



**EUDES DE CASTRO LIMA**

**ANÁLISE DE TÉCNICAS E FERRAMENTAS DE  
DETECÇÃO DE PLÁGIO, E DESENVOLVIMENTO  
DE UM PROTÓTIPO DE NOVA FERRAMENTA.**

**LAVRAS - MG  
2011**

**EUDES DE CASTRO LIMA**

**ANÁLISE DE TÉCNICAS E FERRAMENTAS DE DETECÇÃO DE  
PLÁGIO, E DESENVOLVIMENTO DE UM PROTÓTIPO DE NOVA  
FERRAMENTA.**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel em Sistemas de Informação.

Orientador  
Prof. Dr. Antônio Maria Pereira de Resende

**LAVRAS - MG  
2011**

**EUDES DE CASTRO LIMA**


**ANÁLISE DE TÉCNICAS E FERRAMENTAS DE DETECÇÃO DE  
PLÁGIO, E DESENVOLVIMENTO DE UM PROTÓTIPO DE NOVA  
FERRAMENTA.**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel em Sistemas de Informação.

APROVADA em 29 de Novembro de 2011.

  
Prof. Dr. Heitor Augustus Xavier Costa (DCC/UFLA)

  
Prof.<sup>a</sup>. Dr.<sup>a</sup>. Ana Paula Piovesan Melchiori (DCC/UFLA)

  
Prof. Dr. Antônio Maria Pereira de Resende (DCC/UFLA)  
(Orientador)

**LAVRAS - MG  
2011**

*Dedico este trabalho a minha mãe Creusa, meu pai Érico (In memorian),  
meus irmãos Elvis e Érico Júnior e a minha namorada Jaqueline pelo  
apoio que serviram de base para erguer esta conquista.*

## **AGRADECIMENTOS**

*Ao meu pai Érico (In memorian) e à minha mãe Creusa.*

*A meus irmãos Érico Júnior e Elvis Márcio.*

*A minha namorada Jaqueline.*

*A Universidade Federal de Lavras.*

*Ao professor e orientador Antônio Maria Pereira de Resende.*

*Aos professores do Departamento de Ciência da Computação.*

*Aos amigos e familiares.*

*Ao Google.*

*E a todos que acreditam e acreditaram em mim.*

*"Há homens que perdem a saúde para juntar dinheiro e depois perdem o dinheiro para recuperar a saúde. Por pensarem ansiosamente no futuro, esquecem o presente, de tal forma que acabam por nem viver no presente nem no futuro. Vivem como se nunca fossem morrer e morrem como se nunca tivessem vivido..."*

Kung-Fu-Tze  
(Confúcio)

## RESUMO

O plágio é um problema crescente nas academias pois desrespeita a propriedade intelectual, reduz o aprendizado do aluno dentre outros. Como forma de inibir tal prática, surgiram diversas ferramentas de detecção de plágio, algumas focadas na detecção de plágio em código fonte e outras em documento de texto. Este trabalho de conclusão de curso descreve análises qualitativa e quantitativa das principais ferramentas de detecção de plágio em documentos de texto. Na análise qualitativa, descreve-se e compara-se os principais recursos oferecidos pelas ferramentas. Na análise quantitativa aplicam-se testes de eficácia, sensibilidade e desempenho, sendo apresentado um quadro comparativo. Após análises, realiza-se a modelagem e implementação de um protótipo de detecção de plágio. O protótipo é capaz de identificar similaridades entre documentos suspeitos com documentos presentes na WEB por meio do motor de busca Microsoft BING.

**Palavras-chave:** plágio, ferramentas de detecção de plágio, análise de ferramentas.

## ABSTRACT

Plagiarism is a growing problem in the academy because it breach intellectual property, reduces student learning, etc. As a way to inhibit this practice, several tools for detecting plagiarism, some focused on detecting plagiarism of source code and other of text documents, have emerged. This article describes qualitative and quantitative analysis of the main tools for plagiarism detection in text documents. Qualitative analysis describes and compares the main features offered by the tools, whereas quantitative analysis apply tests of efficiency, sensitivity and performance in order to establish a comparative table. After analysis, we carry out modelling and implementation of a prototype for plagiarism detection. The prototype is able to identify similarities between documents suspected to present documents on the web through the search engine Microsoft BING.

**Keywords:** plagiarism, plagiarism detection tools, analysis tools.



## LISTA DE ILUSTRAÇÕES

Figura 1: Espectro de plágio em código fonte .....	8
Figura 2: Representação de uma AST. ....	10
Figura 3: Exemplo ilustrativo de um PDG. ....	12
Figura 4: Passos do algoritmo <i>winnowing</i> . ....	18
Figura 5: Padrões de plágio e seus níveis de sofisticação.....	22
Figura 6: Tipo de pesquisa. ....	30
Figura 7. Passos para comparação das ferramentas .....	31
Figura 8. Tela principal do Ferret 4.0 .....	38
Figura 9. Sherlock.....	40
Figura 10. Tela inicial do CopyCatch Gold .....	42
Figura 11. Tela inicial da ferramenta WCopyFind .....	44
Figura 12. Tela para submissão de arquivos. ....	46
Figura 13. Diagrama de caso de uso .....	59
Figura 14. Diagrama de atividades .....	60
Figura 15. Diagrama de classes parte 1 .....	61
Figura 16: Diagrama de classes parte 2 .....	62
Figura 17. Tela de <i>login</i> do protótipo .....	63
Figura 18. Tela inicial do protótipo .....	64
Figura 19. Tela de submissão do documento suspeito.....	64
Figura 20. Tela de opções de análise do protótipo.....	65
Figura 21. Tipos de busca .....	66
Figura 22. Tela com os resultados da análise realizada, .....	67

## LISTA DE QUADROS E TABELAS

Tabela 1: Modelo de marca d'água. ....	13
Quadro 1. Resumo dos algoritmos listados em (CHARRAS e LECROQ, 1997). .....	16
Quadro 2. Ferramentas de detecção de plágio. ....	35
Quadro 3. Comparativo entre as ferramentas estudadas. ....	47
Quadro 4. Análise de eficácia das ferramentas de detecção de plágio selecionadas. ....	49
Quadro 5. Resultado da subtração entre similaridade obtida e similaridade esperada em módulo. ....	50
Quadro 6. Análise de Sensibilidade das ferramentas selecionadas. ....	52
Quadro 7. Resultado da subtração entre similaridade obtida e similaridade esperada. ....	52
Quadro 8. Análise de desempenho considerando os grupos A, B e C. ....	55
Quadro 9. Classificação geral das ferramentas. ....	57

## **LISTA DE SIGLAS**

**FORTRAN** - *FORmula TRANslation System*

**ITPAD** - *Instructional Tool for Program ADvising*

**GST** - *Greedy-String-Tilin*

**RKRGST** - *Running-Karp-Rabin Greedy-String-Tiling*

**AST** - *Abstract Syntax Trees*

**YAP** - *Yet Another Plague*

**MOSS** - *Measure of Software Similarity*

**COPS** - *COpy Protection System*

**PDG** - *Program Dependence Graph*

**FAQ** - *Frequently Asked Questions*

**HTML** - *HyperText Markup Language*

**URL** - *Uniform Resource Locator*

**JEE** - *Java Enterprise Edition*

**API** - *Application programming interface*

**UML** - *Unified Modeling Language*

## SUMÁRIO

<b>1. INTRODUÇÃO.....</b>	<b>1</b>
1.1. OBJETIVOS.....	1
1.2. DEFINIÇÃO DO PROBLEMA.....	2
1.3. SOLUÇÃO PROPOSTA.....	2
1.4. ESTRUTURA DO TRABALHO.....	2
<b>2. REFERENCIAL TEÓRICO.....</b>	<b>3</b>
2.1. O PLÁGIO.....	3
2.2. TIPOS DE PLÁGIO.....	4
2.3. DETECÇÃO DE PLÁGIO EM CÓDIGO FONTE.....	4
2.3.1. <i>Contagem de atributos e comparação de estruturas</i> .....	5
2.3.2. <i>História dos sistemas de detecção de plágio em código fonte</i> .....	6
2.3.3. <i>Dificuldades de identificar plágio em código fonte</i> .....	7
2.3.4. <i>Técnicas de detecção de plágio</i> .....	9
2.3.4.1. Técnicas baseadas em texto.....	9
2.3.4.2. Técnicas baseadas em <i>tokens</i> .....	9
2.3.4.3. Técnicas baseadas em Árvores.....	10
2.3.4.4. Técnicas baseadas em grafos.....	11
2.3.4.5. Técnica utilizando marca d'água.....	13
2.3.4.6. Técnica baseada em Código Intermediário.....	15
2.4. ALGORITMOS BÁSICOS DE BUSCA EM <i>STRING</i> .....	15
2.5. ALGORITMOS USADOS NA DETECÇÃO DE PLÁGIO.....	17
2.5.1. <i>Winnowing</i> .....	17
2.5.2. <i>Greedy String Tiling</i> .....	18
2.5.3. <i>Running Karp Rabin Greedy String Tiling</i> .....	20
2.6. DETECÇÃO DE PLÁGIO EM DOCUMENTOS DE TEXTO.....	21
2.6.1. <i>Dificuldades de detectar plágio em documentos de texto</i> .....	21
2.6.2. <i>Técnicas de detecção de plágio em documentos de texto</i> .....	22
2.6.2.1. Técnica Baseada em Fragmentos.....	23
2.6.2.2. Técnica baseada em modelos matemáticos.....	24
2.7. DETECTORES DE PLÁGIO.....	25
2.8. FERRAMENTAS DE DETECÇÃO DE PLÁGIO.....	26
2.8.1. <i>Análise em código fonte</i> .....	26
2.8.2. <i>Análise em documentos de texto</i> .....	27
<b>3. METODOLOGIA.....</b>	<b>29</b>
3.1. TIPOS DE PESQUISA.....	29
3.2. ATIVIDADES.....	31

<b>4.</b>	<b>ANÁLISE COMPARATIVA E DISCUSSÃO.....</b>	<b>34</b>
4.1.	LISTAR AS FERRAMENTAS .....	34
4.2.	CRITÉRIOS DE SELEÇÃO .....	34
4.3.	SELEÇÃO DAS FERRAMENTAS .....	35
4.4.	CRITÉRIOS DE COMPARAÇÃO .....	36
4.5.	DESCRIÇÃO DAS FERRAMENTAS.....	37
4.5.1.	<i>Ferret 4.0</i> .....	37
4.5.2.	<i>Sherlock</i> .....	39
4.5.3.	<i>CopyCatch Gold</i> .....	41
4.5.4.	<i>WCoppyFind</i> .....	43
4.5.5.	<i>DOC Cop</i> .....	45
4.6.	AVALIAÇÃO E DISCUSSÃO DOS DADOS .....	47
4.6.1.	<i>Avaliação qualitativa</i> .....	47
4.6.2.	<i>Avaliação quantitativa</i> .....	48
4.6.2.1.	Eficácia.....	48
4.6.2.2.	Sensibilidade.....	51
4.6.2.3.	Desempenho .....	53
4.6.3.	<i>Classificação geral</i> .....	56
<b>5.</b>	<b>ESTUDO DE CASO .....</b>	<b>58</b>
5.1.	IMPLEMENTAÇÃO.....	58
5.2.	MODELAGEM DO SISTEMA .....	59
5.2.1.	<i>Diagrama de caso de uso</i> .....	59
5.2.2.	<i>Diagrama de atividades</i> .....	60
5.2.3.	<i>Diagrama de classes</i> .....	61
5.3.	DETALHES DO PROTÓTIPO DESENVOLVIDO .....	63
<b>6.</b>	<b>CONCLUSÕES E TRABALHOS FUTUROS .....</b>	<b>68</b>
	<b>BIBLIOGRAFIA .....</b>	<b>71</b>

## 1. INTRODUÇÃO

Copiar fragmentos de textos ou textos completos sem citar os verdadeiros autores, caracteriza plágio. Apesar de ser uma prática crescente no meio acadêmico, ela é ilegal e não desejável por tratar-se de fraude.

Além de possuir diversos tipos, o plágio pode manifestar-se em diferentes áreas como educação, publicidade, música, fotografia, artes e literatura.

A falta de dedicação juntamente com a falta de conhecimento sobre o tema são as principais causas que levam ao plágio. Muitas vezes, copiar da Internet é a maneira mais simples de alcançar os objetivos. Alguns alunos plagam intencionalmente e outros o fazem inconscientemente. Segundo Barnbaum (2002), a falta de conhecimento do que constitui o plágio leva muitos alunos a cometê-lo inconscientemente. Se não sabe exatamente o que o plágio é, não pode evitar fazê-lo.

Identificar o plágio manualmente não é uma tarefa simples, demanda grande esforço e tempo dos professores. Diante do crescimento das ocorrências de plágio, surgem ferramentas automatizadas capazes de identificar similaridade entre documentos de texto.

### 1.1. Objetivos

O objetivo principal deste trabalho é realizar uma análise comparativa das principais ferramentas de detecção de plágio em documentos de texto, a fim de auxiliar professores e interessados na tomada de decisão de qual ferramenta utilizar. Como objetivos específicos, tem-se:

- Conhecer o estado da arte na área de detecção automática de plágio;
- Modelar e implementar um protótipo detector de plágio.

## **1.2. Definição do Problema**

Atualmente, com a expansão do plágio, existem várias ferramentas e técnicas disponíveis no mercado, mas pouco estudo comparativo entre elas.

## **1.3. Solução proposta**

Realizar análise qualitativa e quantitativa das principais ferramentas de detecção de plágio em documentos de texto. Na análise qualitativa, descrevem-se e comparam-se os principais recursos oferecidos pelas ferramentas e, na análise quantitativa, aplicam-se testes de eficácia, sensibilidade e desempenho, a fim de estabelecer um quadro comparativo.

## **1.4. Estrutura do trabalho**

O presente trabalho encontra-se estruturado em 6 capítulos, sendo o primeiro uma breve introdução contendo os objetivos, definição do problema em estudo e a solução proposta.

No capítulo 2, encontra-se o Referencial Teórico onde são fundamentados os principais conceitos deste trabalho.

No capítulo 3, estão definidas a metodologia utilizada, as atividades realizadas e os métodos de avaliação.

No capítulo 4, é apresentada a análise comparativa realizada nas principais ferramentas de detecção de plágio, com base nos métodos de avaliação apresentados no capítulo 3, e os resultados obtidos.

No capítulo 5, encontra-se um Estudo de Caso que mostra a modelagem e a implementação de um protótipo detector de plágio na WEB.

No capítulo 6, são apresentadas as principais conclusões obtidas neste trabalho, as contribuições e os trabalhos futuros.

## 2. REFERENCIAL TEÓRICO

Neste capítulo, são descritas as informações básicas necessárias ao entendimento da essência desse trabalho.

### 2.1. O Plágio

O Plágio pode ter várias definições. Segundo (PLAGIARISM.ORG, 2010), os pontos seguintes são considerados plágio:

- Transformar o trabalho de alguém em seu próprio;
- Copiar palavras ou ideias de alguém sem dar crédito;
- Não colocar a devida citação;
- Dar informações incorretas sobre a origem de uma citação;
- Mudança nas palavras, mas copiar a estrutura das frases de uma fonte sem dar crédito;
- Copiar palavras ou ideias de uma fonte que compõe a maioria de seu trabalho.

Conforme Smith e Wren (2010), plágio consiste na utilização de pensamento ou trabalho de outrem, sem aviso ou autorização. Para Hartmann (2006), plágio consiste na reprodução parcial ou integral de uma propriedade intelectual e ou artística.

Em outro caso específico, mas que não diminui a gravidade do plágio é o compartilhamento de atividades de programação entre alunos da área de Ciência da Computação. Segundo Cosma e Joy (2006), quando se trata de código fonte, os pontos seguintes são considerados plágio:

- Reutilização de código fonte sem fornecer referência adequada;
- Conversão integral ou parcial para outra linguagem de programação;
- Usar *software* para gerar o código fonte sem alertar o fato;
- Pagar alguém para fazer.



## 2.2. Tipos de Plágio

O plágio pode se manifestar de diversas maneiras e tipos. Segundo Barnbaum (2002), os tipos mais frequentes são:

- **Copiar e Colar:** Copiar e colar parte ou integralmente um trabalho, sem citar o verdadeiro autor;
- **Mudança na Frase/Paráfrase:** Reordenação das palavras mantendo o mesmo sentido;
- **Estilo:** Basear-se em trabalho de outros autores mantendo a mesma estrutura, mesmo que o conteúdo seja diferente da fonte original ainda é considerado plágio. Neste caso, o estilo do raciocínio é copiado;
- **Metáforas:** Metáforas são usadas para fazer uma ideia mais clara e dar ao leitor uma analogia que toca os sentidos ou emoções melhor do que uma simples descrição do objeto ou processo. Metáforas é uma parte importante do estilo criativo de um autor;
- **Ideia:** Se o autor do artigo fonte expressa uma ideia ou solução criativa ou sugere uma solução para um problema, ela deve ser claramente atribuída ao autor.

Para Liu *et al.* (2007), o plágio pode ser classificado em dois tipos: o intra-corporal e o extra-corporal. O plágio intra-corporal ocorre com o compartilhamento ao realizar uma mesma atividade. O Plágio extra-corporal ocorre com a cópia de fontes externas como internet, livros, etc.

## 2.3. Detecção de plágio em código fonte

Esta seção descreve as principais metodologias empregadas na detecção de plágio, a história das ferramentas, técnicas e algoritmos utilizados na detecção de plágio.

### 2.3.1. Contagem de atributos e comparação de estruturas

Segundo Ji, Woo e Cho (2007), duas metodologias para detecção de plágio em código fonte são utilizadas: contagem de atributos e comparação de estruturas.

Contagem de atributos é uma metodologia que extrai e calcula informações como: a frequência de palavras e número de ocorrências de atributos de um documento. Para Kleiman (2007), essa metodologia oferece bons resultados para casos onde houve pouca tentativa de alteração.

