Instituto de Matemática e Estatística
Universidade de São Paulo

Trabalho de Formatura Supervisionado: Identificador de Plágio para o Adessowiki

Lucas Ikeda França

Orientador: Prof. Roberto Hirata Jr.

1 de dezembro de 2011

Sumário

Parte I

Parte Técnica

Capítulo 1

Introdução

O plágio é uma questão polêmica que atinge diversos segmentos da sociedade, do artístico ao científico. Com o avanço nas tecnologias de compartilhamento de informação, a prática foi facilitada e sua ocorrência tornou-se mais comum.

É um problema ético, e há um recente exemplo [11] de sua gravidade: o então ministro de defesa da Alemanha, Karl-Theodor zu Guttenberg, renunciou ao cargo após um escândalo causado pela descoberta de plágio em sua tese de doutorado.

No meio acadêmico, a atitude é condenada pelos regimentos internos das instituições de ensino, e há um esforço para identificar e inibir a prática de plágio. Esta preocupação agrava-se em ambientes de educação a distância, onde o controle sobre a maneira que as tarefas são executadas pelos alunos é muito menor do que em situações presenciais. O grande volume de tarefas a ser analisado faz com que a detecção manual nesses casos seja inviável, portanto soluções automatizadas são necessárias.

Embora já existam tais ferramentas disponíveis gratuitamente, não havia até o início deste trabalho qualquer integração delas ao ambiente de ensino Adessowiki, descrito no capítulo a seguir.

O objetivo deste trabalho é fornecer ao Adessowiki uma ferramenta para identificação de plágio em seus conteúdos, tanto código-fonte quanto texto em linguagem natural, a fim de detectar os casos suspeitos: estes serão posteriormente encaminhados para uma análise humana. Espera-se que a existência desse tipo de aparato também desencoraje tentativas de plágio por parte dos alunos, garantindo por consequência um melhor nível de aprendizado.

Nos próximos capítulos serão cobertos os conceitos necessários para a compreensão do funcionamento da ferramenta, bem como um relato do desenvolvimento do trabalho, os resultados obtidos, e as conclusões.

Capítulo 2

Conceitos

2.1 Educação a Distância

É definido em [10] como o processo de aprendizagem onde professores e alunos não estão presentes no mesmo local, ou, inclusive, ao mesmo tempo. A interação entre as partes ocorre por meio de tecnologias de comunicação.

Apesar de não ser uma ideia nova[12], passando ao longo do século XX por meios como correspondência, rádio e televisão, com a popularização da Internet no final do período tornou-se uma forte tendência no meio educacional, pois apesar do distanciamento físico entre as partes envolvidas há um grande ganho no dinamismo da troca de experiências, esclarecimento de dúvidas e obtenção de resultados. Isto ocorre pois o aprendizado não fica restrito ao horário ou local de aula.

É um conceito com potencial para democratizar o acesso à informação e ao conhecimento, uma vez que o acesso à Internet esteja ao alcance de todos[12].

2.2 Plágio

Uma definição[5] comum para plágio é a de apropriação de ideias de terceiros, isto é, um indivíduo se utilizar do trabalho intelectual realizado por outro, sem dar o devido crédito ao autor original. De acordo com [13], substituições sobre um texto que mantenham maior parte, ou a estrutura do original, também caracterizam plágio.

Ainda de acordo com [13], uma simples citação das fontes pode ser suficiente para evitar a maior parte dos casos de plágio.

A prática é antiga, porém sua ocorrência vem crescendo nos últimos anos: em uma pesquisa[2] feita com reitores de faculdades mostrou que 55% deles afirmaram que a incidência de plágio aumentou nos

últimos 10 anos, e 89% destes acreditam que os computadores e a Internet foram os principais fatores que causaram esse crescimento.

Embora sua classificação seja subjetiva, uma vez que algo considerado plágio por alguns pode não ser para outros, é necessário estabelecer um limite de tolerância. Tal limite é um importante parâmetro para as técnicas de identificação de plágio. Porém, a escolha deste limite não é tão bem definida, não havendo uma regra geral que seja aplicável a todo e qualquer cenário.

2.3 Similaridade

Refere-se ao grau de semelhança entre dois documentos. É importante classificar a similaridade em dois níveis: global e local. O primeiro trata de características mais gerais de um documento, como o tema ou tamanho, enquanto o segundo compreende propriedades mais específicas, como sequências de palavras em comum.

