PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA METODOLOGIA PARA RECUPERAÇÃO DE DOCUMENTOS BASEADA EM TEXTOS COMPLETOS UTILIZANDO CONHECIMENTO DE FUNDO

Aluno: Kelvin Vieira Kredens
Orientador: Prof. Dr. Bráulio Coelho Ávila

Curitiba
Março/2012

Kelvin Vieira Kredens

UMA METODOLOGIA PARA RECUPERAÇÃO DE DOCUMENTOS BASEADA EM TEXTOS COMPLETOS UTILIZANDO CONHECIMENTO DE FUNDO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração:
Agentes de Software

Orientador:
Prof. Dr. Bráulio Coelho Ávila

Curitiba
Março/2012

DADOS DA CATALOGAÇÃO NA PUBLICAÇÃO
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
SISTEMA INTEGRADO DE BIBLIOTECAS SIBI PUCPR BIBLIOTECA CENTRAL

Kredens, Kelvin Vieira
UMA METODOLOGIA PARA RECUPERAÇÃO DE DOCUMENTOS BASEADA EM TEXTOS COMPLETOS UTILIZANDO CONHECIMENTO DE FUNDO / Kelvin Vieira Kredens - 2012.

Dissertação (Mestrado) - Pontifícia Universidade Católica do Paraná, Curitiba, 2012
Orientador: Bráulio Coelho Ávila, Programa de Pós-Graduação em Informática.

1 - Recuperação de Informação 2 - Modelo Espaço Vetorial
3 - Extração Sequência de Texto 4 - Sequências Máximas Frequentes

À amada minha esposa Paula, pela compreensão e carinho, aos meus irmãos Kelston e Kenndra, por estarem sempre ao meu lado e principalmente aos meus pais Rubens e Rosemeri pela sua dedicação incondicional para que eu tenha chegado até aqui.

Agradecimentos

Aos meus pais, por me educarem, sempre pelo exemplo, me ensinando que o caminho a ser percorrido não é fácil, mas deve ser percorrido. E que cada queda serve para nos ensinar a ser cada vez melhores, nos deixando mais fortes.

À minha esposa, Paula, por sempre estar ao meu lado, mesmo nos momentos mais difíceis durante esse trabalho.

Ao Prof. Dr. Bráulio Ávlia Coelho, pela paciência e orientação durante a execução deste trabalho, que no decorrer do tempo tornou-se mais do que um orientador, tornou-se um amigo de boas conversas.

A Prof. Dra. Denise Maria Werneck Farani de Carvalho por me auxiliar na coleta de dados e aos seus alunos do 4^o de Comunicação Social por colaborarem com tanta atenção na coleta de dados para os experimentos.

Ao Prof. Dr. Edson Emílio Scalabrin e Prof. Dr. Fabrício Enembreck pela ajuda prestada durante o decorrer do trabalho.

À Jane, que também, já se tornou mais do que uma chefe, pela sua paciência e apoio para a conclusão deste trabalho e que em nome do SERPRO (Serviço Federal de Processamento de Dados) me forneceu condições para a execução deste trabalho.

Ao Milton, meu amigo de todos os dias, pelas palavras de apoio.

Ao Allston, por ter me ajudado, sem tamanha intenção, a resolver um dos quebra-cabeça do trabalho.

À todos os meus amigos, que deixariam a lista muito grande, mas sabem que existem e sempre estão por perto quando precisamos.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O objetivo deste trabalho é criar um mecanismo de recuperação de documentos baseado em exemplos, onde o usuário ao invés de descrever o que procura, ele fornece um documento como exemplo. Com base em tal documento o sistema identifica, em uma coleção de documentos, outros semelhantes e os retorna ranqueados de acordo com a sua semelhança. Para isso utilizou-se de sequências de palavras, ao invés de palavras simples, nos processos de indexação e recuperação. Para a extração das sequências importantes utilizou-se a técnica chamada Sequencias Máximas Frequentes, onde é criada uma representação computacional de cada documento, contendo suas sequências máximas frequentes e o peso de cada uma. Para o cálculo do peso, duas heurísticas novas foram definidas. A primeira em relação ao tamanho das sequências e a segunda com base na região, onde as sequências estão dispostas no corpo do documento, essa sendo o alvo principal da pesquisa.

Palavras-chave: Recuperação de Informação, Sequências Máximas Frequentes, Modelo Espaço Vetorial, Extração Sequência de Texto.

Abstract

The main goal of this work is to create a search engine for documents based on example, where the user instead of having to describe what he wants, it should provide a document as an example. Based on this document the system will identify in a collection of documents, other similar documents and will return they in a ranked list accordantly to the similarity. For that will be proposed the use of sequences of words, rather than simple words,in the process of indexing and search. For the extraction of important sequences will be used technique called Maximal Frequent Sequences, which will create a computational representation of each document, containing his sequences and weight of each. To calculate de weight, two new heuristics will be proposed. The first take in account the size of the word sequences and the second will be based on the region in the text they appear this is the principal point of the work.

Keywords: Information Retrieval, Maximal Frequent Sequences, Space Vetorial Model, Text Sequence Extraction.

Capítulo 1

Introdução

Diz-se que a invenção tecnológica mais importante até hoje é a língua escrita, a medida que ela permite que o conhecimento seja passado de geração em geração sem perder o teor original. Após isso, outra invenção muito importante e diretamente ligada a língua escrita é a invenção da prensa tipográfica, por Gutenberg, em 1439, que possibilitou que o conhecimento escrito em forma de livros se tornasse mais acessível a todas as classes da sociedade. Atualmente estamos vivendo o tempo de outra invenção muito importante, responsável pela chamada revolução da informação, que é a Internet, criada por Tim Berners Lee, em 1989, e que tem mudado drasticamente a maneira como a informação é gerada e consumida, dando mais valor a ela, e, a quem à detêm e sabe fazer bom uso dela. É nesse ponto que os sistemas de recuperação de informação se enquadram, o estudo [ZZY+08], identificou crescimento exponencial, obedecendo a lei de Moore [1] , com uma taxa na qual a Internet dobra de tamanho a cada 5,32 anos. Perante esse tamanho e essa taxa de crescimento, a ação de encontrar algo torna-se um trabalho árduo. Ganha quem encontra o que procura antes. Recuperação de informação, de acordo com [SWY75], pode ser definido como ação de encontrar material, normalmente documentos, de natureza não estruturada(texto livre), que satisfazem uma necessidade de um usuário, dada uma base de documentos.

Dados não estruturados, particularmente, colocam um novo marco e demandam novas tecnologias no tocante a recuperação de informações. São exemplos de informação armazenada na forma de texto livre, artigos, websites, entre outros. Websites, por exemplo, utilizam uma linguagem chamada HTML para sua criação, ou seja, existe uma linguagem que define, de forma estruturada, seu conteúdo, porém a HTML atual só estrutura o conteúdo dos websites quanto ao layout do mesmo, ou seja, a informação que é de interesse do navegador web. Tal informação serve para os navegadores renderizarem a página para o usuário. A HTML não estrutura a informação que é de interesse do usuário final, o conteúdo de uma notícia, os personagens de uma história, entre outros. O mesmo

[1] Lei de Moore previa que a capacidade de processamento dos chips dobraria a cada 18 meses.

acontece com arquivos PDF: existe uma estrutura no documento, mas tal estrutura é de uso somente dos aplicativos que montam o documento para o usuário final possa lê-lo, ou seja, como a HTML, não define estrutura para a informação que está dentro do documento, sendo essa a informação que interessa ao usuário e é essa informação o foco deste trabalho.

Ao contrário do que é praticado pela maioria dos sistema de recuperação de informação, a proposta deste trabalho é utilizar, para representar conteúdo dos documentos, palavras compostas, ao invés de palavras simples. Palavras compostas, chamadas de colocações, que de acordo com [Sma93], podem ser definidas como uma combinação recorrente de palavras, que co-ocorrem com uma frequência maior que ao acaso e correspondem à um uso arbitrário das palavras. Colocações possuem um valor semântico muito maior, e isso é a maior motivação para seu uso, elas expressam mais informações do que uma única palavra.

1.1 Motivação e Hipótese

Com o grande aumento da produção de informação e do valor que tal informação possui, se faz necessário a construção de novos mecanismos de busca, que façam uso de novas ideias, como uso de sequências importantes de texto para indexar e buscar textos, bem como a utilização, em contra partida das listas ranqueadas. Também observa-se a deficiência dos mecanismos de busca atuais, onde o usuário deve descrever o que procura e não pode, simplesmente, fornecer um exemplo.

Propõem-se que a utilização de sequências importantes de texto, chamadas colocações, nos processos de indexação e busca sejam mais eficazes se comparado à utilização de palavras simples. Com isso espera-se, também, que o índice gerado, para representar cada documento, seja menor. Uma vez que são armazenadas somente as colocações que se enquadram como sequências, frequentes e máximas. Tal conceito é apresentado e discutido no Cápitulo 2 Seção 2.3. Em conjunto com o uso das colocações, outra hipótese a ser investigada diz respeito à criação de um conhecimento de fundo. Transformado em uma memória de cálculo a ser utilizado para valorar a importância de sequências de texto de acordo com a disposição das mesmas no corpo do texto.

1.2 Objetivos

O objetivo geral deste trabalho é criar uma metodologia que permita recuperar documentos com base em um exemplo, utilizando técnicas de extração de sequências importantes do texto para serem utilizadas no processos de indexação e recuperação. Tal metodologia faz uso de conhecimentos de fundo para valorar a importância de regiões

do texto, valorar a importância de sequências de texto de acordo com seu tamanho, utilizando para isso sequências de texto como descritores do documento. A metodologia criada realiza a busca com base em exemplo que visa facilitar o processo de busca quando o usuário não sabe o que procurar, ou como procurar, pois ele não precisará descrever o que quer e sim fornecer um exemplo.

1.3 Estrutura do Documento

Este documento está organizado da seguinte maneira. Capítulo 2 aborda a base teórica utilizada neste trabalho, desde Recuperação de Informação, passando pelo Modelo Espaço Vetorial e então descrevendo a teoria das Sequências Máximas Frequentes. Capítulo 3 descreve o método proposto, que a criação de um mecanismo de recuperação de documentos baseado em exemplo onde a valoração da ligação entre os documentos é feita avaliando-se as sequências máximas frequentes, presentes no documento, com base em seu tamanho, frequência e posição no texto. Capítulo 4 apresentada a ferramenta AVALIA, criada para coletar dados à serem utilizados na validação do método proposto. No Capítulo 5, o método proposto é implementado e testado. E, por último o Capítulo 6, onde conclui-se o trabalho e discute-se os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Considerações Iniciais

A princípio, armazenar e recuperar informação é uma tarefa trivial. Tendo-se uma coleção de documentos, um usuário e uma necessidade o usuário precisa ler todos os documentos separando aqueles que atende a necessidade inicial. Porém, sabe-se que essa tarefa é humanamente impossível por conta da quantidade de informação disponível atualmente. É ai que entra a Recuperação de Informação e todos os seus mecanismos, teorias e métodos que automatizam tal tarefa.

Neste capítulo são abordados conceitos referentes a Recuperação de Informação e por fim as MFS (Sequências Máximas de Texto).

2.2 Recuperação de Informação

O primeiro autor a utilizar o termo "Recuperação de Informação", sendo referenciada nesse trabalho como RI, foi o americano Calvin N. Mooers, em seu trabalho [Moo50], onde ele a definiu como sendo o problema de direcionar o usuário à uma informação armazenada, que ele pode nem não ter conhecimento da sua existência. Por conta dessa necessidade, por ele identificada, toda a teoria da RI foi criada e tem se tornada cada vez mais importante por conta da crescente quantidade de informação que tem nos estado disponível por conta da Internet. Em [MRS08] recuperação de informação é definida como:

> Recuperação de Informação(RI) é a ação de encontrar material (normalmente documentos) de natureza não estruturada (normalmente texto livre) que satisfaçam uma necessidade de informação a partir de uma grande coleção de documentos (normalmente armazenados em computadores).

Na Figura 2.1, pode-se verificar a arquitetura básica de um sistema de recuperação de informação e pode ser dividido em 3 macro atividades: 1) Extrair e representar (For-

mulação da Query) a necessidade do usuário, 2) Extrair, representar e armazenar (Indexação) o conteúdo dos documentos da coleção e 3) Comparar a necessidade do usuário aos conteúdos armazenados (índice) dos documentos que compõe a coleção.

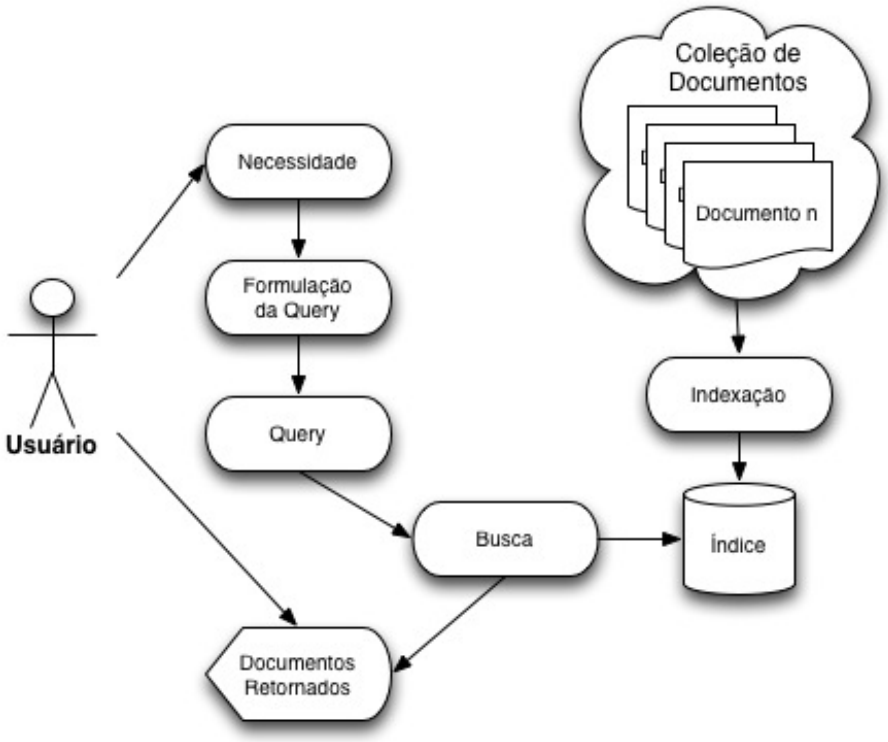

Figura 2.1: Arquitetura do Básica Sistema de Recuperação de Informação

Os sistemas de informação são caracterizados pelo seu modelo interno o qual é o responsável em representar e manipular a necessidade do usuário e o conteúdo dos documentos da coleção e efetuar a comparação entre eles. Existem inúmeros modelos para sistemas de RI, à saber: Booleanos, Probabilísticos, baseados em Redes Bayesianas, Regras Linguísticas e *Page Rank*.

O foco deste trabalho é o uso do Modelo Espaço Vetorial, amplamente utilizado por ser eficaz, extensível e simples

2.2.1 Modelo Espaço Vetorial

Desenvolvido na década de 60 por Salton e sua equipe, [SWY75], este modelo é baseado na ideia de representar e mensurar a similaridade semântica entre documentos de maneira espacial. Cada documento é transformado em um vetor de pares ordenados (token,peso). O conjunto de pesos que compõe um documento é transformado em um ponto em um espaço multi-dimensional. Dessa maneira a similaridade entre documentos pode ser calculado utilizado técnicas clássicas da álgebra linear.

Representação de um Documento. Assumindo um conjunto T, composto pelos tokens que representam um documento D, o vetor no modelo espaço vetorial de D é o vetor $\overrightarrow{D}$ com dimensão igual a T.

$$\overrightarrow{D} = ((t1, w_{d1}), (t2, w_{d2}), (t3, w_{d3}), ..., (tT, w_{dT}))$$

Onde (ti, w_{di}), é o par ordenado composto por ti que é token, associado ao seu peso w_{di}.

O token é a unidade mínima extraído do documento. Normalmente o token é uma palavra que compõe o documento. A extração dos tokens é executada durante o pré-processamento, onde podem ser removidas *stopwords* e/ou realizados processos chamados *stemming*. *Stopwords* é uma lista de palavras que ocorrem com frequência e não agregam valor à representação do documento. O *stemming* é um processo executado com o intuito de se extrair das palavras a serem indexadas somente o radical da mesma. Podem ser removidos plurais, conjugações temporais de um verbo, etc.

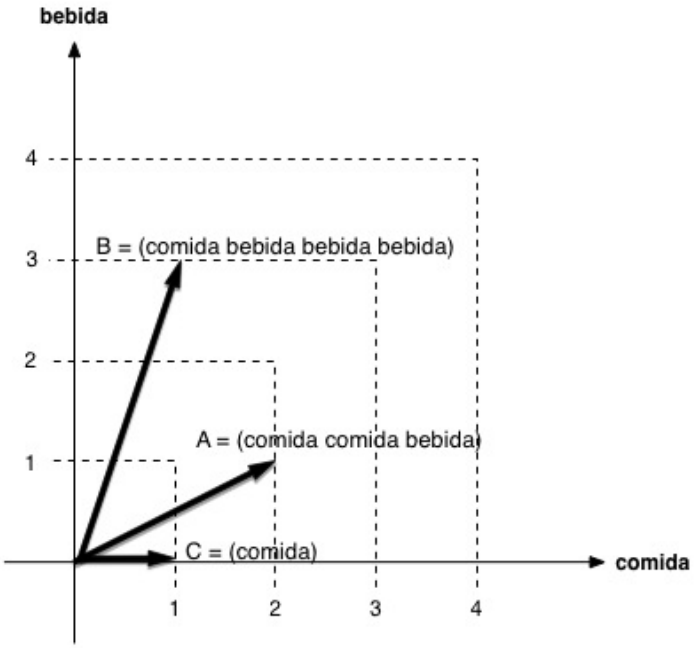

Figura 2.2: Exemplo de um espaço bi-dimensional contendo pontos dos documentos A, B e C

Tomando-se como exemplo, duas palavras *comida* e *bebida* e considerando que os documentos só possam ser composto por essas 2 palavras, então todos os documentos seriam representados por um vetor de 2 posições. Supondo 3 documentos distintos, A, B e C com essas 2 palavras dispostas da seguinte maneira:

- Documento A - " comida comida bebida "
- Documento B - " comida bebida bebida bebida "

- Documento C - “ comida ”

Poderiam ser representados, como segue:

$$\text{Documento A} = \overrightarrow{A} = ((comida, 2), (bebida, 1))$$
$$\text{Documento B} = \overrightarrow{B} = ((comida, 1), (bebida, 3))$$
$$\text{Documento C} = \overrightarrow{C} = ((comida, 1), (bebida, 0))$$

A Figura 2.2 mostra como três vetores são representados em um espaço cartesiano.