Conforme Ji, Woo e Cho (2007), a contagem de atributo pode ser uma metodologia eficaz para a detecção de plágio em documento de texto. No entanto, uma vez que não podem refletir as características estruturais, como controle de fluxo, não é adequado para detectar plágio em código fonte. Além disso, conforme Kleiman (2007), programas diferentes, mas de tamanho semelhante, podem apresentar contagens muito parecidas o que eleva o número de falso-positivo.

Segundo Verco e Wise (2006), os primeiros sistemas automatizados para a detecção de plágio em código fonte utilizavam a técnica de contagem de atributo em suas comparações.

Segundo Ji, Woo e Cho (2007), a metodologia baseada em estrutura é usada para comparar a estrutura do programa. Basicamente, objetiva a busca de trechos de um código fonte que, após preparação, seja similar a outros códigos fonte. A estrutura lógica de um programa pode ser complexa; por isso, para usar essa metodologia é preciso construir outro objeto para representar um determinado programa. Existem diversos métodos para representar programas como: sequência de *tokens*, cadeia de caracteres, árvores e grafos.

A maioria dos sistemas de detecção de plágio em código fonte utiliza a comparação de estrutura, pois, com ela, é possível encontrar trechos de plágio nos casos onde ocorrem o plágio parcial.

### 2.3.2. História dos sistemas de detecção de plágio em código fonte

Segundo Cornic (2008), o primeiro sistema de detecção de plágio foi desenvolvido por (OTTENSTEIN., 1976) que utilizou a metodologia de contagem de atributos para identificar similaridades em código fonte escrito na linguagem FORTRAN.

Conforme Cornic (2008), o sistema desenvolvido por (OTTENSTEIN., 1976) usou métricas de Halstead que sugere:

- N1: número total de ocorrências de operadores;
- N2: número total de ocorrências de operandos;
- n1: número de operadores únicos;
- n2: número de operandos únicos

Era considerado plágio pela ferramenta os pares de programas com n1, n2, N1 e N2 idênticos. Em seguida, os códigos fonte eram submetidos à revisão manual.

Em 1980, os autores Robinson e Soffa (1980) desenvolveram um novo sistema de detecção de plágio chamado *Instructional Tool for Program ADvising* (ITPAD), assim como o sistema proposto em (OTTENSTEIN., 1976), o ITPAD analisava código fonte escrito na linguagem FORTRAN.

As análises realizadas pelo ITPAD passavam por três fases. Na primeira fase, faz-se uma análise léxica onde eram calculadas 14 características, incluindo as métricas de Halstead. Na segunda fase, analisa-se a estrutura do programa por meio de gráficos de fluxo e, na terceira fase, interpretavam-se os resultados obtidos pela segunda fase.

Segundo Cornic (2008), os pesquisadores (DONALDSON, LANCASTER e SPOSATO, 1981) começaram a identificar novas técnicas que estudantes utilizavam para plagiar, por exemplo, renomear variáveis, alterar o formato de declarações. Baseado nessas técnicas, em 1981, Donaldson, Lancaster e Sposato (1981) criaram o primeiro sistema de detecção de plágio baseado em métricas de estruturas. A detecção era feita

através de comparação de estrutura de dois programas. Basicamente, o programa analisava os trabalhos e armazenava as informações sobre os tipos de declarações. Posteriormente, as declarações eram codificadas e comparadas.

Em 1990, Whale (1990) desenvolveu um *software* chamado Plague. O sistema proposto gera perfis dos programas de entrada. Os perfis são sequências de *tokens* compostos de informações estruturais. Os perfis de estrutura semelhante são combinados e suas sequências de *tokens* comparadas para encontrar subsequências comuns.

Em 1993, Wise (1993) criou o algoritmo *Greedy-String-Tilin* (GST) usado para combinar sequências de *tokens*. Esse algoritmo foi usado na ferramenta YAP3 que também foi desenvolvida pelo autor. A ferramenta ainda encontra-se em uso.

Em 2000, Prechelt, Malpohl e Phlippsen (2000) apresentaram um sistema chamado JPlag. A ferramenta transforma os programas em sequências de *tokens* e compara as sequências usando o algoritmo *Running-Karp-Rabin Greedy-String-Tiling* (RKRGST).

### 2.3.3. Dificuldades de identificar plágio em código fonte.

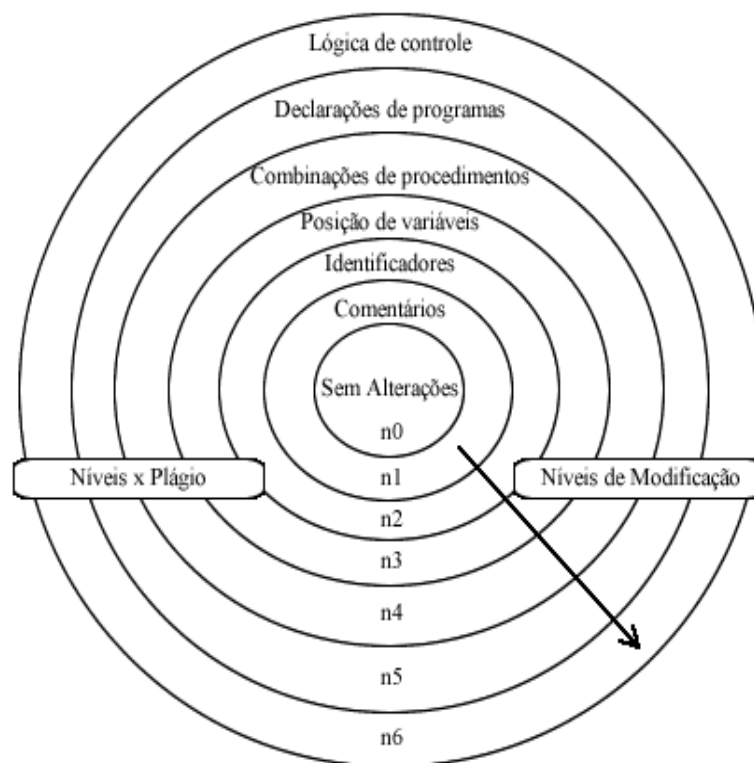
A grande dificuldade de identificar o plágio em código fonte é que os métodos utilizados para disfarçá-los podem variar dos mais simples aos mais complexos. Segundo Whale (1990) os seguintes métodos podem ser utilizados para disfarçar código fonte:

- Alterar comentários ou formatação;
- Alterar ordem dos operandos nas expressões;
- Alterar tipos de dados (por exemplo: substituir *real* para *integer*);
- Alterar identificadores;
- Alterar os laços de repetições (por exemplo: substituir *repeat* por *while*);

- Alterar as condições (por exemplo: substituir *if* por *case*);
- Adicionar variáveis.

Para (FAIDHI e ROBINSON., 1987) *apud* Parker e Hamblen (1989) os níveis de modificações em programas podem se dividir em seis (Figura 1).

Quando se trata de códigos fonte, onde não há alterações (cópia exata), é mais fácil de detectar o plágio do que quando há alterações na lógica de controle. Verifica-se no espectro apresentado na Figura 1 que quanto mais distante do centro, mais complexo se torna o plágio e sua detecção.



**Figura 1:** Espectro de plágio em código fonte

**Fonte:** (FAIDHI e ROBINSON., 1987) *apud* Parker e Hamblen (1989)

### **2.3.4. Técnicas de detecção de plágio**

Esta seção detalha as principais técnicas utilizadas pelas ferramentas de detecção de plágio em código fonte.

#### **2.3.4.1. Técnicas baseadas em texto**

As técnicas baseadas em texto levam este nome por serem puramente baseadas em texto, também são conhecidas como abordagens léxicas.

Nesta técnica, considera-se o código fonte como uma sequência de linhas, sendo cada linha uma sequência de caracteres. Os algoritmos de comparação buscam por sequências de caracteres iguais em dois fragmentos de código.

Faz-se pouca filtragem antes das comparações. De acordo com Roy e Cordy (2007) algumas alterações são comumente aplicadas como: remoção de qualquer tipo de comentário, remoção dos espaços em branco e normalizações básicas, como remoções de literais. Segundo Cornic (2008), o MOSS é um sistema de detecção de plágio *online* que utiliza comparação baseada em texto em suas análises.

#### **2.3.4.2. Técnicas baseadas em *tokens***

Na abordagem de detecção baseada em *tokens*, o código fonte analisado é transformado em uma sequência de *tokens*, posteriormente as comparações são feitas entre as sequências obtidas a fim de encontrar a maior subsequência comum. Os *tokens* representam os menores elementos a serem tratados pelos sistemas de detecção.

Segundo Roy e Cordy (2007), uma abordagem baseada em *tokens* normalmente é mais robusta em relação às alterações no código do que as abordagens baseadas em texto. Para Cornic (2008), o conjunto de *token* representa os elementos chave de um programa, comentários e espaços em

branco são ignorados, pois são os primeiros elementos alterados por um plagiador.

Ainda, segundo Cornic (2008), a principal vantagem desta técnica é a sua generalidade, pois com ela é possível descartar as informações desnecessárias como nomes de variáveis ou métodos. Portanto, a técnica baseada em *tokens* é insensível a alterações "procurar e substituir".

### 2.3.4.3. Técnicas baseadas em Árvores

*Abstract Syntax Trees* (AST) é uma representação hierárquica de um programa. Cada nó representa a construção de uma linguagem de programação e seus filhos são os parâmetros da linguagem programada. Segundo Cornic (2008), os nós de uma AST podem ser operadores matemáticos, chamadas de funções ou estruturas de programação. As folhas são variáveis ou constantes. Na Figura 2(a), tem-se um simples código fonte e, na Figura 2(b), é mostrada sua AST.

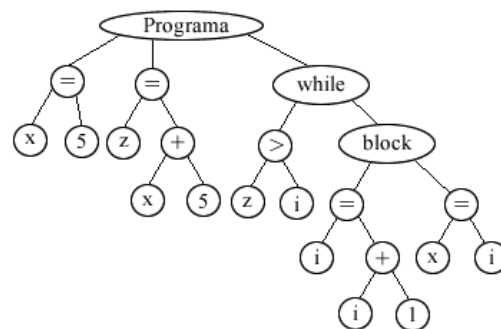
```
x = 5;
z = x + 5;

While ( z > i ){

    i = i + 1;
    x = i;

}
```

(a) Código fonte



(b) AST de (a).

**Figura 2:** Representação de uma AST.

Segundo Cornic (2008), ASTs são utilizadas por compiladores como representação interna de dados. Eles são uma representação intermediária entre o *parser tree* e a estrutura de dados. A diferença entre AST e um *parser tree* é um AST conter apenas informações que afetam a semântica do programa. As informações sintáticas desnecessárias são removidas.

A detecção de plágio utilizando AST consiste em encontrar subárvores comuns em ASTs de dois programas. A utilização de AST torna-se mais robusta contra renomeação quando são eliminados nomes de variáveis e constantes. Contudo, segundo Cornic (2008), as técnicas baseadas em AST são vulneráveis a inserção de código ou códigos reestruturados. Outro problema apresentado por Cornic (2008) é a complexidade, pois, para procurar uma subárvore comum entre duas árvores, exige-se comparação entre as subárvores o que torna-se impraticável para grandes conjuntos de dados.

Em Baxter *et al.* (1998), os autores apresentam um sistema de detecção de plágio usando AST para *software* de grande porte. Existem outros sistemas utilizando AST, porém possuem pouca documentação e com eficiência não comprovada.

#### **2.3.4.4. Técnicas baseadas em grafos**

O *Program Dependence Graph* (PDG) é uma representação em grafo de uma função ou procedimento do código fonte. Para Roy e Cordy (2007), PDGs podem mostrar a estrutura profunda dos programas, pois contém informações sobre o fluxo de controle e o fluxo de dados do *software* analisado. Segundo Cornic (2008), nesta abordagem as declarações são representadas por nós e as arestas incidentes a um nó representam os dados alterados ou usados nesta declaração e as condições de controle.

Diferente das outras representações utilizadas na detecção de plágio, a PDG não armazena informação sintática. Ela apresenta a relação entre as variáveis e as operações como mostra a Figura 3. Assim, as alterações do

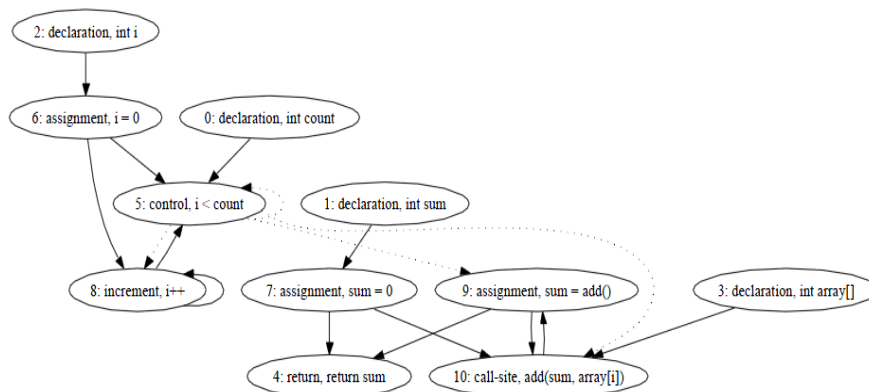


código por um plagiador não irão causar alterações da PDG. Normalmente, os plagiadores realizam modificações, sem qualquer efeito sobre a execução do programa tais como renomeação de variáveis e reordenação declarações. Essas alterações não alteram ou modificam um PDG.

```
int sum(int array[], int count)
{
    int i, sum;
    sum = 0;
    for(i = 0; i < count; i++){
        sum = add(sum, array[i]);
    }
    return sum;
}

int add(int a, int b)
{
    return a + b;
}
```

(a) código fonte



(b) PDG de (a).

**Figura 3:** Exemplo ilustrativo de um PDG.

**Fonte:** (LIU *et al* 2006).

Para modificar um PDG, necessitaria compreender e modificar o funcionamento do programa, o que exigirá uma boa compreensão do mesmo. Dessa forma, a alteração de um PDG é muito provável que exija mais trabalho do que a reformulação do programa, o que entra em contradição

com a principal razão de plagiar: gastar menos tempo do que o necessário para executar uma tarefa ou alcançar resultados que não é capaz de fazer por si mesmo.

GPlag é um sistema de detecção de plágio que baseia-se em grafos. Os programas suspeitos são transformados em um conjunto de PDGs e, em seguida, tenta-se encontrar subgrafos isomorfismo comum. Segundo Cornic (2008), os desenvolvedores do GPlag afirmam que o sistema é capaz de detectar até mesmo o tipo de plágio mais complexo, entretanto há pouca documentação e testes estatísticos que comprove a eficácia do GPlag.

#### 2.3.4.5. Técnica utilizando marca d'água.

Os autores Daly e Horgan (2005) apresentam uma técnica de detecção de plágio capaz de identificar o autor original do documento. A ideia básica da técnica é inserir uma marca d'água invisível ao autor e copiadador, mas detectável pelo sistema. Quando um código fonte é apresentado, é adicionada uma marca d'água invisível no fim do método principal, esta marca é um código binário que permite a identificação do verdadeiro autor.

Segundo Daly e Horgan (2005), o código binário requer trinta e quatro *bits* no qual: dez são para ID do estudante, quatro para o ano de inscrição do estudante, dez para o ID da tarefa, seis para o número de tentativas e quatro para a soma de verificação (Tabela 1).

**Tabela 1:** Modelo de marca d'água.

ID do estudante	Ano	ID da tarefa	Tentativas	Soma de verificação
0000101010	0011	0000101110	000111	0000

**Fonte:** Daly e Horgan (2005).

Alguns editores de texto não mostram o espaço adicional no final da linha, logo é pouco provável uma alteração, mesmo se o programa for plagiado. Posteriormente, a marca d'água é utilizada para identificar quem originalmente apresentou o programa.

Para Daly e Horgan (2005), essa técnica possui algumas vantagens sobre o detector de plágio que utilizam as técnicas convencionais de comparação, como:

- Distingue o verdadeiro autor do código fonte, independente da alteração no código;
- Pode identificar cópias em código fonte muito curto, típicos de disciplinas de iniciação a programação;
- Não requer análise manual para realmente afirmar que é plágio;
- Identifica o plágio logo que o programa é apresentado;
- Independe da linguagem de programação utilizada;
- Pode identificar plágio de trabalhos de anos anteriores.

Para Daly e Horgan (2005), em alguns casos, o plágio pode passar despercebido, pois essa técnica também possui algumas limitações, como:

- Código fonte reescrito não contém a marca d'água, logo a detecção só funciona se o aluno apresentar uma cópia eletrônica do outro trabalho;
- Se o aluno descobrir como funciona a técnica é fácil burlar, basta apagar a marca d'água;
- Em caso de exclusão da marca d'água, o plágio não será identificado.

Esta técnica difere da maioria de técnicas de detecção de plágio, porém, por ser fácil de burlar, não é eficaz.

#### **2.3.4.6. Técnica baseada em Código Intermediário**

A maioria dos sistemas de detecção de plágio baseia-se em comparações de recursos ou estruturas e é desenvolvido apenas para uma determinada linguagem de programação como C, Java e Pascal. Porém, uma prática comum entre os plagiadores é traduzir código fonte de uma linguagem para outra. Por Exemplo: passar um código fonte da linguagem C para Java.

Os autores Arwin e Tahaghoghi (2006) desenvolveram um sistema de detecção de plágio chamado XPlag. Tal sistema possui uma abordagem diferente, pois baseia-se em código intermediário. O XPlag usa um conjunto de compiladores para suportar várias linguagens de programação. Basicamente, a técnica consiste em transformar o código fonte original em código intermediário e em seguida, realizar várias otimizações. O algoritmo do XPlag utiliza o código intermediário como entrada logo após ele sofrer o primeiro nível de otimização.

Segundo Arwin e Tahaghoghi (2006), a principal utilidade do primeiro nível de otimização é incluir as funções com menos de 600 linhas no código. Essa otimização é interessante, pois inibe um dos métodos amplamente utilizados para esconder o plágio, a inserção de funções de códigos externos.

A proposta do sistema é utilizar um conjunto de compiladores para traduzir o código de linguagens diferentes em uma linguagem de baixo nível comum.

