UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SOCRATEXT: BUSCA APROXIMADA DE CADEIAS EM IMAGENS USANDO MATRIZES DE SUBSTITUIÇÃO

MATEUS DE CASTRO POLASTRO

ORIENTADOR: Prof. Dr. NALVO FRANCO DE ALMEIDA JR.

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
ÁREA DE CONCENTRAÇÃO INFORMÁTICA FORENSE E
SEGURANÇA DA INFORMAÇÃO

PUBLICAÇÃO: PPGENE.DM – 087/2011

BRASÍLIA/DF: DEZEMBRO/2011

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SOCRATEXT: BUSCA APROXIMADA DE CADEIAS EM IMAGENS USANDO MATRIZES DE SUBSTITUIÇÃO

MATEUS DE CASTRO POLASTRO

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE PROFISSIONAL EM INFORMÁTICA FORENSE E SEGURANÇA DA INFORMAÇÃO.

APROVADA POR:

NALVO FRANCO DE ALMEIDA JR., Doutor, FACOM/UFMS (ORIENTADOR)

ANDERSON CLAYTON ALVES NASCIMENTO, Doutor, ENE/FT/UnB (EXAMINADOR INTERNO)

MARIA EMÍLIA MACHADO TELLES WALTER, Doutora, CIC/UnB (EXAMINADORA EXTERNA)

FLÁVIO ELIAS DE DEUS, Doutor, ENE/FT/UnB (SUPLENTE)

DATA: BRASÍLIA/DF, 09 DE DEZEMBRO DE 2011.

FICHA CATALOGRÁFICA

POLASTRO, MATEUS DE CASTRO
SOCRATEXT: BUSCA APROXIMADA DE CADEIAS EM IMAGENS USANDO MATRIZES DE SUBSTITUIÇÃO [Distrito Federal] 2011.
(XIV), (82)p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2011).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Informática Forense 2. Busca aproximada
3. OCR 4. Matriz de Substituição

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

POLASTRO, M. C. (2011). SOCRATEXT: BUSCA APROXIMADA DE CADEIAS EM IMAGENS USANDO MATRIZES DE SUBSTITUIÇÃO. Dissertação de Mestrado, Publicação PPGENE.DM – 087/2011, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, (82)p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Mateus de Castro Polastro
TÍTULO DA DISSERTAÇÃO: SOCRATEXT: Busca Aproximada de Cadeias em Imagens usando Matrizes de Substituição.
GRAU/ANO: Mestre/2011.

Mateus de Castro Polastro
Universidade de Brasília
Campus Universitário Darcy Ribeiro – Asa Norte
CEP 70910-900
Brasília/DF/Brasil

Para minha amorosa esposa Aline e para meus maravilhosos filhos Lucca e Carolina.

AGRADECIMENTOS

A Deus.

Aos meus pais e aos meus irmãos, pela educação e formação como pessoa que me proporcionaram.

Ao meu chefe imediato na Polícia Federal, Everaldo, por entender a importância desta pesquisa e por dar o apoio necessário para que eu concluísse o curso.

Aos meus amigos do Setor Técnico-Científico da Polícia Federal de Mato Grosso do Sul, em especial ao Pedro, Marcio e Marcondes, por me incentivarem e suportarem a alta carga de trabalho durante minhas ausências em virtude deste trabalho.

A todos os colegas do Mestrado em Informática Forense, especialmente ao Nassif e ao Dalben, pela amizade, ajuda, troca de conhecimentos e momentos alegres vividos, que tornaram o mestrado uma etapa inesquecível de minha vida.

Ao professor Fábio Henrique Viduani Martinez, da UFMS, pela disponibilidade e apoio no início da pesquisa.

Ao meu orientador, Nalvo, pelo constante apoio, dedicação, paciência, incentivo e amizade, essenciais para o êxito deste trabalho.

A Polícia Federal e a Universidade de Brasília pelo pioneirismo na criação do Mestrado Profissional em Engenharia Elétrica com Ênfase em Informática Forense e Segurança da Informação.

Ao Hélvio Peixoto, por entender a importância da pesquisa científica para a perícia brasileira, buscando formas efetivas de desenvolvê-la.

Aos meus filhos Lucca e Carolina com quem muitas vezes deixei de brincar para que este trabalho fosse concluído e que me davam estímulo ao me perguntarem “Que dia você vai acabar isso papai?”.

À minha amada esposa Aline, por suportar minhas ausências e momentos de convivência que deixamos de desfrutar juntos.

RESUMO

SOCRATEXT: BUSCA APROXIMADA DE CADEIAS EM IMAGENS USANDO MATRIZES DE SUBSTITUIÇÃO

Autor: MATEUS DE CASTRO POLASTRO
Orientador: Prof. Dr. NALVO FRANCO DE ALMEIDA JR.
Programa de Pós-graduação em Engenharia Elétrica
Brasília, dezembro de 2011

A facilidade de armazenamento e troca de documentos no formato digital, juntamente com a redução dos custos de equipamentos de impressão e digitalização de documentos, vem aumentando a necessidade de utilização na Informática Forense de programas para busca aproximada de palavras-chaves em textos originados por reconhecimento ótico de caracteres (OCR – *Optical Character Recognition*). Porém, as ferramentas que levam em conta os erros de OCR ainda não são totalmente satisfatórias, principalmente devido aos vários tipos de degradação comumente encontradas em documentos digitalizados. Neste trabalho, um algoritmo para busca de palavras-chaves em textos gerados a partir de OCR, baseado em programação dinâmica e na utilização de matrizes de substituição de aminoácidos, é descrito e implementado. Os resultados mostraram que o algoritmo proposto superou, principalmente em textos contendo maior número de erros, o algoritmo de distância de edição de Levenshtein, utilizado nas principais ferramentas para esse fim. Além disso, um estudo detalhado de como os erros de OCR afetam a recuperação da informação, a partir de degradações em imagem, é proposto.

ABSTRACT

SOCRATEXT: APPROXIMATE STRING SEARCH IN IMAGES USING SUBSTITUTION MATRIX

Author: MATEUS DE CASTRO POLASTRO
Advisor: Prof. Dr. NALVO FRANCO DE ALMEIDA JR.
Electrical Engineering Graduate Program
Brasília, December of 2011

The ease of storing and exchanging documents in digital format, along with cost reduction of printing and scanning equipments, has increased the need for use in Computer Forensics of programs to conduct approximate keyword search in texts generated by optical character recognition (OCR). However, tools that take into account the OCR errors are not yet fully satisfactory, mainly due to the various types of degradation commonly found in scanned documents. In this work, an algorithm to search for keywords in text generated by OCR, based on dynamic programming and on the use of amino acid substitution matrices, is described and implemented. The results showed that the proposed algorithm outperformed, especially in texts containing more errors, the Levenshtein edit distance algorithm, the most widely used by the tools for this purpose. In addition, a detailed study of how OCR errors affect the information retrieval, under several image degradations, is proposed.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

1 INTRODUÇÃO

A proliferação do uso de computadores pessoais, o fácil acesso a Internet e o surgimento de novos dispositivos de comunicação têm mudado a maneira com que as pessoas utilizam seu tempo de lazer e a forma que elas fazem negócios [Shinder e Tittel, 2002].

No Brasil, em 2005, 17% dos domicílios na área urbana possuíam computador, enquanto no ano de 2010 esse número saltou para 39% [Barbosa et al., 2010]. Um dos fatores que contribuiu para esse crescimento foi o preço dos discos rígidos, que vem sofrendo uma redução exponencial em seu custo [Webb, 2003]. Aliado a esse crescimento, o mundo vem experimentando aumento da capacidade de armazenamento de dados desses dispositivos de tecnologia da informação há décadas. Em 2005, o engenheiro Mark Kryder estabeleceu o que seria conhecida como *Lei de Kryder*: a capacidade de armazenamento dos discos rígidos dobra a cada ano [Walter, 2003]. Essa taxa supera o aumento da velocidade dos processadores, que de acordo com a *Lei de Moore* duplica a cada 18 meses. Assim, o cenário indica que a quantidade de dados armazenados em equipamentos computacionais vem crescendo além da capacidade de processamentos desses dados.

A utilização de todo esse aparato tecnológico para prática de crimes tem sido cada vez mais comum [Shinder e Tittel, 2002], o que vem demandando dos órgãos de criminalística a capacidade de processar e analisar maior quantidade e volume de evidências digitais. Tais crimes podem ser divididos em duas modalidades: **crimes praticados por meio de computadores** e **crimes contra sistemas informatizados**. Os primeiros, embora possam existir mesmo sem a presença dos computadores, podem e vem sendo praticados cada dia mais utilizando sistemas computadorizados. Crimes comumente não relacionados a sistemas computadorizados como tráfico de drogas, extorsão, estelionato e corrupção podem muitas vezes fazer uso de computadores e provas são frequentemente obtidas através da análise desses dispositivos de armazenamento digital. Já a segunda modalidade de crimes, a qual é praticada contra sistemas informatizados, não existiria sem a presença de computadores. Como exemplos desse tipo de crime podem ser citados o ataque a sítios de Internet, o envio de *spams*, *phishing*, entre outros.

Para lidar com esse cenário de maior demanda de trabalho, diversas formas de reduzir o tempo de análise desses dados vêm sendo empregadas pelos profissionais da área de Informática Forense. Muitas ferramentas foram desenvolvidas e aprimoradas ao longo dos anos com o objetivo de tornar o trabalho mais ágil. Ferramentas como EnCase [Encase, 2011] e

FTK [Forensic Toolkit, 2011] são as mais utilizadas no mundo e permitem, entre diversas outras funcionalidades, recuperação de dados apagados, identificação de tipos de arquivos, buscas utilizando expressões regulares e palavras-chaves e buscas indexadas. Funcionalidades como busca por palavras-chaves são essenciais em quase todos os tipos de exame, pois permite que informações de interesse à investigação sejam encontradas de maneira rápida. O uso dessas buscas, embora seja uma estratégia muito útil, ainda possui limitações. No EnCase, por exemplo, essas buscas não são capazes de encontrar textos em documentos escaneados e armazenados em forma de imagem.

1.1 JUSTIFICATIVA

A digitalização de documentos impressos tem sido uma prática cada vez mais comum, seja pela necessidade de trocar informações a partir da Internet seja para arquivamento de documentos em meio digital [Alves, 2003]. A análise desses documentos digitalizados, bem como de outras imagens que possam ter textos nela representados, sem a utilização de ferramentas que auxiliem o trabalho, é lenta, pois é feita de forma visual. Diante disso, surge a necessidade de criar formas de realizar buscas de textos em imagens para que essa atividade não se torne árdua e demorada. Na versão 3.1 do Access Data Forensic Toolkit (FTK), lançada no primeiro semestre de 2010, foi acrescentada a funcionalidade de realizar busca aos textos contidos nas imagens, através da utilização de reconhecimento ótico de caracteres (OCR). No entanto, essa funcionalidade possui limitações, pois nos casos onde o OCR reconhece de forma incorreta diversos caracteres, o termo buscado pode não ser encontrado mesmo com a utilização de algoritmos de busca aproximada, ou, para encontrar o termo, muitos falsos positivos devem ser retornados, aumentando a massa de dados a analisar.

1.2 OBJETIVOS

O principal objetivo deste trabalho é criar um algoritmo que permita a realização de buscas de palavras-chaves em textos gerados a partir do reconhecimento ótico de caracteres em imagens. A ideia principal por trás da criação do algoritmo é levar em consideração os erros típicos que ocorrem durante o reconhecimento de caracteres em imagens, de forma a permitir que as buscas retornem o maior número de documentos relevantes sem, no entanto, encontrar documentos que não sejam relacionados ao termo buscado. O foco principal na criação do algoritmo é sua aplicação no contexto da Informática Forense.

1.3 ABORDAGEM PROPOSTA

A abordagem proposta neste trabalho aplica conceitos utilizados na área de **Bioinformática**, em particular de algoritmos de comparação de proteínas baseados em **Programação Dinâmica**, fazendo uso das chamadas **Matrizes de Substituição**, que levam em conta a frequência em que pares de símbolos (aminoácidos, no caso de comparação de proteínas) acontecem, quando duas sequências próximas são comparadas.

A analogia aqui feita baseia-se na idéia de que, quando uma palavra-chave é buscada em um texto resultante de um programa de OCR, os erros mais frequentes devam ser pontuados pelo algoritmo de programação dinâmica de forma equivalente a dois aminoácidos diferentes cuja substituição (mutação) seja mais frequente.

1.4 CONTRIBUIÇÕES

Este trabalho resultou em duas contribuições. A primeira consiste em um estudo detalhado de como os erros de OCR afetam a recuperação de informação, mostrando como a busca de palavras-chaves comporta-se sob diferentes níveis de degradações de imagem e em diferentes idiomas. O resultado desse estudo foi publicado em [Polastro e F. Almeida Jr., 2011].

A segunda contribuição consiste no desenvolvimento e implementação de um algoritmo baseado em programação dinâmica e no uso de matrizes de substituição de aminoácidos, muito utilizadas em Bioinformática, para a busca aproximada de palavras-chaves em textos gerados a partir de reconhecimento ótico de caracteres em imagens.

1.5 CONVENÇÕES

Neste trabalho os termos “palavra”, “cadeia de caracteres”, “sequência”, “texto”, “palavra-chave”, “termo de busca” e “padrão” são usados como equivalente à palavra inglesa *string*, que consiste numa sequência ordenada de caracteres. Em geral, “palavra-chave”, “termo de busca” e “padrão” são usados no contexto quando essas *strings* serão buscadas. A sequência de caracteres onde será efetuada a busca é geralmente tratada como “texto”. Uma sequência de caracteres que não esteja nesses contextos específicos está referenciada como “palavra”, “cadeia de caracteres” ou “sequência”. Os termos “casamento aproximado de cadeias” e “casamento de cadeias permitindo erros” são considerados sinônimos neste texto.

1.6 ORGANIZAÇÃO DA DISSERTAÇÃO

Os capítulos seguintes desta dissertação estão organizados como descrito abaixo:

- O Capítulo 2 apresenta alguns aspectos gerais da Informática Forense, mostrando alguns problemas encontrados e as principais ferramentas de apoio. Também são mostradas algumas pesquisas científicas que abordaram buscas de textos no âmbito da Informática Forense;
- No Capítulo 3 é dada uma visão geral do funcionamento do OCR, mostrando os fatores que mais afetam seu funcionamento. Uma breve análise da precisão de ferramentas de OCR é feita e algumas estratégias de busca em textos reconhecidos por OCR são abordadas;
- O Capítulo 4 apresenta conceitos ligados a algoritmos de busca e matrizes de substituição;
- O algoritmo proposto neste trabalho é apresentado no Capítulo 5;
- No Capítulo 6 é apresentado um método para avaliar os efeitos dos erros de OCR na recuperação da informação;
- No Capítulo 7 o método desenvolvido para realizar os experimentos relativos ao algoritmo proposto no Capítulo 5 é detalhado e seus resultados são apresentados;
- Finalmente, no Capítulo 8 conclui-se o trabalho, incluindo vantagens e desvantagens do método proposto, e trabalhos futuros são discutidos.

2 A INFORMÁTICA FORENSE

A Informática (ou Computação) Forense é uma área de pesquisa relativamente nova e tem como principal objetivo materializar as provas existentes em sistemas informatizados, de forma que elas possam ser utilizadas em tribunal de forma convincente [Vacca, 2005]. Para atingir esse objetivo, é necessário seguir procedimentos que garantam a qualidade e preservação da prova. Em uma análise forense de evidências digitais, as etapas necessárias desde o início até o fim de um exame podem ser elencadas da seguinte forma [DFRWS, 2001]:

- Preservação: tem como objetivo garantir que as evidências encontradas sejam preservadas o mais próximo possível de seu estado original e qualquer alteração de dados ocorrida deve ser documentada e justificada. Trata-se de uma etapa de extrema importância, pois, se mal executada, pode inviabilizar o uso da prova obtida ao final do processo no tribunal. Ela permeia todo o processo;

- Identificação: é a primeira etapa do processo. Nela deve-se identificar o que, onde e como a evidência está armazenada, para que se possa determinar qual procedimento deve ser seguido para sua obtenção. É também uma etapa que engloba a triagem do que deve ser analisado;

- Aquisição: consiste na obtenção dos dados que serão utilizados nas etapas seguintes. Esses dados podem ser voláteis, como a memória dos computadores ou dados sobre conexões de redes, como também não-voláteis, como os dados armazenados em discos rígidos ou *pendrives*. Esta etapa inclui *hardware* e *software*, além de políticas e procedimentos necessários para a aquisição dos dados;

- Exame: trata como a evidência deve ser vista. Nesta etapa, ferramentas e procedimentos são utilizados para buscar as evidências. Técnicas como filtragem de arquivos, recuperação de arquivos apagados, quebra de senhas e busca de palavras-chaves são utilizadas;

- Análise: os dados obtidos na etapa anterior são analisados com o objetivo de tirar conclusões sobre o caso sob investigação. Esta etapa e a de exame geralmente são as etapas mais demoradas de todo o processo, e busca analisar as evidências no intuito de responder aos questionamentos dos interessados no caso, bem como levantar informações necessárias para alimentar novas investigações;

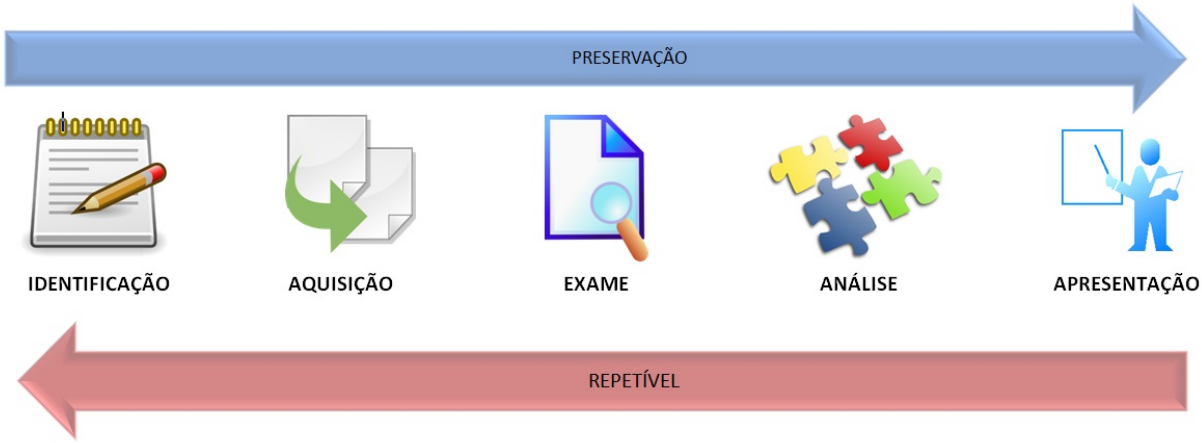


Figura 2.1: Etapas dentro de um exame de Informática Forense

- Apresentação: os dados obtidos nas etapas anteriores são organizados e apresentados de forma que possam ser entendidos pelos envolvidos no processo (advogados, juízes, delegados, etc). O documento final entregue descreve os procedimentos utilizados em todas as etapas, informando as provas encontradas.

Todo esse processo deve seguir metodologia científica e ser capaz de ser reproduzido caso necessário, conforme ilustra a Figura 2.1.

Edmond Locard, um cientista forense francês, formulou o que ficou conhecido como *princípio de Troca de Locard*. Tal princípio estabelece que é impossível um criminoso agir, sem deixar traços de sua presença [Locard, 1934]. Ainda, quando dois objetos ou pessoas entram em contato, há transferência de material entre eles. Esse princípio, embora tenha sido enunciado há muitos anos atrás, quando computadores ainda não existiam, também pode ser aplicado na Informática Forense [Carvey, 2004]. Por exemplo, quando dois computadores se comunicam através de uma rede, informações de cada um deles aparecerão na memória do processo ou em arquivos de *log* do outro. Outro exemplo, quando um *pendrive* é conectado à porta USB de um computador, dados relacionados ao dispositivo conectado bem como a data em que tal evento ocorreu ficarão armazenados no computador. Assim, técnicas apropriadas são necessárias para a correta manipulação e análise de dispositivos de armazenamento computacional.

A busca de vestígios em computadores pode ser um processo lento, seja pela grande quantidade de informações armazenada ou seja pela dificuldade de se encontrar as informações de interesse, muitas vezes por não se saber onde elas são armazenadas. Assim, a utilização de técnicas de busca tem um papel de grande importância na Informática Fo-

rense. Técnicas que podem ser utilizadas para a busca de cadeia de caracteres, incluindo nomes, siglas e sequências numéricas, são utilizadas com sucesso nesse ramo. Se o investigador forense deseja encontrar referências ao nome de uma empresa no disco rígido de um computador apreendido, ele pode conduzir uma busca e, no caso de encontrar ocorrências, deve analisar os locais onde o nome foi encontrado para verificar se é de interesse ao fato investigado. Embora esse tipo de busca funcione em diversos casos, há situações onde o resultado não é exatamente como esperado. No caso onde o nome de uma empresa também seja um nome comum, muitas outras informações não relacionadas à empresa podem ser retornadas, o que aumenta o tempo de análise dos resultados. Em outros casos, o computador suspeito pode ter sido utilizado para trocar informações por *e-mail* a respeito dessa empresa, no entanto, a pessoa que enviava *e-mails* com o nome da empresa digitava o nome dela de forma errada, impossibilitando que técnicas de busca exata encontrem os resultados. Um outro exemplo onde uma busca desse tipo não funcionaria seria um computador que servisse como *backup* de documentos escaneados dessa empresa. Nesse caso, esse tipo de busca também não seria capaz de encontrar o nome da empresa nas imagens dos documentos escaneados, levando o examinador a utilizar outro artifício para encontrar as informações relevantes.

Dessa forma, devido à grande importância da funcionalidade de busca em exames relacionados à Informática Forense, diversas pesquisas têm sido realizadas nessa área, em particular em técnicas para a busca de palavras-chaves em textos.

2.1 TRABALHOS CORRELATOS

Nesta seção, alguns trabalhos relacionados a buscas na área da Informática Forense são abordados.

Uso da Distância de Levenshtein

Tradicionalmente na Informática Forense, um computador que possivelmente contém evidências tem o conteúdo de seu disco rígido copiado *bit* a *bit* para outro disco rígido, gerando a denominada **imagem do disco**. Os dados armazenados nesses discos rígidos podem ser encontrados nas mais variadas formas, incluindo arquivos apagados, arquivos armazenados em áreas marcadas como livres no disco ou em fragmentos de bloco de dados. Para que esses dados possam ser encontrados, buscas de palavras-chaves de interesse podem ser realizadas na imagem do disco. No entanto, devido às características de armazenamento

de dados nos discos rígidos, muitas vezes dados de interesse podem não ser armazenados de forma contígua pelo sistema de arquivos. Além disso, dados apagados podem ser sobrescritos apenas em parte, possibilitando, assim, que trechos de informações possam ser recuperados [Mangnes, 2005].