Uma das vantagens do modelo espaço vetorial é a sua flexibilidade, por exemplo, na representação dos documentos, nas posições do vetor, palavras simples não são a única característica de um documento que pode ser utilizada. Sequências de palavras, ou colocações, também podem ser utilizadas bem como qualquer outra característica mensurável para caracterizar um documento.

Cálculo do peso dos termos. O desempenho de uma aplicação de RI está diretamente ligado ao cálculo dos pesos dos termos. Uma bom cálculo do peso de um termo é aquele que consegue refletir a significância de um termo com relação ao conteúdo do documento onde ele aparece. No exemplo da Figura 2.2 utilizou-se a frequência dos termos no documento como meio de valorar sua importância. Tal medida foi criada originalmente por Luhn, e apresentada no trabalho [Luh57]. Mas sozinho não é uma métrica muito eficaz pois ele não leva em conta todo universo de termos existentes na coleção de documentos.

Isso motivou a criação de outra métrica pra a valoração dos termos em relação a coleção de documentos chamada Frequência Invertida nos Documentos ,iDF, apresentada em [Jon72]. Para evitar resultados muito alterados a Fórmula (2.1) é mais comumente utilizada, baseada em logaritmos.

$$iDF_p = log(\frac{N}{DF_p}) \tag{2.1}$$

Onde N é o número do total de documento da coleção, DF_p é a quantidade de documento que contêm o termo p.

O TF valora a importância de um termo para o documento, enquanto o iDF mede a importância do termo vis-à-vis a coleção de documentos. E, geralmente, são combinados conforme a Fórmula (2.2).

$$TFiDF_p = TF_p \times iDF_p \tag{2.2}$$

O cálculo do peso $TFiDF$ ainda não está completo. Ele não considera o tamanho dos documentos. Um documento sendo maior que os outros da coleção, com maior chance, também possuirá termos mais frequentes e com pesos maiores. Para evitar isso o cálculo do TF pode ser normalizado. Uma das maneiras está apresentada na fórmula (2.3).

$$TF_p = \frac{n_p}{max(n_i)} \tag{2.3}$$

Onde, considerando o documento D e o termo p, contido nesse documento, então n_p é a frequência do termo p e $max(n_p)$ é o tamanho da maior frequência de um termo existente no documento D.

Um termo para receber um peso alto deve ser muito frequente nos documentos onde aparece e pouco frequente em todos os outros documentos da coleção. Por exemplo, o termo "comida", se ele for muito frequente e aparecer em todos os documentos da coleção e o usuário pesquisar por "comida", então, qual documento da coleção vale mais?

Cálculo da similaridade. A partir do momento que cada documento descreve um ponto multi-dimensional em um espaço cartesiano várias medidas podem ser feitas. Uma delas é medir a distância Euclidiana entre os pontos. O problema dessa métrica fica por conta do tamanho dos documentos. Mesmo com a normalização do TF os pontos podem se distanciar consideravelmente um em relação ao outro apenas por conta do tamanho dos documentos.

A métrica mais amplamente utilizada é medir o cosseno formado pelos ângulos das duas retas descritas pelo ponto de cada vetor, conhecida por Similaridade do Cosseno. Ela reduz o problema do tamanho dos documentos e pode ser calculado pela fórmula (2.4).

$$Similaridade(A, B) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\parallel \overrightarrow{A} \parallel \cdot \parallel \overrightarrow{B} \parallel} = \frac{\sum_n^{i=1} A_i \times B_i}{\sqrt{\sum_n^{i=1} (A_i)^2} \times \sqrt{\sum_n^{i=1} (B_i)^2}} \tag{2.4}$$

Se o cosseno for igual a zero, então as retas descritas são perpendiculares, ou seja, os documentos não compartilham nenhum termo. Já, se o resultado for 1, não necessariamente quer dizer que os documentos são iguais. Somente reflete que as 2 retas são coincidentes. Já o vetor que as descreve pode não ser igual, ou seja, os termos podem estar, por exemplo, em ordens diferentes. O que se pode dizer é que os pesos dos termos, ao final dos cálculos, são iguais.

Recuperando documentos de acordo com a *query* .

Após todos os documentos estarem indexados, ou seja, terem seus vetores calculados o processo de recuperação é simples. A necessidade do usuário, normalmente texto livre é convertida na *query*, podendo passar pelo mesmo pré-processamento que os textos passaram ao serem indexados. A *query* é então representada como um vetor, do mesmo modo que os documentos e então é calculada sua similaridade em relação a todos os documentos da coleção. Considerando-se, por exemplo, a Similaridade do Cosseno, então deve-se calcular o angulo formado pela reta da *query* e cada documento da coleção.

2.3 Sequências Máximas Frequentes

Como já dito, no processo de indexação, um documento é dividido em tokens[1]. Normalmente os tokens utilizados para indexar um documento são palavras simples, porém sabe-se que um documento é um conjunto de palavras ordenadas, ou seja, para fazer a leitura, e posterior compreensão, a ordem das palavras é importante. Por esse motivo é que a ordem das palavras não pode ser descartada no processo de indexação. Em [ZTMfE97], são levantados vários problemas quanto ao uso de palavras simples na indexação de documentos, onde, é citado, além do problema da ordem em que as palavras aparecem, a questão de que sequências de palavras podem ter significado totalmente diferente de quando são utilizadas agrupadas, um exemplo é a sequencia de palavras "cachorro quente". Então, como é feito com palavras simples, para se utilizar sequências de palavras para indexar documentos é necessário descobrir tais sequencias. Outro problema em relação ao uso de palavras simples é o tamanho que o vetor que representa do documento, ele acaba ficando com um tamanho muito grande, pois documentos costumam ter muitas palavras.

Existem inúmeras técnicas para es extrair sequências de palavras, que vão desde extração de bigramas, extração com base nas *part-of-speech*, extração de colocação com base no vocabulário ou unidades léxicas.

Porém, como avaliado em [DAm04], as técnicas existentes possuem fraquezas sendo que utilizam regras de linguística muito especificas, não conseguem lidar com coleção de documentos muito grandes ou, principalmente, não são flexíveis o bastante para permitirem que as sequências extraídas possuam palavras entre as palavras que às compõe. A seguir será apresentado método chamado MFS, do inglês Maximal Frequent Sequences, que lida com as fraquezas dos outros métodos. Uma Sequência Máxima Frequente, de acordo com [AM99], é uma sequência frequente de palavras dada uma coleção de documentos e que não está contida em nenhuma outra sequência maior. Entende-se por sequência como um conjunto de palavras, não necessariamente continuas, que pertencem a uma mesma frase, onde a ordem de aparição é respeitada. O conceito de sequência apresentado é outra característica importante desse método, uma vez que ele respeita a ordem em que as palavras aparecem na frase e permite um intervalo entre as palavras. A ordem utilizada não se dá por acaso, essa ordem é escolhida pelo autor para expressar o conteúdo desejado da melhor maneira e segue a gramática da língua utilizada. Como esse método obedece e utiliza a ordem das palavras na maneira como foram escritas, caso a língua dos documentos analisados seja alterada, nenhuma regra linguística precisa ser alterada, como apontado em [DAM10].

[1]Token é um segmento de texto, de um documento, que possui significado semântico

2.3.1 Definições

Sequencias Máximas Frequentes [2] são definidas [AM99] como segue, considerando um conjunto de documentos S:

> **Definição 1** *Uma sequência $p = a_1...a_k$ é uma subsequência da sequência q se todas as palavras $a_i, 1 \leq i \leq k$ ocorrerem em q e na mesma ordem que aparecem em p. Se uma sequência p for uma subsequência de uma sequência q então, p ocorre em q.*

Considerando a sequência **"temporal logic"**, da Figura 2.3, pode-se verificar que ela aparece em todas as frases. Mas ela é subsequência da sequência **"subset ATSQL temporal logic"**, que aparece em duas das três frases, ou seja, a sequência **"temporal logic"** ocorre em **"subset ATSQL temporal logic"**. Podem ser incluídas restrições quanto ao tamanho das sequências geradas, chamadas de *comprimento máximo* e *mínimo* de uma sequência. Normalmente o *comprimento mínimo* utilizado é 2, sendo que o número de sequências frequentes geradas cai drasticamente quando esse número aumento. Ajustar um *comprimento máximo* para as sequências é problemático, porque se existir alguma sequência maior que o *comprimento máximo* definido então centenas de sequências, com comprimento igual ao *comprimento máximo*, serão geradas, sendo que se tal restrição não fosse utilizada, somente uma única sequência seria gerada, que englobaria todas as outras. Quanto ao *comprimento mínimo*, em algumas situações pode ser interessante usa-lo com valor igual a 1, pois podem existir palavras frequentes, ou seja, que são interessantes para indexar o documento e não formam sequência com nenhuma outra e não estão contidas em nenhuma outra sequencia então caso haja interesse nessas palavras o valor do *comprimento mínimo* deve ser 1.

> **Definição 2** *Uma sequência p é frequente em S somente se p for subsequência de no minimo σ documentos, onde σ é o limiar de frequência minima dado.*

Uma sequência é considerada frequente quando aparecer mais que um determinado número de vezes numa coleção de documentos. Verifica-se que a sequência **"temporal logic"**, da Figura 2.3, aparece nas 3 frases. Um *limiar de frequência mínima* deve ser definido de modo a controlar o número mínimo de vezes que uma sequência deve aparecer para ser considerada. Também pode-se definir um *limiar de frequência máxima* que uma sequência pode aparecer no texto, sua utilização depende do domínio estudado e utiliza-se quando se deseja diminuir o espaço de busca, eliminando assim, sequências muito frequentes.

[2] Daqui em diante, Sequencias Máximas Frequentes serão referenciadas pelo seu acrônimo em inglês *MFS*.

Definição 3 *Uma sequência p é uma (sub)sequência frequente e máxima de S somente se não existir nenhuma outra sequência p' em S de forma que p seja uma subsequência de p' e p' seja frequente em S.*

Ainda utilizando a Figura 2.3 como exemplo, pode-se verificar a existência da sequência frequente **"subset ATSQL temporal logic"**, que aparece em duas das três frases apresentadas. Com isso a sequência **"temporal logic"** não é selecionada pois ela é uma subsequência da sequência **"subset ATSQL temporal logic"** que nesse caso é chamada de sequência máxima frequente. Isso considerando-se um limiar de frequência mínima igual a 2.

- 1. The paper provides a complete characterization of a **subset** of **ATSQL** queries equivalent to **temporal logic** in expressive power.
- 2. Indeed, only a **subset** of **ATSQL** can be mapped back to **temporal logic**.
- 3. The paper gives a complete characterization of safety for queries formulated in **temporal logic**.

Figura 2.3: Exemplos de frases para extração de MFS

O padrão esperado para encontrar uma sequência máxima frequente é flexível, uma vez que são permitidas palavras entre as palavras que formam a sequência máxima. [3] Esse conceito é importante e aumenta significativamente a qualidade das sequências geradas à medida que a linguagem escrita permite muitas variações na maneira como se monta uma frase. Tal flexibilidade é o ponto chave de o porque se permite que existam palavras entre as palavras que formam uma sequência máxima. Só se consegue identificar a sequência **"subset ATSQL temporal logic"** graças à esse intervalo que é permitido entre as palavras. Esse intervalo é considerado a maior virtude desse método, uma vez que isso permite um número maior de combinações de palavras para gerar sequências, sempre obedecendo a ordem em que elas aparecem na frase. O número de palavras permitidas entre as palavras que formam a sequência final pode ser restringido. Essa restrição chama-se *tamanho da janela*. Por exemplo, ao definir-se o *tamanho da janela* igual a 0 são geradas sequências utilizando combinação de palavras subsequentes umas as outras. Essa restrição recebe em geral valores entre 1 e 3.

Ao utilizar-se sequências máximas evita-se a sobreposição de informações, à medida que uma sequência maior, além de englobar sequências menores, possui maior significado semântico por representar de maneira mais fidedigna a intenção do autor. Isso gera um grupo de sequências mais compacto e mesmo assim com um grande valor semântico sobre os documentos analisados.

[3] Questões como redução das palavras para o seu radical ou extração de *stopwords* não serão consideradas nesse momento.

2.3.2 Método de Extração

O método criado e apresentado em [Dou05] combina busca em profundidade e em largura para extrair sequências máximas frequentes de qualquer tamanho e com qualquer *tamanho de janela*. Na prática, o *tamanho de janela* e o tamanho das sequências geradas são limitados ao tamanho das frases analisadas. As únicas restrições utilizadas são as de *frequência mínima* e *frequência máxima* que servem para remover do texto palavras que aparecem mais do que a *frequência mínima* e menos do que a *frequência máxima*.

Primeiramente são gerados conjuntos de pares de palavras, com todas as possíveis combinações de palavras da mesma frase e respeitando a ordem em que elas aparecem. Somente são considerados os pares ordenados que sejam frequentes, de acordo com o *limiar de frequência minima* definido. Tais pares ordenados são chamados *2-grams*, por exemplo (A, B), onde as palavras A e B aparecem na mesma frase e ordem e é uma combinação frequente. Em segundo uma lista, chamada *Max*, contendo todos os *2-grams* frequentes é então criada, que é utilizada para armazenar as sequencias máxima frequente geradas pelo processamento.

```
Input: G3: the frequent 3-grams
Output: Max: the set os maximal frequent sequences

1. n := 3
2. Max := the maximal frequent pairs
3.     While Gn is not empty
4.          For all grams g ∈ Gn
5.              If a gram g is not a subsequence of some m ∈ Max
6.                  If a gram g is frequent
7.                      max := Expand(g)
8.                      Max := Max ∪ max
9.                      If max = g
10.                        Remove g from Gn
11.                 Else
12.                     Remove g from Gn
13.         Prune away grams that are not needed any more
14.         Join the grams of Gn to form Gn+1
15.         n := n + 1
16.  Return Max
```

Figura 2.4: Algoritimo MineMFS

No próximo passo os pares são combinados na tentativa de se formar *3-grams*, por exemplo, os pares (A, B), (B, C) e (B, D) podem ser combinados e formar os *3-grams* (A, B, C) e (A, B, D). Como pode-se ver na Figura 2.4, a lista, chamada G_3 dos contendo os *3-grams* frequentes é utilizada como entrada do processamento. Nesse passo são removidos da lista *Max* todos os *2-grams* que são subsequencia de m *3-grams* da lista G_3.

O algoritmo, com base na lista de *3-grams*, tenta combinar cada elemento com outras palavras, de maneira exaustiva. Somente os *3-grams* que não são subsequência de uma sequência máxima são processados (linha 5 da Figura 2.4). Quando um elemento *3-gram* é combinado com outra palavra e isso gera uma sequência frequente, outras possibilidades de expansão não são checadas e essa nova sequência passa pelo processo de expansão até que não seja encontrada outra sequência maior e frequente, sendo que a ultima sequência frequente gerada é a máxima possível (linha 7 da Figura 2.4). O processo de expansão tenta combinar um *3-gra*m com palavras que estejam na mesma frase, repeitando a ordem de aparição. A combinação pode ser incluir a nova palavra no começo, meio ou fim do *3-gram*.

Após todos os elementos *3-grams* serem processados, aqueles que não podem ser utilizados para formar sequencias maiores que 3 são eliminados (linha 13 da Figura 2.4). Os elementos restantes são colocados numa nova lista e combinados para formar elementos de tamanho 4, ou seja, *4-grams*. Esse processo é repetido até que nenhuma sequencia máxima frequente seja descoberta.

O algoritmo apresentado não consegue lidar com coleções contendo muitos documentos, por questões de performance. Para esses casos foi criada outra versão, chamada *MFS MineSweep*. A diferença basicamente é que na nova versão a coleção de documentos é dividida em subcoleções, menores o bastante para poderem ser processadas. Ao final, as listas de MFSs, geradas para cada subconjunto são unidas, gerando assim uma aproximação das MFSs de todos os documentos. Essa abordagem não foi utilizada nos experimentos à medida que a intenção é aplicar ao algoritmo MineMFS para cada documento separadamente, de forma a conseguir um conjunto de MFS para cada documento. Essa lista de MFS será a representação computacional do documento que deve ser utilizada no processo de indexação.

2.4 Considerações Finais

Neste capítulo foram apresentados o funcionamento básico de sistemas de Recuperação de Informação, o conceito de MFS e de que maneira eles se relacionam. No próximo capítulo é introduzido o método proposto, colocado em prática alguns dos conceitos aqui apresentados.

Capítulo 3

Método Proposto

3.1 Considerações Iniciais

O objetivo deste trabalho é criar um método para recuperação de documentos onde a representação computacional de cada documento indexado seja composto pelas MFS deles extraídas. Tal método cobrirá a criação de uma heurística para coleta e uso de um conhecimento de fundo para valoração da importância das regiões do texto, em conjunto com a valoração das MFS extraídas de acordo com seu tamanho. O método será proposto fazendo busca de documento, utilizando o texto completo dos documentos indexados e do documento exemplo, chamado ALVO, inserido pelo usuário como insumo a busca.

Com isso, espera-se obter maior eficácia na recuperação de documentos semelhantes, que todos os documentos relevantes sejam recuperados e poucos não relevantes. Espera-se, também, que o índice dos documentos indexados seja menor quando comparado à indexação que utiliza palavras simples para representar os documentos. A arquitetura do método proposto pode ser conferida na Figura 3.1 que demonstra os 3 momentos em que o usuário irá interagir com a aplicação. No primeiro ele fornece uma lista de documentos à serem indexados. No segundo momento ele fornece um documento que será o ALVO da busca. Esse documento pode ou não estar indexado, caso não esteja, então ele é indexado. Tal documento será então comparado aos demais já indexados. E, por último, o usuário tem acesso a lista ranqueada, contendo os documentos considerados semelhantes e ordenados de acordo com sua semelhança.

