticas. Como apresentado em [37][38], a diferença entre estas classes está relacionada ao dinamismo da linguagem. Dessa forma, colocações são as combinações mais dinâmicas de palavras, conforme elas vão sofrendo um processo de fossilização, se transformam em expressões idiomáticas. A partir desse ponto de vista, são os tipos mais fossilizados de combinações, de forma que as palavras combinadas se "desgarram" do significado das palavras individuais da combinação. Isso quer dizer que a diferença entre esses termos está relacionada com o uso. Exemplos dessa dinâmica são "Vos Mercê", "Você" e "cê", ou mesmo "em boa hora", "embora" e "bora". A freqüência de uso durante uma considerável quantidade de tempo seria responsável pela mudança no *status* de palavra para expressões multi-vocabulares. Por esse motivo, as abordagens estatísticas têm se destacado bastante na solução desses problemas [29].

3.2.4.
Identificação do Início e Fim de Sentenças

A identificação do início e do fim de sentenças em um texto é uma das tarefas mais difíceis da etapa de Pré-processamento e, normalmente, é realizada por último, visto que muitos dos problemas menores de processamento de linguagem natural já foram resolvidos. Para exemplificar a dificuldade inerente, tem se o exemplo da presença do "ponto" no texto, aonde este pode estar indicando o término de uma sentença, presença de uma abreviação ou, em casos mais raros, ambos. Veja os exemplos abaixo:

> “É necessário ter um bom relacionamento com seu orientador de Ph.D.”
>
> “É necessário ter um bom relacionamento com seu orientador de Ph.D. Tenha certeza de que ele será fundamental na definição das pesquisas.”
>
> “É necessário ter um bom relacionamento com o orientador de Ph.D. Fernando Pessoa.”

No primeiro exemplo, o segundo “ponto” da abreviação “Ph.D.” acabou incorporando também a função de delimitador de sentença. Neste caso, não há grandes problemas, pois se trata de uma sentença não seguida por outra. No segundo exemplo, o “ponto” da abreviação assume novamente as duas funções, mas é seguido por outra sentença. Isto dificulta um pouco a identificação dos limites, mas pode ser resolvido pela observação da primeira palavra após o “ponto” – “Tenha” - que apresenta sua letra inicial na forma maiúscula. Entretanto, no terceiro exemplo, esta observação leva a uma decisão errônea sobre os limites, uma vez que a palavra após o segundo “ponto” da abreviação “Ph.D.” é um substantivo próprio e que, por via de regra, deve ter sua letra inicial maiúscula.

A decisão se um determinado caractere é ou não um delimitador de sentença pode ser encarado como um problema de classificação (seção 4.1). Desta forma, é possível que a partir de um conjunto previamente etiquetado de textos e sentenças, possamos “treinar” um classificador que “aprenda” os padrões que permitam um ser humano decidir quando começa e termina uma sentença. Estudos preliminares apontam para uma taxa de acerto de 98% com o uso desta abordagem [39].

Outra abordagem é o uso de heurísticas baseadas em regras, que são ajustadas de acordo com o contexto e com a língua. Apesar de não atingirem a mesma performance dos classificadores, o uso desta abordagem introduz uma ótima alternativa quando não se dispõe de uma base de dados etiquetada para treino. A Figura 5 define um exemplo deste tipo de heurística, conforme também encontrado em [39].

Entrada: Texto com pontuações

Saída: Mesmo texto com **D**elimitadores de **F**im de **S**entença (DFS) claramente identificados.

