UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE ARQUITETURA DE UM SISTEMA COM BASE EM OCR NEURONAL PARA RESGATE E INDEXAÇÃO DE ESCRITAS PALEOGRÁFICAS DO SEC. XVI AO XIX

FÁBIO LÚCIO LOPES DE MENDONÇA

ORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: 341/2008
BRASÍLIA / DF: JUNHO/2008

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UMA PROPOSTA DE ARQUITETURA DE UM SISTEMA COM BASE EM OCR NEURONAL PARA RESGATE E INDEXAÇÃO DE ESCRITAS PALEOGRÁFICAS DO SÉC. XVI AO XIX

FÁBIO LÚCIO LOPES DE MENDONÇA

DISSERTAÇÃO DE MESTRADO ACADÊMICO 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.

APROVADA POR:

RAFAEL TIMÓTEO DE SOUSA JÚNIOR, Dr, ENE/UNB
(ORIENTADOR)

ANDERSON CLAYTON ALVES NASCIMENTO, Dr, ENE/UNB
(EXAMINADOR INTERNO)

MARCOS MAGALHÃES DE AGUIAR, Dr., UNILEGIS
(EXAMINADOR EXTERNO)

BRASÍLIA, 27 DE JUNHO DE 2008.

FICHA CATALOGRÁFICA

MENDONÇA, FÁBIO LÚCIO LOPES.
Proposta de Arquitetura de um Sistema com Base em OCR Neuronal para Resgate e Indexação de Escritas Paleográficas do sec. XVI ao XIX. [Distrito Federal] 2008.
xiv, 104p, 297 mm (ENE/FT/UnB, MESTRE, Engenharia Elétrica, 2008).
Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Optical Character Recognition
2. Paleografia
3. Redes Neurais
4. Inteligência Artificial
I. ENE/FT/UnB.
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

MENDONÇA, Fábio Lúcio Lopes (2008). Proposta de Arquitetura de um Sistema com Base em OCR Neuronal para Resgate e Indexação de Escritas Paleográficas do sec. XVI ao XIX. Dissertação de Mestrado em Engenharia Elétrica, Publicação ENE. DM Jun/2008, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 104p.

CESSÃO DE DIREITOS

AUTOR: Fábio Lúcio Lopes de Mendonça

TÍTULO: PROPOSTA DE ARQUITETURA DE UM SISTEMA COM BASE EM OCR NEURONAL PARA RESGATE E INDEXAÇÃO DE ESCRITAS PALEOGRÁFICAS DO SEC. XVI AO XIX.

GRAU: Mestre ANO: 2008

_____

FÁBIO LÚCIO LOPES DE MENDONÇA
QNM 38 CONJUNTO U CASA 35
CEP 70.145-821 – Taguatinga – DF – Brasil

AGRADECIMENTOS

Ao orientador e amigo professor Dr. Rafael Timóteo Sousa Júnior, que me orientou de forma profissional e amiga nas horas mais complicadas durante a criação deste trabalho e que de uma forma ou de outra aturou tantas dúvidas e problemas relativos ao assunto e outros pequenos detalhes pertinentes à criação desta dissertação.

Aos Professores e amigos Anderson Clayton Alves Nascimento e Marcos Magalhães pelas grandes dicas e ajudas e pelo constante apoio, incentivo, dedicação e amizade, essenciais para o desenvolvimento deste trabalho.

Agradeço às bolsistas do CMD – Centro de Memória Digital, Inara Bezerra de Sousa e Thais Paranhos que auxiliaram bastante para o entendimento sobre paleografia. Agradeço também ao próprio CMD e ao NTI – Núcleo de Tecnologia da Informação, dos quais sou bolsista e tive grande auxilio e incentivo no decorrer desse trabalho.

Aos companheiros do LABREDES (Adriana, Wandemberg, Wesley, Eric, Rafael Baião, Beatriz Santana, Tamer Américo e todos os outros) que incentivaram para o desenvolvimento deste trabalho.

Aos meus pais pelo apoio e incentivo que foi dado durante todo o tempo em que estive envolvido neste trabalho.

Grato em particular aos meus amigos (as) Robson Oliveira, Georges Anvame, Alessandro Mendes e Tatiane Alves que contribuíram de forma fundamental para a conclusão deste trabalho. Meus sinceros agradecimentos.

Agradeço, sobretudo a Deus.

FÁBIO LÚCIO LOPES DE MENDONÇA

Brasília, DF 27 de Junho de 2008

DEDICATÓRIA

Para:

Edson Jr, Robson e Rafael

Meu Irmão e Amigos.

e

Edson Mendonça e Luciairan Mendonça,

Meus Queridos pais.

RESUMO

PROPOSTA DE ARQUITETURA DE UM SISTEMA COM BASE EM OCR NEURONAL PARA RESGATE E INDEXAÇÃO DE ESCRITAS PALEOGRÁFICAS DO SEC. XVI AO XIX.

Autor: Fábio Lúcio Lopes de Mendonça

Orientador: Rafael Timóteo de Sousa Júnior

Programa de Pós-graduação em Engenharia Elétrica

Brasília, Junho de 2008

Este trabalho objetiva propor uma arquitetura de um sistema para tratamento e reconhecimento automático do texto de documentos paleográficos, utilizando um OCR (Optical Character Recognition) com tecnologia de redes neurais artificiais. O sistema proposto deve atuar no contexto de processos de transcrição do texto de documentos de escritas paleográficas do século XVI ao XIX, documentos estes do Brasil colônia que foram digitalizados a partir dos originais impressos arquivados no Arquivo Ultramarino de Lisboa, uma das realizações do Projeto Resgate do Ministério da Cultura brasileiro.

A arquitetura do sistema proposto inclui módulos para segmentar as imagens digitalizadas dos documentos, para análise dos segmentos com OCR na tentativa de reconhecimento do texto, para treinamento do OCR com formação de um dicionário de palavras reconhecidas e para armazenamento do texto transcrito a partir das imagens dos documentos.

Para avaliar essa arquitetura foi desenvolvido um protótipo de software que permite ao usuário segmentar manualmente uma imagem de documento, treinar um OCR simples e extrair com esse OCR algumas informações de texto do documento paleográfico digitalizado. Conclui-se que a arquitetura proposta é funcional, ainda que sejam necessários desenvolvimentos mais profundos no que se refere aos processos de segmentação dos documentos e reconhecimento das escritas paleográficas do século XVI ao XIX.

ABSTRACT

PROPOSAL OF AN SYSTEM ARCHITETURE BASED ON NEURAL OCR FOR RESCUE AND INDEX PALEOGRAPHY WRITENS BETWEEN XVI AND XIX CENTURIES.

Author: Fábio Lúcio Lopes de Mendonça

Supervisor: Rafael Timóteo de Sousa Júnior

Programa de Pós-graduação em Engenharia Elétrica

Brasília, June of 2008

This work propose a system architecture for automatic manipulate and recognize of text on paleographic document, using Optical Character Recognition (OCR) aggregate with artificial neural networks. The system should work on the context of process text transcription on text documents with paleographic writing of century XVI to XIX; those documents are acquired from Brazil on colony age and digitalized from the original files archived on Ultramario Archive of Lisboa, one works of Projeto Resgate from Brazilian Culture Ministry.

The architecture of propose system has modules for segment the digital image of documents, analyze of segments with OCR in try of text recognize, OCR training for compose a dictionary of recognized worlds and also a module for storage the transcript text from document images.

For evaluation has been developed prototype software, where one user could manually segment a document image, simple OCR training and using this OCR gets some text information from a digital paleographic document. We conclude that the propose architecture was functional, but still need more improvements on document segmentation module and on module that recognize the paleographic writings of century XVI to XIX

ÍNDICE

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

ACRÔNIMOS

ADRT – Advance Decisions To Refuse Treatment

API – Application Programming Interface

ASCII - American Standard Code for Information Interchange

GUI – Graphical User Interface

HTML - HyperText Markup Language

IA – Artificial intelligence

ICR – Intelligent Character Recognition

IDE – Integrated Development Environment

IEEE - Institute of Electrical and Electronic Engineers

MAC – Media Access Control

MAN – Metropolitan Area Network

OCR – Optical Character Recognition

ODBC – Open Data Base Connectivity

PDF – Portable Document Format

RAM – Random Access Memory

RNA – Rede Neural Artificial

WEB SERVICES – integração de sistemas e na comunicação entre aplicações diferentes

1 - INTRODUÇÃO

Este trabalho objetiva propor uma arquitetura de um sistema para tratamento e reconhecimento automático do texto de documentos paleográficos, utilizando um OCR (*Optical Character Recognition*) com tecnologia de redes neurais artificiais. O sistema proposto deve atuar no contexto de processos de transcrição do texto de documentos de escritas paleográficas do século XVI ao XIX, documentos estes do Brasil colônia que foram digitalizados a partir dos originais impressos arquivados no Arquivo Ultramarino de Lisboa, uma das realizações do Projeto Resgate [1] do Ministério da Cultura brasileiro.

A leitura de um documento paleográfico é realizada, na maioria das vezes, por historiadores experientes e qualificados no assunto [2]. Os principais atributos para caracterizar uma escrita paleográfica são a época e a região em que se origina o documento a ser estudado, o que permite realizar diversas análises do manuscrito comparando suas características com a de documentos da mesma origem, ou de origem similar ou próxima, já analisados anteriormente. Desse modo, cria-se uma base de conhecimentos, em especial na forma de um dicionário controlado. Assim, as escritas paleográficas envolvem métodos de leitura que são estabelecidos por técnicas de aprendizado direto dos documentos ou por reutilização de conhecimentos já adquiridos e consolidados em um dicionário controlado ou outro artefato semelhante.

Diante do exposto, a proposta de arquitetura aqui apresentada inclui a extração de palavras dos documentos paleográficos digitalizados e também a formação de um dicionário correlacionando tais palavras aos atributos das imagens correspondentes, de modo a formar a base de conhecimentos a ser utilizada na transcrição de documentos do Projeto Resgate [1]. Por tal razão, a arquitetura do sistema proposto inclui módulos para segmentar as imagens digitalizadas dos documentos, para análise dos segmentos com OCR na tentativa de reconhecimento do texto, para treinamento do OCR com formação de um dicionário de palavras reconhecidas e para armazenamento do texto transcrito a partir das imagens dos documentos.

Para avaliar a viabilidade das técnicas propostas foi desenvolvido um protótipo de software que permite ao usuário segmentar manualmente uma imagem de documento, treinar um

OCR simples e extrair com esse OCR algumas informações de texto do documento paleográfico digitalizado.

Para embasar a proposta de arquitetura, uma pesquisa foi realizada sobre as técnicas mais recentes e mais eficientes utilizadas para o reconhecimento de caracteres e palavras. Constatou-se que as técnicas de reconhecimento de caracteres mais utilizadas e pesquisadas atualmente são baseadas em sistemas de OCR para o reconhecimento de caracteres e sistemas de redes neurais artificiais para o aprendizado dos caracteres reconhecidos [3]. Além do reconhecimento do texto impresso, esses dois tipos de técnicas são os mais utilizados em sistemas de reconhecimento de texto manuscrito.

A proposta de arquitetura aqui apresentada, como mencionado, foi projetada para utilização dentro do Projeto Resgate como um mecanismo automático para extração das palavras e textos de escritas paleográficas, armazenando o resultado dos documentos analisados. Assim, as técnicas utilizadas para a implantação dessa arquitetura são baseadas em um sistema de OCR que utiliza a forma de aprendizado baseada em redes neurais artificiais [4]. Para dar suporte a esse OCR, o protótipo implementado permite segmentar manualmente as palavras de um documento digitalizado, de modo a submeter tais palavras à rede neural, para que esta possa passar por um processo de aprendizagem, mas também para tentar o reconhecimento de outras palavras a partir do que foi aprendido. No presente documento, são descritos os módulos dessa arquitetura.

1.1 MOTIVAÇÃO

A Paleografia é o estudo de textos manuscritos antigos e medievais, investigando a origem, a forma e a evolução da escrita, independente do suporte físico onde foi registrada [5]. Mais especificamente, a Paleografia, em seu significado comumente aceito, é a ciência que tem por objeto o estudo das escritas antigas, em qualquer espécie de material, e que compreende a decifração, a descoberta de erros na transmissão [6], a datação de textos, a atribuição de lugar de origem e interpretação, além de ocupar-se da própria história da escrita. Também está intimamente relacionada a outras ciências, como a codicologia, a diplomática, a epigrafia e a papirologia.

As escritas paleográficas são classificadas e se diferenciam, conforme o tipo de escrita, a região na qual o documento se origina, formas de escritas, entre outros aspectos [5]. Sendo assim, as escritas paleográficas pertencem a famílias que se agrupam com relação a vários atributos analisados, assunto este cujos detalhes podem ser encontrados no capítulo 2 do presente trabalho.

Nesse contexto, existe um grande número de historiadores e pesquisadores nas áreas de paleografia, ciência e tecnologia da informação que têm realizado estudos para o desenvolvimento de processos de digitalização e arquivamento de documentos paleográficos preservando a essência dos documentos originais [7]. O Projeto Resgate é um dos esforços nesse sentido.

O projeto de resgate da Documentação Histórica Barão do Rio Branco (Projeto Resgate) [1] foi criado institucionalmente, em 1995, por meio de protocolo assinado entre as autoridades portuguesas e brasileiras no âmbito da Comissão Bilateral Luso-Brasileira. Tem como objetivo principal disponibilizar documentos históricos relativos à História do Brasil existentes em arquivos de outros países, sobretudo Portugal e demais países europeus. Entre os séculos XVI e XVIII, a escrita paleográfica no Brasil se concretiza na forma de testamentos, escrituras de terras, documentos oficiais do governo e da igreja, entre outros.

Visando ampliar os estudos sobre escritas paleográficas dos séculos XVI e XVIII, viabilizando a criação de um acervo histórico digital, para armazenamento e disponibilidade de arquivos históricos o Centro de Memória Digital (CMD) da Universidade de Brasília (UNB) foi criado e, em parceria com o Ministério da Cultura (MC) e com apoio financeiro da Petrobras, teve como meta inicial a criação do maior banco de dados eletrônico de imagens paleográficas do Brasil, sendo o acesso ao acervo das imagens realizado de forma *on-line* [1].

Após alguns meses de trabalho junto à equipe do Projeto Resgate, analisando escritas de documentos históricos do Brasil, dos séculos XVI e XVIII, a observação da dificuldade de leitura desse tipo de escrita foi o principal motivo que levou a acreditar na possibilidade da criação de uma arquitetura para elaboração de um mecanismo de reconhecimento

automático de escritas do tipo paleográficas. O estudo inicial apontou para as possibilidades das redes neurais de constituírem uma técnica básica de um sistema de reconhecimento ótico de caracteres que permitisse a leitura do tipo de escrita dos documentos arquivados no CMD, resultantes do Projeto Resgate. Por tais razões, a pesquisa voltou-se à proposta de uma arquitetura de sistema para tal finalidade e a avaliação de sua efetividade.

Segundo Alspector [8], as redes neurais artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas, possuindo uma capacidade computacional adquirida por meio de aprendizado. O aprendizado em RNAs consiste da fase onde a rede neural absorve dados e, a partir destes, modifica seus parâmetros de entrada. Esta etapa pode ser considerada como uma adaptação da RNA às características intrínsecas de um problema, onde se procura cobrir um grande espectro de valores associados às variáveis pertinentes. Nesse caso, as RNAs serão de grande importância para descrição e apresentação deste trabalho e têm um melhor detalhamento no capítulo 2.

Já os sistemas de OCR têm como característica o reconhecimento de caracteres a partir de um arquivo de imagem ou um mapa de bits [9], extraindo o texto de uma imagem e armazenando tal texto de maneira editável. Sendo assim este trabalho propõem a extração de textos de uma imagem de documento paleográfico, tornando o texto editável. Para que isso ocorra é necessario o estudo de um sistema de OCR, verificando as características em comum de cada sistema, analisando as características das imagens que serão tratadas.

As ferramentas OCR hoje existentes, inclusive aquelas comercializadas, fazem uso de mecanismos de decodificação de textos digitados e manuscritos para um texto editável [10], mas são voltadas ao reconhecimento de texto impresso contemporâneo. Entretanto, devido ao grande número de documentos existentes no Projeto Resgate, há um grande interesse na elaboração de uma arquitetura de um possível sistema de “OCR paleográfico”, tendo a possibilidade de essa aplicação ser utilizada não somente no Brasil, mas também em países como Portugal e Espanha.

1.2 OBJETIVO

Especificamente, o objetivo deste trabalho é a apresentação de uma proposta de arquitetura de OCR baseado em redes neurais artificiais, projetado em código aberto, que possa contribuir para segmentação e leitura de escritas paleográficas do Século XVI ao XIX.

Assim, foram fundamentais para o andamento deste trabalho os estudos sobre captura de imagens, mecanismos de OCR para tratamento e reconhecimento de caracteres, técnicas de reconhecimento e aprendizagem de palavras utilizando redes neurais, o que constitui um material de estudo que contempla um mecanismo de reconhecimento de caracteres, apresentando uma análise do que já existe nesta área e propondo idéias que possam agregar valor a esse tipo de tecnologia.

Este trabalho apresenta um protótipo envolvendo as tecnologias de OCR e redes neurais, oferecendo uma realização da arquitetura proposta para permitir a avaliação dos resultados.

1.3 ORGANIZAÇÃO DO TRABALHO

Trataremos nesta dissertação dos principais assuntos pertinentes ao tema proposto, explicando sua estrutura de funcionamento e utilização. Para um melhor entendimento e organização, este trabalho é dividido nos tópicos a seguir.

No capítulo 2 são explicados os conceitos de paleografia, principais métodos de leitura paleográfica e as diferenças encontradas nas escritas paleográficas de cada século. Além disso, serão mostrados os conceitos de redes neurais artificiais, tipos de redes neurais e métodos utilizados para segmentação de palavras. Na seqüência, alguns mecanismos de OCR e como podem ser implementados associados a redes neurais.

No capítulo 3 é explicada a análise da viabilidade da transcrição automática de documentos paleográficos através de mecanismos de OCR já existentes.

Os conceitos e a proposta de uma arquitetura de um sistema de transcrição automática de documentos paleográficos serão explanados no capítulo 4.

No capítulo 5 temos conclusão desse trabalho e trabalhos futuros.

2 - CONCEITOS E FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordadas as questões relativas aos tipos de escritas e documentos históricos que permitem caracterizar as escritas paleográficas encontradas no Projeto Resgate e utilizadas no desenvolvimento desta dissertação. Além disso, serão abordados os temas de inteligência artificial, redes neurais artificiais, tipos de redes neurais, processos de aprendizado e a utilização de aplicações com redes neurais que se aplicam ao problema objeto do trabalho. Assim, serão mostrados os conceitos relativos a OCR, incluindo arquiteturas de OCR, tipos existentes, OCR utilizando redes neurais artificiais e mecanismos para reconhecimento de caracteres e palavras.

2.1 VISÃO GERAL DE ESCRITAS PALEOGRÁFICAS

A paleografia é o estudo de textos manuscritos antigos e medievais, tratando da origem, da forma e da evolução da escrita, independente do suporte onde foi registrada e da diferenciação desse suporte por século, região e artefatos utilizados [11].

Nas escritas primitivas os caracteres correspondiam a palavras, sílabas ou consoantes; aos fenícios se deve a primeira escrita alfabética, que os Gregos aperfeiçoaram incluindo-lhe vogais. Os alfabetos gregos constituíram os padrões de base para as escritas bizantina e cirílica (búlgara, sérvia e nissa) na Europa Oriental, e na Europa Ocidental para a escrita latina, donde, por sua vez, derivam todas as escritas ocidentais até a atualidade [12]. Dos alfabetos puramente consonânticos, derivaram a escrita árabe e a escrita hebraica.

Na Pré-História o homem buscou se comunicar através de desenhos feitos nas paredes das cavernas. Através deste tipo de representação (pintura rupestre), trocavam mensagens, passavam idéias e transmitiam desejos e necessidades. Porém, ainda não era um tipo de escrita, pois não havia organização, nem mesmo padronização das representações gráficas [12] e [13].

Foi somente na antiga Mesopotâmia que a escrita foi elaborada e criada. Por volta de 4000 a.C, os sumérios desenvolveram a escrita cuneiforme. Usavam placas de barro, onde cunhavam esta escrita. Muito do que sabemos hoje sobre este período da história, devemos

as placas de argila com registros cotidianos, administrativos, econômicos e políticos da época.

Os egípcios antigos também desenvolveram a escrita quase na mesma época que os sumérios. Existiam duas formas de escrita no Antigo Egito: a demótica (mais simplificada) e a hieroglífica (mais complexa e formada por desenhos e símbolos) [14]. As paredes internas das pirâmides eram repletas de textos que falavam sobre a vida dos faraós, rezas e mensagens para espantar possíveis saqueadores. Uma espécie de papel chamada papiro, que era produzida a partir de uma planta de mesmo nome, também era utilizado para escrever [13] e [14].

Já na Roma Antiga, no alfabeto romano havia somente letras maiúsculas. Contudo, na época em que estas começaram a ser escritas nos pergaminhos, com auxílio de hastes de bambu ou penas de patos e outras aves, ocorreu uma modificação em sua forma original e, posteriormente, criou-se um novo estilo de escrita denominado uncial. O novo estilo resistiu até o século VIII e foi utilizado na escritura de Bíblias [15].

Na Alta Idade Média, no século VIII, Alczuíno, um monge inglês, elaborou outro estilo de alfabeto atendendo ao pedido do imperador Carlos Magno. Contudo, este novo estilo possuía letras maiúsculas e minúsculas [16]. Com o passar do tempo, esta forma de escrita também passou por modificações, tornando-se complexa para leitura. Assim, no século XV, alguns eruditos italianos, incomodados com este estilo complexo, criaram um novo estilo de escrita.

No ano de 1522, outro italiano, Lodovico Arrighi, foi o responsável pela publicação do primeiro caderno de caligrafia. Foi ele quem deu origem ao estilo que hoje denominamos itálico. Com o passar do tempo outros cadernos também foram impressos, tendo seus tipos gravados em chapas de cobre (calcografia). Foi deste processo que se originou a designação de escrita calcográfica [14]. Como material suporte da escrita, o papel só é conhecido na Europa a partir do século XIII. Ao longo de processo lento que se estendeu até a atualidade foi substituindo o antigo pergaminho [16] e [17]; Por sua vez este desalojara o papiro desde o inicio da Alta Idade Média, determinando simultaneamente o abandono dos rolos (de papiro) em favor dos códices encadernados (de pergaminho e

depois de papel) até a forma de livro que usamos hoje.

Segundo Ubirajara Dolacio [18], nos séculos XVI e XVII os cronistas, para elaborar as suas obras, necessitavam de recorrer a documentos com princípios empíricos de Paleografia e Diplomática. Além disso, a paciente tarefa de vasculhar os cartórios das suas instituições, lendo os documentos, observando-lhes a letra e forma.

O desenvolvimento das diversas formas da escrita latina (entre elas a gótica) deixa entrever mútua influência das escritas librária e corrente. Segundo Rui Gusmão [19], as escritas librárias (corresponde-lhes desde o começo dos tempos modernos a escrita impressa) devem ser quanto possível agradáveis à vista, regulares e facilmente legíveis; na escrita corrente, ou cursiva, adaptada ao uso quotidiano, importa principalmente uma forma rápida e cômoda (na escrita impressa o cursivo aparece em itálico - antiquar).

A partir do século XVI surgem no Brasil os primeiros documentos com escritas paleográficas, através de escrituras trazidas de Portugal e Espanha. Esses registros têm sido conservados desde a formação do reino de Portugal onde a partir do século XVI que os padres paroquiais da Igreja Católica foram solicitados a começar a registrar batismos, casamentos e falecimentos. Estes são considerados os registros genealógicos mais valiosos em Portugal e no Brasil [20]. Durante o século XVI, outros tipos de registros de valor genealógico também começaram a proliferar, com a necessidade de analisar a autenticidade e veracidade de documentos históricos faz com que vários historiadores de hoje venham cada vez mais se interessar pelo assunto. Saber transpor os caracteres de um documento paleográfico original para caracteres que já estamos familiarizados é uma das importantes habilidades exigidas pelos estudos paleográficos [19] e [21].

2.1.1 Tipos de escritas paleográficas

A principal relação para caracterizar uma escrita paleográfica é saber qual época e região se origina o documento a ser estudado, realizando assim diversas análises em um manuscrito com as mesmas características gráficas do texto original [22].

Os tipos de paleografias se classificam através de exames cuidadosos das formas e

abreviações das letras usadas antigamente em cada variedade de escrita. Os paleógrafos aprendem a decifrar os velhos documentos, que geralmente podem concluir com precisão onde e quando os documentos não datados foram escritos. Além disso, as escritas paleográficas podem ser classificadas em três famílias: visigótica, carolina e gótica. Outro dos itens considerados na classificação foi o desenvolvimento da escrita ao longo do tempo, onde a escrita pode ser caracterizada como inicial, plena ou pode ser influenciada por traços que anunciam uma nova família encontrando-se numa fase de transição. Um terceiro aspecto tido em consideração foi à existência ou não de cursividade e em que grau ela estava presente, pois as escritas foram designadas de cursiva, semi-cursiva, semi-caligráfica e caligráfica. A qualidade de execução foi o quarto item classificativo usado, permitindo distinguir a escrita que atingiu um grau próximo do perfeito, daquela que é apenas regular, semi-regular,irregular ou semi-analfabeta. Outro aspecto tido em conta na classificação das escritas é a dignidade, ou não, do ato que a escrita procura espelhar; existirá assim uma escrita solene, semi-solene, usual e descuidada. Enfim o último aspecto considerado é o da instituição em que o escritor se integra [22].

Dentre os tipos e classificações de escritas paleográficas mostrados, será explicado com mais detalhes os tipos visigótica, carolina e gótica os demais podem ser encontrados com mais detalhes nas referências [22], [23] e [24] deste trabalho.

2.1.1.1 Família de Escrita Visigótica

A escrita visigótica caracteriza-se por em na sua forma mais pura, possuir hastes bastante altas relativamente ao corpo da letra em dimensões reduzidas, e por possuir um conjunto de letras bem diferenciadas [24].

As letras distintivas são: o ‘a’ aberto, o ‘g’ estreito de forma uncial, o ‘t’, que segundo SANTOS, [25] designa como ‘beta invertido’, o ‘z’ designado pela paleografia espanhola de ‘copetudo’ e o ‘et’ em ligadura do ‘e’ com o ‘t’. Possui ainda um sistema abreviativo diferente daquele que se imporá com a carolina; não utiliza abreviaturas por letras sobrescritas e usa profusamente as modificações literais.

A escrita visigótica segundo SANTOS [25], teria sido feita no início com um cálamo,

porque é composta por traços cuja espessura é muito homogênea onde os traços verticais, horizontais e oblíquos, são finos ou grossos mas rigorosamente uniformes não havendo alternância acentuada entre cheios e os finos e nem facturas nas letras.

A Figura 2-1 parece não apresentar as características da escrita Visigótica, pois deve ter sido escrito com uma pena de ave.

Figura 2-1 - Documento do *corpus* em visigótica

Fonte: Este documento foi extraído do tabelionado medieval português, publicada por Alarcão,1959:299-305, são referidas as cartas de sentença, de confirmação e de emprazamento[26].

2.1.1.2 Família de Escrita Carolina

Segundo Jakson [26] e [27], a escrita carolina era uma característica de escrita de ampla difusão, onde as escritas da Idade Média que lhe sucederam, inclusive a do tipo gótica, guardaram a sua forma em ductus. Esta escrita tem como característica a primeira forma harmoniosa conseguida no jogo das hastes e do corpo da letra. O seu alfabeto integra heranças anteriores – por exemplo, o ‘a’ é uncial, o ‘n’ e o ‘g’ merovíngios - mas possui características próprias o ‘s’ carolíngio é longo e distingue-se do ‘f’ porque este possui um pequeno traço horizontal (nem sempre muito visível), o ‘t’ não ultrapassa o corpo da letra

(o que o levará mais tarde a confundir-se com o ‘c’) e é rematado por um traço horizontal, o ‘e’ distingue-se do ‘c’ por possuir um terceiro traço a avançar para frente e por não formar tão facilmente, como o ‘e’ visigótico, ligaduras com as letras que se lhe seguem.

Esta escrita, feita com uma pena de bico direito, possui traços verticais grossos, traços horizontais finos, as curvas vão progressivamente engrossando, não havendo um forte contraste entre cheios e finos e conseguindo um aspecto equilibrado devido à proeminência dos traços perpendiculares [28].

O sistema braquigráfico típico da carolina e que é adaptado quase antes da escrita em si, tem como característica fundamental a utilização das abreviaturas por letras sobrepostas [29] e [30]. O nome ‘minúscula carolina’, pelo qual se designa muitas vezes esta letra, pretende revelar um fenômeno que se inicia neste período e que é a utilização dos alfabetos capitais e unciais para salientar ou ‘maiuscularizar’ títulos, não se tratando, pois de uma característica de letra em si, mais sim de uma prática.