Desta forma, o trabalho propõe a utilização de busca aproximada utilizando a distância de Levenshtein, descrita no Capítulo 4, de forma que os dados que não seriam encontrados através de uma busca exata, possam ser obtidos por um investigador forense. Para alcançar esse objetivo é proposta a divisão em *buffers* dos dados armazenados e as buscas de palavras-chaves são realizadas nesses *buffers*. Assim, após a busca baseada na distância de edição de Levenshtein em todos os *buffers* de uma palavra-chave, é possível identificar quais deles possuem uma possível ocorrência, descartando os demais. Em seguida é possível realizar uma nova rodada de busca nesses *buffers* "suspeitos", utilizando como parâmetro da busca uma distância de edição menor (mais exata) ou até mesmo uma busca exata, sendo independente da primeira rodada de busca.

A utilização dos *buffers* também pode levar a falsos negativos, como nos casos onde os dados estão armazenados em setores não contíguos do disco. Para evitar esse problema, o autor propõe replicar x bytes do *buffer* anterior no *buffer* atual, onde x é o tamanho da termo pesquisado menos um byte. Após a realização de experimentos o autor mostrou que foi possível diminuir o tempo de busca e a quantidade de dados necessária a analisar, se comparado com a técnica de busca exaustiva.

Aplicação de Distância q-Gram

Para que as buscas de textos em evidências digitais sejam tolerantes a falhas, sendo possível encontrar textos distorcidos ou mesmo parcialmente apagados, é proposta a utilização da distância baseada em *q-gramas* [Petrovic e Bakke, 2008]. *Q-gramas* são *substrings* de tamanho q formados a partir de uma janela deslizante de tamanho q sobre a *string* original. No trabalho são feitas comparações entre a estratégia baseada em *q-gramas* e a distância de edição. De acordo com os autores, a complexidade de tempo da busca baseada em *q-gramas* é linear na soma dos tamanhos das sequências envolvidas, já o tempo da distância de edição é quadrática.

Durante os experimentos, as buscas são feitas sem a utilização de índices e são divididas em duas fases. Na primeira fase, chamada de pré-seleção, para cada fragmento de dados de tamanho N, são calculadas as distâncias em *q-gramas* entre a sequência procurada e o fragmento. As distâncias em *q-gramas* obtidas nessa etapa são então ordenadas de forma

crescente. Na segunda fase, uma busca exaustiva é realizada nos fragmentos, começando com os de maior valor de *q-gramas*.

Para os autores, os resultados obtidos nos experimentos demonstram que a utilização da estratégia de utilizar *q-gramas* no lugar de distância de edição mostrou ser possível manter o mesmo nível de precisão, mas em um menor tempo de execução, para um grande número de combinação de parâmetros, tais como o valor de q e tamanho dos fragmentos.

Agrupamento Temático de Documentos

Dependendo da natureza das evidências digitais e das palavras-chaves utilizadas nas buscas textuais, as ferramentas disponíveis para tal tarefa podem retornar ao investigador forense grande quantidade de dados a serem analisados [Beebe e Clark, 2007]. Nesse trabalho, os autores buscaram formas de elaborar o ranqueamento desses resultados de busca através da utilização de algoritmos de agrupamento. Para isso, é realizado um pós-processamento dos dados retornados pela busca para realizar um agrupamento temático dos documentos, baseado em redes neurais artificiais com *Scalabe Self-Organizing Map (SSOM)*. Os agrupamentos obtidos foram considerados de qualidade pelos autores.

Outras técnicas

Para Steven L. Haenchen [Haenchen, 2010], a identificação e classificação de técnicas de busca de textos no intuito de verificar sua aplicabilidade na busca de evidências digitais é o objetivo do trabalho. As buscas foram divididas em quatro categorias: booleana, *lógica fuzzy*, baseada em contexto e baseada em probabilidades. Algoritmos relacionados a essas quatro técnicas foram implementados e aplicadas a casos reais, no intuito de verificar quais técnicas eram as mais apropriadas no que diz respeito aos documentos identificados bem como ao tempo gasto no processo. Como resultado do experimento, foram identificadas quais técnicas podem encontrar documentos adicionais, mas com tempo de processamento muito maior do que uma busca básica (booleana). Nestes casos, foi indicado utilizar essas buscas apenas em casos onde uma cobertura completa é necessária. Em muitos casos a combinação dos métodos foi a melhor alternativa, aproveitando os pontos positivos de cada um deles. De uma forma geral, não foi identificada nenhuma técnica de busca ótima para todos os casos.

2.2 FERRAMENTAS DISPONÍVEIS

Devido à popularização da Informática Forense e da necessidade de se utilizar ferramentas que auxiliem e automatizem parte do processo de análise das evidências, diversas ferramentas comerciais e de código aberto foram desenvolvidas. Algumas ferramentas são desenvolvidas para funções específicas, como extrair vestígios deixados por programas de mensagem eletrônica [Batista de Sousa, 2008] ou buscar por imagens de nudez de crianças [Polastro e da Silva Eleuterio, 2010], podendo até serem distribuídas no meio acadêmico ou entre as forças da lei. No entanto, devido à diversidade de tipos de evidências existentes, as principais e mais utilizadas funcionalidades são incorporadas a programas que funcionam como *suites* forenses.

Entre as ferramentas forenses comerciais, que englobam uma grande quantidade de funcionalidades, as duas mais utilizadas mundialmente são EnCase [Encase, 2011] e FTK [Forensic Toolkit, 2011]. Ambas possuem interface gráfica e são baseadas na plataforma Windows. Todas as fases envolvidas no processo de informática forense são abrangidas por elas e permitem, entre outras funcionalidades: realizar cópia integral *bit* a *bit* dos dados a serem analisados; processar a evidência, criando índices que podem ser usados posteriormente em buscas indexadas; filtrar arquivos por tipos; recuperar arquivos apagados; e gerar relatórios com as provas levantadas durante a fase de análise.

A seguir, serão apresentadas as principais funcionalidades de busca de palavras-chaves dessas duas ferramentas, com o intuito de demonstrar o que vem sendo empregado pelas ferramentas utilizadas nessa área pelas principais polícias do mundo.

2.2.1 EnCase

O EnCase pode realizar suas buscas tanto nos dados armazenados no nível físico, ou seja, *bit* a *bit*, quanto de forma lógica, onde as buscas podem encontrar palavras-chaves que estão armazenadas em locais não contíguos do disco. As buscas no EnCase podem ser indexadas ou não-indexadas, como será visto no Capítulo 4. Nas buscas indexadas, um processamento prévio é necessário, o que geralmente demanda muitas horas de espera. Entretanto, após a indexação finalizada, as buscas podem ser realizadas de forma rápida. Nas próximas seções serão mostradas as principais técnicas desses dois tipos de busca implementadas na ferramenta.

Indexada

O EnCase na sua versão 6 permite a utilização de diversas técnicas de busca que fazem uso do índice criado com os dados presentes na evidência [Bunting, 2008]:

- *GREP*: acrônimo para *Globally search for the Regular Expression and Print*. É uma ferramenta que permite a construção de expressões regulares. Utiliza a mesma sintaxe do comando GREP existente no Unix e permite grande versatilidade nas buscas;
- *Within five words*: digitando-se "informatica forense" como palavras a serem buscadas, serão retornados todos os resultados onde forem encontradas as palavras "informatica" com até 5 palavras de distância de "forense", não importando se a primeira ocorreu antes da segunda ou vice-versa;
- *Ordered within five words*: funciona da mesma forma que a busca anterior, mas a ordem de ocorrência dos termos é considerada;
- *Stemming*: faz busca baseada no radical da palavra. Por exemplo, ao se buscar pela palavra "crime", as palavras "criminoso" e "criminal" podem ser encontradas;
- *Diacritic match*: para realizar buscas sobre palavras que possuem acentos diacríticos (sinais gráficos que alteram o som letras);
- *Umlaut*: são acentos geralmente encontrados na língua alemã, que alteram o som de vogais. Por exemplo, a palavra "über" é armazenada no índice tanto como "uber" quanto "ueber" e ambos serão retornados por este tipo de busca;
- Busca exata: retorna as palavras que coincidem exatamente com a palavra buscada;
- *Any distance*: trata-se de busca por proximidade. Dadas duas palavras, serão retornados os itens onde as duas palavras ocorram, independentemente da quantidade de caracteres separando essas duas palavras.

Não-indexada

A busca não-indexada no EnCase faz uso do GREP, da mesma forma que ocorre na busca indexada. No entanto, como a indexação ignora alguns caracteres não alfanuméricos na criação do índice, a busca não-indexada se torna útil nos casos de presença de caracteres não alfanuméricos nas palavras-chaves da busca.

2.2.2 FTK

O FTK em sua versão 3.2, da mesma forma que o EnCase, permite o uso de busca indexada ou não-indexada, mas com algumas diferenças de funcionalidades. Nas buscas do FTK, os resultados são agrupados por tipos de documentos. Os agrupamentos podem ser por imagens, por *e-mails*, por arquivos de sistema operacional, e outras categorias existentes no FTK. Além disso, para cada arquivo retornado, é indicado um percentual de relevância. Tanto as buscas indexadas quanto as não-indexadas podem ser realizadas de três formas [Access Data, 2010]: busca de texto, padrões (expressão regular) ou hexadecimal.

Indexada

De acordo com [Access Data, 2010], as formas disponíveis são:

- *Stemming*: da mesma forma que no EnCase, realiza busca baseada no radical da palavra;
- Fonética: busca baseada no som produzido ao se pronunciar a palavra. Palavras com pronúncias semelhantes, mesmo que com escritas diferentes, são encontradas. No entanto, essa funcionalidade não leva em consideração a fonética do português;
- Sinônimos: busca de palavras com significado similar. Esta funcionalidade não está disponível pra língua portuguesa;
- emphFuzzy: busca de palavras que podem possuir até uma quantidade pré-determinada de caracteres diferentes. Essa quantidade pode ser de 0 (busca exata) até 10 caracteres de diferença. Por exemplo, a palavra "drogado" poderia ser encontrada no caso de busca pela palavra "droga", com duas diferenças (eliminar "d" e "o").

Não-indexada

Esta funcionalidade funciona da mesma forma que no EnCase.

Busca com uso de OCR

Trata-se de uma funcionalidade disponibilizada a partir da versão 3.1 do FTK. Nela, os arquivos de imagens podem ser convertidos em arquivos de texto a partir de reconhecimento ótico de caracteres. A partir daí, o texto reconhecido poderá ser indexado e tratado como

qualquer outro texto presente no caso, inclusive aplicando os mesmos tipos de buscas disponíveis para o modo indexado.

3 OCR

O *Reconhecimento Ótico de Caracteres* ou simplesmente OCR, é o processo automático de conversão de caracteres presentes em imagens para o formato de texto. O OCR surgiu no início dos anos 50 e vem sendo tema de muitos estudos e, consequentemente, obtendo avanços com relação ao desempenho e à precisão [Fujisawa, 2007]. Dessa forma, seu uso tem sido bastante difundido, pois pode ser uma solução para o problema de armazenamento de dados, já que um documento armazenado em formato de texto ocupa menos espaço que o mesmo documento em formato de imagem. Além disso, muitas outras vantagens são responsáveis por esse crescimento, por exemplo, a possibilidade de se realizar buscas de palavras-chaves nesses documentos, sendo um importante fator na área de recuperação de informação. O processo de reconhecimento ótico de caracteres pode ser resumido em quatro passos [Melo, 2002]. O primeiro é a **aquisição**, que é o ato de digitalizar o documento impresso, geralmente feito através de um *scanner*. É uma etapa que depende do usuário. O segundo é o **processamento da imagem**, onde serão feitas análises de rotação, segmentação da imagem, filtragem, entre outros, dependendo do programa de OCR utilizado. O terceiro passo é a **identificação de caracteres**, onde a imagem obtida na etapa anterior é analisada, geralmente através de técnicas de inteligência artificial. A última etapa é a **geração do arquivo de saída**. Dependendo do seu formato (*txt, rtf, etc*), podem ser reproduzidas figuras e fontes identificadas nas etapas anteriores.

Já que as etapas descritas anteriormente, desde a aquisição da imagem até o arquivo de saída em formato de texto, formam o primeiro estágio em um processo que pode influenciar etapas posteriores, muitos estudos foram realizados no intuito de analisar as características dos erros ocorridos nesse processo, além de sua precisão. Nas próximas seções esse assunto será abordado, devido a sua influência na etapa de recuperação de informação de textos reconhecidos através de OCR.

3.1 PRECISÃO

A medida de precisão de ferramentas de OCR pode ser feita de diversas formas, como pelo cálculo do número de erros por página, número de erros por palavras, número de erros total, entre outras opções. No entanto, uma das medidas fundamentais é aquela que reflete o esforço que seria necessário para uma pessoa corrigir o texto gerado. Dessa forma, em

geral (se não explícito o contrário), na precisão se mede o número mínimo de operações (inserção, remoção e substituição de caracteres) necessárias para corrigir todo o texto, sendo essa quantidade chamada de *erro*. Portanto, a precisão em número de caracteres pode ser expressa [Rice et al., 1995] pela equação 3.1.

$$\text{Precisão} = \frac{\text{núm. de caracteres} - \text{erros}}{\text{núm. de caracteres}} \tag{3.1}$$

Entre os anos de 1992 e 1996 o *Information Science Research Institute* da Universidade de Nevada, Las Vegas, EUA, conduziu anualmente uma série de testes para averiguar a precisão de ferramentas de OCR comerciais e protótipos de pesquisa, disponíveis à época. Elas foram testadas com variados tipos de documentos escaneados (jornais, revistas, documentos legais, cartas, etc), nos idiomas espanhol e alemão, além do inglês. Os resultados obtidos mostraram que não existe uma ferramenta que possui a maior precisão para todos os casos. Além disso, em muitos experimentos taxas de precisão em torno de 99% foram alcançadas no reconhecimento de caracteres. Embora essa seja uma alta taxa de precisão sob o ponto de vista dos caracteres, se forem consideradas palavras corretamente reconhecidas o resultado pode ser bem inferior. Por exemplo, se uma página contiver 3.000 caracteres, essa taxa de precisão resulta em 30 caracteres incorretamente reconhecidos. E, considerando uma média de 6 caracteres por palavra e que esses erros foram distribuídos um por palavra, essa página terá um total de 30 palavras erradas, acarretando em 94% de precisão no reconhecimento de palavras. Tal quantidade de palavras erradas em uma única página pode comprometer a busca de textos nesses documentos reconhecidos via OCR. Assim, para se obter sucesso na recuperação dos dados obtidos via OCR deve-se diminuir essa taxa de erro ou aprimorar os algoritmos de busca.

Para classificar as causas dos erros nos OCRs, foi criado por [Nagy et al., 1999] uma taxonomia composta por quatro elementos: **defeitos de imagem**, **símbolos similares**, **pontuação** e **tipografia**.

Os **defeitos de imagem** são introduzidos desde o processo de impressão até a imagem ser submetida ao OCR. Problemas como espalhamento da tinta em papel poroso ou manchas ocasionadas pela não absorção em papel revestido são frequentes. O processo de digitalização do documento impresso também pode ocasionar mais defeitos nas imagens, pois o contraste oferecido pelo papel geralmente não é suficientemente grande, ou ainda o documento não foi corretamente alinhado resultando em uma imagem rotacionada.

A presença de **símbolos similares** é outro desafio, pois os OCRs se baseiam basicamente no formato da imagem dos caracteres ao realizar o reconhecimento. A semelhança entre letras e entre letras e numerais pode gerar diversas confusões em sua classificação. A letra "l" ("ele") e o numeral "1" ("um") é um exemplo dessa semelhança.

Outro tipo de caractere que pode gerar detecções incorretas pelo OCR é a **pontuação**. De acordo com [Nagy et al., 1999], cerca de 60% das pontuações em textos narrativos são compostas de vírgulas e pontos finais. Por seus tamanhos reduzidos, a distinção entre eles fica limitada a poucos *pixels*. Além disso, a quantidade de símbolos de pontuação pode variar muito de acordo com a língua em que o documento está escrito, aumentando as possibilidades de confusão.

Conforme [Nagy et al., 1999], **tipografia** é a arte ou habilidade de projetar a comunicação por meio da palavra impressa. Portanto, nessa categoria de erros se enquadram os gerados pelos diversos *layouts*, tipos e tamanhos de fontes (caracteres em itálico, negrito, letras maiúsculas e minúsculas, fontes monoespaçadas, serifadas, bem como combinações desses tipos).

Para conduzir testes de precisão em ferramentas de OCR e realizar a análise dos resultados obtidos, os pesquisadores e empresas do ramo necessitam de um banco de dados de imagens de documentos, bem como os caracteres e suas coordenadas dentro dessas imagens. Esse banco de dados é chamado na literatura como *ground-truth*. Ao longo dos anos, diversos *ground-truths* foram criados [Rice et al., 1996, Kantor e Voorhees, 2000, Phillips et al., 1993, Phillips et al., 1996]. No entanto, a criação de forma não automatizada dessas bases demandam grande esforço. Por isso, uma outra metodologia foi criada para sinteticamente degradar documentos [Kanungo, 1996]. Nessa metodologia, primeiramente usa-se um editor de texto para criar o "documento ideal", no idioma, *layout* e fonte desejados. Em seguida, as imagens desses documentos são criadas e nelas aplicadas as degradações desejadas. O processo sintético tem diversas vantagens com relação ao manual. Primeiramente, o "texto ideal" é conhecido, pois foi inserido pelo próprio usuário. Segundo, com base no mesmo texto é possível gerar diversos *layouts* de saída. Terceiro, a degradação está sob o controle do usuário, podendo ser realizada de acordo com suas necessidades. Metodologias para geração automática de *ground-truths* podem ser encontradas em [Kanungo et al., 1994, Kanungo, 1996, Baird, 1995, Kanungo e Haralick, 1999].

As imagens de documentos escaneados podem apresentar diversos ruídos. O mais comum deles é o chamado de **sal-e-pimenta** (*salt-and-pepper*) [Kaur et al., 2010]. Consiste na presença de *pixels* brancos (sal) em regiões escuras e de *pixels* escuros (pimenta) em

Figura 3.1: Imagem original e imagens corrompidas pelo ruído *sal-e-pimenta* com probabilidade de ocorrência nos *pixels* de 5% (meio) e 10% (direita)

regiões claras, que podem ocorrer devido à variação da superfície do material ou iluminação, ou ainda, no processo de conversão de analógico para digital [Shapiro et al., 2001], como ocorre nos *scanners*. Uma maneira simplificada de modelar esse ruído seria a seguinte [Bovik, 2005]: seja $f(x, y)$ a representação da imagem original e $q(x, y)$ a imagem após sofrer alteração pelo ruído *sal-e-pimenta*, considerando a altura de y *pixels* e largura de x *pixels*. Seja α o percentual de *pixels* alterados na imagem. Então:

$$Pr(q = f) = 1 - \alpha, \tag{3.2}$$

$$Pr(q = MAX) = \alpha/2 \text{ e} \tag{3.3}$$

$$Pr(q = MIN) = \alpha/2, \tag{3.4}$$

onde MAX e MIN são os maiores ou menores valores para a representação de cores para as imagens, respectivamente. Por exemplo, em uma imagem com representação de cores de 8 *bits* $MIN = 0$ (preto) e $MAX = 255$ (branco). De acordo com a equação (3.2), a probabilidade dos *pixels* da imagem permanecerem inalterados é $(1-\alpha)$ e, com uma probabilidade α, os *pixels* serão convertidos em *pixels* brancos ou pretos. Sendo a probabilidade α um valor aleatório, os *pixels* alterados ficarão espalhados pela imagem. A Figura 3.1 mostra uma imagem original e os efeitos da aplicação do ruído sal-e-pimenta sobre 5% e 10% dos seus *pixels*.

A presença desse efeito em documentos escaneados pode afetar diretamente a capacidade de reconhecimento de caracteres pelos OCRs. Um estudo realizado mostrou a influência na precisão de duas ferramentas de OCR quando os documentos escaneados recebiam o ruído *sal-e-pimenta* variando de 5% a 30% dos *pixels* da imagem, tendo a taxa de precisão no reconhecimento de caracteres caído de 71,1% para 19,4% para uma das ferramentas analisadas e de 51,03% para 3,19% quando utilizada a outra ferramenta [Premchaiswadi et al., 2010].

Outro defeito de imagem frequente em documentos escaneados é a **inclinação**. Durante o processo de digitalização, o documento pode ser posicionado com um ângulo diferente de 0 grau sobre o leitor ótico do *scanner*, gerando uma imagem inclinada que pode comprometer

o resultado do OCR. Algumas ferramentas OCR são capazes de detectar e corrigir certos tipos de inclinações sem acarretar em prejuízos ao processo de detecção. Uma análise sobre a influência de diversos ângulos de inclinação sobre a precisão de OCRs foi apresentada em [Mello e Lins, 1999]. Nesse estudo, um dos programas analisados manteve altas taxas de precisão quando os documentos foram rotacionados por até 12 graus. No entanto, em outro programa analisado, a precisão era de mais de 99% para um documento quando rotacionado em 1 grau e caiu para 33% quando rotacionado em 3 graus.

A **resolução da imagem** também é um elemento de grande importância para avaliar a precisão oferecida por uma ferramenta de OCR. Em geral, altas resoluções proporcionam maior precisão. Isso pode ser observado por um estudo comparativo de ferramentas de OCR [Mello e Lins, 1999], onde 6 ferramentas foram avaliadas quanto à precisão variando de 250 *dpi* (*dots per inch*) a 75 *dpi*. Para três dessas ferramentas, a variação na precisão, quanto ao correto reconhecimento de palavras, quando a resolução foi diminuída de 250 para 150 *dpi*, foi pequena (cerca de 2%), mas caiu drasticamente para 100 *dpi* e 75 *dpi*, quando chegou a até 0% de detecção. No entanto, duas ferramentas apresentaram melhores resultados para a resolução de 150 *dpi*, piorando para resoluções superiores. Uma alta resolução implica em maior espaço de armazenamento da imagem e, além disso, nem sempre traz maiores taxas de detecção quando submetidas a um OCR. Tal comportamento também pôde ser verificado em [Rice et al., 1996]. Nos experimentos realizados, verificou-se um substancial aumento dos erros gerados quando a resolução caiu de 300 *dpi* para 200 *dpi*. Por outro lado, o aumento da resolução para 400 *dpi* praticamente não apresentou variação na produção de erros.

3.2 FERRAMENTAS DISPONÍVEIS

Um grande número de ferramentas comerciais está disponível no mercado para realizar reconhecimento ótico de caracteres de documentos impressos. Essas ferramentas incorporam o estado da arte da tecnologia de reconhecimento de caracteres [Lecoq et al., 2001]. Durante muitos anos, vários testes comparativos foram executados para avaliar a qualidade dos textos gerados após o reconhecimento ótico [Rice et al., 1995, Rice et al., 1996, Mello e Lins, 1999]. No entanto, a realização de experimentos comparativos não vem ocorrendo com a força que ocorria antigamente. Uma das ferramentas de OCR para a qual recentemente foram realizados testes de precisão foi o **Tesseract**.