Em uma situação de tarefa, onde todos os alunos devem escrever programas a partir da mesma especificação, ou redigir textos sobre um mesmo assunto, é natural que haja similaridade global entre os documentos[5]. Já similaridade local pode ser um indicativo de suspeita de plágio, pois sua ocorrência na situação abordada neste projeto é menos provável, e deve ser analisada cuidadosamente. Este será o tipo de similaridade explorado neste trabalho.

2.4 *Winnowing*

Este é o algoritmo para detecção de plágio utilizado neste projeto. Também é o algoritmo central utilizado pela ferramenta Moss, descrita no próximo capítulo. A seguir será apresentada uma visão geral do algoritmo, conforme explicado em [15].

O termo *winnow* pode ser traduzido do inglês como peneirar, separar ou ainda selecionar[16]. Isto nos fornece a ideia central do algoritmo: separar valores-chave de um texto (*fingerprints*), para então compará-los com valores selecionados de outros textos, em busca de trechos similares comuns. O resultado é um método eficiente para comparação de textos em busca de similaridade local.

Inicialmente o texto original passa por um pré-processamento.

Em seguida são extraídas subsequências do texto pré-processado, denominadas k-gramas. Cada k-grama é uma subsequência de k caracteres seguidos.

Então, para cada k-grama, é calculado um valor de *hash*, a partir de uma função com baixa probabilidade de colisão (isto é, entradas diferentes devem gerar resultados diferentes). Estes valores calculados são denominados *fingerprints* do texto.

Pode-se então comparar os *fingerprints* de dois textos, e em caso de valores iguais, é muito provável que as entradas compartilhem de trechos em comum. Para recuperar estes trechos, os *fingerprints* devem vir acompanhados da respectiva posição do k-grama gerador no texto original.

O algoritmo apresenta as seguintes propriedades, que motivaram sua escolha para este trabalho:

- Insensibilidade a espaços em branco, pontuação, letras maiúsculas ou minúsculas. Qualquer tipo de inserção de espaços ou pontuação, ou a substituição de maiúsculas por minúsculas (e vice-versa) não afeta a comparação dos documentos, pois o pré-processamento normaliza o texto de forma a eliminar essas diferenças.

- Independência da posição de um trecho de texto. A reordenação ou remoção de trechos (suficientemente grandes) do texto original não afetam a correspondência dos *fingerprints*.

- Supressão de ruído. Correspondência de sequências de caracteres curtas não são interessantes, pois não podem ser usadas como indicativo de suspeita de plágio, e devem ser ignoradas.

Além disso, é o algoritmo utilizado pela ferramenta Moss, um dos mais populares identificadores de plágio para código-fonte, o que atesta sua eficácia.

Capítulo 3

Tecnologias Utilizadas

3.1 Adessowiki

É um ambiente de educação a distância online[8] e, como o termo *wiki*[5] tipicamente sugere, colaborativo. Foi desenvolvido em uma parceria entre a Faculdade de Engenharia Elétrica e de Computação da Universidade de Campinas e o Centro de Tecnologia da Informação Renato Archer. Os detalhes a seguir sobre o ambiente foram retirados de [7].

Um de seus maiores diferenciais está na possibilidade do usuário incluir nas páginas não apenas texto em linguagem natural, mas também código-fonte nas linguagens Python e C++, ou seja, não é necessário um interpretador ou compilador instalados em sua máquina: bastam um navegador e acesso à Internet. O código editado em uma página do *wiki* é executado em um *sandbox* no servidor, e os resultados são exibidos no momento da visualização da página. Esta facilidade é especialmente bem-vinda para alunos de cursos de Introdução à Computação, que muitas vezes tem dificuldade em instalar as ferramentas necessárias para programar, e também para alunos iniciando seus estudos em áreas mais específicas, como Visão Computacional e Processamento de Imagens, pois tem, logo de início, muitas das bibliotecas e funções mais utilizadas à sua disposição. Além disso, permite que a programação seja feita a partir de dispositivos móveis ou computadores com baixo poder de processamento, ligados à Internet.

As figuras a seguir mostram a edição e a visualização de um trecho de código em uma página Adessowiki:

