

UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

**Uma Ferramenta para Análise de
Informações Textuais na Internet baseada
em Inteligência Coletiva**

por
Ueider Ferreira de Oliveira

Projeto de Graduação submetido como requisito
parcial à obtenção do grau de Bacharel em
Ciência da Computação

Orientador: Prof. Dr. Stanley Loh

Pelotas, novembro de 2010

*Dedico este trabalho aos meus pais Nilo e Cleusa, ao meu irmão Uesler,
aos meus avós e a todos que me apoiaram nesta caminhada.*

SUMÁRIO

| | |
|---|----|
| LISTA DE FIGURAS | 5 |
| LISTA DE TABELAS | 6 |
| LISTA DE ABREVIATURAS E SIGLAS | 7 |
| RESUMO | 8 |
| RESUMO | 9 |
| 1 INTRODUÇÃO | 10 |
| 2 TRABALHOS CORRELATOS | 12 |
| 2.1 Métodos para Comparar Textos | 12 |
| 2.2 Comparação de Árvores Sintáticas | 15 |
| 2.3 Técnicas para Tratar Variações em Frases ou Palavras | 17 |
| 2.4 Reconhecimento de Vinculação Textual | 17 |
| 2.5 Simplificação Textual | 19 |
| 3 FERRAMENTA VISL PARA ANÁLISE SINTÁTICA | 22 |
| 4 FERRAMENTA PARA ANÁLISE DE INFORMAÇÕES TEXTUAIS NA INTERNET BASEADA EM INTELIGÊNCIA COLETIVA | 25 |
| 4.1 Entrada e pré-processamento da frase original | 26 |
| 4.2 Frase Negativa | 26 |
| 4.3 Troca de posições entre sujeito e complemento | 27 |
| 4.4 Troca por sinônimos | 27 |
| 4.4.1 Verbos | 27 |
| 4.4.2 Substantivos, Adjetivos e Advérbios | 31 |
| 4.4.3 Dicionário | 32 |

| | | |
|------------|-----------------------------|----|
| 4.5 | Troca por antônimos | 32 |
| 4.6 | Buscas na internet | 33 |
| 4.7 | Testes | 33 |
| 4.8 | Tecnologias | 37 |
| 4.8.1 | JSP | 37 |
| 4.8.2 | CSS | 38 |
| 5 | CONSIDERAÇÕES FINAIS | 40 |
| | REFERÊNCIAS | 42 |

LISTA DE FIGURAS

| | | |
|-------------|---|----|
| Figura 2.1 | Coleção de strings (a) e sua lista invertida(b) | 14 |
| Figura 3.1 | Classificação gramatical de uma frase no VISL | 24 |
| Figura 4.1 | Tela inicial da ferramenta | 26 |
| Figura 4.2 | Flexão de número e pessoa | 28 |
| Figura 4.3 | Presente do modo indicativo | 29 |
| Figura 4.4 | Pretérito perfeito do modo indicativo | 29 |
| Figura 4.5 | Pretérito imperfeito do modo indicativo | 29 |
| Figura 4.6 | Pretérito mais que perfeito do modo indicativo | 29 |
| Figura 4.7 | Futuro do presente do modo indicativo | 30 |
| Figura 4.8 | Futuro do pretérito do modo indicativo | 30 |
| Figura 4.9 | Pretérito imperfeito do modo subjuntivo | 31 |
| Figura 4.10 | Presente do modo subjuntivo | 31 |
| Figura 4.11 | Futuro do modo subjuntivo | 31 |
| Figura 4.12 | Dicionário de sinônimos | 32 |
| Figura 4.13 | Dicionário de antônimos | 33 |
| Figura 4.14 | Busca pela frase "beterraba ajuda a retardar envelhecimento | 34 |
| Figura 4.15 | Busca pela frase "Maradona foi melhor que Pelé" | 35 |
| Figura 4.16 | Busca pela frase "Petrobras não encontrou petróleo" | 36 |
| Figura 4.17 | Processamento de uma página JSP | 37 |
| Figura 4.18 | Regra de estilo CSS | 39 |

LISTA DE TABELAS

| | | |
|------------|---|----|
| Tabela 2.1 | Exemplos de fingerprints e suas respectivas strings | 13 |
| Tabela 2.2 | Exemplo de simplificação natural e forte | 20 |
| Tabela 3.1 | Tags de classificação | 23 |
| Tabela 3.2 | Código-fonte Java para conectar ao VISL | 23 |
| Tabela 4.1 | Código-fonte Java para conectar ao Google | 33 |
| Tabela 4.2 | Exemplo de código JSP | 38 |
| Tabela 4.3 | Exemplo de código Java para conexões com sites | 38 |

LISTA DE ABREVIATURAS E SIGLAS

| | |
|-------|------------------------------------|
| AME | Approximate Member Extraction |
| SCOWL | Spell Checker Oriented Word Lists |
| LLM | Local Lexical Matching |
| RTE | Textual Entailment Recognitionn |
| ITG | Inversion Transduction Grammar |
| SE | Simplificador Sintático Estatístic |
| SS | Simplificador Sintático Simbólico |
| VISL | Visual Interactive Syntax Learning |
| CG | Constraint Grammar |
| JSP | Java Server Pages |
| HTML | Hyper Text Markup Language |

RESUMO

Pessoas procuram informações na web sobre como prevenir ou tratar doenças. Em certos momentos a pessoa pode se deparar com informações contraditórias, ou seja, alguma fonte de informação afirmando algo e outra contrariando esta mesma informação. Por isso, este trabalho apresenta uma ferramenta de busca baseada no conceito de Inteligência Coletiva para análise de informações encontradas na internet. Estas informações são encontradas através da análise de similaridade entre textos, frases ou citações, procurando textos que são a favor ou contra um determinado assunto, retornando ao usuário o número de resultados encontrados para as declarações favoráveis e contrárias.

As informações são encontradas usando as seguintes estratégias: busca pela frase original, pela frase negativa, pela frase com elementos sintáticos trocados e por frases com troca de palavras por sinônimos e antônimos.

Palavras-chave: Busca, Informações Repetidas, Inteligência Coletiva, Plágio.

TITLE: “A TOOL FOR ANALYSIS OF TEXTUAL INFORMATION ON THE INTERNET BASED ON COLLECTIVE INTELLIGENCE”

RESUMO

People seeking information on the web about how to prevent or treat disease. At moments one can come across contradictory information, or some source of information and another saying something contrary to this same information. Therefore, this work presents a search tool based on the concept of collective intelligence for analysis of information found on the Internet. This information is found by analyzing the similarity between texts, phrases or quotes, texts that are looking for or against a particular subject, returning to the user the number of results for the statements for and against. The information is found using the following strategies: the search for the original sentence by sentence negative, the sentence with syntactic elements of sentences and exchanged with an exchange of words by synonyms and antonyms.

Palavras-chave: search, repeated information, collective intelligence, plagiarism.

1 INTRODUÇÃO

O número de pessoas que tem acesso à internet é crescente no Brasil e consequentemente a busca por informações relevantes sobre diversos assuntos, inclusive informações que tratam sobre saúde e medicina. Os usuários procuram informações sobre doenças e seus tratamento ou até mesmo para tirar dúvidas se determinada atitude é ou não recomendável, por exemplo, se tomar suco de limão faz bem para gastrite.

No Brasil, segundo o Comitê Gestor da Internet, 39% dos usuários usam a web para procurar informações relacionadas à saúde e esse percentual aumenta para 60% quando se consideram apenas os que têm nível superior, já nos Estados Unidos, de acordo com uma pesquisa realizada pela *Pew Research Center's Internet American Life Project* em parceria com a *California HealthCare Foundation*, 61% dos adultos americanos buscam informações sobre saúde na internet e 60% afirmam que a informação encontrada na web afeta sua decisão em relação ao tratamento da doença [Zero Hora,2010].

Em certos momentos o usuário pode se deparar com informações contraditórias, ou seja, alguma fonte de informação afirmando algo e outra contrariando esta mesma informação. Por isso, este trabalho apresenta uma ferramenta de busca baseada no conceito de Inteligência Coletiva para análise de informações encontradas na internet. Estas informações são encontradas através da análise de similaridade entre textos, frases ou citações, procurando textos que são a favor ou contra um determinado assunto, retornando ao usuário o número de resultados encontrados para as declarações favoráveis e contrárias. Por exemplo, o usuário entra com uma frase no sistema, então é retornando ao usuário o número de resultados encontrados, para a frase original, frase negativa e variações da frase original, usando troca de palavras por sinônimos e antônimos. É importante ressaltar que os resultados encontrados e exibidos ao usuário, vão dar a este um indicativo de verdade, e não a verdade propriamente dita sobre o termo pesquisado.

O desenvolvimento do sistema foi baseado numa grande área chamada Inteligência Coletiva. O termo Inteligência Coletiva tem sido usado em diferentes aplicações, mas sempre tendo em comum a idéia de que “o todo é mais que a mera soma das partes”, ou seja, várias partes integradas podem realizar um objetivo que nenhuma

parte conseguiria em separado [Lévy, 1998 apud Loh, 2007].

Há também outro conceito relacionado: Sabedoria das Massas ou Multidões (*Wisdom of Crowds*), cunhado por Surowiecki (2004) apud Loh (2007). Segundo este conceito, a opinião da maioria das pessoas leva ao que é correto ou ao melhor caminho. Surowiecki comenta alguns casos possíveis de tal sabedoria, como quando a aeronave Challenger explodiu e o mercado de ações reagiu contra um dos fabricantes, mesmo sem análises e resultados da causa ou dos defeitos.

Além disso, esta ferramenta pode ser utilizada para auxiliar na detecção de plágio na Web, permitindo ao usuário encontrar textos similares através das estratégias utilizada para gerar e encontrar variações da frase original, disponíveis na internet.

Segundo Lancaster e Culwin (2005) apud Lukashenko (2007), plágio pode ser definido como o uso de um trabalho de outra pessoa, sem referenciá-lo a fonte original. Comumente, na prática, existem métodos diferentes de plágio. Algumas delas incluem:

- Plágio (copie e cole) - (uma cópia palavra a palavra da informação textual);
- Parafraseando (reafirmando o mesmo conteúdo com diferentes palavras);
- Plágio traduzido (tradução de conteúdo e utilização sem referência ao trabalho original);
- Plágio artístico (apresentando o mesmo trabalho em diferentes mídias: texto, imagens, etc);
- Plágio de idéias (usando idéias similares que não são de conhecimento comum);
- Plágio de código (usando códigos de programas de computador sem autorização ou referência);
- Sem uso adequado de aspas (não identificação de partes do conteúdo exato emprestado);
- Desinformação de referências (adicionando referência à fonte incorreta ou não existente).