Nesse capítulo serão apresentados os tópicos principais envolvendo a método proposto. O foco é apresentar a ideia pelo espectro do *O QUE*, ou seja, *O QUE* está sendo proposto. Mais a frente, no Capítulo 5, será apresentado o *COMO*. Como o método proposto foi implementado e testado.

O método proposto foi dividido em 3 partes: 1) Representação do Documento, 2) Heurísticas para Valoração das MFS e 3) Recuperação com Base em Exemplo, que serão apresentadas e discutidas no cedorrer deste capítulo.

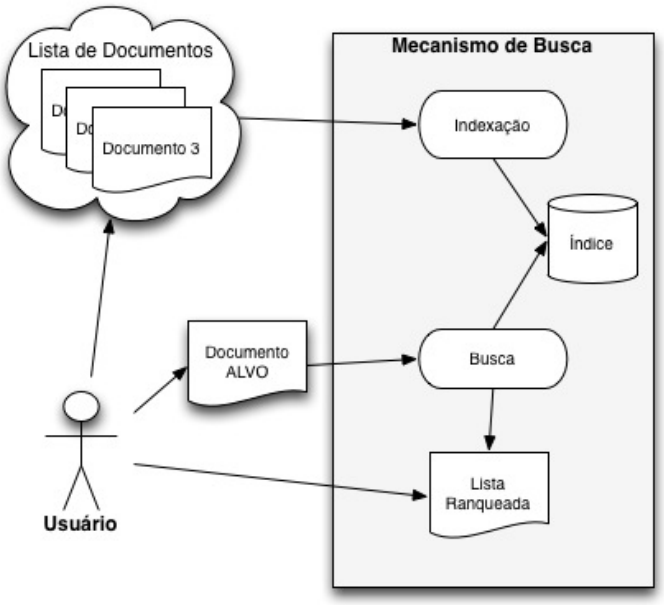

Figura 3.1: Arquitetura do Método Proposto

3.2 Representação do Documento

Em recuperação de informação, os documentos são, geralmente, representados utilizando-se o modelo espaço vetorial, onde cada documento é um vetor formado pelas palavras que o compõe associadas a um peso. As fraquezas apresentadas pro essa abordagem foram apresentadas e discutidas no Capítulo 2 Sessão 2.3.

A proposta deste trabalho é fazer uso de sequências de texto no lugar de palavras simples na representação de documentos dentro do modelo espaço vetorial. Onde as sequências de textos seriam as MFS, conforme apresentadas anteriormente. No processo da indexação as MFS seriam extraídas, valoradas e armazenadas de modo a serem utilizadas no processo de busca. O processo de indexação será detalhado no Capítulo 5.

Com o uso das MFS como tokens para a representação de documentos espera-se obter índices menores e melhor eficácia na recuperação de documentos, uma vez que, entre outras vantagens, a ordem em que as palavras aparecem no texto é respeitada.

3.3 Heurísticas para Valoração das MFS

Ao ponto em que os documentos, sendo representados de acordo com o modelo espaço vetorial e onde os tokens sejam as MFS, o próximo passo é definir como valorar cada posição do vetor. Aqui o objetivo é o que fazer com que o peso de uma MFS reflita a seu teor dentro dos documentos onde aparece em relação a sua expressividade à representação do mesmo.

O objetivo é combinar a fórmula de valoração de tokens, conhecida por *TFiDF*, em

conjunto com as heurísticas propostas a seguir.

Como será descrito no Capítulo 5, a implementação dessas heurísticas será feita de maneira que possam ser ligadas ou desligadas, isso para que sua efetividade possa ser avaliada. As combinações possíveis podem ser conferidas na Figura 3.2.

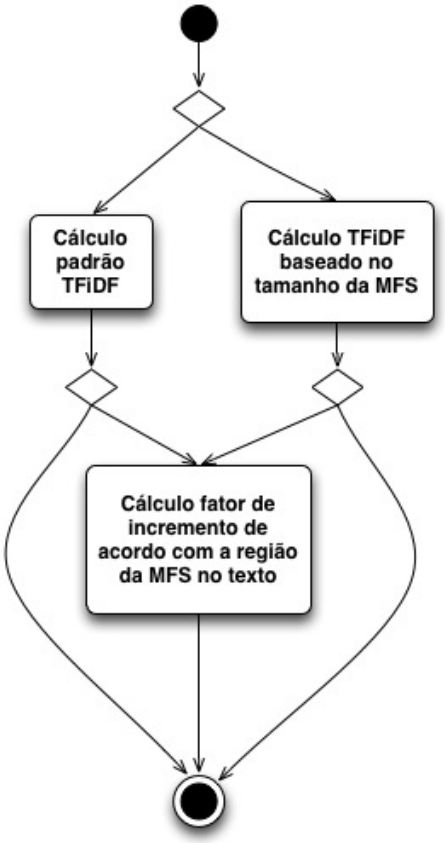

Figura 3.2: Combinações possíveis entre as heurísticas propostas.

3.3.1 Peso pela Região no Texto

A primeira alteração é referente à posição no texto onde uma MFS se encontra. Alvo principal de estudo no trabalho, que é a criação de um conhecimento de fundo. Esse conhecimento foi coletado com avaliadores humanos e transformado em uma memória de cálculo a ser utilizada na valoração de MFS de acordo com a disposição delas no corpo do documento.

Mostramos que dependendo de onde uma MFS apareça no texto ela possua um peso maior, no sentido de representar o conteúdo do documento, em relação a outras. A região de um texto foi calculada pelas frases que o compõe. Um documento com 35 frases terá 35 regiões distintas.

Foi coletado, por meio de uma ferramenta construída, chamada AVALIA, descrita no Capítulo 4 onde, dado um documento avaliadores humanos marcaram, quais as frases

foram consideradas mais importantes do ponto de vista da compreensão do texto. Ou seja, quais são as frases com maior valor semântico referente ao conteúdo expresso pelo documento. Cada frase marcada recebe um ponto. Ao final, as frases mais marcadas terão pontuação maior.

Essas frases foram numeradas de 1 até o tamanho de cada documento e então normalizadas imaginando-se documentos com 100 frases, tamanho máximo considerado possível entre os documentos selecionados para validação da proposta. Com isso o tamanho dos documentos avaliados será normalizado. O par ordenado, posição normalizada e pontução serão então interpolados iniciando na região 1, até a 100 incrementando em 5 posições. Isso irá gerar uma memória de cálculo a ser utilizada pela fórmula (5.2) para identificar qual o peso de uma determinada sequência em relação a um documento.

$$pesoPelaRegiao(MFS) = \frac{pesoPosicao_1 + pesoPosicao_2 + pesoPosicao_n}{totalAparicoes} \tag{3.1}$$

Dada uma MFS, foram recuperadas as frases onde ela ocorre. Cada frase tem sua posição normalizada e então essa posição foig interpolada para se encontrar seu peso referente. Esse processo será iterado repetidades vezes tal qual for a quantidade de aparições da MFS no texto. O peso encontrado será então multiplicado pelo cálculo anterior do TFiDF (metódo padrão ou método proposto referente ao tamanho de uma sequência), de acordo com a fórmula 3.2.

$$pesoFinal = pesoPelaRegiao \times pesoTFiDF \tag{3.2}$$

3.3.2 Peso pelo Tamanho da Sequência

A segunda alteração diz respeito ao tamanho das MFS. A ideia é que quanto maior for uma MFS, mais valor semântico ela carrega e melhor ela servirá para descrever o conteúdo de um documento. Pela fórmula padrão *TFiDF*, uma sequência ganharia peso maior por ser mais frequente em um dado documento e pouco frequente em todos os outros da coleção, ou seja, é uma sequência que serve para discriminar o conteúdo desse documento, mas não considera seu tamanho.

Levanto-se em conta 2 MFS, hipotéticas, que ocorram em igual quantidade no documento X e, também, em igual quantidade entre os documentos que compõe a coleção, elas iram receber o mesmo peso. Porém caso uma dessas MFS seja composta por 2 palavras e a outra por 4 palavras, então, qual deve ser a mais importante? Ao comparar-se o documento X à outros dois documentos, o Y e o Z. Supondo que o documento Y possua a MFS composta por 2 palavras, enquanto o documento Z possua a MFS composta por 4 palavras e supondo também que as ocorrências sejam iguais. Em relação ao documento

X, qual documento deveria ser listado antes? O objetivo é comprovar que valorando as MFS de acordo com seu tamanho melhorará a efetividade da busca.

A fórmula proposta (5.3) é uma alteração da fórmula *TFiDF*, incluíndo um fator de acréscimo de acordo com o tamanho da MFS.

$$peso = (qtdAparicoes \times qtdTotalAparicoes) \times (\frac{tamanhoDaSeq^{fatorIncremento}}{tamanhoMaiorSeq^{fatorIncremento}}) \quad (3.3)$$

Na Tabela 3.1 pode-se conferir uma comparação entre o método proposto e o método clássico. Pode-se verificar que o peso final, no método clássico não sofre alteração de acordo com o tamanho da MFS, enquanto no método proposto tal peso aumenta refletindo o tamanho da MFS. O parâmetro **Fator de Incremento** foi criado se modo a controlar o quanto uma sequência maior deve valer em comparação a uma sequência menor.

Tabela 3.1: Cálculos para exemplificar a comparação entre o método proposto e o método clássico.

Coleção de Documentos								
Total de Documentos = 5								
Total de MFS = 100								
Documento X								
Tamanho da Maior MFS = 5								
Sequência	Tamanho	Qtd	Qtd em outros Documentos	iDF	TF	TFiDF	Fator de Incremento	TFiDF Novo
1	2	3	3	0,22	0,03	0,01	2	3,84
2	3							8,64
3	4							15,36

Na fórmula proposta, foi inserido o parâmetro referente à maior MFS que o documento sendo analisado possui, com o intuito de que as MFS recebam valoração em referência a isso. Caso um documento possua, por exemplo, MFS de tamanho 10, então uma MFS de tamanho 5 vale pouco. Enquanto que caso a maior seja do tamanho 5, então uma do tamanho 5 vale muito.

3.4 Recuperação com Base em Exemplo

Outro ponto que compõe o método proposto pelo trabalho do trabalho é a recuperação de documentos com base em um exemplo. Ou seja, o usuário, ao invés de descrever o que procura ele deverá fornecer um exemplo. Sistemas de recuperação de informações clássicos recuperam um conjunto de documento, relevantes a uma *query* feita pelo usuário,

normalmente escritas em liguagem natural, e os apresentam em forma de uma lista ranqueada. Listas ranqueadas nem sempre satisfazem as necessidades dos usuários. De acordo com [CPGV05b] seu uso é mais adequado quando: 1) o usuário sabe exatamente o que está procurando e 2) o usuário sabe como expressar o que está procurando. Tais pré-requisitos nem sempre existem o que aumenta a dificuldade do usuário encontrar o que está procurando.

O objetivo é que com o uso de um exemplo como base para a busca esses dois problemas sejam resolvidos.

O documento inserido com ALVO seria indexado, da mesma maneira como os demais documento que fazem parte da coleção, para então ter suas MFS comparadas. As MFS seriam as posições no vetor, armazenadas em conjunto com seu peso. Para efetivamente calcular-se a semelhança entre o documento ALVO e os outros documentos a proposta é utilizar o cálculo da Similidaridade do Cosseno.

3.5 Metodologia da Avaliação

Para avaliar-se o método proposto seria necessário compará-lo com o método clássico, que é a indexação dos documentos fazendo uso de palavras simples e o ranqueamento dessas palavras utilizando-se *TFiDF*.

O processo de avaliação criado é composto por 6 grupos de avaliações, cada qual contendo 1 documento chamando ALVO e outros 5 documentos. O documento ALVO é o documento inserido no processo de busca que deve retornar uma lista ranqueada contendo os outros 5 documentos ordenados de acordo com o ALVO. Dado um documento ALVO, qual deveria ser a ordenação correta? Para efetuar os testes, implementou-se o método proposto e o método clássico. Porém, chegou-se a um questionamento, qual a ordenação final era a correta, ou a mais acertada?

Por conta dessa questão, houve a necessidade da criação de dados de controle, os quais pudessem ser comparados entre ambos os métodos no intuito de identificar qual deles mais se aproximou de tais dados. Foi criada uma aplicação chamada AVALIA que tem como principal intuído a coleta das avaliações feitas por seres humanos. Cada avaliador deveria ordenar os 6 grupos de artigos e essa ordenação seria utilizada para comparar os dois métodos. Essa mesma aplicação também será utilizada para coletar informações referentes a que regiões de um documento é mais importante.

Na Tabela 7.1, do Capítulo 7 pode-se verificar a lista de artigos utilizados, relacionando o código ao seu respectivo título.

3.6 Considerações Finais

No próximo capítulo será apresentada a Ferramenta AVALIA que, por conta da metodologia proposta para avaliação, foi criada para coletar dados de avaliações de usuários referentes a analise dos textos.

Capítulo 4

Sistema AVALIA

4.1 Considerações Iniciais

Identificou-se que para fazer a comparação da efetividade entre o método proposto e o método que utiliza somente as palavras simples, seriam necessários dados de controle. Dados esses que serviriam para comparar qual dos métodos obteve melhor resultado na ordenação dos documentos, contidos numa base, com relação à um documento alvo. Os dados de controle utilizados foram coletados com base em avaliações humanas. Especialistas tiveram acesso a documentos alvos os quais deveriam ser comparados à outros documentos e esses, então, seriam ranqueados. Nessa mesma coleta foi solicitado que cada avaliador, marcasse as frases mais importantes de cada documento lido. Essas informações formaram a memória de cálculo utilizada para avaliar a relevância de MFS de acordo com a posição delas no texto. Nesse capítulo será descrita a metodologia utilizada, a ferramenta AVALIA, criada para fazer a coleta das avaliações e os resultados obtidos.

4.2 Metodologia

A metodologia utilizada na coleta dos dados foi derivada da metodologia proposta para a avaliação do método proposto. Foram estipuladas 6 avaliações, cada uma delas contendo um documento chamado de ALVO. Esse documento é a referencia para o ranqueamento e para os métodos automáticos sendo é fornecido como documento exemplo. Outros 5 documentos que devem ser ranqueados com base em sua ligação com o Documento Alvo. Os documentos que compunham o corpo das avaliações, 30 artigos ao total, eram editoriais de jornais de circulação nacional. O tamanho médio de cada artigo avaliado era de 24 frases ou, aproximadamente, 602 palavras.

Participaram do experimento 9 alunos do último ano do curso de Comunicação Social da Pontifícia Universidade Católica do Paraná. A ferramenta foi implementada utilizando PHP com a base de dados em MySQL e disponibilizada para acesso através da WEB.

Eles tiveram 2 semanas para realizar as avaliações.

O avaliador deveria acessar a aplicação e então seriam apresentadas as avaliações que ele deveria fazer, conforme Figura 4.1.

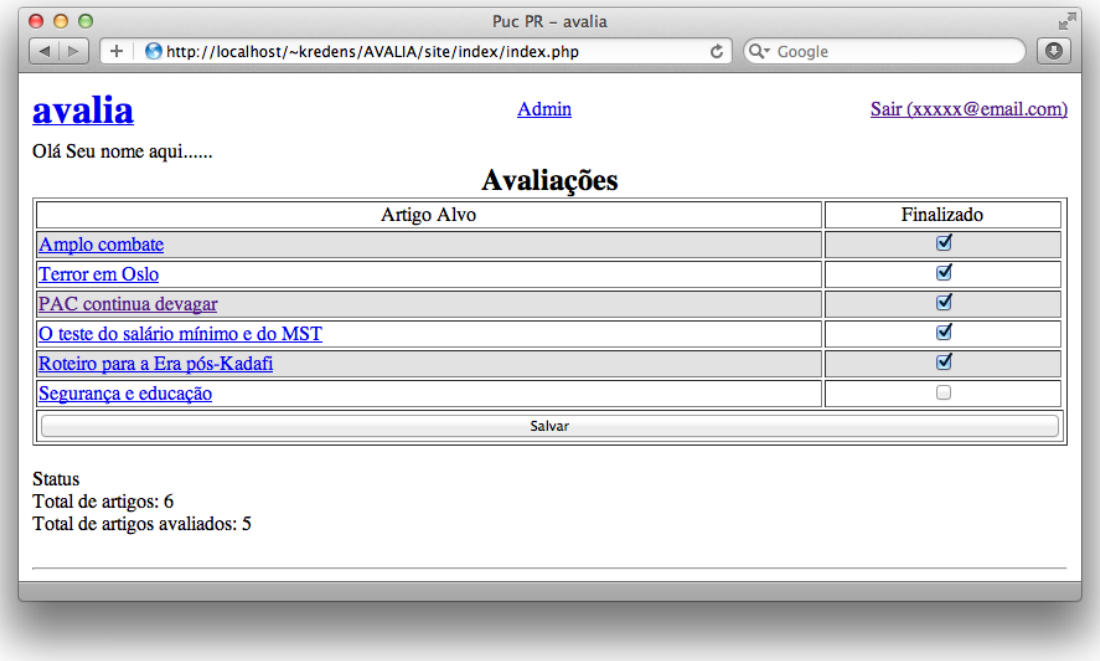

Figura 4.1: Página inicial apresentando avaliações.

Foi implementado comportamento na ferramenta para que ela apresenta-se, por especialista, somente 1 avaliação inacabada por vez. Quando o especialista marcava a avaliação como finalizada, então, a ferramenta apresentava a próxima, sempre escolhendo de maneira aleatória. Tal comportamento foi implementado com o intuito de dificultar que um especialista pude-se ter acesso à avaliação de outro especialista. Ao acessar a avaliação pendente, eram apresentados os 5 artigos que deveriam ser ranqueados. O preenchimento do ranking era feito manualmente e deveria ser informado um valor entre 1 e 100, conforme Figura 4.2. O valor informado não servia como peso, somente para ordenar os documentos. No momento da ordenação final esses valores eram normalizados entre 1 e 5. Um documento poderia receber valor zero (0) para seu ranking. Isso poderia ser feito com o intuito de informar que tal documento não tinha ligação nenhuma com o Alvo.