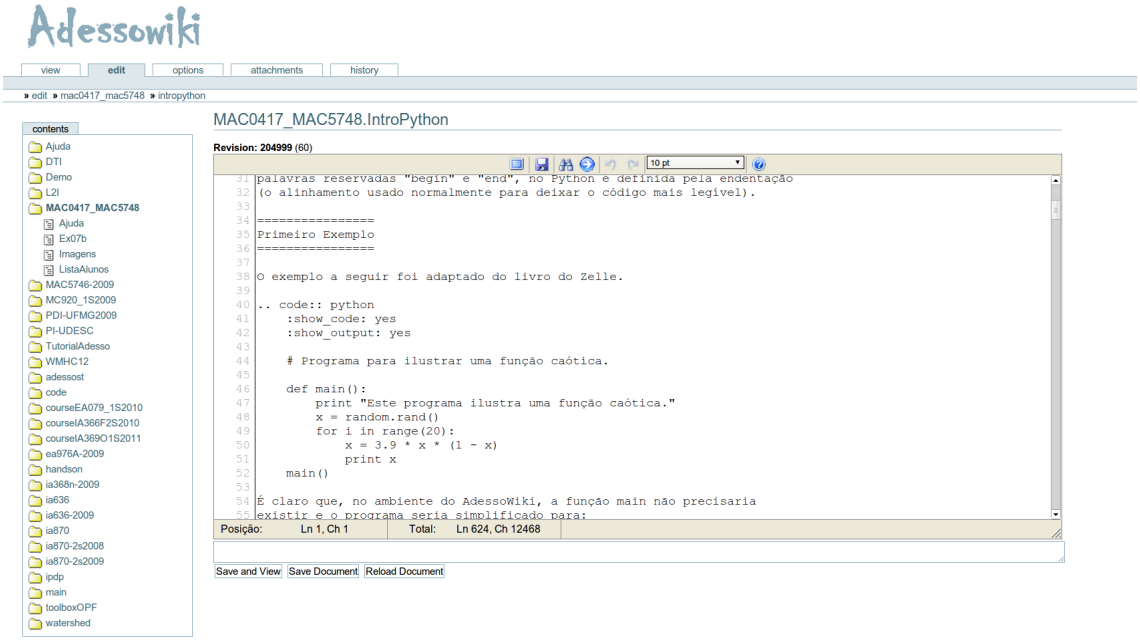

Figura 3.1 – *Edição de código Python por meio do navegador (fonte: [8])*

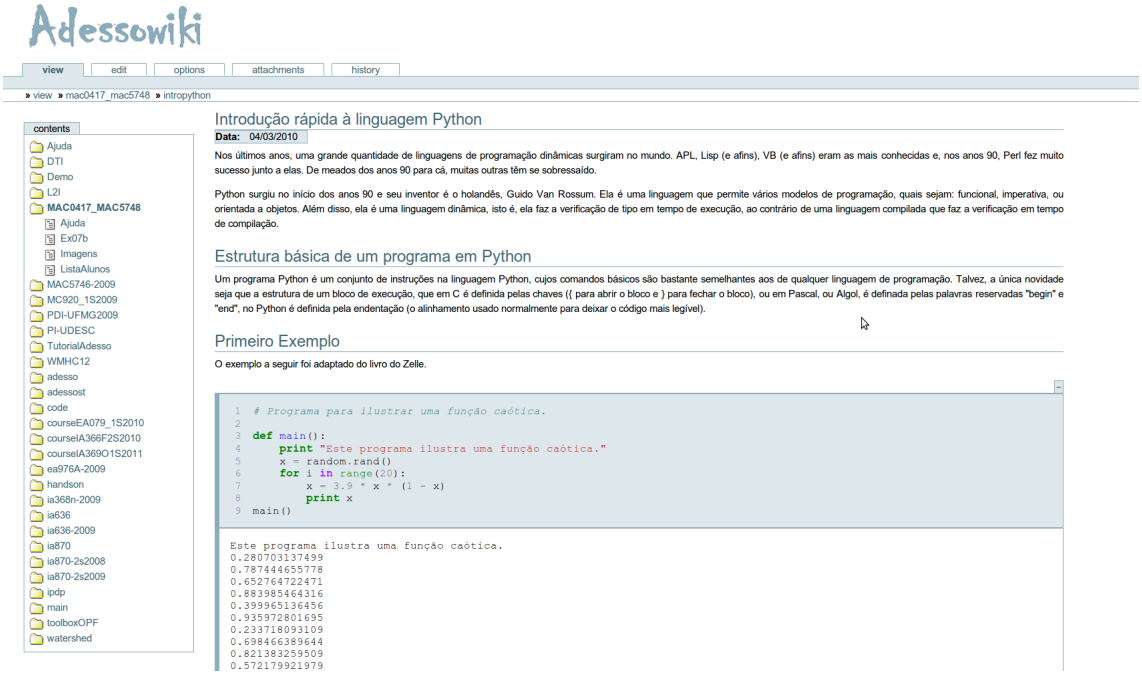

Figura 3.2 – *Visualização do código e sua saída no navegador (fonte: [8])*

Há ainda o aspecto colaborativo, que facilita situações de pesquisa, pois é possível executar diferentes algoritmos sobre determinada entrada e comparar seus resultados diretamente com outras implementações de maneira fácil.