A disponibilidade de documentos digitais na internet abre boas oportunidades de prosperidade para o “plagiarismo”, transformando-se em um processo extremamente fácil e atraente, resultando atualmente, em um grave problema para os editores, pesquisadores e educadores. Para combater o plágio, atualmente são desenvolvidos e utilizados muitos métodos. Estes métodos podem ser divididos em duas classes: os métodos de prevenção de plágio, como políticas de honestidade e/ou sistemas de punição e os métodos de detecção de plágio, como ferramentas de software para revelar o plágio automaticamente.

2 TRABALHOS CORRELATOS

Neste capítulo serão apresentados alguns trabalhos realizados nas áreas de interesse do presente trabalho, como métodos para comparação de texto e árvores sintáticas, técnicas para tratar variações em frases ou palavras, *Recognizing Textual Entailment* (RTE) e simplificação textual.

2.1 Métodos para Comparar Textos

Uma nova abordagem para detectar reutilização e modificação de partes de um texto é apresentada por Seo e Croft (2008). No trabalho foram introduzidas categorias de reutilização que serviram de base para a avaliação experimental. As categorias C1(*Most-Most*), C2(*Most-Considerable*) e C4(*Considerable-Considerable*) correspondem aos casos de quase duplicação; na C1 se enquadram os casos onde dois documentos são quase idênticos; na C2 os casos em que uma pequena passagem é adicionada ao texto de outro documento e na C4, quando um novo documento é composto de grandes partes de outros documentos. Enquanto C3(*Most-Partial*), C5(*Considerable-Partial*) e C6(*Partial-Partial*) correspondem à reutilização de texto local; na categoria C3 um documento inteiro é usado como um texto parcial de um novo documento, já a C5 é geralmente semelhante a C4, exceto para a quantidade de texto compartilhado e a C6 acontece com texto padrão ou frases comuns. *Fingerprints* é a representação numérica de um pedaço de texto ou string de um documento, a conversão de uma string para a forma numérica é feita através de algoritmos de hash como MD5 ou Rabin fingerprinting. Na Tabela 2.1 são apresentados exemplos com a representação numérica de cada string.

Se a semelhança entre os valores de *fingerprints* de dois textos forem maiores que 80%, 50% ou 10%, podemos dizer então que o texto é muito, consideravelmente ou parcialmente reutilizado, respectivamente. Para a detecção eficiente da reutilização de texto local é usada uma lista de índice invertido com os *fingerprints* extraídos do texto original e dos textos que serão testados para saber se existe alguma reutilização. Em seguida essas listas são mescladas e finalmente podem-se encontrar os níveis de reutilização dos textos

Tabela 2.1: Exemplos de fingerprints e suas respectivas strings

| String | Fingerprints |
|----------------------------------|--------------|
| one woman comedy by person Willy | [0x295D0A52] |
| one woman show by person Willy | [0x295D0A52] |
| company scheduled another money | [0xF1315F87] |
| company slated another money | [0xF1315F87] |

de acordo com as categorias definidas. Existem dois tipos de técnicas para recolher os *fingerprints* dos textos, para posteriormente colocá-los nas listas, uma delas é a técnica de sobreposição que utiliza janelas deslizantes, sendo esta deslocada por palavra, uma seqüência de palavras ou o seu valor de *hash* é tratado como um bloco (exemplos de técnicas de sobreposição: *K-gram*, $0 \bmod p$ e *Winnowing*), e a técnica de não-sobreposição que separa o texto em segmentos significativos, como frases ou sentenças, um exemplo desta técnica é o *Hash-breaking*, que apresenta problemas de sensibilidade, portanto Sean e Croft proporam um novo método chamado *fingerprinting* DCT para recolher fingerprints de um texto baseado no conceito de não-sobreposição.

Irving (2004) descreve como o Algoritmo Smith-Waterman pode ser usado para a comparação de textos ou programas de computador, e com isso podendo ser usado como uma poderosa ferramenta para a detecção de plágio ou conluio. O algoritmo de Smith-Waterman é um método clássico usado na biologia molecular para comparar duas seqüências, com vista à identificação de pontos muito semelhantes dentro deles, por exemplo, os chamados alinhamentos locais, dentro de seqüências biológicas. O material textual ordinário será analisado como uma seqüência de palavras, enquanto um programa de computador será analisado como uma seqüência de entidades lexicais da linguagem de programação. O método não depende de propriedades estatísticas, mas da semelhança estrutural entre (partes de) textos.

Ruddle (2006) apresenta um método de *string-matching* para analisar a navegação de um website através dos dados dos arquivos de log usando o algoritmo Smith-Waterman para o cálculo da similaridade. O método é dividido em duas fases que usam uma *string-matching* exata para calcular subseqüências de ligações que foram repetidas em sessões diferentes de navegação (trilhas comuns através do site) e, em seguida *string-matching* inexata para encontrar outras sessões semelhantes (uma comunidade de usuários com interesse similar). A avaliação mostrou como substrings poderiam ser usadas para entender os caminhos que o usuário optou por seguir dentro de um site e que *string-matching* exata ou inexata provêem maneiras complementares de identificar informações que poderiam ter sido de interesse para toda uma comunidade de usuários, mas que só foi encontrada por uma minoria.

Vernica e Li (2009) sugerem três algoritmos para responder a consultas por strings

aproximadas usando coleções de strings. O primeiro algoritmo calcula a similaridade entre as strings usando a função Jaccard com limiar previamente estabelecido. Posteriormente, é realizado um ranking através do grau de similaridade entre as strings. No segundo algoritmo proposto, é usado uma lista de *q-gram* com índice invertido, construído sobre a coleção de strings. Para cada gram da coleção de string, tem uma lista invertida de ids das seqüências que contém este gram, como mostrado na Figura 2.1. Um *q-gram* é uma subsequência de uma dada seqüência, podendo ser fonemas, sílabas, letras ou palavras. Um loop percorre a lista lendo cada elemento, as ids dos elementos similares são guardadas em um buffer, este procedimento é realizado até os resultados encontrados não poderem ser melhorados. A comparação entre os gram é realizada usando a também função de similaridade Jaccard. O terceiro algoritmo é a combinação dos dois algoritmos anteriores.

| ID | String | Weight |
|----|--------|--------|
| 1 | ab | 0.80 |
| 2 | ccd | 0.70 |
| 3 | cd | 0.60 |
| 4 | abcd | 0.50 |
| 5 | bcc | 0.40 |

(a) Dataset

| ab | cc | cd | bc |
|----|----|----|----|
| 1 | 2 | 2 | 4 |
| 4 | 5 | 3 | 5 |
| | | 4 | |

(b) Inverted lists

Figura 2.1: Coleção de strings (a) e sua lista invertida(b)

Lu e outros (2009) estudaram o problema para encontrar uma substring em um documento de texto que corresponde aproximadamente a algumas strings em um dicionário de strings, este problema é chamado de AME (*Approximate Member Extraction*), e propuseram um algoritmo que faz a verificação em duas etapas, sendo que na primeira, são filtradas as strings do dicionário, que são muito diferentes da string do texto e na segunda, são calculadas as similaridades entre as strings restantes usando a função Jaccard. Foram testados dois métodos para a filtragem, que ocorre na primeira etapa do algoritmo, um é baseado em índice invertido do dicionário de strings, neste método as strings do texto são consideradas como coleções de tokens. Para cada token do texto, o índice armazena as ids das strings do dicionário que incluem este token. O segundo método é baseado em assinaturas de geração. Este método centra-se na exploração de esquemas de assinatura que converter uma string para um conjunto de códigos de hash e faz a filtragem das strings irrelevante através de suas assinaturas, comparando a assinatura da string do texto com a assinatura da string do dicionário.

Batu e outros (2003) apresentam um algoritmo para o problema de editar a distância entre duas strings, reduzindo o tempo para estimar a similaridade entre elas, na primeira parte é usado um algoritmo de fraca aproximação que funciona em tempo

sublinear para descartar-se as strings que são muito diferentes e depois é usado outro algoritmo exato para encontrar a string semelhante, sendo assim o algoritmo mais lento e mais preciso pode concentrar-se em uma pequena fração de strings de entrada, já que na primeira parte é eliminadas as strings muito diferentes. Na primeira etapa, para reduzir a complexidade da consulta, é usado um processo chamado “ruler”, que reúne um conjunto sublinear de amostras de ambas as strings, e em seguida, constrói uma estrutura que contém todos os pares da seguinte forma (localização em A, bloco em B). Além disso, é feito o uso de recursão enquanto subdividem-se os blocos, o que permite melhorar ainda mais a complexidade do algoritmo. A semelhança entre as strings é medida através do algoritmo *Hamming distance*.

Bendersky e Croft (2009) testaram técnicas já desenvolvidas para a detecção de reutilização de texto para o cenário de busca da web e utilizam dois métodos para acompanhar o fluxo dessas informações. Primeiramente, para a recuperação inicial de um conjunto de documentos foi desenvolvido um algoritmo simples que usa um motor de busca da web, na primeira etapa a frase é colocada entre aspas no motor de buscas, se houver algum resultado, ele é adicionado ao conjunto de documentos, depois a frase é quebrada em pedaços, sendo feita uma nova pesquisa para cada pedaço da frase, os resultados também são adicionados ao mesmo conjunto. Embora a API do motor de buscas apresente um número limitado de resultados, o algoritmo não adiciona todos ao conjunto de documentos obtidos através das buscas na web. Após a recuperação dos documentos na web é feita uma análise para encontrar reutilização de texto, para isso foram testadas algumas técnicas: *Word Overlap*, *Query Likelihood*, *Mixture Model* e *Dependence Models*. Além disso, duas novas representações para o fluxo de informações foram examinadas: a distribuição temporal dos resultados e o grafo de links. A distribuição temporal dos documentos recuperados e analisados da web se for construída corretamente, permite mostrar os resultados pela ordem cronológica em que os textos foram datados, permitindo saber quem foi que citou aquele determinado assunto primeiro, ou seja, dá uma dimensão cronológica aos resultados e a análise de links, como é amplamente conhecida, permite estimar a qualidade de páginas da web, neste trabalho ela é aplicada para a detecção de reutilização de texto na web, sendo usada para representar os resultados retornados para o usuário como um grafo $G=(R,E)$, onde R é a lista de textos reutilizados e $E=(doc1, doc2)$ é uma aresta entre duas instâncias de reutilização de texto, se houver pelo menos um link entre dois documentos da lista.