Estratégia:

1. Transformar todos os caracteres do texto que não fazem parte da lista de delimitadores por um caractere padrão
2. Aplicar a lista de regras abaixo para todos os delimitadores do texto e marcar os que satisfazem a lista
3. Retransformar os caracteres do passo 1.
4. Após completar o passo 3, todos os DFS estarão claramente identificados no texto

Regras:

1. Todos os caracteres entre parênteses são considerados DFS (!?)
2. **Se** " ou ' aparecer antes de um "ponto", **então** este "ponto" é um DFS
3. **Se** o caractere seguinte ao "ponto" não for um espaço em branco, **então** o "ponto" **não** é um DFS
4. **Se**)}] aparecer antes de um "ponto" **então** este é um DFS
5. **Se** o *token* o qual o "ponto" está "grudado" tiver seu caractere inicial maiúsculo **E** o *token* possui menos que **5** caracteres **E** o *token* seguinte também é iniciado por um caractere maiúsculo **então** este "ponto" não é um DFS.
6. **Se** o *token* o qual o "ponto" está "grudado" possuir também outros "pontos" **então** nenhum deles é um DFS
7. **Se** o *token* o qual o "ponto" está "grudado" começa com caractere minúsculo **E** o próximo *token* é um espaço em branco precedido de um *token* que tem seu primeiro caractere em maiúsculo **então** este "ponto" é um DFS
8. **Se** o *token* o qual o "ponto" está "grudado" possuir menos que **2** caracteres então o "ponto" não é um DFS
9. **Se** o *token* o qual o "ponto" está "grudado" for seguido de um espaço em branco e que, por sua vez, é seguido por um outro *token* que é iniciado por um destes caracteres $({["' **então** este "ponto" é um DFS.
10. **Caso** exista algum "**ponto**" que não se enquadre nas regras acima **então** este não é um DFS.

Figura 5 – Exemplo de uma heurística de detecção de início e fim de sentenças

3.2.5.
Etiquetagem POS

Etiquetagem POS (do inglês, *part of speech*) consiste da identificação sintática de cada *token* extraído do *corpus*. Esta técnica é geralmente utilizada quando se pretende realizar algum tipo de abordagem lingüística mais elaborada, como a **Identificação de Entidades Nomeadas** de um texto, explicada na seção 3.2.6 desta dissertação.

Podemos resumir em oito as principais classes sintáticas da maioria das linguagens, que são: **verbos**, **nomes**, **adjetivos**, **advérbios**, **preposições**, **conjunções**, **pronomes** e **determinantes**. Qualquer *token* que não possa ser enquadrado nestas classes é classificado como sendo pontuação ou interjeição.

A identificação da classe sintática de *tokens* é encarada na literatura como um típico problema de Classificação (seção 4.1) aonde, dada uma observação, deve se optar por uma entre *k* classes distintas, aonde *k* é a classe que mais se identifica com a observação. A Estatística é a abordagem de Classificação em Mineração de Textos que, de longe, é a mais utilizada e que apresenta os melhores resultados. Entretanto, é necessário que haja um conjunto de treinamento e teste (seção 4.1.1) suficientemente grande para a correta calibragem do algoritmo. Para a Etiquetagem POS isto não chega a ser um problema, visto que a maioria dos textos compartilha das mesmas classes sintáticas para as palavras e existem diversos *corpus* anotados na literatura.

Os dois principais algoritmos de Classificação POS são o HMM (do inglês, *Hidden Markov Model*) [40][41][42] e o TBL (do inglês, *Transformation Based Learner*) [43]. Ambos os algoritmos podem ser aplicados em outros objetivos, como a utilização do HMM na identificação da estrutura macro de um texto (ex., autor, título) e do TBL para resolução de ambiguidade sintática.

3.2.6.
Identificação de Entidades Nomeadas

A Identificação de Entidades Nomeadas tem como objetivo encontrar no texto *tokens* que fazem alusão a figuras do mundo real, como personalidades, lugares e organizações. Para fazer a identificação é necessária a utilização de informações adquiridas a partir de técnicas anteriormente apresentadas, como a

etiquetagem POS e a identificação do início e fim de sentenças. O uso de dicionários especializados contendo listas de nomes próprios, lugares e organizações e a utilização de “pistas” textuais, como identificação de palavras que começam por maiúsculas, completam o “kit de ferramentas” que são utilizadas nesta etapa do pré-processamento.