É um sistema escrito na linguagem de programação Python, utilizando o *framework* Django[3]. Sua arquitetura está organizada conforme a imagem a seguir:

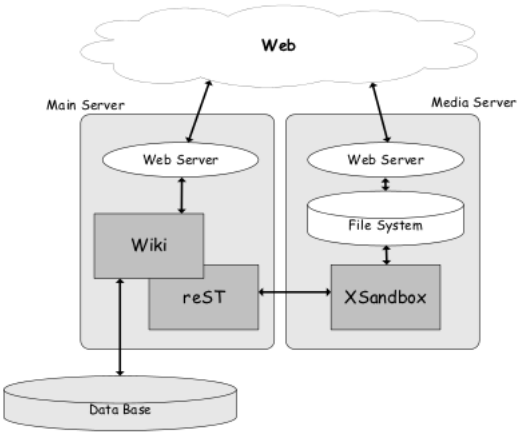

Figura 3.3 *– Organização do sistema nos servidores do Adessowiki (fonte: [7])*

De acordo com [8], o ambiente foi utilizado nos últimos anos principalmente em disciplinas da área de Visão Computacional e Processamento de Imagens, ministradas em renomadas instituições de ensino superior brasileiras, como Universidade de São Paulo, Universidade de Campinas, Universidade Federal de Minas Gerais e Universidade do Estado de Santa Catarina.

3.2 Moss

Moss[1] é uma ferramenta de identificação de plágio gratuita, de código fechado, e muito popular entre professores que ministram cursos de computação ao redor do mundo. Suporta um grande número de linguagens, entre elas Python e C++, porém não é compatível com textos em linguagem natural.

Tem como base a técnica de *winnowing*, descrita no capítulo anterior, porém conta com um pré-processamento adaptado para cada tipo de linguagem de programação suportada: partes não interessantes da entrada, como comentários, são ignoradas; nomes funções e variáveis são trocados pois de outra maneira, substituições simples destes valores causariam a não detecção de trechos suspeitos.

É um serviço online disponível desde 1997, hospedado pela Stanford University, e requer que o usuário faça um cadastro para poder enviar código-fonte para análise, que é feita no servidor. Os resultados são exibidos em uma página gerada para cada rodada de execução, a qual fica disponível para consulta apenas por um determinado tempo, sendo removida após este período.

Foi adotada neste projeto como uma estratégia simplificadora para tratar os trechos de código-fonte nos conteúdos do Adessowiki. O pré-processamento deste tipo de conteúdo, para posterior aplicação do algoritmo do *winnowing*, é difícil, pois é preciso normalizar com nomes de variáveis, nomes de funções, além das diversas particularidades de cada linguagem suportada (por exemplo, a indentação em Python não é desprezível, mas em C++ sim).

3.3 Python

Python[4] é uma popular linguagem de programação de alto nível e código aberto, criada por Guido Van Rossum em 1991. É conhecida por sua simplicidade e flexibilidade, sendo utilizada por programadores de todos níveis de experiência, em uma grande variedade de aplicações.

Algumas de suas vantagens que motivaram sua escolha para este trabalho são: sintaxe clara, portabilidade entre sistemas operacionais, e grande quantidade de bibliotecas disponíveis para diversas finalidades. Há ainda um outro ponto positivo: trata-se da mesma linguagem em que o Adessowiki foi desenvolvido, o que possibilita uma futura integração com o sistema – a ferramenta desenvolvida aqui poderia ser adaptada para ser executada diretamente no servidor do Adessowki.

Uma desvantagem está no fato de ser interpretada, o que a torna mais lenta em determinadas tarefas quando comparada a algumas linguagens compiladas; entretanto, para os requisitos de performance deste trabalho, esta diferença não é grande o suficiente a ponto de justificar seu abandono, dadas as vantagens citadas anteriormente.

Foi utilizada a versão 2.7.1.

Capítulo 4

Atividades realizadas

4.1 Estudo de técnicas de detecção de plágio

Antes de decidir pela técnica de *winnowing*, foram pesquisadas outras propostas para identificação de plágio. As seguintes alternativas foram testadas:

- Métricas[6] – são comparadas características da entrada como quantidade de linhas, laços de repetição e variáveis no caso de código-fonte; e número de parágrafos, e palavras para o caso de linguagem natural. Os dados obtidos para cada entrada são comparados com outros para buscar similaridade.
- Árvore de Sintaxe Abstrata[5] – técnica voltada para detecção em código-fonte; cria-se uma estrutura de dados a partir da entrada, que é comparada com árvores criadas para outras instâncias.
- Modelo de Espaço Vetorial – técnica mais sofisticada, apresentada em [14], consiste na geração de um vetor de características para cada texto; a semelhança entre duas entradas é estimada por meio da comparação dos respectivos vetores, utilizando medidas como seu produto escalar.