O alfabeto carolino de Higounet [30] e [31], é uma boa abstração do que poderia ser um alfabeto carolino - um modelo que raramente se encontra de forma tão ‘pura’ nos manuscritos do séc. IX. (e que pode se observar em algumas reproduções), como mostra a Figura 2-2.

Figura 2-2 - Reprodução do alfabeto da escrita carolina

Fonte: Este documento foi extraído de um dicionário de escrita carolina o alfabeto de Higounet,1986:90 nos manuscritos do séc. IX [32].

Segundo SANTOS, [25], nos documentos dos anos de 822 a 1172 analisados e que possuíam características da letra carolina, só cerca de um terço puderam ser apelidados de carolina ‘plena’ tendo os outros sido classificados de ‘carolino-gótica’ ou carolina de

transição. É o caso da Figura 2-3 que mostra um exemplo de escrita carolina de chancelaria (usualmente referida como minúscula diplomática). Mais detalhes podem ser encontrados nas referências [32],[33] e [34].

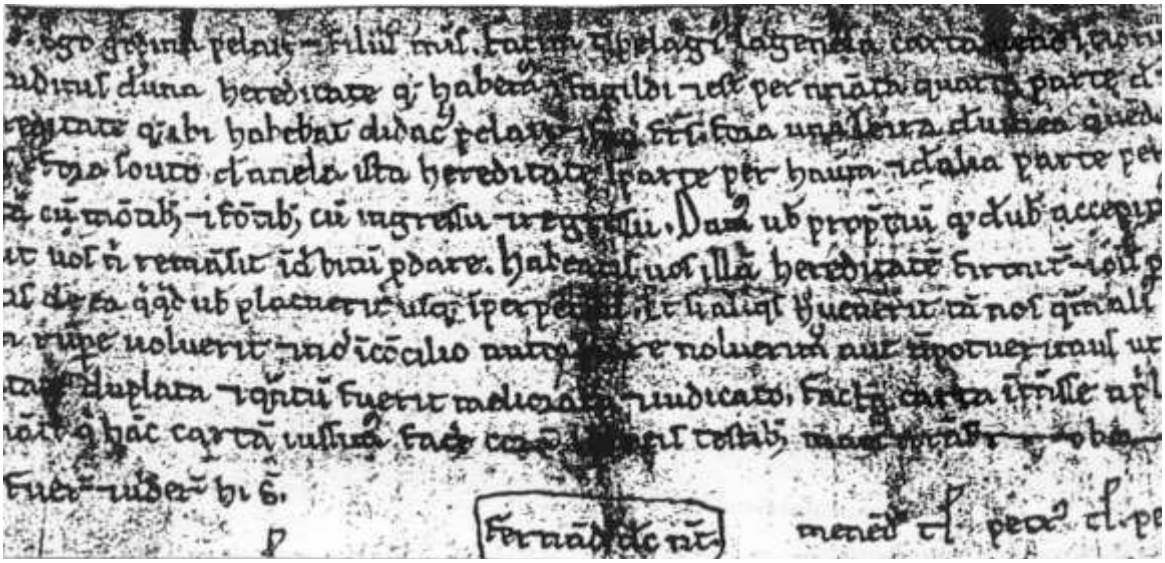

Figura 2-3 - Escrita classificada como carolina de transição

Fonte: Este documento, também foi extraído do tabelionado medieval português, publicada por Alarcão,1959:299-305, são referidas as cartas de sentença, de confirmação e de emprazamento[33].

2.1.1.3 Família de Escrita Gótica

A principal característica deste tipo de letra é a sua forma angulosa, que tem tendência de rapidamente evoluir para formas cursivas, com hastes superiores e inferiores, alongadas e com os traços sem espessura, uniforme e mais fino nas extremidades. O ‘a’ alarga e perde o pequeno traço à esquerda, o ‘c’, o ‘e’ e o ‘t’ confundem-se, o ‘d’ uncial, que existia já em alguns documentos carolinos, passa a dominar, o ‘s’ final é arredondado, o ‘u’ passa a ser substituído no início das palavras por um ‘v’ que muitas vezes adquire uma forma de ‘b’, no ‘m’ e no ‘n’ aparecem pequenos traços adventícios na base. Continuam a existir os nexos entre ‘st’ e a maioria destas góticas comuns coloca em nexo duas letras de curvas contrárias (‘be’, ‘po’, etc.) e evita o encontro entre letras com traços curvos e letras com traços direitos (‘or’) [34] e [35]. Começa a formar-se um verdadeiro alfabeto de maiúsculas

de módulo bastante grande, como por exemplo ‘a’, ‘b’, ‘f’, ‘h’, ‘m’, ‘n’, ‘p’, ‘v’ obtidas pelo engrandecimento das minúsculas e posteriores ornamento com pequenos traços finos e cheios [36]. Embora não existindo nos documentos editados, há um outro tipo de gótica denominada ‘gótica librária’ que coexiste com o tipo anterior, porém este tipo de letra utiliza-se em geral na escrita de livros deste período, mas pode excepcionalmente ser usado outro tipo de suporte.

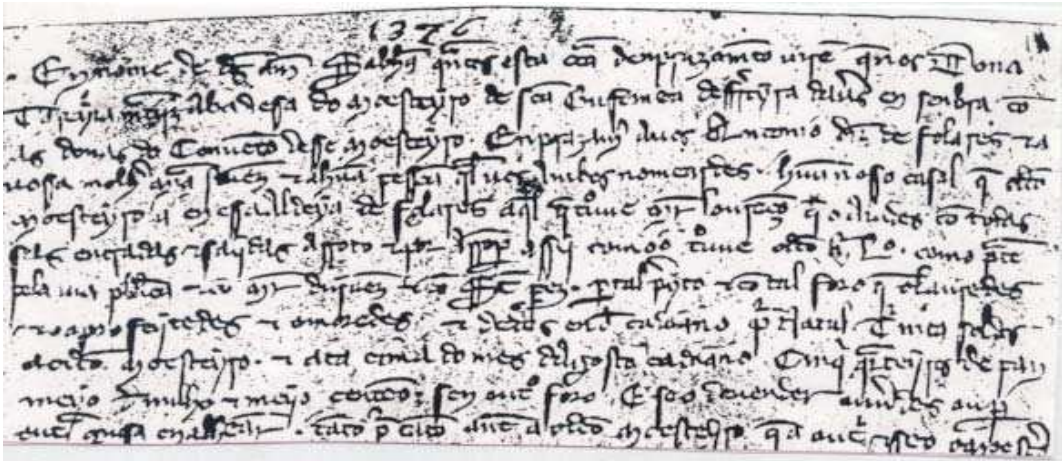

Figura 2-4 - Tipo de escrita gótica cursiva.

Fonte: Este documento foi retirado em NUNES, Eduardo - Álbum de Paleografia Portuguesa, Lisboa: Instituto de Alta Cultura, 1969.

Algumas das características desta letra ‘librária’ já foram apresentadas como características da ‘gótica’ propriamente dita, mas ao contrário desta, a ‘librária’ não tem hastes ascendentes ou descendentes salientes e as letras terminam (inclusive 's' ou 'f') na linha, os traços perpendiculares à linha são todos traçados da mesma forma, ou seja possuem a fractura característica e têm finos traços de ligação [36]. Como mostra a Figura 2-4 pertencente ao período mais inicial da escrita gótica.

2.1.2 Métodos de escritas paleográficas

Nos métodos de escritas, o paleógrafo deve ler os manuscritos antigos (normalmente papiros e pergaminhos) atribuir-lhe uma data e um local de edição ou produção. Em regra, as edições paleográficas dizem respeito a textos antigos escritos em grego e latim. O editor

paleógrafo deve conhecer os diferentes estilos dos copistas e ser capaz de identificar os períodos em que os textos foram escritos [37]; nota-se que nem todos os copistas são confiáveis, pois alguns eram sobretudo desenhadores que não sabiam ler o que copiavam, produzindo erros que se repetiam de cópia em cópia.

Através do exame cuidadoso das formas e abreviações das letras usadas antigamente em cada variedade de escrita, os paleógrafos aprendem a decifrar os velhos manuscritos. Geralmente podem concluir com precisão onde e quando os documentos não datados foram escritos. Desse modo, por vezes descobrem falsificações. A epigrafia, um ramo especializado da paleografia, é o estudo das inscrições feitas em material resistente, como a pedra e o metal [38] e [39].

Segundo Rodrigues [40], as dificuldades encontradas em arranjar uma estrutura classificativa revelam-se na terminologia imperfeita e a falta de uma reflexão profunda sobre o assunto. Abordar uma realidade tão plástica como é a escrita, em que se cruzam tendências, se petrificaram gestos e grafias, onde o movimento de ‘descaligrafização’ e ‘recaligraficação’ é constante, é tarefa árdua, mas estimulante. Subjacente a qualquer escrito concreto existe uma idéia, diferente para cada escritor, de certa forma de letra, que muito dificilmente se consegue explicitar, mas que o paleógrafo acaba também ele, por interiorizar [41].

2.1.3 Documentos históricos do Projeto Resgate

O Projeto Resgate [2] tem como objetivo principal disponibilizar documentos históricos relativos à História do Brasil existentes em arquivos de outros países, sobretudo Portugal e demais países europeus, com os quais tivemos uma história colonial.

O significado e importância deste projeto residem no apoio à preservação da memória histórica nacional e na democratização do acesso ao patrimônio documental brasileiro. O Projeto Resgate conta com aproximadamente 340.000 documentos paleográficos (perto de três milhões de páginas manuscritas) relativos a 18 capitanias da América Portuguesa. É considerado hoje um dos maiores acervos de documentação colonial brasileira no exterior.

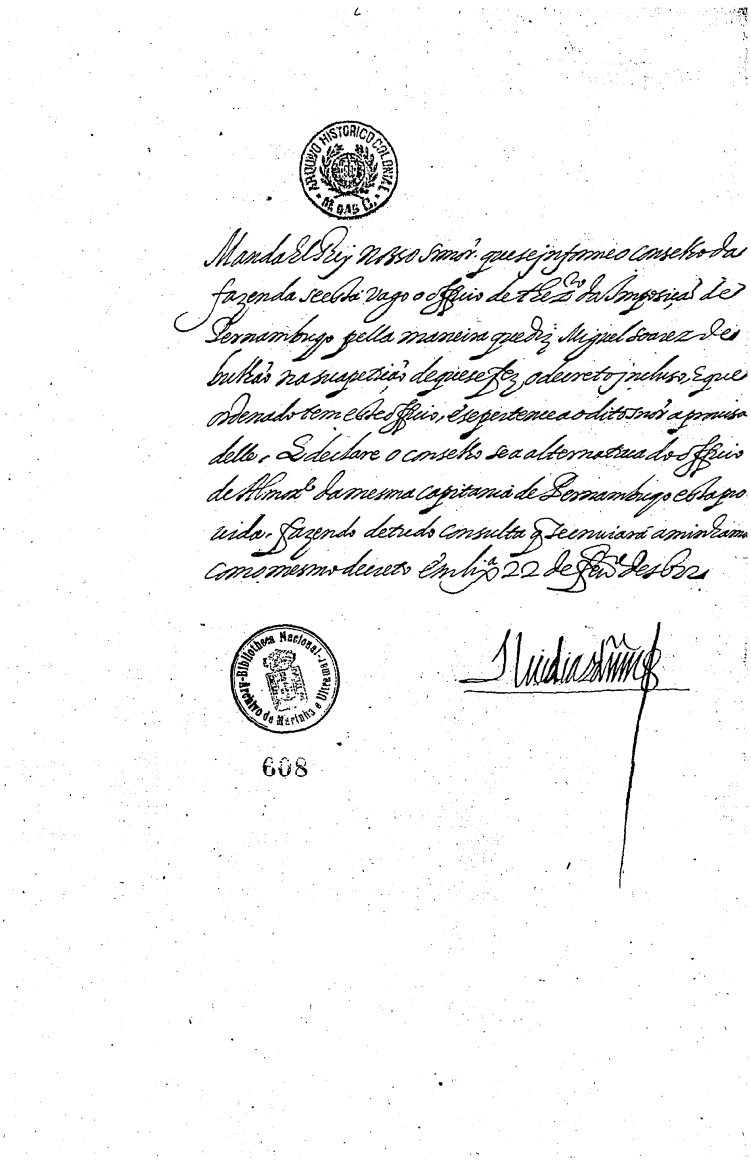

Figura 2-5 - Documento extraído do Projeto Resgate na capitania de Pernambuco
Fonte: Esse documento representa a Certidão (cópia) da provisão que determinou as várias medidas que o ouvidor-geral da comarca de Santa Catarina, bacharel Manuel José de Faria, deveria tomar para a boa administração daquela comarca [2].

Dessa forma, o Projeto Resgate reúne o maior acervo paleográfico do Brasil de maneira digitalizada e armazenada em um banco de dados de imagens, que pode ser pesquisado pelo site www.resgate.unb.br. Neste site existem informações mais detalhadas do projeto, além de poder pesquisar e visualizar diversos documentos de escrituras antigas, como

mostra o as Figura 2-5 e Figura 2-6 extraídas do projeto.

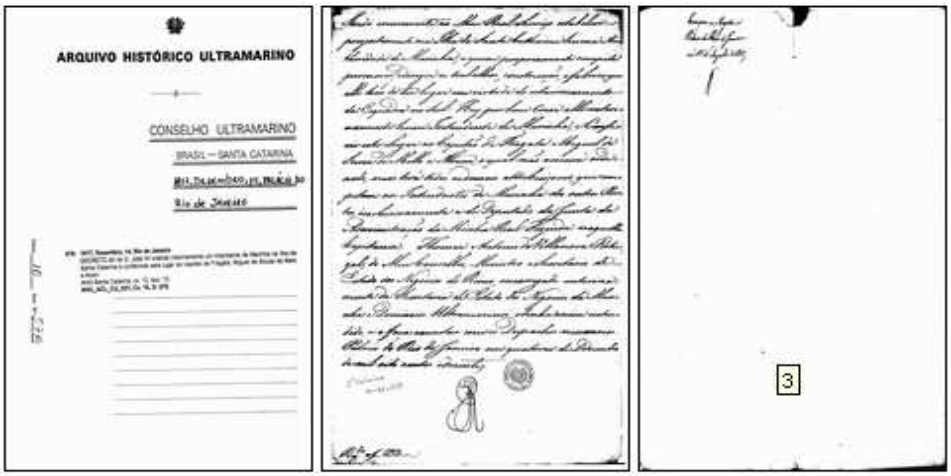

Figura 2-6 - Documento extraído do Projeto Resgate na capitania de Rio Grande do Sul. Fonte: Esse documento representa o DECRETO do rei D. João VI criando interinamente um Intendente de Marinha na ilha de Santa Catarina e conferindo este lugar ao capitão de Fragata, Miguel de Sousa de Melo e Alvim.

Dessa forma, podemos concluir que o processo de entendimento sobre escritas paleográficas e suas características é fator de grande importância para o entendimento deste trabalho.

2.2 REDES NEURAIS ARTIFICIAIS

Para que se tenha um bom entendimento sobre redes neurais artificiais (RNAs), é interessante ver inicialmente alguns conceitos de Inteligência Artificial (IA).

O objetivo da inteligência artificial é poder realizar tarefas cognitivas, para quais os humanos são altamente melhores. O meio pelo qual são realizadas essas tarefas é o desenvolvimento de paradigmas ou algoritmos. Um sistema de IA [42] deve ser capaz de realizar três funções fundamentais, armazenar conhecimentos, aplicar o conhecimento armazenado para resolver problemas e adquirir novos conhecimentos através da experiência.

No decorrer deste capítulo será falado de inteligência artificial e redes neurais artificiais,

mostrando a dependência entre elas.

2.2.1 Inteligência Artificial e Redes Neurais

Basicamente, uma rede neural se assemelha ao cérebro humano em dois pontos: o conhecimento através de etapas de aprendizagem e pesos sinápticos é usado para armazenar o conhecimento. Uma sinapse é o nome dado à conexão existente entre neurônios. Nas conexões são atribuídos valores, que são chamados de pesos sinápticos. Isso deixa claro que as redes neurais artificiais têm em sua constituição uma série de neurônios artificiais (ou virtuais) que serão conectados entre si, formando uma rede de elementos de processamento [43] e [44].

Segundo Luger [45] a Inteligência Artificial é o ramo da ciência da computação que se preocupa a com automação do comportamento inteligente. Este principio inclui uma estrutura de dados usada na representação do conhecimento, alguns algoritmos serão necessários para aplicar tal conhecimento e as linguagens e técnicas de programação são usadas na sua implementação. Sendo assim um sistema IA possui três componentes fundamentais: representação, raciocínio e aprendizagem.

Representação: É o uso difundido de uma linguagem de estruturas simbólicas para representar tanto conhecimento genérico sobre um domínio de um problema de interesse como o conhecimento especifico sobre a solução do problema [46] e [47]. “Conhecimento” pode ser do tipo declarativo ou procedimental. Em uma representação declarativa, o conhecimento é representado como uma coleção estática de fatos, com um pequeno conjunto de procedimentos gerais utilizados para manipular os fatos. Em uma representação procedimental, o conhecimento está incorporado em um código executável que representa o significado do conhecimento.

Raciocínio: É a habilidade de resolver problemas. O sistema deve ser capaz de resolver uma vasta gama de problemas e seus tipos. Sendo capaz de tornar conhecidas para ele tanto a informação explicita como a informação implícita. Por fim o sistema deve ter um mecanismo de controle que determine quais operações devem ser aplicadas para um problema particular, quando uma solução para este problema foi obtida, e quando deve ser

encerrado o tratamento deste problema [47].

Aprendizagem: No modelo simples de aprendizagem de maquina representado pela Figura 2-7, o ambiente fornece alguma informação para um elemento de aprendizagem, com isso o elemento de aprendizagem utiliza esta informação para aperfeiçoar a base de conhecimento e, finalmente, o elemento de desempenho utiliza a base de conhecimento para executar determinada tarefa [46]. Normalmente, a informação que o ambiente de aprendizagem fornece para a maquina é imperfeita, assim o elemento de desempenho "não sabe" previamente como preencher os detalhes ausentes ou ignorar os detalhes que não são importantes. Portanto a maquina opera inicialmente por suposições e depois recebe uma realimentação do elemento de desempenho a partir daí o mecanismo de realimentação permite que a maquina avalie suas hipóteses e se necessário as revise [48].

A aprendizagem de maquina envolve dois tipos bastante diferentes de processamento de informação sendo o indutivo e o dedutivo. No indutivo, padrões gerais e regras são determinadas a partir dos dados brutos e da experiência. A aprendizagem baseada em similaridade utiliza indução, enquanto que a prova de um teorema é uma dedução. Já o processamento de informação dedutivo é utilizado regras gerais para determinar fatos específicos. A aprendizagem baseada em explanação utiliza tanto indução como dedução [49].

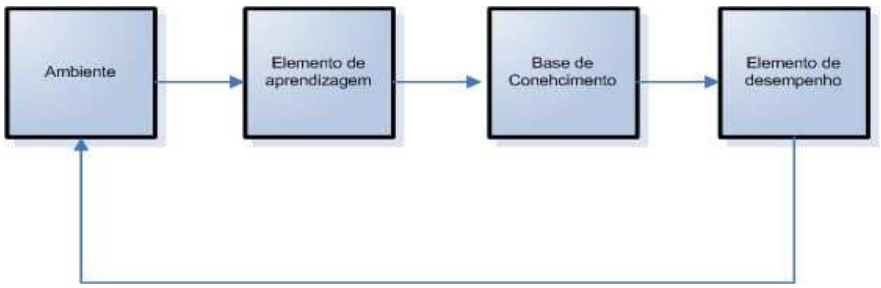

Figura 2-7 - Modelo simples de aprendizagem de maquina.

As Redes Neurais Artificiais segue o principio de IA, [50] podendo dizer que é um tipo de IA baseadas em técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da

experiência.

Uma grande RNA pode ter centenas ou milhares de unidades de processamento já o cérebro humano pode ter muitos bilhões de neurônios [50]. A idéia é realizar o processamento de informações tendo como princípio a organização de neurônios do cérebro.

Como o cérebro humano é capaz de aprender e tomar decisões baseadas na aprendizagem, as RNAs devem fazer o mesmo. Assim, uma rede neural pode ser interpretada como um esquema de processamento capaz de armazenar conhecimento baseado em aprendizagem (experiência) e disponibilizar este conhecimento para aplicação em questão [517].

O neurônio que transmite o pulso que pode controlar a freqüência de pulsos aumentando ou diminuindo a polaridade na membrana pós sináptica [52]. Ele tem um papel essencial na determinação do funcionamento, comportamento e do raciocínio do ser humano. Ao contrário das RNAs, as redes neurais naturais não transmitem sinais negativos, sua ativação é medida pela freqüência com que emite pulsos, freqüência esta de pulsos contínuos e positivos. As redes naturais não são uniformes como as redes artificiais, e apresentam uniformidade apenas em alguns pontos do organismo.

Segundo Simon [51], o sistema nervoso humano é formado por um conjunto complexo de neurônios. Nos neurônios a comunicação é realizada através de impulsos, quando um impulso é recebido, o neurônio o processa, e passado um limite de ação, dispara um segundo impulso que produz uma substância chamada de neurotransmissora o qual flui do corpo celular para o axônio (que por sua vez pode ou não estar conectado a um dendrito de outra célula).

2.2.2 Características das Redes Neurais Artificiais

As RNAs são compostas por várias unidades de processamentos. As unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso, que fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões [53]. O comportamento inteligente de uma RNAs vem das interações entre as

unidades de processamento da rede. A Figura 2-8 apresenta uma característica de RNAs, realizando uma simulação de um neurônio biológico, onde possuem várias entradas, representadas por X_1, X_2 e X_n e uma saída representada por $Y_{k,}$ cujas ligações com o corpo celular artificial são realizadas através de elementos chamados de peso (simulando as sinapses) representado por δ(-). Os estímulos captados pelas entradas são processados pela função de soma, e o limiar de disparo do neurônio biológico foi substituído pela função de transferência. A maioria dos modelos de redes neurais possui alguma regra de treinamento, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados [53] e [54].

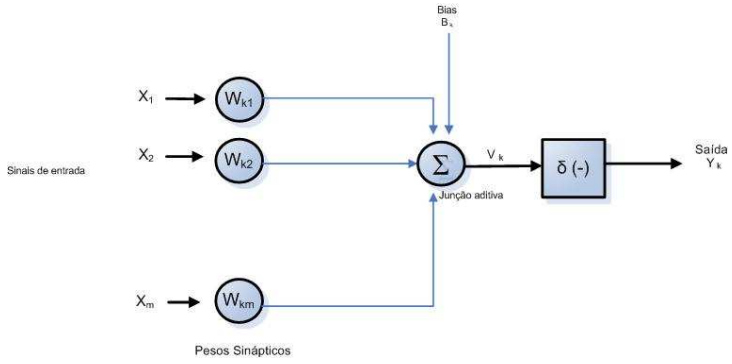

Figura 2-8 - Modelo não linear de um neurônio

As RNAs podem ser caracterizadas também pelo tipo de arquitetura apresentada, assim na maioria dos modelos possuem algumas regras de treinamento que são tipicamente organizadas em camadas, onde podem ser classificadas em três tipos: camada de entrada, intermediárias e camada de saída. Possuindo unidades que podem estar conectadas às unidades da camada posterior. Assim os pesos de suas conexões são ajustados de acordo com os padrões apresentados. Em outras palavras, elas aprendem através de exemplos [54]. A Figura 2-9 mostra uma arquitetura de rede neural organizado por camadas.

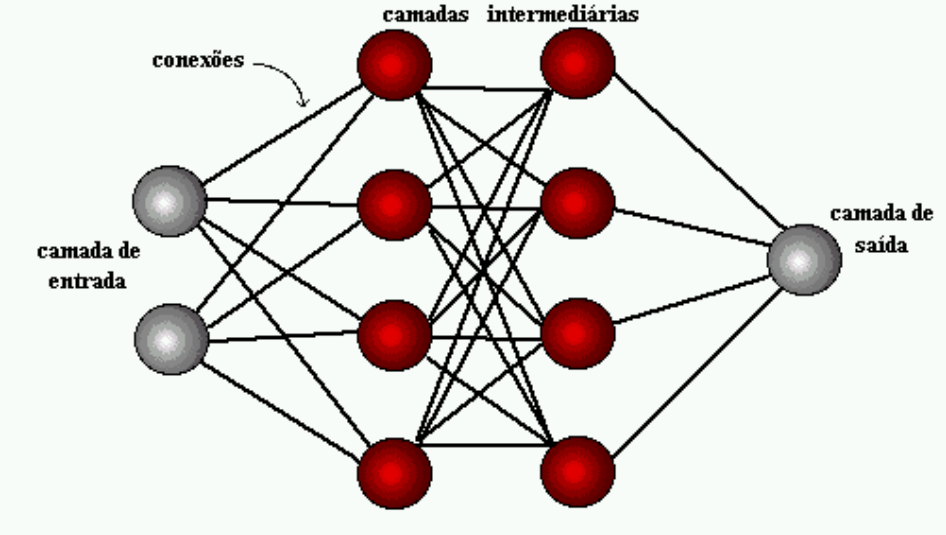

Figura 2-9 - Organização das camadas topologia principal de redes neurais.
Fonte: McCulloch, W. S., and Pitts, W., “A logical calculus of the ideas immanent in nervous activity.”

2.2.3 Classificação de Redes Neurais Artificiais

Segundo Haykin existem diversos tipos de RNA e diferentes maneiras de classificá-las. Talvez a mais importante seja quanto à forma de aprendizado que pode ser de maneira supervisionada e não supervisionada [55].

Nas redes classificadas com o método de aprendizado supervisionado são apresentados um conjunto de padrões de entrada e seus correspondentes padrões de saída. Durante este processo, a rede realiza um ajustamento dos pesos das conexões entre os elementos de processamento, segundo uma determinada “lei de aprendizagem”, até que os erros entre os padrões de saída gerados pela rede alcancem um valor mínimo desejado [56].

Existem dezenas de leis de aprendizagem supervisionada, dentre elas podem especificar as perceptron, perceptron multi camadas, adaline e madaline [57] e [58], que serão apresentadas com mais detalhes no decorre deste capítulo. Outro tipo de aprendizagem similar à supervisionada é a aprendizagem por reforço. Neste tipo de aprendizagem, ao invés de fornecer as saídas corretas para a rede, relativo a cada treinamento individual, a

rede recebe somente um valor que diz se a saída está correta ou não [56].

No aprendizado não-supervisionado a rede “analisa” os conjuntos de dados apresentados, determina algumas propriedades e “aprende” a refletir estas propriedades na sua saída. A rede utiliza padrões, regularidades e correlações para agrupar os conjuntos de dados em classes. As propriedades que a rede vai “aprender” sobre os dados podem variar em função do tipo de arquitetura utilizada e da lei de aprendizagem [56].

As redes também podem ser classificadas quanto à suas características, que pode ser contínua, discreta, determinística e estocástica. Também podem ser classificados quanto à sua estrutura, que subdivide–se em: a) redes de múltiplas camadas (multilayer feedforward network), cujo fluxo de dados segue uma única direção e; b) redes recursivas (recurrent network). Mais informações sobre esses dos tipos de redes podem ser encontradas nas referências [56] e [57].

2.2.3.1 Perpectron com uma camada

Os perceptrons de uma camada são capazes de aprender somente sobre problemas linearmente separáveis por uma reta em um hiperplano [58]. Embora uma única unidade do ponto inicial seja completamente limitada em seu poder computacional, mostra-se que as redes de unidades paralelas do ponto inicial podem aproximar toda a função contínua de um intervalo compacto dos números reais no intervalo de [- 1, 1].

2.2.3.2 Perpectron com multi camada

As redes Multi-camadas podem usar um grande número de técnicas de aprendizado, sendo que a considerada mais popular é a propagação reversa [58]. Os perpectron de múltiplas camadas têm sido aplicados para resolver problemas considerados “difíceis”, através do seu treinamento de forma supervisionada com um algoritmo de retropropagação de erro, onde esse algoritmo é baseado na regra de aprendizagem por correção de erros [54]. Neste caso os valores de saída são comparados com a resposta correta para computar o valor de alguma função-erro predefinida. Por alguma técnica o erro é então alimentado de volta na rede [58]. Usando essa informação, o algoritmo ajusta os pesos de cada conexão para

reduzir o valor da função erro.

Esta classe de rede consiste de múltiplas camadas de unidades computacionais, geralmente interconectadas em uma forma feedforward. Isso quer dizer que cada neurônio em uma camada tem conexões diretas a neurônios da próxima camada. Em muitas aplicações as unidades dessas redes aplicam uma função sigmóide (em forma de S) como a função de ativação [59]. A Figura 2-10 representa uma topologia do perpectron de multi-camadas.