#### **2.4. Algoritmos básicos de busca em *String***

Quando se trata de ferramentas de detecção de plágio, é preciso mencionar os algoritmos de busca em *string*, componentes básicos em suas implementações. Dessa forma, para fornecer uma visão geral dos algoritmos e suas complexidades, apresenta-se no Quadro 1 um resumo do estudo realizado por Charras e Lecroq (1997) onde listam e descrevem alguns

algoritmos de busca em *string*. Considere "m" o tamanho da *string* padrão, "n" o tamanho do texto de entrada e  $\sigma$  o tamanho do alfabeto.

Lista-se no Quadro 1, alguns algoritmos de busca em *string*, seu tempo de preprocessamento e de comparação.

**Quadro 1.** Resumo dos algoritmos listados em (CHARRAS e LECROQ, 1997).

Algoritmos	Tempo de Preprocessamento	Tempo de Comparação
Brute Force	-	$O(mn)$
Deterministic Finite	$O(m\sigma)$	$O(n)$
Karp-Rabin	$O(m)$	$O(mn)$
Shift Or	$O(m\sigma)$	$O(n)$
Morris Pratt	$O(m)$	$O(m+n)$
Knuth-Morris-Pratt	$O(m)$	$O(m+n)$
Simon	$O(m)$	$O(m+n)$
Colussi	$O(m)$	$O(n)$
Galil-Giancarlo	$O(m)$	$O(n)$
Apostolico	$O(m)$	$O(n)$
Not So Naive	-	$O(mn)$
Forward Dawg Matching	-	$O(n)$
Boyer-Moore	$O(m+\sigma)$	$O(mn)$
Turbo-BM	$O(m+\sigma)$	$O(n)$
Apostolico-Giancarlo	$O(m+\sigma)$	$O(n)$
Reverse Colussi	$O(m^2)$	$O(n)$
Horspool	$O(m+\sigma)$	$O(mn)$
Quick Search	$O(m+\sigma)$	$O(mn)$
Zhu-Takaoka	$O(m+\sigma^2)$	$O(mn)$
Berry-Ravindran	$O(m+\sigma^2)$	$O(mn)$
Smith	$O(m+\sigma)$	$O(mn)$
Raita	$O(m+\sigma)$	$O(mn)$
Reverse Factor	$O(m)$	$O(mn)$
Turbo Reverse Factor	$O(m)$	$O(n)$
Backward Oracle Matching	$O(m)$	$O(mn)$
Galil-Seiferas	$O(m)$	$O(n)$
Two Way	$O(m)$	$O(n)$
String Matching on Ordered Alphabets	-	$O(n)$
Optimal Mismatch	$O(m^2+\sigma)$	$O(mn)$
Maximal Shift	$O(m^2+\sigma)$	$O(mn)$
Skip Search	$O(m+\sigma)$	$O(mn)$
KMP Skip Search	$O(m+\sigma)$	$O(n)$
Alpha Skip Search	$O(m)$	$O(mn)$

## 2.5. Algoritmos usados na detecção de plágio

Nesta seção, são apresentados os principais algoritmos utilizados no processo de comparação na detecção de plágio.

### 2.5.1. *Winnowing*

Segundo Schleimer, Wilkerson e Aiken (2003), *Winnowing* é um algoritmo que visa melhorar a eficiência do processo de comparação de documentos com base em assinatura. O algoritmo consiste em obter uma assinatura para um documento de forma que essa assinatura possa ser usada para identificá-lo e detectar similaridade.

O algoritmo *Winnowing* utiliza o conceito de k-gramas. Segundo Kleiman (2007), os k-gramas de uma cadeia S são as sub cadeias de comprimento k contíguas e sobrepostas da cadeia S. Exemplo é mostrado na Figura 4 (a)(b)(c).

Para obter a assinatura de um documento, o texto é dividido em k-gramas, o valor de *hash* de cada k-grama é calculado e um subconjunto desses valores é selecionado para ser a assinatura do documento (Figura 4 (d)).

Segundo Kleiman (2007), um dos requisitos para o algoritmo *Winnowing*, é que qualquer emparelhamento de comprimento igual ou maior ao limite de garantia t será detectado. Analogamente, qualquer casamento de comprimento menor que o limite de ruído k deve ser ignorado. Esses valores, t e k, são escolhidos pelo usuário para determinar o tamanho das janelas. Uma janela é um agrupamento contíguo de assinaturas calculadas como pode ser observado na Figura 4 (e).

A do run run run, a do run run  
 (a) Texto simples

adorunrunrunadorunrun  
 (b) Remoção dos caracteres irrelevantes

adoru dorun orunr runru unrun nrunr runru  
 unrun nruna runad unado nador adoru dorun  
 orunr runru unrun  
 (c) Sequências de 5-gramas derivados do texto

77 74 42 17 98 50 17 98 8 88 67 39 77 74 42  
 17 98  
 (d) Sequência hipotética de hashes dos 5-gramas

(77, 74, 42, 17)	(74, 42, 17, 98)
(42, 17, 98, 50)	(17, 98, 50, 17)
(98, 50, 17, 98)	(50, 17, 98, 8)
(17, 98, 8, 88)	(98, 8, 88, 67)
(8, 88, 67, 39)	(88, 67, 39, 77)
(67, 39, 77, 74)	(39, 77, 74, 42)
(77, 74, 42, 17)	(74, 42, 17, 98)

(e) Janelas de hashes de tamanho 4

17 17 8 39 17  
 (f) Assinaturas selecionadas pelo *Winnowing*

**Figura 4:** Passos do algoritmo *winnowing*.

**Fonte:** Schleimer, Wilkerson e Aiken (2003).

Segundo Schleimer, Wilkerson e Aiken (2003), o tamanho da janela é dado como  $w = t - k + 1$ , sendo o valor de  $k$  menor que ou igual ao valor de  $t$ . O valor de cada janela é escolhido da seguinte forma: um valor mínimo é selecionado, caso possua mais de um valor mínimo, o valor mais a direita é selecionado. Os valores escolhidos formarão a assinatura do documento como mostra a Figura 4 (f).

### 2.5.2. Greedy String Tiling

Introduzido por Wise (1993), o algoritmo *Greedy String Tiling* (GST) compara duas *strings* e determina o seu grau de semelhança. Para compreensão, é importante entender algumas definições. Sendo  $P$  uma *string* padrão e  $T$  uma *string* texto, define-se:

Um casamento-máximo ocorre quando uma *substring*  $P_p$  de uma *string* padrão iniciada em  $p$ , casa perfeitamente, elemento por elemento, com uma *substring*  $T_t$  da *string* texto começada em  $t$ . Considera-se o casamento

mais longo possível, isto é, até que um elemento que não casa ou um *end-of-string* é encontrado, ou até que um dos elementos encontrados está marcado. Casamentos-máximos são temporários e, possivelmente não são associações únicas, isto é, uma *substring* envolvida em um casamento-máximo pode ser parte de muitos outros casamentos-máximos.

Um *tile* é uma associação (um para um) permanente e única de uma *substring* de P com uma *substring* que casa em T. Para se formar um *tile* a partir de um casamento-máximo, os *tokens* das duas *strings* são marcados e, posteriormente, tornam-se indisponíveis para casamentos futuros.

Wise (1993) introduziu o casamento de tamanho mínimo como sendo um parâmetro que representa a quantidade de casamento-máximo ignorado. Esse valor é destinado a melhorar a eficiência do algoritmo, eliminando casamentos insignificantes.

O objetivo do algoritmo é encontrar um conjunto de *tiles* que maximize a cobertura sobre T através de P. O algoritmo GST é baseado na ideia de que casamentos longos são mais interessantes do que os curtos, porque são mais prováveis de representar semelhanças entre as *strings* em vez de coincidências.

O algoritmo executa múltiplas passagens nos dados, cada um deles é composto de duas fases. Na primeira fase, os casamentos-máximos acima de um certo comprimento são coletadas e armazenadas em listas, de acordo com seus comprimentos.

A segunda fase constrói *tiles* com casamento-máximo da primeira fase, começando com a mais longa. Para cada casamento, o algoritmo testa se ele está marcado. Se não, um *tile* é criado com este casamento e os textos correspondentes em P e T são marcados. Quando os casamentos de comprimento considerado forem tratados, um comprimento menor é escolhido e começa novamente a busca da primeira fase.

O algoritmo pára quando o texto completo estiver marcado e, em seu pior caso, possui a complexidade  $O(n^3)$ .



### 2.5.3. *Running Karp Rabin Greedy String Tiling*

O algoritmo *Karp-Rabin* foi criado por Richard M. Karp e Michael O. Rabin em 1987. Ele utiliza assinaturas para encontrar ocorrências de uma *string* em outras. Segundo Cornic (2008), a ideia principal deste algoritmo gira em torno de uma função *hash* que gera uma assinatura para as subcadeias de P e T. De acordo com Cornic (2008), essa assinatura pode ser obtida da seguinte forma:

Dado um k-grama (uma subsequência de tamanho k)  $c_1 \dots c_k$ , considerando esse k-grama como um número de k-dígitos em uma base b qualquer, a assinatura  $H(c_1 \dots c_k)$  desse k-grama é:

$$c_1 * b^{k-1} + c_2 * b^{k-2} + \dots + c_{k-1} * b + c_k$$

Para calcular a assinatura do k-grama  $c_2 \dots c_{k+1}$ , basta subtrair o elemento mais significativo, multiplicar o número por b e somar  $c_{k+1}$ . Dessa forma, tem-se a seguinte identidade:

$$H(c_2 \dots c_{k+1}) = (H(c_1 \dots c_k) - c_1 * b^{k-1}) * b + c_{k+1}$$

Este método permite calcular os valores de *hash* para os k-gramas de um documento em tempo linear. O algoritmo calcula o valor de *hash* da *string* padrão e compara com os valores de *hash* dos k-gramas do documento.

Wise (1993) aplicou a ideia do algoritmo *Karp-Rabin* ao algoritmo GST e criou o algoritmo *Running Karp-Rabin Greedy String Tiling* (RKRGSST).

Segundo Cornic (2008), foram feitas as seguintes alterações: O valor de *hash* é calculado para os k-gramas desmarcados da *string* padrão, em vez de apenas um valor para o padrão. O mesmo é feito para os k-gramas desmarcado da sequência de texto. Os valores de *hash* de cada k-grama da *string* padrão P são comparados com os valores de *hash* da sequência de

texto T. Para reduzir a complexidade, é criada uma *hash-table* de valores *hash* de *Karp-Rabin*. A busca na tabela retorna as posições dos k-gramas com o mesmo valor *hash*. Depois que uma ocorrência for encontrada, o algoritmo tenta estender o casamento, símbolo por símbolo. Após cada iteração, o comprimento da *string* pesquisada (k) é reduzido até o casamento de tamanho mínimo.

Este algoritmo é um dos mais usados nos sistemas detectores de plágio, sua complexidade no pior caso, segundo Wise (1993), é  $O(n^3)$ .

## 2.6. Detecção de plágio em documentos de texto

Esta seção descreve as dificuldades de detectar plágio em documentos de texto e apresenta algumas técnicas utilizadas na detecção.

### 2.6.1. Dificuldades de detectar plágio em documentos de texto.

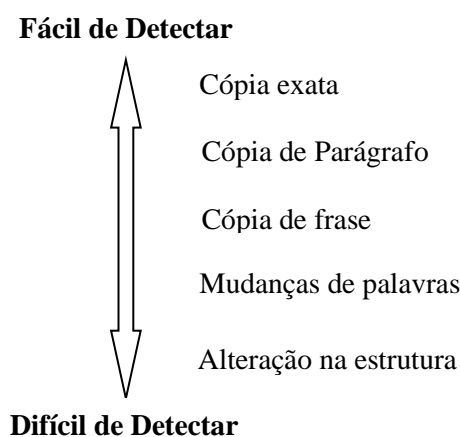
Identificar o plágio não é simples, exige tempo e esforço de quem o procura. Segundo Clough (2000), os fatores seguintes podem ser utilizados pelas ferramentas para detectar o plágio:

- **Uso de vocabulário:** Comparar o vocábulo do documento analisado com vocabulário de documentos conhecidos. Quanto maior for a diferença, menor a possibilidade de ser plagiado;
- **Mudança de vocabulário:** Mudanças de vocabulário no decorrer do texto podem ser indícios de plágio;
- **Pontuação:** Geralmente, diferentes textos possuem diferentes pontuações, caso sejam iguais podem indicar plágio;
- **Quantidade de similaridade entre textos:** Sempre existirá uma similaridade entre os textos escritos sobre o mesmo tema.

Entretanto, é improvável o compartilhamento de grande quantidade de texto;

- **Erros de gramática comum:** A ocorrência de erros gramaticais iguais em dois textos distintos pode indicar o plágio.

Em seu trabalho Kang, Gelbukh e Han (2006) classificam o grau de dificuldade de detecção de plágio em documentos de texto segundo a característica do tipo de plágio. Segundo os autores, é mais fácil identificar plágio em cópias exatas de documentos do que em documentos que tiveram alteração na sua estrutura (Figura 5).



**Figura 5:** Padrões de plágio e seus níveis de sofisticação  
**Fonte:** Kang, Gelbukh e Han (2006)

Para Mussini (2008), o plágio em documentos de texto é mais difícil de detectar do que em código fonte, pois a gramática completa da linguagem de programação é limitada e pode ser definida e especificada. Quando se trata de documento de texto, além de ser difícil de impor limites, pode haver ambiguidade.

### 2.6.2. Técnicas de detecção de plágio em documentos de texto

Esta seção descreve as principais técnicas utilizadas pelas ferramentas de detecção de plágio em documentos de texto.

### 2.6.2.1. Técnica Baseada em Fragmentos

Em seu trabalho Liu *et al.* (2007) propõe uma abordagem baseada em segmentos para detectar o plágio em documentos suspeitos. Tal abordagem possui três etapas principais: extração de segmentos, *ranking* do segmento e identificação da fonte de plágio.

Na etapa de extração de segmentos, os sinais de pontuação são tidos como separadores, porém existem algumas exceções, em alguns casos, o ponto não indica fim de período (por exemplo, Sr.; Sra.).

Na etapa de *ranking*, os segmentos são classificados. O processo de classificação baseia-se na quantidade de pronomes, artigos, preposição, chamados de palavras de parada. Quanto mais palavras de parada, menor a prioridade.

Na etapa de identificação da fonte, é onde ocorrem as buscas e tem-se o retorno das fontes suspeitas.

Em outras palavras, o sistema proposto incorpora a capacidade de pesquisa dos motores de busca. Inicialmente, apresenta-se ao sistema um documento suspeito; em seguida, é executada a etapa de extração de segmentos seguida da etapa de classificação. O próximo passo é a identificação da fonte do plágio. Os segmentos com menos prioridade podem não entrar na análise.

Conforme Knight, Almeroth e Bimber (2004), a busca exaustiva é provavelmente a melhor maneira de detectar documentos plagiados. A busca exaustiva ocorre quando o texto do documento é analisado. No entanto, analisar todas as frases de um documento com várias páginas resultaria em muitas consultas ao buscador, o que elevaria o tempo das análises.

Visando melhorias no tempo de análise, Knight, Almeroth e Bimber (2004) propõe uma busca exaustiva simplificada. Essa técnica fragmenta o texto de uma página em frases e envia as oito primeiras palavras de cada frase para consulta. Isso limita o número de consultas a cerca de 100 por página. O número oito foi escolhido porque ele retorna menor número de

falsos positivos e evita o problema de muitos documentos que contêm frases comuns em o texto.

A ideia apresentada por Knight, Almeroth e Bimber (2004) se aproxima da ideia de Liu *et al.* (2007), ambos fragmentam o texto sob análise, selecionam trechos seguindo suas técnicas e fazem uma análise dos *links* retornados dos buscadores.

Um ponto importante destacado por Knight, Almeroth e Bimber (2004) é os fragmentos, quando enviados ao buscador, são colocados entre aspas para indicar que esta buscando por exatamente aquele fragmento.

### **2.6.2.2. Técnica baseada em modelos matemáticos**

Apresentado por Oliveira e Oliveira (2008), esta técnica é capaz de identificar similaridades entre dois documentos de texto. A técnica baseia-se em modelos matemáticos e visa não somente reconhecer cópias idênticas de texto, mas reconhecer o relacionamento entre as palavras, ou seja, sua estrutura semântica.

A proposta de Oliveira e Oliveira (2008) utiliza o Modelo de Representação Vetorial para avaliar o plágio em documentos de uma coleção. Nesse modelo, segundo (BAEZA-YATES e RIBEIRO-NETO, 1998), *apud* Oliveira e Oliveira (2008), os documentos são representados por vetores no espaço  $\mathbb{R}^n$ .

A proposta de Oliveira e Oliveira (2008) consiste em atribuir peso aos termos de acordo com a frequência que aparecem no texto. Para atribuição de peso, podem ser usadas técnicas sofisticadas, que representem melhor um documento. Após atribuição dos pesos, a similaridade pode ser obtida através do produto vetorial dos pesos, por exemplo: dados dois documentos texto  $v_1$  e  $v_2$  de uma base de documentos  $V = v_1, v_2, \dots, v_n$  A similaridade pode ser obtida usando a seguinte formula:

$$\text{Similaridade}(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| |v_2|} = \cos(\theta)$$

O cosseno do ângulo  $\theta$  é o resultado do produto vetorial entre esses vetores. Quanto mais o  $\cos(\theta)$  se aproxima do 1, maior a similaridade entre os documentos. Quando o  $\cos(\theta) = 1$ , quer dizer que os documentos são exatamente iguais, ou seja, possuem similaridade total. Quando o  $\cos(\theta) = 0$ , indica que não há similaridade entre os documentos analisados.

Por fim, de acordo com Oliveira e Oliveira (2008), o Modelo de Representação Vetorial, por indicar a similaridade entre dois documentos, possui como aplicação o reconhecimento de plágios diretos e até de paráfrases.

## 2.7. Detectores de plágio

Detectores de plágio são ferramentas automatizadas que tem por objetivo identificar similaridade entre dois ou mais documentos, seja ele documento de texto ou código fonte. Segundo McKeever (2006), o processo de detectar plágio possui um grande custo computacional, sendo necessário tomar medidas para reduzir o domínio de comparação e reduzir o tempo das análises.