Reconhecer entidades em um texto é recurso bastante valioso para MT. Entretanto, apesar de parecer um problema trivial de ser solucionado, pode ser extremamente complexo de ser automatizado. Como exemplos seguem os seguintes:

(1) João R. Carrilho Junior
(2) Luiz Inácio Lula da Silva
(3) Ronaldo Fenômeno do Real Madrid

No caso 1 temos um nome próprio não trivial porque contém uma abreviação no meio que poderia ser considerado como ponto final. No caso 2 temos um item funcional “da” que poderia separar o nome em dois distintos: “Luiz Inácio Lula” e “Silva”. Finalmente, o caso 3 contém de fato duas entidades, sendo que não há, aparentemente, como distinguí-lo do caso 2.

3.2.7. *Parsing* (Análise Sintática)

Todas as técnicas de Pré-processamento apresentadas até o momento tinham como objetivo o enriquecimento de *tokens* como se fossem unidades sem ligação. *Parsing* é a técnica de PLN que define uma estrutura aonde pode ser observada a função sintática de cada token em uma sentença (ex., sujeito, objeto, etc.), bem como sua relação com os demais.

Normalmente a estrutura definida é do tipo árvore, com os *tokens* representados como “folhas”, “nós” internos definindo agrupamento entre *tokens* e a raiz, única, definindo a sentença como um todo.

É comum encontrar na literatura o termo **Árvore de Derivação** para a estrutura montada, a qual, durante sua construção, é verificada a adequação das seqüências de palavras às regras de construção impostas pela linguagem, na

composição de frases, períodos ou orações. Dentre estas regras, pode-se citar a concordância e a regência nominal ou verbal, bem como o posicionamento de termos na frase. Um termo corresponde a um elemento de informação (palavra ou expressão), e é tratado como unidade funcional da oração, participando da estrutura como um de seus constituintes, denominados sintagmas.

A análise sintática de uma oração em português deve levar em conta os seguintes sintagmas: **termos essenciais** (sujeito e predicado), **termos integrantes** (complementos verbal e nominal) e **termos acessórios** (adjunto adverbial, adjunto adnominal e aposto). A análise do período, por sua vez, deve considerar o tipo de período (simples ou composto), sua composição (por subordinação, por coordenação) e a classificação das orações (absoluta, principal, coordenada ou subordinada). A Figura 6 ilustra uma árvore de derivação simples para a frase "José comeu o bolo". Os nós interiores da árvore representam os sintagmas (SN significa sintagma nominal e SV sintagma verbal) e os nós folhas representam as palavras.

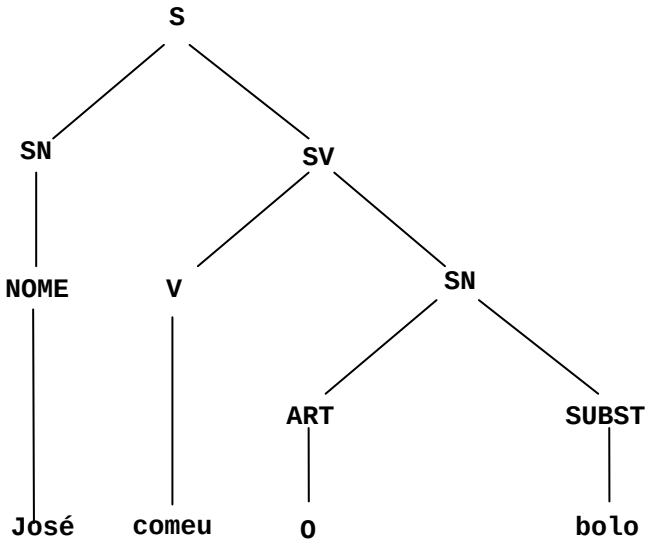

Figura 6 – Árvore de Derivação simples para a frase "José comeu o bolo".

3.3. Indexação

3.3.1. Representação de Documentos

Os sistemas de Recuperação de Informação, cedo ou tarde, esbarram no problema de representação de documentos e consultas de usuários. Um documento precisa ser codificado de uma forma que facilite sua manipulação e que permita uma correta mensuração de seus termos.