Porém, todas as opções acima são particularmente boas para detectar similaridade global, mas não local. Isto faz com que elas não sejam a escolha adequada para o escopo deste trabalho – em um ambiente de educação a distância é esperada similaridade global, conforme explicado anteriormente.

A partir daí, deu-se início a um estudo mais aprofundado do *winnowing*, conforme enunciado em [15], para que a técnica pudesse ser implementada localmente para textos em linguagem natural.

4.2 Desenvolvimento do *parser*

O primeiro passo para o desenvolvimento da ferramenta proposta foi criar um *parser*, que transformasse os conteúdos de páginas do Adessowiki em documentos puros, isto é, contendo apenas um tipo de dado e sem qualquer vestígio de marcadores do Adessowiki. Tais documentos são salvos em um diretório temporário.

A figura a seguir ilustra o funcionamento do parser:

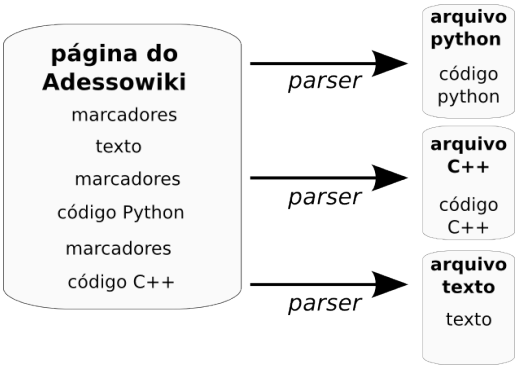

Figura 4.1 *– Aplicação do parser em uma página do Adessowiki*

Uma vez separados os conteúdos, pode-se dar prosseguimento à análise em busca de similaridades entre os documentos, comparando apenas os tipos de conteúdo pertinentes, isto é, texto em linguagem natural não será comparado com código-fonte, e código-fonte em uma determinada linguagem não será comparado com código-fonte em outra linguagem. Desta maneira economiza-se o tempo de comparações que não retornariam qualquer informação útil para o propósito deste trabalho.

4.3 Implementação do *winnowing*

Trata-se da parte mais importante da ferramenta: o conceito da técnica foi apresentado anteriormente, e agora será explicado em detalhes. Vale lembrar que o algoritmo foi implementado localmente para lidar com textos em linguagem natural. O código-fonte das principais funções implementadas encontra-se no Apêndice.

O primeiro passo é o pré-processamento, que visa eliminar características desinteressantes da entrada como espaçamento, pontuação e capitalização. A presença dessas características pode influenciar na detecção de similaridade de baixo nível, pois modificam a correspondência entre dois textos, sem adicionar significado (sob a ótica deste trabalho) aos mesmos.

Simultaneamente, é feito um mapeamento entre as posições do texto original e do pré-processado. Isto é importante para a posterior recuperação dos trechos suspeitos no texto original, pois como a

comparação acontece sobre os textos pré-processados, as correspondências encontradas se referem aos mesmos, e estes são difícil compreensão dadas as operações feitas no pré-processamento.

Finalmente o algoritmo de *winnowing* propriamente dito pode ser aplicado.

Dado um texto pré-processado, são obtidos os seus k-gramas, que são todas as subsequências de k caracteres neste texto. Então, para cada k-grama, calcula-se um valor de *hash*. Para isso foi utilizada a função md5, pois no cenário deste projeto a probabilidade de repetição (colisão) dos valores fornecidos por ela é baixa.

São montadas janelas de comprimento w contendo esses valores de *hash*, que são subsequências de tamanho w, e seleciona-se alguns desses valores da seguinte maneira:

Em uma janela, escolhe-se o menor valor possível. Em caso de empate, escolhe-se o valor mais à direita. Juntamente com o valor selecionado guarda-se a posição em que ele ocorre no texto pré-processado.

Ao passar por todas as janelas, tem-se um conjunto de valores, que são denominados *fingerprints* do documento.