Ao clicar em um artigo o avaliador tinha acesso ao texto do artigo e esse momento o avaliador deveria marcar as frases mais relevantes, como exemplifica a Figura 4.3

A marcação foi dividia em 3 tipos:

- **Tópico Frasal**, poderia ser marcada somente uma por texto;
- **Frase Importante**, poderiam ser marcadas quantas fossem identificadas;

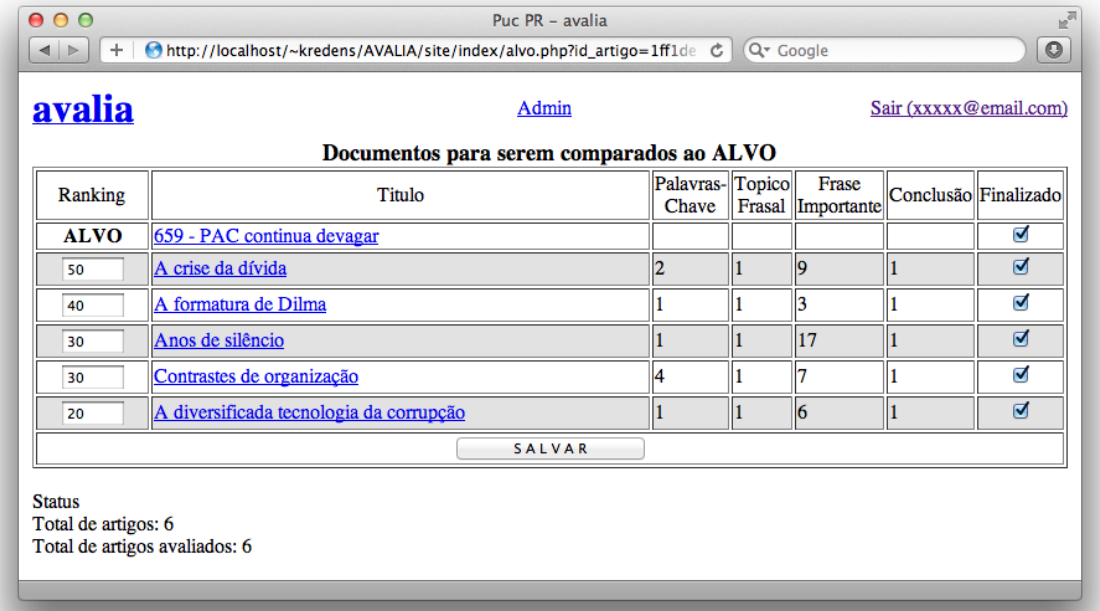

Figura 4.2: Exemplo do preenchimento do ranking.

- **Conclusão**, poderia ser marcada somente uma por texto.

O usuário deveria selecionar o tipo e então clicar na frase que desejava marcar. Para desmarcar ele deveria clicar na frase já marcada. Não era permitido a marcação concorrente, uma frase poderia ser marcada somente com 1 determinado tipo, de acordo com a Figura 4.3. Essa categorização das frases foi criada de modo a induzir os avaliadores a serem mais criteriosos em suas marcações.

Ao término do documento ele o marcava como finalizado e ao termino da avaliação ele à marcava como finalizada e o sistema apresentava a próxima à ser realizada.

4.3 Resultados Obtidos

Os resultados obtidos e utilizados por esse trabalho podem ser divididos em 2 grupos distintos:

- Ranking dos Artigos, total de 6 rankings, cada qual com um documento alvo distinto;

- Distribuição das Frases Marcadas, agrupadas para se obter a memória de cálculo.

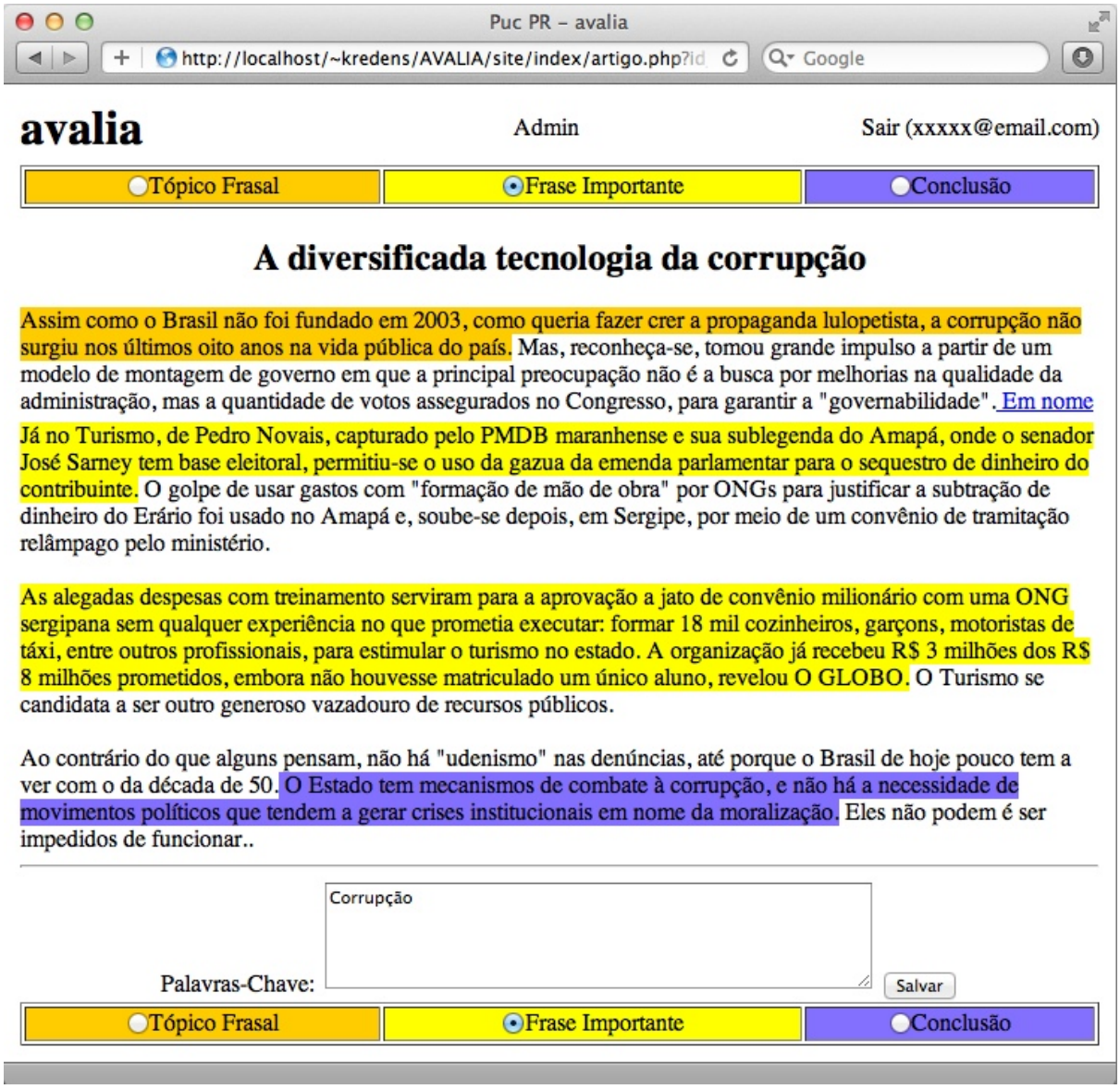

Figura 4.3: Exemplo da marcação das frases importantes.

4.3.1 Ranking dos Artigos

Como descrito na seção anterior, 4.2, ao término dos trabalhos, cada avaliador leu e ranqueou 30 artigos, divididos em 6 avaliações, cada qual contendo 5 artigos. Na Tabela 4.1, tem-se o resultado das avaliações realizadas.

Na primeira coluna está a posição final, no ranking de cada documento, subsequente ao código do artigo. Seus títulos podem ser consultados na tabela 7.1, do Capítulo 7. As próximas colunas, duplas, representam a ordenação dada por cada avaliador. Por exemplo, na Avaliação 1, o avaliador 50, informou os pesos da coluna à esquerda, 50, 15, 30, 40 e 3, no exemplo com fundo cinza escuro Assim como todos os outros avaliadores o fizeram. Esses pesos foram normalizados e estão dispostos nas colunas à direita, no exemplo em cinza claro, abaixo de cada avaliador. Quanto maior o número, melhor, ou

Avaliação 1, artigo alvo 41																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
1^o	17	50	5	60	5	5	5	5	5	50	3	4	5	1	1	100	5	9	2	36	4
4^o	20	15	2	1	2	3	3	1	1	30	2	2	4	7	3	20	3	50	3	23	2,55556
2^o	35	30	3	50	3	4	4	3	3	100	5	4	5	3	2	100	5	96	5	36	4
3^o	36	40	4	10	4	2	2	4	4	90	4	1	3	9	4	30	4	3	1	29	3,22222
5^o	40	3	1	0	1	1	1	2	2	10	1	1	3	11	5	30	4	88	4	22	2,44444

Avaliação 2, artigo alvo 44																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
3^o	23	50	5	5	1	3	3	1	1	90	4	1	1	20	5	50	3	10	2	25	2,77778
4^o	26	30	3	80	4	2	2	4	4	50	2	1	1	3	3	40	2	66	4	25	2,77778
5^o	41	10	1	0	0	1	1	5	5	30	1	1	1	15	4	10	1	3	1	15	1,66667
1^o	43	45	4	50	3	5	5	3	3	80	3	3	2	2	2	60	4	52	3	29	3,22222
2^o	45	20	2	20	2	4	4	2	2	100	5	4	3	1	1	60	4	75	5	28	3,11111

Avaliação 3, artigo alvo 24																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
5^o	22	40	4	20	4	1	1	1	1	10	1	3	4	1	2	35	4	22	2	23	2,55556
1^o	29	30	3	50	3	5	5	5	5	30	3	3	4	9	5	10	2	66	4	36	4
2^o	31	50	5	0	5	3	3	4	4	100	5	4	5	1	2	30	4	53	3	31	3,44444
4^o	32	30	3	20	3	2	2	3	3	20	2	3	4	5	4	40	5	14	1	28	3,11111
3^o	39	20	2	10	2	4	4	2	2	90	4	4	5	2	3	20	3	77	5	31	3,44444

Avaliação 4, artigo alvo 30																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
3^o	19	3	3	0	3	1	1	2	2	80	5	3	5	9	3	3	4	79	5	28	3,11111
2^o	24	1	1	0	1	4	4	4	4	40	3	2	4	12	5	5	5	45	4	30	3,33333
4^o	34	2	2	0	2	3	3	3	3	50	4	3	5	11	4	3	4	12	2	27	3
5^o	38	4	4	0	4	2	2	1	1	10	2	2	4	7	1	5	5	26	3	22	2,44444
1^o	42	20	5	20	5	5	5	5	5	5	1	1	3	8	2	1	3	3	1	30	3,33333

Avaliação 5, artigo alvo 42																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
5^o	18	10	3	0	3	2	2	2	2	50	2	1	3			20	4	66	4	20	2,5
1^o	28	50	5	90	5	5	5	3	3	70	3	4	5			30	5	99	5	36	4,5
2^o	30	30	4	50	4	3	3	1	1	90	5	3	4			1	1	54	3	25	3,125
3^o	44	10	3	0	3	4	4	4	4	80	4	1	3			2	2	1	2	22	2,75
4^o	46	10	3	20	3	1	1	5	5	30	1	3	4			3	3	10	1	21	2,625

Avaliação 6, artigo alvo 46																					
		Avaliador - Ordenação / Ordenação Ajustada																			
Artigo		50		51		52		53		54		56		57		58		61		Σ	Média
4^o	25	10	2	0	2	1	1	5	5	90	4	1	3	0	0	20	3	30	3	21	2,33333
5^o	33	10	2	0	2	2	2	3	3	40	3	1	3	1	3	30	4	21	1	21	2,33333
3^o	37	20	3	0	3	4	4	4	4	100	5	1	3	2	4	20	3	23	2	28	3,11111
1^o	47	40	4	90	4	5	5	2	2	20	1	4	5	4	5	30	4	52	4	35	3,88889
2^o	48	50	5	80	5	3	3	1	1	30	2	2	4	4	5	40	5	39	5	34	3,77778

Tabela 4.1: Resultado consolidado das avaliação.

seja, artigos com ordem 5 são os mais próximos ao alvo.

Os dados das 6 avaliações serão utilizados para comparar a performance da solução proposta, quanto a sua capacidade de ordenar os documentos com relação ao documento alvo.

4.3.2 Distribuição das Frases Marcadas

Com base nas marcações, do tópico frasal, frases importantes e conclusão, feitas pelos avaliadores, espera-se poder criar uma métrica capaz de discriminar a importância das regiões de um texto e com isso poder valorar a importância de uma MFS de acordo com a posição que ela aparece no documento. Para chegar-se à essa métrica, foram executados os seguintes passos.

Primeiro, foram extraídas, por artigo avaliador a quantidade de vezes que cada frase foi marcada, com o intuito de se normalizar o tamanho de cada artigo. Exemplo pode ser conferido na tabela 4.2 que apresenta esses valores para o artigo 31, que possui 28 frases.

Posição Inicial da Frase	Quantidade de Marcações	Posição Normalizada
0	0	0
1	8	3.5714
2	7	7.1429
3	3	10.7143
4	1	14.2857
5	3	17.8571
6	0	21.4286
7	2	25.0
8	2	28.5714
9	0	32.1429
10	6	35.7143
11	0	39.2857
12	5	42.8571
13	1	46.4286
14	4	50.0
15	5	53.5714
16	1	57.1429
17	2	60.7143
18	1	64.2857
19	0	67.8571
20	5	71.4286
21	2	75.0
22	1	78.5714
23	0	82.1429
24	4	85.7143
25	1	89.2857
26	0	92.8571
27	5	96.4286
28	7	100.0

Tabela 4.2: Normalização da posição das frases marcadas de acordo com o tamanho do artigo.

Como pode-se observar na tabela 4.2, estão as marcações acumuladas de todos os avaliadores. Por exemplo, a frase na posição normalizada 3,5741 recebeu 8 marcações, ou seja, quase todos os avaliadores a consideraram importante. Na segunda coluna está quantas vezes a frase foi marcada por algum avaliador. E a última coluna mostra a posição da frase normalizada. A normalização foi realizada utilizando a fórmula (4.1).

$$posicaoNormalizada = \frac{100}{(Qtd.de frasesdoArtigo) \times (Fraseatual)} \tag{4.1}$$

O segundo passo foi realizar o mesmo processo realizado no documento 31, Tabela 4.2,

em todos os outros 29 documentos que compunham a base de avaliação. A quantidade de marcações foi acumulada por posição normalizada, tirando-se a média por avaliação.

Criou-se uma estrutura hipotética de um documento com frases iniciando em 0, 1, 5, (... incrementendo de 5 em 5 ...), até 100. E então os dados das marcaçãos agrupadas de cada avaliação foram interpolados nessas posições com o intuito de chegar-se ao peso dessas respectivas posições. O resultado por ser conferido na Tabela 4.3.

	Avaliação								
Posição	24	30	41	42	44	46	Peso	Intervalo de Confiança Inferior	Intervalo de Confiança Superior
0	0	0	0	0	0	0	0	0	0
1	2,458	2,897	3,117	1,737	2,534	1,996	2,457	2,238	2,675
5	4,816	5,039	10,391	5,768	8,198	6,362	6,762	5,864	7,661
10	4,082	6,097	4,143	4,658	4,709	4,935	4,771	4,465	5,076
15	2,477	3,692	5,756	3,973	4,798	4,373	4,178	3,717	4,639
20	1,753	2,590	4,530	2,659	3,039	4,595	3,194	2,718	3,671
25	2,935	3,477	5,208	2,332	2,904	1,924	3,130	2,649	3,611
30	3,140	2,372	1,543	2,723	4,117	2,646	2,757	2,401	3,113
35	1,697	1,667	4,343	1,742	3,928	4,179	2,926	2,363	3,489
40	2,017	2,226	2,760	1,945	2,786	3,317	2,508	2,285	2,732
45	2,654	1,749	4,039	2,828	3,688	3,252	3,035	2,694	3,376
50	2,529	2,243	2,519	2,537	2,551	2,732	2,519	2,453	2,584
55	1,948	0,559	1,422	2,228	2,108	2,481	1,791	1,498	2,083
60	2,400	1,062	1,679	1,422	3,780	3,151	2,249	1,807	2,691
65	2,493	2,051	2,085	1,845	1,893	4,202	2,428	2,053	2,803
70	2,533	1,286	1,775	2,600	2,069	3,162	2,237	1,959	2,516
75	2,175	2,027	2,744	1,233	2,108	0,910	1,866	1,585	2,147
80	2,214	1,955	1,459	1,584	2,844	2,597	2,109	1,879	2,338
85	1,722	1,631	1,206	2,061	3,201	2,675	2,083	1,775	2,390
90	1,263	3,376	3,675	1,687	2,783	1,422	2,368	1,930	2,805
95	3,444	3,785	4,262	3,706	6,887	5,6199	4,617	4,051	5,183
100	4,8	3,4	7,4	5	5,4	7,4	5,566	4,909	6,223

Tabela 4.3: Distribuição da marcação, feita pelos avaliadores, dos pesos por região do texto.

Referente a Tabela 4.3, na primeira coluna estão dispostas as regiões dos artigos, subdivididas de 5 em 5. As 6 colunas subsequentes, é a média acumulada das marcações feitas pelos avaliadores em cada avaliação, lembrando que cada avaliação compreende 6 artigos, 1 alvo e outros 5 à serem comparados. Após, tem-se a coluna Peso, que representa a média das 6 avaliações por região. E, por último, 2 colunas representando o Intervalo de Confiaça Inferior e Superior em 95%, calculados de acordo com [Mag09]. Pode-se verificar na Figura 4.4, o gráfico apresentando o peso por região.