2.2 Comparação de Árvores Sintáticas

A partir da necessidade de armazenar e recuperar dados complexos, por exemplo, imagens, áudio, estruturas genéticas, que tem um custo de acesso exato muito alto, Bueno

e outros (2005) observaram a necessidade de estudar maneiras de acelerar as operações de buscas a estes dados e apresentaram dois algoritmos para agilizar o processo de busca por similaridade de dados indexados em estruturas métricas. Os dois algoritmos propostos utilizam Algoritmos Genéticos, para encontrar os melhores caminhos na árvore, com o objetivo de melhorar e acelerar o conjunto-resposta, o diferencial entre os dois algoritmos apresentados é a função objetivo dos Algoritmos Genéticos, que é usada para calcular a aptidão de um caminho. O primeiro algoritmo proposto utiliza o algoritmo de correção dos caminhos que podem ser cortados, durante a fase de avaliação, quando um gene do cromossomo representa um nó inválido, esse gene é substituído por outro que leve a uma subárvore factível. Esse nó válido é procurado entre os nós “irmãos” do nó inválido. No caso de não haver mais nenhum nó irmão factível, sobe-se um nível na árvore, repetindo o mesmo mecanismo no nível superior, e a subárvore onde foram esgotadas as possibilidades de busca é marcada como inválida, impedindo uma futura tentativa de explorá-la. No caso de encontrar uma subárvore válida, o cromossomo original é modificado, passando a representar um caminho factível e o segundo algoritmo utiliza a aplicação de penalidades no cálculo da função objetivo, quando um gene indica uma subárvore que não seja válida, e não é encontrada nenhuma subárvore válida no mesmo nó, o algoritmo interrompe sua incursão na árvore, marcando a subárvore como inválida e calculando a distância até o nó-índice alcançado na incursão e também é armazenado o nível alcançado pela incursão, pois esse valor será utilizado na aplicação da penalidade no cálculo da função objetivo dos indivíduos. A aptidão de um caminho é calculada baseando-se na “qualidade” dos objetos alcançados e pela quantidade de objetos qualificados para compor o conjunto-resposta. Quando um cromossomo representa um caminho factível, ou seja, um caminho que leva até um nó-folha, o algoritmo genético analisa o nó-folha, qualificando esse nó utilizando sua distância até o objeto de referência para consulta, e contando quantos objetos são selecionados para compor o conjunto-resposta. O peso da seletividade no cálculo da função objetivo é definido dinamicamente, com pequena contribuição nas primeiras gerações do algoritmo genético e com aumento gradativo durante a evolução.

Um sistema baseado na utilização de um *parser* de cobertura ampla para extrair as relações de dependência e um módulo que obtém relações de vinculação léxica a partir do WordNet, são apresentados por Herrera e outros (2005). O trabalho visa comparar se a adequação de subestruturas de árvores de dependência dá melhores provas de vinculação do que a correspondência de texto simplesmente sozinho. A similaridade entre o texto e a hipótese é definida a partir da proporção de nós pertencentes a hipótese que correspondem aos ramos do texto, o percentual tem que ser superior a 50% para ser considerado similar.

2.3 Técnicas para Tratar Variações em Frases ou Palavras

Celikik e Bast (2009) apresentaram um algoritmo para a busca de textos tolerante a erro na Web. Por exemplo, dada uma consulta, o sistema não só recupera os documentos que contem as palavras da consulta, mas também os documentos que apresentam variantes ortográficas das palavras. O algoritmo proposto realiza na sua primeira etapa a filtragem de todas as palavras que são susceptíveis de serem válidas em um conjunto de palavras que não são válidas (*non-words*) usando um método simples que consiste em uma pesquisa em uma tabela hash de um grande dicionário de Inglês com aproximadamente 400.000 palavras com base em listas de palavras a partir do projeto SCOWL (*Spell Checker Oriented Word Lists*). O SCOWL é uma coleção de listas de palavras divididas em vários tamanhos e categorias, destinadas para uso em corretores ortográficos. Logo depois é realizada uma filtragem léxica permutada para gerar variantes das palavras pesquisadas, por exemplo, para a palavra “algorithm”, seriam geradas as seguintes rotações de caracteres: algorithm, alogritm, algorithm, logarithm, etc. Para encontrar a similaridade entre palavras é utilizado o algoritmo *Levenshtein distance* normalizado. Na etapa final, para eliminar os erros ortográficos falso-positivos, são combinados *insights* do modelo de canal ruidoso e *insights* do algoritmo EM.

O Algoritmo *Levenshtein distance* também é usado por Freeman e outros (2006), onde é apresentada uma solução para o problema da correspondência entre nomes pessoais em Inglês para os mesmos representados no alfabeto Árabe, para isto é usado o algoritmo clássico com classes de equivalência entre os caracteres e processos de normalização, Kashani e outros (2007) também utilizam o mesmo algoritmo, para propor um novo método baseado na ortografia para a transliteração automática de nomes próprios do árabe para o Inglês, que explora vários tipos de alinhamentos baseados em letras. A abordagem consiste em três fases: a primeira usa alinhamentos única letra, a segunda fase usa alinhamentos com mais grupos de letras para lidar com sinais diacríticos e vogais em falta na saída do Inglês, e a terceira fase usa fontes de conhecimento diferentes para reparar eventuais erros remanescentes.

2.4 Reconhecimento de Vinculação Textual

O Recognizing Textual Entailment, traduzido para o português, Reconhecimento de Vinculação Textual, é uma tarefa realizada por sistemas para detectar vinculação semântica entre dois textos em linguagem natural, podendo ser usado para sumarização de documentos e também para a recuperação e extração de informação.

Ennals e outros (2010) apresentam uma extensão do navegador Firefox que alerta

o usuário quando a informação que lêem online é contestada por uma fonte que ele pode confiar, ou seja, se o usuário seleciona em uma frase em um site, em seguida, é mostrada uma lista de artigos que apresentam pontos de vista alternativos ao que está sendo dito naquele site. O método para detectar a vinculação entre os textos pesquisados e os artigos que estão no banco de dados é um algoritmo simples usado no lado do cliente chamado de *Local Lexical Matching* (LLM). O algoritmo divide a página em frases, remove as *stopwords*, aplica-se uma derivação da expressão regular, e, em seguida, procuram-se sentenças que contêm todas as *non-stopwords* contidas em uma das paráfrases de uma alegação conhecida. Se o pedido contém uma palavra de negação (por exemplo, não, nunca, não pode), então isso deve a sentença correspondente. Para melhorar o desempenho, ao invés de rodar o algoritmo LLM para cada par de frases e paráfrases, é comparada cada frase com cada paráfrase ao mesmo tempo procurando por palavras na ordem da sua raridade.

Jijkoun e Rijke (2005) propõem um sistema simples, baseado na semelhança lexical, com duas diferentes medidas de similaridade entre as palavras. Para cada par texto/hipótese (T, H), a frase é considerada como um conjunto de palavras e calcula-se a pontuação de similaridade entre as frases sentenças, para verificar a vinculação, é comparado o resultado contra um limite. Duas medidas de similaridades foram experimentadas, a primeira é *Dekang Lin's dependency* e a segunda foi as cadeias léxicas do WordNet, para ambas medidas, as palavras foram convertidas para lemas.

Em um dado par de textos (o texto e a hipótese), o núcleo da abordagem que Kouylekov e Magnini (2005) apresentam é um algoritmo de edição de distância em árvore aplicado nas árvores de dependência do texto e da hipótese. Se a distância (ou seja, o custo das operações de edição), entre as duas árvores é abaixo de certo limiar, empiricamente estimada com dados de treinamento, então, atribui-se uma relação de vinculação entre os dois textos. A vinculação entre os textos é calculada através da comparação da soma dos pesos dos termos que aparecem em ambos os documentos à soma dos pesos de todos os termos.

Wu (2005) investiga dois novos modelos para o problema de RTE (*Textual Entailment Recognition*), que empregam um simples e genérico, *Inversion Transduction Grammar* (ITG). Formalmente o ITG pode ser definido como um conjunto restrito de sintaxe dirigido a transdução de gramáticas, onde todas as regras são de orientação em linha reta ou invertida. A regra ordinária permite que qualquer permutação dos símbolos do lado direito seja especificada na tradução do idioma de entrada para o idioma de saída. Por outro lado, se uma regra é invertida, a ordem é da esquerda para a direita para o idioma de entrada e da direita para a esquerda para a linguagem de saída. Uma vez que a inversão é permitida em qualquer nível de expansão regra, uma derivação pode misturar as produções de uma orientação dentro da árvore de análise. Cada par texto-hipótese

do conjunto de teste foi pontuado pelo algoritmo biparsing, ou seja, cada unidade recebe um peso. O modelo de pontuação pode ser interpretado como uma generalização da seqüência clássica *Levenshtein* de edição de distância, onde transposições de bloco invertidos também são permitidos. Na versão básica, todas as palavras do vocabulário são incluídas entre as transduções lexicais, permitindo a comparação exata entre o texto e a hipótese. A segunda versão exclui uma lista de 172 palavras de um *stoplist* das transduções lexicais. A motivação para este modelo foi para descontar o efeito de tais palavras como “o” ou “de”, pois, na maioria das vezes, poderiam ser irrelevantes para a tarefa de RTE.

2.5 Simplificação Textual

A simplificação textual é tida como o processo de reduzir a complexidade nos diversos níveis do texto (léxico, sintático e discursivo), mantendo sua significação inicial. O objetivo da simplificação é tornar o texto mais compreensível ao usuário humano (ou outro programa de tratamento de texto) através de tratamento automático, e pode ser vista como uma tarefa de tradução do texto de uma forma livre para uma forma simplificada (sintática e lexical), mantendo a semântica do texto tanto quanto possível [Maziero, 2009]. Sentenças longas, com vários níveis de subordinação, cláusulas embutidas (relativas), sentenças na voz passiva, uso da ordem não canônica para os componentes de uma sentença, além do uso de palavras de baixa frequência aumentam a complexidade de um texto para leitores com problemas de compreensão como, por exemplo, analfabetos funcionais, afásicos e disléxicos [Candido, 2009].

As técnicas apresentadas abaixo podem ser usadas para encontrar informações repetidas (uma frase complexa ou uma frase simples).