Para comparar dois textos, basta comparar seus *fingerprints*. Em caso de correspondência, levanta-se a suspeita de plágio, pois eles compartilham de uma subsequência suficientemente longa para não ser considerada ruído. O limite deste ruído está relacionado ao parâmetro k. A escolha acertada deste valor é fundamental: se for muito alto, casos que são plágio de fato irão passar desapercebidos (falsos negativos); e se for muito baixo, casos que definitivamente não são plágio serão identificados como suspeitos (falsos positivos). Nenhuma das situações é desejável.

Com os dados posicionais armazenados junto com os *fingerprints*, pode-se recuperar facilmente os trechos correspondentes a cada um deles no texto original: uma simples consulta ao mapeamento das posições feito durante o pré-processamento devolve o intervalo ao qual um *fingerprint* corresponde no texto original.

4.4 Integração com o Moss

4.4.1 Motivos

Embora a técnica do *winnowing* seja aplicável a qualquer tipo de texto, inclusive código-fonte, o pré-processamento exigido por este tipo de conteúdo é complicado, e é específico para cada linguagem de programação, o que impossibilita uma solução genérica.

Ainda que tenham sido estudadas maneiras de realizar este passo de pré-processamento, como o uso de árvores sintáticas, as tentativas foram abandonadas por não apresentar bons resultados, e tomar

tempo de desenvolvimento de outras características importantes da ferramenta.

O serviço do Moss suporta atualmente as linguagens presentes nas páginas do Adessowiki, e utiliza a mesma técnica, a menos do pré-processamento e parâmetros, implementada localmente para detecção de plágio. Estes fatores fazem do Moss o candidato ideal para a integração com a ferramenta desenvolvida neste trabalho.

4.4.2 Detalhes

O envio dos dados para o servidor do Moss é feito através de um *script* Perl, disponível para *download* na página do serviço[1].

Este script é invocado a partir da função principal da ferramenta, e seu resultado, um endereço URL, e finalmente, um *parser* interpreta os resultados disponíveis nesse endereço, a fim de agregá-los ao relatório a ser gerado.

4.5 Geração de relatórios

Os resultados das comparações entre as páginas, com os casos suspeitos de plágio, são exibidos como um relatório em uma página no próprio Adessowiki, cujo endereço é passado como parâmetro para o programa no momento de sua chamada.

As suspeitas são exibidas em uma lista, em que cada elemento corresponde ao seguinte: O par das entradas que levantou suspeita, e o trecho em que foi detectada alta similaridade. No caso de conteúdo em linguagem natural, o trecho é exibido diretamente no relatório; caso o conteúdo seja do tipo código-fonte, é exibido um *link* para a página de resultados do Moss, contendo o trecho suspeito destacado.

Para auxiliar a visualização geral dos resultados é exibido um grafo, onde cada vértice representa um tipo de conteúdo em uma página suspeita do Adessowiki, e cada aresta representa a existência de similaridade de baixo nível entre os vértices que ela conecta. O diâmetro de cada vértice está ligado à quantidade de pares em que ele ocorre. A figura a seguir ilustra um exemplo:

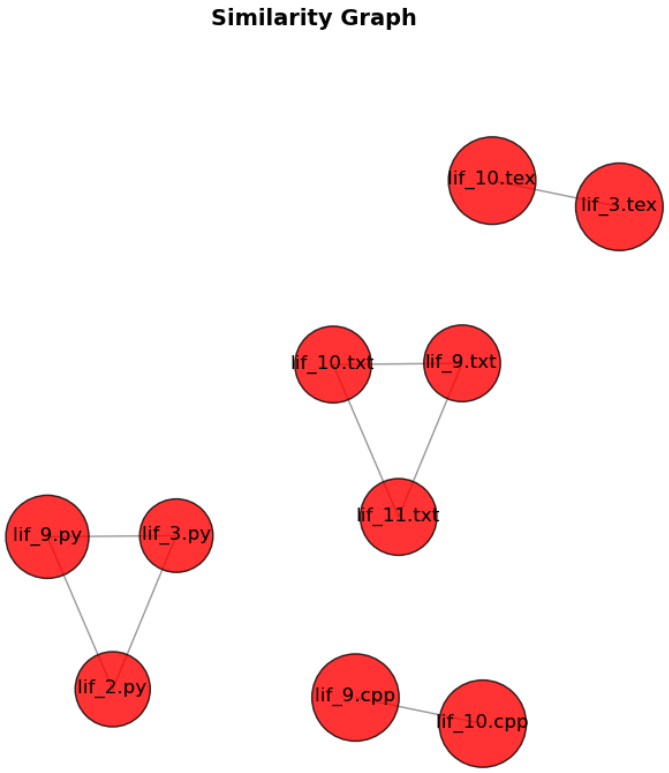

Figura 4.2 – *Exemplo de grafo de similaridade presente em um relatório.*

Capítulo 5

Conclusão

5.1 Considerações finais

Apresentamos uma ferramenta que cumpre o objetivo estabelecido inicialmente: identificar casos de possível plágio no ambiente Adessowiki, tanto para código-fonte quanto para texto em liguagem natural.