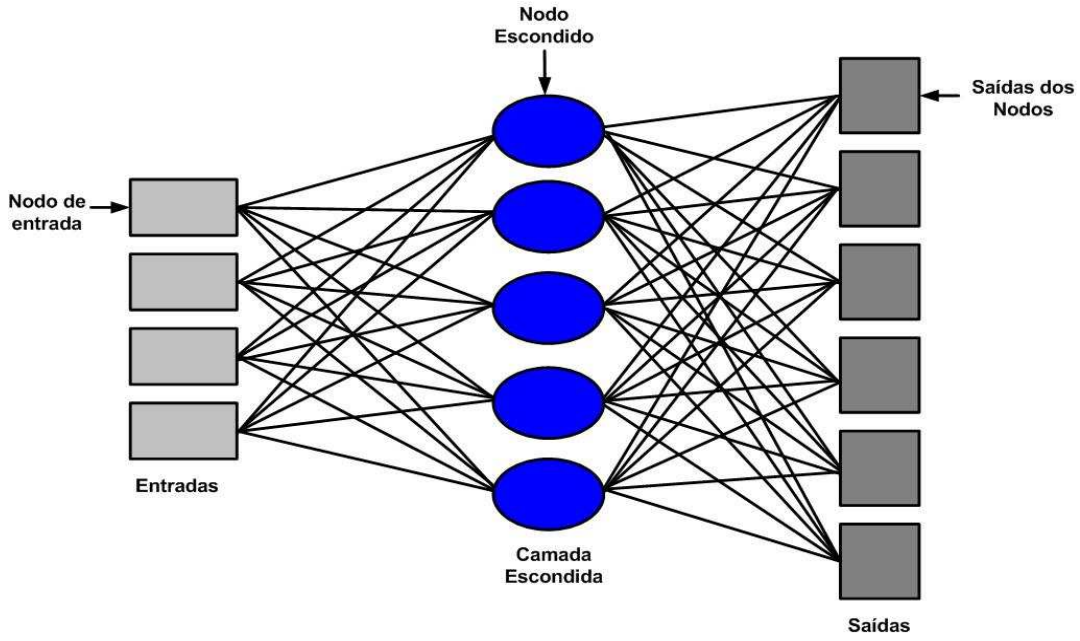

Figura 2-10 - Arquitetura do perpectron de multi-camadas

2.2.4 Funcionamento das Redes Neurais Artificiais

As atividades executadas em redes neurais artificiais são criadas a partir de algoritmos projetados para uma determinada finalidade. É muito improvável criar um algoritmo, sem ter conhecimento de modelos matemáticos que simulem o processo de aprendizado do cérebro humano [53].

Tendo em vista que as redes neurais artificiais têm como principal características esse tipo de processo, lembrando que os processos de aprendizagem inicial das redes neurais muitas das vezes não apresentam valores totalmente satisfatórios.

Sendo assim, a partir do momento em que a rede vai obtendo informações através do treinamento os índices de erro vão diminuindo cada vez mais.

Tendo uma rede neural montada, uma série de valores pode ser aplicada sobre um neurônio, sendo que este está conectado a outros pela rede. Estes valores (ou entradas) são multiplicados no neurônio pelo valor do peso de sua sinapse. Após a aplicação do peso, esses valores são somados. Se esta soma ultrapassar um valor limite estabelecido, um sinal é propagado pela saída (axônio) deste neurônio. Em seguida, essa mesma etapa se realiza com os demais neurônios da rede. Isso quer dizer que os neurônios vão enfrentar algum tipo de ativação, dependendo das entradas e dos pesos sinápticos [54].

Existem várias formas de se desenvolver uma rede neural. Ela deve ser montada de acordo com o problema a ser resolvido. Em sua arquitetura é determinados o número de camadas usadas, a quantidade de neurônios em cada camada e o tipo de sinapse utilizado.

2.2.5 Processo de Aprendizado

O processo de aprendizagem das redes neurais é realizado quando ocorrem várias modificações significantes nas sinapses dos neurônios. Essas mudanças ocorrem de acordo com a ativação dos neurônios. Se determinadas conexões são mais usadas, estas são reforçadas enquanto que as demais são enfraquecidas. É por isso que quando uma RNA é implantada para uma determinada aplicação, é necessário um tempo para que esta seja treinada [53]. A propriedade mais importante das redes neurais é a habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicado a seus pesos, o treinamento. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

Denomina-se algoritmo de aprendizado a um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados [58].

Existem diversos tipos de processos de aprendizagem, dentre eles descreveremos podemos

citar a aprendizagem por correção de erro, baseada em memória, com um professor e sem um professor;

2.2.5.1 Processo de Aprendizagem por Correção de Erro

Nesse processo, o erro de uma RNA pode ser calculado como a diferença entre a saída real gerada pela rede e a saída desejada, fornecida em um ensino supervisionado [54]. Como mostra a formula.

$$e_k = d_k - y_k$$

Onde para um estímulo k,

e - sinal de erro*;*
d - saída desejada apresentada durante o treinamento*;*
y - saída real da rede após a apresentação do estímulo de entrada.

Durante o aprendizado supervisionado, os erros vão sendo calculados sucessivamente, até que cheguem a um valor satisfatório, definido a priori. Sendo assim, surge uma curva de erros, a qual está diretamente relacionada à natureza do modelo de neurônio utilizado.

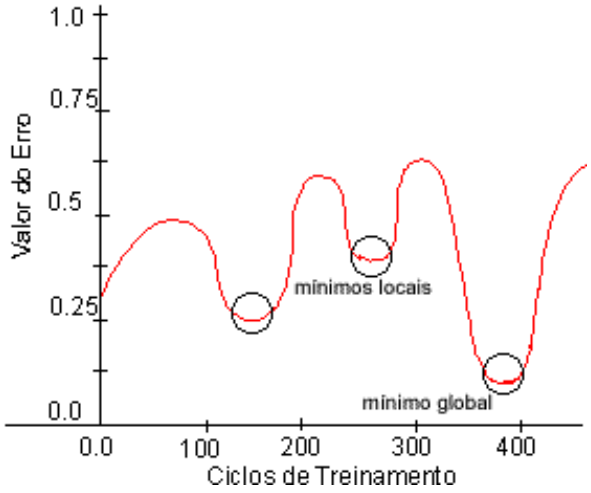

Figura 2-11 - Superfície de erro mostrando os mínimos locais e o mínimo global

Fonte: Swanson, N. e White, H. (1994), "Can Neural Networks Forecast in the Bib Leagues.”

Dessa forma o processo de aprendizado por correção de erros utiliza algoritmos para caminhar sobre a curva de erros, com o intuito de alcançar o menor valor de erro possível, o mínimo global, como mostra a Figura 2-11 [54]. Muitas vezes, o algoritmo não alcança este mínimo global, atingindo o que pode ser chamado de mínimo local. Caso este erro alcançado seja desfavorável, é necessário recomeçar processo de aprendizado [58].

2.2.5.2 Processo de Aprendizagem por Memória

No processo de aprendizado por memória, a maioria das experiências passadas é armazenada em uma grande memória de exemplos de entrada-saída classificada corretamente: $\{(x_i,d_i)^n\}_{i=1}$, onde x_i representa um vetor de entrada e d_i representa a resposta desejada [59].

Num processo de aprendizado padrões são armazenados, e posteriormente, recuperados, na presença de estímulos a eles relacionados. É razoável afirmar que aprendizado e memória estão relacionados. Para que a memória seja útil, além de ser recuperável, através de estímulos, também deve influenciar as ações [58].

As propriedades abaixo, portanto, adéquam a um modelo artificial, porém realístico de memória:

- Distribuição;
- Mapeamento Entrada-Saida;
- Tolerância à falhas, aceitando um grau de ruído nos estímulos;
- Justaposição entre padrões armazenados, a fim de reduzir o tamanho da memória. Porém gerando possibilidade de falhas durante a recuperação das informações.

2.2.5.3 Processo de Aprendizagem Supervisionada

Consiste em um treinamento da RNA, apresentando pares de entradas e saídas, para cada entrada a rede produz uma resposta na saída. A resposta é comparada com o sinal de saída

desejado e assim, a rede gera um sinal de erro que corresponde à diferença desses sinais [58]. Sendo assim o professor tem conhecimento do ambiente e fornece um conjunto de exemplos de entrada-resposta desejada. Através desse conjunto, o treinamento é feito usando o processo de aprendizagem por correção de erro. A Figura 2-12 mostra um diagrama de blocos que ilustra esta forma de aprendizagem.

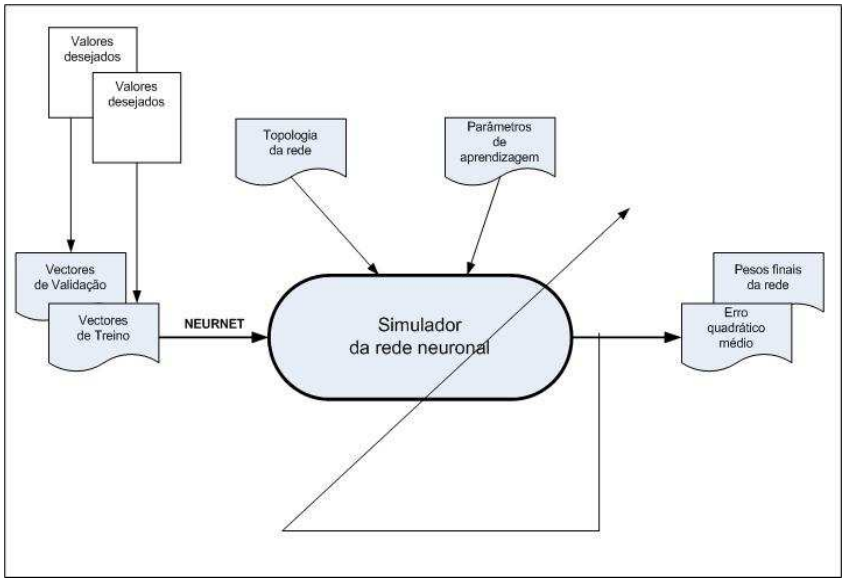

Figura 2-12 - Simulador para realizar a aprendizagem da rede neuronal

2.2.5.4 Processo de Aprendizagem não Supervisionada

No aprendizado não-supervisionado, utilizado em sistemas de classificação, não existe saída desejada. A rede é treinada através de excitações ou padrões de entrada e então, arbitrariamente, organiza os padrões em categorias [60]. Para uma entrada aplicada à rede, será fornecida uma resposta indicando a classe a qual a entrada pertence. Se o padrão de entrada não corresponde às classes existentes, uma nova classe é gerada. Entretanto, nesse processo de aprendizagem como o próprio nome implica, não há um professor para supervisionar o processo de aprendizagem. Isto significa que não há exemplos rotulados da função a ser aprendida [59]. Nesse processo são identificadas duas subdivisões.

Aprendizagem por reforço: O aprendizado que tem um mapeamento de entrada-saida é realizado através da interação continua com o ambiente, visando a minimizar um índice

escalar de desempenho. O sistema é projetado para aprender por reforço atrasado fazendo com que o mesmo observe uma seqüência temporal de estímulos [59] e [60]. Porém duas razoes básicas dificultam a aprendizagem pôr este método: A primeira é pela inexistência de um professor que forneça a resposta desejada em cada passo do processo de aprendizagem. A segunda, é que o atraso ocorrido na geração do sinal de reforço primário faz com que a maquina de aprendizagem torne-se capaz de atribuir crédito ou culpa individualmente a cada ação até o resultado final, enquanto o reforço primário apenas avalia o resultado;

Aprendizagem auto organizada: São dadas condições para a realização de uma medida independente da tarefa de qualidade da representação que a rede deve aprender. Os parâmetros livres da rede são otimizados em relação a esta medida, com isso é desenvolvida a habilidade de formação de representações internas para codificar as características da entrada gerando conseqüentemente novas classes [59] e [60].

Para realizar o aprendizado pode ser utilizada a regra da aprendizagem competitiva ou auto organizada.

2.3 OCR - OPTICAL CHARACTER RECOGNITION

OCR (Optical Character Recognition) é o reconhecimento óptico de caracteres, ou seja, tecnologia que permite reconhecer caracteres de texto em imagens, transformando-os em texto editável [61].

O reconhecimento de caracteres, é um subconjunto da área de reconhecimento de padrões e, foi ele que estabeleceu as bases e a motivação para tornar o reconhecimento de padrões e a análise de imagens campos individuais de interesse da ciência [62]. A importância das pesquisas nesta área não se baseia apenas em aplicações comerciais, mas também no reconhecimento de padrões complexos em 2D e 3D para sistemas de visão robótica e computacional.

O reconhecimento de caracteres propriamente dito apareceu primeiro como um auxílio para deficientes visuais cerca de 1900. Uma versão moderna do OCR apareceu em meados de 1940 com o desenvolvimento do computador digital [63]. Inicialmente o OCR foi

desenvolvido apenas para processamento de informações com uma aplicação limitada ao mundo dos negócios [62] e [63]. A origem do reconhecimento de caracteres provavelmente vem de meados do ano de 1870, quando foi criado o scanner de retina, que é um sistema de transmissão de imagem utilizando um mosaico de fotocélulas [63]. O scanner seqüencial, teve surgimento por volta de 1890 onde foi um avanço para o desenvolvimento da TV moderna e das máquinas de leitura ótica. Os sistemas de reconhecimento se vêem em face a um paradoxo: Como reconhecer sem segmentação, e como segmentar sem reconhecimento? A estratégia usada freqüentemente é a geração de muitas hipóteses de segmentação, seguidas de testes para todas as combinações possíveis [64].

Uma proposta que vem ganhando muitos adeptos e que já representa um novo segmento de pesquisa, o ICR (Inteligent Character Recognition), implica no uso de redes neurais.

Os sistemas de OCR possuem algumas características e funcionalidades interessantes como:

- A utilização de um sistema OCR para modificação de apenas uma parte de um texto que está em formato de papel, modificando-o para um texto editável;
- O reconhecimento de caracteres padronizados de escritas de outras nações como, japonesas, chinesas e árabes, transcrevendo-as para língua portuguesa;
- Na segmentação de caracteres, e dos métodos usados para a extração de características e o reconhecimento dos caracteres das placas de veículos.
- No reconhecimento de escritas manuscritas da língua portuguesa ou demais línguas;

Dessa forma, verifica-se que um sistema de OCR, pode ser utilizado para diversas funcionalidades e ramos de atividade, dependendo da forma que ele seja implementado [65]. Sendo assim é possível identificar e classificar um OCR através do algoritmo que seja implementado no mesmo.

2.3.1 Tipos de OCR

O reconhecimento de caracteres é uma tecnologia que possibilita o usuário reconhecer os caracteres extraídos de documentos impressos e/ou manuscritos. Para que esse reconhecimento ocorra, é necessário que os documentos estejam no formato de imagem,

digitalizada. Para tornar um documento produzido em papel, em uma imagem ou em formato digital, são necessários alguns tipos de mecanismos para o reconhecimento dos caracteres como o caso de um escâner ou uma câmera digital [66].

A Tecnologia de Reconhecimento de Caracteres pode ser divida em dois tipos:

- OCR (Optical Character Recognition) - Reconhecimento de Caracteres Ópticos, melhor entendido como reconhecimentos de caracteres impressos.

- ICR (Intelligent Character Recognition) - Reconhecimento de Caracteres Inteligentes, melhor entendido como reconhecimento de caracteres impressos, manuscritos, código de barra e marcações

Entretanto, os sistemas de ICR são de certa forma, um tipo de OCR avançado capaz de reconhecer diferentes formas de escritas manuscritas. Os sistemas de ICR são aplicados na captura de dados de formulários, reconhecendo caracteres em formulários já impressos anteriormente (formulários pré-existente), ou desenhados através da ferramenta de desenho [66] e [67].

2.3.2 Segmentação e reconhecimento de caracteres

Dentre as modalidades de reconhecimento de padrões, o reconhecimento de caracteres é uma das mais conhecidas e exploradas pela comunidade científica. Prova disso é a grande quantidade de artigos publicados anualmente em periódicos e anais de conferências nacionais e internacionais relacionadas, e até mesmo especializadas no assunto [68].

O reconhecimento de caracteres consiste em características extraídas de um conjunto de caracteres. Além disso, a segmentação de caracteres depende do padrão, da forma e de características que as palavras foram escritas, existindo vários métodos de segmentação que podem ser encontrados em [68] e [69].

Um método bastante utilizado para segmentação de caracteres manuscritos não conectados é o método baseado em histogramas de projeção. O histograma de projeção é uma técnica que faz uso da projeção da imagem sobre cada uma das duas dimensões existentes. A cada

dimensão é associado um vetor onde é armazenado o número de pixels com tonalidade de cinza acima de um determinado limiar (geralmente tomado como a cor de fundo). A segmentação é então realizada aplicando-se uma série de refinamentos sucessivos até se chegar a um resultado satisfatório. O método pode ser visto como uma seqüência de três etapas distintas: inicia com uma compensação da qualidade da imagem, seguida por uma segmentação inicial e então, se necessário, por um conjunto de refinamentos sucessivos [70].

2.3.2.1 Reconhecimento de formato livre

É a capacidade de captar e processar formulários que mesmo sendo semelhantes não seguem exatamente o mesmo desenho, por exemplo, faturas de diversos fornecedores, encomendas de vários clientes ou até mesmo reconhecimento de placas de veículos.

Nesse caso o reconhecimento de caracteres de formato livre segue uma padronização de escritas, e tem como objetivo relacionar uma entrada a um dos caracteres pertencente a um conjunto, denominado alfabeto [70]. Um grande exemplo é o reconhecimento de placas de veículos onde a segmentação consiste em realizar a separação de cada um dos caracteres da placa, criando sete recortes, cada um com a imagem de um caractere da placa, como mostra as Figura 2-13 e Figura 2-14 [70].

Figura 2-13 – Imagem inicial

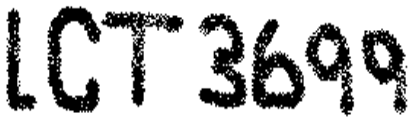

Figura 2-14 – Imagem Segmentada

O percentual de segmentação correta da placa em sete caracteres distintos depende de vários fatores em relação a todo conjunto de dados, até mesmo fatores externos onde várias

imagens apresentam problemas (completamente escuras, placa em mau estado, placa encoberta) alguns inclusive que impossibilitam a própria leitura humana.

São extraídas características de cada imagem segmentada, de forma a compor uma assinatura para cada caractere. A extração é feita por uma técnica desenvolvida onde é derivada dos algoritmos de detecção de contorno que podem ser descritas em [71] e [72].

2.3.2.2 Manuscrito cursivo

O reconhecimento dos caracteres visa classificar as palavras a partir das informações geradas pela extração das suas características. Para a execução de determinadas tarefas é utilizada técnicas de redes neurais artificiais, para o reconhecimento de letras e números.

Segundo Thomé [73] graças ao grande aumento de processamento dos computadores atuais, os mecanismos de OCR e ICR dispõem agora de velocidade e capacidade necessárias para correr os complexos algoritmos necessários para reconhecer validamente as palavras e frases, manuscritas e cursivas. Este tipo de reconhecimento baseia-se menos na classificação individual dos caracteres e mais em reconhecer elementos genéricos de palavras, padrões de “loops” acima e abaixo da linha de escrita [74].

Até agora os sistemas de OCR para escritas manuscritas cursivas, tem sido usado para ambientes em que as frases são curtas, o conteúdo controlado, e há possibilidade de validar o resultado com outra informação do mesmo documento. Exemplo, o valor por extenso nos cheques, formulários de concursos, segmentação de caracteres numéricos para endereçamento postal.

O reconhecimento de caracteres cursivos não é algo tão simples, pois trata de um tipo de caractere sem restrição, que na maioria das vezes estão conectados. Definitivamente é um dos mais complexos e, mas difícil de ser implementado. Até hoje é muito difícil de conseguir qualquer técnica de desempenho garantidamente satisfatório, uma vez que não se podem limitar os inúmeros parâmetros, formas e características individuais da escrita cursiva [74]. O desempenho de um sistema automático de reconhecimento depende da qualidade dos documentos nas suas formas original e digital. Usam-se processos para

compensar uma qualidade pobre nos originais e nas imagens após a captura e digitalização. Estes processos incluem realces, remoção de linhas, de sublinhados e de ruídos, além de outros. Os problemas geralmente relacionados com a qualidade e a dificuldade de tratamento da imagem e do texto segundo Berthold [75] são: ruído, distorção, variação de estilo, translação, escala, rotação, texturas e traços.

O método de escritas cursivas de padrões numéricos pode ser vista como um caso mais simples do cursivo genérico, uma vez que, por sua natureza, os dígitos em geral não são escritos de forma conectada. È o caso de reconhecimento de padrões cursivos de CEP (Código de Endereçamento Postal).

2.3.3 Reconhecimento de padrões OCR em redes neurais

As tarefas de reconhecimento de padrões podem ser facilmente implementadas através do uso de um algoritmo de aprendizado supervisionado, onde necessita apenas de um conjunto de exemplos de imagens previamente “etiquetadas” com as respectivas classes, a serem usadas no treinamento da rede [71]. O exemplo clássico deste tipo de aplicação é o reconhecimento de caracteres em imagens de textos que foram digitalizados (OCR). O reconhecimento de caracteres é uma tarefa difícil, dadas as inúmeras variações em relação à posição, orientação, tamanho e mesmo forma e erros de leitura dos caracteres, que dificultam a aplicação de um algoritmo predeterminado para o reconhecimento destes.

Um algoritmo de aprendizado permite ao sistema de OCR que ele seja “treinado” para reconhecer novas fontes de textos. Os sistemas OCR são bastante complexos, consistindo de múltiplos módulos, onde se pode usar uma rede neural como mecanismo de reconhecimento de padrões [71] e [72]. Não é tão difícil de encontrar imagens de caracteres extraídos de textos digitalizados. Estes caracteres podem ser identificados com o uso de um sistema OCR, onde a partir da matriz de pontos devem ser capazes de retornar a classe ao qual esta matriz pertence. Uma rede neural pode receber como entrada os dados contidos na matriz, e como saída retornar o código correspondente ao caracter fornecido.

2.4 SÍNTESE DO CAPÍTULO

Conforme observamos, as escritas paleográficas são documentos medievais que relatam as

estórias de vários países como o próprio Brasil. As escritas existentes no Brasil estão ligadas diretamente ao tipo de escritas existentes em outros países de acordo com a época. No Brasil existe um projeto muito importante, que disponibiliza o maior acervo de documentos paleográficos digitalizados entre os séc. XVI ao XIX. O Projeto Resgate [2], na qual terá grande contribuição para o andamento desse trabalho, pois através deste projeto, que poderei apresentar uma proposta de um produto OCR para escritas paleográficas.

Foram apresentados também nesse capítulo uma visão bastante simplificada sobre redes neurais, dando uma breve introdução à área das redes neurais artificiais. A principal característica deste trabalho é a familiarização com os conceitos básicos da neurocomputação, neurônio artificial, topologia da rede neural, paradigmas de aprendizagem e regras para a adaptação dos pesos. Ao mesmo tempo esclarece as capacidades e limitações de uma rede neural artificial.

Sistemas baseados em reconhecimento de caracteres ainda são motivos de estudos nas áreas de ciência da computação e engenharia. O seu uso ainda é restrito em alguns casos específicos. Entretanto, existem diversos sistemas para reconhecimento de caracteres inclusive de reconhecimento de caracteres manuscritos.

Apesar das demonstrações e trabalhos realizados, existem diversos problemas ainda relacionados especificamente para um OCR com reconhecimento de cores e escritas, principalmente no que se relaciona ao objetivo deste trabalho, que é um OCR para reconhecimento de escritas paleográficas com seus mecanismos interligados a redes neurais.

3 - VIABILIDADE DE TRANSCRIÇÃO DE DOCUMENTOS PALEOGRÁFICOS

Neste capítulo será abordada a viabilidade da transcrição automática de documentos paleográficos através de mecanismos de OCR já existentes. Além disso, serão mostrados alguns problemas encontrados no tratamento dessas escritas com o uso de sistemas de OCR “convencional”.

Como já visto, os sistemas de OCR se classificam e se diferenciam através das características de leitura neles existentes. Sendo assim, nesse capítulo serão analisados diversos mecanismos de OCR já existente para realizar vários testes em diferentes tipos de documentos e imagens, verificando o comportamento em cada tipo. Inclusive em documentos de escritas paleográficas.

3.1 CONSIDERAÇÕES INICIAIS

Hoje existem vários tipos de mecanismos de OCR, para diversos tipos de documentos como: OCR para tratamento de imagens, com caracteres padronizados (padrão ANSI), reconhecimento de números e até mesmo de escritas manuscritas.

Entretanto, existem mecanismos de OCR capazes de reconhecer mais de uma característica de escrita, reconhecendo e convertendo caracteres alfanuméricos de um documento digitalizado.

A criação de reproduções das imagens dos documentos oferece a habilidade de consultas de textos para encontrar documentos ou conteúdo específicos. Usuários podem também localizar instantaneamente palavras ou frases específicas dentro de um documento ou grupo de documentos, o que aperfeiçoa sensivelmente mesmo a mais exaustiva das tarefas de pesquisa [76]. Alguns documentos são freqüentemente digitalizados e convertidos por OCR como caso de correspondências comerciais, memorandos jurídicos, contratos, folhas de especificações e manuais.

Após vários testes em diversos mecanismos de OCR, encontramos alguns módulos que

oferecem várias opções de formatação. Reproduções de imagens que podem ser armazenadas em formatos de texto ASCII, PDF, DOC ou HTML. A formatação pode ser mantida para conservar o aspecto original do documento ou removida, deixando apenas o texto do documento. A velocidade do OCR pode ser ajustada para melhor reconhecimento de imagens, dependendo das necessidades e da qualidade do documento original [77]. Os módulos de OCR permitem que diferentes opções sejam gravadas para cada tipo de documento a ser processado. O processo de OCR pode ser realizado em imagens em preto-e-branco, escalas de cinza, ou mesmo à cor, e suporta reconhecimento de caracteres muitas vezes em até 6 idiomas.

3.2 DESCRIÇÃO DO AMBIENTE DE TESTE

Com o objetivo de demonstrar a proposta deste trabalho, foram analisados alguns sistemas de OCR que pudesse comprovar a viabilidade e funcionalidade para tipos de documentos específicos, como é o caso de documentos do tipo paleográfico. Para a demonstração desses sistemas, foi necessária a criação de um ambiente com os dispositivos devidamente configurados para idealização e avaliação do ambiente proposto.

Para isto, foi feito um estudo em alguns sistemas existentes e verificou-se a possibilidade do uso dos mesmos para situações ainda não exploradas. Sendo assim, a proposta deste trabalho explora algumas características existentes nos sistemas apresentados, de forma que possa adotar uma métrica, para os diferentes tipos de testes. Foram apresentados dois mecanismos de OCR com características semelhantes, porém um de código aberto e outro proprietário, os aplicativos analisados foram o Pytesser (softwares de código livres) [78] e o ABBYY Fine (proprietário) [79].

O ambiente utilizado para as simulações do trabalho é composto por 1 computador do tipo desktop com processador Pentium Dual Core E2140 1.6/800/1Mb, com 2GB de memória RAM, com um espaço em disco de 160GB e com uma placa de vídeo do tipo Off Board com 256Mb. Além disso, foi utilizado um mecanismo de Scanner Multifuncional da HP (Hewlett-Packard) modelo laserjet 3030, para as digitalizações das imagens. Com relação à parte de softwares, foi utilizado como sistema operacional, o Microsoft Windows XP

Professional, como editor de texto utilizou-se o Office 2003 e o Adobe Photoshop CS2 para edição de imagens. Como linguagem de programação o Python em sua versão 2.3 e o PyScripter for Windows, como plataforma de desenvolvimento.

Para que as simulações ocorressem da melhor maneira possível, resolvemos adotar algumas padronizações nas imagens utilizadas. Onde, as imagens testadas utilizaram extensão do tipo jpeg, com tamanho físico de aproximadamente 1250 KB. As imagens não possuíam cores (somente preto e branco) e o padrão de caracteres utilizados foi de 11 caracteres cada imagem. No caso dos caracteres não houve uma razão especifica para utilizar 11 caracteres, poderia ter sido 12 ou 13, desde que a quantidade de caracteres fosse à mesma para todas as imagens. Além disso, todas as simulações foram realizadas somente utilizando o sistema operacional e o sistema de OCR, nenhum outro software rodando na maquina, para que não houvesse nenhuma variação relacionada à concorrência de processamento e memória.

Dessa forma, serão apresentados às simulações realizadas com os tipos de imagens já descritas e com os aplicativos de OCR já apresentados.