Candido e outros (2009) apresentam um sistema de autoria criado para permitir aos autores, simplificar os textos durante a sua criação, antes de serem publicados. O sistema é baseado em um Simplificador Sintático Simbólico (SS), que é direcionado a analfabetos funcionais do nível rudimentar como pessoas que possuem capacidade de localizar informações explícitas em textos curtos, como um anúncio ou pequena carta. Um Simplificador Sintático Estatístico (SE) também se encontra em desenvolvimento, sendo direcionado a pessoas alfabetizadas no nível básico, isto é, pessoas que possuem capacidade de localizar informações em textos um pouco mais extensos, podendo realizar pequenas inferências. Enquanto que o SS aplica um conjunto de operações de simplificações pré-definidas para fazer uma oração o mais simples possível, o SE aprende a gerar textos cujas simplificações são mais naturais. Esta “naturalidade” é baseada em um grupo de fatores que são difíceis de definir usando regras manuais, então são aprendidas via métodos de aprendizado de máquina a partir de exemplos de simplificações naturais gerados por

Tabela 2.2: Exemplo de simplificação natural e forte

| | |
|---|---|
| A | As salas de cinema de todo o mundo exibiam uma produção do diretor Joe Dante em que um cardume de piranhas escapava de um laboratório militar e atacava participantes de um festival aquático. Quase 30 anos depois, (...). Mais de 20 pessoas foram mordidas por palometas (<i>Serrasalmus spilopleura</i> , espécie de piranha) que vivem nas águas da barragem Sanchuri. |
| B | As salas de cinema de todo o mundo exibiam uma produção do diretor Joe Dante. Na produção, um cardume de piranhas escapava de um laboratório militar e atacava participantes de um festival aquático. Quase 30 anos depois, (...). Mais de 20 pessoas foram mordidas por palometas que vivem nas águas da barragem Sanchuri. Palometas são <i>Serrasalmus spilopleura</i> , espécie de piranha. |
| C | As salas de cinema de todo o mundo exibiam um filme do diretor Joe Dante. No filme, um cardume de piranhas escapava de um laboratório militar. O cardume de piranhas atacava participantes de um festival aquático. Quase 30 anos depois, (...). Palometas morderam mais de 20 pessoas. As palometas vivem nas águas da barragem Sanchuri. Palometas são <i>Serrasalmus spilopleura</i> , espécie de piranha. |

humanos.

Na Tabela 2.2 é mostrada a versão original de um texto (A), além disso, uma versão onde foi utilizada uma simplificação natural (B) e na outra uma simplificação forte (C) [Gasperin, 2009].

O Simplificador Sintático Simbólico consiste em um conjunto de regras de reescrita sintática aplicadas sobre a saída de um analisador sintático para um dado texto de entrada. A análise sintática permite representar cada sentença do texto de entrada em um formato XML, incluindo informações sobre sua árvore sintática. A interação entre o sistema e o SS é feita via chamadas de scripts. As possíveis operações de simplificação incluídas no arquivo XML são: (a) dividir orações; (b) trocar marcadores discursivos; (c) mudar para a voz ativa; (d) inverter de ordem das cláusulas; (e) mudar a oração para ordem Sujeito-Verbo-Objeto (SVO); (f) topicalizar e de-topicalizar (mover adjuntos adverbiais e cláusulas reduzidas para o início ou fim da sentença). Existe ainda a operação “Não Simplifica”, para casos em que não é necessária nenhuma simplificação. Cada operação é associada a um ou mais fenômenos lingüísticos passíveis de simplificações. Os fenômenos tratados são: voz passiva; aposto; orações coordenadas (e seus subtipos); orações subordinadas (e seus subtipos); orações relativas (e seus subtipos); orações fora da ordem SVO e orações com adverbiais em posição temática [Candido, 2009].

Watanabe e outros (2009) apresentam uma aplicação que realiza a adaptação automática de conteúdos textuais disponíveis em sites e aplicações web em Conteúdo Facilitado, acessível a usuários analfabetos funcionais. Esse Conteúdo Facilitado é um

conteúdo também textual, no entanto sua estrutura será simplificada, conforme definido com base na *Plain Language*, sistemas de simplificação para o Inglês e na análise de corpus de textos reescritos para atender um público mais amplo na Web Brasileira. O tamanho do texto também será reduzido, visto a dificuldade de usuários com baixo nível de alfabetização com textos extensos. Para realizar a elaboração desse conteúdo de forma automática, são utilizadas as operações de “Simplificação” e “Sumarização” textual. Especificamente no sistema, são utilizadas regras de simplificação definidas no Manual de Simplificação Sintática para o Português, criado no projeto PorSimples. Por exemplo, nas orações subordinadas adverbiais temporais que indicam a circunstância de tempo em que ocorre a ação do verbo da oração principal: quando, enquanto, assim que, logo que, até que, depois de, desde que, apenas, mal, sempre que, cada vez que, antes que, etc [Specia, 2008].

Regra: dividir a sentença em duas:

1. Sentença para a cláusula subordinada, eliminado-se o marcador discursivo e adequando-se o tempo verbal se necessário. Pode ser necessário incluir o sujeito constituído do núcleo do termo da oração principal a que se refere à subordinada (ou o termo completo, caso o núcleo não seja suficiente).
2. Sentença original sem a cláusula subordinada. O tempo verbal pode precisar ser alterado.

Exemplos:

Ao conversar com a reportagem na manhã de ontem, Williamson ainda não havia recebido as notícias sobre o depoimento de Buratti e não escondeu sua surpresa com as suspeitas.

Williamson conversou com a reportagem na manhã de ontem. Williamson ainda não havia recebido as notícias sobre o depoimento de Buratti e não escondeu sua surpresa com as suspeitas.

3 FERRAMENTA VISL PARA ANÁLISE SINTÁTICA

A ferramenta para análise sintática descrita neste capítulo é usada no sistema proposto neste trabalho para geração de frases simplificadas ou modificadas.

O VISL, que significa “*Visual Interactive Syntax Learning*”, é um projeto de investigação e desenvolvimento do Instituto de Linguagem e Comunicação (ISK), University of Southern Denmark (SDU) - Campus Odense. Desde setembro de 1996, funcionários e estudantes da ISK tem sido designados para concepção e implementação de ferramentas de gramática com base na Internet para educação e pesquisa. No início do projeto, foram envolvidas quatro línguas: Inglês, Francês, Alemão e Português, em breve a ser seguido por dinamarquês, espanhol e esperanto. Desde então, muitos outros idiomas aderiram ao projeto. A análise automática gramatical de textos em Português do sistema VISL é baseada em um parser *Constraint Grammar* multi-nível (PALAVRAS), desenvolvido por Eckhard Bick.

O analisador morfológico é baseado em um léxico de 50.000 formulários de base, e sua saída é processado por cerca de 5.000 regras de *Constraint Grammar* para desambiguação sintática, morfológica (e - em parte - semântica). *Constraint Grammar* (CG) é um paradigma metodológico para análise de Linguagem Natural (NLP). As regras dependentes do contexto são compiladas em uma gramática que atribui tags gramaticais (“readings”) para palavras ou outros símbolos no texto corrente. Tags típicas endereçam lemmatisation (lexema ou forma base), flexão, derivação, função sintática, dependência, valência, case roles, tipo semântico, etc. Cada regra adiciona, remove, substitui ou seleciona uma tag ou um conjunto de tags gramaticais de acordo com o contexto da sentença. Condições de contexto podem ser vinculadas a qualquer tag ou conjunto de tags de qualquer palavra em qualquer parte da frase, tanto localmente (distâncias definidas) ou globalmente (distâncias indefinidas). CG típico consiste de milhares de regras, que são aplicadas um set-wise em progressivas etapas, abrangendo níveis cada vez mais avançados de análise. Dentro de cada nível, as regras de segurança são usadas antes

de regras heurísticas, e não é permitido remover a última leitura de um determinado tipo, proporcionando assim, um alto grau de robustez [Bick, 2000].

Na tabela 3.1 é apresentada os principais tags de classificação usadas pelo sistema de análise sintática do VISL ¹.

Tabela 3.1: Tags de classificação

| Tag | Significado |
|------|--|
| N | Substantivos |
| PROP | Substantivo Próprio (nomes) |
| SPEC | Especificador (pronome ou adjetivo) |
| DET | Determinantes |
| PERS | Pronome Pessoal |
| ADJ | Adjetivos |
| ADV | Advérbios |
| V | Verbos (todos os verbos, auxiliares) |
| NUM | Numeral (cardinais) |
| PRP | Preposição |
| KS | Conjunção Subordinativa |
| KC | Conjunção Coordenativa |
| IN | Interjeição |
| EC | Prefixo separado por hífen (“elemento composto”) |

O código fonte mostrado na Tabela 3.2, é usado para realizar a conexão com a ferramenta para análise sintática VISL e armazenar o conteúdo da página acessada. Na primeira linha é criado um objeto URL para informar o endereço a ser acessado e na segunda linha do código fonte é feita a conexão, logo depois os dados presentes na página web são armazenados usando um buffer de entrada de dados, para ser usado no pré-processamento da frase inserida pelo usuário na ferramenta apresentada neste trabalho.

Tabela 3.2: Código-fonte Java para conectar ao VISL

```
URL myUrl = URL(http://beta.visl.sdu.dk/visl/pt);
URLConnection myConn = myUrl.openConnection();
BufferedReader br = new BufferedReader( new
InputStreamReader(myConn.getInputStream()));
```

Portanto, a partir da frase inserida pelo usuário e da conexão realizada, o VISL classifica gramaticalmente cada palavra presente na frase, por exemplo, a frase “limão é bom para gastrite” como mostra a Figura3.1 é classificada da seguinte forma:

- limão - (N M S) substantivo masculino no singular;

¹<http://beta.visl.sdu.dk/visl/pt/>

- é – (V PR 3S IND VFIN) verbo no presente, terceira pessoa do singular, indicativo e finito;
- bom – (ADJ M S) adjetivo masculino no singular;
- para –(PRP) proposição;
- gastrite – (N F S) substantivo feminino no singular

Portuguese -> Automatic parse -> Flat structure

SYDDANSK UNIVERSITET Skip | Games | Quizzes | Tools | Sentence Analy

Upgrading from ISO-8859-1 to UTF-8 in progress...things may be unsta

Portuguese VISL

- Overview
- Credits
- Info

Sentence Analysis

- Pre-analyzed
- Pre analyzed sentences
- Floresta Sintá(c)tica
- Floresta symbol set

Machine Analysis

- Flat structure
- Tree structure
- Dependency links
- Complex interface
- Upload interface
- Remote interface

Flat structure

Enter Portuguese text to parse:

limão é bom para gastrite

Parser: Full morphosyntactic parse Visualization: Default

limão [limão] <fruit> **N M S @SUBJ**>
é [ser] <vK> <fmc> **V PR 3S IND VFIN @FMV**
bom [bom] **ADJ M S @<SC**
para [para] **PRP @A<**
gastrite [gastrite] <sick> **N F S @P<**

Figura 3.1: Classificação gramatical de uma frase no VISL

4 FERRAMENTA PARA ANÁLISE DE INFORMAÇÕES TEXTUAIS NA INTERNET BASEADA EM INTELIGÊNCIA COLETIVA

A ferramenta de busca apresentada neste trabalho é baseada no conceito de Inteligência Coletiva, apresentado por Lévy (1998) apud Loh (2007), onde a opinião da maioria das pessoas leva ao que é correto ou ao melhor caminho, para análise de informações textuais na internet.