Existem no mercado diversas ferramentas automatizadas de detecção de plágio, algumas são gratuitas e outras pagas. Em sua essência, as ferramentas são similares, pois possuem o mesmo propósito, identificar o plágio, mas cada uma delas implementam técnicas e algoritmos diferentes.

Conforme Maurer, Kappe e Zaka (2006), existem três categorias de descoberta de plágio através de *software*:

- **Comparação entre documentos:** é a abordagem mais comum, é através da comparação do documento com um conjunto de documentos;

- **Busca por parágrafo suspeito na internet:** submete para análise nos motores de buscas frases ou expressões suspeitas selecionadas pelo professor;
- **Estilometria:** analisa o estilo da escrita do texto através de comparações com documentos previamente escritos pelo mesmo autor. Este método é o mais complicado, pois envolve técnicas sofisticadas de inteligência artificial para a confecção do *software*.

## 2.8. Ferramentas de detecção de plágio

Nesta seção, algumas ferramentas pesquisadas foram divididas em dois grupos. O primeiro são as ferramentas que fazem análise em código fonte e o segundo são as ferramentas que fazem análise em documentos de texto.

### 2.8.1. Análise em código fonte

Nesta seção foram descritas algumas ferramentas que detectam o plágio em código fonte.

- **YAP3:** Foi desenvolvida por Wise (1996) para combater o plágio em código fonte, é a terceira versão de uma série chamada YAP (*Yet Another Plague*). Essa ferramenta utiliza o algoritmo *Running Karp-Rabin Greedy String Tiling*. (RKRGST) proposto pelo próprio autor em 1993. A vantagem dessa ferramenta em relação às outras é que o algoritmo utilizado é robusto em situações em que apenas trechos de código fonte são copiados;
- **JPlag:** Foi desenvolvida utilizando a linguagem de programação Java por Prechelt, Malpohl e Philippsen (2000). Essa ferramenta é capaz de encontrar semelhanças entre vários conjuntos de arquivos de código fonte. Atualmente suporta os formatos Java, C #, C, C + + e Scheme. A análise efetuada pelo JPlag é baseada na estrutura do código e utiliza

uma versão modificada do algoritmo RKRGSST. É uma ferramenta robusta contra vários tipos de plágio;

- **MOSS:** *Measure of Software Similarity (MOSS)* é um sistema automático de detecção de similaridade entre código fonte desenvolvido em 1994 na Universidade de Stanford. É uma ferramenta baseada na WEB, possui seu código fonte fechado e permite análise em mais de 24 linguagens de programação;
- **Marble:** Segundo Hage, Rademaker e Vugt (2010), Marble é uma ferramenta de detecção de plágio em código fonte desenvolvida em 2002 na Universidade de Utrecht. Além disso, é uma ferramenta de código fechado que roda localmente. Sua utilização é por linha de comando e efetua análise em códigos em JAVA;
- **SIM:** Segundo Hage, Rademaker e Vugt (2010), o SIM é um sistema capaz de identificar similaridades em códigos fonte escritas nas linguagens C, Java, Pascal, Lisp dentre outras. Foi desenvolvida em 1989 por Dick Grune da Universidade Amsterdã. Essa ferramenta possui o código aberto e roda localmente. Assim como a ferramenta Marble, sua utilização é por linha de comando.

### 2.8.2. Análise em documentos de texto.

Nesta seção foram descritas algumas ferramentas que detectam o plágio em documentos de texto.

- **EVE2:** Conforme (EVE2, 2011), EVE2 é uma ferramenta de detecção de plágio proprietária voltada para instituições de ensino. Essa ferramenta permite análises em documentos de texto puro e Microsoft Word. Além disso, roda localmente e estende sua capacidade de busca a documentos presente na WEB. O sistema ao término das análises retorna uma lista de endereços junto com o grau de similaridade, a partir daí é



preciso uma revisão manual. Essa ferramenta implementa várias técnicas para diminuir os falsos-positivos dos relatórios;

- **DOC Cop:** Foi desenvolvida por Marcos McCrohon. A princípio o objetivo da ferramenta era auxiliar professores de uma Universidade australiana. Atualmente, a ferramenta é usada por pesquisadores, jornalistas, editores dentre outros. DOC Cop é uma ferramenta WEB e gratuita que necessita apenas de um cadastro. A ferramenta possui uma interface gráfica limpa. Além disso, é possível fazer análises entre documentos no formato DOC, DOCX e PDF e trechos de texto na WEB (DOC Cop, 2011);
- **COPS:** Segundo Mussini (2008), o *COPY Protection System* (COPS) é um protótipo funcional de um sistema de detecção de cópias destinado a uma biblioteca digital da Universidade de Stanford. A função do COPS no projeto é detectar cópias exatas ou parciais na biblioteca. São permitidos documentos no formato troff, Tex e DVI;
- **VIPER:** desenvolvido pela Scanmyessay, o Viper é uma ferramenta de detecção de plágio gratuita que roda localmente. Com ela é possível verificar se o documento suspeito está na WEB. Das ferramentas citadas, é a única que retém uma cópia do arquivo analisado. Ele suporta documentos no formato TXT, DOC, RTF e HTML (SCAN MY ESSAY, 2011);
- **PlagiarismDetect.com:** essa ferramenta oferece serviços gratuitos e pagos. No serviço gratuito, as análises são lentas e permite apenas documentos no formato de texto puro. O serviço pago possui diversas vantagens sobre o gratuito, como: permite análise em 8 formatos de arquivos, permite fazer *upload* de 20 arquivos e oferece um *plug-in* para o MS Word 2007 que verifica o plágio no momento da digitação (PLAGIARISM DETECT, 2011).

### **3. METODOLOGIA**

Neste capítulo, é apresentado o tipo de pesquisa, as atividades realizadas e os métodos de avaliação utilizados na comparação.

#### **3.1. Tipos de pesquisa**

De acordo com Jung (2004), um trabalho desenvolvido pode ser classificado quanto a sua natureza, objetivos e procedimentos.

Quanto à natureza, uma pesquisa pode ser classificada como básica/fundamental ou aplicada/tecnológica. Segundo Jung (2004), uma pesquisa básica/fundamental objetiva entender, descrever ou explicar os fenômenos naturais, com foco em conhecimentos básicos e fundamentais. Enquanto a pesquisa aplicada/tecnológica objetiva a utilização dos conhecimentos básicos e tecnologias existentes na criação de novos produtos ou processos.

Este trabalho de conclusão de curso teve sua natureza classificada como tecnológica/aplicada, pois utiliza conhecimentos básicos no estudo das principais técnicas e ferramentas de detecção de plágio e objetiva a construção de um protótipo detector de plágio.

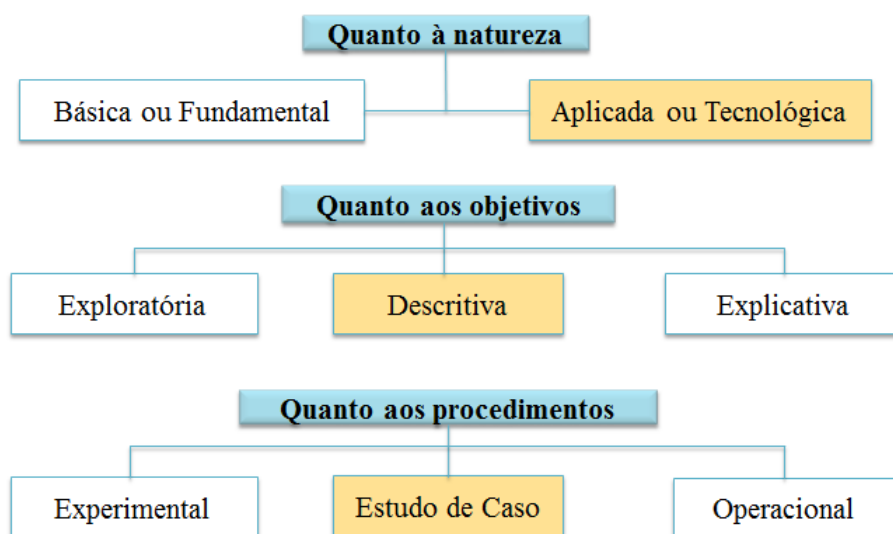
Quanto aos objetivos, uma pesquisa pode ser classificada como exploratória, descritiva ou explicativa. De acordo com Jung (2004), a pesquisa exploratória visa à descoberta, ao achado, à elucidação de fenômenos ou à explicação daqueles não aceitos apesar de evidentes. A pesquisa descritiva visa à identificação, registro e análise das características, fatores ou variáveis que se relacionam com o fenômeno ou processo. A pesquisa explicativa visa a identificar os fatores que contribuem para a ocorrência dos fenômenos ou variáveis que afetam o processo. Explica o “porquê das coisas”.

Este trabalho de conclusão de curso teve seus objetivos classificados como Descritivo, pois visa a identificar e a analisar as principais

características dos sistemas detectores de plágio, a fim de gerar um quadro comparativo.

Quanto aos procedimentos, uma pesquisa pode ser classificada como experimental, operacional ou estudo de caso. Conforme Jung (2004), a pesquisa experimental viabiliza a descoberta de novos materiais, componentes, métodos, técnicas, dentre outros. A pesquisa operacional trata através do uso de ferramentas estatísticas e métodos matemáticos da otimização para a seleção do meio mais adequado para se obter o melhor resultado. O estudo de caso permite investigar um fenômeno dentro do contexto local, real e especialmente quando os limites entre fenômeno e o contexto não estão claramente definidos.

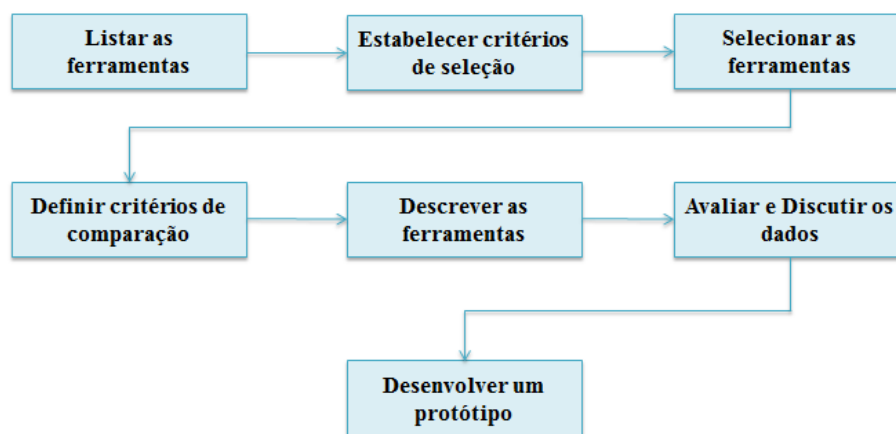
Este trabalho de conclusão de curso teve seu procedimento classificado como estudo de caso. A Figura 6 resume o tipo de pesquisa adotado neste trabalho.



**Figura 6:** Tipo de pesquisa.

### 3.2. Atividades

Os passos realizados neste trabalho estão representados pela Figura 7 e, posteriormente, são detalhados.



**Figura 7.** Passos para comparação das ferramentas

#### a) Listar as ferramentas

Neste passo, foram realizadas várias pesquisas na WEB com o intuito de listar as principais ferramentas de detecção de plágio existentes no mercado para serem descritas e comparadas ao longo do trabalho.

#### b) Estabelecer critérios de seleção

Neste passo, foram definidos os critérios de seleção que servem para efetuar um corte inicial das ferramentas candidatas e não candidatas usadas nas comparações.

#### c) Selecionar as ferramentas

Neste passo, os critérios de seleção estabelecidos são aplicados as ferramentas listadas. Com isso, obtêm-se como resultados as ferramentas de detecção de plágio aptas a comparação.

**d) Definir os critérios de comparação**

Neste passo, foram definidos alguns critérios de comparação considerados importantes. Os critérios foram elaborados a partir do levantamento de características presentes nas ferramentas existentes e das necessidades que devem estar presentes nas ferramentas.

**e) Descrever as ferramentas**

Neste passo, após definir os critérios de comparação, as ferramentas foram descritas com base em informações contidas na documentação, *website* e artigos referentes.

**f) Avaliar e discutir os dados**

Duas avaliações foram realizadas. A primeira qualitativa, baseado no trabalho de Hage, Rademaker e Vugt (2010) onde se aplicam os critérios de comparação nas ferramentas. A segunda, quantitativa que baseia-se nos resultados obtidos através da aplicação dos testes de eficácia, sensibilidade e desempenho.

**• Teste de eficácia**

O objetivo do teste de eficácia é verificar se as similaridades apresentadas pelas ferramentas em suas análises correspondem com as similaridades esperadas. Em seu trabalho Franco, Milanez e Santos (2008) realizam o teste de sensibilidade na ferramenta Sherlock com o mesmo objetivo apresentado neste trabalho.

**• Teste de sensibilidades**

O objetivo do teste de sensibilidade é verificar o quão as ferramentas de detecção de plágio são sensíveis a alterações nos documentos de entrada e

identificar em quais condições as ferramentas apresentam os melhores resultados.

- **Teste de desempenho**

O objetivo do teste de desempenho realizado neste trabalho é averiguar o tempo de análise das ferramentas em documentos de tamanhos variados. Esse teste é interessante, pois o desempenho é um fator importante na hora da escolha da ferramenta. O ideal para este teste seria realizar uma análise baseada nos algoritmos, porém, os fabricantes das ferramentas não disponibilizam informações a respeito dos algoritmos ou técnicas utilizadas. Também, poderia ser examinado baseando no tempo do sistema, mas algumas ferramentas não fornecem o tempo de análise de plágio. Sendo assim, os testes foram realizados manualmente com auxílio de um cronômetro. Logo, o teste realizado neste trabalho não é preciso, mas sugere um tempo de execução aproximado de cada ferramenta. Após testes e análises das ferramentas selecionadas, um quadro comparativo contendo um *ranking* das ferramentas analisadas foi criado. Desta forma, é possível identificar a melhor ferramenta.

- g) Desenvolver um protótipo**

Neste passo, foram feitas a modelagem e o desenvolvimento de um protótipo detector de plágio na WEB utilizando o motor de busca Microsoft BING.

## **4. ANÁLISE COMPARATIVA E DISCUSSÃO**

Apresentam-se neste capítulo, as comparações realizadas de acordo com os métodos de avaliação apresentado na seção 3.2, e também, apresenta os resultados obtidos.

### **4.1. Listar as ferramentas**

Para determinar o conjunto de ferramentas a serem analisadas neste trabalho de pesquisa, fez-se diversas pesquisas no Google (GOOGLE, 2011). Foram utilizadas palavras-chave como: “*Detector de plágios*”, “*Plagiarism detection*”, “*Plagiarism detection software*”, “*detect copying in text documents*” e “*automated document comparison*”. Para cada consulta foram verificadas as 10 primeiras páginas de resposta com 10 ocorrências cada. As ferramentas encontradas nessas buscas estão listadas na Quadro 2.

### **4.2. Critérios de seleção**

Para definir as ferramentas de detecção de plágio a serem comparadas foram adotados dois critérios de seleção:

- Ser gratuita;
- Permitir análises em documentos de texto;

**Quadro 2.** Ferramentas de detecção de plágio.

Ferramentas	Gratuita?	Permite análise em documentos de texto?
Plagius	Não	Sim
Farejador de Plágio	Não	Sim
Ferret 4.0	Sim	Sim
Yap3	Sim	Não
Sherlock	Sim	Sim
Moss	Sim	Não
EVE2	Não	Sim
CROT Antiplagiarism	Não	Sim
iPlagiarism Check	Não	Sim
Viper	Sim	Sim
Plagiarism Detector	Não	Sim
AntiPlagiarist	Não	Sim
CopyCatch Gold	Sim	Sim
Turnitin	Não	Sim
Plagiarized.org	Sim	Não
CrossRefme	Sim	Não
WCopFind 2.7	Sim	Sim
Grammarly	Não	Sim
Pl@giarism	Sim	Sim
Dupli Checker	Sim	Não
Check for plagiarism	Não	Sim
Academic Plagiarism	Não	Sim
Plagiarism Search	Não	Sim
Plagiarisma.net	Não	Sim
Plagiarism Checker	Sim	Não
The Plagiarism Checker	Não	Sim
CopyScape	Sim	Não
The Plagiarism	Não	Sim
DOC Cop	Sim	Sim
Plagiarism detection	Não	Sim
SID	Sim	Não

### 4.3. Seleção das ferramentas

Aplicando os critérios de seleção da sessão 4.2 às ferramentas listadas no Quadro 2, foram selecionadas: Ferret 4.0, Sherlock, Viper, CopyCath Gold, WCopFind 2.7, Pl@giarism. No entanto, duas ferramentas não foram consideradas: Pl@giarism e Viper. A primeira encontra-se em fase de testes e a segunda vem passando por problemas técnicos segundo informações obtidas em (FACEBOOK, 2011).



#### 4.4. Critérios de comparação

Os seguintes critérios de comparação foram adotados:

- **WEB:** A ferramenta é oferecida como serviço WEB?
- **Análise em código fonte:** As ferramentas oferecem análise em código fonte? Quais linguagens de programação são suportadas?
- **Apresentação dos resultados:** A quantidade de informações e sua organização foram consideradas relevantes para avaliação?
- **Extensões suportadas:** Quais extensões de documentos de texto são suportadas?
- **FAQ:** A ferramenta possui *Frequently Asked Questions (FAQ)*?
- **Múltiplaplataforma:** É uma ferramenta que pode ser usada em vários Sistemas Operacionais? Quais os sistemas operacionais são suportados?
- **Usabilidade:** A ferramenta possui interface gráfica? A disposição dos botões da funcionalidade proporcionam facilidades de uso?
- **Análise na WEB:** A ferramenta estende sua capacidade de análise aos motores de busca da WEB? É possível selecionar qual motor utilizar?
- **Cadastro:** Para adquirir a ferramenta, é necessário cadastrar?
- **Manual:** A ferramenta possui manual ou instruções de uso?