Para realização das simulações em textos padronizados, para ambos os sistemas de OCR foram adotadas as seguintes características:

- Foi analisado um texto do MS Word convertido para imagem;
- O documento gerado foi digitalizado em um scanner, conforme descrito anteriormente;
- A imagem foi salva em uma extensão do tipo jpeg;
- Foi utilizada uma quantidade padrão de 15 repetições.

Para realização das simulações com imagens de escritas manuscritas, foram adotadas as seguintes características:

- Foi utilizado um texto escrito de próprio punho em um papel comum;
- O documento foi digitalizado em um scanner com característica já apresentadas;

- A imagem foi salva em uma extensão do tipo jpeg;
- Foi utilizada uma quantidade padrão de 15 repetições;

A realização das simulações com imagens de escritas paleográficas, serão melhor descritas no contexto de cada aplicativo.

3.3 TIPOS DE OCR APRESENTADOS

As ferramentas de OCR aqui apresentadas serão utilizadas para simulações em palavras padronizadas, palavras do tipo manuscritas e palavras do tipo paleográficas. Além disso, será realizado um comparativo entre os dois aplicativos a fim de identificar se os mesmos são capazes de reconhecer documentos do tipo paleográfico.

Serão apresentadas duas ferramentas de OCR. O PyTesser que é um tipo de OCR que têm como característica o reconhecimento de caracteres do tipo numéricos e padronizados, não possuindo características bem desenvolvidas para o reconhecimento de caracteres manuscritos, porém pode ser adaptados a esse tipo de escritas, em alterações no código. O outro aplicativo é o ABBYY Fine, que trata-se de um aplicativo proprietário que possui características de reconhecimento de diversos tipos de escritas inclusive manuscritas.

3.3.1 OCR PyTesser

O OCR PyTesser [78] trata-se de um módulo de Reconhecimento de Caráter Óptico padronizado, na linguagem de programação Python. Tem como característica a funcionalidade de introduzir uma imagem, convertendo a mesma para um formato aceito e fazendo uma chamada ao executável de Tesseract, que é um motor de OCR baseado em código livre para leitura de imagens dos tipos tif, jpeg, bmp e outras extensões. Esse aplicativo também permite que sejam feitas alterações em seu código para o reconhecimento de outros tipos de escritas.

Com o intuito de demonstrar a funcionalidade do sistema de OCR apresentado, foram realizados testes em algumas imagens para o reconhecimento de caracteres dos tipos padronizados (padrão ANSI), manuscritas cursivas e de escritas paleográficas.

Entretanto, foi verificado o comportamento do aplicativo na medida em que eram realizadas modificações no seu código fonte. De maneira que o mesmo conseguisse um maior número de acertos possíveis, sendo assim foi realizado um comparativo entre as imagens testadas, conforme apresentado no Figura 3-1.

3.3.1.1 Reconhecimento de imagens com escritas padronizadas (padrão ANSI).

O objetivo deste teste foi avaliar o comportamento do aplicativo para imagens com caracteres do tipo padrão ANSI. Onde, já foram apresentadas na descrição do ambiente as características utilizadas na simulação.

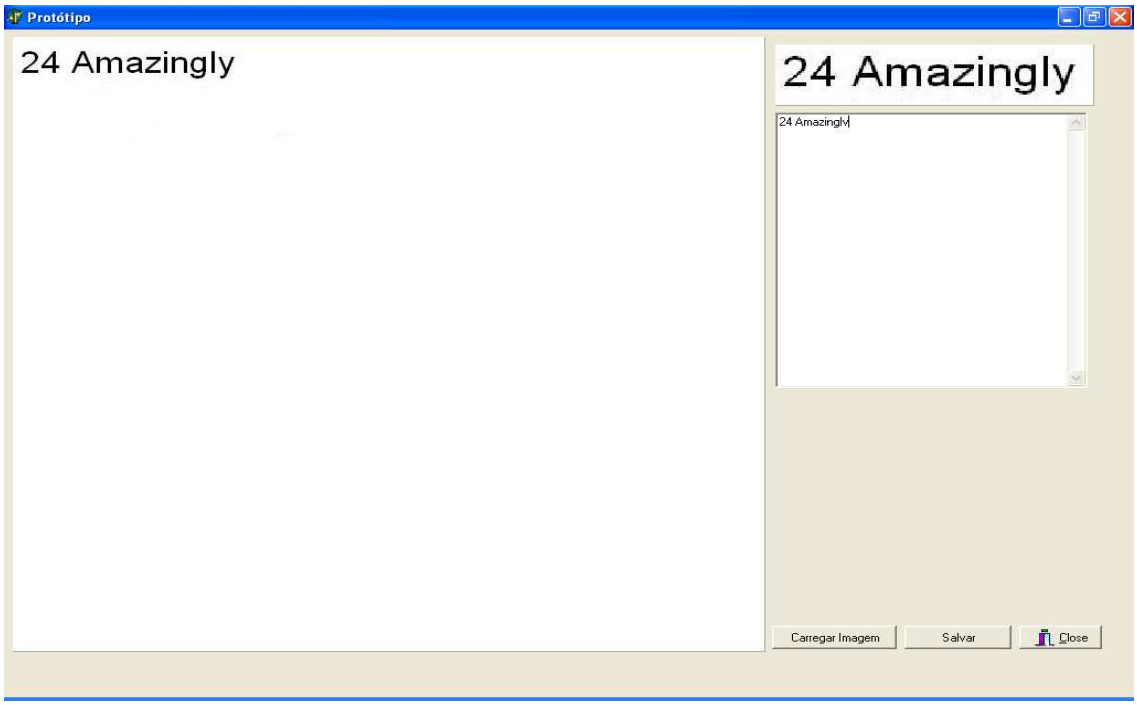

Figura 3-1 – Um dos resultados do reconhecimento de escritas padronizadas

O aplicativo Pytesser, em seu modo de analisador de caracteres, analisou a figura digitalizada (testepadrao.jpeg) e a partir daí foram extraídas as informações necessárias para a obtenção alguns resultados. Conforme demonstrado na Figura 3-1, que mostra o melhor resultado apresentado após uma seqüência de repetições e alterações no código do aplicativo.

A Figura 3-1 demonstra o resultado extraído da palavra 24Amazingly, onde o último e o melhor resultado adquirido foi ***24Amazinglv,*** podendo observar que o aplicativo não conseguiu traduzir a letra y. A análise das variações ocorridas nos testes com imagens padronizadas, também podem ser vista na Tabela 3-1.

Tabela 3-1 - Teste de reconhecimentos de caractere padronizados

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	6	5	11	24 Anr|2ingiv
2	7	4	11	24 Amr2imglv
3	8	3	11	24 Ama2imglv
4	7	6	11	Z4 AmaZim
5	8	3	11	24 AmaZimgiv
6	8	3	11	24 Ama2imgiv
7	9	2	11	24 Ama2imglv
8	9	2	11	24 Ama2imglv
9	10	1	11	24 Amazimglv
10	10	1	11	24 Amazinglv
11	9	2	11	24 Amazingl
12	10	1	11	24 Amazinglv
13	10	1	11	24 Amazinglv
14	10	1	11	24 Amazinglv
15	10	1	11	24 Amazinglv

Como representado na Tabela 3-1, podemos observar que o analisador obteve algumas variações de erros e acertos, após uma seqüência de testes realizados com o mesmo tipo de escritas, onde o mesmo sofreu variações durante os processos de leitura do documento. Além disso, do passo 12 em diante houve uma estabilidade no resultado com relação ao número de acertos. Esse é um motivo pelo o qual a Tabela 3-1 possui somente 15 repetições, desconsiderando os resultados posteriores.

Como análise, nos testes realizados com o aplicativo Pytesser para o reconhecimento de documentos padronizados, verificou-se que o mesmo obteve resultados e um comportamento, bastante satisfatório, para essa característica de documentos.

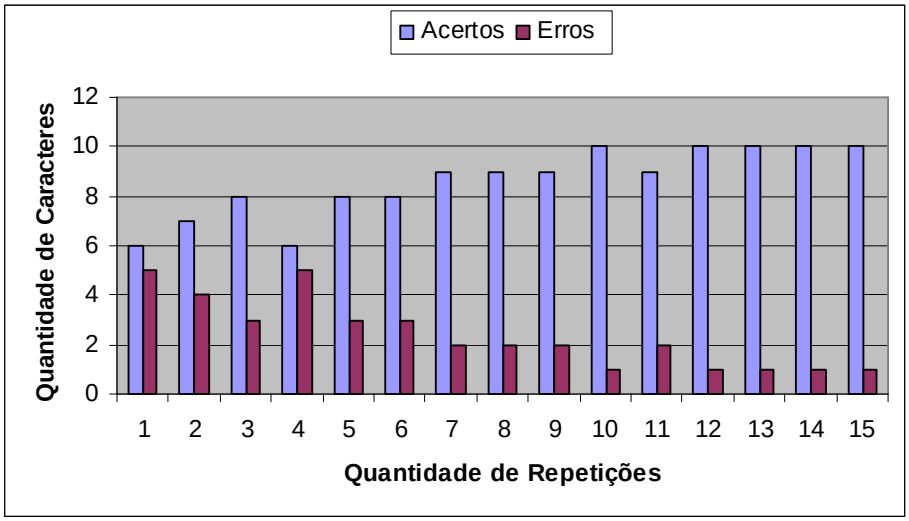

Figura 3-2 - Resultados encontrados no reconhecimento de caracteres padronizados

De acordo com o apresentado no Figura 3-2, temos picos referentes à quantidade de acertos e erros, ou seja, o número de acertos vai crescendo à medida que o número de repetições vai aumentando.

Sendo assim observando a variação dos picos, temos que nas repetições de número 4 e 11 houve uma queda na quantidade de acertos, pelo fato da modificação ocorrida no código fonte do aplicativo. Entretanto nas demais repetições o número de acertos vão aumentando e a partir da 12ª repetição em diante se mantiveram estáveis.

3.3.1.2 Reconhecimento de imagens com escritas manuscritas

O objetivo deste teste é fazer o reconhecimento de imagens com tipo de escrita manuscrita, avaliando o comportamento do aplicativo para caracteres do tipo manuscritos, que seria o tipo que mais pode se aproximar de escritas paleográficas.

As características utilizadas para as simulações com documentos do tipo manuscritos estão mais bem detalhadas na descrição do ambiente deste capítulo.

Sendo assim, a imagem foi analisada pelo aplicativo PyTesser e obteve uma variação no número de acertos e erros, decorrentes da quantidade de repetições e das modificações ocorridas no código fonte. Conforme mostrado na Tabela 3-2.

A Figura 3-3 mostra o melhor resultado extraído da palavra JOSE DA SILVA escrita de forma manuscrita.

Figura 3-3 - Um dos resultados do reconhecimento de escritas manuscritas

Tabela 3-2 mostra a seqüência de teste realizado para as escritas manuscritas.

Tabela 3-2 - Teste de leitura de documentos manuscritos

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	0	11	11	awl
2	0	11	11	awwgel
3	1	10	11	Jawwgel
4	1	10	11	J /%M4
5	3	8	11	Jvzg lie VA
6	3	8	11	Jvzg lie VA
7	3	8	11	Jvzi g;LvA
8	7	4	11	JOSF SLLVA
9	7	4	11	Josi M5ILVA
10	5	6	11	Josi 5;;.%/A
11	8	3	11	Josi A 5;;.%/A
12	8	3	11	Josi MSILVA
13	9	2	11	Jose mSILVA
14	9	2	11	Jose MSILVA
15	9	2	11	Jose MSILVA

O que se pode observar nos resultados apresentados na Tabela 3-2 foi que em alguns testes o aplicativo reconheceu as letras como um tipo de caractere especial (; % | /). Isso ocorreu devido à modificação realizada no código fonte, pois nesses casos não foram modificados uma variável que regula a freqüência dos caracteres encontrados na imagem. Além disso, a estabilidade do aplicativo passou a ser apresentada a partir a 13ª repetição, não havendo modificações posteriores no resultado, lembrando que em cada repetição o código do aplicativo era aprimorado para admitir o maior número de acertos possíveis.

O Figura 3-4 mostra a variação na quantidade de acertos e erros, para escritas do tipo manuscritas. Onde o primeiro teste realizado houve 100% de erros e à medida que os testes vão se repetindo e que o código fonte vai sendo alterado, o número de acertos tendem a aumentar até chegar um grau de satisfação. Além disso, observando a variação dos picos, temos que na repetição de número 10 houve uma queda na quantidade de acertos, fato pelo qual houve um erro no código alterando o resultado.

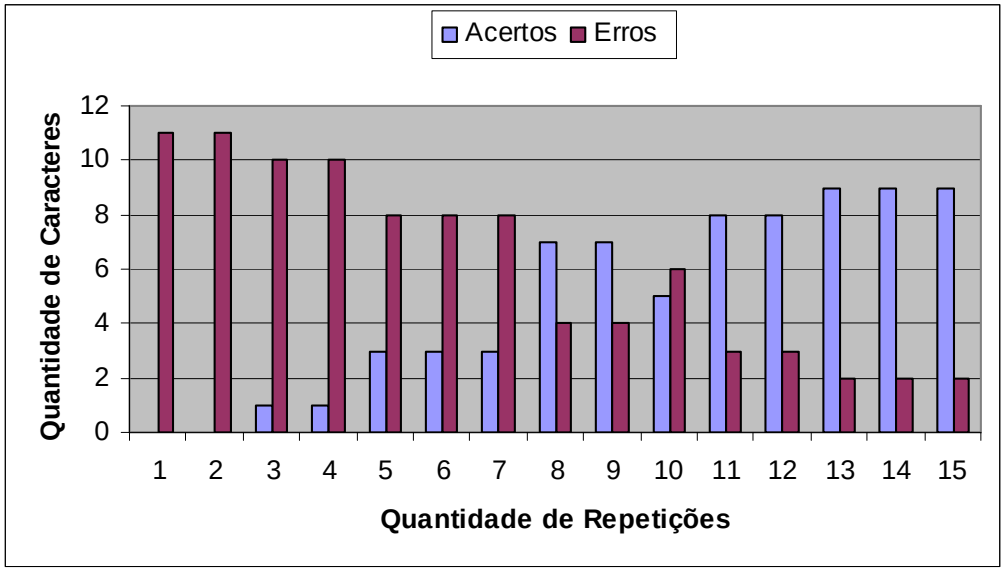

Figura 3-4 - Análise do reconhecimento de caracteres manuscritos

Como visto nos testes realizados com o aplicativo Pytesser para o reconhecimento de documentos manuscritos, verificou-se que o mesmo não obteve 100% de acertos, porém mostrou um resultado bem satisfatório, para essa característica de documentos.

3.3.1.3 Reconhecimento de imagens com escritas paleográficas

O objetivo deste teste é verificar se o aplicativo Pytesser é capaz de reconhecer imagens com tipo de escrita paleográfica.

As imagens do tipo paleográficas utilizadas para as simulações foram extraídas do Projeto Resgate, onde por sua vez, os documentos originais são extremamente antigos, geralmente dos séculos XVI e XVII. São documentos de difícil manuseio e que apesar da utilização de mecanismo e equipamentos de grande tecnologia e de alta qualidade os mesmos sofreram grande perda de qualidade após serem digitalizados. Sendo assim foram utilizados nos testes somente os documentos que pudessem ser pelo menos visíveis, com a melhor qualidade possível, e que se aproximasse das características apresentadas na descrição do ambiente, de maneira que os resultados encontrados não viessem sofrer muitas variações por causa da qualidade das imagem.

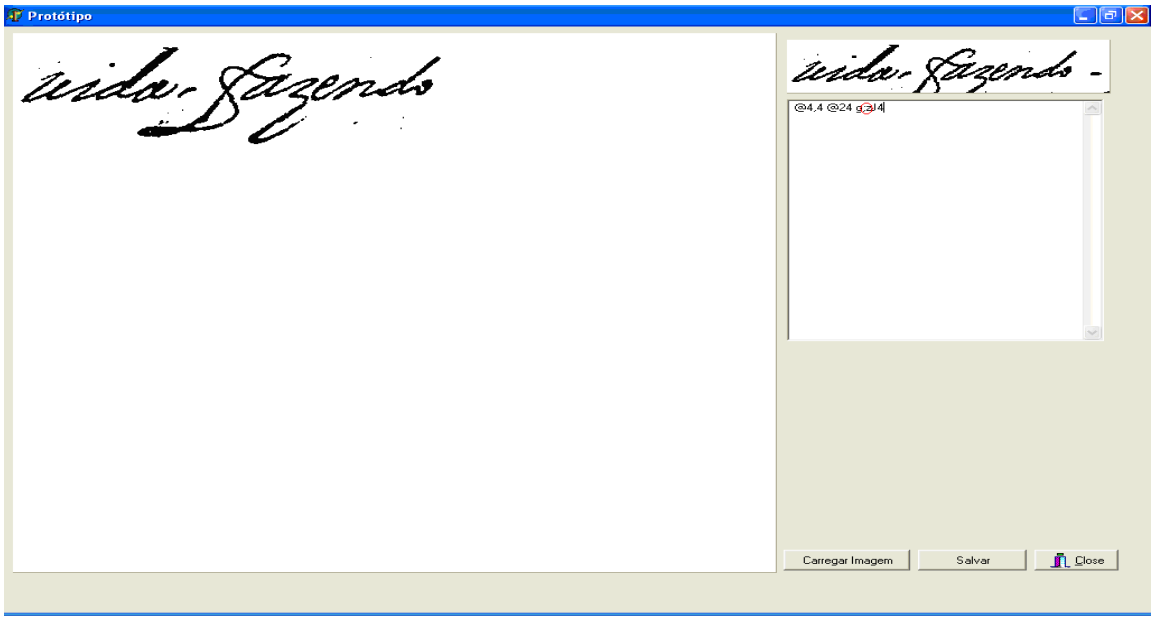

Figura 3-5 - Um dos resultados do reconhecimento de escritas paleográficas

Entretanto, como pode ser visto na Figura 3-5 que apresenta um dos resultados apresentados, entre os diversos testes realizados com uma imagem.

Esse teste foi realizado utilizando os mesmos parâmetros dos testes anteriores. Imagens como a mesma extensão, mesma quantidade de caracteres na palavra, modificações no código entre outros.

Figura 3-5 - Um dos resultados do reconhecimento de escritas paleográficas

A Figura 3-5 mostra como resultado uma seqüência de caracteres, onde apresenta a letra "z" como parte deles, porém o melhor resultado obtido, não pode ser considerado como valido, pelos seguintes motivos:

- Foram realizadas várias repetições e o resultado encontrado apareceu uma única vez, tendo em vista que o sistema possa ter lançado de forma aleatória, dando uma visão de coincidência.
- Em todas as repetições foram realizados melhorias no código e o aplicativo não repetiu em nenhuma vez os mesmos caracteres, como ocorreu nos testes anteriores.
- Foi realizado um teste verificando caractere por caractere, mesmo assim o aplicativo não foi capaz de reconhecer nenhum dos caracteres.

Tabela 3-3 - Teste de leitura de documentos paleográficos

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	0	11	11	lmslxrn hetninlcu
2	0	11	11	snuncxarmis
3	0	11	11	omnraaqmnsprd
4	0	11	11	gressosmsnes
5	0	11	11	ssdu did oqz
6	0	11	11	hciei bi!
7	0	11	11	2@!|asr%
8	0	11	11	@4,4@24gz!4
9	0	11	11	:--#~g=g
10	0	11	11	gjaia||
11	0	11	11	Josi A 5;;.%/A
12	0	11	11	___i,,?...--i
13	0	11	11	,;4?g
14	0	11	11	7%%%
15	0	11	11	9%`m%4

Tabela 3-3 mostra os resultados dos testes realizados com as escritas paleográficas, o que pode se observar, é que os testes realizados com o Pytesser não foram satisfatórios, tendo

em vista que o aplicativo não foi capaz de reconhecer se quer, algum caractere da imagem apresentada.

A Figura 3-6 representa o resultado dos testes realizados nas escritas do tipo paleográficas, onde se pode observar que o mesmo obteve como resultado 100% de erros. Além disso, mostra uma relação entre a quantidade de caracteres usados na palavra e o número de repetições, na idéia de verificar se ocorriam variações nos resultado por conta das repetições.

Entretanto o que se pode observar foi que, o PyTesser no reconhecimento de escritas paleográficas, obteve no gráfico picos constantes em número de acertos e erros, sendo 0 e 11 respectivamente.

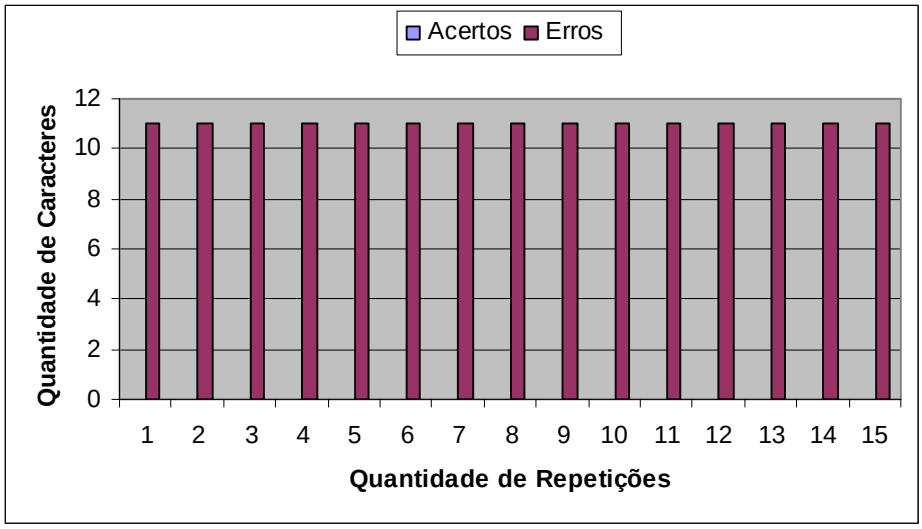

Figura 3-6 - Análise do reconhecimento de caracteres paleográficos

Sendo assim, o que podemos concluir com relação ao PyTesser, é que no reconhecimento de escritas do tipo paleográficas, apesar de várias alterações em seu código fonte e após várias simulações o mesmo não conseguiu ler nenhum caractere.

3.3.1.4 Comparatvos de resultados do OCR PyTesser

A Tabela 3-4 e o Figura 3-7 mostram um comparativo dos resultados apresentados após os teste realizados em três tipos de características de escritas, com uma quantidade minima de

15 repetições diferentes para cada tipo de documento, verificando o comportamento do aplicativo em cada repetição, em muito dos casos alterando o seu código fonte para se adequar ao tipo de escrita testada.

Podemos assim concluir que o Pytesser no reconhecimento de escritas do tipo padronizados e manuscritas, obteve um sucesso considerável. Entretanto, não pode ser utilizado para escritas paleográficas, pois não obteve resulatdos para esse tipo de documento.

Tabela 3-4 - Comparativo entre os três tipos de escritas

	Escritas padronizadas	Escritas manuscritos	Escritas Paleográficas
Acertos	10	9	0
Erros	1	2	11
Total de caracteres	11	11	11

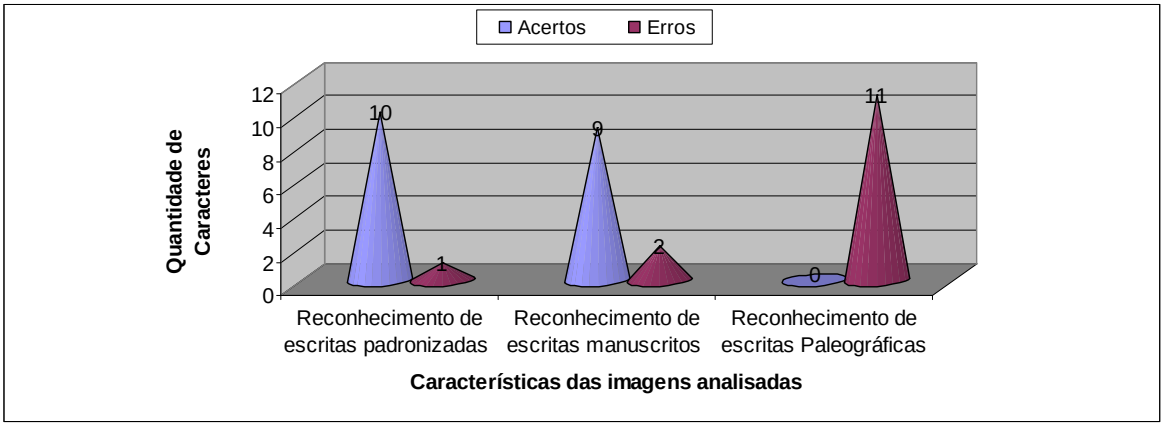

Figura 3-7 - Comparativo de acertos/erros entre os tipos de documentos analisados

A Figura 3-7 mostra uma relação entre os três tipos de documentos analisados, representando os acertos e erros conforme a característica do documento, mostrando uma variação para o reconhecimento de escritas padronizadas de 10 para 1, ou seja, de 10 acertos para 1 erro, nas escritas manuscritos ficou de 9 acertos para 2 erros. Já nas paleográficas não houve acertos, um total de 11 erros, ou seja, 100% de erros.

3.3.2 OCR ABBYY Fine Reader 9.0

O OCR ABBYY Fine Reader 9.0 [79], trata-se de um poderoso software proprietário para Reconhecimento de Caráter Óptico. Trata-se de um aplicativo baseado em uma inovação de novas plataformas de reconhecimento de palavras e comportando uma Tecnologia de Reconhecimento de Documento Adaptável (ADRT™) [80]. Além disso, é inteligente suficiente para extrair automaticamente arquivos em formato de documento nativo, tornando–os de forma de documentos editáveis, tendo capacidade de extração das palavras de documentos com os formatos PDF, Jpeg, tiff, png, entre outros, ou até mesmo de documentos digitalizados na hora.

Apesar do ABBYY Fine tratar-se de um software com características bem avançadas, os mesmos testes realizados no Pytesser para imagens com diferentes tipos de escritas, foram realizados no ABBYY Fine Reader. Conforme apresentado na descrição do ambiente.

Sendo assim, serão realizadas simulações em documentos padronizados, manuscritos e com escritas paleográficas, verificando o seu comportamento, de maneira que consiga um maior número de acertos possíveis, realizando um comparativo entre os tipos de imagens. Porém não teremos a possibilidade da alteração do seu código fonte.

3.3.2.1 Reconhecimento de imagens com escritas padronizadas (padrão ANSI).

O objetivo deste teste foi avaliar o comportamento do aplicativo ABBYY Fine para imagens com caracteres do tipo padrão ANSI.

A Figura 3-8 mostra a tela do aplicativo ABBYY Fine, após a extração de uma imagem digitalizada, no lado esquerdo da tela tem-se a imagem original de forma segmentada, embaixo tem a demonstração da palavra que será traduzida, e no lado direito mostra o resultado da imagem após sua extração.

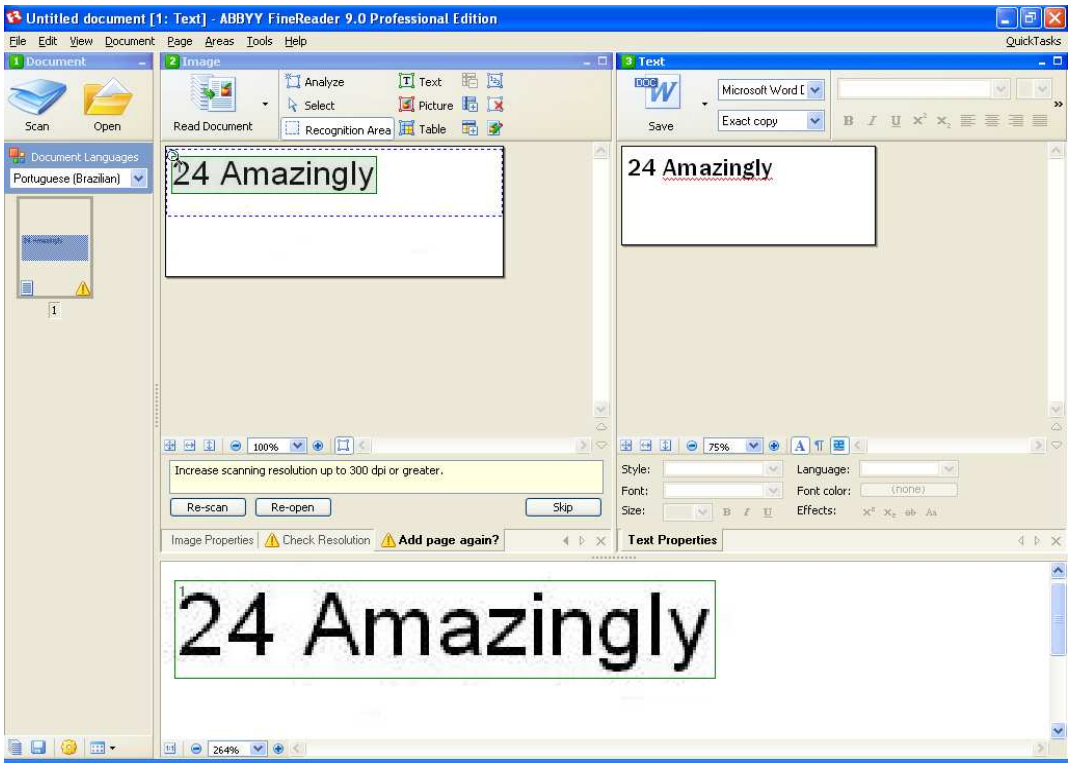

Figura 3-8 - Um dos resultados do reconhecimento de palavras em escritas padronizadas

O último resultado obtido, das simulações relalizadas na Tabela 3-5, está demosntrado na Figura 3-8, onde o aplicativo analisou a palavra 24Amazingly e o resultado foi justamente o mesmo encontrado na imagem original, não sofrendo variações nos caracteres analisados.