A partir de uma frase inserida pelo usuário a ferramenta realiza buscas pela frase original, frase negativa, frase com elementos sintáticos trocados e frases com troca de palavras por sinônimos e antônimos. Estas buscas têm o objetivo de encontrar textos que tratam sobre informações semelhantes (contradições ou complementações) em relação à frase originalmente inserida pelo usuário e exibir o número de resultados encontrado para cada uma das frases. A partir da análise dos resultados encontrados pelo sistema para cada frase, o usuário poderá saber se tem mais fontes de informação afirmando ou contrariando determinado assunto.

Uma das possíveis aplicações desta ferramenta é ser usada na busca de informações sobre saúde para fornecer um indicativo de verdade sobre alguma dúvida que uma pessoa tenha sobre determinada doença como tratamento ou prevenção. Portanto, a partir da análise por parte do usuário do número de resultados encontrados, a ferramenta vai ajudar ao usuário a decidir o que verdadeiro e o que falso.

Além disso, a ferramenta pode ser utilizada para detecção de plágio na internet, pois através das frases geradas pelo sistema é possível não só encontrar frases iguais a original, mas também variações, possibilitando ao usuário encontrar algum texto semelhante ao original sem referência, já que este é um grande problema para editores, pesquisadores e educadores.

4.1 Entrada e pré-processamento da frase original

No momento que o usuário entra na página inicial da ferramenta disponível na internet, como mostra a Figura 4.1, e insere uma frase e clica em pesquisar, esta frase é considerada como a frase original. Então ela é colocada entre aspas, para permitir que sejam encontrados textos que contenham frases iguais à original, eliminado o risco de encontrar textos que possuem algumas palavras da frase, mas que não tratam sobre o mesmo assunto referido na frase inserida pelo usuário.

A frase original tanto pode ser uma frase positiva como negativa. No caso de a frase ser positiva, primeiro é feita a busca pela frase original e depois é feita a busca pela frase negativa como é explicado seção 4.2. Quando a frase original é negativa, primeiro então é feita a busca pela frase original e depois a palavra negativa é retirada da frase para fazer uma nova busca, agora pela frase na forma positiva.



Figura 4.1: Tela inicial da ferramenta

4.2 Frase Negativa

Após o processamento da frase original é feita a geração da frase negativa, caso a frase original seja positiva, usando a ferramenta para análise sintática VISL, onde é encontrado o verbo presente na frase. Logo depois é feita a inserção da palavra “não” antes do verbo e a frase é colocada entre aspas, formando a frase negativa em relação à frase original, permitindo encontrar textos que contrapõem a informação originalmente inserida na ferramenta.

4.3 Troca de posições entre sujeito e complemento

Para encontrar resultados que possuem frases que contrapõem a frase inicial sem utilizar a palavra negativa é realizada a troca de elementos sintáticos dentro da frase, ou seja, trocam-se posições das palavras pertencentes à frase, quando são encontrados um sujeito e um complemento. Por exemplo, a frase inicial é “Maradona foi melhor que Pelé”, após encontrar o sujeito e o complemento da frase através da ferramenta VISL é realizada a troca de posições entre eles, formando a seguinte frase “Pelé foi melhor que Maradona”. Além disso, o sistema identifica o verbo da frase e faz uma nova busca com a frase na forma negativa, ou seja, “Pelé não foi melhor que Maradona”.

4.4 Troca por sinônimos

Para encontrar variações da frase que o usuário inseriu no sistema é realizada a busca por frases com palavras substituídas por sinônimos entre aspas. O primeiro passo para realizar o procedimento de substituição é identificar através do VISL quais palavras da frase são verbo, substantivo, adjetivo, advérbio ou nome próprio. Depois da identificação é realizada uma busca para cada palavra classificada no dicionário de sinônimos que está armazenado em um arquivo texto, sendo que para cada palavra substituída é gerada uma nova frase.

4.4.1 Verbos

No caso da substituição dos verbos por seus sinônimos, primeiro é identificado o verbo sem flexão, o tempo verbal e se está no plural ou singular, após é realizada uma busca no dicionário de sinônimos usando o verbo sem flexão. Logo depois de encontrar os sinônimos eles são armazenados e flexionados conforme o verbo original.

Verbos são palavras que expressam uma ação, um estado ou mudança de estado. A correta compreensão do papel do verbo obedece a uma premissa lógica fundamental. Toda ação ou estado tem uma causa e um causador e produz um efeito. O verbo é a classe de palavras mais rica em flexões; assim sendo, o verbo reveste diferentes formas para indicar a pessoa do discurso, o número, o tempo, o modo e a voz.

O verbo sofre variações para indicar a pessoa e o número, como mostra a Figura 4.2.

Os verbos regulares no infinitivo podem ter até três desinências (ar, er e ir). Chamam-se os verbos terminados em -ar, da primeira conjugação; em -er, da segunda conjugação; e, em -ir, da terceira conjugação. Em cada conjugação, o verbo flexiona-se diferentemente. Verbos que não seguem estas conjugações, são chamados de irregulares (ou anômalos), os que seguem, são chamados de regulares. O verbo além de flexionar-se

| | singular | plural |
|--|-----------|--------------|
| 1ª pessoa (aquela que fala) | eu penso | nós pensamos |
| 2ª pessoa (aquela com quem se fala) | tu pensas | vós pensais |
| 3ª pessoa (aquela de quem se fala) | ele pensa | eles pensam |

Figura 4.2: Flexão de número e pessoa

em número e pessoa, também flexiona-se em tempo e modo (seguindo as conjugações). Os modos são três, Indicativo, Subjuntivo ou Conjuntivo e Imperativo.

Neste trabalho, somente são flexionados os verbos regulares do modo indicativo e do modo subjuntivo.

O Modo indicativo apresenta seis tempos verbais, sendo um deles representando o presente do enunciado, três o passado enunciado, que dividem-se em perfeitos (completos) e em imperfeito (incompleto), e dois representando o futuro, um o futuro do enunciado e outro o futuro do passado do enunciado. Este último (denominado Futuro do Pretérito) apresenta características especiais, muitas vezes semelhantes ao modo subjuntivo.

Exemplo das flexões deste modo, em tempos:

- Presente (eu levo, tu levavas, ele leva, nós levamos, vós levais, eles levam);
- Pretérito Imperfeito (eu levava, tu levavas, ele levava, nós levávamos, vós leváveis, eles levavam);
- Pretérito Perfeito (eu levei, tu levaste, ele levou, nós levamos, vós levastes, eles levaram);
- Pretérito Mais-que-Perfeito (Eu levaria, tu levarias, ele levaria, nós levaríamos, vós levaríeis, eles levariam);
- Futuro do Presente (eu levarei, tu levarás, ele levará, nós levaremos, vós levareis, eles levarão);
- Futuro do Pretérito (ou Condicional - eu levaria, tu levarias, ele levaria, nós levaríamos, vós levaríeis, eles levariam).

Para realizar as flexões dos verbos (sinônimos) no modo indicativo para o tempo verbal do verbo original da frase, primeiro é identificado à terminação, ou seja, “ar”, “er” ou “ir”. Logo depois é feita a substituição de acordo com a terminação para aquele tempo verbal conforme as suas regras: presente, conforme a Figura 4.3, pretérito perfeito, conforme a Figura 4.4, pretérito imperfeito, conforme a Figura 4.5, pretérito mais

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|----------------|----------------|----------------|
| eu | <i>falo</i> | <i>como</i> | <i>abro</i> |
| tu | <i>falas</i> | <i>comes</i> | <i>abres</i> |
| você, ele, ela | <i>fala</i> | <i>come</i> | <i>abre</i> |
| nós | <i>falamos</i> | <i>comemos</i> | <i>abrimos</i> |
| vós | <i>falais</i> | <i>comeis</i> | <i>abris</i> |
| vocês, eles, elas | <i>falam</i> | <i>comem</i> | <i>abrem</i> |

Figura 4.3: Presente do modo indicativo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|-----------------|-----------------|-----------------|
| eu | <i>falei</i> | <i>comi</i> | <i>abri</i> |
| tu | <i>falaste</i> | <i>comeste</i> | <i>abriste</i> |
| você, ele, ela | <i>falou</i> | <i>comeu</i> | <i>abriu</i> |
| nós | <i>falamos</i> | <i>comemos</i> | <i>abrimos</i> |
| vós | <i>falastes</i> | <i>comestes</i> | <i>abristes</i> |
| vocês, eles, elas | <i>falaram</i> | <i>comeram</i> | <i>abriram</i> |

Figura 4.4: Pretérito perfeito do modo indicativo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|------------------|-----------------|-----------------|
| eu | <i>falava</i> | <i>comia</i> | <i>abria</i> |
| tu | <i>falavas</i> | <i>comias</i> | <i>abrias</i> |
| você, ele, ela | <i>falava</i> | <i>comia</i> | <i>abria</i> |
| nós | <i>falávamos</i> | <i>comíamos</i> | <i>abríamos</i> |
| vós | <i>faláveis</i> | <i>comíeis</i> | <i>abríeis</i> |
| vocês, eles, elas | <i>falavam</i> | <i>comiam</i> | <i>abriam</i> |

Figura 4.5: Pretérito imperfeito do modo indicativo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|------------------|------------------|------------------|
| eu | <i>falara</i> | <i>comera</i> | <i>abrirá</i> |
| tu | <i>falaras</i> | <i>comeras</i> | <i>abrirás</i> |
| você, ele, ela | <i>falara</i> | <i>comera</i> | <i>abrirá</i> |
| nós | <i>faláramos</i> | <i>coméramos</i> | <i>abríramos</i> |
| vós | <i>faláreis</i> | <i>coméreis</i> | <i>abríreis</i> |
| vocês, eles, elas | <i>falaram</i> | <i>comeram</i> | <i>abriram</i> |