Pode-se verificar que os avaliadores consideram como região mais importante o início

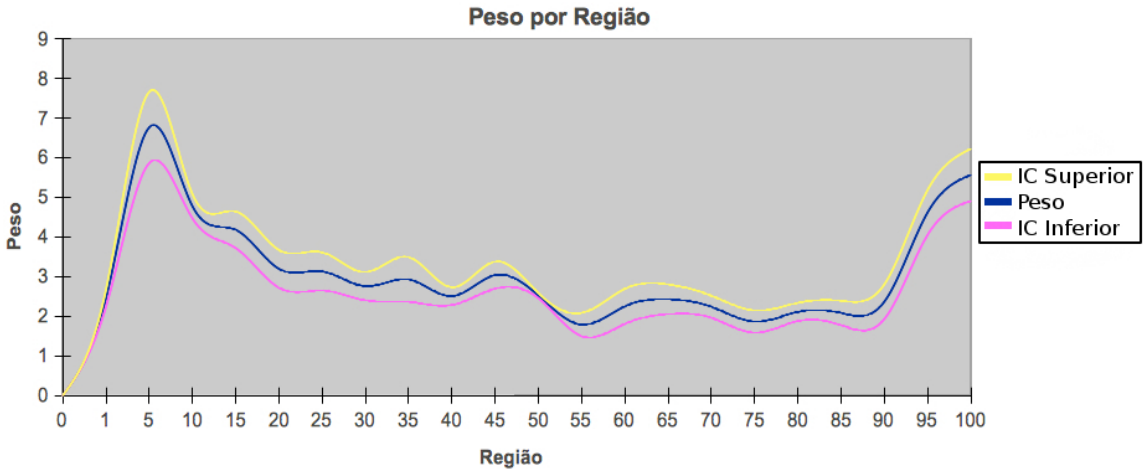

Figura 4.4: Exemplo da marcação das frases importantes.

e o término dos textos, respectivamente a introdução e a conclusão.

Os valores apresentados na Tabela 4.3 foram armazenados de acordo com a Tabela 4.4 para uso futuro. A coluna Peso da Região é média das marcações realizadas. O objetivo é utilizar essa informação no processo de cálculo do peso das MFS durante a indexação. Uma MFS que disposta no inicio do texto, entre a região 1 e 10 é mais importante do que sequências dispostas, por exemplo, no meio do texto.

Região Ajustada (X)	Peso da Região (Y)
0	0
1	2.4570146192
5	6.7627919381
10	4.7710056018
15	4.178767641
20	3.1949156871
25	3.1305683437
30	2.7573819477
35	2.9262474949
40	2.5089367316
45	3.0356339288
50	2.5191156614
55	1.7912519528
60	2.2494794266
65	2.4285244428
70	2.2378331596
75	1.8666858155
80	2.1091149327
85	2.0833552605
90	2.3681650051
95	4.6177048518
100	5.5666666667

Tabela 4.4: Valores obtidos com base na média das marcações.

4.4 Considerações Finais

O Ranking dos Artigos foi armazenado para efetuar a comparação da efetividade entre o método proposto e a abordagem clássica, com uso de palavras simples. Os dados coletados, referentes a Distribuição das Frases Marcadas, após terem sido normalizadas, acumuladas e ter seus pontos interpolados inseridos no algorítimo implementado que avalia uma sequência de acordo com sua região no texto.

A comparação entre os métodos e o uso a descrição do algoritmo que faz a avaliação de acordo com a região no texto serão discutidos no próximo capítulo.

Capítulo 5

Ambiente de Testes, Experimentos e Resultados Encontrados

5.1 Considerações Iniciais

Com base na teoria e no método proposto, já apresentados, neste capítulo será apresentado o ambiente de testes criados, no qual as aplicações de testes foram implementadas para realizar os testes necessários á validação da eficácia do método proposto. Também serão apresentados Juntamente com os experimentos realizados e os resultados encontrados.

5.2 Ambiente de Testes

Foram implementadas duas aplicações distintas. Na primeira, Aplicação de Testes 1, implementou-se o método proposto no Capítulo 3 e na segunda, Aplicação de Testes 2, para efeito de comparação dos resultados, foi implementado mecanismo de indexação e busca utilizando palavras simples como tokens. Ambas aplicações foram implementadas utilizando a linguagem de programação Java, com a uso da IDE Eclipse. Os modelos de dados foram fisicalizados utilizando SGBD MySQL.

Não foi alvo deste trabalho avaliar a performance do método proposto, dessa maneira, cada implementação focou facilitar os experimentos e a depuração do código levando então á uma implemtação extremamente modularizada, com implementações atômicas, separando cada passo do processo e sem reaproveitamento de dados entre os passos.

A seguir será descrita a implementação de cada aplicação. Para ambas aplicações a entrada é um documento, em formato TXT, PDF ou HTML e a saída é uma lista ordenada, dos documentos que compõe o indice, de acordo com seu relacionamento ao documento ALVO. Cada aplicação é composta por 2 programas. O primeiro utilizado para indexar documentos e a segundo que recebe o documento ALVO, compara-o com os

documentos indexados e retorna a lista ordenada.

5.2.1 Aplicação de Testes 1

Para atender as necessidades da Aplicação de Testes 1, tanto a indexação quanto busca, foi criado e fisicalizado o modelo de dados apresentado na Figura 5.1. Ele será detalhado ao longo desta seção, a cada módulo da aplicação.

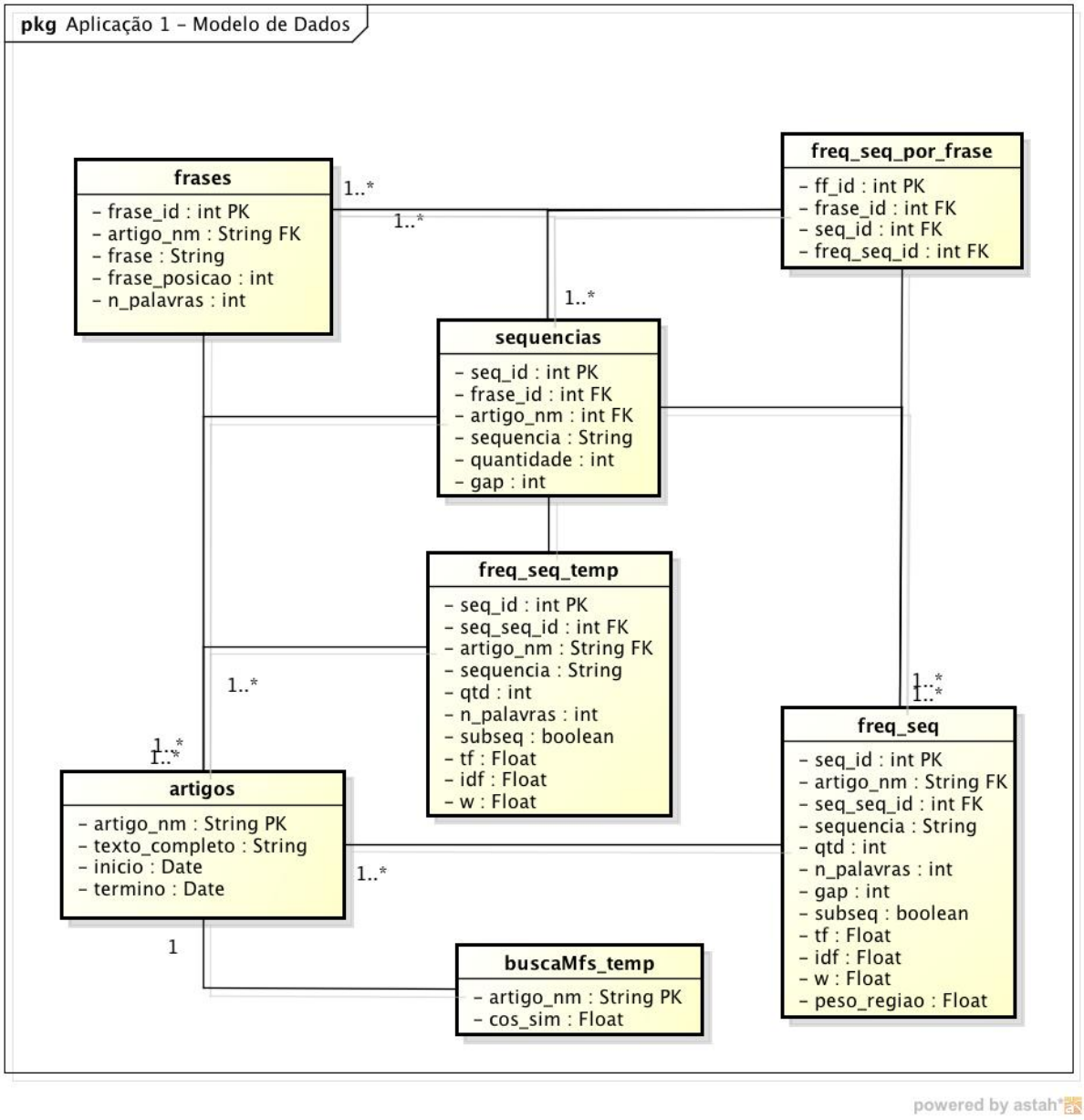

Figura 5.1: Aplicação de Testes 1 - Modelo de Dados implementado.

Indexação

O processo de indexação foi implementado de acordo com diagrama de atividades apresentado na Figura 5.2. A indexação é executada tendo como alvo somente um único artigo por vez.

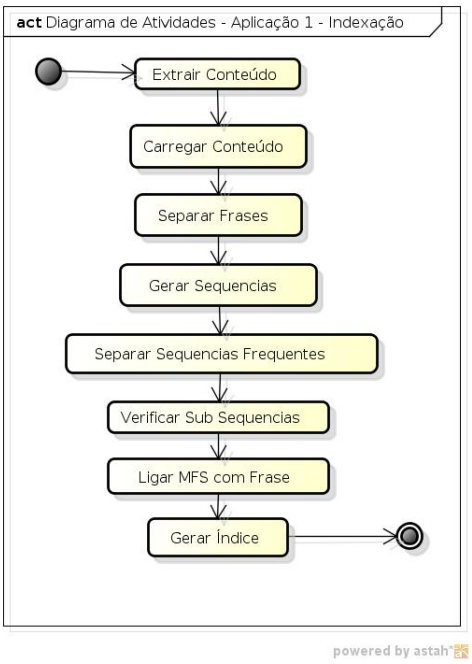

Figura 5.2: Aplicação de Testes 1 - Diagrama de Atividades - Indexação

Com base na teoria apresentada no Capítulo 2 Sessão 2.3, no processo de indexação, os parâmetros abaixo podem ser configurados:

- **Tamanho Máximo da Sequência**, maior tamanho que uma sequência gerada pode ter.
- **Tamanho Mínimo da Sequência**, menor tamanho que uma sequência gerada pode ter.
- **Tamanho Máximo da Janela**, quantidade máxima de palavras que podem distanciar outras duas palavras de uma sequência gerada.
- **Valor considerado Frequente**, quantidade mínima de vezes que uma sequência deve aparecer no texto para ser considerada frequente.
- **Peso pelo Tamanho da Sequência**, opção que pode ser ligada ou desligada. Quando ligada, é utilizado na fórmula alterada do *TFiDF* valor que aumenta o peso da sequência de acordo com seu tamanho, sendo assim, quanto maior a sequência, maior será seu peso.

- **Fator de Incremento pelo tamanho Sequência**, é o valor utilizado na fórmula para incrementar o peso de uma sequência de acordo com seu tamanho.
- **Peso pela Região** opção que pode ser ligada ou desligada. Quando ligada ao cálculo do peso final da sequência será acrescido peso de acordo com a região que a sequência aparecesse.

Os parâmetros Peso pelo **Tamanho da Sequência** e **Peso pela Região no texto** foram incluídos para que se possa comparar a efetividade de cada um deles de maneira isolada em cada experimento realizado.

Dentre as atividades implementadas, as **Gerar Sequências**, **Separar Sequências Frequentes** e **Verificar Subsequencias** são a implementação do algoritmo no Capítulo 2 na Figura 2.4.

1. Extrair Conteúdo Nessa atividade o conteúdo do documento é extraído de acordo com o formato do arquivo fornecido, como segue:

- **TXT**, o conteúdo é extraído sem nenhuma alteração;
- **PDF**, é extraído somente o texto do artigo, utilizando biclioteca chamada PDFBox;
- **HTML**, as tags HTML são removidas e então somente o texto é extraído.

O conteúdo extraído é então carregado para dentro de um arquivo TXT, que fica armazenado no sistema de arquivos. Esse passo foi criado para que o processo de extração não precise ser repetido mais de uma vez para o mesmo artigo, dessa maneira, a aplicação verifica se, pelo nome do documento, o mesmo já teve seu conteúdo extraído. Esse conteúdo extraído não é, já no inicio, carregado no banco e sim em um arquivo, pois como eram previstos vários testes e cada teste possuiria a sua própria base de dados, a cada inicio de um novo teste somente os arquivos TXT precisavam ser replicados para a pasta do novo teste.

2. Carregar Conteúdo É nessa atividade que o conteúdo, já em formato TXT, é carregado para dentro do banco de dados. Em alguns testes verificou-se que ao processar a parte de referencias bibliográficas em artigos científicos acabava-se gerando sequencias não importantes por conta das abreviações utilizadas. Desse modo foi criada opção, que poderia ser ligada ou desligada a cada nova indexação, para que fosse desconsiderado todo o texto que estivesse após texto que iniciava a sessão de bibliografia. De acordo com universo de artigos testados, foram utilizadas as seguintes *strings* para procurar o inicio da sessão de bibliografia: “References”e “Bibliografia”.

O conteúdo do documento, antes de ser carregado passava por um pré-processamento onde todas as letras eram convertidas para formato em minúsculo, eram removidos caracteres especiais, as vogais com acento eram substituídas pelas mesmas sem acento e os símbolos: ";", ":", "?"e "!"substituídos por ".".

E então o texto tratado era inserido, como um novo registro, na tabela *artigos* populando a coluna *texto_completo*. Referente a essa mesma tabela, as colunas *inicio* e *termino* eram populadas com a data/hora do inicio e do termino do processamento. Essa informação era utilizada para identificar processos travados.

3. Separar Frases O texto carregado era recuperado do banco e então passava por um processo que o quebrava em frases a cada ocorrência do, ponto final("."). As frases eram inseridas em um vetor onde cada uma passava por um novo processamento, de acordo com a configuração do processo em execução. O processo de Separar Frases recebia como entrada, dos parâmetros de configuração, qual era o idioma do texto sendo processado. Foram feitas implementações para o idioma Português do Brasil e o Inglês. O processamento feito em cada frase compreendia em remover as *stopwords* e fazer o *stemming* das palavras.

Para as remoção das *stopwords*, tanto para o idioma Português quanto para o Inglês, foram utilizadas listas retiradas do mecanismo de busca chamadao Lucene [1].

Após a remoção das *stopwords* cada palavra de cada frase era submetida ao algoritmo de *stemming*, também, de acordo com a língua de cada texto. Para o Português utilizou-se a biblioteca PtStemmer [2] e para o Inglês usou-se o algoritmo Porter [Por80] [3].

Cada frase tratada era então inserida, como um novo registro na tabela *frases*, gerando para si uma chave nova *frase_id*. A coluna *frase_posicao* era preenchida com a posição da frase em relação ao texto, que era contado iniciando em 1 e a coluna *n_palavras* recebia a quantidade de palavras que compunham a frase após o processamento.

4. Gerar Sequências As frases são recuperadas do banco e então, frase por frase, são geradas todas as sequencias possíveis, combinando as palavras que as compõem, respeitando a ordem em que aparecem.

Todas as sequências geradas são armazenadas na tabela *sequencias*, juntamente com a quantidade que a sequencia aparece no documento e o seu *gap*, que é controlado pelo parâmetro **Tamanho Máximo da Janela**. O *gap* é um valor numérico, que pode variar de 0 a n, onde n é o tamanho máximo permitido, que é a quantidade de palavras, que foram omitidas, e que estavam entre as palavras utilizadas para montar a sequência em questão.

[1] http://lucene.apache.org/core/
[2] http://code.google.com/p/ptstemmer/
[3] http://tartarus.org/martin/PorterStemmer/java.txt

5. Separar Sequências Frequentes As sequências geradas são recuperadas da tabela *sequencias*, contadas e são, então, copiadas para a tabela *freq_seq*, somente as que obtiverem contagem superior ao parâmetro **Valor considerado Frequente**.

O seguinte comando SQL faz a cópia das sequências frequentes:

```
insert into FREQ_SEQ
   select null, SEQUENCIA, SEQ_ID, count(*), false,
          0,  ‘ + Artigo sendo analisado + ’ , 0, 0, 0, gap, 0
     from SEQUENCIAS
    where ARTIGO_NM = ‘ + Artigo sendo analisado + ’
 group by SEQUENCIA
   having count(distinct FRASE_ID) >= Valor Considerado Frequente;
```

Esse comando irá inserir o resultado do *select* que tem como objetivo recuperar as sequências, agrupando as pelo próprio valor da sequência onde a quantidade de aparições seja superior ao parâmetro **Valor Considerado Frequente**

Esse passo foi criado com o objetivo de deixar a execução da atividade **Gerar Índice** mais rápida. Como comentado anteriormente, não é objetivo avaliar performance das soluções, mas em testes verificou-se que a separação das sequências frequentes, em uma nova tabela, aumentava a performance da aplicação, pois ele, ao invés de percorrer todas as sequências e então ter que verificar quais são as frequentes, passa a ler uma tabela que já contem, separadas, as sequências frequentes. Mesmo não sendo objetivo avaliar a performance, muitos testes precisaram ser feitos então performance foi alvo de atenção tentando-se melhorar(diminuir) o tempo de processamento.

6. Verificar Subsequências São recuperadas as sequências frequentes da tabela *freq_seq*, somente as que são menores que o parâmetro **Tamanho Máximo da Sequência**, uma vez que as sequências que tiverem tamanho igual ao tamanho máximo não poderão ser subsequência de nenhuma outra. Cada sequência recuperada tem os espaços em branco substituidos pelo caracter "%", que é o caracter coringa para recuperação de textos no *MySQL*. Esses caracteres são inseridos, também, no ínicio e no término de cada sequencia, por exemplo:

Sequência original: “salari minim”
Sequência alterada: “%salari%minim%”

A sequência alterada é utilizada no comando SQL:

```
select count(*)
  from FREQ_SEQ
 where SEQUENCIA like ' + Sequência Alterada + '
   and SUBSEQ is false
   and SEQ_ID <> ' + ID da Sequência sendo analisada + '
   and ARTIGO_NM = ' + Artigo sendo analisado + ';
```

O objetivo desse comando é verificar se existe mais alguma sequência, que já não seja uma subsequência, que possua as mesmas palavras que a sequência sendo analisada. A ordem das palavras é respeitada e é descartada a sequência que está sendo analisada. Caso o retorno seja igual a 0 então a sequência não é uma subsequência. Nos casos onde o retorno seja maior que 0 as sequências tem o valor da coluna *freq_seq.SUBSEQ* alterado para *true* sendo então descartadas nos próximos passos.