Tabela 3-5 - Teste de leitura de documentos padronizados

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	11	0	11	24 Amazingly
2	11	0	11	24 Amazingly
3	11	0	11	24 Amazingly
4	11	0	11	24 Amazingly
5	11	0	11	24 Amazingly
6	11	0	11	24 Amazingly
7	11	0	11	24 Amazingly
8	11	0	11	24 Amazingly
9	11	0	11	24 Amazingly
10	11	0	11	24 Amazingly
11	11	0	11	24 Amazingly
12	11	0	11	24 Amazingly
13	11	0	11	24 Amazingly
14	11	0	11	24 Amazingly
15	11	0	11	24 Amazingly

Como visto na Tabela 3-5 de resultados do programa, pode-se observar que o analisador obteve 100% de acertos em todas as repetições para o reconhecimento de caracteres padronizados, não obtendo variações em nenhuma das repetições.

A Figura 3-9 mostra uma relação entre a quantidade de caracteres usados na palavra e o número de repetições, verificando se ocorriam variações nos resultados por conta das repetições. Entretanto, o que se pode observar foi que o ABBYY Fine no reconhecimento de escritas padronizadas, obteve no gráfico picos constantes em número de acertos e erros, 11 acertos e nenhum erro.

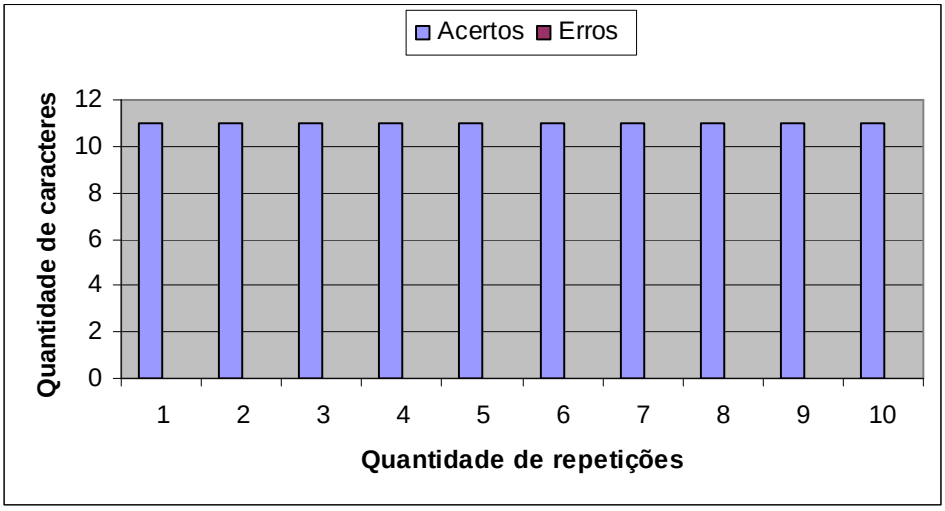

Figura 3-9 - Análise do reconhecimento de escritas padronizadas

3.3.2.2 Reconhecimento de imagens com escritas manuscritas

Nesse tipo de escrita, o objetivo deste teste foi também avalizar o comportamento do aplicativo ABBYY Fine, verificando as variações dos resultados, conforme as repetições ocorridas.

Para realização desta simulação, obedecemos ao mesmo padrão de imagem utilizado na descrição do ambiente. Já no processo de digitalização da imagem, utilizamos um mecanismo existente no próprio ABBYY Fine, onde é possível digitalizar a imagem direta para o aplicativo, sem precisar salva-la em disco.

Esse aplicativo possui uma característica bastante interessante, apesar de ser um sistema “travado” não existindo a possibilidade de alteração em seu codigo. A medida que vamos segmentado as imagens e executando novas leituras os resultados vão se modificando e tornando mais proximo do real, como se utilizasse um mecanismo de aprendizagem.

A Figura 3-10 mostra a tela de um dos resultado encontrado pelo aplicativo ABBYY Fine, após a digitalização direta de uma imagem, onde, do lado esquerdo aparece à imagem recém digitalizada com a palavra “totalmente selecionada”, no lado direito mostra o resultado da imagem após sua extração. que conforme as funcionalidades existentes.

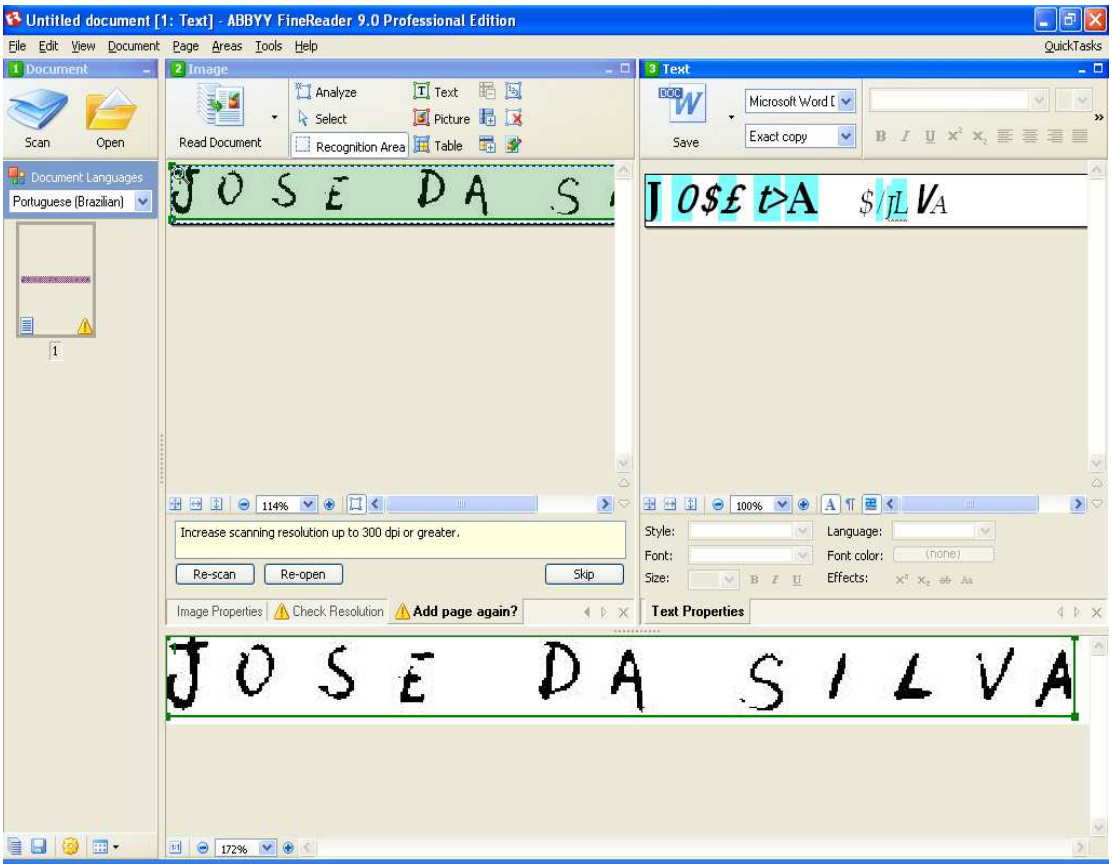

Figura 3-10 – Um dos resultados do reconhecimento de palavras em escritas manuscritas

Tabela 3-6 - Teste de leitura de documentos manuscritos

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	8	3	11	Jo$£ DA $ilva
2	8	3	11	Jo$£ DA $ilva
3	8	3	11	Jo$£ DA $ilva

4	10	1	11	JoS£ DA Silva
5	10	1	11	JoS£ DA Silva
6	11	0	11	Jose da Silva
7	11	0	11	Jose da Silva
8	11	0	11	Jose da Silva
9	11	0	11	Jose da Silva
10	11	0	11	Jose da Silva
11	11	0	11	Jose da Silva
12	11	0	11	Jose da Silva
13	11	0	11	Jose da Silva
14	11	0	11	Jose da Silva
15	11	0	11	José da Silva

A Tabela 3-6 apresenta o resultado do reconhecimento de escrita do tipo manuscrita, podendo observar que o aplicativo após a sexta repetição obteve uma constante em relação ao número de acertos.

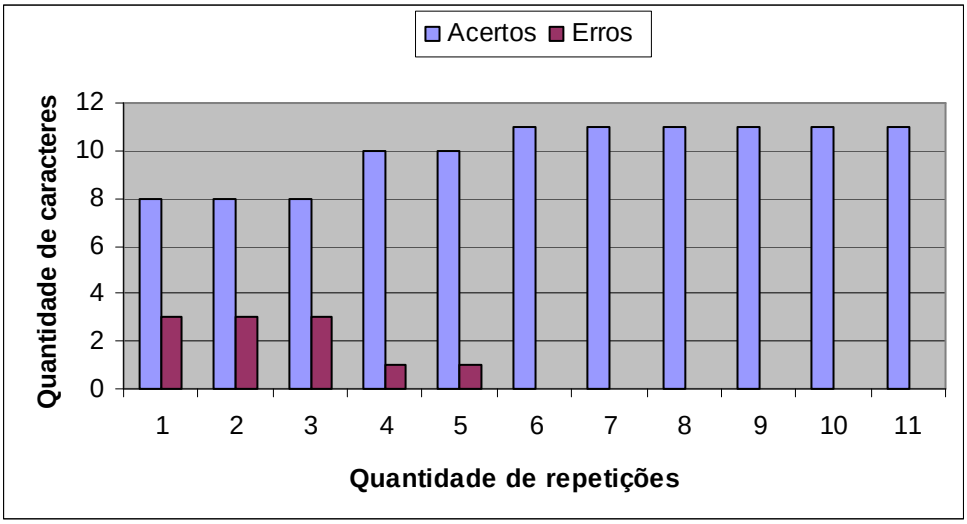

Figura 3-11 - Análise do reconhecimento de escritas manuscritas

A Figura 3-11 mostra os resultados apresentados extraídos de Tabela 3-6, representando uma relação entre a quantidade de caracteres com o número de repetições, iniciando na primeira repetição com uma quantidade de 8 acertos, que vai crescendo conforme o número de repetições. Entre a 2ª e a 3ª repetição ocorre um aumento para 10 acertos, que logo em seguida alcança os 100% de acertos.

3.3.2.3 Reconhecimento de imagens com escritas palogragáficas

As simulações anteriores, mostraram a utilização do ABBYY Fine no tratamento de escritas padronizadas e manuscritas, o objetivo deste teste agora é verificar se o aplicativo é capaz de reconhecer imagens com tipo de escrita paleográfica da mesma maneira que ocorreu nos testes anteriores.

As imagens do tipo paleográficas utilizadas para as simulações, também foram extraídas do Projeto Resgate, conforme foram utilizadas no aplicativo PyTesser.

Sendo assim, a Figura 3-12 apresenta o melhor resultado de todas as imagens e simulações realizadas, utilizando os mesmos parâmetros da descrição do ambiente.

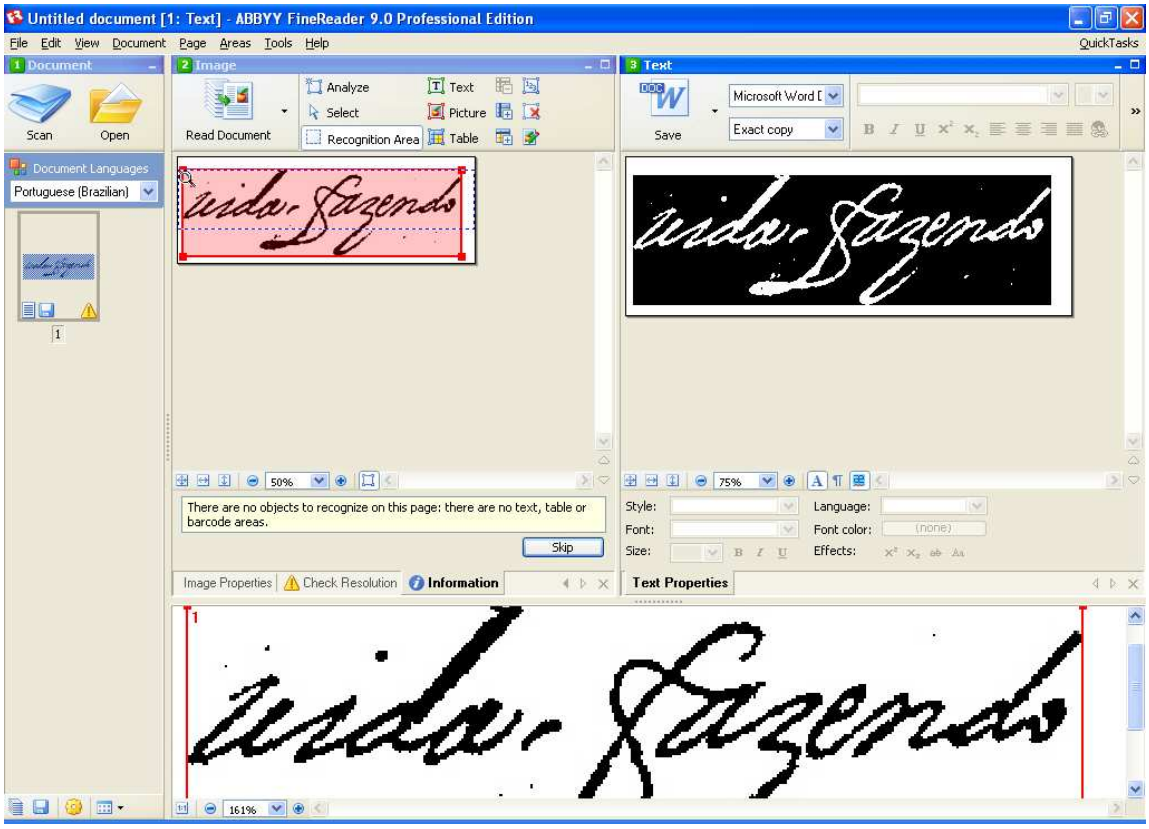

Figura 3-12 – Resultado do reconhecimento de escritas paleográficas

Na Figura 3-12 temos o resultado da imagem após sua extração utilizando o aplicativo ABBYY Fine. O resultado apresentado simula uma forma bem definida, porém conforme os testes mostrados na Tabela 3-7 pode-se verificar que em todos os resultados o aplicativo reconheceu a imagem original como uma imagem e não como caracteres.

Tabela 3-7 - Teste de leitura de documentos paleográficos

Repetições	Acertos	Erros	Quantidade de caracteres	Resultados encontrados
1	0	11	11	Imagem
2	0	11	11	Imagem
3	0	11	11	Imagem
4	0	11	11	Imagem
5	0	11	11	Imagem
6	0	11	11	Imagem
7	0	11	11	Imagem
8	0	11	11	Imagem
9	0	11	11	Imagem
10	0	11	11	Imagem
11	0	11	11	Imagem
12	0	11	11	Imagem
13	0	11	11	Imagem
14	0	11	11	Imagem
15	0	11	11	Imagem

O ABBYY Fine, possui uma característica muito importante, que é a capacidade de reconhecer uma escrita dentro de uma imagem, porém no caso das escritas paleográficas, mesmo utilizando esse recurso o mesmo não foi capaz de reconhecer algum caractere. Podemos verificar que todos os resultados foram os mesmos.

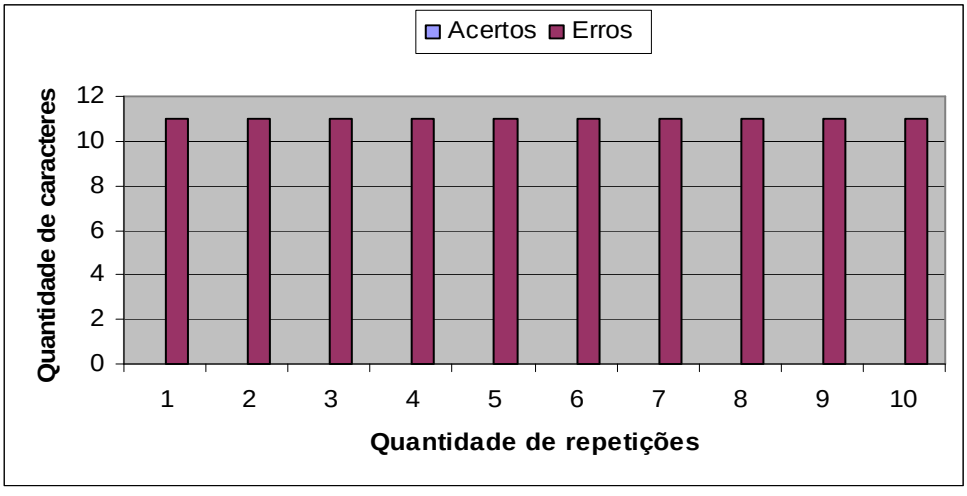

Figura 3-13 - Análise do reconhecimento de escritas paleográficas

A Figura 3-13 representa o resultado dos testes realizados nas escritas do tipo paleográficas, onde se pode observar que não houve acertos. O resultado inicializa com nenhum acerto, e à medida que as repetições vão ocorrendo à quantidade de acerto segue constante. Da mesma maneira ocorre para quantidade de erros, que nesse caso é de 100% obtendo no gráfico, picos constantes em número de acertos e erros, 0 e 11 respectivamente.

Dessa forma o que se pode concluir sobre o aplicativo ABBYY Fine, para o tratamento de imagens com escritas padronizadas e manuscritos, ele se comportou de maneira satisfatória, já no tratamento de imagens paleográficas pode se considerar que o ABBYY Fine não obteve resultados positivos. Possuindo características bastante interessantes, porém não para esse tipo escrita.

3.3.2.4 Comparatvos de resultados do ABBYY Finer Reader

O comparativo dos resultados apresentados no ABBYY Fine após os teste resalizados em três tipos diferentes de características de escritas, com uma quantidade minima de 10 repetições diferentes para cada tipo de documento, podem ser melhor visualizados na Tabela 3-8

Tabela 3-8 - Comparativo entre os três tipos de escritas

	Escritas padronizadas	Escritas manuscritos	Escritas Paleográficas
Acertos	11	11	0
Erros	0	0	11
Total de caracteres	11	11	11

A Figura 3-14 mostra uma relação entre os três tipos de documentos analisados, representando a variação de acertos e erros, conforme a característica do documento, onde mostra uma variação para o reconhecimento de escritas padronizadas de 11 para 0, ou seja, de 11 acertos para nenhum erro, ou seja, 100% de acertos, nas escritas manuscritos a variação ficou também de 100% sendo 11 acertos para nenhum erro. Já nas paleográficas não houve acertos para um todas de 11 erros, ou seja, 100% de erros.

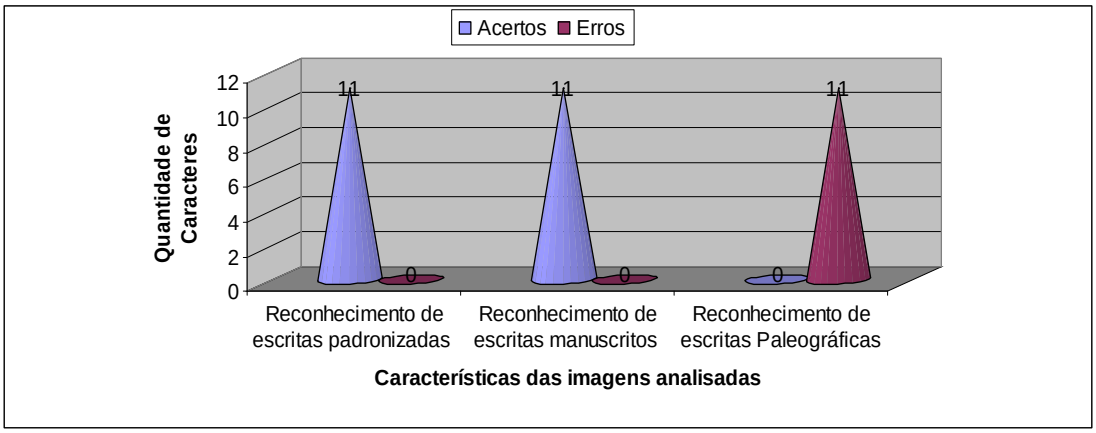

Figura 3-14 - Comparativo de acertos/erros entre os tipos de documentos analisados

3.3.3 Comparativo dos Resultados Apresentados

Os dois aplicativos utilizados para reconhecimento de caracteres através de uma imagem foram bastante eficientes no reconhecimento de caracteres padronizados e manuscritos, possuindo algumas diferenças de resultados, porém bem satisfatório.

Entretanto, no reconhecimento de escritas paleográficas, tanto o aplicativo PyTesser, quanto o ABBYY Fine não obtiveram resultados positivos em relação a número de acertos. O PyTesser não foi capaz reconhecer caracteres paleográficos, porém conseguiu reconhecer os caracteres como caráter. Já o ABBYY Fine em todos os testes realizado em imagens paleográficas não conseguiu reconhecer as palavras em forma de caráter e sim em forma de imagem.

A tabela 3-9 e Figura 3-15 mostram um comparativo para o número de acertos entre os dois aplicativos, avaliando o comportamento para os três tipos de escritas utilizadas.

Tabela 3-9 – Resultado do comparativo entre os aplicativos

	Escritas padronizadas	Escritas manuscritos	Escritas Paleográficas
Pytesser	10	9	0
ABBYY Fine	11	11	0
Total de caracteres	11	11	11

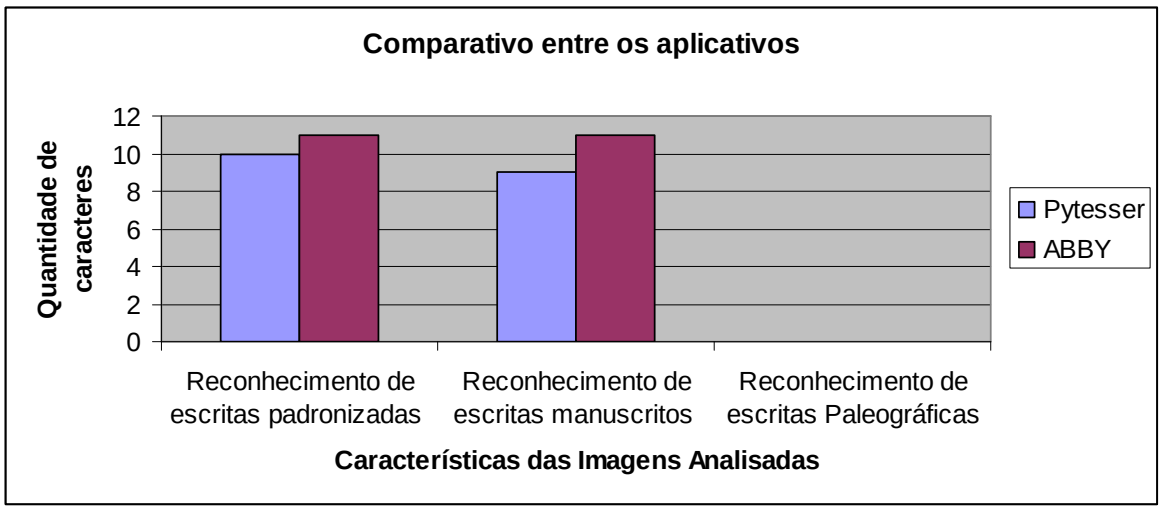

Figura 3-15 - Resultado do comparativo entre os aplicativos

3.4 SÍNTESE DO CAPÍTULO

Sistemas baseados em mecanismo de OCR ainda são motivos de estudos nas áreas de ciência da computação e engenharia. O seu uso já é bastante comum, porém ainda é restrito em alguns casos específicos. Entretanto, estudiosos das mais diversas áreas relacionadas à inteligência artificial, computação, engenharia e áreas afins, tem demonstrado através de trabalhos e pesquisas relacionadas ao tema, o grande potencial do uso de OCR e os benefícios que podem ser alcançados.

Apesar das demonstrações e trabalhos realizados, existem grandes problemas ainda relacionados, especificamente no que se relaciona a escritas o tipo paleográfico.

Como um primeiro exemplo tem-se a falta de padronização e o método utilizado para esse tipo de escrita. Os motivos desta dificuldade de padronização são inúmeros. Como exemplo pode-se citar que as características das escritas são referentes aos séculos XVI e XVII, onde os escritores seguiam um modelo de escrita da época, que em sua grande maioria os documentos eram escritos utilizando penas ao invés de canetas. Dificultando na adaptação do tipo de escrita e até mesmo na leitura dos mesmos, uma vez que as penas geravam borrões.

Um segundo exemplo pode ser dado com relação à qualidade dos documentos, pois são documentos muito antigos, onde ocorreu uma perda de qualidade em suas digitalizações. Gerando grandes dificuldades para um sistema de OCR interpretara. Um dos problemas de uma escrita paleográfica se dá também pala forma de escrita, uma vez que as palavras são bem juntas, sendo difícil de identificar aonde acaba uma e começa a outra.

Sendo assim, nesse capítulo foi apresentada uma série de simulações, em diversas imagens para diversos tipos de escritas, com o objetivo que os softwares utilizados fossem capazes de reconhecer escritas do tipo paleográficas.

Entretanto, podemos concluir que existe uma viabilidade da implementação de um mecanismo que consiga reconhecer palavra do tipo de escritas paleográficas, de maneira foram testados vários mecanismos e nenhum deles obteve resultados válidos ou coerentes para o reconhecimento do texto escrito.

4 - PROPOSTA DE ARQUITETURA E RESULTADOS OBTIDOS

Com o objetivo de validar a proposta de arquitetura descrita neste capítulo e de gerar parâmetros de avaliação para cada módulo apresentado, foi necessária a implementação de um protótipo que demonstrasse algumas métricas de funcionamento da arquitetura sugerida. Além disso, é necessário que o mesmo consiga simular um mecanismo de "OCR paleográfico", com algumas características utilizadas em outros sistemas de "OCR convencional".

O protótipo utilizado para representar a arquitetura foi desenvolvido utilizando a linguagem de programação Python, com auxilio de alguns programas como o MatLab. Além disso, foram utilizados alguns plugins em sua maioria com licenças gratuitas ou de código aberto, para interação do Python com o MatLab. Outro ponto avaliado para escolha dos programas foi à capacidade que ele possui para solucionar determinados problemas, que serão mostrados com maior detalhe no item 4.3 deste capítulo.

A metodologia de desenvolvimento do sistema seguiu métodos e padrões da engenharia de software convencional [80], como a decomposição, abstração e organização, utilizando-se ainda de programação orientada a objetos e tecnologias de redes neurais.

Desta forma, será tratada neste capítulo, a arquitetura lógica, ambiente de desenvolvimento, a instalação e configuração da aplicação para o Projeto, descrição dos principais módulos, os diagramas de casos de uso, bem como a implementação e alguns resultados encontrados.

Após a descrição da arquitetura lógica, será mostrada a descrição do ambiente de desenvolvimento do protótipo, mostrando as ferramentas utilizadas, além disso, descreveremos os cenários de testes.

4.1 REPRESENTAÇÃO DA ARQUITETURA

Esta seção apresenta a estrutura de componentes da arquitetura proposta neste trabalho para o desenvolvimento de um OCR paleográfico. A arquitetura proposta foi divida em seis módulos distintos. Estes são interligados de maneira que teremos: um módulo de interface com o usuário (GUI), um módulo principal, um de pré processamento, um de segmentação das imagens, um para extração de características e por fim um classificador. A Tabela 4-1 mostra todos os módulos com suas respectivas funções e os que foram devidamente implementados no protótipo.