Figura 4.6: Pretérito mais que perfeito do modo indicativo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|------------------|------------------|------------------|
| eu | <i>falarei</i> | <i>comerei</i> | <i>abrirei</i> |
| tu | <i>falarás</i> | <i>comerás</i> | <i>abrirás</i> |
| você, ele, ela | <i>falará</i> | <i>comerá</i> | <i>abrirá</i> |
| nós | <i>falaremos</i> | <i>comeremos</i> | <i>abriremos</i> |
| vós | <i>falareis</i> | <i>comereis</i> | <i>abrireis</i> |
| vocês, eles, elas | <i>falarão</i> | <i>comerão</i> | <i>abrirão</i> |

Figura 4.7: Futuro do presente do modo indicativo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|-------------------|-------------------|-------------------|
| eu | <i>falaria</i> | <i>comeria</i> | <i>abriria</i> |
| tu | <i>falarias</i> | <i>comerias</i> | <i>abririas</i> |
| você, ele, ela | <i>falaria</i> | <i>comeria</i> | <i>abriria</i> |
| nós | <i>falaríamos</i> | <i>comeríamos</i> | <i>abriríamos</i> |
| vós | <i>falaríeis</i> | <i>comeríeis</i> | <i>abriríeis</i> |
| vocês, eles, elas | <i>falariam</i> | <i>comeriam</i> | <i>abririam</i> |

Figura 4.8: Futuro do pretérito do modo indicativo

que perfeito, conforme a Figura 4.6, futuro do presente, conforme a Figura 4.7 e futuro do pretérito, conforme a Figura 4.8.

O modo subjuntivo ou conjuntivo exprime dúvida, desejo. Apresenta três tempos bem distintos um dos outros. O passado deste tempo é o mais utilizado, junto à conjunção *Se*. O presente e o futuro não são tão utilizados, utiliza-se *Que* e *Quando*, respectivamente.

Os tempos do Modo Subjuntivo são:

- Pretérito Imperfeito (*Se eu levasse, se tu levasse, se ele levasse, se nós levássemos, se vós levásseis, se eles levassem*);
- Presente (*Que eu leve, que tu leves, que ele leve, que nós levemos, que vós leveis, que eles levem*);
- Futuro (*Quando eu levar, quando tu levares, quando ele levar, quando nós levarmos, quando vós levardes, quando eles levarem*).

Para realizar as flexões dos verbos (sinônimos) no modo indicativo para o tempo verbal do verbo original da frase, primeiro é identificado a terminação, ou seja, “ar”, “er” ou “ir”. Logo depois é feita a substituição de acordo com a terminação para aquele tempo verbal conforme as suas regras: pretérito imperfeito, conforme a Figura 4.9, presente, conforme a Figura 4.10 e futuro subjuntivo, conforme a Figura 4.11.

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|-------------------|-------------------|-------------------|
| eu | <i>falasse</i> | <i>comesse</i> | <i>abrisse</i> |
| tu | <i>falasses</i> | <i>comesse</i> | <i>abrisse</i> |
| você, ele, ela | <i>falasse</i> | <i>comesse</i> | <i>abrisse</i> |
| nós | <i>falássemos</i> | <i>coméssemos</i> | <i>abrissemos</i> |
| vós | <i>falásseis</i> | <i>comésseis</i> | <i>abrisseis</i> |
| vocês, eles, elas | <i>falassem</i> | <i>comessem</i> | <i>abrissem</i> |

Figura 4.9: Pretérito imperfeito do modo subjuntivo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|----------------|----------------|----------------|
| eu | <i>fale</i> | <i>coma</i> | <i>abra</i> |
| tu | <i>fales</i> | <i>comas</i> | <i>abras</i> |
| você, ele, ela | <i>fale</i> | <i>coma</i> | <i>abra</i> |
| nós | <i>falemos</i> | <i>comamos</i> | <i>abramos</i> |
| vós | <i>faleis</i> | <i>comais</i> | <i>abrais</i> |
| vocês, eles, elas | <i>falem</i> | <i>comam</i> | <i>abram</i> |

Figura 4.10: Presente do modo subjuntivo

| | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
|-------------------|-----------------|-----------------|-----------------|
| eu | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
| tu | <i>falares</i> | <i>comeres</i> | <i>abrires</i> |
| você, ele, ela | <i>falar</i> | <i>comer</i> | <i>abrir</i> |
| nós | <i>falarmos</i> | <i>comermos</i> | <i>abrirmos</i> |
| vós | <i>falardes</i> | <i>comerdes</i> | <i>abrirdes</i> |
| vocês, eles, elas | <i>falarem</i> | <i>comerem</i> | <i>abrirem</i> |

Figura 4.11: Futuro do modo subjuntivo

4.4.2 Substantivos, Adjetivos e Advérbios

Para substantivos, adjetivos e advérbios presentes na frase, primeiramente, é feita a classificação em gênero e pessoa, depois são buscados no dicionário os sinônimos e flexionados conforme o gênero e pessoa da palavra original. Por exemplo, para tornar uma palavra masculina terminada em “o” para feminina é preciso substituir por “a” e para passar o sinônimo encontrado para o plural, no caso em que a terminação é com vogais é só acrescentar “s”.

As regras usadas para transformar uma palavra no singular para o plural são as seguintes:

- Palavras terminadas em vogais acrescenta-se “s”;
- Palavras terminadas em “r” ou “z” acrescenta-se “es”;
- Palavras terminadas em “n” acrescenta-se “s”;
- Palavras terminadas em “al”, “el”, “ol” ou “ul” acrescenta-se “is”;

4.4.3 Dicionário

O dicionário de sinônimos utilizado neste trabalho é do projeto OpenThesaurus ¹ e apresenta 4.000 linhas de sinônimos, conforme a Figura 4.12, porém os usuários podem sugerir novos sinônimos através do link “Sinônimos/Antônimos”.

| | |
|----|--|
| 1 | ab-rogar, abortar, anular, cancelar, cassar, derrogar, eliminar, infirmar, invalidar, rescindir, revogar, terminar |
| 2 | ababelado, atarantado, atrapalhado, baralhado, desnortado |
| 3 | ababelar, atrapalhar, baralhar |
| 4 | abadessa, prelada, prioresa, superiora |
| 5 | abadia, Á;dito, basilica, catedral, convento, igreja, mosteiro, presbitÁ©rio, santuÁ;rio, sÁ©, templo |
| 6 | abafadiÁ©, abafante, asfixiante, sufocante |
| 7 | abafado, abafado, abafado, abrasado, abrasado, agasalhado, aquecido |
| 8 | abafado, abafado, abafado, asfixiado, sufocado |
| 9 | abafado, abafado, abafado, empalmado, escamoteado, furtado, larapiado, rapinado, roubado, surripiado |
| 10 | abafar, abrasar, agasalhar, aquecer |
| 11 | abafar, arrebatado, empalmar, escamotear, furtar, larapiar, rapinar, roubar, saquear, surripiar, tirar |
| 12 | abafar, dominar, subjugar, vencer, ganhar, derrotar |
| 13 | abafaÁ©, abafamento, asma, sufocaÁ© |
| 14 | abafaÁ©, ocultaÁ©, sonegaÁ© |
| 15 | abafo, agasalho |
| 16 | abaixado, acabrunhado, achatado, achatado, aperreado, humilhado, oprimido, vexado |
| 17 | abaixamento, abatimento, abatimento, abatimento, abatimento, depressÁ©, humilhaÁ© |
| 18 | abaixar, agachar, arriar, baixar, descair, descer |
| 19 | abaixo-assinado, petiÁ©, requerimento, solicitaÁ©, subscriÁ© |
| 20 | abaixo, aquÁ©, inferior |
| 21 | abajur, lucivelo, pantalha, quebra-luz |
| 22 | abalada, partida, saída |
| 23 | abaladiÁ©, bambo, flÁ;cido, frouxo, inseguro, instÁ;vel, lasso, oscilante, suxo, vacilante |
| 24 | abalado, aluido, comovido, exitante, impressionado, inseguro |
| 25 | abalanÁ©, compensar, contrabalanÁ©, equilibrar, igualar |
| 26 | abalar, abalar, combalir, elanguescer, enfraquecer |
| 27 | abalar, abalar, fugir, partir, zarpar |

Figura 4.12: Dicionário de sinônimos

4.5 Troca por antônimos

A troca de palavras da frase por antônimos segue os mesmos procedimentos da troca por sinônimos, mudando apenas o dicionário que também se encontra armazenado em um arquivo texto, conforme a Figura 4.13, no próprio servidor em que a ferramenta está hospedada.

Os usuários da ferramenta podem sugerir novos antônimos que o dicionário não tenha, através do próprio site, acessando o link “Sinônimos/Antônimos”.

¹<http://openthesaurus.caixamagica.pt/>

| | |
|----|------------------------------|
| 1 | aberto, fechado |
| 2 | fechado, aberto |
| 3 | alto, baixo |
| 4 | baixo, alto |
| 5 | bem, mal |
| 6 | mal, bem |
| 7 | bom, mau, ruim |
| 8 | mau, bom |
| 9 | ruim, bom |
| 10 | bonito, feio |
| 11 | feio, bonito, lindo |
| 12 | mais, menos |
| 13 | menos, mais |
| 14 | doce, amargo, salgado, azedo |
| 15 | amargo, doce |
| 16 | azedo, doce |
| 17 | salgado, doce |
| 18 | dentro, fora |
| 19 | fora, dentro |
| 20 | gordo, magro |
| 21 | magro, gordo |
| 22 | seco, molhado |
| 23 | molhado, seco |
| 24 | grosso, fino |
| 25 | fino, grosso |
| 26 | duro, mole |
| 27 | mole, duro |
| 28 | |

Figura 4.13: Dicionário de antônimos

4.6 Buscas na internet

Para realizar as buscas pelas frases geradas e pela frase original a ferramenta se conecta ao Google usando a classe URL do Java e retorna os resultados encontrados para cada uma destas frases, exibindo-os para o usuário, além disso, ao lado de cada resultado é disponibilizado o link da página de resultados do Google para facilitar a conferência dos resultados encontrados.

A classe URL, como mostra a Tabela 4.1, permite o acesso aos dados de uma página web a um nível mais alto do que os Sockets. O armazenamento dos dados acessados é feito através da classe BufferedReader, que é um buffer de fluxo de entrada de dados.