O Tesseract é uma ferramenta de código aberto que foi originalmente desenvolvida pela Hewlett-Packard (HP) entre 1984 e 1994, mas nunca foi explorada comercialmente [Smith, 2007]. Ela foi muito bem qualificada nos testes realizados em 1995 pela Universidade de Nevada, Las Vegas [Rice et al., 1995], mas teve seu desenvolvimento paralisado até que em 2005 a HP tornou público o seu código-fonte. Hoje, a ferramenta pode ser obtida gratuitamente a partir de [Smith, 2011]. Desde então, muitas melhorias e correções de erros foram incorporadas e o teste realizado em [Rice et al., 1995] foi reaplicado em 2007 utilizando a então recém-lançada versão 2.00 do Tesseract [Smith, 2007]. Os resultados mostraram uma melhoria geral de 7,31% com relação aos valores obtidos em 1995. No ano de 2010 foi lançada a versão 3.00 do Tesseract, incluindo atualizações como a inclusão do suporte à língua portuguesa e melhorias na análise de *layouts* das páginas. O Tesseract oferece suporte a diversos sistemas operacionais como Linux, Windows e Mac, é licenciado de acordo com os termos de licença Apache 2.0 e tem recebido da comunidade de código aberto e da Google bastante atenção em seu aprimoramento [Smith, 2011]. Outras informações a respeito do modo do funcionamento do Tesseract podem ser obtidas em [Smith, 2007] e [Smith, 2011]. Também há diversas ferramentas comerciais disponíveis, entre elas: OmniPage, Abby FineReader e OcrIRIS.

3.3 BUSCAS EM TEXTO RECONHECIDO ATRAVÉS DE OCR

Os textos gerados através de reconhecimento ótico de caracteres podem conter erros. A utilização de dicionários é uma das estratégias utilizadas para correção desses erros. No entanto, ela não se mostra uma boa alternativa já que geralmente as coleções de documentos podem possuir um enorme número de palavras que não podem ser encontradas no dicionário, tais como palavras estrangeiras e termos técnicos [Hawking, 1996]. Além disso, os OCRs mais frequentemente reconhecem de forma errada palavras pouco comuns (mas mais seletivas na busca), como nomes próprios [Kantor e Voorhees, 2000].

Os erros de OCR são dependentes da qualidade da imagem, da ferramenta de OCR, da fonte utilizada no documento, do *layout*, entre outros fatores. Apesar de haver muitas fontes de erros, é possível agrupá-los da seguinte maneira [Mello e Lins, 1999]:

- Substituição de um caractere por outro (“c” por “e”);
- Substituição de um caractere por mais de um (“m” por “rn”);
- Susbtituição de mais de um caractere por somente um (“iii” por “m”);

- Supressão de caractere;
- Supressão de espaço em branco, juntando duas palavras ("busca aproximada" por "buscaaproximada");
- Junção de duas palavras ocorrendo perda de caractere ("busca aproximada" por "buscaproximada");
- Supressão de linhas inteiras de texto;
- Inserção de caracteres;
- Ruídos.

Um estudo [Taghva et al., 1994] avaliou os efeitos dos erros de OCR na recuperação da informação. Foram realizadas buscas em textos considerados sem erros de OCR (manualmente corrigidos) e textos reconhecidos via OCR. Dos 15 documentos não retornados para o conjunto de textos obtido pelo OCR, 6 foram especificamente devidos a erros do OCR, comprovando sua influência na recuperação de informação. Para lidar com buscas em textos que podem conter erros, técnicas de busca aproximada podem ser aplicadas. Um estudo sobre essas técnicas pode ser encontrado em [Navarro, 2001]. Entretanto, quando a aplicação é específica para buscas em texto gerado por OCR, há outras ideias a explorar, pois pode-se utilizar o padrão de erros que esse tipo de documento pode conter.

Devido à importância na recuperação de informação desse tipo de texto, em 1996 foi realizada a quinta *Text REtrieval Conference* (TREC-5). Uma das trilhas da conferência (*Confusion Track*) buscava investigar como o desempenho na recuperação de informação pode ser afetado por textos reconhecidos por OCR [Voorhees e Harman, 1996]. Três versões de documentos foram disponibilizadas, sendo uma a versão original (livre de erro) e as outras duas o texto OCR dos originais com uma taxa de 5% e 20% de erro. O objetivo dessa trilha era avaliar o efeito de diferentes níveis de erro de OCR na recuperação da informação. Essa trilha foi aberta para que universidades e empresas pudessem submeter suas soluções. Essas soluções deveriam retornar os primeiros 1000 documentos que estivessem relacionados com a busca. Cinco grupos submeteram soluções [Voorhees e Harman, 1996].

O método *Rutgers SCILS APLab* baseou-se no número de resultados aproximados por palavras retornadas em cada documento. Um resultado aproximado foi definido aplicando a expansão do termo de busca baseado no que foi denominado *"double-dot-5-grams"*. Um *"double-dot-5-grams"* é um *5-grams* onde uma posição pode casar com 0, 1 ou 2 caracteres.

A solução proposta por *Australian National University* buscou expandir de forma automática e manual os termos de busca, baseado em erros típicos de OCR. Em seguida, todas essas derivações foram buscadas nos textos para produzir o resultado final. O grupo *George Mason University* enviou dois conjuntos de soluções. Ambos utilizaram *4-grams*, onde *stop-words* e os 150 mais comuns *4-grams* foram removidos. Os termos de busca foram, então, expandidos por um método baseado em buscas previamente realizadas no grupo de documentos, resultando em novos *4-grams*. Esses três métodos basearam-se, portanto, na expansão dos termos de busca [Voorhees e Harman, 1996].

Os outros dois métodos restantes preferiram tratar o texto onde a busca seria realizada [Voorhees e Harman, 1996]. O método de *CLARITECH Corporation* aplicou métodos estocásticos nos documentos para corrigir palavras numa análise feita frase a frase. A correção das palavras foi baseada na modelagem estatística do bigrama (*2-grams*) delas. O trabalho sumbetido pelo *Swiss Federal Institute of Technology* também se baseou em modelagem estatística e utilizou *4-grams* para o texto de 5% de degradação e *3-grams* para o de 20% de degradação.

No estudo conduzido por [Collins-Thompson et al., 2001], outras técnicas foram utilizadas para fazer buscas de palavras-chaves dentro de documentos. Primeiramente, é utilizado um algoritmo rápido de busca aproximada, com o intuito de filtrar os textos mais prováveis de casarem com o termo de busca. Para isso, é utilizado um valor $k = 5/8$ (ou outro valor derivado, proporcional a esse), que significa que serão aceitos resultados que contenham 5 erros em cada 8 caracteres. Em seguida, é modelado um conjunto confusão para mapear a probabilidade de erros típicos de OCR ocorrerem com seus respectivos custos de edição, utilizando o Teorema de Bayes. O modelo é então treinado por um conjunto de documentos não utilizados durante a avaliação do método. Há uma etapa opcional do algoritmo que trabalha com outras abordagens como heurísticas de palavras e modelos baseados na linguagem utilizada. Para realizar a busca de palavras-chaves propriamente dita, é utilizada busca aproximada com o custo de edição baseado nos valores calculados anteriormente. Os candidatos que apresentarem custo de edição menor ou igual a um limiar pré-definido são considerados resultados da busca.

A ideia de análise dos erros de OCR para validação de alguns modelos de degradação de erros de documentos foi explorada em [Li et al., 1996]. Apesar de não ter focado nas buscas aproximadas, uma das ideias do trabalho, chamada de *"multi-substitution"* (ou multi-subsituição), é uma extensão do modelo de distância de edição e poderia ser aplicada a buscas de palavras-chaves em imagens de documentos. A *multi-substituição* trata de re-

presentar os erros de segmentação que tipicamente ocorrem em OCRs, como reconhecer o padrão “m” como “rn”. Para os autores, esse erro $(1:2)$ representa um único erro e não uma inserção seguida de uma subtituição (ou vice-versa). Para mapear isso, estende o algoritmo tradicional de cálculo de distância de edição para permitir *multi-substituições* de g caracteres por h caracteres, representada por $c_{sub_{g:h}}$, cuja relação de recorrência considerando uma multi-substituição máxima de 4 caracteres é dado pela equação 3.5:

$$d_{i,j} = \min_{0 \leq g,h \leq 4} [d_{i-j,j-h} + sub_{g:h}(s_i \dots s_{i+g-1}, t_j \dots t_{j+h-1})]. \tag{3.5}$$

4 ALGORITMOS DE BUSCA E MATRIZES DE SUBSTITUIÇÃO

Neste capítulo são apresentados aspectos relacionados a busca de cadeias de caracteres, mostrando alguns conceitos relacionados a buscas exatas e aproximadas, abordando também a utilização ou não de índices. É apresentada, ainda, uma breve introdução a respeito da comparação de proteínas e da utilização de matrizes de substituição nesse processo. Antes, serão descritas aqui algumas notações utilizadas.

- Seja Σ o alfabeto de tamanho $|\Sigma| = \sigma$.
- Então Σ^* é o conjunto de todas as possíveis palavras sobre o alfabeto Σ.
- Seja $s = s_1 s_2 \ldots s_m$ um padrão de tamanho $|s| = m$ e
- s_i o i-ésimo caractere de s, onde $1 \leq i \leq m$.
- Seja $t = t_1 t_2 \ldots t_n$ um texto de tamanho $|t| = n$ e
- t_i o i-ésimo caractere de t, onde $1 \leq i \leq n$.
- A cadeia vazia é denotada por ϵ. Assim, $|\epsilon| = 0$.

4.1 BUSCA EXATA

O casamento de cadeias de caracteres ou simplesmente busca de texto é o problema mais estudado no que diz respeito a algoritmos relacionados a palavras e muitos algoritmos foram desenvolvidos para resolver o problema de forma eficiente [Crochemore e Rytter, 2003]. Algoritmos que realizam busca exata estão disponíveis na maioria dos editores de texto, navegadores de Internet, entre diversos outros programas utilizados no dia-a-dia.

A busca exata consiste em encontrar todas as ocorrências de um padrão $s = s_1 s_2 \ldots s_m$ em um texto $t = t_1 t_2 \ldots t_n$, onde s e t são sequências de caracteres de um conjunto finito de caracteres Σ [Navarro e Raffinot, 2002]. Existem muitos algoritmos para resolver esse problema, sendo os dois mais famosos *Knuth-Morris-Pratt* e *Boyer-Moore*, surgidos no ano de 1977. Desde essa época, os estudos em algoritmos foram realizados buscando as mais diversas melhorias: algoritmos mais simples, menor complexidade no pior caso,

possibilidade de se buscar por padrões estendidos como curingas e expressões regulares, algoritmos constantes no uso de espaço, entre outros [Navarro e Raffinot, 2002].

Os algoritmos de *Knuth-Morris-Pratt* e de *Boyer-Moore* consistem em duas fases, sendo a primeira o pré-processamento do padrão e a segunda a fase efetivamente de busca. A seguir será apresentado um algoritmo mais básico e intuitivo do que esses dois, conhecido como algoritmo *força bruta*, mas que serviu de inspiração para suas criações [Crochemore e Rytter, 2003].

No algoritmo *força bruta* nenhum pré-processamento do texto é necessário. Ele, assim como diversos outros, utiliza uma *janela de busca* do mesmo tamanho do padrão s buscado, a qual é movimentada da esquerda para a direita em exatamente uma posição no decorrer da busca, partindo da posição 1 até a posição $n - m + 1$ do texto t. A cada tentativa, a *janela de busca* posicionada em t é comparada com s. Caso haja diferença, para-se a comparação e a janela é deslocada um caractere à frente. A complexidade desse algoritmo é $O(m \times n)$. Embora seja simples, não é o algoritmo usado na maioria das funções de busca encontradas nos editores de texto, onde geralmente está implementada uma versão simplificada do algoritmo *Boyer-Moore* [Charras e Lecroq, 2004].

4.2 BUSCA APROXIMADA

A busca aproximada consiste em encontrar todas as ocorrências de um padrão em um texto permitindo um número limitado de diferenças entre eles. As primeiras referências a esse problema apareceram entre os anos 60 e 70 e tinham como motivação a solução de problemas relacionados a biologia computacional, processamento de sinais e buscas de textos, as quais ainda têm sido as áreas de maior aplicação até hoje [Navarro, 2001]. A seguir serão apresentados alguns conceitos necessários para o entendimento formal do problema.

4.2.1 Conceitos Básicos

- Seja $k \in \mathbb{N}$ o número máximo de erros permitido ou similaridade mínima entre as sequências;
- Seja M a matriz de programação dinâmica $m \times n$;
- Seja $f : \Sigma^* \times \Sigma^* \rightarrow \mathbb{N}$ a função de distância ou similaridade;

- O problema consiste em, dados s, t, k e $f()$, retornar todas posições j no texto t tal que $M_{m,j} \geq k$ para similaridade ou $M_{m,j} \leq k$ para distância de edição.

Similaridade e distância são duas abordagens utilizadas na comparação de sequências. Na similaridade, o alinhamento entre s e t mede o grau de semelhança entre elas. Já a distância de edição é a medida do grau de diferença entre as sequências, o que implica em saber o número mínimo de operações necessárias (inserção, remoção e substituição) para transformar uma sequência na outra. Em muitos casos essas medidas (similaridade e distância de edição) podem ser convertidas entre si, com a utilização de uma fórmula simples [Setubal e Meidanis, 1997].

A distância de edição foi proposta pelo russo Vladimir Levenshtein [Levenshtein, 1965] e por isso também recebeu o nome de **distância de Levenshtein**. A busca aproximada baseada na distância de edição é uma variação desse problema, onde não se compara as duas sequências como um todo, mas busca-se na sequência t trechos que casam, com uma distância de edição $\leq k$, com a sequência s [Sellers, 1980]. Portanto, trata-se de um problema de minimização.

Needleman e Wunsch [Needleman e Wunsch, 1970] adaptaram o algoritmo da distância de edição para aplicação sobre sequências biológicas, propondo uma medida de similaridade entre elas. Nessa abordagem passou-se a ter um problema de maximização e o algoritmo ficou conhecido como um algoritmo de **alinhamento global**.

Um **alinhamento** é definido como a inserção de espaços em posições arbitrárias ao longo das sequências de forma que elas fiquem com o mesmo tamanho e mantenham a ordem de seus símbolos [Setubal e Meidanis, 1997]. O alinhamento é um instrumento muito utilizado quando se deseja identificar regiões similares de duas sequências.

Anos depois, Smith e Waterman [Waterman, 1981] realizaram adaptações no algoritmo de Needleman e Wunsch com o objetivo de encontrar regiões de grande similaridade entre duas sequências, o que é conhecido na literatura como **alinhamento local**.

No contexto deste trabalho, o que se pretende é encontrar todas as ocorrências de um padrão s em um texto t (onde geralmente $m << n$), através de um alinhamento conhecido como **semi-global**. Esse alinhamento tem esse nome porque do ponto de vista do texto o alinhamento é local, enquanto que do ponto de vista do padrão s, o alinhamento é global.

Esse problema pode ser resolvido de diversas formas, as quais podem agrupadas em 4 categorias: programação dinâmica, autômatos, filtros e bit-paralelismo [Navarro, 2001]. Nas próximas seções serão apresentados os principais conceitos envolvendo busca aproximada

através da técnica de programação dinâmica, pois foi a técnica escolhida na abordagem proposta neste trabalho.

4.2.2 Programação Dinâmica

A programação dinâmica é tipicamente aplicada quando se deseja obter uma solução ótima para um problema que pode ser dividido em sub-problemas similares. Na medida em que os sub-problemas são resolvidos, os dados são acumulados em uma tabela de forma que não seja necessário recalculá-los [Cormen et al., 2001]. A solução encontrada pode, muitas vezes, não ser a única solução ótima, pois podem existir outras soluções que possuem um mesmo valor ótimo para o problema. A programação dinâmica foi o primeiro algoritmo utilizado para resolver o problema da busca aproximada. Embora não seja o algoritmo mais eficiente, é um dos mais flexíveis para adaptar a diferentes funções de distância [Navarro, 2001] (ou similaridade).

Antes da utilização de um algoritmo de programação dinâmica para realizar buscas aproximadas, é necessário decidir se será utilizado o conceito de *distância de edição* ou de *similaridade*. Na *distância de edição de Levenshtein* é atribuído o custo 1 para cada operação que deve ser realizada para transformar s em $t_i \ldots t_j$. Por exemplo, se deseja-se buscar todas as ocorrências da palavra "crime" no texto "há um criminoso envolvido" com no máximo uma diferença, será possível encontrar a palavra "crimi", pois uma única substituição de letra é necessária (substituição da letra i por e transforma "crimi" em "crime").

Se for utilizado o conceito de *similaridade*, passa-se de um problema de minimização para um problema de maximização, pois terão que ser encontradas posições no texto t que possuam similaridade maior ou igual a um limiar k definido. Também é necessário definir qual será a função de similaridade, ou seja, quais serão os valores retornados quando os caracteres comparados ($s_i \times t_j$) forem iguais ou diferentes. Para fins de ilustração, será atribuído o valor -1 quando $s_i \neq t_j$ e $+1$ quando $s_i = t_j$.

Outro fator importante na utilização da matriz de programação dinâmica para realização de busca aproximada é a inicialização dos valores da matriz. A matriz de programação dinâmica M utilizada na busca aproximada do padrão s no texto t será a matriz $M_{0..m,0..n}$. A posição $M_{i,j}$ representa a similaridade entre a sequência $s_{1..i}$ e a sequência $t_{j'..j}$, onde $1 \leq j' < j$. A relação de recorrência para o preenchimento da matriz de programação dinâmica é a seguinte:

$$M_{i,0} = i \times (-1) \tag{4.1}$$

$$M_{0,j} = 0 \tag{4.2}$$

$$M_{i,j} = \begin{cases} M_{i-1,j-1} + 1 & \text{se } s_i = t_j \\ \max\left(M_{i-1,j}\,, M_{i,j-1}\,, M_{i-1,j-1}\right) - 1 & \text{se } s_i \neq t_j \end{cases} \tag{4.3}$$

Na equação (4.1), $M_{i,0}$ representa a similaridade entre o padrão $s_{1..i}$ e uma cadeia vazia. Como o padrão s pode ser casado em qualquer posição de t, toda a primeira linha é inicializada com o valor 0, conforme (4.2).

A Figura 4.1 exemplifica a aplicação das equações (4.1), (4.2) e (4.3) na busca do padrão "aproximar" no texto "busca aproximada". À medida que são feitos os cálculos para a célula $M_{i,j}$, é armazenada em uma matriz auxiliar a escolha feita *(ponteiro de retorno)*, de forma que ao final do processo seja possível obter os alinhamentos realizados, conforme ilustra Figura 4.2.

Os ponteiros de retorno podem ser $\uparrow, \nwarrow$ e $\leftarrow$, e possuem os seguintes significados:

- $\uparrow$: alinhamento de s_i com espaço.
- $\nwarrow$: casamento do caractere s_i com t_j, se $s_i = t_j$, ou substituição, caso contrário.
- $\leftarrow$: alinhamento de espaço com t_j.

Portanto, para o exemplo ilustrado nas Figuras 4.1 e 4.2 tem-se dois alinhamentos realizados (o resultado da busca aproximada), considerado o limiar de similaridade $k = 7$.

A relação de recorrência indicada nas equações (4.1), (4.2) e (4.3) pode ser resolvida em $O(nm)$ através da matriz de programação dinâmica $M_{m+1,n+1}$ criada.

Um caso especial de similaridade entre cadeias de caracteres é a chamada LCS (*Longest Common Subsequence*), onde apenas as operações de inserção e remoção de caracteres são aceitas. Para se definir a LCS, primeiramente deve ser definido o significado de *subsequência*. Em uma cadeia S, uma *subsequência* é definida como um subconjunto dos caracteres de S dispostos relativamente em sua ordem original, ou seja, uma *subsequência* S de comprimento n é especificada como uma lista de índices $i_1 < i_2 < i_3 < \cdots < i_k$, para algum $k \leq n$ [Gusfield, 1997]. Uma subsequência se diferencia de uma subcadeia porque na subcadeia os caracteres devem ser contíguos. Assim, dadas duas cadeias de caracteres S_1 e S_2, uma subsequência em comum é aquela que aparece em ambas as cadeias. O problema da maior subsequência em comum é encontrar a maior subsequência em comum entre S_1 e S_2.

	t	b	u	s	c	a		a	p	r	o	x	i	m	a	d	a
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	-1	-1	-1	-1	-1	1	0	1	0	-1	-2	-3	-4	-5	-4	-5	-4
p	-2	-2	-2	-2	-2	-1	-1	0	2	1	0	-1	-2	-3	-4	-5	-6
r	-3	-3	-3	-3	-3	-2	-2	-1	1	3	2	1	0	-1	-2	-3	-4
o	-4	-4	-4	-4	-4	-3	-3	-2	0	2	4	3	2	1	0	-1	-2
x	-5	-5	-5	-5	-5	-4	-4	-3	-1	1	3	5	4	3	2	1	0
i	-6	-6	-6	-6	-6	-5	-5	-4	-2	0	2	4	6	5	4	3	2
m	-7	-7	-7	-7	-7	-6	-6	-5	-3	-1	1	3	5	7	6	5	4
a	-8	-8	-8	-8	-8	-7	-7	-4	-4	-2	0	2	4	6	8	4	6
r	-9	-9	-9	-9	-9	-8	-7	-5	-5	-1	-1	1	3	5	**7**	**7**	6

Figura 4.1: Algoritmo de programação dinâmica para buscar as ocorrências do padrão "aproximar" no texto "busca aproximada" utilizando medida de similaridade $+1$ para caracteres iguais e -1 para caracteres diferentes. As células em negrito indicam as posições do texto onde foram encontrados resultados para um valor pré-definido de $k = 7$.

	t	b	u	s	c	a		a	p	r	o	x	i	m	a	d	a
s																	
a								↖									
p									↖								
r										↖							
o											↖						
x												↖					
i													↖				
m														↖			
a															↖		
r															↑	↖	

Figura 4.2: Ponteiros de retorno para os alinhamentos com similaridade $\geq k$.

A relação de recorrência para solucionar o problema da LCS é mostrada na equação 4.4, que também pode ser resolvido com uso da programação dinâmica.

$$M_{i,j} = \begin{cases} 0 \text{ se } i = 0 \text{ ou } j = 0 & \\ M_{i-1,j-1} + 1 & \text{se } s_i = t_j \\ \max\left(M_{i-1,j}, M_{i,j-1}\right) & \text{se } s_i \neq t_j \end{cases} \tag{4.4}$$

4.3 BUSCAS NÃO-INDEXADAS E INDEXADAS

A **busca não-indexada** ou **busca online** consiste em encontrar um padrão em um texto sem que haja um pré-processamento dele, embora possa existir um pré-processamento do padrão. Tem grande utilidade quando não há tempo nem espaço disponível para armazenar os dados pré-processados. Esses algoritmos têm sido bastante pesquisados e melhorados desde os anos 60 e por isso muitos algoritmos competitivos existem na atualidade [Navarro, 1998].

Mesmo tendo atualmente algoritmos de busca não-indexada eficientes, quando se tem grandes massas de dados para se realizar buscas ou mesmo quando há buscas que são realizadas frequentemente, a velocidade desses algoritmos não é satisfatória [Navarro, 1998]. Nesses casos os algoritmos de **busca indexada** são os indicados, pois assim os textos são pré-processados de forma a agilizar as buscas sobre ele. Essa indexação tem como desvantagem a necessidade de utilização de espaço para armazenamento dos índices (dados pré-processados) além do tempo para esse processamento.