Para classificar a usabilidade, foram adotados os critérios:

- (\*) – Ruim: Ferramenta que não possui interface gráfica;
- (\*\*) – Razoável: Possui interface pouco atrativa que não proporciona facilidades de uso;
- (\*\*\*) – Boa: Possui uma interface simples e objetiva que proporciona facilidades de uso.

Para apresentação dos resultados, foram adotados:

- (\*) – Ruim: Possui relatório que omite informações relevantes, mal estruturado, redundante;

- (\*\*) – Razoável: Possui relatório de difícil interpretação, mal organizado, possui informações irrelevantes;
- (\*\*\*) – Boa: Relatório simples e organizado, com informações relevantes.

#### 4.5. Descrição das ferramentas

Após selecionar as ferramentas e definir os critérios de comparação, foram realizadas várias pesquisas sobre cada ferramenta para adquirir informações e responder aos critérios de comparação estabelecidos.

##### 4.5.1. Ferret 4.0

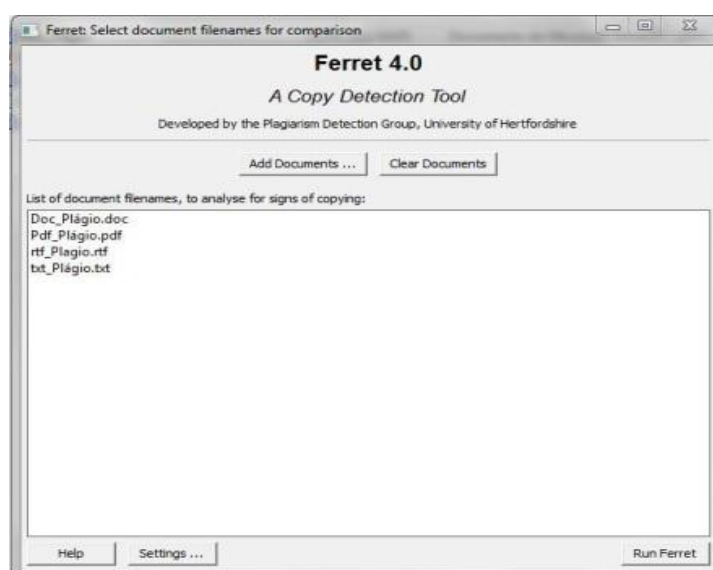
O Ferret foi desenvolvido no Departamento de Ciência da Computação da Universidade de Hertfordshire, UK. O algoritmo foi elaborado por Caroline Lyon e codificado por Bob Dickerson e Malcolm James, posteriormente outras versões foram escritas por vários membros do departamento de Ciência da Computação.

Segundo Lyon, Barrett e Malcolm (2003), no início, o Ferret foi considerado um sucesso técnico, mas, em diferentes estágios de sua evolução, teve partes implementadas em JAVA, C++, Visual Basic, Visual C++. Por conta disso, o código tornou-se insustentável e tornou-se um desastre da Engenharia de Software.

O Ferret 4.0 é uma ferramenta gratuita projetada para executar localmente, pode ser executado nas plataformas Apple Mac, Microsoft Windows, Linux e outras versões do Unix. Para obtê-la, é necessário fazer *download*. Para isso, não é preciso cadastrar.

A ferramenta possui uma interface gráfica simples e objetiva como mostra a Figura 8. A forma com que os botões estão organizados na tela principal proporciona facilidades de uso.

A análise constitui-se de seleção, comparação dos documentos e exibição dos resultados. Na seleção, ocorre a escolha dos documentos suspeitos. A ferramenta suporta 300 documentos de 10.000 palavras a cada análise e pode processar documentos no formato DOC, RTF, TXT e PDF. Ainda é possível configurar a ferramenta para fazer análise em código fonte. Na comparação, ocorre a procura pelo plágio. Os documentos são comparados aos pares em busca de similaridades. Ao término das comparações, os resultados são exibidos em forma de tabela. O Ferret 4.0 não possui versão WEB e não estende suas análises a documentos presentes na WEB.



**Figura 8.** Tela principal do Ferret 4.0

Após análise, os pares de documentos analisados e suas similaridades são exibidos. A similaridade varia de 0 (indica ausência de plágio) a 1 (indica 100% de plágio).

A ferramenta oferece algumas opções pós-análises, como: ordenação dos documentos de acordo com o nome ou a similaridade, exibição dos documentos lado a lado e a opção para salvar o relatório da análise em PDF.

O relatório gerado possui apenas o número de documentos selecionados, o número de pares de documentos comparados e uma tabela com os pares de documentos analisados e suas similaridades.

O manual é explicativo e contém imagens e exemplos de funcionamento da ferramenta. O *download* da ferramenta e do manual pode ser feito através do *site*: <http://homepages.feis.herts.ac.uk/~pdgroup/>. No *site* da ferramenta, não há FAQ ou suporte aos usuários.

#### 4.5.2. Sherlock

A primeira versão do Sherlock foi desenvolvida por Rob Pike. Inicialmente, era constituída de dois programas *sig* e *comp*, posteriormente, Loki o adaptou em apenas um programa, chamado Sherlock (SHERLOCK, 2011).

Sherlock é uma ferramenta *opensource*. Para executar, é necessário um compilador para linguagem C. Além do mais, é uma ferramenta múltiplaplataforma, pode ser compilada e executada localmente em diferentes sistemas operacionais.

A ferramenta não possui interface gráfica com o usuário, suas instruções são passadas por linhas de comando, como mostra a Figura 9. O Sherlock não possui versão WEB e não estende as suas análises aos documentos presentes na WEB.

A ferramenta detecta similaridades entre documentos no formato TXT. Além disso, faz análise em código fonte na linguagem JAVA. Para detectar similaridade em documentos, a ferramenta calcula uma assinatura digital para cada linha do texto. Em seguida, as assinaturas são comparadas com as assinaturas dos demais documentos. Completada a etapa de comparação, os resultados são exibidos. Mais detalhes do seu funcionamento podem ser encontrados em (FRANCO, MILANEZ e SANTOS, 2008).

Segundo, quatro parâmetros podem ser configurados no Sherlock:

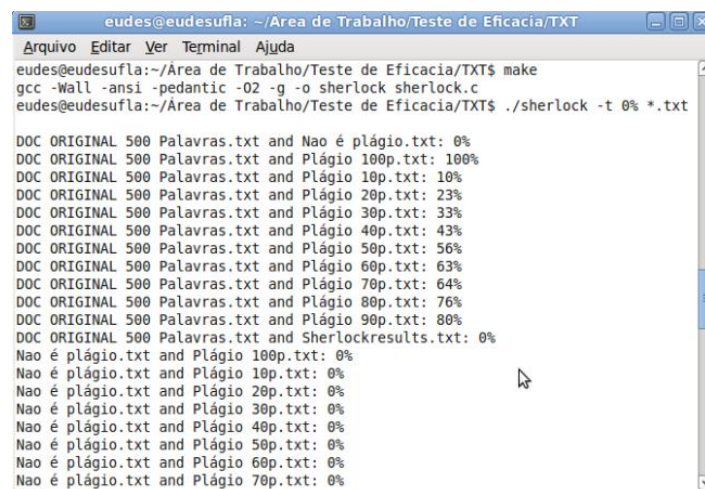
- *threshold*: Apenas similaridades acima do valor passado são exibidas. O padrão adotado é 20%;

- *zerobits*: Controla a “granularidade” da comparação. Quanto maior esse número, mais grosseira, mais rápida e menos precisa é a detecção. Esse parâmetro pode variar de 0 a 31. O padrão adotado 4;
- *number of words*: Especifica quantas palavras são usadas para formar uma assinatura digital. Quanto mais elevado for este parâmetro, mais lenta consequentemente, são mais exata. O padrão é de 3 palavras, que funciona bem na maioria dos casos;
- *outfile*: Opção utilizada para especificar o arquivo de saída.

Exemplo de comando usado na comparação de dois documentos:

```
>> sherlock -t 40% documento1.txt documento2.txt
```

Com esse comando, a ferramenta compara o documento1.txt com o documento2.txt. O *threshold* foi definido em 40%, ou seja, os resultados são exibidos apenas se os documentos comparados possuírem similaridade acima de 40%. Neste exemplo, os parâmetros *zerobits* e *number of words* não foram definidos.



```
eudes@eudesufila: ~/Área de Trabalho/Teste de Eficacia/TXT
Arquivo Editar Ver Terminal Ajuda
eudes@eudesufila:~/Área de Trabalho/Teste de Eficacia/TXT$ make
gcc -Wall -ansi -pedantic -O2 -g -o sherlock sherlock.c
eudes@eudesufila:~/Área de Trabalho/Teste de Eficacia/TXT$ ./sherlock -t 0% *.txt

DOC ORIGINAL 500 Palavras.txt and Nao é plágio.txt: 0%
DOC ORIGINAL 500 Palavras.txt and Plágio 100p.txt: 100%
DOC ORIGINAL 500 Palavras.txt and Plágio 10p.txt: 10%
DOC ORIGINAL 500 Palavras.txt and Plágio 20p.txt: 23%
DOC ORIGINAL 500 Palavras.txt and Plágio 30p.txt: 33%
DOC ORIGINAL 500 Palavras.txt and Plágio 40p.txt: 43%
DOC ORIGINAL 500 Palavras.txt and Plágio 50p.txt: 56%
DOC ORIGINAL 500 Palavras.txt and Plágio 60p.txt: 63%
DOC ORIGINAL 500 Palavras.txt and Plágio 70p.txt: 64%
DOC ORIGINAL 500 Palavras.txt and Plágio 80p.txt: 76%
DOC ORIGINAL 500 Palavras.txt and Plágio 90p.txt: 80%
DOC ORIGINAL 500 Palavras.txt and Sherlockresults.txt: 0%
Nao é plágio.txt and Plágio 100p.txt: 0%
Nao é plágio.txt and Plágio 10p.txt: 0%
Nao é plágio.txt and Plágio 20p.txt: 0%
Nao é plágio.txt and Plágio 30p.txt: 0%
Nao é plágio.txt and Plágio 40p.txt: 0%
Nao é plágio.txt and Plágio 50p.txt: 0%
Nao é plágio.txt and Plágio 60p.txt: 0%
Nao é plágio.txt and Plágio 70p.txt: 0%
```

**Figura 9.** Sherlock

O resultado apresentado é uma lista de pares de documentos analisados e os respectivos índices de similaridades. Esse índice é uma

porcentagem que varia de 0%, que indica a ausência de similaridade, a 100% que indica alta similaridade. Quando obtém 100%, não significa que os arquivos são exatamente iguais, pois a ferramenta elimina alguns dados para acelerar o processo de análise. Esses resultados podem ser visualizados no terminal ou em um arquivo TXT caso o *outfile* seja especificado. O relatório TXT gerado é bem simples, contém apenas os pares de documento e a similaridade entre eles.

No *site* oficial da ferramenta, não possui FAQ. Para fazer *download* do código fonte e fazer uso dos recursos não é necessário cadastrar. A ferramenta não possui manual, porém as instruções de uso podem ser encontradas no *site*: <http://sydney.edu.au/engineering/it/~scilect/sherlock/>.

#### 4.5.3. CopyCatch Gold

CopyCatch Gold foi criada pela CFL *Software Development*. A ferramenta é a evolução de um conjunto de ferramentas de análises linguística e é destinada principalmente a Instituições de ensino (CFL SOFTWARE LIMITED, 2011).

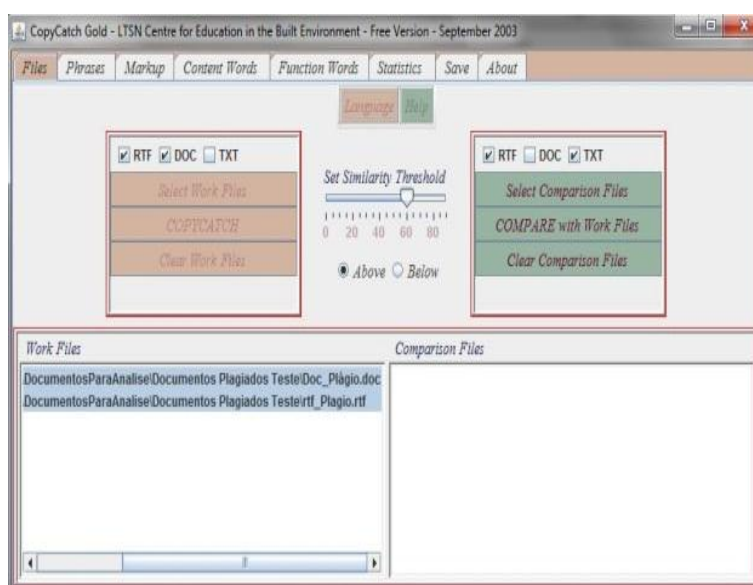
É uma ferramenta de detecção de plágio gratuita foi desenvolvida utilizando a linguagem de programação JAVA. Para utilizá-la é necessário possuir o *Java Runtime Environment*. CopyCatch Gold é uma ferramenta projetada para executar localmente em diversos sistemas operacionais.

O CopyCatch Gold possui uma interface gráfica simples e foi projetada para ser facilmente utilizada. A ferramenta é organizada em abas como mostra a Figura 10. Suas principais funções foram agrupadas nas seguintes abas:

- *Files*: Escolha da extensão dos documentos, seleção dos documentos, ajuste do *threshold*, início da análise e ajuda;
- *Phrases*: Exibe a similaridade entre pares de documentos analisados, as frases relacionadas e opções de marcação dos trechos similares;

- *Markup*: Caso os documentos tenham sido marcados na aba *Phrases*, é nessa aba que as partes similares dos documentos analisados são destacadas;
- *Content work* e *Function work*: Exibe as frequências das palavras principais e dos conectivos;
- *Statistics*: Exibe uma tabela com o nome do documento, frequência das palavras, frequência dos conectivos e uma porcentagem de compartilhamento dessas palavras;
- *Save*: Opção para salvar o relatório da análise.

A ferramenta permite a seleção de vários documentos. As extensões suportadas são DOC, TXT e RTF. Na tela inicial, é possível ajustar o *threshold* para exibir as ocorrências acima do nível determinado.



**Figura 10.** Tela inicial do CopyCatch Gold

A ferramenta compara os documentos selecionados aos pares, não possui versão WEB e não faz análises em código fonte. Durante as análises, as opções não podem ser acessada. Ao término da análise, os resultados são processados e exibidos na aba *Phrases*.

Através da aba *Phrases*, é possível verificar a similaridade e as frases compartilhadas entre os documentos. Aqueles que sofreram marcação podem ser visualizados na aba *Markup*.

A aba *save* possui opções relacionadas ao relatório. É possível selecionar a extensão do relatório que pode ser HTML e RTF. Além do mais, é possível escolher o conteúdo que terá no relatório, pode gerar um relatório simples contendo apenas o par de documento analisado e a similaridade entre eles ou um relatório complexo contendo as frases compartilhadas e suas ocorrências.

No *site* da ferramenta não possui FAQ e não é preciso cadastrar. O manual e a ferramenta podem ser adquiridos através do *site*: <http://cebe.cf.ac.uk/learning/Plagiarism/index.php>. Outras informações podem ser obtidas no *site*: <http://www.cflsoftware.com/>.

#### **4.5.4. WCopyFind**

Segundo Bloomfield (2010), WCopyfind é uma ferramenta de detecção de plágio desenvolvida por Louis A. Bloomfield, professor de física da Universidade da Virgínia. Foi desenvolvida usando a linguagem Microsoft Visual C++.Net e teve sua primeira versão disponível em 2002.

É uma ferramenta gratuita, projetada para executar localmente. Além do mais é múltiplaplataforma, pois pode executar em diferentes sistemas operacionais. Atualmente, possui versões para Linux e Windows. Para fazer o *download* da ferramenta, não é necessário cadastrar.

WCopyFind é uma ferramenta simples, objetiva e fácil de usar. As funções estão dispostas em uma tela como mostra a Figura 11. Na parte superior, ocorre a seleção dos documentos e são suportadas 500 documentos nas extensões TXT e DOC.

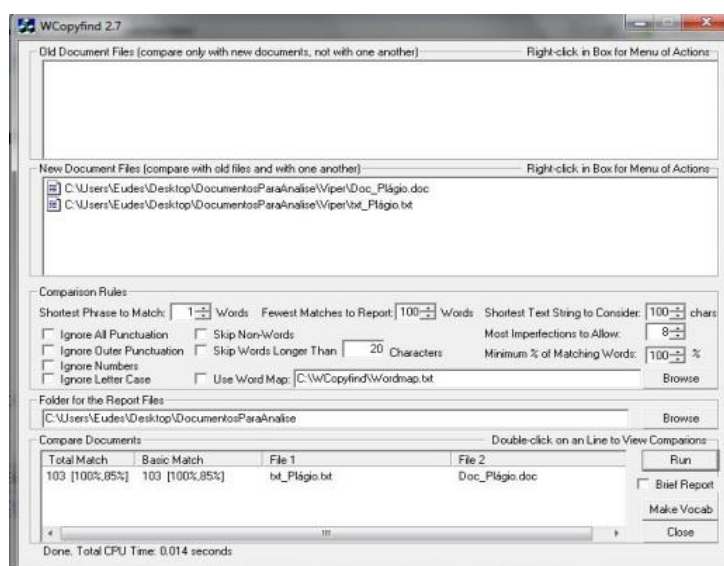
O campo de seleção, alguns parâmetros de comparação podem ser configurados. Dependendo da configuração, pode aumentar ou diminuir a



eficiência da ferramenta, assim como o tempo de comparação. São exemplos de parâmetros de comparação da ferramenta:

- Ignorar toda a pontuação;
- Ignorar somente os pontos que delimita o fim do parágrafo;
- Ignorar os números;
- Ignorar caracteres especiais, exceto hífen e apóstrofo.

É possível escolher o local onde os relatórios são salvos. Depois de cada análise, os resultados são exibidos em forma de tabela na parte inferior. A tabela gerada exibe o nome dos documentos analisados, a similaridade entre eles e a quantidade de comparações realizadas. Além disso, exibe o tempo gasto em cada análise.