Tabela 4.1: Código-fonte Java para conectar ao Google

```
URL myUrl = new URL("http://www.google.com.br/searchhl=pt-
BR&rlz1B3GGGL_ptBRBR238BR280&q=%22"+pesquisa+"%22");
URLConnection myConn = myUrl.openConnection();
myConn.setRequestProperty("User-agent", "text/xml");
BufferedReader br = new BufferedReader( new
InputStreamReader(myConn.getInputStream(), "UTF-8"));
```

4.7 Testes

Nesta seção, serão apresentados alguns testes realizados utilizando diferentes frases, ou seja, frase original na forma positiva e na forma negativa, frase com su-

jeito e complemento e frases onde poderia ser feita a troca de palavras por sinônimos e antônimos. A Figura 4.14 mostra os resultados encontrados pela ferramenta para a frase original “beterraba ajuda a retardar envelhecimento”. Através da análise das frases geradas pelo sistema e do número de resultados encontrados para cada uma delas, podemos destacar o seguinte:

- A frase negativa foi gerada através da inserção da palavra “não” antes do verbo “ajuda”, portanto, a ferramenta elaborou a frase negativa de forma correta;
- As frases com elementos sintáticos trocados não foram gerados, pois a frase “beterraba ajuda a retardar envelhecimento” não apresenta complemento;
- As frases com troca de palavra por sinônimos foram construídas com a troca da palavra “ajuda” pelos seus sinônimos: acolita, acude, auxilia, coadjuva, salva e socorre;
- Frase com palavras trocadas por antônimos não foram geradas pois nenhum das palavras presentes na frase possui antônimos no dicionário de antônimos;
- Para a frase original foram encontrados 96 resultados, enquanto que para as outras frases não foram encontrados resultados. Portanto, podemos dizer que tem mais pessoas afirmando ou confirmando esta informação do que negando.

Resultados

Frase Original: beterraba ajuda a retardar envelhecimento
Resultados: 96 [ver resultados no Google](#)

Frase Negativa: beterraba não ajuda a retardar envelhecimento
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com elementos sintáticos trocados(+): envelhecimento ajuda a retardar beterraba
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com elementos sintáticos trocados(-): envelhecimento não ajuda a retardar beterraba
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): beterraba acolita a retardar envelhecimento
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): beterraba acude a retardar envelhecimento
Resultados: 0 [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): beterraba auxilia a retardar envelhecimento
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): beterraba coadjuva a retardar envelhecimento
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Figura 4.14: Busca pela frase “beterraba ajuda a retardar envelhecimento”

Na Figura 4.15, a frase pesquisada é “Maradona foi melhor que Pelé”, neste exemplo, podemos observar o seguinte:

- Na frase negativa, a palavra “não” foi inserida antes do verbo “foi”, transformando a frase original na frase negativa de forma correta;
- As frases com elementos sintáticos trocados foram construídas devido a frase original apresentar sujeito e complemento, sendo “Maradona” o sujeito e “Pelé” o complemento. É possível notar que mesmo com a troca, a frase não perdeu concordância e formou uma frase negativa em relação à frase original sem utilizar a palavra “não”;
- Na segunda frase com elementos sintáticos trocados, foi inserida a palavra “não” antes do verbo para criar mais uma variante da frase inicialmente inserida pelo usuário;
- Outra frase construída pela sistema foi a frase com troca por antônimos, neste caso a troca da palavra “melhor” por “pior”;
- Através da análise dos resultados encontrados, podemos obter indícios de que Maradona foi melhor que Pelé, pois 2.044 pessoas afirmam a informação e 952 são contrários a esta informação.

The screenshot shows a search engine interface. At the top, there is a search bar with the text "Pesquisar" and a search button. Below the search bar, the results are displayed in a vertical sidebar labeled "Resultados". The results are as follows:

| Search Query | Number of Results |
|--|-------------------|
| Frase Original: Maradona foi melhor que Pelé | 2240 |
| Frase Negativa: Maradona não foi melhor que Pelé | 661 |
| Frase com elementos sintáticos trocados(+): Pelé foi melhor que Maradona | 342 |
| Frase com elementos sintáticos trocados(-): Pelé não foi melhor que Maradona | 4 |

Figura 4.15: Busca pela frase “Maradona foi melhor que Pelé”

Na Figura 4.16, a frase original está na forma negativa, ou seja, “Petrobras não encontrou petróleo”. Podemos destacar o seguinte:

- A frase original é negativa, portanto, neste caso é realizada a transformação da frase original para a forma positiva, sendo retirada a palavra “não”;
- A troca por sinônimos foram realizadas em duas palavras da frase, na palavra “encontrou”, que foi substituída por “achou” e “descobriu”. E a palavra “não” que foi substituída por “jamais” e “nunca”.
- As frases com elementos sintáticos trocados não foram gerados, pois a frase “Petrobras não encontrou petróleo” não apresenta complemento;
- Através da análise dos resultados encontrados, podemos dizer que tem mais pessoas afirmando que a Petrobras encontrou petróleo do que negando, pois são 2.170 resultados para a frase positiva e 120 resultados para as frases negativas, ou seja, frase “Petrobras não encontrou petróleo” e “Petrobras não achou petróleo”.

Pesquisar

Resultados

Frase Original: Petrobras não encontrou petróleo
Resultados: 119 [ver resultados no Google](#)

Frase Positiva: Petrobras encontrou petróleo
Resultados: 2170 [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): Petrobras não achou petróleo
Resultados: 1 [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Verbo): Petrobras não descobriu petróleo
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Advérbio): Petrobras jamais encontrou petróleo
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Frase com troca de palavra por sinônimo (Advérbio): Petrobras nunca encontrou petróleo
Resultados: Nenhum resultado encontrado [ver resultados no Google](#)

Figura 4.16: Busca pela frase “Petrobras não encontrou petróleo”

A partir dos resultados encontrados e exibidos pela ferramenta, que é baseada no conceito de Inteligência Coletiva, os usuários poderão ter um indicativo se tem mais pessoas afirmando ou contrariando uma informação, ajudando-os a ter uma ideia do que seja o mais correto, já que opinião da maioria das pessoas leva ao que é correto ou ao melhor caminho, segundo Surowiecki (2004) apud Loh (2007). Portanto, é o usuário que vai decidir se vai aceitar ou não o que os resultados encontrados apontam como um indício de veracidade.

4.8 Tecnologias

Para o desenvolvimento desta ferramenta de busca foi utilizada a linguagem JSP, que é a linguagem Java para web, na parte dinâmica e na parte estática a linguagem CSS.

4.8.1 JSP

No final de 1999, a Sun Microsystems adicionou um novo elemento para a coleção de ferramentas do *Java 2 Enterprise Edition (J2EE)*, o *Java Server Pages (JSP)*.

JSP é uma tecnologia para desenvolvimento de aplicações web do lado do servidor, que usa Java como sua linguagem de scripts, portanto, podendo ser aplicados todos os conceitos de reutilização de componentes, desacoplamento e encapsulamento.

As páginas JSP, são páginas HTML que incluem código Java e outros tags especiais, dessa forma, permitindo separar a programação lógica (parte dinâmica) da programação visual (parte estática), facilitando o desenvolvimento de aplicações mais robustas, onde programador e designer podem trabalhar no mesmo projeto, mas de forma independente.

O desenvolvimento de aplicações em JSP requer o conhecimento tanto da linguagem Java como a linguagem HTML, para a estruturação da página JSP.

A Figura 4.17 mostra um esquema das etapas de execução de um página JSP na primeira vez que é requisitada. Na etapa (1) a requisição é enviada para um servidor Web que reencaminha a requisição (etapa 2) para o container Servlet/JSP. Na etapa (3) o container verifica que não existe nenhuma instância de Servlet correspondente à página JSP. Neste caso, a página JSP é traduzida para código fonte de uma classe Servlet que será usada na resposta à requisição. Na etapa (4) o código fonte do Servlet é compilado, e na etapa (5) é criada uma instância da classe. Finalmente, na etapa (6) é invocado o método `service()` da instancia Servlet para gerar a resposta à requisição.

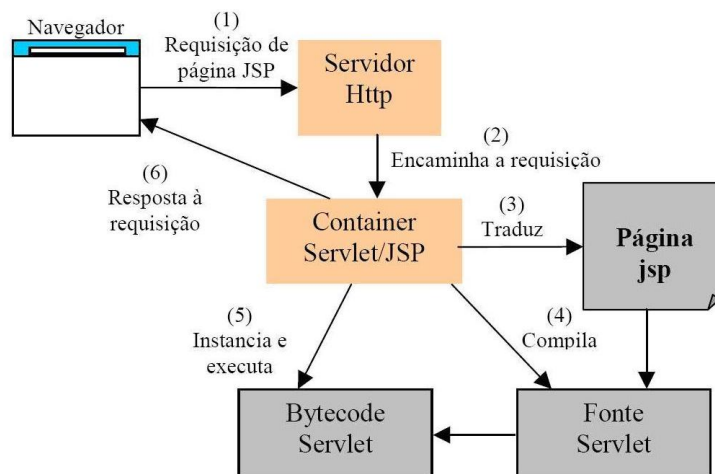


Figura 4.17: Processamento de uma página JSP

Na Tabela 4.2 é exibido um exemplo de código-fonte JSP para se ter uma ideia de como fica o código Java dentro do código HTML.

Tabela 4.2: Exemplo de código JSP

```
<html>
  <head>
    <title>Primeira página JSP!</title>
  </head>
  <body>
    <%
      String x = "Olá Mundo!";
    %>
    <%=x%>
  </body>
</html>
```

As conexões com o Google e a ferramenta VISL são realizadas através da classe URL da linguagem Java. Esta classe permite o acesso aos dados de uma página web a um nível mais alto do que os Sockets. O armazenamento dos dados acessados é feito através da classe BufferedReader, que é um buffer de fluxo de entrada de dados.

O código fonte mostrado na Tabela 4.3 é um exemplo de como é feita uma conexão com um site e como é armazenado seu conteúdo. Na primeira linha é criado um objeto URL para informar o endereço a ser acessado e na segunda linha do código fonte é feita a conexão, logo depois é feito o armazenamento dos dados presentes na página web usando um buffer de entrada de dados.

Para desenvolver uma aplicação em JSP é necessária a instalação do kit de desenvolvimento JSDK, disponibilizado pela Sun Microsystems, um container web, por exemplo, o Apache Tomcat e um bloco de notas.