Existem diversos modelos para representação de documentos na literatura de RI. Entretanto, a grande contribuição para a Mineração de Textos é, sem dúvida, o **Modelo de Espaço Vetorial** (em inglês, *Vector Space Model*), que representa um documento utilizando uma abstração geométrica. Desta forma, documentos são representados como pontos em um espaço Euclidiano t-dimensional em que cada dimensão corresponde a um *token* do léxico. Desta forma, D_i diz respeito ao i-ésimo componente do documento D, o qual possui um peso associado.

O Modelo de Espaço Vetorial é a forma mais comum de representar documentos. A principal vantagem na representação vetorial está na natureza da maioria dos algoritmos da etapa de Mineração, os quais já estão preparados para lidar com esse tipo de codificação, como o ***Naive Bayes*** (seção 4.1.3) para o problema de Classificação e o ***K-Means*** [54] para o de *Clusterização*.

O principal problema da utilização deste modelo está na alta dimensionalidade inerente à Mineração de Textos, pois dado um *corpus* com um pouco mais de algumas centenas de documentos, o número de *tokens* facilmente ultrapassa a marca de centenas de milhares.

Na literatura de Mineração de Textos é comum se usar o termo “saco de palavras” (do inglês, *bag of words*) para este tipo de representação. A analogia é explicada devido ao próprio formato, aonde um documento é visto como um *container* de *tokens*, aonde a ordem e a ligação entre os *tokens* não tem nenhum valor para o sistema. Esta modelagem é visivelmente pobre em relação a todos os recursos que o vocabulário de uma língua pode oferecer, inviabilizando grandes técnicas de PLN. Entretanto, a codificação *bag of words* vem apresentando bons resultados na literatura, justificando a sua abordagem puramente estatística. Em

[44] é introduzido um estudo comparativos com novas abordagens para o problema de representação de documentos. A Figura 7 ilustra como fica a representação de um documento utilizando o modelo “saco de palavras”.

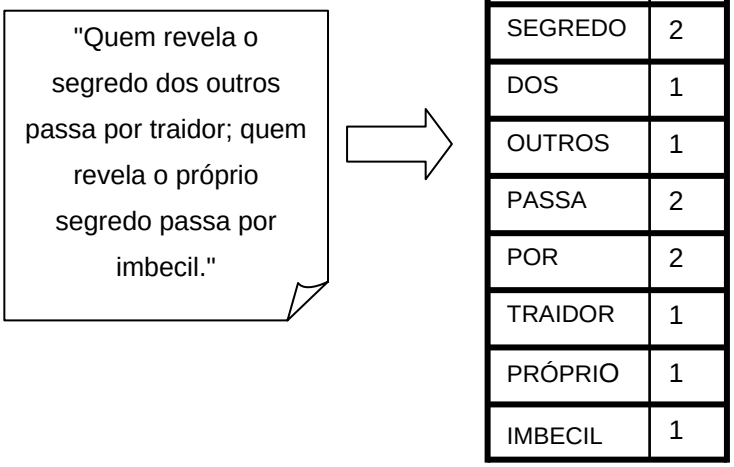

Figura 7 – Exemplificação do modelo “saco de palavras”.

3.3.2. Medidas de Similaridade entre Documentos

No processo de Recuperação de Informação é necessário que, dado um documento *D*, seja possível o cálculo de quão similar *D* é em relação aos demais documentos que compõem o *corpus*. Partindo da premissa que dois documentos são idênticos se compartilham do mesmo conjunto de *tokens*, é intuitivo que o contrário também defina dois documentos totalmente diferentes, com a conclusão de que não há similaridade entre eles.

Sendo assim, a medida de similaridade mais óbvia entre dois documentos é o **número de *tokens* em comum.** Caso se esteja usando a representação “saco de palavras” na forma binária, cada documento é mapeado em um vetor de “zeros” e “uns”, aonde “zero” assinala a ausência de determinado *token* e “um” assinala a presença. Desta forma, matematicamente a similaridade entre dois documentos é o

produto dos dois vetores, visto que o único resultado diferente de zero é quando há coincidência de *tokens*, acarretando na verificação de um *token* em comum.