**Figura 11.** Tela inicial da ferramenta WCopyFind

A ferramenta não estende suas análises a WEB e não realiza detecção em código fonte. O WCopyFind gera um relatório em HTML contendo o texto do documento com as frases semelhantes sublinhadas.

No relatório, possui as configurações usadas nas análises e uma tabela contendo o número de buscas realizado e a similaridade dos documentos. É possível abrir os documentos separadamente com as partes

similares destacadas ou abrir os documentos lado a lado. O manual e a ferramenta podem ser encontrados em: <http://www.plagiarism.phys.virginia.edu/>, Na página da ferramenta, ainda é possível encontrar as Perguntas Frequentemente Respondidas (FAQ).

#### **4.5.5. DOC Cop**

Desenvolvida por Marcos McCrohon, a ferramenta DOC Cop à princípio visava a auxiliar os professores de uma Universidade australiana a verificar similaridades entre trabalhos de estudantes. Mas, devido ao sucesso obtido pela ferramenta, hoje é adotada por pesquisadores, jornalistas, editores dentre outros (DOC Cop, 2011).

DOC Cop é uma ferramenta gratuita oferecida como um serviço WEB. Para utilizá-la, é necessário ter acesso à internet e um navegador WEB como Firefox ou Internet Explorer. Além disso, para realizar as análises, é preciso de um ID obtido através de um cadastro. Por se baseada em navegadores WEB, essa ferramenta pode ser utilizada por vários sistemas operacionais.

A ferramenta possui uma interface gráfica simples, limpa e objetiva. Com ela, é possível fazer análises entre documentos no formato DOC, DOCX e PDF ou em trechos de texto na WEB, não há opção para análise em código fonte. A ferramenta oferece opções que reduz o tempo das análises, porém, dependendo da configuração utilizada, haverá um grande número de falsos-positivo.

A ferramenta possui um indicador de processamento, onde exhibe informações como: horário de início da análise, ID do usuário realizando a análise, fila de requisições e um indicador para informar se o tempo de carga está rápido, normal ou demorado.

As análises nessa ferramenta são demoradas, pois nela o tempo de carga, tempo comparação dos documentos, tempo de espera caso exista fila e o tempo de *feedback*. Ao término de cada análise, o relatório gerado é

enviado para o *e-mail* do usuário e os documentos submetidos são deletados de seu servidor.

DOC Cop possui a opção que permite fazer detecção na internet, mas, diferente da análise entre documentos, que precisa-se submeter um document. Na análise WEB, basta copiar e colar o trecho suspeito e aguardar o resultado.

É possível selecionar em qual extensão o relatório será gerado, podendo ser nos formatos HTML DOC ou DOCX. O relatório possui a porcentagem de similaridade entre os documentos. As partes consideradas suspeitas destacadas e o horário que o relatório foi gerado. Quando se trata de uma análise de um trecho de texto na WEB, o *link* da fonte é incluso no relatório.

Na tela de submissão dos documentos, o botão enviar e as opções de configurações não ficam aparentes e não possui barra de rolagem como mostra a Figura 12, dessa forma impedem alguns usuários utilizem a ferramenta.

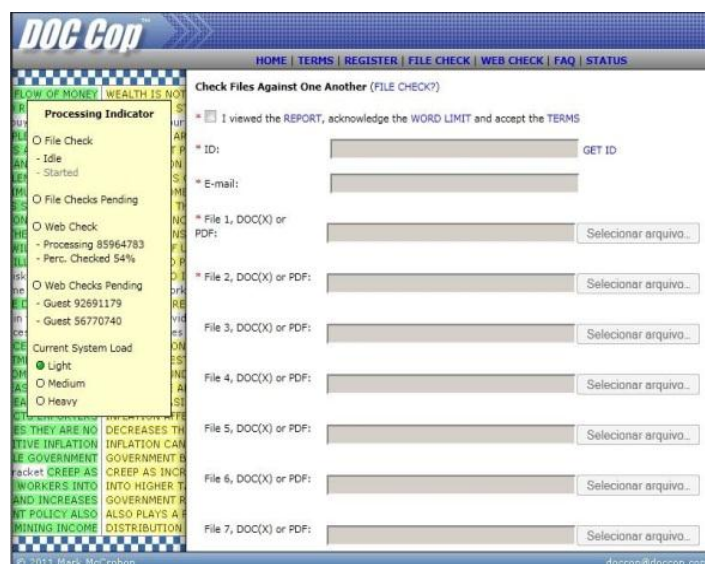


Figura 12. Tela para submissão de arquivos.

A ferramenta possui FAQ, mas não possui manual. Para utilizar a ferramenta, basta acessar o *site*: <http://www.doccop.com>.

#### 4.6. Avaliação e discussão dos dados

Esta seção apresenta a avaliação qualitativa e quantitativa das ferramentas selecionadas e a discussão dos resultados obtidos.

##### 4.6.1. Avaliação qualitativa

Apresenta-se no Quadro 3 um resumo das respostas dos critérios de comparação propiciando uma comparação direta entre as ferramentas.

**Quadro 3.** Comparativo entre as ferramentas estudadas.

<b>Crítérios de Comparação</b>	<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WCopFind</b>	<b>DOC Cop</b>
<b>WEB</b>	Não	Não	Não	Não	Sim
<b>Análise em código fonte</b>	Sim	Sim	Não	Não	Não
<b>Apresentação dos resultados</b>	**	*	**	**	***
<b>Extensões suportadas</b>	DOC, RTF, TXT, PDF	TXT	RTF, DOC, TXT	TXT, DOC	PDF, DOC, DOCX
<b>Manual</b>	Sim	Não	Sim	Sim	Não
<b>FAQ</b>	Não	Não	Não	Sim	Sim
<b>Múltiplaplataforma</b>	Sim	Sim	Sim	Sim	Sim
<b>Usabilidade</b>	***	*	**	***	**
<b>Análise na WEB</b>	Não	Não	Não	Não	Sim
<b>Cadastro</b>	Não	Não	Não	Não	Sim

Das ferramentas comparadas verifica-se que DOC Cop é a única WEB, necessita de cadastro e faz análises de trechos de texto em documentos presente na Internet, através do motor de busca do BING.

As ferramentas Ferret, CopyCatch Gold e WCopyFind possuem manual detalhado de suas funções. No entanto, apenas as ferramentas DOC Cop e WCopyFind possuem FAQ em seu *website* oficial.

As cinco ferramentas apresentadas são múltiplaplataforma. Além disso, permitem análises em mais de um formato, exceto a ferramenta Sherlock que permite análises apenas em documentos no formato TXT. Das ferramentas, apenas o Ferret 4.0 e Sherlock detectam plágio em códigos fontes.

A ferramenta Sherlock foi a única classificada como ruim nos critérios usabilidade e na apresentação dos resultados. A falta de uma interface gráfica tornou-a de difícil utilização. As ferramentas WCopyFind e Ferret apresentaram melhor usabilidade.

No critério apresentação dos resultados, apenas a ferramenta DOC Cop foi classificada como boa, mas, em geral, as ferramentas apresentam relatórios mal organizados, com informações irrelevantes ou poucas informações. Em alguns casos o relatório é de difícil compreensão.

#### **4.6.2. Avaliação quantitativa**

Para compor a base de testes, foram utilizados 47 livros retirados do Projeto Gutenberg (PROJECT GUTENBERG, 2011), onde encontram-se uma grande quantidade de livros digitalizados e disponíveis.

Os valores apresentados nos quadros a seguir são a média dos resultados obtidos em 5 execuções, sendo cada execução um grupo de documento distinto.

A aplicação dos testes de eficácia, sensibilidade e desempenho, assim como os resultados obtidos encontram-se nas seções seguintes.

##### **4.6.2.1. Eficácia**

Para realizar o teste a eficácia das ferramentas, foram criados dez documentos a partir de um documento original de 500 palavras. Os

documentos possuíam similaridades diferentes e conhecidas. Para mostrar 0%, foi utilizado um documento de 500 palavras com texto diferente. Aplicando este teste, esperam-se os seguintes resultados:

- Documento original e doc diferente: 0%
- Documento original e doc 1: 10%
- Documento original e doc 2: 20%
- Documento original e doc 3: 30%
- ...
- Documento original e doc 8: 80%
- Documento original e doc 9: 90%
- Documento original e doc 10: 100% (cópia exata)

**Quadro 4.** Análise de eficácia das ferramentas de detecção de plágio selecionadas.

<b>Documentos / % esperada</b>	<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WcopyFind</b>	<b>DOC Cop</b>
<b>Documento diferente / 0%</b>	0%	0%	21%	15%	0%
<b>Documento 1 / 10%</b>	9%	8%	23%	10%	55%
<b>Documento 2 / 20%</b>	18%	22%	39%	20%	60%
<b>Documento 3 / 30%</b>	27%	33%	52%	30%	65%
<b>Documento 4 / 40%</b>	39%	48%	63%	40%	70%
<b>Documento 5 / 50%</b>	48%	55%	72%	50%	75%
<b>Documento 6 / 60%</b>	59%	66%	80%	60%	80%
<b>Documento 7 / 70%</b>	69%	74%	85%	70%	85%
<b>Documento 8 / 80%</b>	78%	82%	90%	79%	90%
<b>Documento 9 / 90%</b>	87%	88%	95%	90%	95%
<b>Documento 10 / 100%</b>	100%	100%	100%	100%	100%

Para obter uma pontuação que indique a melhor ferramenta, as similaridades obtidas no Quadro 4 foram subtraídas das similaridades

esperadas. Em seguida, o módulo dos valores foram tirados e, por fim, foi realizado o somatório dos valores das colunas do Quadro 5. Neste teste, a ferramenta que possui o menor somatório, é considerada a melhor.

**Quadro 5.** Resultado da subtração entre similaridade obtida e similaridade esperada em módulo.

	<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WCopFind</b>	<b>DOC Cop</b>
	0	0	21	15	0
	1	2	13	0	45
	2	2	19	0	40
	3	3	22	0	35
	1	8	23	0	30
	2	5	22	0	25
	1	6	20	0	20
	1	4	15	0	15
	2	2	10	1	10
	3	2	05	0	5
	0	0	0	0	0
<b>Somatório das colunas:</b>	<b>17</b>	<b>35</b>	<b>169</b>	<b>16</b>	<b>224</b>

Verifica-se que as ferramentas não identificam o plágio como esperado. Os motivos podem ser vários, como: técnicas utilizadas, os algoritmos de comparação ou os métodos utilizados para calcular a similaridade entre os documentos.

Dentre os testes de eficácia realizados, as ferramentas CopyCatch Gold e DOC Cop apresentaram os maiores somatório, pois foram as que apresentaram os piores resultados, as similaridades encontradas foram muito acima das similaridades esperadas.

A ferramenta WCopyFind obteve os melhores resultados, pois apresentou as menores variações nas similaridades encontradas, conseqüentemente, o menor somatório.

As ferramentas CopyCatch Gold e WCopyFind foram as únicas ferramentas que apresentaram similaridades diferentes de 0% nos testes realizados com documentos diferentes. As ferramentas comparadas mostraram-se eficazes na análise de cópias exatas.

#### 4.6.2.2. Sensibilidade

Para compor o teste de sensibilidade, foram usados 10 documentos modificados a partir de um original de 500 palavras. As modificações feitas foram organizadas em três grupos:

- **Deslocamento:** parágrafos e frases foram deslocados aleatoriamente;
- **Inserção:** foram inseridas no documento original novas palavras. Essas palavras foram organizadas em parágrafos;
- **Paráfrase:** o texto original teve trechos parafraseados, mantendo o sentido original e a mesma quantidade de palavras do documento original.

Variando os documentos de entrada e efetuando a comparação com o documento original, espera-se que as medidas exibidas pelas ferramentas sejam sempre de 100%, ou seja, independente da alteração que tiver no documento a ferramenta deve ser capaz de constatar similaridades.

Os resultados dos testes realizados podem ser conferidos no Quadro 6.



**Quadro 6.** Análise de Sensibilidade das ferramentas selecionadas.

		<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WCoppyFind</b>	<b>DOC Cop</b>
<b>Deslocamento</b>	<b>Parágrafos.</b>	96%	96%	100%	100%	100%
	<b>Frases.</b>	90%	90%	100%	100%	100%
	<b>Parágrafos e frases.</b>	90%	81%	100%	100%	99%
<b>Inserção</b>	<b>100 novas palavras.</b>	85%	82%	92%	100%	88%
	<b>300 novas palavras.</b>	65%	57%	82%	100%	79%
	<b>500 novas palavras.</b>	56%	43%	74%	100%	72%
<b>Paráfrase</b>	<b>de 05% do Texto.</b>	93%	93%	99%	98%	94%
	<b>de 15% do texto.</b>	84%	80%	97%	95%	86%
	<b>de 30% do texto.</b>	73%	71%	95%	91%	76%
	<b>de 50% do texto.</b>	60%	54%	92%	86%	64%

Para obter uma pontuação que indique a melhor ferramenta, as similaridades obtidas no Quadro 6 foram subtraídas das similaridades esperadas (neste caso sempre será 100%). Em seguida, foram calculados os somatórios dos valores das colunas do Quadro 7. Considera-se a melhor ferramenta neste teste aquela que possuir o somatório mais próximo de zero.

**Quadro 7.** Resultado da subtração entre similaridade obtida e similaridade esperada.

	<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WcopyFind</b>	<b>DOC Cop</b>
	4	4	0	0	0
	10	10	0	0	0
	10	19	0	0	1
	15	18	8	0	12
	35	43	18	0	21
	44	57	26	0	28
	7	7	1	2	6
	16	20	3	5	14
	27	29	5	9	24
	40	46	8	14	36
<b>Somatório das Colunas:</b>	<b>207</b>	<b>254</b>	<b>69</b>	<b>30</b>	<b>142</b>

Verifica-se que as cinco ferramentas são sensíveis a determinadas alterações, ou seja, documentos plagiados que sofrem alterações podem não ser identificados como plágio em algumas ferramentas ou podem apresentar baixa similaridade.

As ferramentas WCopyFind e CopyCatch Gold mostraram-se insensíveis a deslocamento de texto. No entanto, somente a ferramenta WCopyFind mostrou-se insensível a inserção de texto. No que se refere à paráfrase, CopyCatch Gold mostrou-se pouco sensível.

As ferramentas Sherlock, Ferret e DOC Cop apresentaram os maiores somatórios, pois mostraram-se sensíveis nos três grupos analisados. A ferramenta WCopyFind apresentou o menor somatório. Para esse teste, foi considerada a melhor ferramenta.

#### **4.6.2.3. Desempenho**

Para o teste de desempenho, foram utilizados 10 documentos que variam de 10.000 palavras (25 páginas) a 100.000 palavras (180 páginas) com intervalo de 10.000 palavras como mostrado abaixo.

- Documento 1 = 10.000 palavras;
- Documento 2 = Documento 1 + 10.000 palavras; (totalizando 20.000 palavras);
- ...
- Documento 10 = Documento 9 + 10.000 palavras. (totalizando 100.000 palavras).

Para cada ferramenta, os testes foram realizados com três grupos de documentos, são eles:

- **Grupo A** - Documentos idênticos (100% plágio);
- **Grupo B** - Documentos semelhantes (50% plágio);
- **Grupo C** - Documentos distintos (0% plágio).

As análises foram realizadas da seguinte forma: Documento1 (original de 10.000 palavras) foi comparado com Documento1 do grupo A (10.000 palavras - cópia exata); Documento1 (original de 10.000 palavras) foi comparado com Documento 1 do grupo B (10.000 palavras – 50% plagiado); Documento1 (original de 10.000 palavras) foi comparado com Documento 1 do grupo C (10.000 palavras - Diferente). Esse procedimento foi repetido nos documentos, até chegar em Documento10 (100.000) palavras.

O tempo mostrado no Quadro 8, não é somente o tempo gasto nas comparações, mas desde o momento que o documento foi submetido a análise até o momento em que os resultados foram exibidos, ou seja, inclui o tempo de processamento do documento, de comparação, processamento dos resultados.

**Quadro 8.** Análise de desempenho considerando os grupos A, B e C.

	Ferret			Sherlock			CopyCatch Gold			WcopyFind			DOC Cop		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
<b>Documento 1</b>	0.34s	0.38s	0.37s	-	-	-	2.27s	1.84s	1.35s	0.70s	0.95s	0.94s	434.47s	313.57s	198.21s
<b>Documento 2</b>	0.38s	0.45s	0.43s	-	-	-	13.37s	7.34s	5.14s	0.98s	1.13s	1.03s	1738.33s	928.48s	275.84s
<b>Documento 3</b>	0.45s	0.50s	0.48s	-	-	-	28.56s	16.40s	11.58s	1.12s	1.22s	1.23s	3525.18s	1664.48s	902.78s
<b>Documento 4</b>	0.51s	0.57s	0.57s	-	-	-	48.89s	30.50s	21.16s	1.17s	1.34s	1.32s	6203.48s	3191.43s	1237.61s
<b>Documento 5</b>	0.61s	0.62s	0.64s	-	-	-	77.23s	46.22s	34.15s	1.27s	1.40s	1.44s	9494.93s	8220.03s	1457.57s
<b>Documento 6</b>	0.66s	0.72s	0.70s	-	-	-	112.47s	68.22s	45.81s	1.32s	1.48s	1.51s	-	-	-
<b>Documento 7</b>	0.72s	0.79s	0.76s	0.19s	0.20s	0.19s	147.84s	94.23s	60.23s	1.41s	1.63s	1.57s	-	-	-
<b>Documento 8</b>	0.78s	0.84s	0.81s	0.22s	0.23s	0.22s	190.82s	127.70s	91.51s	1.49s	1.67s	1.66s	-	-	-
<b>Documento 9</b>	0.88s	0.90s	0.89s	0.27s	0.28s	0.27s	247.51s	165.06s	122.56s	1.55s	1.76s	1.73s	-	-	-
<b>Documento 10</b>	1.01s	1.02s	1.06s	0.32s	0.32s	0.31s	317.07s	216.38s	175.83s	1.62s	1.85s	1.79s	-	-	-
<b>Somatório da Coluna:</b>	6.36s	6.78s	6.69s	1.00s	1.02s	1.00s	1186.03s	773.89s	569.33s	12.64s	14.43s	14.21s	21396.39s	14317.99s	4072.01s
<b>Soma Total:</b>	19.83s			3.02s			2529.25s			41.28s			39786.39s		

A ferramenta Sherlock apresentou tempo de análise extremamente baixo quando foram testados os documentos de 10.000 a 60.000 palavras. Desta forma, inviabilizou o uso do cronômetro. Das ferramentas comparadas, Sherlock apresentou os menores tempos de execução. Os tempos de análise nos três grupos testados tiveram pouca variação.