Tabela 4.3: Exemplo de código Java para conexões com sites

```
URL myUrl = new URL("http://www.google.com.br");
URLConnection myConn = myUrl.openConnection();
myConn.setRequestProperty("User-agent", "Mozilla/4.0");
BufferedReader br = new BufferedReader(new
InputStreamReader(myConn.getInputStream()));
```

4.8.2 CSS

A linguagem CSS é a sigla em inglês para *Cascading Style Sheet* que em português foi traduzido para folha de estilo em cascata, é definida como uma linguagem para estilos que define o layout de documentos HTML. Por exemplo, CSS controla fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, posicionamentos e muito mais. A

diferença entre o HTML e CSS é que o primeiro é usado para estruturar conteúdos e o segundo é usado para formatar conteúdos estruturados.

CSS foi inventada para colocar à disposição dos webdesigners meios sofisticados de projetar layouts suportados por todos os navegadores. E ao mesmo tempo a separação dos estilos de apresentação da marcação dos conteúdos torna a manutenção dos sites bem mais fácil.

A estrutura dos estilos é bastante simples. Consiste de uma lista de regras. Cada regra possui um bloco, entre chaves (`{` e `}`), de uma ou mais declarações aplicáveis a um ou mais seletores. Um seletor é algo no qual se pode aplicar um estilo. Pode ser um descritor HTML, uma hierarquia de descritores ou um atributo que identifique um grupo de descritores. Uma folha de estilos consiste de uma ou mais linhas de regras, da seguinte forma, como mostrado na Figura 4.18.



Figura 4.18: Regra de estilo CSS

A aplicação dos estilos CSS usada foi a de lincar vários documentos HTML para uma mesma folha de estilos. Em outras palavras isto significa que um simples arquivo será capaz de controlar a apresentação de muitos documentos HTML.

Esta técnica pode economizar uma grande quantidade de trabalho. Se por exemplo, você quiser trocar a cor do fundo de um site com 100 páginas, a folha de estilos evita que você edite manualmente uma a uma as páginas para fazer a mudança nos 100 documentos HTML. Usando CSS a mudança se fará em uns poucos segundos trocando-se a cor em uma folha de estilos central.

5 CONSIDERAÇÕES FINAIS

O estudo de trabalhos relacionados às áreas de interesse do trabalho permitiu conhecer diferentes métodos, algoritmos e funções utilizadas para encontrar similaridade entre textos. Além disso, permitiu avaliar quais métodos poderiam ser usados no presente trabalho para encontrar informações semelhantes na internet.

A ferramenta para análise sintática VISL foi de grande importância, pois foi através dela que se pode identificar a classificação gramatical das palavras e com isto gerar variações da frase original como a inserção da palavra negativa, a troca de palavras por sinônimos e antônimos e a troca de elementos dentro da frase.

A escolha da linguagem JSP para o desenvolvimento da ferramenta foi feita por ser uma linguagem Java que apresenta diversas vantagens, entre elas: portabilidade, robustez, orientação a objetos, segurança e é livre.

A ferramenta fornece aos seus usuários um indicativo de verdade sobre informações contraditórias baseando-se no conceito de Inteligência Coletiva. É importante ressaltar que no caso de busca por informações sobre saúde a ferramenta não tem o objetivo de substituir a orientação médica, ou seja, ela serve apenas como um mecanismo de consulta para que o usuário tenha um indicativo se mais pessoas estão afirmando ou contrariando uma determinada informação.

A elaboração deste trabalho desde a fase inicial de leitura de trabalhos correlatos até a fase de desenvolvimento da ferramenta trouxe grande aprendizado e satisfação de estar fazendo um trabalho que pode ajudar pessoas a discernir entre o que é verdadeiro e o que é falso, através da análise por parte do usuário dos números de resultados encontrados e também a encontrar plágio na internet, já que este é um problema que precisa ser combatido.

A maior dificuldade encontrada durante o desenvolvimento da ferramenta foram os problemas de codificação dos caracteres, pois a ferramenta foi desenvolvida no sistema operacional Windows e quando a ferramenta foi colocada no servidor que roda sobre o sistema operacional Linux, gerou problemas na codificação dos caracteres. Para isto, quando a frase original apresenta caracteres como acento ou cedilha é feita a troca de

caracteres, por exemplo, na frase original tem "Ã©", então é trocado por "é" para que a frase possa ser enviada para o VISL, sem apresentar problemas no momento de sua classificação.

Para trabalhos futuros fica como sugestão o uso do sistema Simplifica para gerar frases ou textos simplificados e fazer buscas com as frases traduzidas para outras línguas e com isso observar se tem fontes de informações de outros países afirmando ou contrariando uma informação.

REFERÊNCIAS

- BATU, T., ERGÜN, F., KILIAN, J., MAGEN, A., RASKHODNIKOVA, S., RUBINFELD, R., SAMI, R. (2003). A sublinear algorithm for weakly approximating edit distance. In 35th Annual ACM Symposium on Theory of Computing, pages 316-324.
- BENDERSKY, M., CROFT, W.B. (2009). Finding text reuse on the web. In WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining, pages 262–271, New York, NY, USA. ACM.
- BUENO, R., TRAINA, A. J. M., TRAINA JR., Caetano. (2005). Accelerating approximate similarity queries using genetic algorithms. Proceedings of the ACM Symposium on Applied Computing, 1:617 - 622.
- CANDIDO JR. A, OLIVEIRA, M., ALUÍSIO, S.M. (2009). Simplifica: um Sistema Web de Autoria de Textos Simplificados. In the Proceedings of WEBMEDIA 2009 – Simpósio Brasileiro de Sistemas Multimídia e Web, Fortaleza - CE. v. 2. p. 55 - 58.
- CELIKIK, M., BAST, H. (2009). Fast Error-Tolerant Search on Very Large Texts. In: Symposium on Applied Computing: Proceedings of the 2009 ACM Symposium on Applied Computing, New York, NY, USA, ACM 1724–1731.
- ENNALS, R., TRUSHKOWSKY, B., AGOSTA, J.M., RATTENBURY, T., HIRSCH, T. (2010). Highlighting Disputed Claims on the Web. In Proceedings of the 19th international conference on World Wide Web - WWW 2010.
- FREEMAN, Andrew T., CONDON, Sherri L., ACKERMAN, Christopher M. (2006). ‘Cross Linguistic Name Matching in English and Arabic: A “One to Many Mapping” Extension of the Levenshtein Edit Distance Algorithm’, HLT-NAACL, New York, NY.
- GASPERIN, C., SPECIA, L., PEREIRA, T., ALUÍSIO, S.M. (2009). Learning When to Simplify Sentences for Natural Text Simplification. In: Proceedings of the Encontro Nacional de Inteligência Artificial (ENIA), Bento Gonçalves, Brazil, 809-818.
- HERRERA, J., PEAS, A., VERDEJO, F. (2005). Textual Entailment Recognition Based on Dependency Analysis and WordNet. In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK.

- IRVING, R. (2004). Plagiarism and collusion detection using the Smith-Waterman algorithm. Tech. Rep. TR-2004-164, University of Glasgow, Computing Science Department Research Report.
- JIJKOUN, V., RIJKE, M. de. (2005). Recognizing textual entailment using lexical similarity. In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK, pages 73–76.
- KASHANI, Mehdi M., POPOWICH, Fred, SARKAR, Anoop. (2007). Automatic Transliteration of Proper Nouns from Arabic to English, Proceedings of the Second Workshop on Computational Approaches to Arabic Script-based Languages.
- KOUYLEKOV, M., MAGNINI, B. (2005). Recognizing Textual Entailment with Tree Edit Distance Algorithms. In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK, pages 17.20.
- LANCASTER, T., CULWIN, F. (2005). Classifications of Plagiarism Detection Engines. *ITALICS* Vol. 4 (2).
- LOH, S., LORENZI, Fabiana, GARCIA, Luís Fernando Fortes, FRANCESCHI, Analucia Schiaffino Morales de, BORGES, Karen. (2007) Inteligência coletiva com sistemas multiagente inteligentes. *Logos (Canoas)*. , v.18, p.1.
- LÉVY, Pierre (1998) *A inteligência coletiva: por uma antropologia do ciberespaço*. São Paulo: ed. Loyola.
- LU, J., HAN J., MENG, X. (2009). Efficient algorithms for approximate member extraction using signature-based inverted lists. *The 18th ACM Conference on Information and Knowledge Management*: 315-324.
- LUKASHENKO, R., GRAUDINA, V., GRUNDSPENKIS, J. (2007). Computer-Based Plagiarism Detection Methods and Tools: An Overview. *Proceeding of the International Conference on Computer Systems and Technologies- CompSysTech'07*, Rouse, Bulgaria, pp. IIIA.18-1 – IIIA.18-6.
- MAZIERO, E.G.; PARDO, T.A.S.; ALUÍSIO, S.M. (2009). Ferramenta Automática de Simplificação Textual. In the Proceedings of the STIL Student Workshop on Information and Human Language Technology - TILic, pp. 1-4. September 9, São Carlos/SP, Brazil.
- RUDDLE, R. A. (2006). Using String-matching to Analyze Hypertext Navigation. *Proceedings of the 17th ACM Conference on Hypertext and Hypermedia (HT'06)*, pp. 49-52. New York: ACM.
- SEO, J., CROFT, W.B. (2008). Local text reuse detection. In: *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, ACM 571–578.

SPECIA, L.; ALUÍSIO, S.M.; PARDO, T.A.S. (2008). Manual de Simplificação Sintática para o Português. Série de Relatórios do NILC. NILC-TR-08-06. São Carlos-SP, Junho, 27p.

SUROWIECKI, James (2004). The wisdom of crowds: why the many are smarter than the few and how collective wisdom shapes business, economies, societies and nations. Little, Brown.

VERNICA, R., LI, C. (2009). Efficient top-k algorithms for fuzzy search in string collections. The First International Workshop on Keyword Search on Structured Data: 9-14.

WATANABE, W.M.; FORTES, R.P.M.; PARDO, T.A.S.; ALUÍSIO, S.M. (2009). Facilita: auxílio à leitura de textos disponíveis na Web. In the Proceedings of the Brazilian Symposium on Multimedia and the Web - Webmedia, Vol. 2, pp. 27-30. October 5-7, Fortaleza/CE, Brazil.

WU, Dekai. (2005). Textual entailment recognition based on inversion transduction grammars. In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK, pages 37-40.

Zero Hora. (2010). Cerca de 40% dos internautas usam web para saber mais sobre a própria saúde. Disponível por WWW em <http://www.clicrbs.com.br/especial/rs/bem-estar/19,0,3086608,Cerca-de-40-dos-internautas-usam-web-para-saber-mais-sobre-a-propria-saude.html>, acessado em novembro/2010.