Em um número elevado de dimensões é possível que seja difícil a real discriminação de quão similar é um documento do outro simplesmente usando o critério da contagem de *token*s. Ao invés disso, pode ser necessário usar um critério que leve em consideração também a freqüência na qual estes aparecem no léxico. Assim, a métrica de **contagem de tokens com bônus** é definida pelas duas equações abaixo:

$$Similaridade(D(i)) = \sum_{j=1}^{K} w(j)$$

$$w(j) = \begin{cases} 1 + 1/df(j) & se\ token(j)\ ocorre\ em\ ambos\ os\ documentos, \\ 0 & caso\ contrário. \end{cases}$$

Na primeira equação tem se um novo documento com *K tokens* sendo comparado a um documento *D(i)*. O cálculo da similaridade é computado, primeiramente, através do número de *token*s em comum, conforme realizado na métrica anterior. Em seguida, calcula-se também o **bônus**, que é justamente o diferencial na verificação da similaridade. Para cada *token* encontrado em ambos os documentos, é calculado o bônus de *1/df(j)*, onde *df(j)* é o número de documentos em que o *token j* ocorre no léxico. Desta forma, se um *token* ocorre em muitos documentos, o bônus é pequeno. Logo, se o *token* aparece em poucos documentos, o bônus é maior.

Por último, a métrica mais utilizada e que apresenta os melhores resultados na literatura é a similaridade baseada no ângulo co-seno formado pelos vetores que representam os documentos – ***Cosine Similarity***, em inglês. Para *Cosine Similarity*, apenas os documentos positivos são computados, isto é, apenas os *tokens* que aparecem em ambos os documentos. A freqüência dos *tokens* também é considerada, fazendo com que o cálculo da similaridade resulte no conjunto de equações abaixo:

$$w(j) = tf(j) * \lg(N / df(j))$$

$$norm(D) = \sqrt{\sum w(j)^2}$$

$$\cos ine(d_1, d_2) = \sum (w_{d1}(j) * w_{d2}(j)) / (norm(d_1) * norm(d_2))$$

O peso de um *token* em um documento *w(j)* é computado pela fórmula *tf-idf*, onde *j* é o j-ésimo *token* no léxico, *tf(j)* é a freqüência deste *token* no documento, *N* é o número de documentos no *corpus* e *df(j)* é o número de documentos no qual o *token* figura. *Cosine* é definido na literatura como a métrica padrão de medida de similaridade entre documentos, servindo como *benchmark* para efeito de comparações entre novas medidas propostas por pesquisadores e estudiosos.

3.3.3. Listas Invertidas

A estrutura de dados que "alavancou" a área de Recuperação de Informação, principalmente no campo de Máquinas de Busca, foi a estrutura de **Listas Invertidas**. A principal característica desta estrutura é que, ao invés de termos documentos "apontando" para *tokens*, temos os *tokens* indicando em quais documentos estes estão contidos. A Figura 8 ilustra uma estrutura aonde os documentos apontam para os *tokens*. Já na Figura 9, é ilustrada a estrutura de Listas Invertidas e sua principal característica.