A partir desse ponto, todas as sequências armazenadas na tabela *freq_seq* e que possuam a coluna *subseq* = *false* são consideradas MFS.

7. Ligar MFS com Frase Por conta de um dos objetivos do método proposto onde tem-se a intenção de valorar as MFS compartilhadas entre os documentos de acordo com a sua região no texto do documento, esse passo identifica e relaciona MFS às suas frases de origem. Essa informação será utilizada na próxima atividade, onde o índice final será gerado. Essa informação é armazenada na tabela *freq_seq_por_frase*.

8. Gerar Índice Esse é a única atividade que é executada levando-se em conta o artigo sendo processado e todos os outros já indexados. Como o objetivo do trabalho é avaliar a melhor combinação entre as propostas, o cálculo do TFiDF foi implementado passo a passo onde o resultado de cada passo é armazenado separadamento para então ser combinado no final processo.

São recuperadas todas as sequências armazenadas na tabela *freq_seq*, onde a coluna *subseq* seja *falso*, e então seu peso é calculado. Primeiramente são calculados os valores básicos a serem utilizados na fórmula TFiDF, que são:

- **Total de Aparições**, total de aparições da sequência sendo analisada, levando-se em conta todos os documentos *indexados*, utilizando o comando:

```
select sum(QTD) as totalAparicoes
  from FREQ_SEQ
 where SUBSEQ is false
   and SEQUENCA = ' + Sequência sendo analisada  + ';
```

- **Total de Documentos**, total de documentos que compõem o índice, utilizando o comando:

```
select count(ARTIGO_NM) as totalDocumentos
  from ARTIGOS;
```

- **Total de Documentos que contém a sequência**, total de documentos, dentre os que compõem o índice, que possuem a sequência sendo analisada, utilizando o comando:

```
select count(ARTIGO_NM) as totalDocumentosContemSequencia
  from FREQ_SEQ
 where SUBSEQ is false
   and SEQUENCIA = ‘ + Sequência sendo analisada + ’ ;
```

- **iDF (Frequência Inversa nos Documentos**, frequência invertida da sequência sendo avaliada em relação aos documentos do índice, calculada de acordo com a fórmula (5.1).

$$iDF = log_{10}(\frac{totalDocumentos}{totalDocumentosContemSequencia}) \tag{5.1}$$

Após, é criado uma lista composta por todos os documentos indexados detentores da sequência sendo analisada para então, por documento, ser calculado o TF(term frequency), de acordo com o parâmetro da configuração **Peso pelo Tamanho da Sequência**. A lista é criada pelo seguinte comando:

```
select ARTIGO_NM, QTD, N_PALAVRAS
  from FREQ_SEQ
where SUBSEQ is false
  and SEQUENCIA = ‘ + Sequência sendo avaliada + ’;
```

Peso pelo Tamanho da Sequência - DESLIGADO, fórmula (5.2)

$$TF = (\frac{qtd}{maiorFrequenciaMfs}) \tag{5.2}$$

Onde, *qtd* é a quantidade de aparições da sequência sendo avaliada no documento da vez.

Peso pelo Tamanho da Sequência - LIGADO
Foi implementada a fórmula apresentada no Capítulo 3 Sessão 3.3. Primeiro recupera-se o tamanho da maior MFS contida no documento sendo avaliado. Com o comando:

```
select max(N_PALAVRAS) as tamanhoMaiorSequencia
  from FREQ_SEQ
 where SUBSEQ is false
   and ARTIGO_NM = ' + Documento sendo avaliado + ';
```

O tamanho da maior sequência é então inserido na fórmula (5.3).

$$TF = (qtd * totalAparicoes) * (\frac{tamanhoSequenciaSendoAvaliada^{FatordeIncremento}}{tamanhoMaiorSequencia^{FatordeIncremento}}) \tag{5.3}$$

Onde, *qtd* é a quantidade de aparições da sequência sendo avaliada no documento da vez.

E então o peso final é calculado, de acordo com fórmula (5.4).

$$w = TF \times iDF \tag{5.4}$$

Pode-se verificar na Tabela 5.1 exemplo dos valores encontrados durante os cálculos realizados para encontrar-se o peso de algumas sequências. Nessa mesma pode-se verificar como o cálculo que considera o tamanho da sequência valora com maior peso as sequências maiores. Enquanto o cálculo sem esse critério, além de não valorizar essas sequências em algumas situações as coloca num mesmo nível de peso.

Tabela 5.1: Comparação entre os pesos gerados com e sem valoração por tamanho da sequência.

Artigo	Sequencia	Qtd	n_palavras	iDF	TF [1]	w [1]	TF [2]	w [2]
41	que pod	2	2	0.301	5.333	8.783	0.333	0.100
41	uma civil	2	2	0.778	1.777	7.568	1	0.778
44	posica pel ministr	2	3	0.778	4	14.348	1	0.778
23	agi consorci venc	2	3	0.778	1.44	5.014	1	0.778
26	que govern petist enfrent	2	4	0.778	4	11.375	1	0.778
23	aeroport sao goncal amar natal	2	5	0.778	4	17.950	1	0.778

[1] Avaliação considerando tamanho da sequência
[2] Avaliação sem considerar tamanho da sequência

No passo subsequente é, caso o parâmetro **Peso pela Região** esteja LIGADO, deve-se calcular valor à ser incrementado ao peso final de acordo com a região em que a sequência se encontra dispersa no documento.

Recupera-se a posição ajustada das frases, no documento em questão, com o seguinte comando:

```
select F.FRASE_ID, ARTIGO_NM,
       ((select (100/count(*))
           from FRASES
          where ARTIGO_NM = F.ARTIGO_NM)
        + F.FRASE_POSICAO) as posicao_ajustada
  from FREQ_SEQ_POR_FRASE as fs, FRASES as f
 where F.FRASE_ID = FS.FRASE_ID
   and FS.FREQ_SEQ_ID = Id da Sequência sendo avaliada;
```

Com base na posição ajustada recuperada de cada frase é então interpolado o valor do seu peso na região do texto. Esse processo é realizado a partir dos dados coletados e apresentados no Capítulo 4, na Tabela 4.4. Utilizou-se a biblioteca [4] para efetuar cálculo da interpolação. Nela são inseridos os dados conhecidos, as colunas X e Y e então é acionada informando o X conhecido (posição ajustada onde a sequência ocorrer) para obter o Y referente a tal posição.

Por exemplo, a sequência "presid dilm rousseff", do documento 43, foi avaliada de acordo com a Tabela 5.2.

Tabela 5.2: Resultado da avaliação da sequência "presid dilm rousseff".

Artigo_nm	Sequencia	n_palavras	Posição	Posição Ajustada	Peso da Região	Peso Final
43	presid dilm rousseff	3	6	10.54	4.592	4.349
			11	15.54	4.105	

O peso final é então multiplicado ao peso w encontrado anteriormente, obtendo assim o peso final que será utilizado no cálculo da similaridade entre documentos. O processo de indexação é repetido, sequência por sequência até que todas possuam valores associados de peso.

[4] http://www.ee.ucl.ac.uk/ mflanaga/java/CubicSpline.html

Busca

O processo de Busca também baseia-se no modelo de dados apresentado na Figura 5.1. O indice, à ser utilizado nos cálculos de similaridade e que é composto por todos os documentos indexados está armazenado na tabela *freq_seq*. A arquitetura do processo da Busca está representado na Figura 5.3. Suas atividades serão detalhadas na sequência.

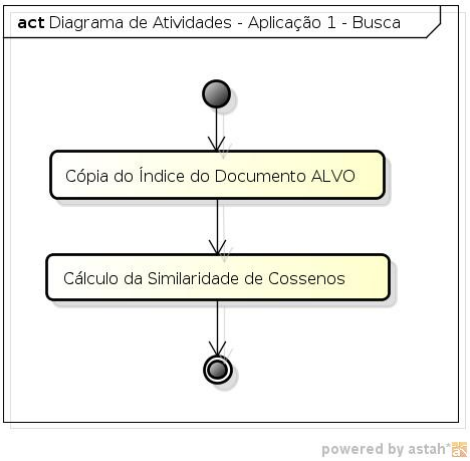

Figura 5.3: Diagrama de Atividades - Aplicação 1 - Busca

A entrada nesse processo é feita pelo usuário, que deve informar um documento, chamado ALVO, para ser comparado aos demais, também indexados, para que seja retornada uma lista ordenada de acordo com a similaridade encontrada entre o ALVO e os demais documentos.

1. Cópia do Índice do Documento Alvo Nesta atividade, os dados referentes ao índice do documento ALVO, contidos na tabela *freq_seq* são copiados para a tabela *freq_seq_temp*. O único objetivo dessa atividade é facilitar o cálculo a ser executado, uma vez que o mesmo foi realizado utilizando-se comando SQL e o fato de tais informações, índice do ALVO e o índice dos demais documentos, estarem em tabelas distintas facilita a operação.

2. Calculo da Similaridade de Cossenos Aqui são recuperados todos os documentos indexados, exceto o ALVO e, para cada documento, é calculada a Similaridade de Cossenos. Os pesos das sequências de cada documento descrevem um ponto em um espaço euclidiano. E esse ponto, relacionado ao ponto 0 descrevem uma reta. O objetivo então é calcular o cosseno do ângulo formado entre as retas descritas pelo documento ALVO e outro determinado documento objetivo da comparação. Quanto mais próximas as retas, mais semelhantes serão, chegando a 1 quando são exatamente iguais, ou seja, uma reta está sobreposta a outra. Conforme explanação feita no Capítulo 2, Sessão

2.2, nesse ponto foi implementada a fórmula da Similaridade de Cossenos, utilizando o seguinte comando SQL:

```
insert into BUSCAMFS_TEMP
 select '+Doc+',
        (( select sum(F1.W * F2.W)
             from FREQ_SEQ F1, FREQ_SEQ_TEMP F2
            where F1.SUBSEQ is false
              and F1.SEQUENCIA = F2.SEQUENCIA
              and F1.ARTIGO_NM in ('+Doc+') )
         / ((select sqrt(sum(pow(W,2)))
               from FREQ_SEQ_TEMP) * (select sqrt(sum(pow(W,2)))
                                        from FREQ_SEQ
                                       where SUBSEQ is false
                                         and ARTIGO_NM in ('+Doc+'))));
```

O comando acima insere na tabela *buscaMfs_Temp* o valor encontrado. A ordenção resultante pode ser obtida com o seguinte comando SQL:

```
  select ARTIGO_NM, COS_SIM
    from BUSCAMFS_TEMP
order by COS_SIM desc;
```

Na Tabela 5.3 pode-se conferir o resultado de uma busca, onde o ALVO é o documento 44.

artigo_nm	cos_sim
41	0.0145475
45	0.0046729
23	0.00301276
26	0.00106507
43	0.000940586

Tabela 5.3: Ordenação obtida em um dos testes realizados.

Como pode-se verificar, no teste em questão, o documento 41 foi o mais próximo. Lembrando que, o processo de Busca independe da maneira como os documentos foram indexados, ou seja, se foi utilizada métrica por tamanho da sequência ou região no texto.

Ele leva em conta somente os pesos w gerados no processo da indexação e armazenados no índice.

5.2.2 Aplicação de Testes 2

A Aplicação de Testes 2 foi criada com o objetivo de se comparar seu resultado, na comparação entre um documento ALVO e outros indexados, em relação aos resultados obtidos e apresentados no Capítulo 4 e a Aplicação de Testes 1, que é o método proposto nesse trabalho. Para sua implementação foi criado o modelo de dados apresentado na Figura 5.4.

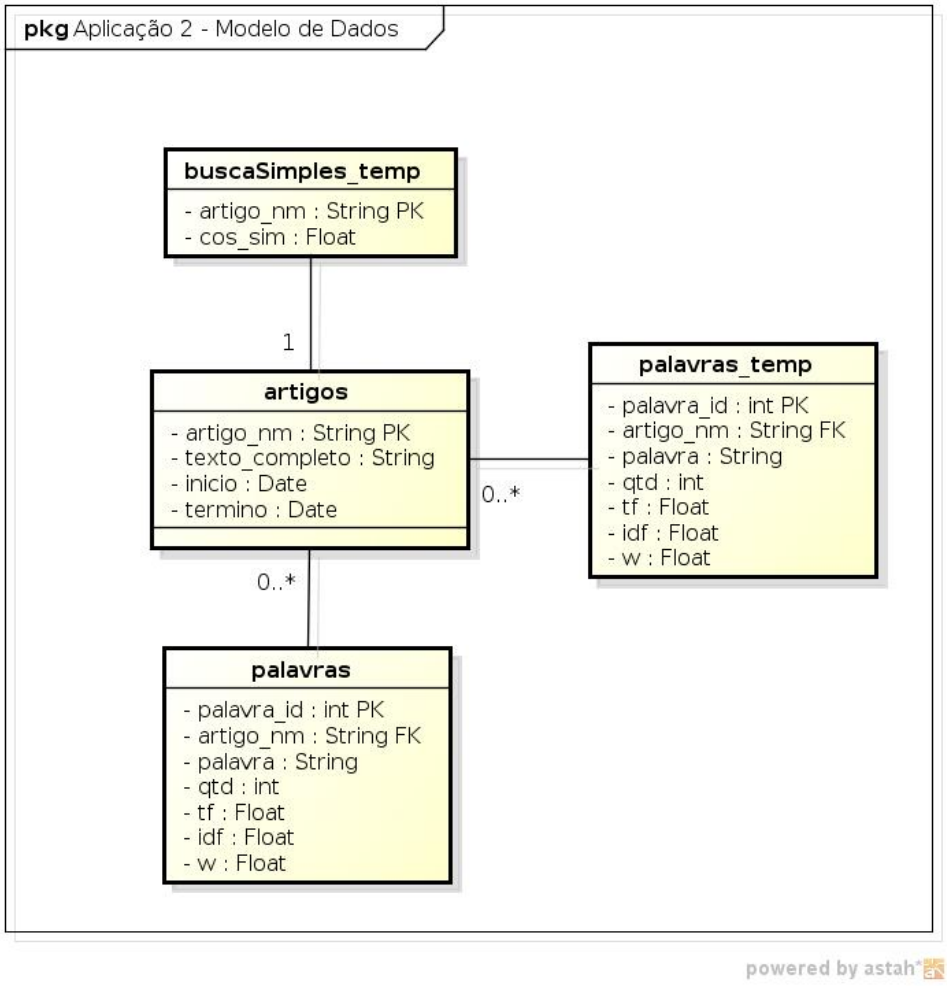

Figura 5.4: Aplicação de Testes 2 - Modelo de Dados implementado.

Indexação

O processo de indexação, assim como para a Aplicação de Testes 1, tem o intuito de extrair, tratar e indexar as informações que representem o conteúdo do documento sendo analisado. Para a Aplicação de Testes 2, esse conteúdo são todas as palavras, exceto

as *stopwords*, contidas no texto. Todas as palavras serão extraídas, armazenadas e seu peso, de acordo com o TFiDF será calculado. As atividas que compõe o processo estão apresentadas na Figura 5.5 e serão descritas a seguir.

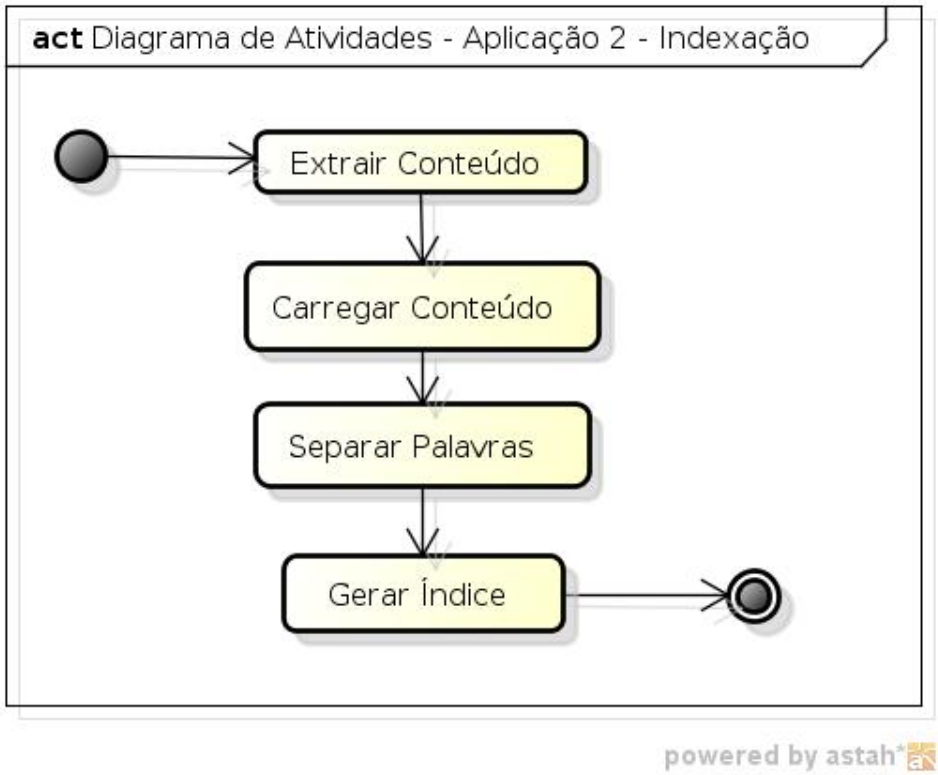

Figura 5.5: Aplicação de Testes 2 - Diagrama de Atividades - Indexação

1. Extrair Conteúdo Seu funcionamento é semelhante a Aplicação de Testes 1. Nessa atividade o conteúdo do documento é extraído de acordo com o formato do arquivo fornecido, como segue:

- **TXT**, o conteúdo é extraído sem nenhuma alteração;
- **PDF**, é extraído somente o texto do artigo, utilizando biclioteca chamada PDFBox;
- **HTML**, as tags HTML são removidas e então somente o texto é extraído.