Apesar do programa ainda estar em fase de testes, não tendo sido empregado no decorrer de qualquer curso, os resultados foram satisfatórios, e os casos de plágio foram corretamente detectados. A calibragem dos parâmetros que acusam a suspeita de plágio (comprimento dos k-gramas e das janelas) pode ser feita facilmente, caso seja decidido pelo usuário que a ferramenta está devolvendo resultados ruins.

A principal limitação da ferramenta está na sua dependência de um componente externo (Moss) para a análise dos códigos-fonte extraídos. Ainda que raramente, podem ocorrer falhas no servidor do serviço, fazendo o sistema parar de responder, e impossibilitando a busca por suspeitas de plágio em conteúdos do tipo código-fonte.

Apesar da limitação descrita anteriormente, a ferramenta cumpre o objetivo previsto, e espera-se que este projeto possa ser empregado futuramente nos cursos que utilizam o Adessowiki, contribuindo de alguma maneira para uma melhor qualidade de ensino.

5.2 Trabalhos Futuros

Para a continuidade do projeto, seria interessante tratar localmente os conteúdos do tipo código-fonte, conseguindo assim uma independência de ferramentas externas. Também poderiam ser aplicadas outras técnicas de detecção de plágio, em conjunto com o *winnowing*, melhorando os resultados fornecidos pelo programa.

Outra possível melhoria está ligada à apresentação dos relatórios gerados. Conteúdos dinâmicos

poderiam melhorar não apenas a interatividade, mas também a quantidade de informação exibida para o usuário. Porém o suporte a este tipo de conteúdo depende de configurações no servidor do Adessowiki, e no momento estão fora do escopo do projeto.

Referências Bibliográficas

[1] Alex Aiken. Plagiarism detection. http://theory.stanford.edu/~aiken/moss/.

[2] Pew Reseasrch Center. The digital revolution and higher education. http://www.pewinternet.org/Reports/2011/College-presidents/Summary.aspx.

[3] Django Software Foundation. Django : The web framework for perfectionists with deadlines. https://www.djangoproject.com/.

[4] Python Software Foundation. Python programming language. http://www.python.org/.

[5] Wikimedia Foundation. Wikipedia, the free enciclopedia. http://en.wikipedia.org/.

[6] Edward L. Jones. Metrics based plagiarism monitoring. *6th Annual CCSC Northeastern Conference*, 2001.

[7] R. A. Lotufo, R. C. Machado, A. Körbes, and R. G. Ramos. Adessowiki on-line collaborative scientific programming platform. *The 5th International Symposium on Wikis and Open Collaboration*, 2009.

[8] Roberto Lotufo and Rubens Machado. Adessowiki. www.adessowiki.org.

[9] Paul McGuire. Pyparsing. http://pyparsing.wikispaces.com/.

[10] José Manuel Moran. Educação a Distância. http://www.eca.usp.br/prof/moran/dist.htm.

[11] BBC News. German Defence Minister Guttenberg resigns over thesis, 2011. http://www.bbc.co.uk/news/world-europe-12608083.

[12] Ivônio Barros Nunes. Educação a Distância: Definição, Características e Evolução Histórica. *Revista Educação a Distância*, (4):7–25, 1994.

[13] Plagiarism.org. What is plagiarism? http://www.plagiarism.org/plag_article_what_is_plagiarism.html.

[14] Radim Rehurek. Plagiarism Detection through Vector Space Models Applied to a Digital Library.

[15] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local Algorithms for Document Fingerprinting. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[16] Walter Weiszflog. *Michaelis – Moderno Dicionário: Inglês-Português/Português-Inglês*. Editora Melhoramentos.

Parte II

Parte Subjetiva

Capítulo 6

Desafios e Frustrações

O maior desafio foi conciliar o tempo e energia gastos entre o projeto, o trabalho e as duas últimas disciplinas restantes. O planejamento feito no início do primeiro semestre teve que ser adaptado às condições adversas do segundo semestre.

Um outro problema foi a falta de referências no assunto (identificação de plágio). Embora haja inúmeras ferramentas disponíveis para esta finalidade, foi bem difícil encontrar artigos explicando detalhadamente as ideias utilizadas. Uma possível explicação para este fato pode estar na quantidade de ferramentas proprietárias para este fim, em contraste com poucas alternativas de código aberto, além da resistência de parte dos docentes em divulgar os métodos que utilizam para detectar desonestidade acadêmica, com receio de que esta informação possa ser usada erroneamente.