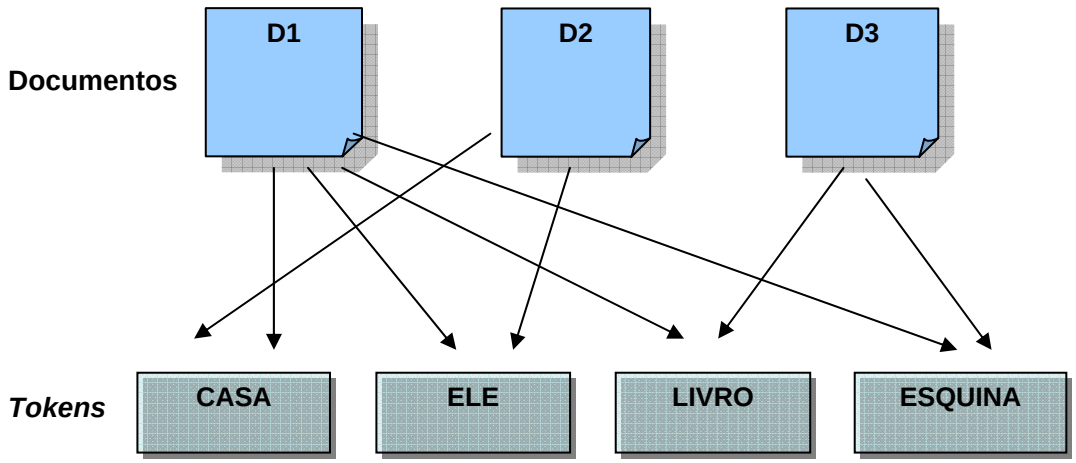

Figura 8 – Documentos "apontando" para seus *tokens.*

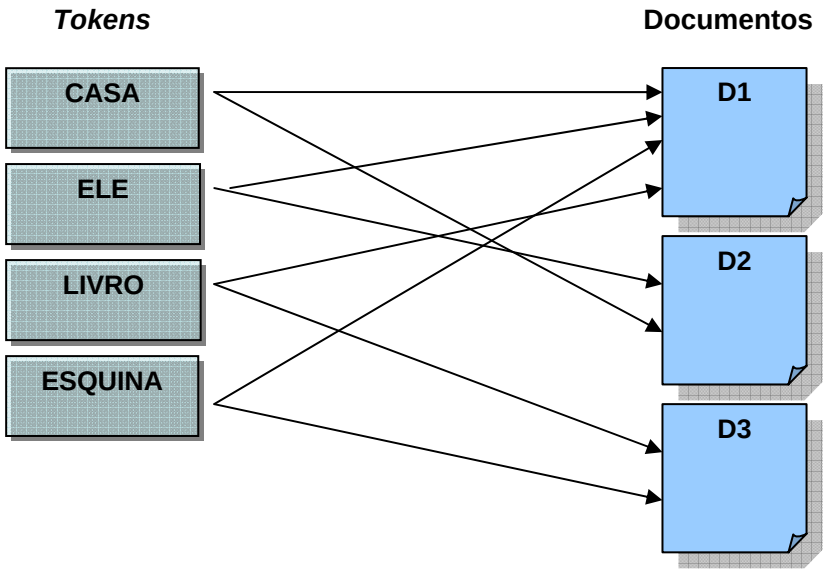

Figura 9 – Estrutura de Lista Invertida com os *tokens* “apontando” para os documentos.

O objetivo principal de um sistemas de Recuperação de Informação é otimizar a velocidade de processamento de uma consulta solicitada por um usuário. A estrutura de Listas Invertidas possibilita o rápido acesso para a resolução de consultas, uma vez que o acesso a cada palavra digitada pelo usuário é feito automaticamente pelo índice, evitando que seja feita uma “varredura” seqüencial em todos os arquivos, a fim de se encontrar as palavras envolvidas na consulta.

Tipicamente, existem dois tipos de índices: o primeiro tipo é tido como simplório onde cada *token* de um léxico referencia os documentos aonde ele se encontra; e o tipo mais especializado que, além de indicar em quais documentos ele está, informa também em que posição dentro do arquivo ele figura. Essa segunda versão, sem dúvida, permite a elaboração de consultas mais complexas e eficientes, com a penalização de demandar muito mais tempo para a criação e manutenção do índice, bem como um aumento significativo do espaço de armazenamento gasto.

3.3.4. Processamento de Consultas

Após ter sido realizado o processo de Indexação de todos os documentos que compõem o *corpus*, o sistema de Recuperação de Informação está pronto para que consultas sejam realizadas. Entende-se por consulta uma necessidade

específica de um usuário, que ao precisar de determinada informação acessa o sistema e descreve o que deseja obter através de palavras-chave.

As Máquinas de Buscas da Internet, como o Google, se utilizam de tudo o que é possível para o processamento da consulta do usuário. Para tanto, partem do princípio que a informação, de fato, existe e que está "perdida" em algum lugar na *Web* e sua principal função é encontrá-la.

O poder de Recuperação de um sistema está, principalmente, na capacidade deste encontrar **exatamente** o que o usuário precisa. Para tanto, é necessário que as consultas possam ser elaboradas de forma flexível. Por exemplo, determinado usuário gostaria de obter todos os documentos aonde apareçam as palavras "Bush" e "Iraque". Podemos definir os principal operadores, que são: **AND, OR** e **NOT.** As consultas que se utilizam de tais operadores são chamadas de consultas *booleanas*, aonde caso não são encontrados exatamente o que se deseja, nada é retornado ao usuário. Abaixo estão três exemplo de utilização dos operadores descritos:

Pelé **AND** Política
(Pelé **AND** Futebol) **NOT** Maradona
(Pelé **OR** Garrincha) **AND** Argentina

A interpretação destas consultas é bem simples. Na primeira, o usuário está interessado em ver todos os documentos que, de alguma forma, citam Pelé e Política juntos. Já na segunda consulta, o interesse está em achar documentos que ligam Pelé a Futebol, sem que haja qualquer citação sobre o também ex-colega de profissão Maradona. Finalmente, na terceira consulta o usuário necessita obter documentos que citem Pelé ou Garrincha e que, necessariamente, citem o país vizinho Argentina.

Existe ainda outras funcionalidades que podem ser disponibilizadas, como a **Busca por Frase** e a utilização de caracteres **"curinga"**. A Busca por Frase ou *Phrase Query*, em inglês, é o nome dado à formulação de consulta através da concatenação de palavras, formando uma frase, conforme o exemplo "Romário jogou no Flamengo". Note que para que algum documento seja recuperado, ele necessariamente precisa conter as palavras "Romário","jogou","no" e "Flamengo", respeitando a ordem de aparição e a sua correta concatenação. Desta

forma, se algum documento contiver a frase “Romário jogou há muito tempo atrás no Flamengo”, este não será recuperado, dado que existem outras palavras intermediárias não desejadas entre as que estão sendo procuradas. Para que haja uma implementação de processamento de consulta que contemple a Busca por Frase, é necessário que se faça uso do índice que, além de armazenar os *tokens* e seus documentos, registre também a posição de cada *token*, conforme visto na seção 3.3.3.

Quanto à utilização de caracteres curingas, estes servem para que sejam recuperados documentos que contenham palavras parcialmente informadas pelo usuário. Por exemplo, supondo que o usuário gostaria que fossem recuperados todos os documentos que contenham as palavras “urubu” e “urucum”. A busca poderia ser formulada através da utilização de curingas, como “uru*”, aonde o * faz o papel de “curinga”. Todos os documentos que contivessem ambas as palavras seriam recuperados, assim como ocorreria na consulta “urubu” **OR** “urucum”. A diferença está que, na consulta que utiliza curingas, também são retornados documentos que contenham quaisquer palavras que comecem com “uru”, como “uruguaiana”. A estrutura de dados auxiliar que possibilita a utilização desse tipo de busca é chamada de **árvore B** [45].

3.3.5. Avaliação das Consultas

Avaliar o resultado das consultas é o último estágio de um processo de Recuperação de Informação. Existem, basicamente, dois critérios que são utilizados para se avaliar uma consulta: **tempo de resposta** e **qualidade dos resultados.** Um sistema de busca ideal deveria retorna os documentos mais relevantes no menor tempo possível. Entretanto, a relação entre tempo de resposta e qualidade dos dados é conflitante, isto é, quanto mais preciso for o resultado da consulta, mais tempo leva para que se operacionalize sobre a estrutura de índices.

Existem vários outros fatores que afetam essa relação, como a estrutura de *hardware* disponível, implementação e tamanho do índice, tempo de latência e tempo de acesso ao disco.

Focando apenas na qualidade dos dados retornados pelos sistemas de RI, estão duas métricas consideradas, hoje em dia na literatura, como sendo padrão na

avaliação do resultado do processamento de consultas – ***Precisão*** e ***Recall***- mas que não são de uso exclusivo dos sistemas de RI, sendo bastante utilizadas também para efeito de medição do resultado de algoritmos de Classificação de Textos, Descoberta de Entidades Nomeadas entre outros. Tanto Precisão como *Recall* são explicadas de forma detalhada na seção 4.1.2.