O conteúdo extraído é então carregado para dentro de um arquivo TXT, que fica armazenado no sistema de arquivos. Esse passo foi criado para que o processo de extração não precise ser repetido mais de uma vez para o mesmo artigo, dessa maneira, a aplicação verifica se, pelo nome do documento, se o mesmo já teve seu conteúdo extraído. Esse conteúdo extraía não é, já no inicio, carregado no banco e sim em um arquivo, pois como era previsto que vários testes precisariam ser efetuados e cada teste possuiria a sua própria base de dados, a cada início de um novo teste somente os arquivos TXT precisavam ser replicados para a pasta do novo teste.

2. Carregar Conteúdo É nessa atividade que o conteúdo, já em formato TXT, é carregado para dentro do banco de dados. O conteúdo do documento, antes de ser carregado passava por um pré-processamento onde todas as letras eram convertidas para formato em minúsculo, eram removidos carecteres especiais, as vogais com acento eram substituidas pelas mesmas sem acento.

E então o texto tratado era inserido, como um novo registro, na tabela *artigos* populando a coluna *texto_completo*. Referente a essa mesma tabela, as colunas *inicio* e *termino* eram populadas com a data/hora do inicio e do termino do processamento. Essa informação era utilizada para identificar processos travados.

3. Separar Palavras Primeiramente, o texto era recuperado do banco e então passava por um processo onde as *stopwords* eram removidas, foram utilizadas as mesmas listas da Aplicação de Testes 1 e, após isso, o texto era inserido em um novo processo que faz o *stemming* das palavras, onde também foram utilizadas as mesmas bibliotecas da Aplicação de Testes 1.

Após o texto ter suas *stopwords* removidas e realizado o *stemming*, ele era quebrado nos espaços em branco, com isso, obtinha-se uma lista contendo todas as palavras que deveriam ser armazenadas. Cada palavra era analisada e, caso fosse uma nova palavra, considerando somente o documento sendo processado, ela era inserida, na tabela *palavras*. Caso contrário, recupera-se o registro que já há continha e então o contador, coluna *palavras.qtd*, era incrementado.

4. Gerar Índice Com seu funcionamento semelhante à Aplicação de Testes 1 , essa é a única atividade que é executada levando-se em conta o artigo sendo indexado e todos os outros já indexados. O cálculo do *TFiDF* foi implementado passo a passo onde o resultado de cada passo é armazenado separadamento para então ser combinado no final processo.

São recuperadas todas as palavras armazenadas na tabela *palavras* e então seu peso é calculado. Primeiramente são calculados valores básicos a serem utilizados na fórmula *TFiDF*, que são:

- **Total de Documentos**, total de documentos que compõem o índice, utilizando o comando:

```
select count(ARTIGO_NM) as totalDocumentos
  from ARTIGOS;
```

- **Total de Documentos que contém a palavra**, total de documentos, dentre os que compõem o índice, que possuem a palavra sendo analisada, utilizando o comando:

```
select count(ARTIGO_NM) as totalDocumentosContemPalavra
  from PALAVRAS
 where SEQUENCIA = ‘ + Palavra sendo analisada + ’ ;
```

- **iDF (inverse document frequency**, frequência invertida da palavra sendo avaliada em relação aos documentos do índice, calculada pela fórmula (5.5).

$$iDF = log_{10}(\frac{totalDocumentos}{totalDocumentosContemPalavra}) \tag{5.5}$$

Após, é criado uma lista composta por todos os documentos indexados detentores da palavra sendo avaliada para então, por documento, ser calculado o TF(term frequency). A lista é criada pelo seguinte comando:

```
select ARTIGO_NM, QTD
 from PALAVRAS
where PALAVRA = ‘ + Palavra sendo avaliada + ’;
```

O TF é então calculado pela fórmula (5.6):

$$TF = (\frac{qtd}{maiorFrequencia}) \tag{5.6}$$

Para então o peso final ser calculado, pela fórmula (5.7):

$$w = TF \times iDF \tag{5.7}$$

O peso final, w, é então armazenado, assim como o TF e o iDF, na tabela *palavras*, cada qual em suas respectivas colunas. Esse peso será utilizado no cálculo da Similaridade do Cosseno no processo de busca, dessa maneira, as informações contidas na tabela *palavras* compõe o índice.

Busca

Possui basicamente o mesmo funcionamento que na Aplicação de Testes 1, a única diferança fica por conta do modelo de dados, diferente, mas na síntese são consideradas as mesmas informações, ou seja, o peso que palavra possui dentro dos documentos.

O processo de Busca também baseia-se no modelo de dados apresentado na Figura 5.4. O índice, à ser utilizado nos cálculos de similaridade e que é composto por todos os

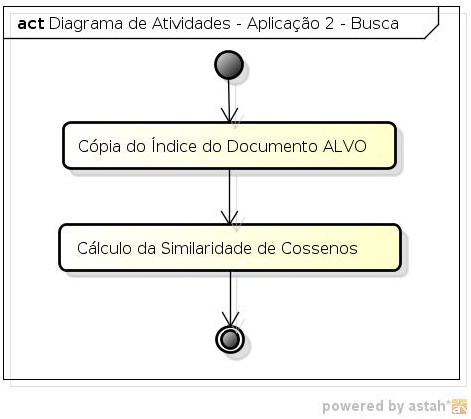

Figura 5.6: Diagrama de Sequência - Aplicação 2 - Busca

documentos indexados está armazenado na tabela *palavras*. A arquitetura do processo da Busca está representado na Figura 5.6. Suas atividades serão detalhadas na sequência.

A entrada nesse processo é feita pelo usuário, que deve informar um documento, chamado ALVO, para ser comparado aos demais, também indexados, para que seja retornada uma lista ordenada de acordo com a similaridade encontrada entre o ALVO e os demais documentos.

1. Cópia do Índice do Documento Alvo Nesta atividade, os dados referentes ao índice do documento ALVO, contidos na tabela *palavras* são copiados para a tabela *palavras_temp*. O único objetivo dessa atividade é facilitar o cálculo a ser executado, uma vez que o mesmo foi realizado utilizando-se comando SQL e o fato de tais informações, índice do ALVO e o índice dos demais documentos, estarem em tabelas distintas facilita o cálculo.

2. Cáculo da Similaridade de Cossenos Aqui são recuperados todos os documentos indexados, exceto o ALVO e, para cada documento, é calculada a Similaridade de Cossenos. Os pesos das palavras de cada documento descrevem um ponto em um espaço euclidiano. E esse ponto, relacionado ao ponto 0 descrevem uma reta. O objetivo então é calcular o cosseno do ângulo formado entre as retas descritas pelo documento ALVO e outro determinado documento objetivo da comparação. Quanto mais próximas as retas, mais semelhantes serão, chegando a 1 quando são exatamentes iguais, ou seja, uma reta está sobreposta a outra. Conforme explanação feita no Cápitulo 2, Seção 2.2, nesse ponto foi implementada a fórmula da Similaridade de Cossenos, utilizando o seguinte comando SQL:

```
insert into BUSCASIMPLES_TEMP
```

```
select '+Doc+',
       (( select sum(F1.W * F2.W)
            from PALAVRAS F1, PALAVRAS_TEMP F2
           where F1.PALAVRA = F2.PALAVRA
             and F1.ARTIGO_NM in ('+Doc+') )
        / ((select sqrt(sum(pow(w,2)))
              from PALAVRAS_TEMP) * (select sqrt(sum(pow(W,2)))
                                       from PALAVRAS
                                      where ARTIGO_NM in ('+Doc+'))));
```

O comando acima insere na tabela *buscaSimples_Temp* o valor encontrado. A ordenação resultante pode ser obtida com o seguinte comando SQL:

```
  select ARTIGO_NM, COS_SIM
    from BUSCASIMPLES_TEMP
order by COS_SIM desc;
```

Na Tabela 5.4 pode-se conferir o resultado de uma busca, onde o ALVO é o documento 44.

Tabela 5.4: Exemplo de ordenação obtida em um dos testes realizados.

artigo_nm	cos_sim
45	0.0118489
26	0.0116184
41	0.00949486
43	0.00933527
23	0.0088136

5.3 Experimentos

A metodologia para avaliar a eficácia do método proposto, conforme descrito no Capítulo 3 na Seção 3.5, foi combinada com as possíveis formas de realizar o processo de indexação, com base nos parâmetros do processo de indexação implementados na Aplicação de Testes 1:

- **Tamanho Máximo da Sequência**

- **Tamanho Mínimo da Sequência**
- **Tamanho Máximo da Janela**
- **Valor considerado Frequente**
- **Peso pelo Tamanho da Sequência**
- **Peso pela Região no texto**
- **Fator de Incremento pelo tamanho Sequência**

Os parâmetros 1) **Tamanho Máximo da Sequência**, 2) **Tamanho Mínimo da Sequência**, 3) **Tamanho Máximo da Janela**, 4) **Valor considerado Frequente** e 5) **Fator de Incremento pelo tamanho Sequência** foram configurados, em todos os testes, respectivamente, com os valores: 1) Sem limite, 2) 1, 3) 3, 4) 2 e 5) 2. Esses valores não foram alvos de testes, ou seja, não sofreram alterações.

O **Tamanho Máximo da Sequência** foi criado e utilizado em testes anteriores , com documentos científicos da área médica, onde as sequências frequentes geradas chegavam a alcançar mais de 12 palavras e isso acabava gerando um *overhead*, por isso, em alguns momentos esse parâmetro era utilizado com o intuito de limitar o avanço do algoritmo. O problema gerado na utilização desse parâmetro é que ao limitar o tamanho máximo de uma sequência em, por exemplo, 10 e existir sequência do com 20 palavras, então, seriam geradas 10 subsequências, contendo 10 palavras cada uma. Com isso podem ser geradas subsequências qye representam uma única sequência, essa sim, máxima. Nos textos utilizados, detalhados no Capítulo 4, o problema de *overhead* por conta do tamanho de sequências frequentes geradas não ocorreu, por uma característica própria dos textos. A maior sequência gerada foi de 6 palavras.

Em relação ao **Tamanho Mínimo da Sequência**, foi fixado em 1 pois tinha-se a intenção de que a representação computacional dos artigos fosse composta, também, por sequências com somente uma única palavra, caso, sendo frequente, não fosse uma subsequência de uma sequência frequente. Isso possibilita a existencia de sequências máximas frequentes com apenas uma única palavra.

Já o **Tamanho Máximo da Janela**, limitado em 3, aumentado em 1 em relação ao trabalho [AM], com o objetivo de gerar mais sequências. Mesmo assim, menos de 1% de todas as sequências geradas possuiam mais de 2 palavras na janela entre as demais palavras.

O **Valor considerado Frequente**, configurado em 2, com o objetivo de que toda a sequência que apareça mais de uma vez no texto seja considerada frequente e seja indexada. Com isso, espera-se, aumentar o número de sequências na representação de um documento.

E, o **Fator de Incremento pelo tamanho Sequência** estipulado em 2, pois, como o incremento dessa fator é exponencial quando aplicado ao peso, qualquer incremento aumenta em muito a distância entre as sequências maiores e as menores, o que não é interessante. Alguns testes preliminares chegaram a ser executados, com esse parâmetro setado em 3, e não apresentaram nenhum ganho. O que pode ser notado foi somente um aumento na distância entre os pesos das sequências menores em relação às maiores.

Os demais parâmetros, 1) **Peso pelo Tamanho da Sequência**, 2) **Peso pela Região** foram variados, bem como a remoção ou não das *stopwords*. Tais parâmetros sofreram variações pois compõem a lista de objetivos do método proposto, ou seja, para validar o método proposto precisa-se avaliar o seu comportamento com, sem e com a combinação deles. A remoção das *stopwords* sofreu variações nos experimentos com o objetivo de visualisar o comportamento das mesmas na formação das sequências. As combinações criadas estão listadas na Tabela 5.5.

Tabela 5.5: Experimentos criados com base nas combinações possíveis.

Experimento	Tipo dos Tokens	Remoção de *stopwords*	Peso pelo Tamanho da Sequência	Peso pela Região no Texto
1	MFS	Sim	Não	Não
2	MFS	Sim	Não	Sim
3	MFS	Sim	Sim	Não
4	MFS	Sim	Sim	Sim
5	MFS	Não	Não	Não
6	MFS	Não	Não	Sim
7	MFS	Não	Sim	Não
8	MFS	Não	Sim	Sim
9	Palavras	–	–	–

O Tipo dos Tokens diz respeito a utilização de sequências máximas frequentes(MFS) ou palavras simples(Palavras). Os experimentos envolvendo Tokens formados por palavras simples não sofreram variação e sempre foram feitos removendo as *stopwords*.

Como pode-se verificar na Tabela 5.5, por conta das combinações possíveis, foram geradas 9 experimentos diferentes. Combinar esses 9 experimentos com as 6 avaliações propostas, cada uma contendo 6 documentos, chegando aos 54 testes, cada qual indexando 6 documentos distintos. Ao todo, a etapa de indexação foi realizada 324 vezes. Por conta da arquitetura das Aplicações de Testes 1 e 2, para cada teste foi criada um banco de dados exclusivo.

Os resultados dos testes serão apresentados e discutidos a seguir.

5.4 Resultados Encontrados

A partir da combinação dos experimentos e das avaliações, propostas como metodologia de avaliação e realizadas, foram criados 54 testes, divididos em 6 grupos, respectivos e referentes às avaliações, que constão no Capítulo 3 na Tabela TABELA CONTENDO AS COMBINAÇOES DAS AVALIACOES. A seguir, estão os resultados das execuções, comparados entre si. As linhas realçadas em verde representam o experimento que obteve o melhor resultado.

O Percentual de Acerto, utilizado para compara a eficácia entre um experimento e outro, foi calculado com a fórmula (5.8):

$$PercentualdeAcerto = \frac{(P1 + P2 + P3 + P4 + P5)}{5} \tag{5.8}$$

Onde: Pn são as posições no ranking final e recebe valor de acordo com a distância entre o experimento e a marcação dos avaliadores. Quando o artigo encontra-se na mesma posição que a informada pelos avaliadores, então o valor é 1. Tal valor é decrementado em 0,2 para cada posição deslocada, para a esquerda ou direita.

O Tamanho do Índice foi calculado exportando-se os dados das tabelas *freq_seq*, no caso dos Tokens MFS, e *palavras*, no caso dos Tokens Palavras. Foi exportado somente os dados necessários a execução do procedimento da Busca, conforme descrito anteriormente.

Os dados referentes às avaliações coletadas pelo Sistema AVALIA podem ser consultadas no Capítulo 4, Tabela 4.1. A colocação final, de cada avaliação, encontra-se na primeira coluna de cada tabela.

Avaliação 1

Sendo o ALVO o documento 41 - "Roteiro para a Era pós-Kadafi"e indexados os documentos:

- 17 - O regime de Kadafi agoniza
- 20 - A base desensarilha as armas
- 35 - O futuro sem Kadafi
- 36 - O perigoso jogo dos extremistas
- 40 - Novo prazo para a PDVSA

Tem seus resultados apresentados na Tabela 5.6

Tabela 5.6: Resultado dos experimentos realizados referentes à Avaliação 1.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	35	17	36	20	40	–	100%
1	20	36	17	40	35	16,4KB	56%
2	35	17	20	40	36	16,4KB	84%
3	20	17	35	40	36	16,4KB	64%
4	20	17	35	40	36	16,4KB	64%
5	35	20	36	17	40	26,4KB	84%
6	35	20	36	17	40	26,4KB	84%
7	17	20	35	40	36	26,4KB	68%
8	17	20	35	40	36	26,4KB	68%
9	17	35	36	40	20	69,5KB	84%

Avaliação 2

Sendo o ALVO o documento 44 - "O teste do salário mínimo e do MST"e indexados os documentos:

- 23 - A primeira privatização
- 26 - O novo PT de sempre
- 41 - Roteiro para a Era pós-Kadafi
- 43 - O MST na era Dilma Rousseff
- 45 - A propósito dos objetivos do MST

Tem seus resultados apresentados na Tabela 5.7

Avaliação 3

Sendo o ALVO o documento 24 - "PAC continua devagar"e indexados os documentos:

- 22 - A formatura de Dilma
- 29 - Contrastes de organização
- 31 - A crise da dívida
- 32 - Anos de silêncio
- 39 - A diversificada tecnologia da corrupção

Tem seus resultados apresentados na Tabela 5.8

Tabela 5.7: Resultado dos experimentos realizados referentes à Avaliação 2.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	43	45	23	26	41	–	100%
1	41	45	23	26	43	17,7KB	68%
2	41	45	23	26	43	17,7KB	68%
3	43	41	45	23	26	17,7KB	76%
4	43	45	41	23	26	17,7KB	84%
5	43	45	23	41	26	39,9KB	92%
6	43	45	41	23	26	39,9KB	84%
7	43	45	26	41	23	39,9KB	84%
8	43	45	26	41	23	39,9KB	84%
9	45	26	41	43	23	64,1KB	60%

Tabela 5.8: Resultado dos experimentos realizados referentes à Avaliação 3.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	29	31	39	32	22	–	100%
1	31	22	39	32	29	21,8KB	68%
2	31	22	39	32	29	21,8KB	68%
3	22	31	39	32	29	21,8KB	68%
4	22	31	39	32	29	21,8KB	68%
5	39	31	22	29	32	35,9KB	68%
6	39	31	22	29	32	35,9KB	68%
7	39	22	31	32	29	35,9KB	60%
8	39	22	31	32	29	35,9KB	60%
9	31	32	39	29	22	64,3KB	80%

Avaliação 4

Sendo o ALVO o documento 30 - “Amplo combate”e indexados os documentos:

- 19 - O dilema de Dilma
- 24 - Contrastes de organização
- 34 - A crise da dívida
- 38 - Anos de silêncio
- 42 - A diversificada tecnologia da corrupção

Tem seus resultados apresentados na Tabela 5.9

Tabela 5.9: Resultado dos experimentos realizados referentes à Avaliação 4.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	42	24	19	34	38	–	100%
1	38	19	34	24	42	19,7KB	52%
2	38	34	19	42	24	19,7KB	52%
3	34	38	24	42	19	19,7KB	48%
4	34	24	38	42	19	19,7KB	64%
5	19	34	24	38 [1]	42 [1]	29,7KB	40%
6	19	34	24	38 [1]	42 [1]	29,7KB	40%
7	34	19	24	38 [1]	42 [1]	29,7KB	40%
8	34	19	24	38 [1]	42 [1]	29,7KB	40%
9	38	24	42	19	34	64,7KB	68%

Avaliação 5

Sendo o ALVO o documento 30 - "Terror em Oslo"e indexados os documentos:

- 18 - Lambança nos projetos
- 28 - O terror na sociedade ideal
- 30 - Amplo combate
- 44 - O teste do salário mínimo e do MST
- 46 - Segurança e educação

Tem seus resultados apresentados na Tabela 5.10

Avaliação 6

Sendo o ALVO o documento - "Terror em Oslo"e indexados os documentos:

- 25 - Uma escolha infeliz
- 33 - Redenção social
- 37 - Gestão municipal pode adotar novas práticas
- 47 - Controle da verba para Ensino é tíbio
- 48 - Educação não é carente apenas de verbas

Tem seus resultados apresentados na Tabela 5.11

[1]Nenhuma ligação com o ALVO foi identificada.
[2]Nenhuma ligação com o ALVO foi identificada.
[3]Nenhuma ligação com o ALVO foi identificada.