A não implementação de outras técnicas para detecção de similaridade também causou frustração. É claro que o escopo do projeto limita as possibilidades, mas algumas estratégias pareceram bastante interessantes, e seriam aplicações práticas de conceitos abstratos aprendidos com muito custo durante o curso. Além disso, o tempo gasto em partes auxiliares da ferramenta, como os *parsers* foi muito maior do que o esperado.

Capítulo 7

Disciplinas e Conceitos Relevantes

A seguir, uma lista das disciplinas mais importantes para a realização do projeto, e os conceitos relevantes para este trabalho abordados em cada uma delas.

- **MAC0122 – Princípios de Desenvolvimento de Algoritmos:** Forneceu a base necessária para compreender o funcionamento de algoritmos, e em especial o estudo de busca de padrões foi útil para a compreensão e implementação do *winnowing*.
- **MAC242 – Laboratório de Programação II:** Aqui foram aprendidos conceitos de processamento de texto e desenvolvimento de *parsers*, peças fundamentais para este projeto.
- **MAC0414 – Linguagens Formais e Autômatos:** Trata-se de uma disciplina mais teórica, porém foi importante para a compreensão do uso de gramáticas e expressões regulares no *parser* que foi implementado.
- **MAC0417 – Visão e Processamento de Imagens:** Apesar deste trabalho não estar diretamente relacionado com a área, foi nesta disciplina que tive o primeiro contato com a linguagem Python, na qual o projeto foi desenvolvido, e que tornou-se minha linguagem favorita.

Capítulo 8

Agradecimentos

Gostaria de agradecer primeiramente ao professor Roberto Hirata, que sempre foi muito solícito, especialmente neste período de orientação, e por acreditar no projeto quando tudo parecia perdido; e ao professor Carlos Eduardo Ferreira pela preocupação e dedicação em garantir o sucesso dos TCCs. Agradeço ainda aos professores Roberto Lotufo e Rubens Machado pela permissão de uso do Adessowiki neste trabalho, e pelos esclarecimentos e sugestões dadas.

Também deixo minha gratidão aos demais professores e funcionários do IME-USP, pelos ensinamentos dados e serviços prestados durante minha graduação.

Finalmente, não poderia deixar de mencionar os amigos que fiz nestes longos anos de BCC: obrigado pelo apoio!

Apêndice A

Código-fonte do *winnowing*

A seguir, o código em Python das principais funções usadas na técnica de *winnowing*, para detecção de prováveis trechos plagiados entre textos.

A.1 Pré-processamento e mapeamento do texto

```
def cleanAndMapText(self, textFile):
    inputText = open(textFile, 'r').read()
    cleanText = ''
    positionMap = []
    irrelevantChars = [' ', ',', '\n', '.', '\t', '=', '-', ':']
    for i in range(0, len(inputText)):
        if inputText[i] not in irrelevantChars:
            cleanText += inputText[i].lower()
            positionMap.append(i)
    return cleanText, positionMap
```

A.2 *Winnowing*

```
def winnow(self, text, windowSize, k):
    rawHashes = []      #k-grams hashes
    fingerPrints = {}   #saves hash and position
    for i in range(0, len(text) - k + 1):
        rawHashes.append(hashlib.md5(text[i:i+k]).hexdigest())
```

```
    min = 0
    #given a window length w, picks a hash
    for i in range(len(rawHashes) - windowSize + 1):
        window = rawHashes[i:i+windowSize]
        for j in range(0, len(window)):
            if window[j] <= rawHashes[min] or min < i:
                min = i+j
        fingerPrints[rawHashes[min]] = min
    return fingerPrints
```

A.3 Comparação entre dois conjuntos de *fingerprints*

```
def compareFingerPrints(self, fingerPrintsA, fingerPrintsB):
    a = set(fingerPrintsA.keys())
    b = set(fingerPrintsB.keys())
    sharedFingerPrints = list(a.intersection(b))
    return sharedFingerPrints
```

A.4 Busca pela *substring* delimitada por um conjunto de *fingerprints*

```
def findSubstring(self, sharedFingerPrints, fingerPrints, map, text):
    positions = []
    for f in sharedFingerPrints:
        positions.append(fingerPrints[f])
    if positions != []:
        start = map[min(positions)]
        end = map[max(positions)]
        return text[start:end]
    else:
        return ''
```