A avaliação de Máquinas de Busca da *Web* introduziu uma nova forma de avaliar os resultados das consultas, através do critério de ***ranking***. O principal objetivo desse novo critério foi observar se, dada uma consulta por palavras-chaves, o resultado continha os documentos mais relevantes nas primeiras posições, aparecendo logo que possível para o usuário. Isto é extremamente importante para o acesso, visto que quase toda consulta na *Web* retorna uma grande quantidade de documentos, dada a grande proporção que tomou a maior rede de computadores.

Para que se implemente o *ranking* do resultado, algumas estratégias focadas no ambiente *web* são importantes, levando em consideração a própria estrutura de documentos e interligação entre estes que, como já citado, é similar a um grafo direcionado.

A mais intuitiva estratégia baseia-se no conceito de **"popularidade"** de um documento, ou no caso da Internet, de uma página HTML. Assim, são contabilizados o número de *in-links* que um página possui, ou seja, o número de *links* que "apontam" para determinada página HTML e que, fisicamente, estão dentro de outras páginas HTML. *Websites* como *cnn.com* e *terra.com.br*, teoricamente, deveriam possuir um maior número de *in-links* que um site pessoal ou de menor abrangência.

Existem na literatura dois importantes algoritmos que implementam o *ranking* na *web* através de sua estrutura. O primeiro é o **HITS** [46] criado por Jon Kleinberg [47] e que tem como princípio básico a divisão das páginas na Internet em duas categorias: *Hubs* e *Authorities*. O outro é o ***PageRank*** [48], famoso por ser o algoritmo utilizado pela Máquina de Buscas do Google, tendo sido criado pelos próprios fundadores da empresa, Larry Page e Sergey Brin como parte de suas pesquisas de Doutorado na Universidade de Stanford.

3.4. Mineração

A fase de Mineração envolve decidir quais algoritmos deverão ser aplicados sobre a massa de dados desenvolvida até o momento. Para tanto, deve se optar por uma ou mais Tarefas de Mineração, que nada mais é do que decidir o que se quer obter de informação. Por exemplo, se a necessidade de informação do usuário é obter o relacionamento entre documentos, verificando o grau de similaridade e a formação de grupos naturais, então a tarefa a ser escolhida é a ***clusterização.*** Em contrapartida, se estes grupos de documentos já existem, seja pela execução de algoritmos ou pelo conhecimento prévio de especialistas, então a indicação de aonde um novo documento deve ser "encaixado" é conseguida através de algoritmos de **classificação.**

Embora as tarefas de *clusterização* e classificação sejam compartilhadas entre Mineração de Textos e Mineração de Dados, outras são específicas da primeira, como a **sumarização** e **extração de características.** No próximo capítulo, são exploradas todas as Tarefas de Mineração de Textual, assim como a relação dos principais algoritmos.

3.5. Análise da Informação

A Etapa de Análise da Informação também pode ser chamada de Pós-processamento de dados e diz respeito à verificação da eficiência da aplicação dos algoritmos da etapa anterior. Em outras palavras, é o momento de se avaliar se o objetivo foi cumprido da melhor forma possível, que é descobrir conhecimento novo e inovador a partir de pilhas de documentos não-estruturados.

Existem diversas maneiras de se avaliar a mineração como um todo, seja de forma **qualitativa** ou **quantitativa**. A utilização de **métricas**, conforme já mencionado, é considerada uma forma quantitativa, ao passo que a utilização do **conhecimento de especialistas** no domínio é considerada uma forma qualitativa. Os especialistas devem sempre ser consultados, em todas as etapas da Mineração, balizando a análise, ajudando a resolver situações de conflito, indicando caminhos e complementando informações. Entretanto, alguns conflitos podem ocorrer como

a divergência de opiniões entre dois ou mais especialistas, bem como, a própria mudança de opinião de um mesmo ao longo do tempo.

Por último, a forma mais intuitiva de se analisar um resultado é fazendo uso de elementos gráficos, através de **ferramentas de visualização**. A introdução de gráficos, com noções de cores e distâncias, ajuda a entender o sentido de grandes e complexos conjuntos de dados, que não são facilmente manuseados.