Tabela 5.10: Resultado dos experimentos realizados referentes à Avaliação 5.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	28	30	44	46	18	–	100%
1	28	44	18	46	30	17,8KB	76%
2	28	18	44	46	30	17,8KB	76%
3	28	46	44	18	30	17,8KB	76%
4	28	46	44	18	30	17,8KB	76%
5	44	28	18	30 [2]	46 [2]	27,1KB	40%
6	44	28	18	30 [2]	46 [2]	27,1KB	40%
7	44	28	18	30 [2]	46 [2]	27,1KB	40%
8	44	28	18	30 [2]	46 [2]	27,1KB	40%
9	28	18	46	30	44	57,7KB	68%

Tabela 5.11: Resultado dos experimentos realizados referentes à Avaliação 6.

	Posição						
Experimento	1	2	3	4	5	Tamanho do Índice	Percentual de Acerto
Avaliadores	47	48	37	25	33	–	100%
1	48	33	47	25	37 [3]	15,9KB	68%
2	48	33	47	25	37 [3]	15,9KB	68%
3	33	48	47	25	37 [3]	15,9KB	68%
4	33	48	47	25	37 [3]	15,9KB	68%
5	48	33	47	25	37 [3]	25,7KB	68%
6	48	47	33	25	37 [3]	25,7KB	76%
7	48	47	33	25	37 [3]	25,7KB	76%
8	48	47	33	25	37 [3]	25,7KB	76%
9	25	33	37	47	48	61,8KB	52%

Avaliação dos Resultados

O resultado consolidado, que é a comparação entre os experimentos realizados para cada avaliação pode ser conferido na Tabela 5.12. A coluna Percentual Final representa a média entre os percentuais de acerto de todas as avaliações.

Em primeiro lugar ficou a proposta do trabalho, que é o uso das *MFS*, em conjunto com valoração da importância da *MFS* de acordo com seu tamanho e região que aparece no texto. O segundo lugar foi a opção sem valorar a importância da *MFS* de acordo com seu tamanho mas calculado em relação à região que elas estão dispostas no texto. E o terceiro lugar ficou com a opção clássica que é o uso de palavras simples como descritores de um documento. Todas as 3 primeiras colocações fizeram uso da remoção de *stopwords*.

Os experimentos que ficaram abaixo da metade da classificação, abaixo da posição

Tabela 5.12: Resultado final consolidado.

		Percentual de Acerto por Avaliação						
Colocação	Experimento	1	2	3	4	5	6	Percentual Final
1^o	4	64%	84%	68%	64%	76%	68%	70,667%
2^o	2	84%	68%	68%	52%	76%	68%	69,333%
3^o	9	84%	60%	80%	68%	68%	52%	68,667%
4^o	3	64%	76%	68%	48%	76%	68%	66,667%
5^o	5	84%	92%	68%	40%	40%	68%	65,333%
6^o	6	84%	84%	68%	40%	40%	76%	65,333%
7^o	1	56%	68%	68%	52%	76%	68%	64,667%
8^o	7	68%	84%	60%	40%	40%	76%	61,333%
9^o	8	68%	84%	60%	40%	40%	76%	61,333%

5, incluindo ela, são os experimentos que não removeram as *stopwords* o que não se mostrou algo muito eficaz. A única exceção fica por conta do experimento 1, que teve as *stopwords* removidas mas não teve nenhuma outra heurística associada. Isso demonstra que colocando-se lado a lado o uso de MFS com o de palavras simples, no cenário proposto, as palavras simples se sobressaem. Comportamento esse facilmente compreendido ao levar-se em conta que a fórmula de valoração *TFiDF* foi criada para fucnionar no cenário de palavras simples. O que demonstra a necessidade de se criar novas métricas para o calculo do peso em se tratando de MFS, ou qualquer outro tipo de sequência de texto como descritores, *tokens*, de um documento.

Como pode-se verificar nos resultados das Avaliações 4 e 5, nos experimentos 5, 6, 7 e 8 alguns artigos não obtiveram nenhuma ligação com os demais, enquanto nos testes 1, 2, 3 e 4 tal ligação ocorreu. Isso se deve a não remoção das *stopwords*, que ocasiona em sequências muito específicas. Pode-se conferir que os índices dos testes onde não ocorre a remoção das *stopwords* sempre são maiores, isso porque mais sequências são geradas, mas tais sequências passam a contar com mais palavras que, além de não servir na diferenciação uma vez que são palavras consideradas comuns e de grande ocorrência, levam a criação de sequências muito específicas e diminui as chances de se encontrar a mesma sequência em documentos distintos. São consideradas muito específicas pois, ao não se remover as *stopwords*, as sequências geradas acabam carregando com mais facilidade, na integralidade, o estilo de escrita do autor. O que dificulta o processo de busca, pois as sequências de textos que serviriam para ligar um documento ao outro estão muito específicas diminuindo a assertividade.

Outro ponto interessante de se notar é o tamanho do índice gerado. Ao se comparar o tamanho do índice dos experimentos 4 e 2, que foram, respectivamente, o primeiro e o segundo colocados, em média, eles são 40,82% menor que o índice gerado para o uso de palavras simples. Isso acontece pois uma sequência, ao ser considerada máxima, o que é necessário para ser uma MFS, acaba englobando várias outras sequências menores.

Tabela 5.13: Comparação entre o tamanho dos índices gerados.

	Percentual de Acerto por Avaliação						
Experimento	1	2	3	4	5	6	Média
1	16,4KB	17,7KB	21,8KB	19,7KB	17,8KB	15,9KB	18,21KB
2	16,4KB	17,7KB	21,8KB	19,7KB	17,8KB	15,9KB	18,21KB
3	16,4KB	17,7KB	21,8KB	19,7KB	17,8KB	15,9KB	18,21KB
4	16,4KB	17,7KB	21,8KB	19,7KB	17,8KB	15,9KB	18,21KB
5	26,4KB	39,9KB	35,9KB	29,7KB	27,1KB	25,7KB	30,78KB
6	26,4KB	39,9KB	35,9KB	29,7KB	27,1KB	25,7KB	30,78KB
7	26,4KB	39,9KB	35,9KB	29,7KB	27,1KB	25,7KB	30,78KB
8	26,4KB	39,9KB	35,9KB	29,7KB	27,1KB	25,7KB	30,78KB
9	69,5KB	64,1KB	66,3KB	64,7KB	57,7KB	61,8KB	64,01KB

No caso da indexação onde os tokens são MFS, o tamanho do índice só sofre algum impacto por conta da remoção ou não das *stopwords*. Outro motivo para gerar impacto no tamanho seria a não execução do *stemming*, o que não ocorreu em nenhum experimento. As heurísticas implementas, valoração pelo tamanho da sequência e região no texto, não geram alteração no seu tamanho, o que pode ser conferido na Tabela 5.13.

5.5 Considerações Finais

A implementação do ambiente de testes e os experimentos realizados mostraram que o uso de sequências máximas frequentes, em conjunto com as heurísticas referentes ao tamanho de uma sequência e à região delas no texto, é eficaz em comparação ao uso de palavras simples, tanto do percentual de acerto quanto no tamanho dos índices gerados. Isso encoraja um estudo mais aprofundado e, principalmente, a realização de mais testes em outros tipos/tamanho de documentos para se confirmar sua eficácia.

Capítulo 6

Conclusão

O objetivo inicial do trabalho era propor uma metodologia a qual fizesse uso de sequências de texto, as MFS, como tokens básicos para representação de documentos em conjunto com o Modelo Espaço Vetorial. Porém, no decorrer das pesquisas e dos testes verificou-se a importância do conhecimento de fundo que, levantado junto aos avaliadores, mostrou-se uma ótima ideia para a valoração das regiões do texto o que acabou alterando o foco do trabalho. O foco do trabalho foi alterado, visando tratar o conhecimento de fundo coletado criando-se uma memória de cálculo, utilizada, em conjunto com a valoração referente ao tamanho das sequências, e que apresentou resultados superiores ao uso de palavras simples.

As duas heurísticas propostas, **Peso pela Região** e **Peso pelo Tamanho** se mostraram válidas. A **Peso pela Região** apresentou os melhores resultados, sendo utilizada sozinha obteve segundo lugar na classificação geral.

Esse conhecimento de fundo, após coletado para uma determinada área do conhecimento, política, por exmeplo, pode ser replicado a outros documentos da mesma área.

Outro ponto que apresentou vantagem foi em relação ao tamanho do índice, que, em todos os experimentos realizados, quando os tokens eram MFS chegavam a ficar com quase a metade do tamanho quando se utilizada palavras simples na indexação.

Conforme a proposta do trabalho, não foi alvo de investigação a performance do método proposto, apesar disso, nos inúmeros testes realizados era facilmente visível que quando o processo de indexação de documento era realizado com o objetivo de se extrair MFS ele chegava a demorar quase o dobro do tempo que se comparado a extração de palavras simples. É um problema conhecido da técnica que é alvo de estudos tentando-se criar uma algoritmo mais eficaz para a extração de MFS. Vale lembrar que o processo de indexação é executado somente uma única vez por documento, fato esse que, em conjunto com os resultados encontrados, ainda assim demonstram vantagens no uso de MFS.

Contudo, mais testes precisam ser realizados com o intuito de se comprovar a sua eficácia. Variando os dados coletados para a avaliação ou até mesmo a forma de avaliação.

6.1 Limitações

A principal limitação deste trabalho diz respeito ao tamanho dos textos utilizados nos experimentos. Foram utilizados, por dois motivos, textos pequenos:

- A indexação de textos grandes ocasionava em um tempo de processamento muito alto, o que é conhecido como uma fraqueza da técnica MFS, principalmente por conta da não limitação do tamanho máximo que uma MFS pode ter.
- Optou-se por utilizar dados feitos por humanos para as comparações de efetividade, o que resultou na escolha de textos menores para facilitar o processo de coleta de dados.

Considera-se uma grande limitação pois, em testes anteriores, utilizou-se artigos científicos com tamanho médio maior em relação aos textos utilizados nos experimentos. Pela quantidade maior de texto, mais MFS eram geradas, o que facilitava e aumentava a ligação entre os documentos. E, os textos utilizados nos textos eram editoriais sobre política, enquanto nos testes preliminares eram artigos científicos. Por características próprias, talvez do estilo de escrita dos autores, ou talvez pelo público alvo de cada um, dos textos científicos eram extraídas MFS maiores e com mais repetições, ou seja, a qualidade dos índices gerados era extremamente boa.

Apesar dessa limitação, os testes finais comprovaram as vantagens do uso de MFS.

6.2 Trabalhos Futuros

Frente aos bons resultados obtidos pelo método proposto, abrem-se várias linhas de investigação, como segue:

- Heurística: Peso pela Região. Realizar novos testes para comprovar sua eficácia. Tais testes devem ser feitos com objetivo de se valiar a eficácia do método quando a categoria do conteúdo dos documentos utilizados na coleta dos dados não é igual ao dos documentos sendo avaliados.
- Heuristica: Peso pelo Tamanho. Novos estudos podem ser realizados afim de se avaliar a fórmula proposta para valoração de uma MFS de acordo com seu tamanho.
- *Stopwords*. Pode-se avaliar melhor a remoção das *stopwords*, no sentido de se identificar a lista ideal. A ideia clássica das remoção de *stopwords* é remover palavras que aparecem muito nos textos, com isso, além de não carregarem valor semântico, por serem extremamente frequentes não servem para diferenciar os documentos. Talvez, no caso de sequências de texto, por exemplo as MFS, o mais interessante seria remover as palavras com o intuito de remover das sequências geradas o estilo

de escrita do autor. Por outro lado, a maioria das listas de *stopwords* são constituídas por alguns verbos que, no caso da formação das sequências de texto pode ser interessante que sejam mantidos.

- Tamanho da Janela(*gap)*. Nova heurística para valoração das MFS pode ser criada com base na quantidade de palavras que estavam inseridas entre as selecionadas. Uma MFS de, por exemplo, tamanho 3 que no texto original eram 3 palavras consecutivas pode ter seu peso aumentado em relação a outra MFS, também de tamanho 3, mas que no texto original possuia outras palavras entre as 3 já selecionadas.

- Combinar palavras simples com MFS. Apesar de os testes terem sido realizados considerando-se MFS com tamanho mínimo de 1 palavra, o que pode ser considerada uma palavra simples, essa MFS teve seu peso valorado utilizando as técnicas de valoração propostas para as MFS. O que pode ser estudado é a combinação das duas técnicas de valoração. O cálculo do peso utilizando *TFiDF* as MFS com somente uma única palavra e as heurísticas para MFS quando forem sequências compostas por 2 ou mais palavras.

- Ánalise Formal de Conceitos. De acordo com [Pri96], [CPGV05a] e [Pri97], entre outros, apontam as vantagens do uso da FCA na Recuperação de Informação, criando uma nova maneira, em contra partida as listas ranqueadas, de visualizar-se o resultado de uma pesquisa. A proposta é utilizar MFS como atributos, a intenção, dos objetos, a extensão, dentro de um contexto formal.

Referências Bibliográficas

[AM] Helena Ahonen-Myka. Discovery of frequent word sequences in text.

[AM99] Helena Ahonen-Myka. Finding all maximal frequent sequences in text, 1999.

[CPGV05a] Juan M. Cigarr, Anselmo Peñas, Julio Gonzalo, and Felisa Verdejo. Automatic selection of noun phrases as document descriptors in an fca-based information retrieval system. *LNCS*, 3403:49–63, 2005.

[CPGV05b] Juan M. Cigarrán, Anselmo Peñas, Julio Gonzalo, and Felisa Verdejo. Automatic selection of noun phrases as document descriptors in an fca-based information retrieval system. In *In B. Ganter and R. Godin (Eds.): ICFCA 2005, LNCS*, pages 49–63, 2005.

[DAm04] Antoine Doucet and Helena Ahonen-myka. Non-contiguous word sequences for information retrieval. In *In Proceedings of the 42nd annual meeting of the Association for Computational Lingustics, Workshop on Multiword Expressions: Integrating Processing*, 2004.

[DAM10] Antoine Doucet and Helena Ahonen-Myka. An efficient any language approach for the integration of phrases in document retrieval. *Language Resources and Evaluation*, 44(1-2):159–180, 2010.

[Dou05] Antoine Doucet. Advanced document description, a sequential approach, 2005.

[Jon72] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[Luh57] Hans Peter Luhn. A statistical approach to mechanical encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.

[Mag09] A. C. P. Magalhães, M. N. & LIMA. *Noções de Probabilidade e Estatística*. 2009.

[Moo50] C.N. Mooers. Information retrieval viewed as temporal signaling. 1950.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, Vol. 14,No.3:pp. 130–137., 1980.

[Pri96] Uta Priss. Formal concept analysis in information sciente. *ANNUAL REVIEW OF INFORMATION SCIENCE AND TECHNOLOGY*, 40:521–543, 1996.

[Pri97] Uta E. Priss. A graphical interface for document retrieval based on formal concept analysis. In *Midwest Artificial Intelligence and Cognitive Science Conference*, 1997.

[Sma93] Frank Smadja. Retrieving collocations from text: Xtract. *Journal of Computational Conference on Extending Database Technology*, EDBT' 96:3–17, 1993.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[ZTMfE97] Chengxiang Zhai, Xiang Tong, Natasa Milic-frayling, and David A. Evans. Evaluation of syntactic phrase indexing - clarit nlp track report. In *The Fifth Text REtrieval Conference (TREC-5)*, pages 347–358, 1997.

[ZZY+08] Guo-Qing Zhang, Guo-Qiang Zhang, Qing-Feng Yang, Su-Qi Cheng, and Tao Zhou. Evolution of the internet and its cores. *New Journal of Physics*, 10, 2008.

Capítulo 7

Apendice A

7.1 Sistema AVALIA

Na Tabela 7.1, estão listados o código e o titulo de cada artigo utilizado no levantamento dos dados necessários para efetuar-se a comparação entre o método proposto e uma busca baseada em palavras simples.

Tabela 7.1: Marcações acumuladas por avaliação

Codigo	Titulo
17	O regime de Kadafi agoniza
18	Lambança nos projetos
19	O dilema de Dilma
20	A base desensarilha as armas
22	A formatura de Dilma
23	A primeira privatização
24	PAC continua devagar
25	Uma escolha infeliz
26	O novo PT de sempre
28	O terror na sociedade ideal
29	Contrastes de organização
30	Amplo combate
31	A crise da dívida
32	Anos de silêncio
33	Redenção social
34	O Supremo congestionado
35	O futuro sem Kadafi
36	O perigoso jogo dos extremistas
37	Gestão municipal pode adotar novas práticas
38	Corrida contra o tempo nos aeroportos
39	A diversificada tecnologia da corrupção
40	Novo prazo para a PDVSA
41	Roteiro para a Era pós-Kadafi
42	Terror em Oslo
43	O MST na era Dilma Rousseff
44	O teste do salário mínimo e do MST
45	A propósito dos objetivos do MST
46	Segurança e educação
47	Controle da verba para Ensino é tíbio
48	Educação não é carente apenas de verbas