Tabela 4-1 – Descrição dos módulos apresentado na arquitetura

Nome	Função	Status do Protótipo
Módulo de interface com o usuário – GUI	Apresenta a interface gráfica com usuário.	Implementado
Módulo principal	Interação com os demais módulos realizando um papel de gerente.	Implementado
Módulo de pré processamento	Tratamento da imagem	Não implementado
Módulo de Segmentação	Recebe imagens já processadas e realiza a segmentação em palavras.	Não implementado, porém simula de forma manual
Módulo de extração de características	Avalia e extrai características relevantes existentes de uma imagem.	Parcialmente Implementado utilizando recursos do Matlab
Módulo de classificação	Treina a rede neural, classificando e apresentando os resultados para o módulo principal.	Implementado

A Figura 4-1 trata-se de uma arquitetura em módulos que ilustra os módulos (subsistemas) de um sistema de OCR. Para a apresentação desse diagrama (definição da arquitetura) utilizou-se alguns padrões de engenharia de software definido como design parttern (padrão de projeto) facade e Singleton [86]. Sendo assim, cada módulo deve implementar uma classe facade, com a finalidade de simplificar o uso e facilitar o entendimento de cada módulo, além de reduzir a dependência entre eles. O objetivo de uso de padrões nesta arquitetura é fornecer modelos reusáveis para o desenvolvimento de um sistema de OCR. A interface que será apresentada por cada facade será explicada abaixo.

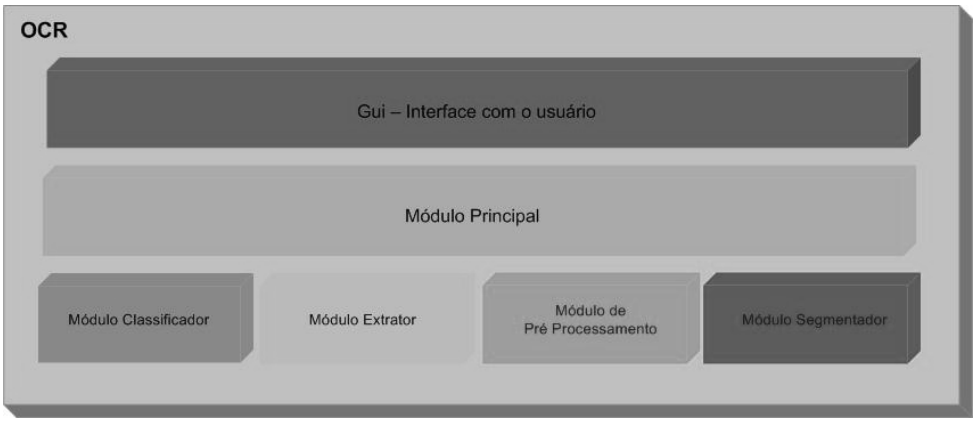

Figura 4-1 – Arquitetura em módulos para um sistema de OCR

4.1.1 Módulo de interface gráfica com o usuário

Esse módulo tem o objetivo de promover uma interface gráfica com o usuário, apresentando resultados extraídos de outros módulos. Além disso, é através deste módulo que se tem a possibilidade de realizar algumas manipulações na imagem como: visualização da imagem, recortes, buscas, adicionar padrões de treinamento para a rede neural. Sendo assim o mesmo irá possuir como parâmetro de entrada o caminho das imagens a ser tratada e como saída a visualização da imagem selecionada.

Em relação à troca de informações existente nesse módulo, o mesmo se comunica somente com o módulo principal onde este se comunica com os demais módulos, que veremos com maiores detalhe adiante.

Os serviços de um OCR Paleográfico podem ser fornecidos aos usuários via um protocolo específico, como por exemplo: desktop ou até mesmo utilizando uma interface que implemente serviços web (Web Services).

4.1.2 Módulo principal

Este módulo tem como funcionalidade a interação dos mecanismos propostos, possuindo uma característica de gerência dos demais módulos, realizando um papel fundamental para a arquitetura apresentada. A comunicação com este módulo é feita através da interface OcrFacade.

As tarefas do módulo principal são executadas pela classe OcrManager, que por sua vez repassa as funções executadas para a interface OcrFacade que tem a função de interagir com os demais módulos para atender às diversas solicitações como: classificar e treinar a rede neural como mostra a Figura 4-2.

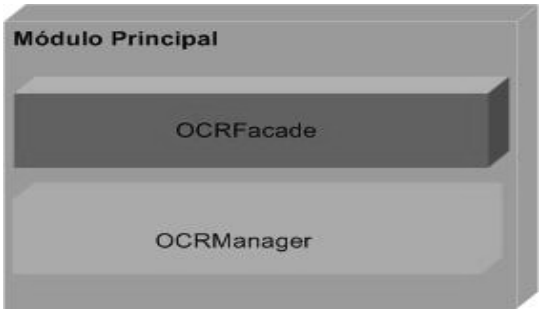

Figura 4-2 - Subdivisão do módulo principal

4.1.3 Módulo de pré processamento

Este módulo tem como funcionalidade realizar tratamentos nas imagens. A comunicação com este módulo é feita através da interface ProcessadorFacade.

As tarefas do módulo de pré processamento são executadas pelo processador, conforme demonstradas a seguir.

A classe processador irá tratar as imagens realizando um trabalho de regularização. Sendo assim o módulo de pré processamento deve possui funções para o tratamento das imagens, tais como: conversão da imagem para uma escala de cinza, retirar ruídos, realçar contorno e transformar a imagem em binário, com a finalidade de auxiliar na extração de características e o auxilio na segmentação.

- ***Converter para escala de cinza:*** Essa funcionalidade tem como objetivo converter uma imagem RGB para níveis de cinza, que possibilita uma análise estatística da distribuição de tons de cinza, que por sua vez pode ser utilizada para extração das características da imagem.

- ***Retirar ruídos:*** uma imagem digitalizada apresenta algumas irregularidades no qual chamamos de ruídos, que dificultam a classificação e o reconhecimento da imagem. Sendo assim essa funcionalidade tem por objetivo eliminar o máximo possível de ruídos existentes na imagem.
- ***Realçar contorno:*** Essa funcionalidade tem como objetivo realçar os contornos da palavra para que possa começar a identificar uma característica da escrita. Além disso, é importante a utilização de técnicas especificas para realização desse método.
- ***Imagem binária:*** Essa funcionalidade tem a função converter uma imagem para binário com objetivo facilitar a extração características nela existente.

Na Figura 4-3 temos a representação da subdivisão do módulo de pré processamento. Sendo assim, o módulo aqui apresentado tem como funcionalidade o tratamento das imagens, retirando o máximo de imperfeições encontradas nas mesmas e criando características para um melhor reconhecimento através da rede neural.

Figura 4-3 - Subdivisão do módulo de pré processamento

4.1.4 Módulo de segmentação

O processo de reconhecimento da escrita manuscrita, cursiva ou não, ou de texto impresso, inclui diversas fases, que vão desde a análise dos caracteres individuais, até a utilização de informações léxicas, sintáticas, semânticas e contextuais, para a validação do texto como

um todo. Geralmente, a primeira fase é constituída pela segmentação da imagem, que vem a ser uma das tarefas mais complexas e relevantes do processo.

A segmentação de caracteres cursivos precisa lidar com problemas como a segmentação de caracteres inclinados, sobrescritos e com traços conectados. Algumas técnicas existentes utilizam a extração de características baseadas nos dígitos, através de processos que refinam a imagem, afinando o caractere [57]. Existem ainda processos que geram vetores de características, baseando-se nas características das curvas que formam os caracteres, tais como a direção dos segmentos, pontos terminais, interseções e ciclos [59]. Além disso, existem técnicas de segmentação por histogramas [58], que representa uma estrutura que armazena o resultado da projeção da imagem sobre cada uma das duas dimensões nele existentes.

Entretanto, o módulo de segmentação é responsável por tomar cada uma das regiões que devem ser reconhecidas e quebrá-las em partes menores, cada parte contendo a imagem de apenas uma palavra ou um caráter. A comunicação com este módulo é feita através da interface SegmentarFacade. Sendo assim as tarefas do módulo de segmentação são executadas pelo subsistema segmentador, como mostra a Figura 4-4.

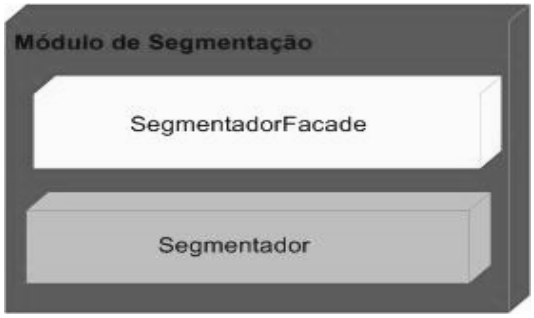

Figura 4-4 - Subdivisão do módulo de segmentação

4.1.5 Módulo de extração de características

Os chamados métodos de extração de características possuem habilidades de extrair características das imagens e também de reduzir a dimensionalidade dos dados gerando assim o chamado vetor de características.

Os métodos de extração de características que podem ser usados ou levados em consideração para um documento do tipo paleográfico, seguem as seguintes características: o número de pixels, centoid, elliptical parentenres, Skewness, entre outros. Mais detalhes sobre as mesmas podem ser consultados nas referências [57] [81] [82] [83].

Entretanto, o módulo de extração de característica apresentado na arquitetura é responsável pela tarefa de extrair característica de uma imagem já segmentada. A comunicação com este módulo é feita através da interface ExtratorFacade. As tarefas do extrator de característica são executadas pelo subsistema extrator, como mostra a Figura 4-5.

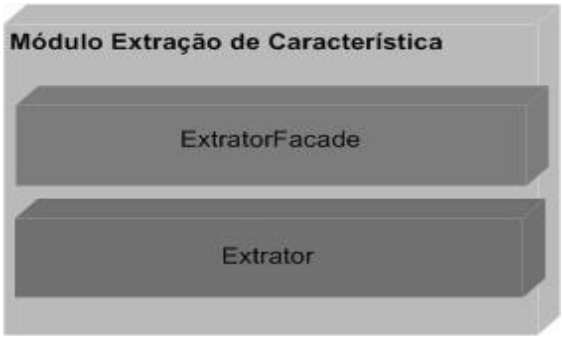

Figura 4-5 - Subdivisão do módulo de extração de características

4.1.6 Módulo de classificação

Após a segmentação das palavras e extração de características, tem-se uma seqüência de processos que tem como objetivo classificar as imagens baseado em suas características. Sendo assim, o módulo de classificação define e executa as funções uma rede neural artificial baseada no aprendizado. A comunicação com este módulo é feita através da interface ClassificadorFacade e as tarefas do módulo Classificar são executadas pelo subsistema classificador, como mostra a Figura 4-6.

Figura 4-6 – Subdivisão do módulo classificador

4.1.7 Inter-Relacionamento entre os módulos

Esta seção apresenta os inter-relacionamentos entre os módulos componentes da arquitetura proposta. Os inter-relacionamentos entre os componentes da arquitetura evidenciam a funcionalidade do módulo principal onde mostra a comunicação entre os módulos do sistema que será criado a partir desta arquitetura. A identificação destes inter-relacionamentos facilita o entendimento do comportamento do sistema gerado.

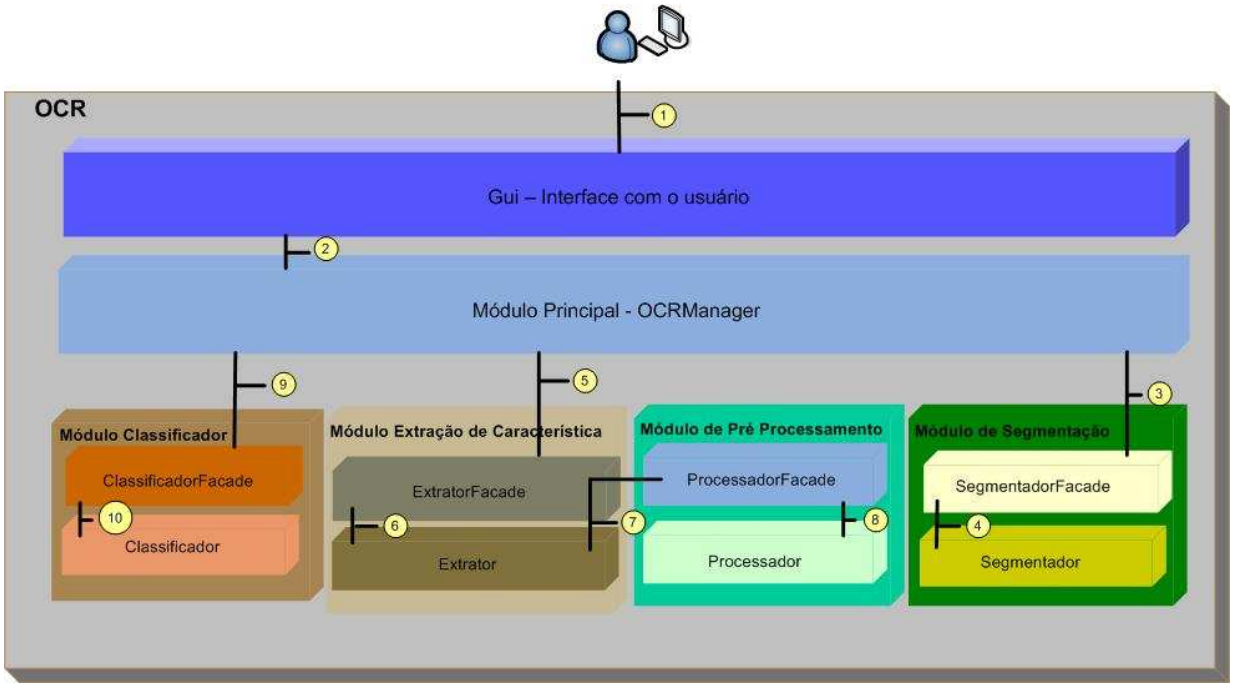

Figura 4-7 - Detalhamento dos inter-relacionamentos da arquitetura

Os exemplos da Figura 4-7 a seguir evidenciam os inter-relacionamentos entre os módulos da arquitetura de um OCR. Podem ser vistos os passos executados por um OCR para responder uma requisição. O primeiro passo, identificado pelo número 1, é a requisição do cliente ao sistema. O cliente solicita uma imagem, dada seu local ou sua URL. O sistema recebe a requisição através da interface GUI. Esta, por sua vez, comunica-se com o módulo OcrManager (número 2) para solicitar as requisições.

Para isso, a requisição é enviada para interface SegmentadorFacade (número3). Este envia a para a classe (subsistema) segmentador (número 4) que realiza a segmentação da imagem e retorna o resultado para o OcrManager, que envia para módulo Extração de característica fazendo a comunicação com a interface ExtratorFacade (número 5) que envia para a classe (subsistema) extrator, responsável por extrair características na imagem (número 6). Que por sua vez envia para o módulo de pré processamento junto a interface processadorFacade (número 7) que envia para classe processador (número 8) e retornar o resultado ao módulo principal.

Por fim, após a imagem já extraída as características é enviadas para o módulo classificador (número 9), na qual as funções é recebida pela interface ClassificadorFacade (número 10) que por sua vez repassa as funções para a classe classificador que executa o treinamento da rede neural.

4.1.8 Padrões utilizados

4.1.8.1 Padrão Facade

O Facade (fachada) é um padrão de projeto estrutural de engenharia de software [86] utilizado para fornecer uma interface unificada para os módulos de um sistema. O objetivo deste padrão é definir uma interface de alto nível para facilitar o uso das funcionalidades do módulo.

A arquitetura proposta nesta dissertação está organizada em camadas compostas por módulos. Um objetivo comum entre projetos orientados a objetos é diminuir a

comunicação e as dependências entre as classes do sistema (reduzir o acoplamento), de forma a facilitar a manutenção do sistema. Suponhamos a situação de alto acoplamento mostrada pela Figura 4-8, onde no modelo (a) os módulos X, Y e Z acessam diretamente as classes A1, A2 e A3 do módulo A. no modelo (b) é possível perceber a redução do acoplamento através do uso do padrão

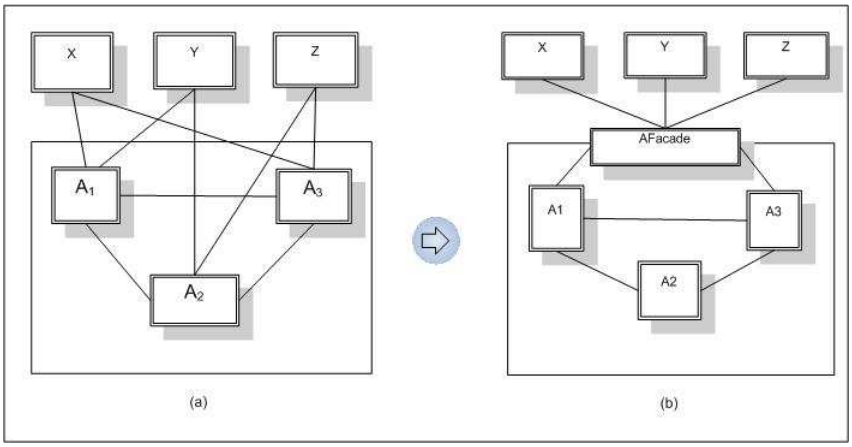

Figura 4-8 - Uso do padrão de projeto Facade

4.1.8.2 Padrão Singleton

O padrão Singleton [87] é um padrão de projeto de criação de objetos. O objetivo é garantir uma instância única e acessível de forma global e uniforme para toda classe que implemente este padrão. Neste contexto, este padrão foi usado para tornar a interface de cada módulo da arquitetura única e acessível de forma global.

Outra forma de garantir o acesso global é o uso de uma variável global. Porém a variável global não garante a unicidade. A solução proposta pelo padrão Singleton é tornar a própria classe responsável pela sua única instância. A classe que implementa o padrão Singleton garante o acesso à sua instância e ainda intercepta as requisições para criação de novos objetos, garantindo que nenhuma outra instância seja criada.

4.2 DESENVOLVIMENTO DO PROTÓTIPO

Para desenvolvimento do protótipo aqui proposto, seguiu-se o modelo de arquitetura apresentada no item 4.1, utilizando todas as ferramentas descritas no item 4.3, de maneira que o mesmo tem como característica simular o funcionamento de um OCR paleográfico.

A divisão da arquitetura em módulos deve-se as características individuais de cada um, ou seja, um módulo que verifica as imagens, um que extrai as caracterizas entre outros, sendo cada um para uma função especifica. Entretanto para representação dessa arquitetura foi desenvolvido um protótipo que apresenta as funcionalidades de treinamento da rede neural e reconhecimento da imagem.

A seguir veremos o diagrama de classes implementado no protótipo, identificando as funcionalidades de cada módulo desenvolvido e onde reside a sua inteligência. Além disso, critérios para a tomada de decisão e mecanismos gerais de funcionamento.

O diagrama de classes representado pela Figura 4-9 segue a arquitetura apresentada no item 4.1, porém sem a presença de alguns módulos, onde os mesmos não foram devidamente implementados.

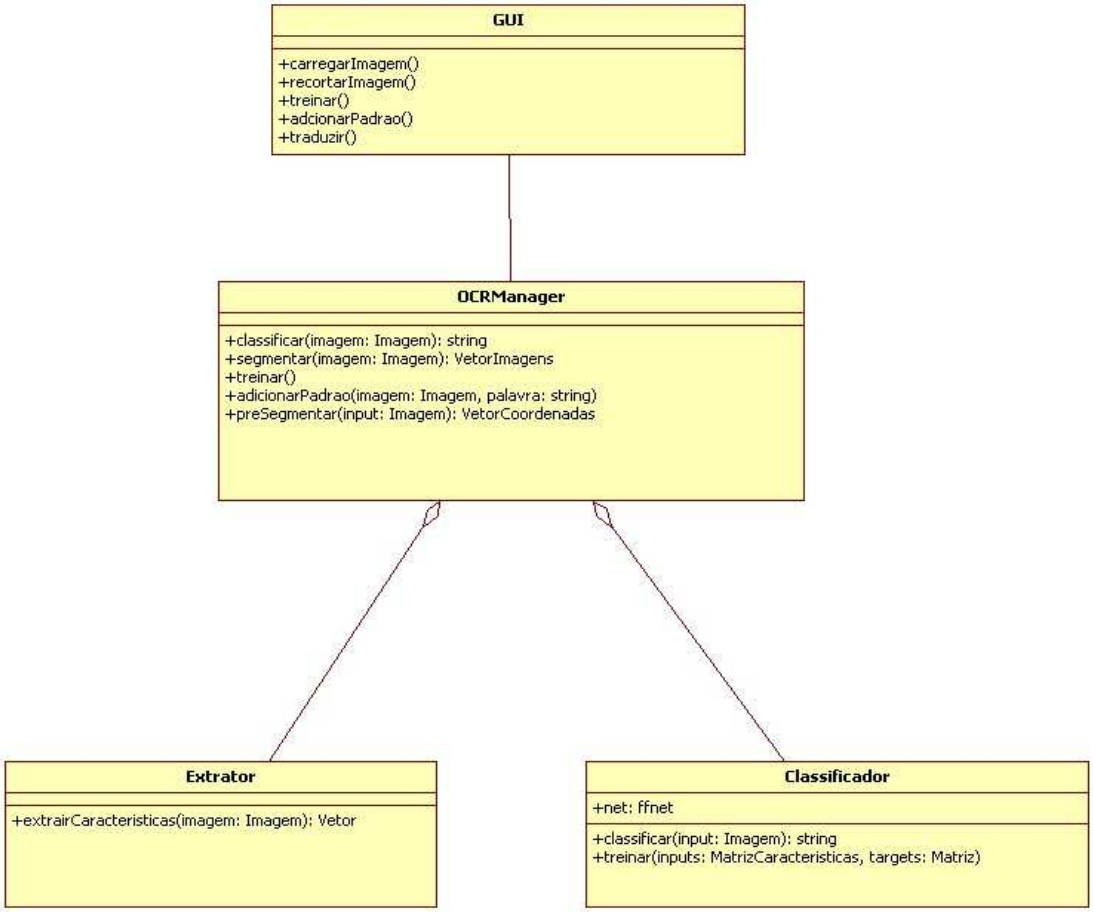

Figura 4-9 – Diagrama de classe do protótipo

4.2.1 Pontos Iniciais

A Figura 4-9 apresenta o diagrama de classe com os métodos e seus parâmetros de entrada e saída. Além disso, esta seção apresenta os inter-relacionamentos entre os módulos componentes da arquitetura proposta. Onde os componentes evidenciam a comunicação entre os módulos do sistema que será criado a partir desta arquitetura. A identificação destes inter-relacionamentos facilita o entendimento do comportamento do sistema gerado.

Dessa forma serão apresentadas a seguir as funcionalidades, e características dos métodos de cada classe aqui apresentado.

4.2.2 Classe GUI

Esta classe representa o módulo de interface com o usuário que é responsável por atender as operações acessadas pelo módulo principal, o qual é representado pela classe OcrManager. Além disso, para implementação dessa classe utilizou-se a biblioteca Python wxPython expendendo a classe wx.Frame. Esta biblioteca permite construção de interfaces gráficas win32, mais detalhes sobre a biblioteca podem ser encontrados no item 4.3 deste capítulo.

A interface gerada por esta classe permite recortes de uma imagem simulando a o módulo de segmentação. Além disso, possui outras funcionalidades mais que serão mostradas na Figura 4-15

4.2.3 Classe OcrManager

Está classe representa o módulo principal da arquitetura. É composta por instancias das classes extrator e classificador, através de um mecanismo de agregação, como mostra a Figura 4-10

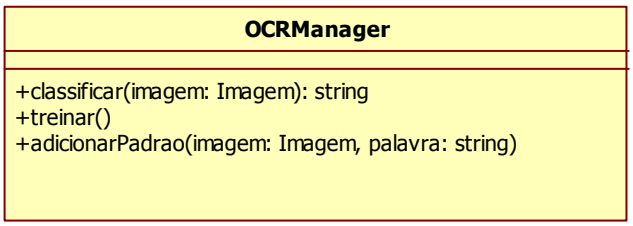

Figura 4-10 – Classe do módulo principal

Na Figura 4-10 temos a representação da interface da classe OcrManager, detalhando todos os métodos utilizados. Conforme descrito a seguir.

- ***classificar:*** Este método recebe como parâmetro uma imagem e retorna uma string que representa o texto contido na imagem. Na implementação deste método existe uma interação com a classe extrator

e classificador.

- ***adicionarPadrao:*** Este método recebe como entrada dois parâmetros, a imagem segmentada e a palavra referente. Na sua implementação interage com a instância da classe extratora e gera um banco de padrões que será utilizado no treinamento da rede neural.
- ***treinar:*** Este método tem como objetivo treinar a rede neural. Na sua implementação é repassado o banco de padrões para instância da classe classificadora, onde ocorre o treinamento da rede.

4.2.4 Classe Extrator

Esta classe representa o módulo de extração de característica, possui um único método chamado *extrairCaracteística*, que recebe como parâmetro uma imagem e retorna um vetor de característica.

Na implementação deste método utilizou-se um biblioteca Python chamada Pylab, para interagir com o software MatLab. Nessa interação é executado um script[xx] Matlab responsável por extrair características da imagem, mais informações sobre o Pylab e o Matlab encontram-se no item 4.3 deste capítulo.

É importante ressaltar que as características extraídas através do script MatLab, não são suficientes para um reconhecimento eficaz de escritas do tipo paleográficas.Tendo em vista que o MatLab foi utilizado como uma prova de conceito para contribuir na implementação do protótipo.

A Figura 4-11 representa a classe extrator.

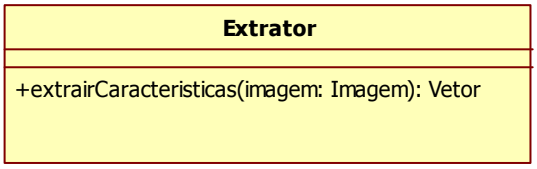

Figura 4-11 - Classe do módulo extrator

4.2.5 Classe Classificador

Está classe representa o módulo classificador e possui como atributo uma instancia de uma rede neural implementada pela biblioteca Python chamada ffnet, mais detalhes sobre o ffnet pode ser encontrado no item 4.1

A rede neural utilizada no protótipo é constituída de uma arquitetura de quatro camadas, onde possuem 35 nodos na camada de entrada e 20 nodos na camada de saída. Além disso, possui duas camadas ocultas com 10 nodos cada.

Na Figura 4-12 temos a representação da interface da classe classificador, detalhando todos os métodos utilizados. Conforme descrito a seguir.

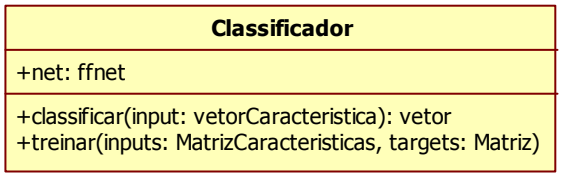

Figura 4-12 - Classe do módulo Classificador

- ***classificar:*** Este método recebe como parâmetro um vetor de características e retorna um vetor numérico, onde cada número corresponde ao código ASCII da cada letra do alfabeto. Os parâmetros são passados para a instância da rede neural que irá gerar a saída desejada.
- ***treinar:*** Este método recebe como parâmetro uma matriz de características e uma matriz numérica (targets), onde cada número corresponde um código ASCII. Cada linha da matriz de característica possui uma linha correspondente da matriz numérica. Estes parâmetros também são passados para a instância da rede neural com a finalidade de treiná-la.

4.3 FERRAMENTAS UTILIZADAS.

A implementação do módulo principal descrito na arquitetura em questão foi realizado com ferramentas de desenvolvimento de código aberto ou de uso livre e distribuído gratuitamente na Internet, sendo todas multi-plataforma. O uso de tais ferramentas permitiu a modificação do código necessário para a implementação dos conceitos de redes neurais, sistema de reconhecimento de palavras (OCR) e criação de interfaces. Os tópicos seguintes tratam em específico de cada uma das ferramentas utilizadas para a implementação do ambiente.

4.3.1 Python

O uso do Python [80] como linguagem de programação deve-se a vários motivos, entretanto podemos citar alguns pontos fortes que influenciaram na implementação deste trabalho.

O primeiro ponto forte é que é uma linguagem altamente portável. Isto quer dizer que o desenvolvimento de eventuais classes ou soluções podem ser migrado para outras plataformas sem grande esforço. Outro ponto em questão é uma linguagem de programação dinâmica que pode ser utilizada em diversos tipos de aplicações (Web Aplication, Win32 App, Redes, Aplicações científicas, móbile entre outras). Além disso, é uma linguagem de fácil integração com outras linguagens e ferramentas (C++, java, .NET, C, MatLab e delphi) possuindo uma extensa biblioteca.

A linguagem Python segundo [81] foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros. Sendo assim, a versão utilizada nesta implementação foi a Python 2.4 para isso foram utilizados alguns módulos ou plugins que auxiliaram para a implementação do protótipo, sendo associados a outras arquiteturas. Como pode ser descrito a seguir.