Nos testes realizados, as ferramentas WCopyFind e Ferret 4.0 mostraram-se mais rápidas nas comparações dos documentos do grupo A, ou seja, quando há 100% de plágio. No entanto, mostraram-se lentas nas análises do grupo B, quando há 50% de similaridade. Entre essas duas ferramentas, a Ferret 4.0 apresentou os melhores tempos.

A ferramenta CopyCatch Gold, diferente das outras ferramentas mostrou-se mais lenta nas comparações de documentos idênticos e mais rápido nos documentos não plagiados.

A ferramenta que apresentou os piores resultados foi a DOC Cop, pois o somatório dos tempos foi elevado. Entretanto, deve-se considerar que a ferramenta DOC Cop possui tempo de carga do documento, fila de espera e tempo de envio do resultado, por ser WEB,

#### **4.6.3. Classificação geral**

O Quadro 9 apresenta uma classificação geral das ferramentas baseando-se nos resultados dos testes obtidos neste trabalho. Para determinar a melhor ferramenta, foram calculados os somatórios e os produtórios das posições obtidas pelas ferramentas em cada teste. A melhor ferramenta é a que possuir menor somatório. O produtório será utilizado como critério de desempate dos somatórios.

**Quadro 9.** Classificação geral das ferramentas.

	<b>Ferret</b>	<b>Sherlock</b>	<b>CopyCatch Gold</b>	<b>WCopYFind</b>	<b>DOC Cop</b>
<b>Teste de eficácia</b>	2°	3°	4°	1°	5°
<b>Teste de sensibilidade</b>	4°	5°	2°	1°	3°
<b>Teste de desempenho</b>	2°	1°	4°	3°	5°
$\Sigma$	8	9	10	5	13
<b>II</b>	16	15	32	3	75

Baseando-se nos valores da Tabela 8, verifica-se que a ferramenta WCopYFind apresentou os melhores resultados e se sobressaiu em detrimento das outras, pois, apresentou o menor somatório.

## 5. ESTUDO DE CASO

Após listar, descrever e comparar as principais ferramentas de detecção de plágio, verificou-se que apenas a ferramenta DOC Cop é oferecida como um serviço WEB. No entanto, os documentos enviados para análise não são comparados com documentos presente na WEB onde encontram-se grande parte das fontes de plágio.

Baseando-se nesta falta de ferramentas que receba um documento e faça análise na WEB, este trabalho de conclusão de curso propõe um protótipo capaz de comparar os documentos submetidos para análise com documentos presente na WEB através dos motores de busca existente.

O propósito do protótipo não é acusar o plágio, mas identificar fontes suspeitas. Ainda caberá ao utilizador verificar a fonte suspeita e confirmar o plágio.

### 5.1. Implementação

Para o desenvolvimento do protótipo, utilizou-se a linguagem JAVA com tecnologia JEE, banco de dados MySQL responsável pelo armazenamento dos dados dos usuários e das análises realizadas. Como container WEB, utilizou-se o Apache Tomcat 6 e a IDE utilizada para desenvolvimento foi o Netbeans 6.8.

Para realizar os testes, foram utilizados um Computador Core 2 Quad com 4 Gbytes de memória RAM, um disco rígido Serial ATA II de 7200 RPM, rodando Windows 7 Ultimate, e o navegador Mozilla Firefox 7.

O mecanismo de busca adotado foi o Microsoft BING. Para realizar as buscas, utilizou-se a *Application Programming Language* (API) Java disponibilizada pela empresa para desenvolvimento.

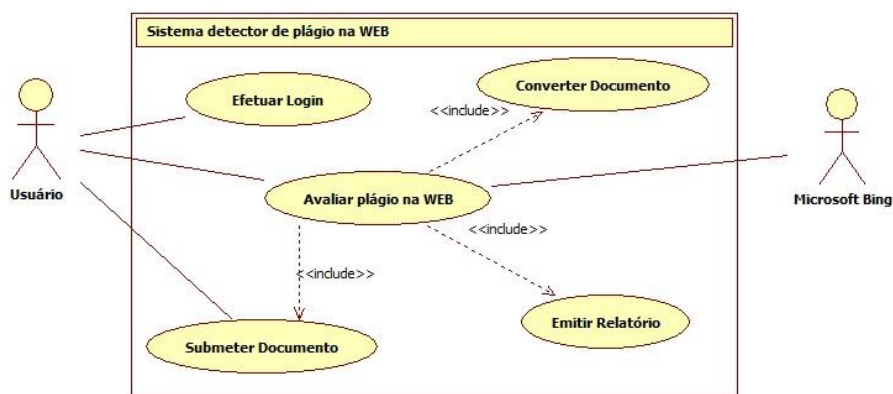
Os formatos permitidos pelo protótipo são PDF e DOC. Para manipulá-los e convertê-los em formato de texto puro, foram utilizados as API PDFBox (para manipular os arquivos no formato PDF) e Apache POI (para manipular os documentos DOC). Para realizar o *upload* dos documentos, utilizou-se a API *Jakarta Commons-FileUpload*.

## 5.2. Modelagem do sistema

A modelagem foi feita utilizando a linguagem *Unified Modeling Language* (UML) para visualizar o sistema antes de ser construído. Foram criados os diagramas de caso de uso, de atividades e de classe nas seções seguintes.

### 5.2.1. Diagrama de caso de uso

A Figura 13 apresenta o diagrama de caso de uso elaborado para propiciar uma visão geral do sistema proposto e representar graficamente os casos de uso, os atores e os relacionamentos entre eles.

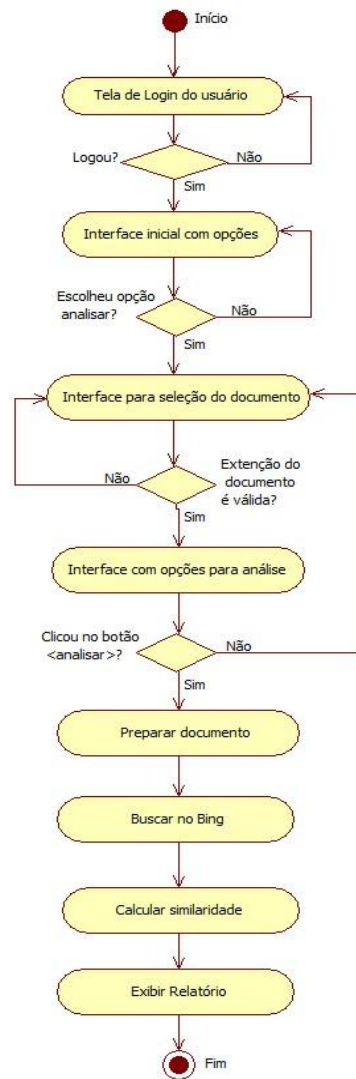


**Figura 13.** Diagrama de caso de uso



### 5.2.2. Diagrama de atividades

A Figura 14 apresenta o diagrama de atividades criado para mostrar o fluxo das atividades realizadas pelo sistema proposto.



**Figura 14.** Diagrama de atividades

### 5.2.3. Diagrama de classes

A Figura 15 e 16 apresentam o diagrama de classes para representar a estrutura das principais classes do sistema e estabelecer a relação entre elas.

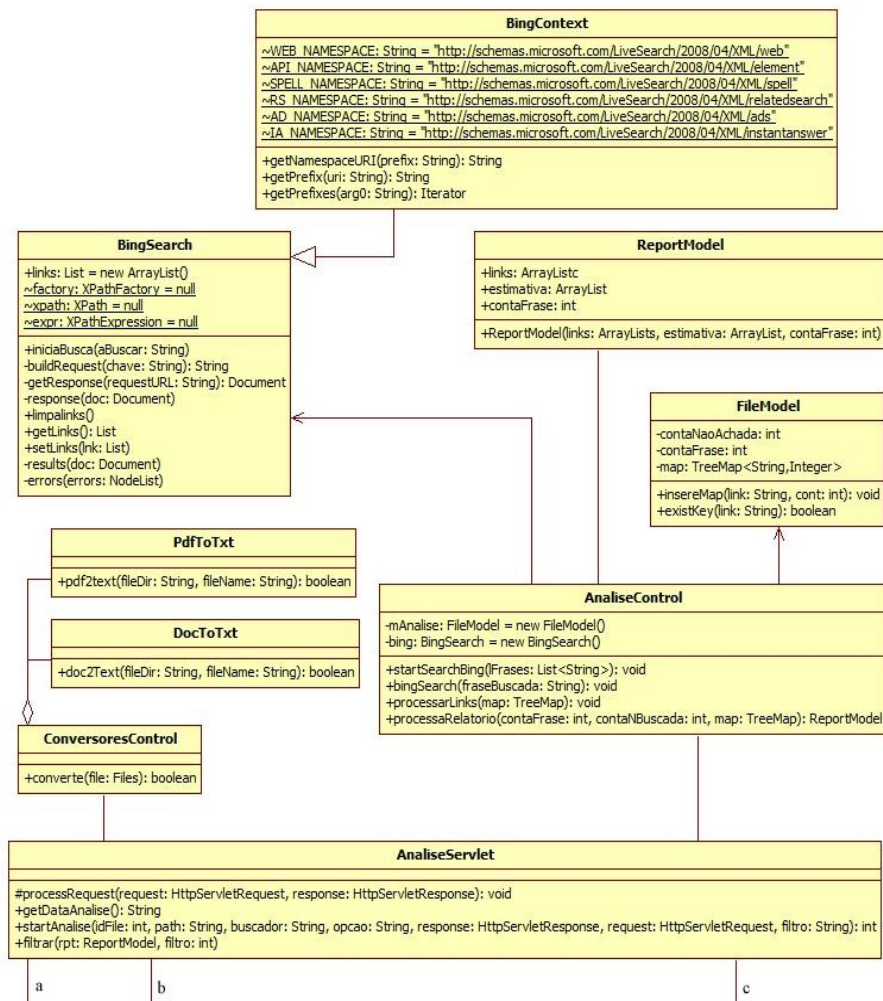


Figura 15. Diagrama de classes parte 1

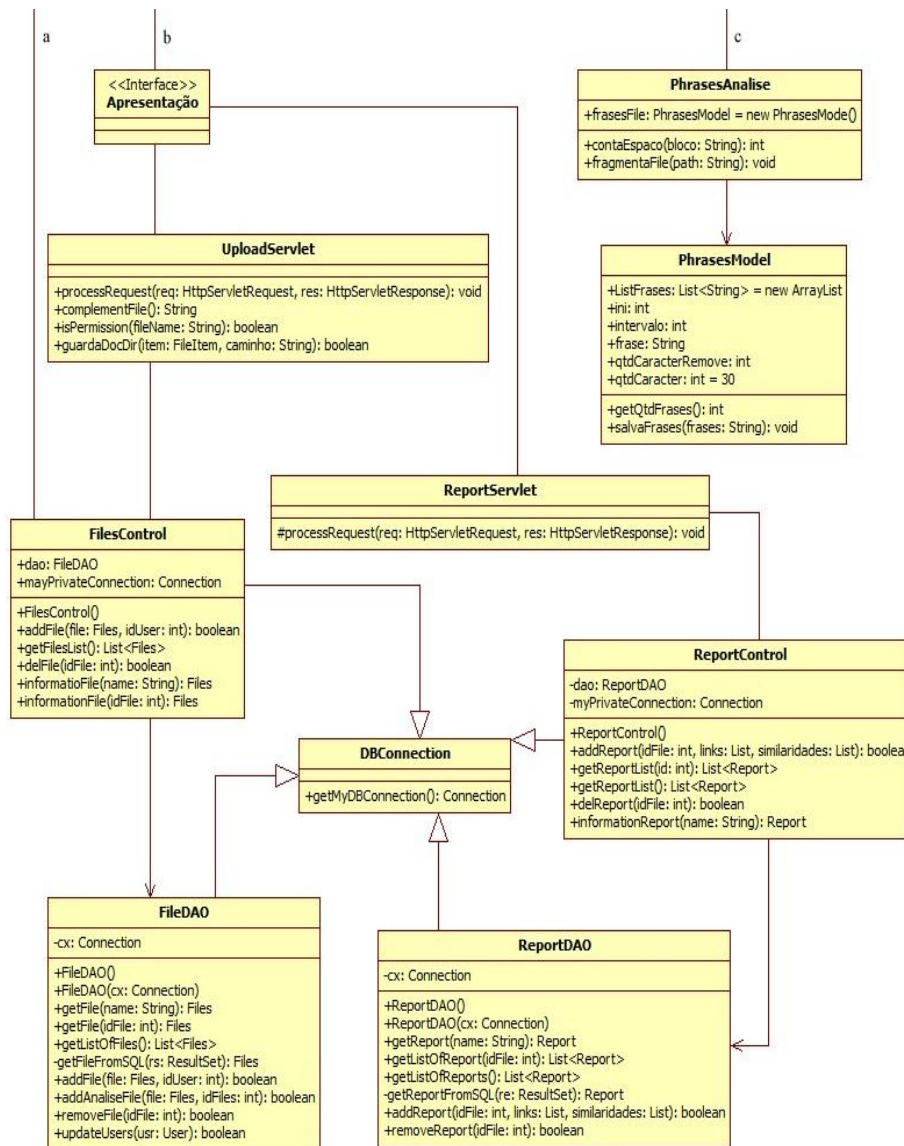
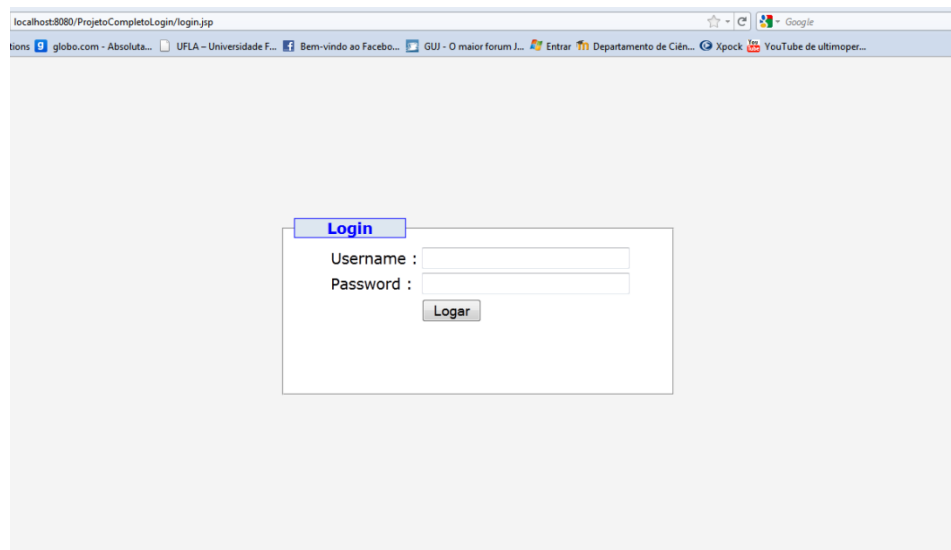


Figura 16: Diagrama de classes parte 2

### 5.3. Detalhes do protótipo desenvolvido

O protótipo desenvolvido é exclusivamente destinado a identificar similaridades em documentos presentes na WEB através do motor de busca do BING.

O protótipo dispõe inicialmente de uma tela de *login* como mostra a Figura 17. Após efetuar o *login*, o usuário é redirecionado para a tela inicial onde tem acesso as funções disponíveis no sistema (Figura 18).



**Figura 17.** Tela de *login* do protótipo



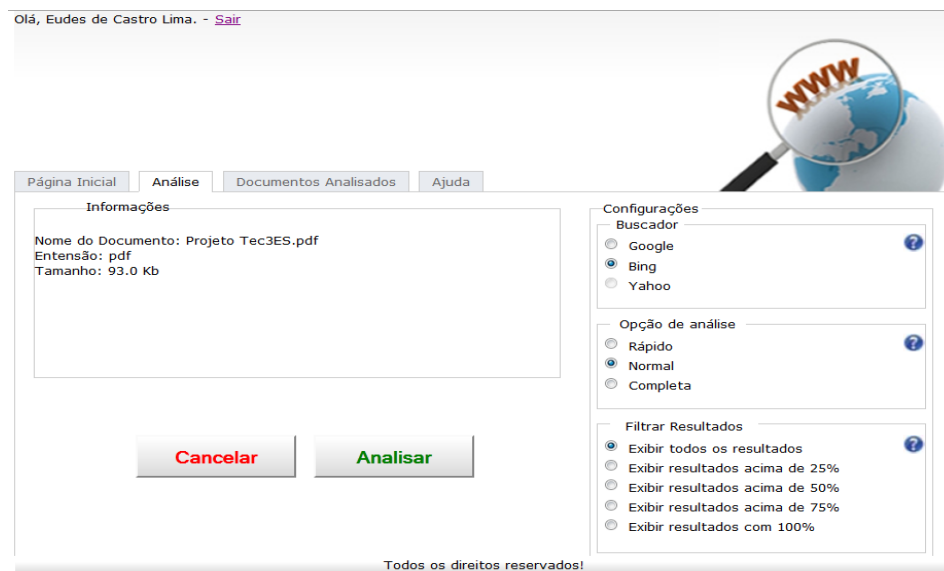
**Figura 18.** Tela inicial do protótipo

Quando o usuário seleciona a opção de análise, na tela inicial é encaminhado para a tela de *upload* onde deve fazer a seleção e o envio do documento suspeito (Figura 19). Feito a seleção, o documento é enviado para um diretório no servidor e as principais informações como: nome, tamanho, extensão e caminho são armazenados em um banco de dados.



**Figura 19.** Tela de submissão do documento suspeito

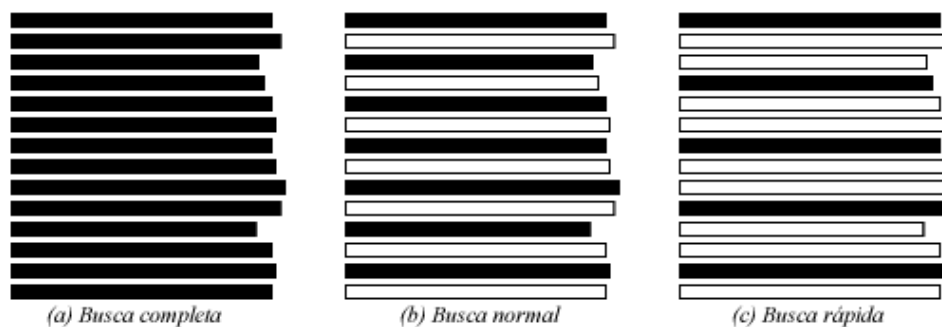
Automaticamente, após o envio, o usuário é encaminhado para a tela de opções de análise (Figura 20). As opções oferecidas são: selecionar o buscador, tipo de análise e filtro dos resultados.



**Figura 20.** Tela de opções de análise do protótipo

A opção selecionar o buscador delimita o motor de busca usado na análise. Inicialmente, a proposta era fazer as análises utilizando os motores de busca Google, BING e Yahoo, porém o único buscador que não restringiu a quantidade de buscas e apresentou condições favoráveis para desenvolvimento do detector de plágio foi o BING.

O objetivo da opção de análise é oferecer a opção de reduzir o tempo das análises. As opções oferecidas são: busca completa, busca normal e busca rápida. Na busca completa, o documento completo é analisado (Figura 21 (a)). Na busca normal, 1/2 do documento é analisado (Figura 21 (b)). Na busca rápida, apenas 1/3 do documento é analisado (Figura 21 (c)).



**Figura 21.** Tipos de busca

A opção filtro dos resultados estabelece um limite para exibição das ocorrências encontradas. Se a opção exibir resultados for acima de 50% as ocorrências abaixo de 50% são removidas.

Quando o botão analisar é acionado o documento suspeito será convertido para um formato de texto puro (TXT) e processado para que seja enviado ao buscador.

No processamento, remove-se o símbolo de % não permitido pelo motor de busca e divide-se o texto do TXT em frases de 30 caracteres. Em seguida, adicionam-se essas frases a uma lista. Por exemplo, para um texto de 3000 caracteres é gerada uma lista com 100 frases.

As frases da lista são colocadas entre aspas e enviadas ao motor de busca BING. Quando um conjunto de palavras é colocado entre aspas, o buscador busca e retorna as ocorrências que possuam exatamente esse conjunto de palavras. Ao término das buscas, o BING retornará até 10 URLs para cada frase buscada. As URLs são armazenadas em uma TreeMap, conjunto de mapeamento que possui um objeto chave e um objeto valor associado, onde as chaves são únicas. Na TreeMap, a URL é o objeto chave e a quantidade de vezes que apareceu durante as análises é o valor associado.

A similaridade é calculada baseando-se na quantidade de ocorrências de uma determinada URL e a quantidade de buscas realizadas. Por exemplo, se uma URL aparece em todas as buscas realizadas pelo BING a similaridade é 100%.

$$\text{Similaridade} = \frac{\text{Qtd de vezes que uma URL apareceu durante as buscas}}{\text{Total de Buscas}}$$

Ao término da análise, as URLs e suas similaridades são salvas no banco de dados e o usuário é encaminhado para a tela (Figura 22) onde pode consultar os resultados obtidos com a análise.



**Figura 22.** Tela com os resultados da análise realizada

O sistema desenvolvido mantém o documento original no servidor até que o usuário solicite a sua exclusão.



## 6. CONCLUSÕES E TRABALHOS FUTUROS

O plágio pode apresentar vários tipos e níveis de complexidade. Por isso, é difícil de identificar. Em alguns casos, uma análise manual torna-se impraticável. Diante disso, surgem diversas ferramentas automatizadas de detecção de plágio, algumas focadas na detecção de plágio em código fonte e outras em documento de texto. Entretanto, selecionar qual ferramenta utilizar tornou-se uma tarefa árdua.

Como forma de facilitar e reduzir o esforço empregado na escolha das ferramentas, este trabalho propôs a elaboração de um quadro comparativo gerado a partir dos testes de eficácia, sensibilidade e desempenho realizado na comparação quantitativa.

Durante as pesquisas, trinta e uma ferramentas foram encontradas. No entanto, apenas cinco passaram pelos critérios de seleção estabelecidos na sessão 4.2, foram: Ferret, Sherlock, CopyCatch Gold, WCopyFind e DOC Cop.

Na análise qualitativa, as ferramentas foram comparadas de acordo com dez critérios considerados importantes. Entretanto, verifica-se que a análise qualitativa não fornece medidas precisas das ferramentas. Dessa forma, visando obter medidas que pudessem apontar a melhor ferramenta, propôs-se uma análise quantitativa onde realizou-se testes de eficácia, sensibilidade e desempenho das ferramentas selecionadas.

No teste de eficácia, foram criadas dez amostras distintas com similaridades conhecidas para verificar se as similaridades apresentadas pelas ferramentas correspondem às similaridades esperadas. Neste teste, a ferramenta WCopyFind se sobressaiu em relação as outras.

No teste de sensibilidade das ferramentas, foram usados dez documentos com diferentes modificações, como: deslocamento de parágrafos e frases, inserção de novos textos e paráfrase de partes do texto, para verificar o quanto as

ferramentas são sensíveis a alterações nos documentos. Neste teste, WCopyFind apresentou os melhores resultados, os piores foram apresentados pela ferramenta Sherlock.

O teste de desempenho geralmente é baseado na complexidade do algoritmo, porém, em algumas ferramentas, essas informações são omitidas pelos proprietários. Dessa forma, o teste de desempenho foi realizado com um cronômetro, o qual não é preciso, mas sugere um tempo de execução aproximado de cada ferramenta.

Para compor o teste desempenho, foram utilizados dez documentos de tamanhos variados. Estes documentos foram analisados e o tempo medido com um cronômetro. Dentre as ferramentas comparadas, a ferramenta Sherlock apresentou os melhores resultados.

Considerando-se os três testes realizados, eficácia, sensibilidade e desempenho, verificou-se que a ferramenta WCopyFind se sobressaiu em detrimento das outras.

Grande parte dos documentos plagiados encontram-se na WEB. Neste contexto, durante a seleção das ferramentas, observou-se que a ferramenta DOC Cop é a única que estende suas análises a documentos presente na WEB. Diante disso, além da comparação das ferramentas, propôs-se neste trabalho de conclusão de curso a modelagem e implementação de um protótipo de detecção de plágio capaz de localizar documentos suspeitos na WEB.

Como trabalhos futuros, propõem-se:

- Aplicação dos testes em ferramentas de detecção de plágio em código fonte;
- Análise estatística dos resultados obtidos nos testes realizados visando melhorar as conclusões;

- Aplicação dos testes realizados neste trabalho no protótipo implementado e comparar os resultados obtidos com as ferramentas comparadas.
- Implantação na UFLA de uma ferramenta de detecção de plágio para texto e código fonte, com busca em repositório local e WEB;
- Aprofundamento do estudo das técnicas de detecção de plágio, a fim de adotar a(s) melhor(es) técnicas no protótipo;
- Desenvolvimento de um roteador de algoritmo capaz de escolher e aplicar a melhor técnica de detecção de plágio, um pré-processamento do texto ou código fonte a ser comparado.
- Modelar uma curva que norteie a relação precisão e tempo de processamento para cada técnica aplicada.

## BIBLIOGRAFIA

ARWIN, C.; TAHAGHOGHI, S. M. M. **Plagiarism Detection across Programming Languages**. Proceedings of the 29th Australasian Computer Science Conference. Hobart: Australian Computer Society, Inc. Janeiro 2006. p. 277-286.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval**. 1. ed. New York: Addison-Wesley, 1998.

BARNBAUM, C. **PLAGIARISM: A Student's Guide to Recognizing It and Avoiding It**. Valdosta State University, 2002. Disponível em: <[http://www.valdosta.edu/~cbarnbau/personal/teaching\\_MISC/plagiarism.htm](http://www.valdosta.edu/~cbarnbau/personal/teaching_MISC/plagiarism.htm)>. Acesso em: 12 Outubro 2010.

BAXTER, I. D. et al. **Clone Detection Using Abstract Syntax Trees Software Maintenance**. Proceedings of the International Conference on. Bethesda, MD , USA : IEEE. Novembro 1998. p. 368 - 377.

BLOOMFIELD, L. The Plagiarism Resource Site. **University of Virginia**, 2010. Disponível em: <<http://plagiarism.phys.virginia.edu/>>. Acesso em: 20 Abril 2011.

CFL SOFTWARE LIMITED. CopyCath Gold. Disponível em: <<http://www.copycatchgold.com>>. Acesso em: 20 Abril 2011.

CHARRAS, C.; LECROQ, T. Handbook of exact String-Matching algorithms. **Université de Rouen, França**, 1997. Disponível em: <<http://www-igm.univ-mlv.fr/~lecroq/string/string.pdf>>. Acesso em: 01 Outubro 2011.

CLOUGH, P. **Plagiarism in natural and programming languages: an overview of**. Department of Computer Science, University of Sheffield, p. 1-31. 2000.

CORNIC, P. **Detection using model-driven software development in eclipse platform**. University of Manchester (dissertação de mestrado). Manchester, p. 101. 2008.

COSMA, G.; JOY, M. **Source-code plagiarism:** A UK academic perspective. Proceedings of the 7th Annual Conference of the HEA Network for Information and Computer Sciences. HEA Network for Information and Computer. 2006. p. 116.

DALY, C.; HORGAN, J. A Technique for Detecting Plagiarism in Computer Code. **The Computer Journal**, v. 48, n. 6, p. 662-666, Novembro 2005.

DOC COP. DOC Cop, 2011. Disponível em: <<http://www.doccop.com/>>. Acesso em: 14 Abril 2011.

DONALDSON, J. L.; LANCASTER, A.; SPOSATO, P. H. A Plagiarism Detection System. **ACM SIGCSE Bulletin - Proceedings of the 12th SIGCSE symposium on Computer science education**, New York, NY, USA , v. 13, n. 1, p. 21-25, Fevereiro 1981.

EVE2. What is EVE2? **Essay Verification Engine**. Disponível em: <<http://www.canexus.com/abouteve.shtml>>. Acesso em: 22 ago. 2011.

FACEBOOK. Viper Plagiarism Scanner. **Facebook**. Disponível em: <<http://www.facebook.com/viper.plagiarism.scanner?ref=ts>>. Acesso em: 03 Maio 2011.

FAIDHI, J. A. W.; ROBINSON., S. K. An empirical approach for detecting program similarity and plagiarism within a university programming environment. **Computers & Education**, Oxford, UK, UK, v. 11, n. 1, p. 11-19, Janeiro 1987.

FRANCO, L. R. H. R.; MILANEZ, J. R. C.; SANTOS, F. A. O. Implantação de um software detector de plágio para análise das questões dissertativas do ambiente virtual de aprendizagem TelEduc. **Revista Brasileira de Aprendizagem Aberta e a Distância**, v. 7, p. 1-8, 2008.

GOOGLE. Disponível em: <<http://www.google.com.br/>>. Acesso em: 03 Abril 2011.

HAGE, J.; RADEMAKER, P.; VUGT, N. V. **A comparison of plagiarism detection tools**. Utrecht University. Utrecht, The Netherlands, p. 28. 2010.

HARTMANN, E. Variações sobre plágio. **Confraria - Arte e Literatura**, 2006.  
Disponível em: <<http://acd.ufrj.br/~confrariadovento/numero8/ensaio03.htm>>. Acesso em: 12 Outubro 2010.

JI, J. H.; WOO, G.; CHO, H.-G. **A source code linearization technique for detecting plagiarized programs**. Annual Joint Conference Integrating Technology into Computer Science Education Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. Dundee, Scotland: ACM. Setembro 2007. p. 73 - 77.

JUNG, C. F. Metodologia Para Pesquisa & Desenvolvimento - Aplicada a Novas Tecnologias, Produtos e Processos. 1ª Edição. Axel Books, v. 1, 2004. p. 131-159.

KANG, N.; GELBUKH, E.; HAN, S. PPChecker: Plagiarism Pattern Checker in Document Copy Detection. In: SOJKA, P.; KOPECEK, I.; PALA, K. **Text, Speech and Dialogue**: Springer Berlin / Heidelberg, v. 4188, 2006. p. 661-667.

KLEIMAN, A. B. **Análise e comparação qualitativa de sistemas de detecção de plágio em tarefas de programação**. Dissertação (mestrado), Universidade Estadual de Campinas. Campinas, SP, p. 94. 2007.

KNIGHT, A.; ALMERTH, K.; BIMBER, B. **Introduction An Automated System for Plagiarism Detection Using the Internet**. Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications. Chesapeake, VA: ACE. 2004. p. 3619-3625.

LANCASTER, T. **Effective and Efficient Plagiarism Detection**. London South Bank University. London, UK, p. 368. 2003.

LIU, C. et al. **GPLAG**: detection of software plagiarism by program dependence graph analysis. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. Philadelphia, PA, USA: ACM. 2006. p. 872--881.

LIU, Y.-T. et al. **Extending web search for online plagiarism detection**. Proceedings of the IEEE International Conference on Information. Las Vegas, USA: IEEE Systems, Man, and Cybernetics Society. 2007. p. 164-169.

LYON, C.; BARRETT, R.; MALCOLM, J. **Experiments in Electronic Plagiarism Detection Computer Science Department, TR 388**. University of Hertfordshire. Hatfield, Hertfordshire, England, p. 15. 2003.

MAURER, H.; KAPPE, F.; ZAKA, B. Plagiarism - A Survey. **Journal of Universal Computer Science**, v. 12, n. 8, p. 1050-1084, Agosto 2006.

MCKEEVER, L. Online plagiarism detection services – Saviour or scourge? **Assessment & Evaluation in Higher Education**, v. 31, n. 2, p. 155-165, 2006.

MUSSINI, J. A. **Novas arquiteturas Para detecção de plágio baseadas em redes P2P**. PUC Paraná. Curitiba PR, p. 107. 2008.

OLIVEIRA, M. G. D.; OLIVEIRA, E. **Uma Metodologia para Detecção Automática de Plágios em Ambientes de Educação a Distância**. Congresso Brasileiro de Ensino Superior a Distância. Gramado: ESUD. 2008. p. 1-20.

OTTENSTEIN., K. J. Further Investigation into a Software Science Relationship. **ACM SIGCSE Bulletin**, New York, NY, USA, v. 8, n. 4, p. 195~198, Dezembro 1976.

PARKER, A.; HAMBLEN, J. O. Computer algorithms for plagiarism detection. **IEEE Transactions on Education**, v. 32, n. 2, p. 94-99, 1989. ISSN 10.1109/13.28038.

PLAGIARISM DETECT. Services. **Plagiarism Detect: The Power of Uniqueness**. Disponível em: <<http://www.plagiarismdetect.com/services.html>>. Acesso em: 22 ago. 2011.

PLAGIARISM.ORG. What is plagiarism? **Plagiarism.org**. Disponível em: <[http://www.plagiarism.org/plag\\_article\\_what\\_is\\_plagiarism.html](http://www.plagiarism.org/plag_article_what_is_plagiarism.html)>. Acesso em: 14 outubro 2010.

PRECHELT, L.; MALPOHL, G.; PHILIPPSEN, M. **JPlag: Finding Plagiarisms among a Set of Programs**. 2000. Facultad fur Informatik, Universitat Karlsruhe. 2000.

PROJECT GUTENBERG. Project Gutenberg, 2011. Disponível em: <<http://www.gutenberg.org/browse/scores/top>>. Acesso em: 26 Maio 2011.

ROBINSON, S. S.; SOFFA., M. L. **An instructional aid for student programs.** In Proceedings of the eleventh SIGCSE technical symposium on Computer science education (SIGCSE '80). New York, NY, USA: ACM. 1980. p. 118-129.

ROY, C. K.; CORDY, J. R. **A Survey on Software Clone Detection Research.** School of Computing Queen's University at Kingston. Ontario, Canada, p. 109. 2007.

SCAN MY ESSAY. **Viper:** The Anti-plagiarism Scanner. Disponível em: <<http://www.scanmyessay.com/>>. Acesso em: 21 ago. 2011.

SCHLEIMER, S.; WILKERSON, D. S.; AIKEN, A. **Winnowing:** Local Algorithms for Document Fingerprinting. SIGMOD 2003. San Diego, CA: ACM. Junho 2003. p. 9-12.

SHERLOCK. The Sherlock Plagiarism Detector. **University of Sydney.** Disponível em: <<http://sydney.edu.au/engineering/it/~scilect/sherlock/>>. Acesso em: 20 Abril 2011.

SMITH, N.; WREN, K. R. Ethical and legal aspects part 2: plagiarism-"what is it and how do I avoid it?" **American Society of PeriAnesthesia Nurses**, v. 25, n. 5, p. 327-330, Outubro 2010.

VERCO, K. L.; WISE, M. J. **Software for detecting suspected plagiarism:** comparing structure and attribute-counting systems. ACM International Conference Proceeding Series. Sydney, Australia: ACM. 1996. p. 81 - 88.

WHALE, G. Identification of program similarity in large populations. **The Computer Journal**, Oxford, UK , 2 Abril 1990. 140-146.

WISE, M. J. **String Similarity via Greedy String Tiling and Running Karp–Rabin Matching.** Department of Computer Science, University of Sydney. Australia, p. 17. 1993.

WISE, M. J. **YAP3:** Improved Detection of Similarities in Computer Program and Other Texts. Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education. 1996. p. 130-134.