4.4 BUSCAS EM BIOINFORMÁTICA

Algoritmos de busca ou de alinhamento no campo da Bioinformática possuem expressiva importância. Supondo que se possua as sequências genômicas de dois insetos que suspeita-se, sejam relacionados, uma forma de verificar qual trecho de seus genomas são semelhantes e quais são diferentes seria utilizar algoritmos de alinhamento exato (solução ótima, como apresentado anteriormente neste capítulo). No entanto, a comparação de duas sequências genômicas inteiras poderia demorar um longo tempo [Jones e Pevzner, 2004]. Além disso, comparação de longas sequências genômicas quase sempre apresentam diferenças. Para resolver esse tipo de problema foi criado o BLAST. O BLAST, *Basic Local Alignment Tool*, é um dos programas mais utilizados para procurar sequências em um grande banco de dados de sequências [Setubal e Meidanis, 1997], em busca de similaridades.

BLAST alinha a sequência buscada com cada uma das sequências do banco de dados. A cada candidato a *hit* é atribuído um *score* de similaridade S, além de um valor associado à possibilidade de aquele *hit* ter sido encontrado ao acaso. A esse valor é dado o nome de ***E-value*** (E) e ele pode ser um parâmetro de entrada para a busca no BLAST. O *E-value* pode ser calculado a partir do chamado *bit score* (S') que é obtido a partir da Equação 4.5, onde K e λ são constantes que podem ser obtidas conforme [Korf et al., 2003] e S é o chamado *raw score*, obtido somando-se os valores da coluna do alinhamento. O cálculo do

E-value é dado pela Equação 4.6 [National Center for Biotechnology Information, 2011]. Pelo fato do *E-value* indicar o número de *hits* ao acaso que pode se esperar na busca em um banco de dados de sequências de um determinado tamanho, quanto menor o seu valor calculado, mais significativo é o resultado.

$$S' = \frac{\lambda\, S\ - \ln K}{\ln 2} \tag{4.5}$$

$$E = m\ n\ 2^{-S'} \tag{4.6}$$

O BLAST é uma heurística e, para realizar suas buscas, utiliza matrizes de substituição para melhorar a eficiência e introduzir regras mais precisas [Jones e Pevzner, 2004]. É essa matriz que fornece os valores para as colunas do alinhamento e o consequente cálculo de S. A subseção seguinte dará uma visão geral das matrizes de substiuição, em especial da matriz PAM, que, embora não seja utilizada atualmente pelo BLAST, tem grande relacionamento com este trabalho.

4.4.1 Matrizes de Substituição

As matrizes de substituição estão geralmente relacionadas a Bioinformática e são utilizadas na comparação de proteínas. Para entender a evolução das espécies, é importante compreender a evolução das proteínas. Isso significa que, dadas duas proteínas, encontrar a relação de evolução entre elas pode ser importante. E quando não se tem outras informações a respeito delas, uma das saídas é alinhá-las de forma a identificar as mutações que nelas ocorreram com o passar do tempo evolutivo [Eidhammer et al., 2004]. Na comparação de proteínas, a utilização de uma pontuação simples como $+1$ para casamento e -1 para não casamento não é sufiente. O elemento que forma a proteína, chamado aminoácido, possui propriedades bioquímicas que influenciam suas substituições ao longo do processo evolutivo [Setubal e Meidanis, 1997]. Por isso é importante utilizar um modelo de pontuação que reflita essas probabilidades ao longo da evolução para que seja possível realizar um alinhamento biológico adequado das sequências.

Um procedimento padrão para alcançar esse objetivo é a utilização de uma importante família de matrizes de substituição, chamada **matrizes PAM**. O acrônimo *PAM* signi-

fica *Point of Accepted Mutation*, que é a substituição de um aminoácido por outro em uma proteína que foi aceito pela evolução das espécies, ou seja, não resultou na morte dela. As matrizes PAM são construídas a partir da observação de mutações na natureza [Eidhammer et al., 2004]. A matriz PAM-1 é uma matriz de cadeia de *Markov* aplicada para um período de tempo onde se espera uma média de 1% de substituição dos aminoácios para uma determinada proteína. Da PAM1 podem ser derivada diversas outras matrizes PAM, como PAM50, PAM100 e PAM250. A seguir serão apresentados os conceitos envolvidos na construção da matriz PAM, em especial da matriz PAM1, a partir de definições obtidas em [Setubal e Meidanis, 1997].

No modelo das matrizes PAM são consideradas apenas a mutações imediatas $a \rightarrow b$ e não as mediatas como $a \rightarrow c \rightarrow b$. Os ingredientes necessários para a construção da matriz PAM1 M são:

1. Uma lista das mutações aceitas;

2. A probabilidade de ocorrência p_a para cada aminoácido a.

O item 2, necessário para a criação da matriz PAM1, pode ser estimado observando a frequência relativa de ocorrência de aminoácidos sobre um conjunto de proteínas suficientemente grande e variado. Essa probabilidade deve satisfazer:

$$\sum_a p_a = 1 \tag{4.7}$$

Da lista de mutações aceitas é possível calcular f_{ab}, que consiste no número de vezes que a mutação $a \leftrightarrow b$ ocorreu. Como aqui estão sendo tratadas apenas as mutações imediatas, então $f_{ab} = f_{ba}$. Assim, o número total de mutações em que a está envolvido é dado por:

$$f_a = \sum_{b \neq a} f_{ab} \tag{4.8}$$

e o número total de aminoácidos envolvidos nas mutações é:

$$f = \sum_a f_a \tag{4.9}$$

A matriz PAM1 M é uma matriz 20×20 (20 é o número de aminoácidos) onde M_{ab} é a probabilidade do aminoácido a se transformar no aminoácido b. E M_{aa} é a probabilidade do aminoácido a se manter durante um intervalo de tempo. O cálculo de M_{aa} é feito baseado no conceito de *mutabilidade relativa* do aminoácido a, que é definida como:

$$m_a = \frac{f_a}{100 f p_a} \tag{4.10}$$

Mutabilidade relativa é a probabilidade que um determinado aminoácido vai mudar num determinado período de interesse. A probabilidade do aminoácido a permanecer como está é sua probabilidade complementar:

$$m_{aa} = 1 - m_a \tag{4.11}$$

Já a probabiliade do aminoácido a se transformar no aminoácido b pode ser calculada como o produto da probabilidade condicional de que a irá se transformar em b, dado que a mudou, multiplicado pela probabilidade de a mudar:

$$\begin{aligned} M_{ab} &= P(a \to b) && (4.12)\\ &= P(a \to b | a \text{ mudou}) P(a \text{ mudou}) && (4.13)\\ &= \frac{f_{ab}}{f_a} m_a && (4.14) \end{aligned}$$

A independência do passado leva a um modelo de *Markov*, que possui propriedades matemáticas interessantes. Assim, M tem as seguintes propriedades:

$$\sum_b M_{ab} = 1 \tag{4.15}$$

$$\sum_a p_{aa} M_{aa} = 0,99 \tag{4.16}$$

A equação 4.15 está indicando que, se somadas a probabilidade de a permanecer o mesmo e dele se transformar em qualquer outro aminoácido, obtém-se o valor 1. A matriz de probabilidade de transição foi normalizada para refletir o fato de que a evolução irá mudar 1 em cada 100 aminoácidos, como pode ser visto na equação 4.16 e através do denominador 100 na equação 4.10.

Agora é possível definir as **matrizes de substituição** (*scoring matrices*). Os valores dessa matriz estão relacionadas à taxa entre a probabilidade de um par ser uma mutação e uma ocorrência aleatória. Isso é chamado de taxa *likelihood* ou *odds* $\frac{M_{ab}}{p_b}$.

A matriz de substituição para uma distância k PAM é definida como:

$$score_k(a, b) = 10 \log_{10} \frac{M_{ab}^k}{p_b} \tag{4.17}$$

Com o parâmetro k é possível gerar outras matrizes PAM, como por exemplo, alterando k para 120 a matriz gerada é a PAM120. A matriz de substituição *score* da equação refeq:pamk é uma matriz simétrica, ou seja, $score_k(a, b) = score_k(b, a)$. Apesar de ser gerada a partir de observações de *mutações aceitas* e de uma análise geral das frequências dos aminoácidos de forma isolada, as matrizes de substituição podem refletir importantes propriedades químicas e físicas dos aminoácidos, como afinidade a moléculas de água ou seus tamanhos. Por isso, aminoácidos com maior semelhança recebem uma maior pontuação na matriz do que os que não apresentam muitas similaridades.

A ideia principal por trás do algoritmo proposto neste trabalho, apresentado no Capítulo 5, é o uso desse tipo de matriz para medir a frequência em que os erros conhecidos de OCR acontecem.

5 SOCRATEXT

Este capítulo apresenta o algoritmo desenvolvido neste trabalho para realização de buscas de palavras-chaves em textos gerados por OCR. Conforme descrito no Capítulo 3, o texto gerado a partir do reconhecimento ótico de caracteres contidos na imagem de um documento pode possuir diversos erros. A frequência e tipo dos erros que ocorrem variam conforme a ferramenta de OCR utilizada e a imagem de documento submetida a ela. Em geral, quanto menor a qualidade da imagem maior a quantidade de erros no texto reconhecido. Fazendo-se um paralelo aos seres vivos, seria possível comparar o texto que está representado na imagem com a sequência genética de um indivíduo e o texto reconhecido efetivamente pelo OCR como essa sequência genética após sofrer mutação em alguns pontos. Se essa alteração foi aceita pelo OCR, então teria ocorrido a chamada "mutação aceita" (do inglês *accepted mutation*), termo comum na Bioinformática. "Mutação aceita" é a mutação que ocorreu e foi positivamente selecionada pelo ambiente, não ocasionado na morte do indivíduo [Setubal e Meidanis, 1997].

Este trabalho visa a utilizar algumas técnicas desenvolvidas e aplicadas no campo da Bioinformática, combinadas com outras estratégias, para realizar buscas em textos gerados por OCRs, tratando-os como textos que sofreram mutação. O algoritmo criado foi denominado **SOCRATEXT** - "**S**earch in **OCR** **A**cquired **TEXT**" e será apresentado na Seção 5.1. Na Seção 5.2 será mostrado no que se baseia a construção da matriz de substituição utilizada pelo SOCRATEXT.

5.1 ALGORITMO

O SOCRATEXT implementa um algoritmo de **programação dinâmica** (Seção 4.2) que utiliza **matriz de substituição** (Seção 4.4.1) para realizar busca aproximada baseada em similaridade em textos reconhecidos via **OCR**. Diversos tipos de erros gerados pelas ferramentas de OCR ocorrem comumente devido à semelhança entre os símbolos. Como ocorre nos aminoácidos, onde aqueles com propriedades similares têm maior probabilidade de serem substituídos [Setubal e Meidanis, 1997], nos caracteres, as combinações que apresentam maior semelhança de forma também têm maior probabilidade de serem substituídos. Na natureza há muitos fatores que influenciam a probabilidade de substituição mútua de aminoácidos e por isso a observação direta das taxas atuais de substituição é a

melhor maneira de quantificar a similaridade entre eles. No caso dos textos onde o OCR reconheceu caracteres erroneamente, também há vários fatores que influenciam, como sujeira no *scanner*, defeito no documento original, baixa qualidade da digitalização, entre outros. Assim, para se obter quais os erros mais comuns, a melhor forma é a observação dos resultados de diversos reconhecimentos sob diferentes situações. Dessa forma, a ideia básica do algoritmo consiste nos seguintes pontos:

- A partir dos caracteres (ou conjunto de caracteres) que foram mais frequentemente reconhecidos de forma errada (caracteres "mutantes"), é possível montar uma matriz de substituição PAM com esses dados, conforme descrito na Seção 5.2;
- Utilizar a programação dinâmica para realizar a busca aproximada de um padrão s em um texto t, baseado em similaridade. No entanto, na comparação de s com t é considerado um conjunto de caracteres de cada um simultaneamente, de tamanho máximo α. Portanto, não se usa as operações de inserção e remoção. Apenas a operação de substituição é aceita. Assim, as operações de inserção e remoção são modeladas pelas substituições de $0 : w$ e $q : 0$. Trata-se, então, de uma substituição $q : w$ onde $0 \leq q, w \leq \alpha$. O custo da operação de substituição, é obtido pela matriz de substituição construída a partir de erros comumente encontrados em OCR. Isso precisa ser feito porque o espaço em branco também é um símbolo do alfabeto. O que se pretende é não penalizar muito os erros que são mais comuns de ocorrerem.
- Ao final, procura-se na última linha da tabela de programação dinâmica os valores obtidos acima um certo valor de similaridade, selecionando, dessa maneira, as ocorrências do padrão s no texto t, caracterizando assim o algoritmo com o de alinhamento semi-global (local do ponto de vista do texto e global do ponto de vista da palavra-chave).

O algoritmo necessita das seguintes entradas:

M: matriz de substituição;

τ: ordem da matriz de substituição M;

s: padrão a ser buscado;

t: texto onde será efetuada a busca;

m: tamanho de s;

n: tamanho de t;

α: tamanho da maior combinação de símbolos aceita nos erros;

Σ: alfabeto;

Σ': combinações de tamanho $\leq \alpha$ cada uma, contendo símbolos de Σ;

E: *E-value* desejado para a busca.

As saídas produzidas são as seguintes:

$A_{m,n}$: matriz de programação dinâmica contendo as similaridades calculadas;

$B_{m,n}$: matriz contendo os caminhos a serem percorridos em $A_{m,n}$ (ponteiros de retorno).

Para preencher essas matrizes A e B, o algoritmo utiliza programação dinâmica. A matriz de programação dinâmica A é inicializada de forma que o casamento entre o padrão e o texto possa ocorrer em qualquer ponto do texto, assim $A_{0,j} = 0$ para todo $0 \leq j \leq n$. A primeira coluna de A é preenchida obedecendo a $A_{i,0} = A_{i-1,0} + M[s[i], \epsilon]$, onde ϵ é a cadeia de caractere vazia e $1 \leq i \leq m$. Na prática, a inicialização da primeira coluna faz com seja calculada a similaridade entre s e a cadeia vazia ϵ de acordo com o valores contidos na matriz de substituição.

Após as inicializações, o preenchimento da matriz A se dá a partir da relação de recorrência da equação 5.1.

$$A_{i,j} = \max \begin{cases} \max\limits_{1 \leq k \leq \alpha} (A_{i,j-k} + \sum\limits_{r'=0}^{k-1} M[\emptyset, t[j - r' - 1]]) \\ \max\limits_{1 \leq k \leq \alpha} (A_{i-k,j} + \sum\limits_{r=0}^{k-1} M[s[i - r - 1], \emptyset]) \\ \max\limits_{1 \leq r,r' \leq \alpha} (A_{i-r,j-r'} + M[s[i - r..i - 1], t[j - r'..j - 1]]), \end{cases} \tag{5.1}$$

onde $1 \leq r \leq i$; $1 \leq r' \leq j$; e $k \in \mathbb{N}$. A matriz B, que armazena as decisões tomadas, é preenchida da forma que se observa na equação 5.2.

$$B_{i,j} = (i - q, j - q'), \tag{5.2}$$

tal que q e q' foram determinados pelo máximo encontrado no cálculo de $A_{i,j}$ na Equação 5.1.

Tabela 5.1: Matriz de programação dinâmica A para a busca do termo "aue" no texto "aaiiea".

		a	a	i	i	e	a
	0	0	0	0	0	0	0
a	-2	4	4	2	0	2	4
u	-4	2	2	2	7	5	3
e	-6	0	4	2	5	11	9

Para ilustrar o funcionamento do algoritmo, suponha um alfabeto somente com as vogais $\Sigma = \{a, e, i, o, u\}$ e $\alpha = 2$. Suponha ainda que a partir da análise de erros de OCR para reconhecimento de textos contendo apenas esses caracteres, foi criada a seguinte matriz de substituição M:

	a	e	i	o	u	ϵ	ii
a	4	2	-2	-2	-2	-2	-2
e	2	4	-2	-2	-2	-2	-2
i	-2	-2	4	-2	-2	-2	-2
o	-2	-2	-2	4	-2	-2	-2
u	-2	-2	-2	-2	4	-2	3
ϵ	-2	-2	-2	-2	-2	3	-2
ii	-2	-2	-2	-2	-2	-2	3

Cada célula $M_{i,j}$ da matriz armazena a similaridade entre o caractere da linha i com o da coluna j. Por exemplo, a similaridade entre "a" e "e" ($M_{1,2}$) é 2. Na linha 6 da matriz M (considerando que inicia-se na linha 1) encontram-se as similaridades entre a cadeia vazia ϵ e as outras sequências de caracteres. Assim, cada célula dessa linha indica a similaridade entre uma cadeia vazia e o símbolo na posição da coluna j. É importante observar que na linha 7 há um símbolo composto por 2 caracteres ("ii"). Isso permite que seja mapeada a similaridade entre símbolos que são compostos por um conjunto de caracteres, como por exemplo a similaridade entre a letra "u" e as duas letras "i" juntas.

Nesse exemplo, pretende-se encontrar o padrão "aue" no texto "aaiiea". Primeiramente, inicializa-se a matriz de programação dinâmica A. Em seguida, aplica-se a relação de recorrência (5.1), preenchendo a matriz de programação dinâmica A e a matriz com os ponteiros de retorno B, conforme Matrizes 5.1 e 5.2, respectivamente.

Tabela 5.2: Ponteiros de retorno na busca do termo “aue” no texto “aaiiea”.

		a	a	i	i	e	a
a			↖				
u				←	←		
e						↖	

E finalmente, para encontrar as ocorrências do padrão “aue” no texto “aaiiea”, deve-se buscar na última linha da matriz de programação dinâmica A, quais células apresentaram a similaridade desejada. Para a determinação da similaridade desejada, utilizou-se o conceito de *E-value*, utilizado no **BLAST** (Capítulo 4). O *E-value* no algoritmo SOCRATEXT é determinado utilizando as equações 4.5 e 4.6. No entanto, as seguintes adaptações foram realizadas na escolha dos parâmetros envolvidos no cálculo:

- K: constante estabelecida a partir de valor sugerido em [Lawless, 2003] e validada em experimentos preliminares. O valor fixado foi 0,035;
- λ: constante estabelecida a partir de valor sugerido em [Lawless, 2003] e validada em experimentos preliminares. O valor fixado foi 0,252;
- m: tamanho do termo de busca s;
- n: tamanho do texto t onde está sendo efetuada a busca;
- S: *score* calculado pelo algoritmo SOCRATEXT e obtido a partir da última linha da matriz de programação dinâmica M.

Com esses parâmetros é possível calcular o *E-value* de cada *score* S_j presente na última linha da matriz de programação dinâmica M a partir da equação 4.6. Assim, a identificação do último caractere do(s) *hit(s)* da busca do termo s no texto t é dada pela relação descrita na equação 5.3.

$$\textit{E-value}(S_j) \leq E. \tag{5.3}$$

Encontradas as posições que satisfazem a relação 5.3, os *hits* podem ser construídos a partir da matriz B.

5.1.1 Complexidade do Algoritmo

O algoritmo SOCRATEX faz uso de algoritmo de programação dinâmica para solucionar a relação de recorrência descrita na equação 5.1. No preenchimento da matriz de programação dinâmica, também há busca na matriz de substituição por combinações de p caracteres por l caracteres, onde $0 \leq p, l \leq \alpha$. A matriz de substituição possui ordem τ e a cada busca de correspondência nela, caso implementada como uma busca binária, custa $\log_2 \tau$. Assim, a complexidade de tempo do algoritmo SOCRATEXT é $O(m\ n\ \alpha^2\ \log_2\ \tau)$.

Considerando que α e τ são valores muito pequenos, quando comparados com n, pode-se dizer que o SOCRATEXT tem complexidade de $O(mn)$.

5.2 MATRIZ DE SUBSTITUIÇÃO

A matriz de substituição permite que os erros e acertos mais comuns no reconhecimento ótico de caracteres sejam mapeados em forma de pontuações que indicam a similaridade entre os símbolos. Sua construção é feita baseada nas matrizes PAM. Recapitulando da Seção 4.4.1, para a construção da matriz PAM é necessário possuir a lista de mutações aceitas e a probabilidade de ocorrência para cada aminoácido. Para a montagem da matriz de substituição utilizada no SOCRATEXT, são necessárias as seguintes informações:

- A lista dos caracteres corretamente reconhecidos;
- A lista dos caracteres (ou conjunto deles) reconhecidos de forma errada (mutação);
- A probabilidade do reconhecimento correto e incorreto para o conjunto de texto utilizado.

De forma a obter esses dados, é necessário ter um método de comparação entre os textos originais (sem erros) e os textos reconhecidos via OCR. Através dessa comparação poderão ser mapeados quais caracteres foram corretamente reconhecidos e quais geraram erros, incluindo a frequência de ocorrência deles. Para identificar os erros de OCR foi utilizado o conceito de LCS (*Longest Common Subsequence*) [Gusfield, 1997]. Primeiramente, é calculada a LCS entre o texto original e o gerado por OCR. Em seguida, percorre-se os ponteiros de retorno da matriz de LCS gerando a lista de erros.

A algoritmo acima descrito devolve uma lista de erros no formato $l \rightarrow p$, onde l e p são sequências de caracteres de tamanho ≥ 0, indicando que a sequência de caractere(s) l foi incorretamente reconhecida como p. Em seguida, são agrupados e contados os acertos e os

erros iguais (l e p iguais). Como os erros podem ser compostos de combinações de caracteres (como "$cl \rightarrow d$", "$w \rightarrow vv$" e "$iii \rightarrow m$"), nesse momento é criado um novo alfabeto Σ', que é formado pelos caracteres do alfabeto Σ além dos novos símbolos formados, compostos por combinações de caracteres desse alfabeto.

Com base nesse novo alfabeto Σ', é montada uma matriz contendo a frequência de todos os reconhecimentos realizados pelo OCR (corretos ou não). A essa matriz é dado o nome de **Matriz de Frequência**. A matriz de frequência contém os dados necessários para a construção da **Matriz de Substituição**, de forma semelhante à criação da matriz PAM (Seção 4.4.1). O Capítulo 7 apresenta detalhes de como todo o processo de montagem da matriz de substituição foi elaborado.

6 DEGRADAÇÃO DE IMAGEM E RECUPERAÇÃO DA INFORMAÇÃO

O uso da tecnologia de OCR é uma alternativa quando se deseja pesquisar por palavras-chaves em imagens de documentos. No campo da computação forense, essa tecnologia foi recentemente incorporada à versão 3.1 do FTK, permitindo realizar buscas de palavras-chaves em imagens. Neste capítulo, que traz resultados de um trabalho publicado em [Polastro e F. Almeida Jr., 2011], é proposto um método para avaliar os efeitos dos erros de OCR na recuperação da informação em textos na língua inglesa e portuguesa usando este recurso do FTK com diferentes níveis de degradação de imagens. O método é descrito em detalhes, e usa ferramentas e dados públicos, permitindo que o experimento seja reaplicado a outras versões de OCR, com outro conjunto de imagens, com textos em outra língua e com outro programa de OCR.

De uma forma geral, este capítulo tem como objetivo responder às seguintes perguntas:

- É possível realizar busca de palavra-chave em imagens utilizando a tecnologia OCR na área da Informática Forense?
- Como a busca de palavra-chave em imagens se comporta com diferentes níveis de degradação de imagens?
- A funcionalidade de utilização de OCR para realizar busca de textos em imagens do FTK produz resultados equivalentes para textos em português e inglês?

6.1 EXPERIMENTOS

Os experimentos têm como objetivo verificar como se comporta a busca de palavras-chaves em imagens utilizando a funcionalidade de OCR do software FTK, tanto para textos em português quanto em inglês. Para isso, foram efetuadas buscas de palavras-chaves conhecidas em diversos conjuntos de imagens, tendo cada conjunto imagens com níveis de degradação diferentes. As degradações aplicadas às imagens neste experimento foram: diminuição de resolução, rotação e sal-e-pimenta. O processo de teste engloba desde a geração dos documentos a serem escaneados até a verificação dos resultados obtidos da busca

de palavras-chaves do FTK. Uma visão geral desse processo está ilustrada na Figura 6.1. Nas subseções seguintes cada um desses passos é detalhado.

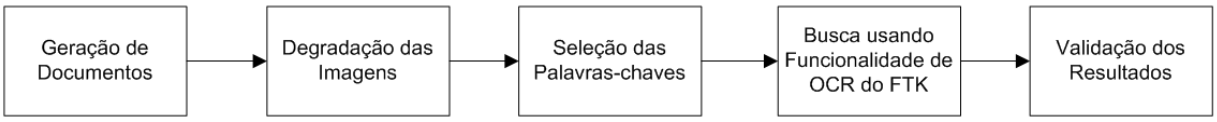


Figura 6.1: Visão geral do experimento de verificação da funcionalidade de busca de textos em imagem do FTK.

6.1.1 Geração dos Documentos

Os documentos foram criados a partir de simples arquivos de texto, obtidos do *corpus* "Summ-it" [Collovini et al., 2007]. O *corpus* consiste de 50 textos jornalísticos em português do caderno de Ciência do jornal Folha de São Paulo. Desse *corpus* foram escolhidos dez arquivos para realização dos experimentos. Os textos em inglês foram obtidos a partir desses dez arquivos de texto, após tradução automática realizada pelo Google Tradutor [Google, 2011]. A Tabela 6.1 exibe o nome dos arquivos originais escolhidos do *corpus* "Summ-it" e o número de caracteres presentes em cada um deles. Em seguida, esse conjunto, de agora vinte arquivos, foi formatado na fonte "Times New Roman" de tamanho 12 utilizando LaTeX. Através do programa **pdflatex**, esses arquivos foram convertidos para pdf, resultando em documentos de apenas uma folha cada.

Todos os documentos pdf foram impressos em uma impressora Hewlett-Packard Laserjet 5L com 300 dpi de resolução utilizando papel Chamex A4 com 210x297mm e 75 g/m2. Em seguida, esses textos impressos foram escaneados de forma cautelosa, buscando evitar a inserção de ruídos como rotação, utilizando um scanner Hewlett-Packard Scanjet 8270, produzindo imagens de 300 dpi do tipo *tif*. Assim, ao final dessa etapa foram geradas vinte imagens de documentos, sendo dez em português e dez em inglês.

6.1.2 Degradação das Imagens

As imagens criadas como descrito na Seção 6.1.1 são consideradas de alta qualidade para a utilização no reconhecimento ótico de caracteres, pois não apresentam variações de *layout*, possuem fonte conhecida, não estão rotacionadas e possuem resolução considerada ótima para utilização em OCRs. Como a maioria das imagens encontradas em exames de informática forense não apresenta tais qualidades, uma série de degradações será aplicada a

Tabela 6.1: Arquivos selecionados do *corpus* Summ-it e a quantidade de caracteres neles presentes para as versões em português e inglês.

Nome do Arquivo no *corpus* Summ-it	Português	Inglês
CIENCIA_2001_19858.txt	2921	2822
CIENCIA_2002_22005.txt	2841	2724
CIENCIA_2002_22023.txt	2406	2197
CIENCIA_2002_22027.txt	2952	2720
CIENCIA_2003_24212.txt	2896	2733
CIENCIA_2004_26417.txt	2748	2591
CIENCIA_2005_28743.txt	2632	2468
CIENCIA_2005_28752.txt	2778	2630
CIENCIA_2005_28754.txt	2838	2665
CIENCIA_2005_28756.txt	2790	2620
Total	27802	26170

elas, de forma a simular um comportamento mais próximo do mundo real. As degradações serão aplicadas de forma separada nas imagens. Uma das vantagens da degradação sintética de imagens é a possibilidade de mantê-las sob controle [Kanungo, 1996], fazendo alterações conforme os objetivos do experimento.

Para a degradação das imagens foram utilizados comandos do software de processamento de imagens **imagemagick** [ImageMagick, 2011]. Para aplicação da degradação sal-e-pimenta foi utilizado o comando *imnoise* do *software* **GNU Octave** [GNU, 2011]. Na Tabela 6.2, estão ilustrados os tipos de degradações aplicados, os comandos utilizados, e o nome de cada conjunto criado, para futuras referências.

Tabela 6.2: Tipos de degradações aplicadas às imagens.

Tipo de Degradação	Comando	Nome do Grupo
Resolução 200 dpi	convert -resample 200 -depth 8 -type Grayscale [pdf_file] [tif_file]	Res200
Resolução 150 dpi	convert -resample 150 -depth 8 -type Grayscale [pdf_file] [tif_file]	Res150
Rotação +3	convert –rotate 3 -depth 8 -type Grayscale [pdf_file] [tif_file]	Skw3
Rotação +6	convert –rotate 6 -depth 8 -type Grayscale [pdf_file] [tif_file]	Skw6
Sal-e-pimenta 1%	imnoise([tif_file],'salt & pepper', 0.01)	S&P1
Sal-e-pimenta 3%	imnoise([tif_file],'salt & pepper', 0.03)	S&P3

6.1.3 Seleção das Palavras-chaves

Os textos em português e em inglês criados na etapa de geração de documentos, descritos na Seção 6.1.1, foram processados para separar todas as palavras que neles ocorrem. Para

esta separação, foram considerados como *tokens* os seguintes caracteres:“[espaço] . , ; : - “ ( ) [ ]”. A partir daí, foram identificadas as palavras que aparecem apenas uma vez em cada conjunto de textos, sendo elas separadas em dois grupos: palavras entre 5 e 7 caracteres, inclusive; e palavras com 8 ou mais caracteres.

Para os textos em português foram identificadas 451 palavras únicas com acentos entre 5 e 7 caracteres e 561 maiores ou iguais a 8 caracteres, e 405 e 372 para os textos em inglês, respectivamente. Para as palavras em português sem acentos foram identificadas 375 e 424 palavras únicas para os grupos entre 5 e 7 caracteres e maiores ou iguais a 8, respectivamente. Essas palavras foram então salvas em dois arquivos textos, codificados em UTF-8, com cada palavra ocupando uma linha do arquivo. Em experimentos preliminares a este, foi identificado que a funcionalidade de OCR utilizada pelo FTK não trabalha bem com palavras acentuadas. Desta forma, foi criado um conjunto de testes a mais para a categoria de textos em português. Esse conjunto é composto por todas as palavras do outro conjunto em português, exceto as que apresentam pelo menos um dos seguintes caracteres em sua formação: “á ã â é ê í ó ô õ ú ü ç”, ou seus respectivos em caixa alta. A Tabela 6.3 exibe os grupos criados, juntamente com suas principais características.

Tabela 6.3: Arquivos de palavras-chaves criados.

Descrição do Arquivo	Estat. (palavras)	ID
Português com acentos (palavras entre 5 e 7 caract.)	451	pt1
Português com acentos (palavras $\geq$ 8 caract.)	561	pt2
Português sem acentos (palavras entre 5 e 7 caract.)	375	pt3
Português sem acentos (palavras $\geq$ 8 caract.)	424	pt4
Inglês (palavras entre 5 e 7 caract.)	398	en1
Inglês (palavras $\geq$ 8 caract.)	368	en2

6.1.4 Busca Usando a Funcionalidade de OCR do FTK

A etapa de degradação das imagens (Seção 6.1.2) gerou 12 conjuntos de imagens (6 para cada linguagem), cada um contendo 10 imagens de documentos. Para cada um desses conjuntos foi criada uma pasta. No FTK foi criado um novo caso para cada uma dessas pastas, aplicando, além das configurações padrões, a conversão de imagens para textos através de OCR. Foram retiradas todas as limitações quanto a tamanhos mínimo e máximo de arquivos de imagens a serem convertidos. Após o processamento do caso pelo FTK, foi realizada a busca indexada das palavras criadas na etapa anterior. Para realizar tal

busca, foi importado o arquivo de palavras-chaves na linguagem adequada e realizadas as buscas em todos os arquivos com extensão "ocr", que é a extensão utilizada pelo FTK para representar os textos gerados pelo OCR nativo da ferramenta. Todas as buscas realizadas foram buscas exatas, ou seja, somente são retornadas as *strings* iguais às procuradas.

6.1.5 Validação dos Resultados

Para a validação dos resultados, foi efetuada comparação entre o número de *hits* retornados pela busca indexada com o número de palavras contidas no arquivo de busca importado para a ferramenta. Se o resultado do OCR fosse 100% correto, esses números deveriam ser iguais. No entanto, com a ocorrência de erros OCR, a busca exata retorna menos *hits*. Sendo k o número de palavras a serem buscadas e h o número de hits retornados, a taxa de acerto é calculada conforme a equação 6.1.

$$Taxa\ de\ Acerto = \frac{h}{k} \tag{6.1}$$

A Tabela 6.4 identifica os grupos de imagens de documentos que foram usados no experimento, com suas características principais. Para o grupo "ptDOC" foram realizadas buscas de palavras-chaves acentuadas e não acentuadas (veja Tabela 6.3).

Tabela 6.4: Identificação dos grupos de imagens utilizados no experimento.

Grupo	Subgrupo	Nome do subgrupo	Palavras-chaves usadas
Documentos em Português (ptDOC)	Resolution 200 dpi	Res200	pt1, pt2, pt3, pt4
	Resolution 150 dpi	Res150	pt1, pt2, pt3, pt4
	Skew +3	Skw3	pt1, pt2, pt3, pt4
	Skew +6	Skw6	pt1, pt2, pt3, pt4
	Salt-and-pepper 1%	S&P1	pt1, pt2, pt3, pt4
	Salt-and-pepper 3%	S&P3	pt1, pt2, pt3, pt4
Documentos em Inglês (enDOC)	Resolution 200 dpi	Res200	en1, en2
	Resolution 150 dpi	Res150	en1, en2
	Skew +3	Skw3	en1, en2
	Skew +6	Skw6	en1, en2
	Salt-and-pepper 1%	S&P1	en1, en2
	Salt-and-pepper 3%	S&P3	en1, en2

A Figura 6.2 ilustra todo o método proposto, identificando todos os passos descritos acima e suas interações.

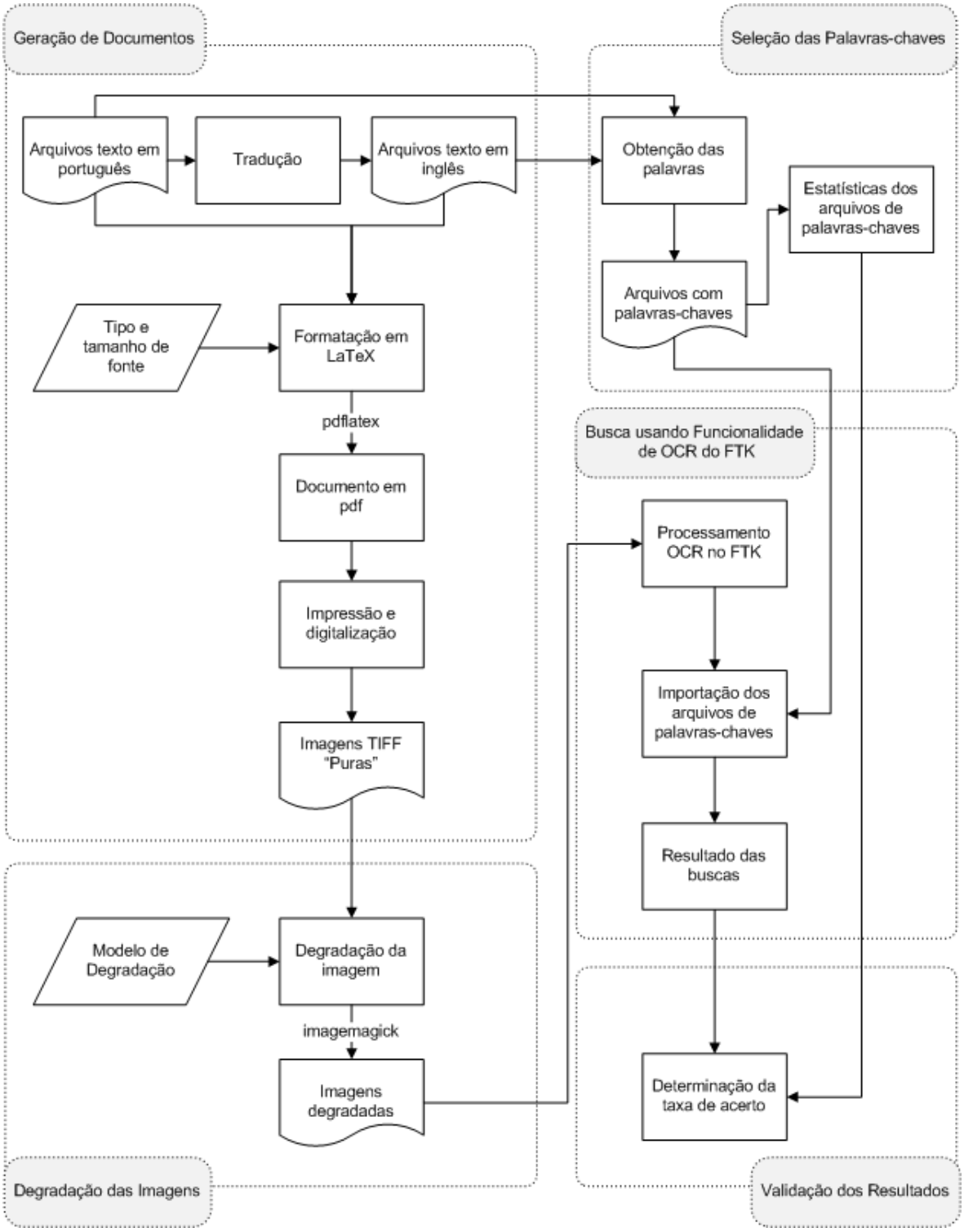


Figura 6.2: Método proposto para avaliação de funcionalidade de busca de palavra-chave em imagem no FTK.

6.2 RESULTADOS E DISCUSSÃO

Os resultados obtidos após aplicar o método proposto são sumarizados em 4 gráficos. Os gráficos exibidos na Figura 6.3 são relacionadas a documentos em português e os exibidos na Figura 6.4 aos documentos em inglês. Os gráficos mostram o percentual de acerto nas buscas de palavras nos textos reconhecidos por OCR, baseado na Equação 6.1. Em todos os gráficos são apresentados os valores obtidos para diferentes níveis de degradação em ordem crescente de acerto.

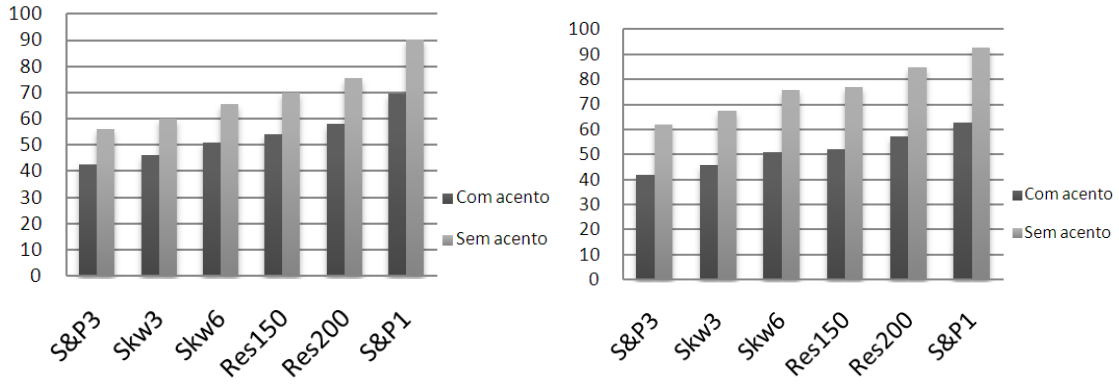


Figura 6.3: Resultados das buscas de palavras-chaves (com e sem acento) em português para palavras com tamanho ≥ 5 e ≤ 7 (esq.) e ≥ 8 (dir.).

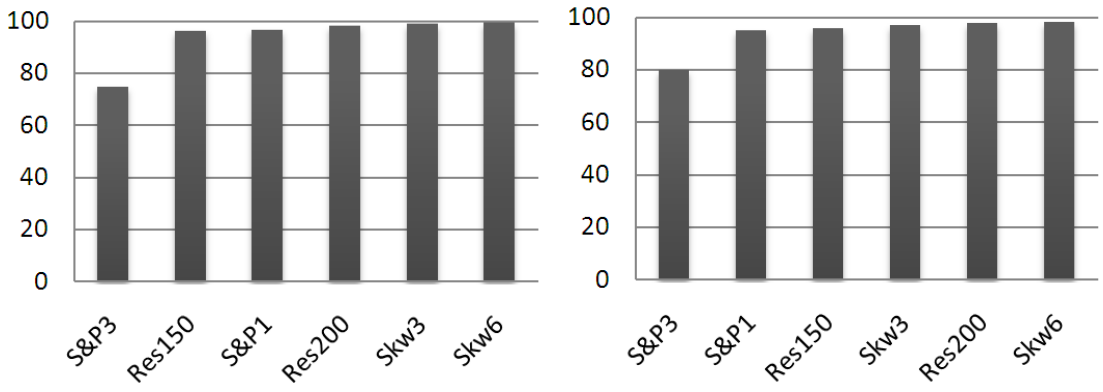


Figura 6.4: Resultados das buscas de palavras-chaves em inglês para palavras com tamanho ≥ 5 e ≤ 7 (esq.) e ≥ 8 (dir.).

A partir da análise dos gráficos da Figura 6.3 é possível identificar que a taxa de acerto em palavras não acentuadas foi consideravelente maior. A análise dos textos gerados

pelo OCR do FTK indica que na maioria dos casos o acento (ou caracteres especiais) foi ignorado ou gerou confusão no processo de reconhecimento, ocasionando em erros como, por exemplo, troca da letra "ç" por "e" diversas vezes.

Para o grupo de documentos em inglês, a variação do comprimento das palavras influenciou a taxa de acerto para diferentes tipos de degradação de imagem, conforme pode ser visto nos gráficos exibidos na Figura 6.4. Nos textos em inglês a degradação de sal-e-pimenta 3% diminuiu consideravelmente o acerto, quando comparado às outras degradações. Embora fosse esperado que uma maior rotação às imagens implicaria em menores taxas de acerto, isso não aconteceu. A causa para esse comportamento não foi identificada, mas pode estar relacionada à habilidade do *software* de OCR detectar e corrigir inclinações dos textos [Smith, 2007].

O método desenvolvido pode levar a um pequeno grau de erro, pois algumas palavras podem ser reconhecidas pelo OCR de maneira errada, criando outra palavra que também está na lista de palavras-chaves a serem buscadas. Embora haja essa possibilidade, a chance de ocorrer é pequena e não invalida os resultados obtidos.

De uma forma geral, os resultados apontaram resultados bem superiores para as buscas efetuadas sobre textos em inglês. Enquanto a média de acerto para textos em português foi abaixo de 80%, nos textos em inglês esse valor alcançou cerca de 95%. Possivelmente, a razão para esta grande diferença no sucesso entre as duas línguas é o fato de que o FTK não está preparado para usar o Tesseract OCR [Smith, 2011] em textos na língua portuguesa, embora o Tesseract já tenha suporte para reconhecimento de textos em português. As degradações de imagem, em especial a aplicação do ruído sal-e-pimenta em 3%, teve uma grande influência nos resultados obtidos. Uma vez que é impossível garantir a qualidade de imagens que os peritos de informática forense vão receber durante o exames em mídias de armazenamento digital, a degradação de imagem pode ser um grande problema no uso da tecnologia OCR. Assim, é necessário criar mecanismos para melhorar os resultados da busca. Uma maneira seria implementar algoritmos de busca aproximada que consideram os erros mais comuns de OCR, a fim de garantir que os documentos importantes possam ser recuperados. Embora o FTK tenha o recurso de busca aproximada, esta pode não ser a solução ideal, porque os erros comuns de OCR não são levados em conta, o que pode levar a muitos falsos positivos e falsos negativos.

O estudo conduzido neste capítulo publicado em [Polastro e F. Almeida Jr., 2011] permitiu verificar que as degradações de sal-e-pimenta, rotação e resolução em imagens de documentos podem afetar substancialmente o resultado das buscas por gerarem níveis

diferenciados de erros. Assim, no próximo capítulo serão conduzidos experimentos relacionados ao algoritmo SOCRATEXT, que leva em consideração os erros de OCR mais comuns, além de utilizar como *benchmark* os resultados obtidos utilizando a busca aproximada com a distância de Levenshtein, que é a utilizada pela busca aproximada do FTK (*Fuzzy Search*).

7 EXPERIMENTOS E RESULTADOS

Este capítulo descreve os experimentos conduzidos para a montagem da matriz de substituição e para as buscas de palavras-chaves utilizando o algoritmo SOCRATEXT. Os métodos utilizados nos experimentos são descritos em detalhes, assim como os textos utilizados na geração dos documentos e imagens degradadas. O principal objetivo dos experimentos é avaliar a hipótese de que levar em consideração os erros típicos de OCR e o uso de matrizes de substituição no cálculo de similaridade entre cadeias de caracteres pode melhorar os resultados das buscas de palavras-chaves em textos reconhecidos por programas de OCR.

7.1 *CORPORA*

Para os experimentos, dois *corpora* distintos foram escolhidos, sendo um utilizado para a montagem da matriz de substituição e outro para realização de buscas de palavras-chaves, doravante denominados ***Corpus* de Treinamento** e ***Corpus* de Testes**, respectivamente. Todos os arquivos utilizados foram convertidos para o formato UTF-8, para compatibilidade com os resultados do OCR utilizado.

7.1.1 *Corpus* de Treinamento

A coleção de textos escolhida para a construção da matriz de substituição foi a **Coleção Dourada (CD) Harem** [Mota et al., 2008]. A CD Harem é um *corpus* criado para o uso da avaliação dos sistemas no HAREM, que é uma avaliação conjunta na área do reconhecimento de entidades mencionadas em português. Tal coleção é composta por um conjunto de vários textos de diversas fontes contendo palavras anotadas para utilização no reconhecimento de entidades. Para obter a coleção de texto utilizada neste trabalho, foram retiradas da versão original do *corpus* no formato *XML* as anotações, restando o texto sem formatação e livre de anotações, totalizando **342.287 caracteres**.

7.1.2 *Corpus* de Testes

Para a realização dos testes de validação do algoritmo SOCRATEXT foi utilizado o *corpus* **Summ-it** [Collovini et al., 2007]. Como dito anteriormente, trata-se de uma coleção composta por 50 textos jornalísticos na língua portuguesa extraídos do caderno de Ciências

do jornal Folha de São Paulo. Entre os arquivos disponíveis foram escolhidos 10 em seu formato original (sem anotações), os quais estão listados na Tabela 7.1, totalizando **25.907 caracteres**.

Tabela 7.1: Arquivos selecionados do *Corpus* Summ-it para teste do algoritmo SOCRATEXT.

Nome Original no *corpus* Summ-it	Número de Caracteres
CIENCIA_2001_19858.txt	2.921
CIENCIA_2002_22005.txt	2.841
CIENCIA_2002_22023.txt	2.406
CIENCIA_2002_22027.txt	2.952
CIENCIA_2003_24212.txt	2.896
CIENCIA_2000_17088.txt	1.921
CIENCIA_2000_17113.txt	2.304
CIENCIA_2005_28752.txt	2.778
CIENCIA_2003_24219.txt	2.256
CIENCIA_2005_28743.txt	2.632
Total	25.907

7.2 FERRAMENTA DE OCR UTILIZADA

A ferramenta de OCR escolhida neste trabalho foi a versão 3.00 do **Tesseract** [Smith, 2011], cuja descrição mais detalhada foi feita na Seção 3.2. Os principais motivos que levaram a sua utilização foram:

- Ter precisão comprovada em testes realizados [Rice et al., 1995];
- Ser *software livre*;
- Ter suporte ao reconhecimento de textos em português;
- Possibilitar a utilização por meio de linha de comando;
- Estar em constante evolução.

No melhor do nosso conhecimento, não existe nenhuma outra ferramenta de OCR que atenda a todos os requisitos acima elencados.

7.3 TREINAMENTO

Para a utilização do algoritmo SOCRATEXT proposto no Capítulo 5, é necessária a criação da matriz de substituição, que pode ser entendida como uma etapa de treinamento. A matriz de substituição tem um papel essencial no sucesso do algoritmo, já que ela será a fonte de pesquisa para cálculo de similaridade entre as sequências comparadas. Uma matriz de substituição ideal para o SOCRATEXT é a que mapeia todos os erros possíveis de ocorrer para qualquer sistema de OCR utilizado, com qualquer qualidade e tipo de documento escaneado. Como a diversidade de fatores que afetam esse processo é muito grande, a construção dessa matriz "ideal" é inviável. Assim, é aplicado um método prático que tem como objetivo mapear algumas dessas possibilidades de forma a validar o algoritmo proposto.

O fluxograma exibido na Figura 7.1 ilustra uma forma com que a identificação de erros OCR pode ser realizada. No entanto, para que seja possível sua aplicação, é necessária a definição de quais textos serão utilizados como os textos originais e quais são seus respectivos textos de OCR. Há diversas formas de se realizar tal tarefa, dentre elas:

1. Utilizar imagens de documentos já escaneados, sem possuir o documento em formato eletrônico. Uma desvantagem dessa abordagem é que o texto original deverá ser digitado baseado na imagem dele, o que, além do tempo necessário para a digitação, não está à prova de erros. Outra desvantagem é que a inserção de degradação no texto (rotação, defeitos na imagem, etc) não estaria totalmente sob controle;

2. De posse do documento em formato texto no computador, imprimí-lo e escaneá-lo. A vantagem desse caso é que já se teria o texto original disponível. A desvantagem é que tem o custo associado à impressão e, além disso, da mesma forma que o método anterior, a degradação não estaria sob total controle pois o simples ato de escanear o documento pode gerar degradações inesperadas;

3. Utilizar bases de dados públicas contendo tanto os textos originais quanto suas respectivas imagens. Não foram encontrados textos em português nessas bases públicas, não podendo ser aplicadas ao contexto deste projeto;

4. Criação sintética do documento escaneado. As principais vantagens desse método estão descritas na Seção 3.1, o que inclui principalmente controle em todo o processo e facilidade de gerar novos documentos sob diferentes formatos e línguas.

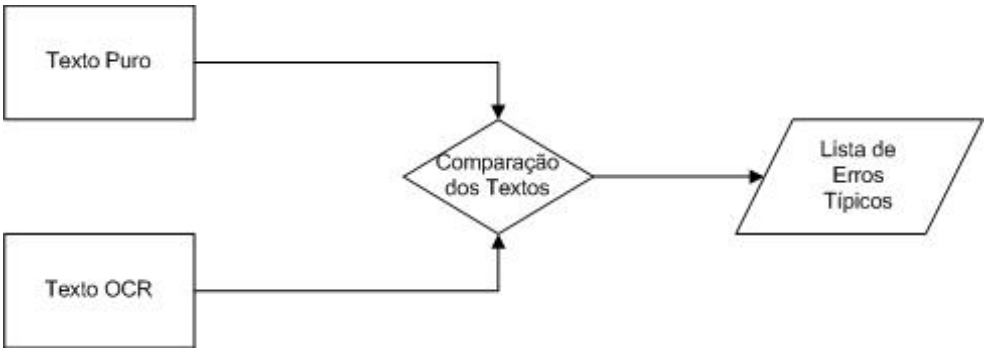


Figura 7.1: Levantamento dos erros típicos de OCR.

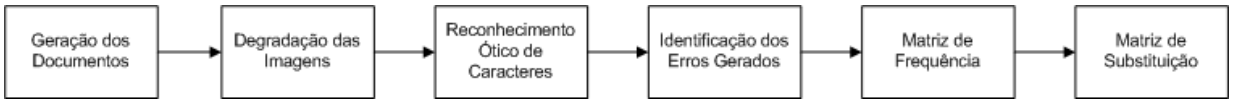


Figura 7.2: Método proposto para geração da matriz de substituição.

Neste trabalho foi escolhida a opção 4 (criação sintética do documento escaneado). O processo proposto para a montagem da matriz de substituição a partir de textos plenos é composto de diversas etapas, as quais estão sumarizadas na Figura 7.2. Nas próximas seções, cada uma dessas etapas será detalhada.

7.3.1 Geração dos Documentos

Os documentos foram gerados a partir dos textos contidos no *corpus* descrito na Seção 7.1.1. Os textos dessa coleção foram separados em arquivos de texto pleno de modo que nas etapas seguintes cada arquivo *pdf* e *tif* gerados a partir do texto não ocupasse mais do que uma página, no intuito de manter compatibilidade com as ferramentas de processamento de imagem e OCR utilizadas. Em seguida, todos os arquivos no formato texto obtidos foram formatados nas fontes "Times New Roman" e "Arial" em tamanho 12 pontos. Essas duas fontes foram escolhidas por serem muito utilizadas atualmente e também por representarem duas categorias de fontes distintas: as serifadas e as não serifadas. A formatação dos documentos foi feita utilizando comandos LaTeX, de modo que a página não utilizasse estilos e que as palavras não fossem hifenizadas ao final da linha.

A partir dos fontes *tex* foram gerados documentos no formato *pdf* com o uso do programa **pdflatex**, que, por sua vez, foram convertidos para arquivos de imagem no formato

tif utilizando o programa **imagemagick** [ImageMagick, 2011]. Essas imagens geradas podem ser consideradas imagens limpas, pois foram criadas de forma eletrônica sem aplicação de nenhum tipo de degradação.

7.3.2 Degradação das Imagens

Esta etapa consiste em aplicar degradações às imagens *tif* geradas anteriormente de forma a simular defeitos que podem ser inseridos no processo de digitalização de documentos impressos. Para isso, é utilizado o mesmo método descrito no Capítulo 6. As degradações de rotação, resolução e sal-e-pimenta são aplicadas separadamente em cada imagem original. Os tipos de degradações aplicados e os comandos e ferramentas utilizados estão exibidos na Tabela 7.2.

Tabela 7.2: Degradações aplicadas às imagens no processo de treinamento.

Degradação	Programa	Comando
Resolução 200 dpi	ImageMagick	convert -resample 200 -depth 8 -type Grayscale [pdf_file] [tif_file]
Resolução 150 dpi	ImageMagick	convert -resample 150 -depth 8 -type Grayscale [pdf_file] [tif_file]
Rotação +5o	ImageMagick	convert -rotate 5 -depth 8 -type Grayscale [pdf_file] [tif_file]
Rotação -5o	ImageMagick	convert -rotate -5 -depth 8 -type Grayscale [pdf_file] [tif_file]
Sal-e-pimenta 1%	Octave	imnoise([tif_file],'salt & pepper', 0.01)
Sal-e-pimenta 3%	Octave	imnoise([tif_file],'salt & pepper', 0.03)

7.3.3 Reconhecimento Ótico de Caracteres

O reconhecimento ótico dos caracteres contidos nas imagens degradadas geradas na etapa anterior foi realizado com a utilização do **Tesseract**, utilizando o treinamento nativo da ferramenta para a língua portuguesa. O comando utilizado está exibido na Tabela 7.3. O "[arquivo_entrada]" é o arquivo *tif* contendo a imagem do documento com uma das degradações listadas na Tabela 7.2 aplicada. O "[arquivo_saida]" é o arquivo texto pleno gerado pelo Tesseract após o reconhecimento ótico de caracteres, no formato UTF-8. O parâmetro "-l por" indica ao Tesseract que deve ser utilizada a base de dados de treinamento para a língua portuguesa.

7.3.4 Identificação dos Erros Gerados

Para a identificação dos erros gerados no processo de reconhecimento ótico de caracteres, é necessária a comparação de cada texto original (provenientes do *corpus* e utilizados para

Tabela 7.3: Linha de comando utilizada no reconhecimento ótico de caracteres utilizando o Tesseract.

Linha de Comando
tesseract [arquivo_entrada] [arquivo_saida] -l por

gerar as imagens sem degradações) com o respectivo texto gerado pelo OCR na imagem degradada, conforme fluxo ilustrado na Figura 7.1. A forma como essa comparação foi realizada está detalhada na Seção 4.4.1.

Durante todo o treinamento, foram identificados 88.824 erros. Esses erros estavam no formato $l \rightarrow p$, onde l e p são sequências de caracteres de tamanho ≥ 0. Foi então realizada uma filtragem para eliminar erros em que ou l ou p possuíssem mais do que 4 caracteres, por serem pouco frequentes e por se repetirem com menor frequência. Desta forma, a quantidade de erros caiu para 58.863. Por fim, esses erros remanescentes foram agrupados. Após o agrupamento a lista de erros ficou com 168 erros distintos do tipo $l \rightarrow p$. Assim, embora tenha-se iniciado com 88.824 erros, após a filtragem e agrupamento a lista se resumiu a 168 erros apenas, indicando que os erros de OCR se repetem em grande escala. A Tabela 7.4 exibe a lista dos vinte erros de OCR mais frequentes para o conjunto de treinamento.

O quarto e décimo terceiro erros que mais ocorreram envolveram o caractere único "LATIN SMALL LIGATURE FI" do UTF-8 na coluna **para** e que dificilmente pode ser detectada diferença de forma visual para as letras "f" e "i" em sequência. Outro caractere bastante semelhante e pouco comum que foi trocado é o **para** localizado na décima nona posição, que é o caracter "DEGREE SIGN" do UTF-8.

7.3.5 Matriz de Frequência

A matriz de frequência é utilizada para armazenar o número de vezes que um símbolo pertencente ao alfabeto utilizado foi reconhecido correta ou incorretamente. Nela, cada posição $f_{i,j}$ indica quantas vezes o símbolo localizado na posição i foi reconhecido pelo OCR como sendo o símbolo localizado na posição j. Trata-se de uma matriz quadrada, onde as linhas e coluna mapeiam os símbolos do alfabeto e, portanto, a diagonal principal armazena os valores para os símbolos corretamente reconhecidos pelo OCR. A montagem da matriz de frequência é realizada da seguinte forma:

Tabela 7.4: Lista dos 20 erros de OCR que mais ocorreram durante o processo de treinamento.

Ranking	**De**	**Para**	**Núm. Ocorrências**
1	[espaço]	[nada]	18.153
2	l	I	7.011
3	[nada]	[espaço]	4.470
4	fi	fi	2.601
5	[nada]	A	1.030
6	rn	m	766
7	[nada]	,	733
8	o	0	650
9	»	>	601
10	á	a	597
11	[nada]	.	573
12	[nada]	S	555
13	[nada]	fi	533
14	if	[nada]	530
15	j	i	506
16	[nada]	E	491
17	[nada]	i	484
18	[espaço]	f	449
19	º	∘	412
20	m	rn	407

1. Criação da matriz F com a inserção dos caracteres que podem ser impressos da tabela ASCII (*American Standard Code for Information Interchange*), com código decimal entre 32 e 126;

2. Adição de outros caracteres comumente encontrados na língua portuguesa: “á”, “à”, “ã”, “é”, “ê”, “í”, “ó”, “ô”, “õ”, “ú”, “ü”, “ç”, “Á”, “À”, “Ã”, “É”, “Ê”, “Í”, “Ó”, “Ô”, “Õ”, “Ú”, “Ü” e “Ç”;

3. Para todo conjunto de texto utilizado no treinamento, conta o número de vezes que cada caractere aparece e preenche a diagonal principal de F. Caso exista algum símbolo não previamente mapeado, esse símbolo é inserido na matriz. Esse passo considera que todos os caracteres foram corretamente reconhecidos. O restante da matriz é preenchido com valor zero;

4. Para cada erro de OCR identificado, a matriz F é atualizada. Um erro é visto da forma "de $\rightarrow$ para", indicando que o caractere (ou conjunto) "de" foi reconhecido como "para". Caso "de" e/ou "para" seja composto por mais de um caractere e esse símbolo não esteja mapeado na matriz F, ele é adicionado ao alfabeto Σ' e inserido na matriz na posição $f_{i,j}$, onde i e j são as posições dos símbolos "de" e "para" na matriz, respectivamente. A inserção e atualização da matriz verifica os caracteres previamente mapeados nos passos anteriores, subtraindo suas ocorrências deles.

Ao final desse processo é obtida uma matriz com os acertos e erros do OCR mapeados, para o conjunto de documentos de treinamento utilizado. No entanto, para que essa matriz possa ser utilizada no próximo passo, ela precisa ter todos os valores maiores do que zero. Dessa forma, um novo processamento é feito na matriz, da seguinte maneira:

Percorrer todas as células da matriz F

Se $f_{i,j} = 0$ e $i = j$, então $f_{i,j} := 1$

Senão $f_{i,j} := 0.00001$

Tal ajuste é necessário para que mapeamentos não encontrados durante o treinamento tenham uma contagem de frequência baixa e, consequentemente, obtenham pontuação mais baixa se comparada aos acertos.

A matriz de frequência construída foi uma matriz de ordem 234, o que significa que 234 símbolos foram mapeados, com no máximo 4 combinações de caracteres por símbolo, de acordo com a lista de erros de OCR utilizada em sua montagem.

7.3.6 Matriz de Substituição

A montagem da matriz de substituição segue o descrito na Seção 4.4.1. Ela possui a mesma ordem da matriz de frequência utilizada como entrada. As Figuras 7.3 e 7.4 ilustram trechos da matriz de substituição gerada pelo treinamento.

Cada célula dessa matriz representa um *score* que será utilizado pelo SOCRATEXT no cálculo da similaridade entre cadeias de caracteres. É importante notar que a diagonal principal apresenta valores positivos, indicando que o OCR utilizado acertou na maioria das vezes na identificação correta desses caracteres. Se forem observados caracteres que possuem formas semelhantes como "l" e "i", é possível notar que o reconhecimento do

	i	j	k	l	m	n	o	p	q	r	s	t
i	12.994	-77.236	-72.925	-88.103	-89.872	-90.659	-93.784	-87.578	-83.236	-91.863	-92.377	-90.447
j	-0.195	27.254	-58.451	-73.630	-75.399	-76.186	-79.311	-73.105	-68.762	-77.390	-77.904	-75.974
k	-72.925	-58.451	31.779	-69.318	-71.087	-71.874	-74.999	-68.793	-64.451	-73.079	-73.592	-71.662
l	0.085	-73.630	-69.318	16.346	-86.266	-87.052	-90.177	-83.971	-79.629	-88.257	-88.771	-16.122
m	-89.872	-75.399	-71.087	-86.266	14.821	-18.778	-91.946	-85.741	-81.398	-90.026	-90.540	-88.610
n	-90.659	-76.186	-71.874	-87.052	-88.821	14.045	-92.733	-86.527	-82.185	-90.813	-91.326	-89.397
o	-93.784	-79.311	-74.999	-90.177	-91.946	-92.733	10.914	-89.652	-85.310	-93.938	-94.451	-92.522
p	-87.578	-73.105	-68.793	-83.971	-85.741	-86.527	-89.652	17.126	-79.104	-87.732	-88.245	-86.316
q	-83.236	-68.762	-64.451	-79.629	-81.398	-82.185	-85.310	-79.104	21.468	-83.390	-83.903	-81.973
r	-91.863	-77.390	-73.079	-88.257	-90.026	-90.813	-93.938	-87.732	-83.390	12.841	-92.531	-90.601
s	-92.377	-77.904	-73.592	-88.771	-90.540	-91.326	-94.451	-88.245	-83.903	-92.531	12.327	-91.115
t	-90.447	-75.974	-71.662	-86.841	-88.610	-89.397	-92.522	-86.316	-81.973	-90.601	-91.115	14.257

Figura 7.3: Trecho da matriz de substituição PAM1 criada.

	fi	fi	fi	if	fl	fl	A	rm	nn	ç	1	S
fi	27.569	62.182	-33.456	-31.635	-26.797	-1.929	-1.929	-35.372	-1.929	-1.929	-1.929	-1.929
fi	-1.929	64.641	1.291	3.112	7.949	32.817	32.817	-0.625	32.817	32.817	32.817	32.817
fi	-33.456	1.291	32.139	-28.415	-23.577	1.291	1.291	-32.152	1.291	1.291	1.291	1.291
if	-31.635	3.112	-28.415	33.858	-21.756	3.112	3.112	-30.331	3.112	3.112	3.112	3.112
fl	-26.797	7.949	-23.577	-21.756	36.050	63.264	7.949	-25.493	7.949	7.949	7.949	7.949
fl	-1.929	32.817	1.291	3.112	7.949	64.641	32.817	-0.625	32.817	32.817	32.817	32.817
A	-1.929	32.817	1.291	3.112	7.949	32.817	64.641	-0.625	32.817	32.817	32.817	32.817
rm	-35.372	-0.625	-32.152	-30.331	-25.493	-0.625	-0.625	31.527	53.354	-0.625	-0.625	-0.625
nn	-1.929	32.817	1.291	3.112	7.949	32.817	32.817	-0.625	64.641	32.817	32.817	32.817
ç	-1.929	32.817	1.291	3.112	7.949	32.817	32.817	-0.625	32.817	64.641	32.817	32.817
1	-1.929	32.817	1.291	3.112	7.949	32.817	32.817	-0.625	32.817	32.817	64.641	32.817
S	-1.929	32.817	1.291	3.112	7.949	32.817	32.817	-0.625	32.817	32.817	32.817	64.641

Figura 7.4: Outro trecho da matriz de substituição PAM1 criada mostrando símbolos compostos por mais de um caractere.

caractere “l” como “i” apresenta um *score* maior do que o contrário, indicando que quando o OCR errou ele trocou mais frequentemente o “l” por “i”.

Em geral, matrizes PAM1 são utilizadas para comparação de sequências com alta similaridade, enquanto matrizes como PAM120 e PAM250 são utilizadas para sequências que apresentam menor similaridade. Dessa forma, neste trabalho foram também geradas as matrizes PAM120 e PAM250 a partir da matriz PAM1 criada, utilizando a Equação 4.17.

7.4 TESTES

Os testes têm como objetivos validar o algoritmo criado, e também compará-lo a outro algoritmo comumente utilizado para buscas em textos aceitando erros. A próxima seção mostra como o conjunto de arquivos de teste foi gerado e a seção seguinte a forma como os testes foram conduzidos.

7.4.1 Geração do Conjunto de Arquivos de Teste

Esta etapa consiste na utilização dos textos presentes no *corpus* de testes para geração de imagens degradadas que serão utilizadas como entrada para o Tesseract, gerando arquivos em texto pleno, onde as buscas serão realizadas.

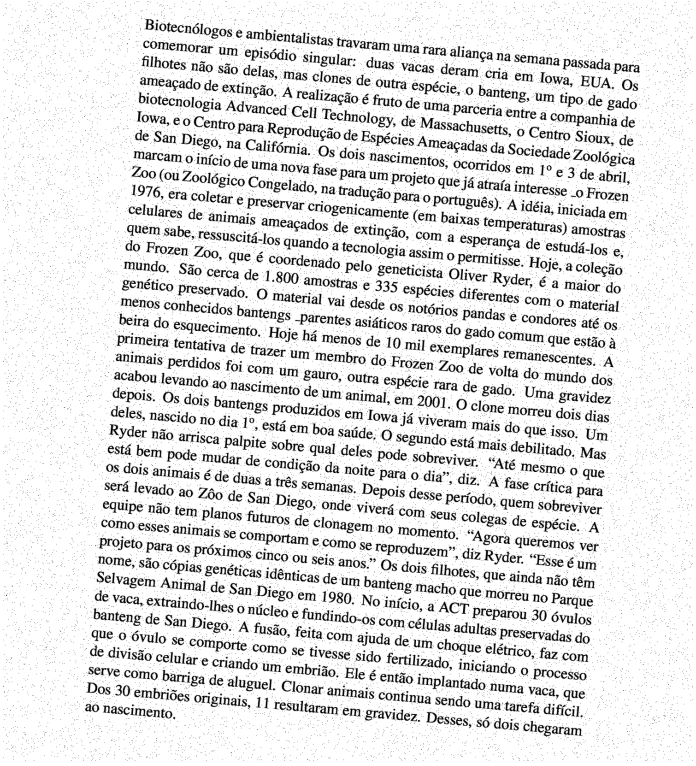
Para a aplicação das degradações de forma a simular documentos encontrados no mundo real, primeiramente foram gerados documentos da mesma forma que os gerados na etapa de treinamento (Seção 7.3.1), exceto que a origem dos textos é o *corpus* de testes e não o de treinamento, gerando conjuntos de imagens de documentos livres de degradações nas fontes “Times New Roman” e “Arial”. Os 5 primeiros arquivos da Tabela 7.1 foram formatados em “Arial” e os 5 restantes em “Times New Roman”.

Nessas imagens, as degradações foram aplicadas de duas maneiras. Na primeira delas, as degradações de rotação, resolução e sal-e-pimenta foram aplicadas separadamente em cada imagem. Na segunda delas foram gerados documentos com maior grau de degradação, sendo aplicadas as três degradações cumulativamente em uma mesma imagem. A Tabela 7.5 mostra o grau e forma de aplicações das degradações, bem como em quais fontes estavam os caracteres presentes nos documentos.

As Figuras 7.5 e 7.6 mostram, respectivamente, um documento do conjunto Degrad-TesteCum e seu texto gerado a partir do OCR.

Tabela 7.5: Degradações aplicadas nos dois conjuntos de testes gerados.

Identificação	Degradação	Fator	Forma de Aplicação	Fontes
DegradTesteSep	Rotação	5	Separadamente	Times New Roman e Arial
	Resolução	150 dpi		
	Sal-e-pimenta	3%		
DegradTesteCum	Rotação	5	Cumulativamente	Times New Roman e Arial
	Resolução	150 dpi		
	Sal-e-pimenta	3%		



Biotecnólogos e ambientalistas travaram uma rara aliança na semana passada para comemorar um episódio singular: duas vacas deram cria em Iowa, EUA. Os filhotes não são delas, mas clones de outra espécie, o banteng, um tipo de gado ameaçado de extinção. A realização é fruto de uma parceria entre a companhia de biotecnologia Advanced Cell Technology, de Massachusetts, o Centro Sioux, de Iowa, e o Centro para Reprodução de Espécies Ameaçadas da Sociedade Zoológica de San Diego, na Califórnia. Os dois nascimentos, ocorridos em 1º e 3 de abril, marcam o início de uma nova fase para um projeto que já atraía interesse o Frozen Zoo (ou Zoológico Congelado, na tradução para o português). A idéia, iniciada em 1976, era coletar e preservar criogenicamente (em baixas temperaturas) amostras celulares de animais ameaçados de extinção, com a esperança de estudá-los e, quem sabe, ressuscitá-los quando a tecnologia assim o permitisse. Hoje, a coleção do Frozen Zoo, que é coordenado pelo geneticista Oliver Ryder, é a maior do mundo. São cerca de 1.800 amostras e 335 espécies diferentes com o material genético preservado. O material vai desde os notórios pandas e condores até os menos conhecidos bantengs parentes asiáticos raros do gado comum que estão à beira do esquecimento. Hoje há menos de 10 mil exemplares remanescentes. A primeira tentativa de trazer um membro do Frozen Zoo de volta do mundo dos animais perdidos foi com um gauro, outra espécie rara de gado. Uma gravidez acabou levando ao nascimento de um animal, em 2001. O clone morreu dois dias depois. Os dois bantengs produzidos em Iowa já viveram mais do que isso. Um deles, nascido no dia 1º, está em boa saúde. O segundo está mais debilitado. Mas Ryder não arrisca palpite sobre qual deles pode sobreviver. "Até mesmo o que está bem pode mudar de condição da noite para o dia", diz. A fase crítica para os dois animais é de duas a três semanas. Depois desse período, quem sobreviver será levado ao Zôo de San Diego, onde viverá com seus colegas de espécie. A equipe não tem planos futuros de clonagem no momento. "Agora queremos ver como esses animais se comportam e como se reproduzem", diz Ryder. "Esse é um projeto para os próximos cinco ou seis anos." Os dois filhotes, que ainda não têm nome, são cópias genéticas idênticas de um banteng macho que morreu no Parque Selvagem Animal de San Diego em 1980. No início, a ACT preparou 30 óvulos de vaca, extraindo-lhes o núcleo e fundindo-os com células adultas preservadas do banteng de San Diego. A fusão, feita com ajuda de um choque elétrico, faz com que o óvulo se comporte como se tivesse sido fertilizado, iniciando o processo de divisão celular e criando um embrião. Ele é então implantado numa vaca, que serve como barriga de aluguel. Clonar animais continua sendo uma tarefa difícil. Dos 30 embriões originais, 11 resultaram em gravidez. Desses, só dois chegaram ao nascimento.

Figura 7.5: Exemplo de um documento do conjunto DegradTesteCum

.

Figura 7.6: Texto gerado pelo OCR a partir do documento mostrado na Figura 7.5.

7.4.2 Realização dos Testes de Busca

Os testes de busca têm como finalidade a verificação da aplicabilidade do algoritmo SOCRATEXT nas buscas de palavras-chaves em textos reconhecidos por OCR. Além disso, deseja-se comparar os resultados obtidos com a estratégia utilizada especialmente no *software* forense FTK, que permite a busca aproximada de palavras-chaves em texto reconhecidos por OCR. A busca aproximada implementada no FTK utiliza a distância de edição de *Levenshtein*, explicada no Capítulo 4.

Dessa forma, para as buscas que utilizam o algoritmo de distância de edição utilizou-se diversas distâncias de edições (1, 2, 3, ...) para variar os resultados das buscas. Para o caso do algoritmo SOCRATEXT, utilizou-se a medida *E-value*, descrita nos Capítulos 4 e 5, e o número máximo de combinação de símbolos (α), descrito no Capítulo 5. Todos os resultados de busca apresentados para o algoritmo SOCRATEXT foram obtidos a partir da matriz PAM1, descrita na Seção 7.3. Em experimentos preliminares também foram utilizadas a matrizes PAM120 e PAM250. No entanto, os resultados foram consideravelmente inferiores aos obtidos com a matriz PAM1 e por isso essas matrizes não foram utilizadas nos resultados aqui apresentados. O mau desempenho do SOCRATEXT utilizando matrizes PAM120 e PAM250 pode ser explicado considerando-se que essas matrizes são geralmente utilizadas para comparação de proteínas com maior distância evolucionária, ou seja, com maior grau de diferenças para o caso de cadeias de caracteres. As cadeias de caracteres

geradas pelo OCR em comparação com as cadeias originais podem ser consideradas cadeias próximas e por isso a matriz PAM1 apresentou resultados muito superiores.

Para a identificação dos termos que seriam buscados, para cada arquivo de texto original do *corpus* de teste, foram obtidas palavras ou frases únicas dentro do texto utilizando espaço como delimitador de palavras e esses termos são buscados no texto reconhecido por OCR. Foram definidos 3 cenários distintos de teste:

- **Conjunto de Teste 1**: busca no conjunto **"DegradTesteSep"** por palavras ≥ 6 caracteres, totalizando 4.215 palavras. Neste conjunto foram feitas duas formas de busca baseadas em distância de edição. A primeira delas considera distância de edição estática, ou seja, para qualquer tamanho de palavra sendo buscada utiliza-se apenas uma distância de edição. Na segunda, utilizou-se distância de edição relativa, com a distância de edição variando conforme o tamanho da palavra sendo buscada. As variações aplicadas estão exibidas na Tabela 7.6;
- **Conjunto de Teste 2**: busca no conjunto **"DegradTesteSep"** por frases ≥ 16 caracteres contendo exatamente 3 palavras, totalizando 2.622 termos;
- **Conjunto de Teste 3**: busca no conjunto **"DegradTesteCum"** por frases ≥ 16 caracteres contendo exatamente 3 palavras, totalizando 874 termos.

Tabela 7.6: Identificação dos grupos de distância de edição relativa. A distância de edição varia de 1 a 3 de acordo com o tamanho da palavra sendo buscada.

ID Grupo D. E. Relativa	D.E. = 1	D. E. = 2	D. E. = 3
1	6 a 7 caract.	8 a 10 caract.	$\geq$ 11 caract.
2	6 a 8 caract.	9 a 11 caract.	$\geq$ 12 caract.
3	6 a 9 caract.	10 a 12 caract.	$\geq$ 13 caract.
4	6 a 10 caract.	11 a 13 caract.	$\geq$ 14 caract.
5	6 a 12 caract.	13 a 15 caract.	$\geq$ 16 caract.
6	6 a 14 caract.	15 a 16 caract.	$\geq$ 17 caract.

Devido à grande quantidade de buscas a serem realizadas, a análise dos resultados manualmente é uma tarefa árdua. No entanto, embora seja esperado somente um *hit* como retorno em cada busca, mesmo que a busca retorne apenas um *hit*, o resultado pode estar incorreto, pois a palavra retornada pode ser uma outra que a busca aproximada considerou

que fosse a palavra procurada. Para que seja possível identificar quando o resultado retornado está correto, é necessário analisar o contexto em que o *hit* se encontra. Para isso, foi definido como contexto os 30 caracteres imediatamente anteriores ou posteriores ao termo encontrado. Assim, cada termo de busca tem associado a ele um identificador de contexto que é comparado ao identificador do *hit* usando um cálculo de distância de Levenshtein entre eles. O *hit* é considerado acerto se a distância de edição entre o contexto anterior ou posterior e os respectivos do termo de busca for menor ou igual a 18. Esse valor foi definido por meio de experimentos e mostrou ser um valor suficiente para o objetivo. A Figura 7.7 ilustra essa estratégia, onde T_1 é um trecho do texto reconhecido por OCR, T_2 um trecho do texto original e "tanque" a palavra que está sendo buscada. Nesse exemplo, a distância de edição entre os 30 caracteres imediatamente anteriores à palavra "tanque" dos textos T_1 e T_2 é de 19 e para os 30 caracteres imediatamente posteriores é de 10. Dessa forma, o palavra "tanque" encontrada seria considerada dentro do contexto devido à distância de edição dos 30 caracteres imediatamente posteriores a ela ser menor ou igual a 18.

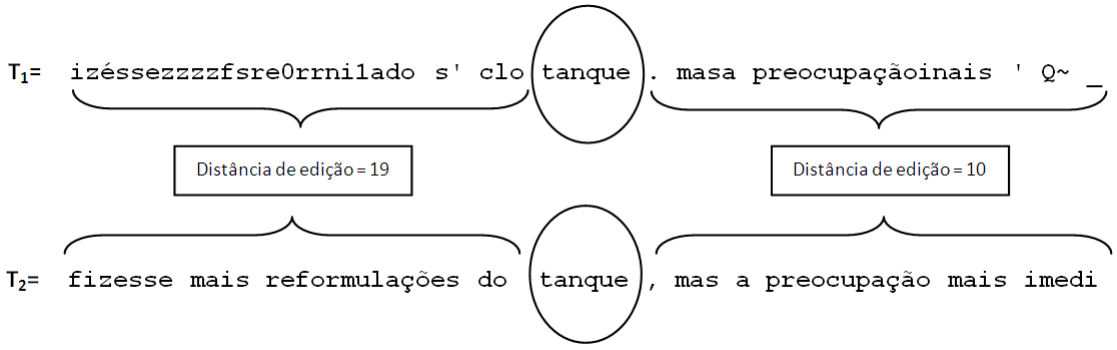


Figura 7.7: Exemplo de identificação de contexto utilizando os 30 caracteres imediatamente anteriores e posteriores de uma palavra.

.

Por ser tratar de busca aproximada, também foi necessário definir uma estratégia para que dois candidatos a *hits* não figurassem como resposta quando possuíssem posições em comum no texto. Por exemplo, no caso da busca do termo "droga" no texto "Alto índice de drogados no Brasil" permitindo uma distância de edição de 1, teria-se vários candidatos a *hits*: " droga", "drogad" e "droga". No entanto, a busca deveria retornar somente um deles: o que apresenta maior semelhança com o termo buscado. Assim, no caso de existência de mais de um candidato a *hit*, foram considerados apenas os que não apresentassem

	Relevantes	Irrelevantes
Recuperados	A	B
Não Recuperados	C	D

Figura 7.8: Tipos de resultados que podem ser obtidos na recuperação de informação.

intersecção de posições com os demais, sendo eliminados os que apresentassem menores *scores*, para o algoritmo SOCRATEXT, ou o de maiores distâncias de edição, para o algoritmo de distância de edição.

Para avaliar o desempenho na área de Recuperação da Informação, duas medidas frequentemente utilizadas são: ***recall*** e ***precision***. O *recall* é a fração dos termos relevantes existentes que foram retornados, enquanto a *precision* é a fração dos *hits* que são relevantes [Baeza-Yates e Ribeiro-Neto, 1999].

A Figura 7.8 ilustra os possíveis resultados que podem ser obtidos na recuperação de informação e as Equações 7.1 e 7.2 as fórmulas para cálculo do *recall* e *precision*, respectivamente.

$$recall = \frac{A}{A \cup C} \tag{7.1}$$

$$precision = \frac{A}{A \cup B} \tag{7.2}$$

Algumas vezes é necessário unir as medidas de *recall* e *precision* em apenas uma medida, o que pode ser feito a partir da medida conhecida como *F-measure* [van Rijsbergen, 1979], calculada pela equação 7.3.

$$F_\beta = \frac{(\beta^2 + 1) \cdot recall \cdot precison}{\beta^2 \cdot precision + recall}, \quad \beta \in [0, \infty) \tag{7.3}$$

Nessa fórmula, é possível estabelecer pesos diferenciados para o *recall* e a *precision*, de acordo com a necessidade. Quando $\beta = 1$ é dado peso igual aos dois valores, para $\beta > 1$ é dado maior peso ao *recall* e para $\beta < 1$ é dado maior peso para a *precision*. No contexto da Informática Forense a medida de *recall* tem importância maior do que a *precision* na maioria dos casos, pois deixar de recuperar um documento importante pode ter consequências graves na investigação.

Em todo o conjunto de teste aplicado neste trabalho são calculados os valores de *recall*, *precision* e *F-measure* para $\beta = 2$, indicando, nesse último valor, que o *recall* tem o dobro de importância da *precision* para fins de análise.

As Tabelas 7.7, 7.8 e 7.9 exibem os resultados obtidos para o **Conjunto de Teste 1**, para os algoritmos SOCRATEXT e distância de edição, respectivamente. As Tabelas 7.10 e 7.11 exibem os resultados obtidos para o **Conjunto de Teste 2**, para os algoritmos SOCRATEXT e distância de edição, respectivamente. As Tabelas 7.12 e 7.13 exibem os resultados obtidos para o **Conjunto de Teste 3**, para os algoritmos SOCRATEXT e distância de edição, respectivamente.

As colunas das Tabelas têm o seguinte significado:

- **A**: número de termos de buscas relevantes (corretos) que foram recuperados;
- **B**: número de termos de buscas irrelevantes (incorretos) que foram recuperados;
- **A+C**: número total de termos de buscas relevantes;
- **Recall.**: fração dos termos relevantes existentes que foram retornados, conforme Equação 7.1;
- **Prec.**: fração dos *hits* que são relevantes, conforme Equação 7.2;
- $\mathbf{F_2}$: valor de *F-measure* para $\beta = 2$, conforme Equação 7.3;
- **E-value**: número provável de sequências iguais à encontrada que poderiam ter sido encontradas ao acaso, conforme definição na Seção 4.4;
- α: número máximo de combinações de símbolos aceitas no algoritmo SOCRATEXT;
- **D. E.**: distância de edição;
- **D. E. Rel.**: distância de edição relativa. A distância de edição varia conforme o tamanho da palavra sendo buscada.

A partir dos valores contidos nas Tabelas 7.7 a 7.13 foram elaborados 7 gráficos que permitem visualizar a evolução da *F-measure* para $\beta = 2$ quando variam-se o *E-value* e o α no algoritmo SOCRATEXT ou a distância de edição no algoritmo de distância de edição de Levenshtein. Tais gráficos estão exibidos nas Figuras 7.9 a 7.15. Esses valores foram definidos a partir de experimentos preliminares, de forma que pudesse ser possível alcançar um valor ótimo e que pudessem indicar tendências dos valores de *recall* e *precision*.

Tabela 7.7: Resultados das buscas com o algoritmo SOCRATEXT por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep**. A célula destacada indica o melhor resultado obtido para F_2.

E-value	α	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1,00E-10	1	2.068	384	4.215	0,491	0,843	0,535
1,00E-10	2	2.107	513	4.215	0,500	0,804	0,541
1,00E-10	3	2.107	519	4.215	0,500	0,802	0,541
1,00E-10	4	2.107	523	4.215	0,500	0,801	0,541
1,00E-05	1	4.179	1.838	4.215	0,981	0,751	0,925
1,00E-05	2	4.179	1.838	4.215	0,991	0,695	0,913
1,00E-05	3	4.179	1.891	4.215	0,991	0,688	0,911
1,00E-05	4	4.179	1.968	4.215	0,991	0,680	0,908
1,00E-01	1	4.183	2.508	4.215	0,992	0,625	0,888
1,00E-01	2	4.201	3.961	4.215	0,997	0,515	0,839
1,00E-01	3	4.201	4.316	4.215	0,997	0,493	0,828
1,00E-01	4	4.201	4.772	4.215	0,997	0,468	0,813

Tabela 7.8: Resultados das buscas com o algoritmo de distância de edição por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep**. A célula destacada indica o melhor resultado obtido para F_2.

D. E.	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1	4.072	1.648	4.215	0,966	0,712	0,902
2	4.184	4.332	4.215	0,993	0,491	0,824
3	4.194	26.383	4.215	0,995	0,137	0,442
4	4.205	153.206	4.215	0,998	0,027	0,121

Tabela 7.9: Resultados das buscas com o algoritmo de distância de edição por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep** utilizando as distâncias de edição relativa de acordo com a Tabela 7.6. A célula destacada indica o melhor resultado obtido para F_2.

Grupo de D. E. Rel.	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1	4.159	2.326	4.215	0,987	0,641	0,891
2	4.140	2.045	4.215	0,982	0,669	0,898
3	4.123	1.895	4.215	0,978	0,685	0,901
4	4.112	1.805	4.215	0,976	0,695	0,903
5	4.094	1.665	4.215	0,971	0,711	0,905
6	4.078	1.651	4.215	0,967	0,712	0,903

Tabela 7.10: Resultados das buscas com o algoritmo SOCRATEXT por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteSep**. A célula destacada indica o melhor resultado obtido para F_2.

E-value	α	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1,00E-25	1	1.436	37	2.622	0,548	0,975	0,600
1,00E-25	2	2.150	65	2.622	0,820	0,971	0,846
1,00E-25	3	2.152	65	2.622	0,821	0,971	0,847
1,00E-25	4	2.152	65	2.622	0,821	0,971	0,847
1,00E-18	1	2.479	83	2.622	0,945	0,968	0,956
1,00E-18	2	2.582	158	2.622	0,985	0,942	0,976
1,00E-18	3	2.582	163	2.622	0,985	0,941	0,976
1,00E-18	4	2.583	174	2.622	0,985	0,937	0,975
1,00E-10	1	2.598	178	2.622	0,991	0,936	0,979
1,00E-10	2	2.613	480	2.622	0,997	0,845	0,962
1,00E-10	3	2.613	567	2.622	0,997	0,822	0,901
1,00E-10	4	2.613	681	2.622	0,997	0,793	0,883

Tabela 7.11: Resultados das buscas com o algoritmo de distância de edição por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteSep**. A célula destacada indica o melhor resultado obtido para F_2.

D. E.	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1	2.218	49	2.622	0,846	0,978	0,869
2	2.413	85	2.622	0,920	0,966	0,929
3	2.457	143	2.622	0,937	0,945	0,939
4	2.478	307	2.622	0,945	0,890	0,933
5	2.499	535	2.622	0,953	0,824	0,924

Tabela 7.12: Resultados das buscas com o algoritmo SOCRATEXT por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteCum**. A célula destacada indica o melhor resultado obtido para F_2.

E-value	α	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1,00E-25	1	521	46	874	0,596	0,919	0,641
1,00E-25	2	658	205	874	0,753	0,762	0,755
1,00E-25	3	660	272	874	0,755	0,708	0,745
1,00E-25	4	663	346	874	0,759	0,657	0,736
1,00E-20	1	717	83	874	0,820	0,896	0,834
1,00E-20	2	803	398	874	0,919	0,669	0,855
1,00E-20	3	804	523	874	0,920	0,606	0,834
1,00E-20	4	804	660	874	0,920	0,549	0,810
1,00E-10	1	842	370	874	0,963	0,695	0,894
1,00E-10	2	858	1.989	874	0,982	0,301	0,676
1,00E-10	3	861	3.043	874	0,985	0,221	0,582
1,00E-10	4	862	4.435	874	0,986	0,163	0,490

Tabela 7.13: Resultados das buscas com o algoritmo de distância de edição por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteCum**. A célula destacada indica o melhor resultado obtido para F_2.

D. E.	**A**	**B**	**A+C**	**Recall**	**Prec.**	**F_2**
1	366	15	874	0,419	0,961	0,472
2	512	21	874	0,586	0,961	0,635
3	602	37	874	0,689	0,942	0,728
4	670	77	874	0,767	0,897	0,790
5	710	133	874	0,812	0,842	0,818
6	746	222	874	0,854	0,771	0,836
7	778	369	874	0,890	0,678	0,838
8	803	722	874	0,919	0,527	0,800
9	827	1.815	874	0,946	0,313	0,674
10	837	5.484	874	0,958	0,132	0,426

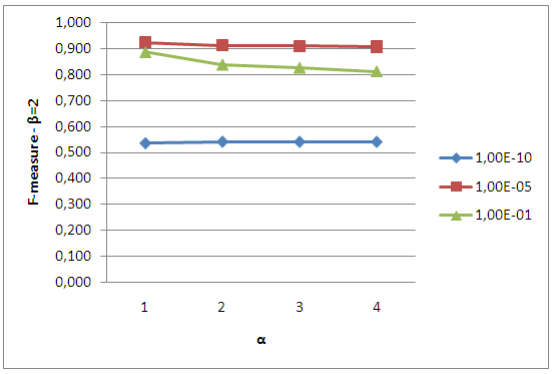


Figura 7.9: Gráfico com resultados das buscas com o algoritmo SOCRATEXT por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep**.

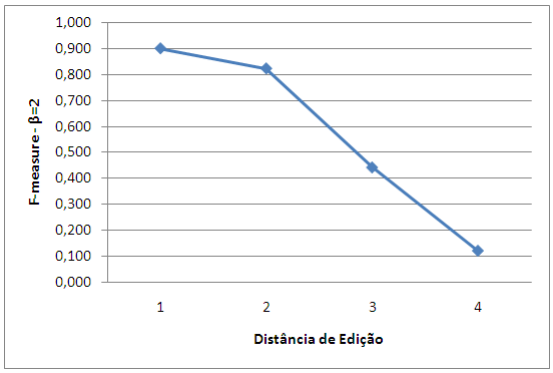


Figura 7.10: Gráfico com resultados das buscas com o algoritmo de distância de edição por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep**.

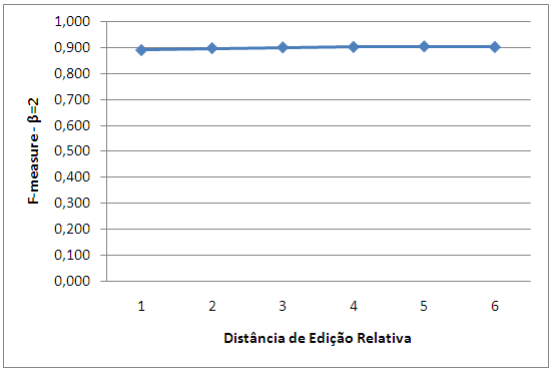


Figura 7.11: Gráfico com resultados das buscas com o algoritmo de distância de edição por palavras de comprimento ≥ 6 caracteres no conjunto **DegradTesteSep** utilizando as distâncias de edição relativa de acordo com a Tabela 7.6.

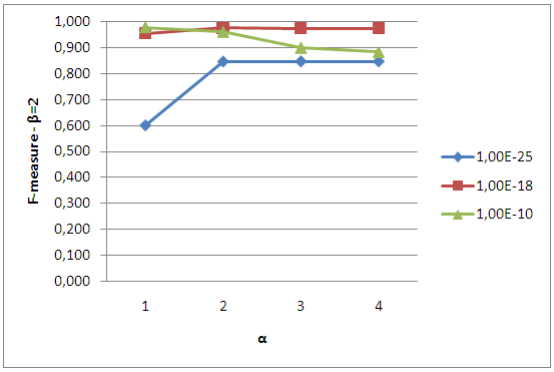


Figura 7.12: Resultados das buscas com o algoritmo SOCRATEXT por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteSep**.

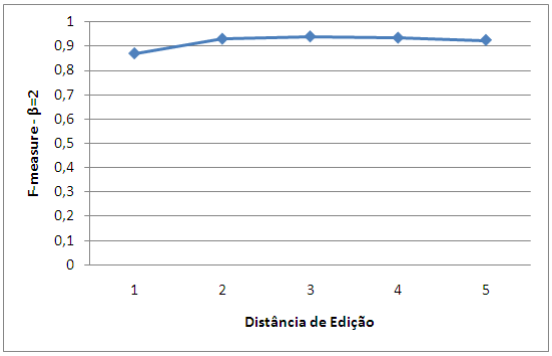


Figura 7.13: Gráfico com resultados das buscas com o algoritmo de distância de edição por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteSep**.

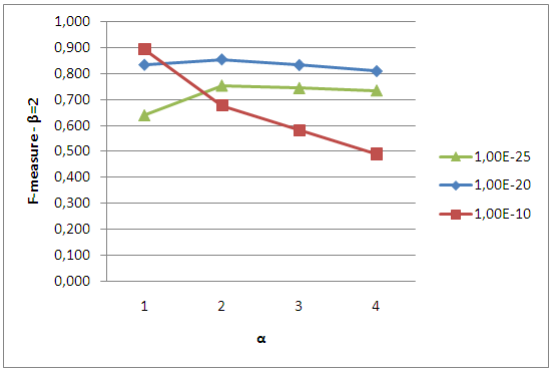


Figura 7.14: Gráfico com resultados das buscas com o algoritmo SOCRATEXT por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteCum**.

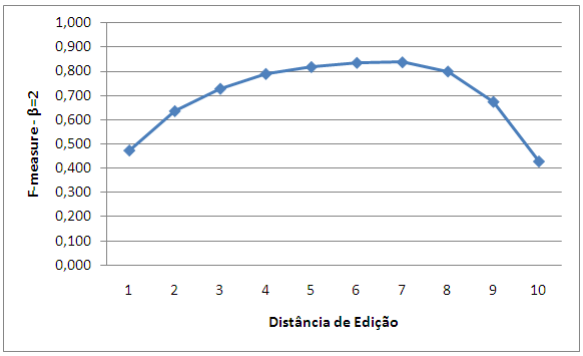


Figura 7.15: Gráfico com resultados das buscas com o algoritmo de distância de edição por frases de comprimento ≥ 16 caracteres contendo 3 palavras no conjunto **DegradTesteCum**.

7.5 ANÁLISE DOS RESULTADOS

Os experimentos realizados, cujos resultados estão sumarizados nas Tabelas 7.7 a 7.13 e nas Figuras 7.9 a 7.15, procuraram identificar qual a influência da variação do *E-value* e do α no algoritmo SOCRATEXT e a distância de edição no algoritmo de distância de edição de Levenshtein. O desempenho é avaliado sob o ponto de vista de *F-measure* pelo fato de combinar duas medidas em apenas uma, além de possibilitar dar maior importância ao *recall* em relação à *precision*.

No primeiro experimento **(Conjunto de Teste 1)**, o foco estava em testar a busca de palavras isoladas. No SOCRATEXT os melhores resultados foram obtidos com *E-value* $1E-5$ e, especificamente, quando $\alpha = 1$. Nesse caso a combinação de símbolos pelo algoritmo não foi capaz de trazer melhoras nos resultados, pois à medida que o α variou de 1 a 4, o valor de *F-measure* foi decaindo, embora ainda se mantivessem superiores aos resultados apresentados pelo algoritmo de distância de edição. Com o *E-value* sendo $1E-1$ também foi observado dimuição do desempenho conforme se aumentava o valor de α, o que não ocorreu com *E-value* $1E-10$. Esse comportamento pode ser observado no gráfico exibido na Figura 7.9. Utilizando o algoritmo de distância de edição não foi possível alcançar resultados superiores aos do algoritmo SOCRATEXT nem com a utilização de distância de edição estática e nem com a relativa. Para a distância de edição estática o melhor resultado foi alcançado para a distância de edição 1 e foi decaindo conforme se aumentava esse valor. Tal comportamento pode ser explicado porque conforme se aumenta a distância de edição permitida se perde muito em *precision*, principalmente com palavras

pequenas onde percentualmente a permissão de uma diferença a mais tem maior influência. Isso acarreta em recuperar muitos falsos positivos. Já o ganho em *recall* não foi tão grande com o aumento da distância de edição permitida, sugerindo que a maioria das palavras reconhecidas pelo OCR possuía no máximo um caractere errado. Com o intuito de se levar em consideração o tamanho da palavra para determinação da distância de edição, foram conduzidos os experimentos que seguiam a Tabela 7.6. Os resultados mostraram bastante estabilidade nas combinações utilizadas, tendo superado, no melhor caso, a distância de edição estática em apenas 0,3%. Os gráficos exibidos nas Figuras 7.10 e 7.11 ilustram o resultado desse experimento.

No segundo experimento **(Conjunto de Teste 2)**, foi simulado o comportamento nas buscas de frases contendo 3 palavras totalizando pelo menos 16 caracteres. O algoritmo SOCRATEXT alcançou *F-measure* 97,9% para *E-value* $1E-10$ e $\alpha = 1$. Para esse *E-value* o aumento de α diminuiu o desempenho, enquanto para os demais *E-values* testados ($1E-18$ e $1E-25$) o desempenho melhorou conforme se aumentava o α. Nesses dois casos o aumento de α propiciou melhoras siginificativas na *recall* sem alterar muito a *precision*, principalmente na transição do valor de α de 1 para 2. Para o valor de *E-value* $1E-10$ o aumento do α alterou principalmente a *precision*, diminuindo de 93,6% para $\alpha = 1$ para 88,3% para $\alpha = 4$. O melhor desempenho obtido pelo algoritmo de distância de edição para o segundo experimento foi de 93,9% para a distância de edição 3, 4% abaixo do melhor desempenho do SOCRATEXT. Os gráficos exibidos nas Figuras 7.12 e 7.13 ilustram os resultados desse experimento.

No último experimento **(Conjunto de Teste 3)** buscou-se avaliar o comportamento dos algoritmos na busca em textos com elevado grau de degradação. Como era de se esperar, o resultado obtido com ambos os algoritmos foram inferiores aos obtidos nos experimentos anteriores. No entanto, a diferença de desempenho entre os algoritmos foi a maior verificada, com 89,4% para o SOCRATEXT e 83,8% para a distância de edição. Isso sugere que o algoritmo SOCRATEXT é menos sensitivo aos erros, já que se baseia na ideia de mapear os erros comuns de ocorrerem e por isso não os penaliza muito quando ocorrem. Embora o melhor desempenho do SOCRATEXT tenha sido abaixo de 90%, a *recall* foi de 96,3%, o que significa que mesmo em textos obtidos de imagens bastante degradadas é possível recuperar grande parte dos textos de interesse, sem penalizar em excesso a *precision*, que se manteve em torno de 70%. A análise do gráfico exibido na Figura 7.14 permite concluir que o desempenho diminui bastante com o aumento de α

quando o *E-value* é $1E-10$ enquanto para outros *E-values* houve melhoras quando o α passou de 1 para 2.

De uma forma geral, os melhores resultados foram obtidos quando o α utilizado foi 1, e, quando isso não ocorreu, o valor 2 mostrou melhores resultados. Isso indica que combinações de símbolos acima de 2 não propiciaram melhorias no desempenho do SOCRATEXT para a configuração de imagens, OCR e matriz de substituição utilizadas.

8 CONCLUSÃO

Este trabalho teve como principais contribuições a criação e validação de um algoritmo adaptado da programação dinâmica que foi denominado SOCRATEXT. Ele permite a realização de buscas em textos reconhecidos via OCR com maior grau de sucesso quando comparado ao algoritmo de distância de edição de Levenshtein. Os resultados obtidos, especialmente no contexto da Informática Forense, mostram que a obtenção de textos por OCR para permitir busca de palavra-chave é viável e pode alcançar melhores resultados utilizando o algoritmo SOCRATEXT no lugar da distância de edição de Levenshtein, que é o algoritmo mais utilizado para buscas permitindo erros nas ferramentas forenses. Os ganhos apresentados pelo algoritmo desenvolvido neste trabalho foram mais significativos comparativamente quando a busca de palavras-chaves foi realizada em imagens com maior grau de degradação, indicando sua robustez.

Outra contribuição foi comprovar que a utilização de técnicas desenvolvidas no âmbito da Biologia Computacional podem ser adaptadas e utilizadas com sucesso em outros contextos, como na busca de textos.

A matriz de substituição, embora tenha sido utilizada para mapeamento de erros comuns de OCR, poderia ser usada no mapeamento de outros conjuntos de erros, como erros comuns de digitação ou de escrita, por exemplo. Apenas alterando esses mapeamentos na matriz de substituição, o algoritmo SOCRATEXT, sem nenhuma modificação, continuaria sendo válido.

O trabalho também propôs e avaliou um método para analisar os efeitos dos erros de OCR na recuperação da informação em textos na língua inglesa e portuguesa usando este recurso do *software* forense *FTK*, sob diferentes níveis de degradação de imagens. O método, publicado em [Polastro e F. Almeida Jr., 2011] pode ser utilizado para avaliar outras versões de OCR, outro conjunto de imagens e textos em outra língua.

8.1 TRABALHOS FUTUROS

A criação de interface gráfica para o algoritmo desenvolvido seria de grande importância para permitir seu uso no dia-a-dia como uma forma de otimizar ou até de viabilizar a busca de palavras-chaves em imagens de documentos. Em especial, no caso de peritos da área de

Informática Forense, seu uso poderia ocorrer na busca de palavras-chaves em dispositivos de armazenamento computacional.

Outro ponto de interesse seria aplicar o algoritmo SOCRATEXT e o método de geração da matriz de substituição para outros contextos, diferente do OCR. Alguns contextos possíveis seriam: mapeamento de erros de digitação, erros na escrita, reconhecimento de fala, entre outros.

A matriz de substiuição gerada, bem como os textos onde foram realizadas as buscas, foram obtidos a partir de imagens degradadas sinteticamente. Poderiam ser realizados experimentos para verificar como a matriz criada neste trabalho e o algoritmo SOCRATEXT se comportam para imagens do mundo real e também com outras ferramentas de OCR.

A utilização da matriz de substituição criada a partir de textos em português poderia ser avaliada para buscar textos também em outras línguas e comparar com as buscas realizadas utilizando uma matriz gerada especificamente para a língua, de forma a avaliar se há necessidade de criação de novas matrizes de substituição de acordo com a língua de interesse. Uma abordagem que vem sendo muito utilizada na área de buscas é não diferenciar letras maiúsculas de minúsculas ou letras acentuadas de não acentuadas. Essa seria uma melhoria que pode ser incorporada ao SOCRATEXT.

REFERÊNCIAS BIBLIOGRÁFICAS

[Access Data, 2010] Access Data (2010). *FTK 3.2 User Guide*. John Wiley & Sons.

[Alves, 2003] Alves, N. F. (2003). Estratégias para melhoria do desempenho de ferramentas comerciais de reconhecimento Óptico de caracteres. Dissertação de mestrado, Universidade Federal de Pernambuco.

[Baeza-Yates e Ribeiro-Neto, 1999] Baeza-Yates, R. A. e Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Baird, 1995] Baird, H. S. (1995). Document image analysis. In *World Scientific*, pages 315–325, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Barbosa et al., 2010] Barbosa, A.; Cappi, J. e Jereissati, T. (2010). Pesquisa TIC domicílios 2010. `http://www.cetic.br/usuarios/tic/2010/apresentacao-tic-domicilios-2010.pdf`. [Online; Acessado em junho de 2011].

[Batista de Sousa, 2008] Batista de Sousa, G. (2008). Wmm - uma ferramenta de extração de vestígios deixados pelo windows live messenger. In *Third International Conference on Forensic Computer Science (ICoFCS)*, pages 104–111.

[Beebe e Clark, 2007] Beebe, N. L. e Clark, J. G. (2007). Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. *Digital Investigation*, 4:49–54.

[Bovik, 2005] Bovik, A. C. (2005). *Handbook of Image and Video Processing (Communications, Networking and Multimedia)*. Academic Press, Inc., Orlando, FL, USA.

[Bunting, 2008] Bunting, S. (2008). *EnCase computer forensics: the official EnCE : EnCase certified examiner study guide*. John Wiley & Sons.

[Carvey, 2004] Carvey, H. (2004). *Windows Forensics and Incident Recovery (The Addison-Wesley Microsoft Technology Series)*. Addison-Wesley Professional.

[Charras e Lecroq, 2004] Charras, C. e Lecroq, T. (2004). *Handbook of Exact String Matching Algorithms*. King's College Publications.

[Collins-Thompson et al., 2001] Collins-Thompson, K.; Schweizer, C. e Dumais, S. T. (2001). Improved string matching under noisy channel conditions. In *International Conference on Information and Knowledge Management*, pages 357–364.

[Collovini et al., 2007] Collovini, S.; Carbonel, T.; Fuchs, J. T.; Coelho, J. C.; Rino, L. e Vieira., R. (2007). Summ-it: Um corpus anotado com informações discursivas visando à sumarização automática. In *5o Workshop em Tecnologia da Informação e da Linguagem Humana (TIL'2007)*, Rio de Janeiro.

[Cormen et al., 2001] Cormen, T. H.; Stein, C.; Rivest, R. L. e Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

[Crochemore e Rytter, 2003] Crochemore, M. e Rytter, W. (2003). *Jewels of stringology: text algorithms*. World Scientific.

[DFRWS, 2001] DFRWS (2001). Digital forensic research workshop (2001) a road map for digital forensic research. In *Report from the First Digital Forensic Research Workshop (DFRWS)*.

[Eidhammer et al., 2004] Eidhammer, I.; Jonassen, I. e Taylor, W. R. (2004). *Protein bioinformatics - an algorithmic approach to sequence and structure analysis.* Wiley.

[Encase, 2011] Encase (2011). Guidance software. `http://www.guidancesoftware.com/`. [Online; Acessado em julho de 2011].

[Forensic Toolkit, 2011] Forensic Toolkit, F. (2011). Access data. `http://www.accessdata.com/`. [Online; Acessado em julho de 2011].

[Fujisawa, 2007] Fujisawa, H. (2007). A view on the past and future of character and document recognition. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 01*, pages 3–7, Washington, DC, USA. IEEE Computer Society.

[GNU, 2011] GNU (2011). GNU octave. `http://www.gnu.org/software/octave/`. [Online; Acessado em julho de 2011].

[Google, 2011] Google (2011). Google tradutor. `http://translate.google.com.br`. [Online; Acessado em julho de 2011].

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, EUA.

[Haenchen, 2010] Haenchen, S. L. (2010). Advanced text searching of electronic information related to forensic discovery. Dissertação de mestrado, University of Kansas.

[Hawking, 1996] Hawking, D. (1996). Document retrieval in OCR-scanned text. In *Proc. Sixth Parallel Computing Workshop*, pages P2–F.

[ImageMagick, 2011] ImageMagick (2011). Imagemagick. `http://www.imagemagick.org`. [Online; Acessado em julho de 2011].

[Jones e Pevzner, 2004] Jones, N. C. e Pevzner, P. A. (2004). *An Introduction to Bioinformatics Algorithms*. MIT Press.

[Kantor e Voorhees, 2000] Kantor, P. B. e Voorhees, E. M. (2000). The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval*, 2:165–176.

[Kanungo, 1996] Kanungo, T. (1996). *Document degradation models and a methodology for degradation model validation*. Tese de doutorado, University of Washington, Seattle, WA, USA. UMI Order No. GAX96-30083.

[Kanungo e Haralick, 1999] Kanungo, T. e Haralick, R. M. (1999). An automatic closed-loop methodology for generating character groundtruth for scanned documents. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:179–183.

[Kanungo et al., 1994] Kanungo, T.; Haralick, R. M. e Phillips, I. (1994). Non-linear local and global document degradation models. *International Journal of Imaging Systems and Technology - IJIST*, pages 220–30.

[Kaur et al., 2010] Kaur, S.; H.K.Kaura e Ojha, M. (2010). Development of optical character recognition software package for mobile phones. *IJCCT*, 1:3–5.

[Korf et al., 2003] Korf, I.; Yandell, M. e Bedell, J. (2003). *BLAST*. O'Reilly Series. O'Reilly.

[Lawless, 2003] Lawless, J. F. (2003). *Statistical Models and Methods for Lifetime Data*. John Wiley & Sons, 2nd edition.

[Lecoq et al., 2001] Lecoq, J. C.; Najman, L.; Gibot, O. e Éric Trupin (2001). Benchmarking commercial OCR engines for technical drawings indexing. In *International Conference on Document Analysis and Recognition*, pages 138–142.

[Levenshtein, 1965] Levenshtein, V. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17.

[Li et al., 1996] Li, Y.; Lopresti, D. P.; Nagy, G. e Tomkins, A. (1996). Validation of image defect models for optical character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:99–108.

[Locard, 1934] Locard, E. (1934). *Manuel de Technique Policière*. Ed. Payot.

[Mangnes, 2005] Mangnes, B. (2005). The use of levenshtein distance in computer forensics. Dissertção de mestrado, Gjøvik University College.

[Mello e Lins, 1999] Mello, C. e Lins, R. D. (1999). A comparative study on OCR tools. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 312–316.

[Melo, 2002] Melo, C. A. B. (2002). *Aspectos computacionais do processamento de imagens de documentos históricos*. Tese de doutorado, Universidade Federal de Pernambuco.

[Mota et al., 2008] Mota, C.; Santos, D.; Carvalho, P. e Oliveira, C. F. H. G. (2008). Apêndice h: Apresentação detalhada das colecções do segundo harem. In *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, pages 355–377.

[Nagy et al., 1999] Nagy, G.; Nartker, T. A.; Rice, S. V.; Nartkerb, T. A. e Ricec, S. V. (1999). Optical character recognition: An illustrated guide to the frontier. In *Storage and Retrieval for Image and Video Databases*.

[National Center for Biotechnology Information, 2011] National Center for Biotechnology Information, N. (2011). The statistics of sequence similarity score. `http:`

//www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html. [Online; Acessado em novembro de 2011].

[Navarro, 1998] Navarro, G. (1998). *Approximate Text Searching*. Tese de doutorado, Universidade do Chile, Santiago, Chile.

[Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM Comput. Surv.*, 33:31–88.

[Navarro e Raffinot, 2002] Navarro, G. e Raffinot, M. (2002). *Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences*. Cambridge University Press.

[Needleman e Wunsch, 1970] Needleman, S. B. e Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453.

[Petrovic e Bakke, 2008] Petrovic, S. e Bakke, S. (2008). Application of q-gram distance in digital forensic search. In *Proceedings of the 2nd international workshop on Computational Forensics*, IWCF '08, pages 159–168, Berlin, Heidelberg. Springer-Verlag.

[Phillips et al., 1996] Phillips, I.; Chanda, B. e Haralick, R. (1996). Uw-iii english/technical document image database. [CD-ROM].

[Phillips et al., 1993] Phillips, I. T.; Chen, S. e Haralick, R. M. (1993). CD-ROM document database standard. In *Proc. 2nd IAPR ICDAR*, pages 478–483.

[Polastro e da Silva Eleuterio, 2010] Polastro, M. d. C. e da Silva Eleuterio, P. M. (2010). Nudetective: A forensic tool to help combat child pornography through automatic nudity detection. In *Proceedings of the 2010 Workshops on Database and Expert Systems Applications*, DEXA '10, pages 349–353, Washington, DC, USA. IEEE Computer Society.

[Polastro e F. Almeida Jr., 2011] Polastro, M. d. C. e F. Almeida Jr., N. (2011). Ocr errors and their effects on computer forensics. In *Proceeding of the Sixth International Conference on Forensic Computer Science ICoFCS 2011*, ICoFCS '11, pages 115–121, Brasília-DF. Associação Brasileira De Especialistas Em Alta Tecnologia (ABEAT).

[Premchaiswadi et al., 2010] Premchaiswadi, N.; Yimgnagm, S. e Premchaiswadi, W. (2010). A scheme for salt and pepper noise reduction and its application for OCR systems. *W. Trans. on Comp.*, 9:351–360.

[Rice et al., 1995] Rice, S. V.; Jenkins, F. R. e Nartker, T. A. (1995). The fourth annual test of ocr accuracy. *Technical Report 95-04*.

[Rice et al., 1996] Rice, S. V.; Jenkins, F. R. e Nartker, T. A. (1996). The fifth annual test of OCR accuracy. *Technical Report 96-01*.

[Sellers, 1980] Sellers, P. H. (1980). The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1:359–373.

[Setubal e Meidanis, 1997] Setubal, J. e Meidanis, J. (1997). *Introduction to computational molecular biology*. PWS Publishing.

[Shapiro et al., 2001] Shapiro, L. G.; Stockman, G. C.; Shapiro, L. G. e Stockman, G. (2001). *Computer Vision*. Prentice Hall.

[Shinder e Tittel, 2002] Shinder, D. e Tittel, E. (2002). *Scene of the cybercrime: computer forensics handbook*. Elsevier Science.

[Smith, 2007] Smith, R. (2007). An overview of the tesseract OCR engine. In *International Conference on Document Analysis and Recognition*, pages 629–633.

[Smith, 2011] Smith, R. (2011). Tesseract open-source OCR engine. `http://code.google.com/p/tesseract-ocr/`. [Online; Acessado em outubro de 2011].

[Taghva et al., 1994] Taghva, K.; Borsack, J.; Condit, A. e Erva, S. (1994). The effects of noisy data on text retrieval. *Journal of The American Society for Information Science and Technology*, 45:50–58.

[Vacca, 2005] Vacca, J. (2005). *Computer forensics: computer crime scene investigation*. Number v. 1 in Networking Series. Charles River Media.

[van Rijsbergen, 1979] van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths.

[Voorhees e Harman, 1996] Voorhees, E. M. e Harman, D. (1996). Overview of the fifth text retrieval conference (trec-5). In *Text REtrieval Conference*, pages 1–25.

[Walter, 2003] Walter, C. (2003). Kryder's law. http://www.iacis.org/iis/2003_iis/PDFfiles/Webb.pdf. [Online; Acessado em julho de 2011].

[Waterman, 1981] Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197.

[Webb, 2003] Webb, G. K. (2003). A rule based forecast of computer hard drive costs. http://www.iacis.org/iis/2003_iis/PDFfiles/Webb.pdf. [Online; Acessado em julho de 2011].