4.3.2 Eclipse e PyDev

O Eclipse é uma plataforma livre usada para o desenvolvimento de projetos em diferentes linguagens de programação para distintos sistemas operacionais. Atualmente, ele suporta C/C++, COBOL e Java. É bastante usada em aplicações desktop fazendo o uso de um IDE extensível através o uso de vários plugins.

O PyDev trata-se de um plugin que permite aos usuários utilizar o software de desenvolvimento eclipse para desenvolver aplicações em Python e Jpython. Esse plugin permite o desenvolvimento de aplicativos Python para desktops, possuindo vários recursos como auto-complete, detecção de erros, entre outros. Entretanto, embora esses recursos não sejam possíveis com a API do PyS60, mas podem ajudar bastante no desenvolvimento de aplicações, especialmente por causa do HighLighting automático e na detecção de erros comuns da sintaxe Python.

O motivo principal da utilização do PyDev para essa aplicação, foi devido a sua grande interação com o software eclipse, facilitando para construção de aplicações em python. Mais informações sobre o PyDev podem ser encontradas nas referências [81] e [82] .

4.3.3 WxPython

O wxPython [84] é uma API de biblioteca wxWidgets. Por sua vez é uma ferramenta utilizada na construção da interface com o usuário que permite que as aplicações sejam mais facilmente portáveis e que tenham a aparência de uma aplicação nativa do Windows. Além disso, permite escrever aplicações-GUI multi-plataformas no estilo de pyQT ou pyGTK[85]

O wxWidgets [85] é componente de software que viabiliza a interação com o usuário, sendo um utilitário para criação de widgets multi-plataforma, possuindo uma biblioteca com elementos básicos para a construção de interfaces gráficas para o usuário, possibilitando até mesmo conexões com bancos de dados ODBC e conectividade via sockets. Permite que desenvolvedores criem aplicações para Win32, Sistemas operacionais MAC, entre outros. Pode ser usado a partir de linguagens como C + +, Python, Perl, C # e .

NET.

Sendo assim, o wxPython será utilizado para essa aplicação, com o objetivo da criação de uma interface com o usuário (GUI).

4.3.4 Ffnet

O ffnet (Fast Artificial Neural Network Library python) é uma biblioteca de redes neurais baseada na característica de treinamento que utiliza soluções em python. De maneira que o treinamento pode ser apresentado com o uso de vários esquemas de otimização.

Entretanto o ffnet foi utilizado na implementação do protótipo justamente para simulação da rede neural utilizada.

4.3.5 Matlab

O MatLab é um software interativo de alta performance voltado para cálculos numéricos. Trata-se de uma linguagem de alto nível em um ambiente que permite executar diversas tarefas computacionais, de forma rápida e eficaz. Além disso, é bastante utilizado para cálculos com matrizes, processamento de sinais e construções de gráficos.

Esse sistema permite uma combinação perfeita entre as linguagens de programação mais utilizadas, tais como, Java e Basic, C, Python entre outras. Esta é então uma das grandes vantagens do Matlab.

Dessa forma, o MatLab contribuiu para implementação do protótipo, na utilização de uma API em seu de código, que são responsáveis para extração de característica de uma imagem. Uma vez que o software é bastante eficiente na resolução desse tipo de problema.

4.3.6 Pylab

O Pylab é um módulo da linguagem Python que permite gerar gráficos de duas dimensões de excelente qualidade, permitindo edição interativa, animações, inúmeros tipos de gráficos diferentes, anotações em sintaxe Latex e salvamento das imagens geradas em diversos formatos diferentes. A sintaxe de criação e manipulação das imagens será familiar

para quem já trabalhou com o software comercial Matlab, mas provendo muito mais capacidades, além de uma interface baseada em objetos, para quem conhece a técnica.

O Pylab permite trabalhar com diversos tipos de gráficos diferentes, entre eles: gráficos de funções, múltiplos gráficos, histogramas, funções discretas, torta, barra, etc. Fornece funções para a customização dos gráficos, podendo trabalhar com diversas fontes diferentes, cores, tamanhos de página, e muito mais. Além disso, existem funções para a manipulação e análise de imagens e sinais.

No caso do protótipo implementado, o Pylab tem a funcionalidade de interagir com o MatLab para a criação de algoritmos de extração de caracterizas de uma imagem.

4.4 RESULTADOS OBTIDOS

Esse tópico tem como finalidade apresentar os resultados obtidos na implementação do protótipo apresentado, mostrado quais as funcionalidades foram implementadas e a forma de como foram implementadas, de maneira a contribuir para representação da arquitetura proposta, como mostra a seguir.

4.4.1 Reconhecimento da imagem

Conforme já descrito na arquitetura apresentada sobre o processo de reconhecimento de imagens. O protótipo implementado simula as funcionalidades descritas na Figura 4-13. Onde o primeiro passo implementado identificado pelo número 1, é a escolha de uma imagem pelo o usuário, o número 2 possibilita o usuário recortar a imagem, este executa o botão de treinar identificado pelo número 3, onde por sua envia para o módulo principal para que seja analisada a requisição e enviada para o módulo responsável, representado pelo número 4. Após verificado o destino da imagem o módulo principal envia para o módulo de extração de características, representado pelo número 5, esse por sua vez extrai as características da imagem reapresentada e retorna o resultado para o módulo principal representado pelo número 6, que envia para o módulo classificador representado pelo número 7 que tem a função de treinar a rede neural conforme pode ser visto no próximo item.

Dessa forma, o mecanismo de reconhecimento de imagem possui como resultados as seguintes funcionalidades.

- Permite o usuário carregar uma imagem,
- Fazer um recorte selecionando a palavra, simulando o módulo de segmentação;
- Após o recebimento da imagem já segmentada, ocorre uma interferência humana informando qual o valor (descrição da palavra) da imagem selecionada;
- Após o recorte da imagem, o usuário ocorre o reconhecimento, e o tratamento das palavras selecionadas.

A Figura 4-13 representa o diagrama de seqüência para o reconhecimento das imagens.

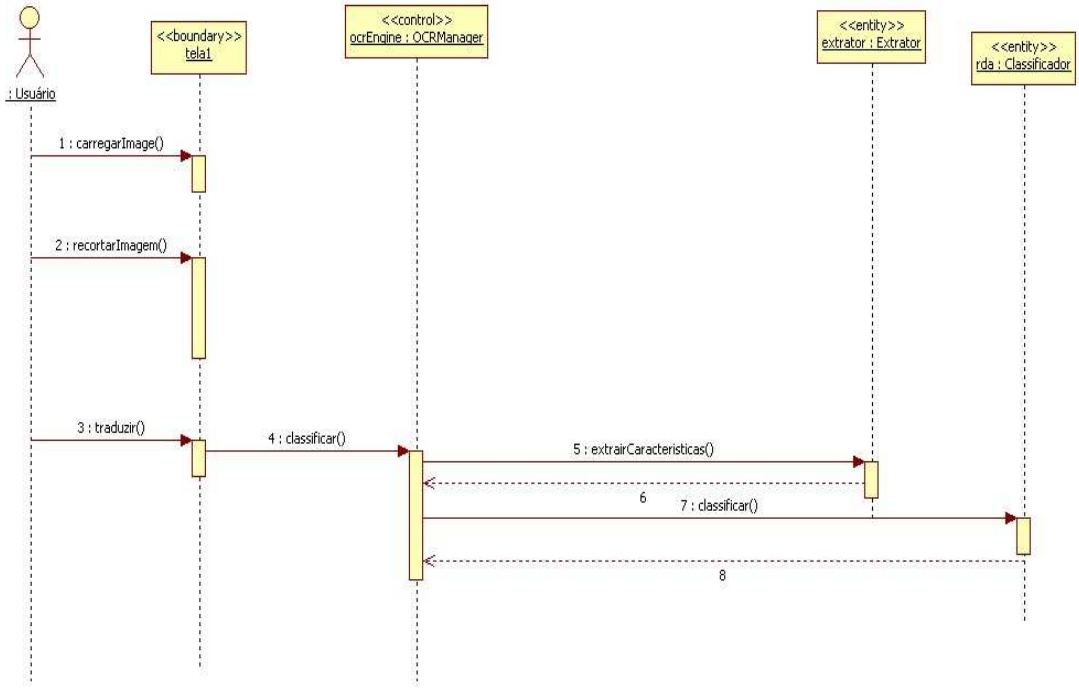

Figura 4-13 – Diagrama de seqüência do reconhecimento de imagem

4.4.2 Treinamento da rede neural

O tratamento da rede neural é realizado pelo módulo classificador, após a extração de todas as características existente na imagem, uma vez que as características da imagem são os parâmetros e entrada da rede neural. Sendo assim a Figura 4-14 apresentada o processo de treinamento da rede neural, segundo o medo de arquitetura proposto neste trabalho. Onde o primeiro passo implementado identificado pelo número 1, é a escolha de uma imagem pelo o usuário que já tenha sido devidamente treinada. O número 2 possibilita o usuário recortar a imagem e adicionar padrões na mesma, identificado pelo número 3, onde por sua envia para o módulo principal para que seja analisada a requisição e enviada para o módulo responsável por adicionar padrões na imagem, representado pelo número 4. Depois de verificado o destino da imagem o módulo principal envia para o módulo de extração de características, representado pelo número 5, esse por sua vez verifica as características da imagem apresentada e retorna o resultado para o módulo principal representado pelo número 6, que envia para tela de apresentação (GUI) representada pelo número 7. A partir daí a imagem é armazenada ao Banco de padrões para que seja devidamente treinada e classificada representado pelos 9,10 e 11. Após o treinamento da rede neural e mostrado o resultado para o módulo principal, representado pelo número 12.

Dessa forma, o mecanismo de treinamento da rede neural possui como resultados as seguintes funcionalidades.

- Permite o usuário carregar uma imagem já segmentada e treinada
- Após a imagem carregada, o usuário seleciona a palavra desejada e solicita o reconhecimento da mesma.
- Após o reconhecimento o valor interpretado pela rede neural é apresentado.

A Figura 4-14 representa o diagrama de seqüência para o treinamento da rede neural

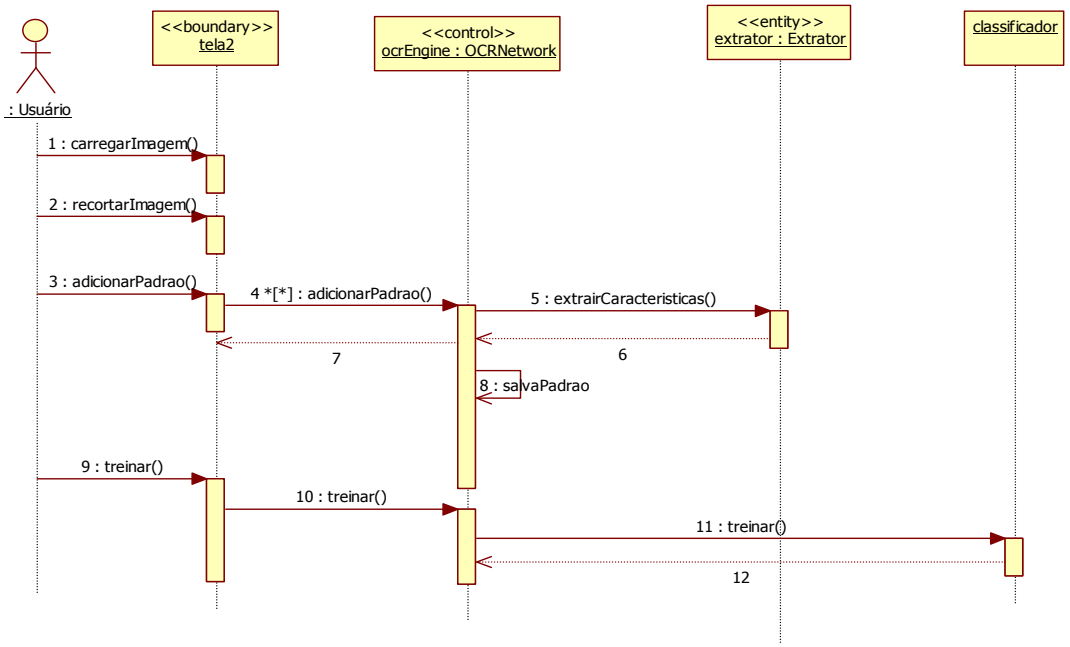

Figura 4-14 - Diagrama de seqüência do treinamento da rede neural

Entretanto, os resultados obtidos neste protótipo seguiram a metodologia da arquitetura proposta, sendo implementados os módulos descritos na Tabela 4-1, que realizaram as funções de reconhecimento da imagem e treinamento junto à rede neural conforme falado nos itens 4.4.1 e 4.4.2. A seguir mostra um resumo das implementações realizadas e demonstração das telas do protótipo:

- A primeira parte: trata da segmentação da palavra, onde o protótipo realiza de forma manual. É interessante que o recorte na imagem seja de forma precisa minimizando os erros de reconhecimento.
- A segunda é a extração de características. Pois essa funcionalidade é implementada com o auxilio do software MatLab. De maneira que ocorre uma avaliação sobre as técnicas de reconhecimento, para que haja um melhor reconhecimento das características da palavra. Nesse caso os resultados são extraídos para um vetor de característica.
- A terceira demonstra o treinamento da rede neural que tem como objetivo classificar a imagem após seu tratamento.

4.4.3 Funcionalidades do protótipo

A Figura 4-15 representa uma primeira tela (passo1) do protótipo, onde ocorre o reconhecimento da imagem, mostrando os resultados após seu recorte e o processo de extração de característica, para que assim seja treinada pela rede neural.

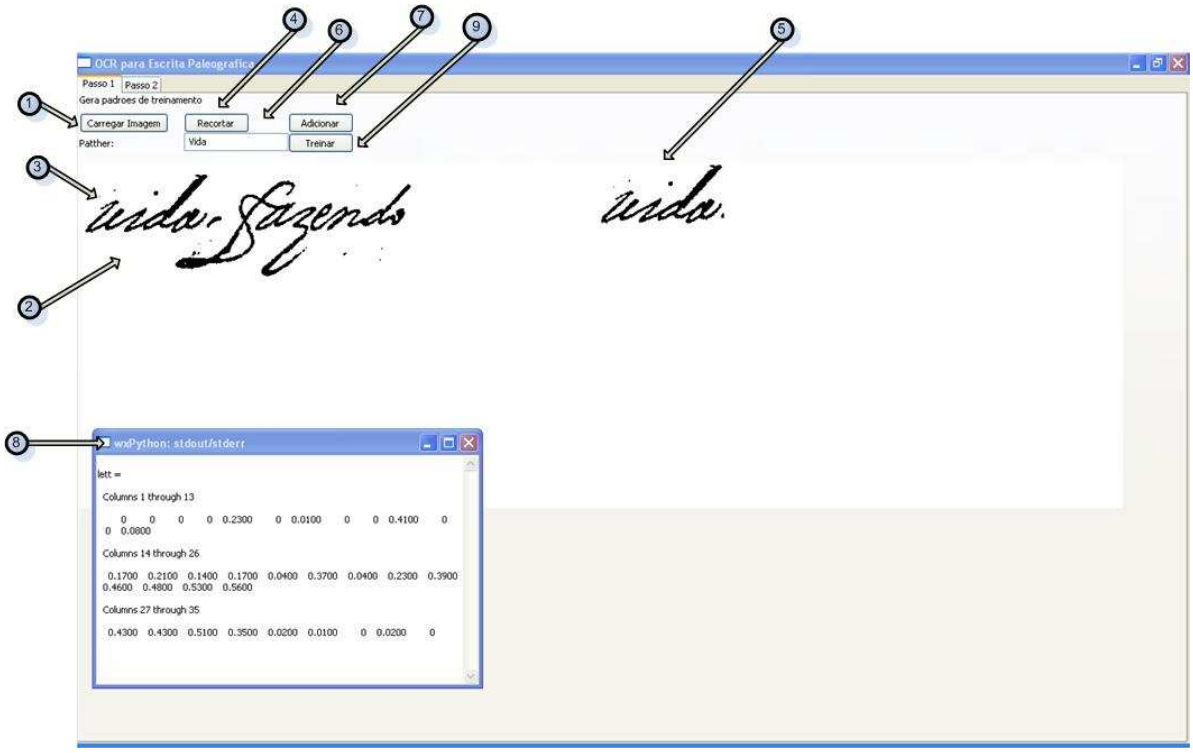

Figura 4-15 - Tela de apresentação do protótipo implementado - passo1.

O detalhamento do protótipo é mostrado itens descritos a seguir:

1. Botão de carregar imagem: nesse item, o usuário tem a possibilidade de selecionar uma imagem armazenada em um disco local e visualizar na tela. Ao se clicar nesse botão, abrirá o caminho onde o usuário pretende buscar a imagem.

2. Imagem: esse item é a visualização da imagem selecionada.

3. Recorte da imagem: esse item trata-se do processo de recorte da imagem selecionada, simulando o processo de segmentação. O usuário pode arrastar com o mouse realizando o recorte necessário.

4. Botão recortar: após realizado o recorte na imagem (item 3), ao clicar no botão de recorte o mesmo irá mostrar o resultado da imagem recortada (item 5).

5. Resultado do recorte: nesse item é apresentado o resultado do recorte

6. Campo patther: Esse campo permite que o usuário indique qual o valor referente a imagem recortada.

7. Botão adicionar: após o usuário informar qual o valor da imagem recortada (item 6), o mesmo clica no botão adicionar que irá realizar a função de todo o processo existente no módulo de extração de características, mostrado na tela representada pelo item 8

8. Tela de resultado da matriz de característica: Essa tela mostra o resultado da matriz do módulo de extração de características, utilizando a biblioteca WxPython.

9. Botão treinar: esse botão tem como objetivo realizar o treinamento da rede neural através do resultado do item 7 realizando a função do módulo classificador.

A Figura 4-16 representa a segunda tela (passo2) do protótipo, onde ocorre o processo de reconhecimento da imagem já treinada.

O detalhamento do protótipo é mostrado itens descritos a seguir:

10. Botão de carregar imagem: nesse item, o usuário tem a possibilidade de selecionar uma imagem armazenada em um disco local e visualizar na tela. Ao se clicar nesse botão abrirá o caminho onde o usuário pretende buscar a imagem.

11. Imagem: esse item é a visualização da imagem selecionada.

12. Recorte da imagem: esse item trata-se do processo de recorte da imagem selecionada e já treinada.

Figura 4-16 - Tela de apresentação do protótipo implementado - passo 2.

13. Botão reconhecer: após realizado o recorte na imagem (item 11), ao clicar no botão de reconhecer o mesmo irá mostrar um pré resultado da imagem recortada, que envia para o resultado final.

14. Resultado final: após o pré resultado o item 13 mostra o resultado final da imagem recortada após seu treinamento junto a rede neural.

Sendo assim, os resultados obtidos referente à arquitetura proposta estão sendo representados pelo protótipo, nos módulos que foram implementados. A Figura 4-15 e Figura 4-16 representam a parte gráfica do protótipo e seu código fonte está demonstrado nos anexos desse trabalho.

5 - CONCLUSÕES

As conclusões aqui descritas mostram o resultado final a que se propôs o trabalho. O resultado final deste foi um conjunto de mecanismos a serem implementadas em um sistema de OCR paleográfico. Os mecanismos que foram demonstrados através de uma proposta de arquitetura foram divido em módulos, onde cada módulo representa uma funcionalidade a ser implementada.

A apresentação da arquitetura proposta foi através de uma metodologia de engenharia de software baseada em diagramas modulares, de forma que pudesse facilitar o reuso e o incremento de funcionalidades adjacentes existentes em cada módulo. Além disso, foi realizada uma análise de técnicas para reconhecimento, extração de características e segmentação de escritas, na qual essas técnicas foram aplicadas nos módulos apresentados.

As tecnologias de redes neurais artificiais, de reconhecimento de escritas e de segmentação de palavras foram essenciais para conclusão da arquitetura proposta, uma vez que essas tecnologias são bastante usadas em escritas de difícil interpretação e em manuscritos cursivas. Sendo assim, fez-se uso dessas tecnologias para o desenvolvimento de um protótipo que permitiu a validação da arquitetura proposta.

O protótipo desenvolvido neste trabalho apresentou os módulos de extração de características, realizando um processo de reconhecimento das imagens e o processo de treinamento das palavras junto à rede neural. Para auxiliar no processo de reconhecimento das imagens foram utilizados recursos do software MatLab, uma vez que o mesmo já possui funcionalidades referente ao tratamento e segmentação das palavras.

Ainda que sejam necessários desenvolvimentos mais profundos no que se refere aos processos de segmentação dos documentos e reconhecimento das escritas paleográficas do século XVI ao XIX. Podemos concluir que arquitetura proposta é funcional, uma vez que o protótipo implementado apresenta funcionalidades que seguem a arquitetura e tem como perspectiva futura a implementação dos módulos que não foram devidamente implementados.

Nesse caso, o Projeto Resgate necessita de uma ferramenta que resolva por completo todos os problemas encontrados no decorrer deste trabalho, no que se refere ao reconhecimento de escritas paleográficas, criando também mecanismos que comunique diretamente de forma online com a base de dados já existente no Projeto.

5.1 TRABALHOS FUTUROS

Em uma primeira análise, o protótipo desenvolvido não realiza uma implementação direta de todos os módulos propostos na arquitetura.

Problemas de segmentação, dificuldades no reconhecimento e tratamento das imagens, demonstram, de fato, que o reconhecimento de escritas paleográficas é um processo que requer melhores técnicas. Dessa forma são sugeridos como proposta de trabalhos futuros, os itens descritos a seguir.

- Propor uma metodologia de segmentação para escritas paleográficas, de maneira que possa ser implementado um mecanismo de segmentador automático e não mais de forma manual.
- Aprimorar as técnicas para extração de características de escritas paleográficas, aplicadas após uma etapa de pré-processamento, com o objetivo de abstrair as informações relevantes contidas em cada imagem, utilizando metodologias já exploradas. Criando assim uma independência junto ao MatLab.
- Identificar de maneira mais eficaz, uma a melhor arquitetura de rede neural para o reconhecimento e treinamento de palavras, uma vez que a arquitetura utilizada torna-se limitada em alguns pontos. De maneira possa ser aplicada para o reconhecimento de documentos paleográficos.
- Uma vez resolvido todos os problemas de reconhecimento de escritas paleográficas existentes no Projeto Resgate. Propor um mecanismo de reconhecimento de escritas paleográficas de outros idiomas, baseando-se na arquitetura proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Site do CMD, http://www.resgate.unb.br acessado em 10/04/2008.

[2] MARQUES, João M. Silva. Estudos de paleografia portuguesa. s.l., s.ed., 1938. RGPL

[3] Srikantan, G.; Lam; S. W.; Srihari, S, N; "Gradiente-Based Contour Encoding For Character Recognition, Pattern Recongnition", Vol.29, No. 7, pp. 1147-1160,1996.

[4] Hecht-Nielsen, R. 1982. Neural analog processing. Proc. SPIE, 360. PP. 180-189. Bellingham, WA

[5] RODRIGUES, Otávio Calasans. Noções gerais de paleografia e diplomática. Rio de Janeiro: 1951. Apostila de curso, Arquivo Nacional.

[6] ACCIOLI, Vera Lucia Costa. A escrita no Brasil Colônia. Recife: Massangana, 1994.

[7] POVOAS, Joaquim de Mello E; BENCHIMOL, Samuel. Cartas do primeiro governador da capitania de sao jose do rio negro. Manaus: Univ Amazonas, 1983. 398 p

[8] Alspector, J.,A. Jayakumar, and S. Luna, 1192. " Experimental evalution of learning in a neural microsystem," Advances in Neural Information Processing Systems, vol. 4, pp. 871-878, San Mateo, CA: Morgan Kaufmann.

[9] T. Yamasaki, y T.Hattori, "A new data tablet system for Handwrinting characters and drawing base don the image processing", IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, pp. 428-431, Octubre 14-17, 1996

[10] Ing. Nancy Rangel Galán, Ing. Israel Delgado Guerra,Ing. Bravo Zúñiga Haydee Posgrado de Ingeniería, Facultad de Ingeniería, UNAM

[11] BERWANGUER, Ana Regina; LEAL, João Eurípedes Franklin. Noções de

paleografia e diplomática. Santa Maria, RS: Universidade Federal, 1991.

[12] COSTA, Vera Lucia Santina. Nocoes de paleografia. Recife: Univ Cat Pernambuco, 1970. 75 p.

[13] http://www1.ci.uc.pt/ipd/index.htm acessado em 04/2008

[14] http://www.unl.pt/guia/2007/fcsh/historia-area-de-especializacao-em-historia-moderna-e-dos-descobrimentos. acessado em 03/2008

[15] NUNES, Eduardo. Vária paleográfica. Lisboa: Barbosa e Xavier, 1973. OBSERVAÇÕES sobre o estudo da paleografia em Portugal. Porto: Centro de Estudos Humanísticos / Faculdade de Letras, 1967. RGPL

[16] OLIVEIRA, José Teixeira de. A fascinante história do livro. Rio de janeiro: Kosmos, 1987.

[17] MENDES, Ubirajara Dolacio. Nocoes de paleografia. Sao paulo: Estado De Sao Paulo(O), 1953. 123 p

[18] http://www.aab.org.br/relbibpal.htm acessado em 03/2008.

[19] VALENTE, José Augusto Vaz. Álbum de paleografia portuguesa. São Paulo: USP/ECA, 1983. De re paleografhica. Marília, SP: FFCL, 1970

[20] ZAMBEL, Miriam Mani. Breve história da escrita. São Carlos, SP: Universidade, 1984.

[21] PRATESI, Alessandro – *Genesi e forme Del documento medievale*, 2ª Ed., Roma: Jouvence, 1987

[22] TESSIER, Georges – *La Diplomatique*, 3ª Ed., Paris: PUF-Que Sais-Je?, 1966.

[23] MARQUES, José A influência da bulas papais na documentação medieval portuguesa, Sep. Da Revista da Faculdade de Letras, 2ª série, Vol. XIII, Porto,

1996.

[24] SANTOS, Graça A . Inscrições tumulares por siglas. s.l., s.ed., 1942. RGPL

[25] D. Manuel C. Díaz y Diaz. “Notas para la historia de la escritura visigótica en su periodo primitivo”, Bivium-Homenaje a. Madrid, p.175-96;

[26] Jakson,1981:80-1 “This style allowed far more words to be crammed onto the page, particularly if the text was arranged in two-column layout: and it has been calculated that a book in this style needed as little as one third of the page area required by open Carolingian letters of the same height.”

[27] DOBIACHE-ROJDESTVENSKY, Olga (1925). “Quelques considérations sur les origines de l'écriture dite ‘gothique’”, Mélanges d'histoire du moyen âge offerts à M. Ferdinand Lot par ses amis et ses élèves. Paris, citada por Boussard:1951.

[28] Higounet Charles, “história concisa da escrita” - 4ª Ed., Portugal: 1986..

[29] FLORIANO CUMBREÑO, Antonio – Curso General de Paleografia y Paleografia y Diplomática Españolas, Oviedo: 1946.

[30] COELHO, Maria Helena da Cruz, Maria José Azevedo Santos, Saul António Gomes, Maria do Rosário Morujão, Estudos de Diplomática Portuguesa, Lisboa, Edições Colibri/Faculdade de Letras da Universidade de Coimbra, 2001.

[31] ARAÚJO, Maria Carmem de C. Silva. Perspectiva Histórica da Alfabetização. Viçosa, MG: UFV,1995.

[32] STORIG, Hans Joaquim. A aventura das línguas. São Paulo: Melhoramentos, 1990. O TESTAMENTO de D. Lourenço Vicente e as suas capelas na Sé de Braga e na Lourinhã. Braga: Faculdade de Letras, 1993.

[33] DIAS, J.; MARQUES, Oliveira; RODRIGUES, T. Álbum de paleografia. Lisboa: Estampa, 1987. DIRINGER, David. A escrita. S.L. S.ED., 1968. RGPL

[34] ROMÁN BLANCO, Ricardo. Estudos paleográficos. São Paulo: Laserprint, 1987.

Lâminas de paleografia selecionadas, transcritas e comentadas. São Paulo: Faculdade de Filosofia, Ciências e Letras, 1956. BN

[35] McMURTRIE, Douglas C. Normas para transcrição paleográfica da documentação brasileira. Rio de Janeiro: UNI-RIO, 1990.

[36] SÃO PAULO (estado) departamento do Arquivo. Curso livre de paleografia... São Paulo: Dep. do Arquivo do Estado [1952] Mimeo. NA

[37] COSTA, Avelino de Jesus, padre. Álbum de paleografia e diplomática portuguesa. 3. ed. Coimbra: DGAC, 1975. RGPL.

[38] ARNS, Paulo Evaristo, bispo. A técnica do livro segundo S. Jerônimo. São Paulo: Imagino, 1993. ARQUIVO HISTÓRICO DO RIO GRANDE DO SUL. A publicação de documentos históricos no Arquivo Histórico do Rio Grande do Sul. Porto Alegre: SEC, Subsecretaria de Cultura, 1986. AN.

[39] http://www.uefs.br/filologia/index.swf acessado em 03/2008

[40] RODRIGUES, Otávio Calasans. Noções gerais de paleografia e diplomática. Rio de Janeiro: 1951. Apostila de curso, Arquivo Nacional.

[41] Maria Helena Lopes de CASTRO - "Normas de transcrição para textos medievais portugueses", in Boletim de Filologia, Lisboa, Centro de Estudos Filológicos, 22, 1973.

[42] LUGER, George F. Inteligência Artificial: Estruturas e Estratégicas para a Resolução de Problemas Complexos / George F. Luger - 4ª Ed. – Porto Alegre: Bookmann, 2004.

[43] HARMON, P. and King, D. 1985 Expert Systems: Artificial Intelligence in Business. New York: Wiley. Harmon, P.,Maus, R., and Morrissey,W. 1988. Expert Systems: Tools and Applications. New York: Wiley

[44] Korf, R. E. 1998. Artificial intelligence search algorithms. In CRC Handbook of

Algorithuns and Theory of Computation, M. J Atallah, ed. Boca Raton, FL: CRC Press

[45] IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21, NO. 3, MARCH 1999

[46] P. Lingras. Comparasion of Neofuzzi and rough neural Networks. Information Sciences, pages 207-215,1998.

[47] Proceedings of the Seventh International Conference on Document Anlysis and Recognition (ISDAR'03) 0-7695-1960-1/03

[48] F.H. Cheng, "Multi-Stroke Relaxation Matching Method for Handwritten Chinese Character Recognition," *Pattern Recongnition*, vol.31, nº 4, pp. 401-140,1998

[49] H. Kato, S. Yokozuka, and H.Kida, "Recognition of Handwritten Character Using Peripheral Bend Contributivity," IEICE Technical, Report, PRU95-3,1995

[50] Hecht-Nielsen, R. 1982. Neural analog processing. Proc. SPIE, 360. PP. 180-189. Bellingham, WA

[51] Hinton, G. E. and Sejnowski, T. J. 1987. Neural network architectures for AI. Tutorial. AAAI Conference.

[52] Hopfield, J.J. 1984. Neural Network and Physical sustems with emergent collective computational abilities. Proceedings of the National Academy of Sciences 79.

[53] R. J. Rodrigues, E. Silva and A. C. G. Thomé Feature extractions Using Contour Projection, accepted for presentation at SCI2001. Orlando – USA July 2001.

[54] G. V. Kuppac; R. J. Rodrigues, and A. C. G. Thomé. "Extração de características para reconhecimento de dígitos cursivos", 7th Brazilian simposium on Neural Networks, Rio de Janeiro-RJ, November 2000.

[55] Mathworks, (2002), In: http://www.mathworks.com, acessado em 04/2008.

[56] Silva, Eugênio, Rodrigues, Roberto José e Thomé, Antonio C. G.; Extração de características para o Reconhecimento de Letras Manuscritas, V Simpósio Brasileiro de Automação Inteligente, Canela, novembro 7-9,2001.

[57] Pavilidis, T.; Algorithms for Graphics and Image Processing, Computer Science Press, Inc.; 1982.

[58] Robert J. Schalkoff, Pattem Recognition: Statistical, Structural and Neural Approaches, Jonh Wiley & Sons, 1992

[59] Anil K. Jain, Jianchang Mao, "Artificial Neural Networks: A Tutorial", computer , vol 29, nº 3, pp. 31-44, March 1996.

[60] [Hopfield,1982] Hopfield, J. J., "Neural networks and physical Systems with emergent collective computacional habilitéis," Proc. Of the National Academy of Sciences, USA, vol 79, pp. 2554-2558,1982, Reprited in [Anderson and Rosefeld, 1988]

[61] T. Caesar, J.M. Gloer, E. Mandler, "Prepocessing and feature extraction for a handwriting recongnition system", IEEE Transactions on pattem anlysis and machine intelligence, pp. 408-411, 1993.

[62] Andrew W. Senior, Anthony J. Robinson, "An off-line Cursive Handwriting Recognition System", IEEE Transactions on pattem análisis and machine intelligence, vol. 20, nº 3, pp.309-321, Marzo 1998.

[63] Trieri, O.D.; Jain, A.K.; Taxt, T.; "Feature extraction Methods for Character Recongnition – A Survey"; Pattem Recognition, Vol. 29 nº 4, pp. 641-662,1996

[64] Yang, L.; Prasad, R.; "Online Recognition of Handwritten Characters Using Differential Angles and Structural Descriptors", Pattem Recognition Letters, nº 14, Dec-93, North-Holland, pp: 1019-1024.

[65] Pavilidis, T.; "Algorithms for Graphics and Image Processing", Computer Science

Press, Inc, 1982.

[66] Rodrigues, R. J.; Kupac, G. V.; Thomé, A. C. G.; “Character Feature Extraction Using Polygonal Projection Sweep (Contour Detection)”, accepted for presentation at IWANN2001. Granade –Spain, June 13 a 15.

[67] E. Anquetil, G. Lorette, “Off-line cursive handwriting segmentation”, IEEE Transactions on pattern análisis and machine intelligence, pp. 112-117,1997

[68] http://www.heatonresearch.com/articles/7

[69] Suen, C.Y.; Berthold, M.; Mori, S.; “Automatic Recognition of Handprinted Characters – The State of The Art”, Proceedings of the IEEE, Vol. 68, No. 4,April 1980.

[70] Srikantan, G.; Lam; S. W.; Srihari, S, N; “Gradiente-Based Contour Encoding For Character Recognition, Pattern Recongnition”, Vol.29, No. 7, pp. 1147-1160,1996.

[71] Kupac, G.V.; Rodrigues, R. J.; Thomé, A. C. G,; “Extração de características para reconhecimento de dígitos cursivos”, Vlth Brazilian Symposium on Neural Networks – SBRN 2000, Rio de Janeiro, november 2000.

[72] http://www.heatonresearch.com/articles/42/page1.html acessado em 03/2008.

[73] Rodrigues, R. J.; Thomé, A. C. G.; “Cursive character recognition – a character segmentation method using projection profile-based technique”, The 4th World Multiconferece on Systemics, Cybernetics and Informatics SCI 2000 and The 6th International Conference on Informations Systems, Analysis and Synthesis ISAS 2000 – Orlando, USA – August 2000.

[74] Duda, R. O; Hart, P. E. "Pattern Classification and Scene Analysis", John Wiley & Sons, 1973, pp.

[75] Andries P. Engelbrecht 2002: Computational Intelligence. An Introduction.

[76] Discriminative Training for HMM-Based Offline Handwritten Character

Recognition IEEE

[77] Guingo, Bruno Clemente. Rodrigues, Roberto José. Técnicas de Segmentação de Imagens, Extração de Características e Reconhecimento de Caracteres Universidade Federal do Rio de Janeiro Área de Ensino e Pesquisa, NCE/IM,

[78] http://www.stimuli.com.br/trane/links/2007/may/18/pytesser---google-code/ acessado em 04/2008.

[79] http://www.abbyy.com/ acessado em 04/2008

[80] http://www.adrtnhs.co.uk/ acessado em 04/2008

[81] http://www.labic.nce.ufrj.br/downloads/vsbai_2001.pdf acessado em 04/2008

[82] Nei Kato, Member, IEEE, Masato Suzuki A Handwritten Character Recognition System Using Directional Element Feature and Asymmetric Mahalanobis Distance.

[83] Horst Bunke. Recognition of Cursive Roman Handwriting - Past, Present and Future. Department of Computer Science

[84] http://www.wxpython.org/ e http://sourceforge.net/project/showfiles.php?group acessado em 05/2008.

[85] http://www.wxwidgets.org/ acessado em 05/2008

[86] http://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf e http://www.gui.com.br/posts/list/39105.java acessado em 05/2008

[87] http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/pat/singleton.htm e http://portuguese.onbase.com/Products/OnBaseProductModules/ImagingCaptureModules/OCR/default.aspx

APÊNDICE A – RESULTADOS DO CÓDIGO FONTE DO PROTÓTIPO – PASSO 1

```
# -*- coding: cp1252 -*-
import win32com.client as w32c
from win32com.client import constants
from mlabwrap import mlab
from numpy import *
import Image
from ffnet import ffnet, mlgraph, readdata, savenet, loadnet
import scipy
from scipy.io import  read_array, write_array
from pylab import save, load

class OCRNetwork:
"""
    OCR baseado em Rede Neural Artificial.
"""
    def __init__(self):
        self.extrator = Extrator()
        self.classificador = Classificador()
        self.segmentador = Segmentador()
        self.loadNet()
    def classificar2 (self, image_name):
        imagem = mlab.imread(image_name)
        return self.classificar(imagem)
    def classificar(self, imagem):
    caracteristicas = self.extrator.extrairCaracteristicas(imagem)
        output = self.classificador.classificar(caracteristicas)
        return self.array_para_palavra(output)
    def segmentar(self, imagem):
        """
  Recebe uma imagem como parâmetro de entreda e retorna uma string
contendo a palavra conteda na imagem
        """
        return self.segmentador.segmentar(imagem)
        def treinar(self):
        """
        Este método treina a RDA.
        """
        inputsFile = 'inputs.txt'
        targetsFile = 'targets.txt'
        scipyArrayInputs = load(inputsFile)
        scipyArrayTargets = load(targetsFile)
        return self.classificador.treinar(scipyArrayInputs,
scipyArrayTargets)

    def array_para_palavra(self, output):
        """
        Converte um vetor de número para uma string
        """
        palavra = ''
        for num in output:
            palavra += chr(num)
```

```
        return palavra
    def palavra_para_array(self, palavra):
        """
        Converte uma string em um vetor de códigos ASCII
        """
        l = []
        for letra in palavra:
            l.append(ord(letra))
        if len(l) < 20:
            l += [32,] * (20 - len(l))
        if len(l) > 20:
            l = l[:20]
        #l = array(l)
        #l.shape = (20,1)
        return l

    def adicionarPadrao2(self, imagem_name, palavra):
        """
        Adiciona padrão (Image, Palavra) ao banco de padrões
        """
        im = mlab.imread(imagem_name)
        return self.adicionarPadrao(im, palavra)
       def adicionarPadrao(self, imagem, palavra):
        """
        Adiciona padrão (Image, Palavra) ao banco de padrões
        """
        inputsFile = 'inputs.txt'
        targetsFile = 'targets.txt'
        #scipyArrayInputs = read_array(inputsFile)
        #scipyArrayTargets = read_array(targetsFile)
        try:
            scipyArrayInputs = load(inputsFile)
            scipyArrayInputs = scipyArrayInputs.tolist()
            if type(scipyArrayInputs[0]) != type([]):
                scipyArrayInputs = [scipyArrayInputs,]
        except:
            scipyArrayInputs = []
        try:
            scipyArrayTargets = load(targetsFile)

            scipyArrayTargets = scipyArrayTargets.tolist()
            if type(scipyArrayTargets[0]) != type([]):
                scipyArrayTargets = [scipyArrayTargets,]

        except:
        scipyArrayTargets = []
        inputs = self.extrator.extrairCaracteristicas(imagem)
        targets = self.palavra_para_array(palavra)
        scipyArrayInputs.append(inputs[0])
        scipyArrayTargets.append(targets)
        save(inputsFile, scipyArrayInputs)
        save(targetsFile, scipyArrayTargets)
      def loadNet(self):
             self.classificador.carregar()
      def saveNet(self):
             self.classificador.salvar()
```

```
      class Extrator:

           """Classe responsavel por extrair caracteristicas de uma
imagem"""
           def __init__(self):
           pass
           def extrairCaracteristicas2(self, imagem):
        """Recebe uma imagem como entrada e retorna um vetor de
caracteristicas"""
        lett = []
        Igray = mlab.rgb2gray(imagem)
        bw2 = mlab.im2bw(Igray)
        mlab.imshow(bw2)
        bw_7050=mlab.imresize(bw2,[70,50])
        for cnt in range(1,8):
            for cnt2 in range(1,6):
      Atemp=mlab.sum(bw_7050(range(cnt*10-9,cnt*10 +1),range(cnt2*10-
      9,cnt2*10 +1)))
             lett[(cnt-1)*5+cnt2]=mlab.sum(Atemp)
        lett=(100-array(lett))/100
        lett.shape = (1,lett.shape[0])
        return [list(left)]
       def extrairCaracteristicas(self, imagem):
        """
          Recebe uma imagem como entrada e retorna um vetor de
caracteristicas.
          Obs.: Utiliza um script do matlab para extrair características
da imagem
        """
        return mlab.extrator(imagem)
      class Classificador:
       """Classe que encapsula uma rede neural (Feed-forward neural
network)"""
       net = None
       def __init__(self, maxfun=2000):
        self.maxfun = maxfun

        self.carregar()
    def classificar(self, input):
       """
        Este método recebe um vetor de características e
        retorna um vetor onde cada elemento do vetor é um código ASCII de
uma letra da palavra
        """
        answer = self.net( input )
        return answer

    def testa(self, inputs,targets):
        print "Testando RDA.."
        output, regression = self.net.test(inputs, targets, iprint = 2)
        return output, regression

    def treinar(self, inputs,targets):
```

```
        """
        Este método recebe duas matrizes que correspondem aos INPUTS e
TARGETS que serão usados para treinar a RDA
        """


        print "Treinando RDA..."
        self.net.train_tnc(inputs, targets, maxfun = self.maxfun,
messages=1)
    def salvar(self):
        print "Salvando RDA..."
        savenet(self.net, "ocr.net")

    def carregar(self):
        try:
            self.net = loadnet("ocr.net")
        except:
            # Generate standard layered network architecture and create
network
            conec = mlgraph((35,10,10,20))
            self.net = ffnet(conec)
class Segmentador:
    """Classe responsavel por segmentar uma imagem"""
    def __init__(self):
        pass
    def segmentar(self, image):
        """
        Este método recebe como entrada uma imagem e retorna como
           sáida uma coleção de imagem, sendo que cada imagem corresponde
a uma palavra
        """
        pass
    def preSegmentar(self, image):
        """
        Este método recebe como entrada uma imagem e retorna como saída
uma coleção de coordenadas
        (poligonos) que indicam a área na imagem que possui uma palavra
        """
   if __name__=='__main__':
    ocr = OCRNetwork()
    #im = Image.open('amazingly.bmp')
    im = mlab.imread('cropImage.bmp')
    im3 = mlab.imread('discotheques2.bmp')
    #ocr.adicionarPadrao(im,'vida')
    #ocr.adicionarPadrao(im3,'discotheques')
    #ocr.treinar()
    im2 = mlab.imread('amazingly.bmp')
    palavra = ocr.classificar(im)
    palavra2 = ocr.classificar(im2)
    ocr.saveNet()
    print '-' * 50
    print 'reconheceu: '
    print '-' * 50
    print palavra,palavra2
```

APÊNDICE B – RESULTADOS DO CÓDIGO FONTE DO PROTÓTIPO – PASSO 2

```
# -*- coding: cp1253 -*-
import wx
import os
from  ocr import OCRNetwork
from bitmapToPil import *
# Módulo que contem classes/telas (GUI) que fazem a interface entre
usuário e sistema OCR.
# This is how you pre-establish a file filter so that the dialog
# only shows the extension(s) you want it to.
wildcard = "All files (*.*)|*.*"
#---------------------------------------------------------------------
---
def loadFile(self, evt):
        # Create the dialog. In this case the current directory is forced
as the starting
        # directory for the dialog, and no default file name is forced.
This can easilly
        # be changed in your program. This is an 'save' dialog.
        #
        # Unlike the 'open dialog' example found elsewhere, this example
does NOT
        # force the current worknig directory to change if the user
chooses a different
        # directory than the one initially set.
        dlg = wx.FileDialog(
            self, message="Save file as ...", defaultDir=os.getcwd(),
            defaultFile="", wildcard=wildcard, style=wx.SAVE
            )

        # This sets the default filter that the user will initially see.
Otherwise,
        # the first filter in the list will be used by default.
        dlg.SetFilterIndex(2)
        # Show the dialog and retrieve the user response. If it is the OK
response,
        # process the data.
        if dlg.ShowModal() == wx.ID_OK:
            path = dlg.GetPath()
            # Normally, at this point you would save your data using the
file and path
            # data that the user provided to you, but since we didn't
actually start
            # with any data to work with, that would be difficult.
            # The code to do so would be similar to this, assuming 'data'
contains
            # the data you want to save:
            # fp = file(path, 'w') # Create file anew
            # fp.write(data)
            # fp.close()
            # You might want to add some error checking :-)
            return path
```

```
    # Destroy the dialog. Don't do this until you are done with it!
    # BAD things can happen otherwise!
    dlg.Destroy()
    return None
class wxPyRubberBander:
""" A class to manage mouse events/ rubberbanding of a wxPython
      canvas object """

def __init__(self, canvas):
    # canvas object
    self._canvas = canvas
    # mouse selection start point
    self.m_stpoint=wx.Point(0,0)
    # mouse selection end point
    self.m_endpoint=wx.Point(0,0)
    # mouse selection cache point
    self.m_savepoint=wx.Point(0,0)
    # flags for left click/ selection
    self._leftclicked=False
    self._selected=False
    # Register event handlers for mouse
    self.RegisterEventHandlers()
 def RegisterEventHandlers(self):
    """ Register event handlers for this object """
    wx.EVT_LEFT_DOWN(self._canvas, self.OnMouseEvent)
    wx.EVT_LEFT_UP(self._canvas, self.OnMouseEvent)
    wx.EVT_MOTION(self._canvas, self.OnMouseEvent)
def OnMouseEvent(self, event):
    """ This function manages mouse events """
    if event:
        # set mouse cursor
        self._canvas.SetCursor(wx.StockCursor(wx.CURSOR_ARROW))
        # get device context of canvas
        dc= wx.ClientDC(self._canvas)
        # Set logical function to XOR for rubberbanding
        dc.SetLogicalFunction(wx.XOR)
        # Set dc brush and pen
        # Here I set brush and pen to white and grey respectively
        # You can set it to your own choices
        # The brush setting is not really needed since we
        # dont do any filling of the dc. It is set just for
        # the sake of completion.
        wbrush = wx.Brush(wx.Colour(255,255,255), wx.TRANSPARENT)
        wpen = wx.Pen(wx.Colour(200, 200, 200), 1, wx.SOLID)
        dc.SetBrush(wbrush)
        dc.SetPen(wpen)
    if event.LeftDown():
       # Left mouse button down, change cursor to
       # something else to denote event capture
       self.m_stpoint = event.GetPosition()
       cur = wx.StockCursor(wx.CURSOR_CROSS)
       self._canvas.SetCursor(cur)
```

```
# invalidate current canvas
            self._canvas.Refresh()
            # cache current position
            self.m_savepoint = self.m_stpoint
            self._selected = False
            self._leftclicked = True
         elif event.Dragging():
             # User is dragging the mouse, check if
             # left button is down
             if self._leftclicked:
                 # reset dc bounding box
                 dc.ResetBoundingBox()
                 dc.BeginDrawing()
                 w = (self.m_savepoint.x - self.m_stpoint.x)
                 h = (self.m_savepoint.y - self.m_stpoint.y)
                 # To erase previous rectangle
                 dc.DrawRectangle(self.m_stpoint.x, self.m_stpoint.y, w,h)
                 # Draw new rectangle
                 self.m_endpoint =  event.GetPosition()
                 w = (self.m_endpoint.x - self.m_stpoint.x)
                 h = (self.m_endpoint.y - self.m_stpoint.y)
                 # Set clipping region to rectangle corners
              dc.SetClippingRegion(self.m_stpoint.x, self.m_stpoint.y,w,h)
              dc.DrawRectangle(self.m_stpoint.x, self.m_stpoint.y, w, h)
              dc.EndDrawing()
              self.m_savepoint = self.m_endpoint # cache current endpoint
         elif event.LeftUp():
             # User released left button, change cursor back
             self._canvas.SetCursor(wx.StockCursor(wx.CURSOR_ARROW))
             self._selected = True  #selection is done
             self._leftclicked = False # end of clicking
        def GetCurrentSelection(self):
         """ Return the current selected rectangle """
         # if there is no selection, selection defaults to
         # current viewport
         left = wx.Point(0,0)
         right = wx.Point(0,0)
         # user dragged mouse to right
         if self.m_endpoint.y > self.m_stpoint.y:
             right = self.m_endpoint
             left = self.m_stpoint
         # user dragged mouse to left
         elif self.m_endpoint.y < self.m_stpoint.y:
             right = self.m_stpoint
             left = self.m_endpoint
         return (left.x, left.y, right.x, right.y)
        def ClearCurrentSelection(self):
         """ Clear the current selected rectangle """
        box = self.GetCurrentSelection()
         dc=wx.ClientDC(self._canvas)
         w = box[2] - box[0]
         h = box[3] - box[1]
         dc.SetClippingRegion(box[0], box[1], w, h)
         dc.SetLogicalFunction(wx.XOR)
         # The brush is not really needed since we
```

```
        # dont do any filling of the dc. It is set for
        # sake of completion.
        wbrush = wx.Brush(wx.Colour(255,255,255), wx.TRANSPARENT)
        wpen = wx.Pen(wx.Colour(200, 200, 200), 1, wx.SOLID)
        dc.SetBrush(wbrush)
        dc.SetPen(wpen)
        dc.DrawRectangle(box[0], box[1], w,h)
        self._selected = False
        # reset selection to canvas size
        self.ResetSelection()
       def ResetSelection(self):
        """ Resets the mouse selection to entire canvas """
        self.m_stpoint = wx.Point(0,0)
        sz=self._canvas.GetSize()
        w,h=sz.GetWidth(), sz.GetHeight()
        self.m_endpoint = wx.Point(w,h)
class ImageWindow(wx.ScrolledWindow):
    """Window for displaying a bitmapped Image"""
    def __init__(self, parent):
        wx.ScrolledWindow.__init__(self, parent)
        self.SetScrollRate(5,5)
        self.Bind(wx.EVT_PAINT, self.OnPaint)
    def SetBitmap(self, bitmap):
        self.bitmap = bitmap
        self.SetVirtualSize(bitmap.GetSize())
        # The following does nothing when called before the app starts.
        cdc = wx.ClientDC(self)
        cdc.DrawBitmap(self.bitmap, 0, 0)
    def OnPaint(self, event):
        try:
            dc = wx.BufferedPaintDC(self, self.bitmap)
        except:
            pass
class PageOne(wx.Panel):
    def __init__(self, parent):
        wx.Panel.__init__(self, parent)
        t = wx.StaticText(self, -1, "Gera padroes de treinamento",
(20,20))
        panel = self
        self.parent = parent
        vbox = wx.BoxSizer(wx.VERTICAL)
        vbox.Add(t)
        # panel1
        panel1 = wx.Panel(panel, -1)
        grid1 = wx.GridSizer(2, 3)
        btn1 = wx.Button(panel1, -1,'Carregar Imagem')
        btn2 = wx.Button(panel1, -1,'Adicionar')
        btn3 = wx.Button(panel1, -1,'Recortar')
        btn4 = wx.Button(panel1, -1,'Treinar')
        grid1.Add(btn1, 0,  wx.ALIGN_CENTER_VERTICAL)
        grid1.Add(btn3)
        grid1.Add(btn2)
        grid1.Add(wx.StaticText(panel1, -1, 'Patther: ', (5, 5)), 0,
wx.ALIGN_CENTER_VERTICAL)
        self.text = wx.TextCtrl(panel1, -1, size=(120, -1))
        grid1.Add(self.text)
```

```
          grid1.Add(btn4)
          panel1.SetSizer(grid1)
          vbox.Add(panel1, 0, wx.BOTTOM | wx.TOP, 9)
          # panel2
          panel2 = wx.Panel(panel, -1,size=(600, 800))
          panel2.SetBackgroundColour(wx.WHITE)
          panel2.CentreOnParent()
          hbox2 = wx.BoxSizer(wx.HORIZONTAL)
          self.wind = ImageWindow(panel2)
          self.wind.SetSize((600,400))
          self.selector = wxPyRubberBander(self.wind)
         self.wind2 = ImageWindow(panel2)
          self.wind2.SetSize((600,400))
          hbox2.Add(self.wind)
          hbox2.Add(self.wind2)
          panel2.SetSizer(hbox2)
          vbox.Add(panel2, 0, wx.BOTTOM, 9)
          self.SetSizer(vbox)
         #BINDING
          self.Bind(wx.EVT_BUTTON, self.onLoadButton, btn1)
          self.Bind(wx.EVT_BUTTON, self.onAdicionar, btn2)
          self.Bind(wx.EVT_BUTTON, self.onRecortar, btn3)
          self.Bind(wx.EVT_BUTTON, self.onTreinar, btn4)

    def onLoadButton(self, evt):
        file_path = loadFile(self, evt)
        if file_path:
                    self.picture = wx.Image(file_path,
wx.BITMAP_TYPE_ANY)
                    im =  self.picture.ConvertToBitmap()
                    self.wind.SetBitmap(im)
      self.Fit()

    def onAdicionar(self, event):
        ocr.adicionarPadrao2('cropImage.bmp', self.text.GetValue())


    def onRecortar(self, event):
        area =    self.selector.GetCurrentSelection()

        i = bitmapToPil(self.wind.bitmap)
        i2 = i.crop(area)
        i2.save('cropImage.bmp')
        #i2.show()
        self.wind2.SetBitmap( wx.EmptyBitmap(600, 400))
        self.wind2.SetBitmap(pilToBitmap(i2))
        #self.Fit()

    def onTreinar(self, event):
        ocr.treinar()
        ocr.saveNet()

class PageTwo(wx.Panel):
    def __init__(self, parent):
```

```
        wx.Panel.__init__(self, parent)
        t = wx.StaticText(self, -1, "Reconhecendo palavras", (20,20))
        panel = self
        vbox = wx.BoxSizer(wx.VERTICAL)
      vbox.Add(t)

        # panel1
        panel1 = wx.Panel(panel, -1)
        grid1 = wx.GridSizer(2, 3)
        btn1 = wx.Button(panel1, -1,'Carregar Imagem')
        btn2 = wx.Button(panel1, -1,'Reconhecer')
        grid1.Add(btn1, 0,  wx.ALIGN_CENTER_VERTICAL)
        grid1.Add(btn2)
        self.label = wx.StaticText(panel1, -1, '', (5, 5))
        #self.label.SetBackgroundColour('Red')
        grid1.Add(self.label, 0, wx.ALIGN_CENTER_VERTICAL)


        panel1.SetSizer(grid1)
        vbox.Add(panel1, 0, wx.BOTTOM | wx.TOP, 9)
        # panel2
        panel2 = wx.Panel(panel, -1,size=(500, 400))
        panel2.SetBackgroundColour(wx.WHITE)
        panel2.CentreOnParent()
        hbox2 = wx.BoxSizer(wx.HORIZONTAL)
        self.wind = ImageWindow(panel2)
        self.wind.SetSize((600,400))
        self.selector = wxPyRubberBander(self.wind)
        hbox2.Add(self.wind)
        panel2.SetSizer(hbox2)
        vbox.Add(panel2, 0, wx.BOTTOM, 9)
        self.SetSizer(vbox)
        #BINDING
        self.Bind(wx.EVT_BUTTON, self.onLoadButton, btn1)
        self.Bind(wx.EVT_BUTTON, self.onReconhecer, btn2)

    def onLoadButton(self, evt):
        file_path = loadFile(self, evt)
        if file_path:
                    self.picture = wx.Image(file_path,
wx.BITMAP_TYPE_ANY)
                    im =  self.picture.ConvertToBitmap()
                    self.wind.SetBitmap(im)
      self.Fit()

    def onReconhecer(self, event):
        area =    self.selector.GetCurrentSelection()
        i = bitmapToPil(self.wind.bitmap)
        i2 = i.crop(area)
        i2.save('cropImage.bmp')
        i2.show()
        palavra = ocr.classificar2('cropImage.bmp')
        self.label.SetLabel(palavra)
class MainFrame(wx.Frame):
     def __init__(self):
         wx.Frame.__init__(self, None, title="OCR para Escrita
Paleografica")
```

```
            # Here we create a panel and a notebook on the panel


            p = wx.Panel(self)

            nb = wx.Notebook(p)
            # create the page windows as children of the notebook


            page1 = PageOne(nb)
            page2 = PageTwo(nb)
            # add the pages to the notebook with the label to show on the
tab
            nb.AddPage(page1, "Passo 1")
            nb.AddPage(page2, "Passo 2")

            # finally, put the notebook in a sizer for the panel to manage
            # the layout
            sizer = wx.BoxSizer()
            sizer.Add(nb, 1, wx.EXPAND)
            p.SetSizer(sizer)
            self.Centre()

#Cria ocr
ocr = OCRNetwork()
ocr.loadNet()
if __name__ == "__main__":
     app = wx.App()
     frame = MainFrame().Show()
     app.MainLoop()
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo