UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA


# Sistema de Reconhecimento de Palavras Manuscritas em Domínios Restritos

Trabalho de Graduação

Everson Veríssimo da Silva

Orientador: George Darmiton da Cunha Cavalcanti

Recife, 2010

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

# Sistema de Reconhecimento de Palavras Manuscritas em Domínios Restritos

Trabalho de Graduação

Everson Veríssimo da Silva

Projeto de Graduação apresentado no Centro de Informática da Universidade Federal de Pernambuco por Everson Veríssimo da Silva, orientado pelo PhD. George Darmiton da Cunha Cavalcanti, como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: George Darmiton da Cunha Cavalcanti

Recife, 2010

*"O importante é isso: Estar pronto para, a qualquer momento, sacrificar o que somos pelo que poderíamos vir a ser."*
Charles Du Bois

# AGRADECIMENTOS

Há tanto e a tantos a agradecer por todos esses anos. Muitos desafios apareceram, mas graças a eles pude superar cada um deles. Conheci pessoas muito especiais, as quais eu tive o grande privilégio de conviver.

Primeiramente agradecer àquele a que devo tudo: Deus, por estar presente em todos os momentos.

Agradecer à minha mãe, Nancy, que me preparou para os momentos pelos quais passei, suportou minha ausência e impaciência, e fez o que pôde para me auxiliar, com dedicação e alegria. E a minha irmã, Evelyn, pelo apoio e pela ajuda oferecidos, inclusive no preparo deste trabalho.

Agradeço a George, por ter aceitado me orientar, e ter me aturado durante a realização deste trabalho. E a Leandro, pela sua grande colaboração ao trabalho, inclusive por ter me introduzido ao tema.

Aos amigos, por também conseguirem me aturar com tanta presteza. A Yane, minha primeira e grande amiga da faculdade, e eterna dupla de IP. A Marcos, por ter me ajudado tanto nos estudos e trabalhos. A Pedro, pelas caronas e músicas de Jack Johnson em seu carro que não mudavam nunca (até ele se encantar pelo brega!). A Felype, por suas manias esquisitas. A Severo, por conseguir rir das minhas conversas mais sem graça. A Nivan, por querer ver minha apresentação da monografia via Skype. E a Morato, por ter comido chocolate da rua, o que me rendeu ótimas conversas.

A todos os meus outros amigos, que sempre acreditaram em mim, e fazem parte importante de minha vida: Beth, Carol, Claríssima, Chica, Eli, Lígia e Will.

A toda nossa turma, que permaneceu grande parte unida, durante todo esse tempo, sempre se ajudando. A todos os professores que colaboraram para nossa formação, e para nossa insônia e a todos os funcionários do Centro de Informática, que prestam um ótimo serviço, sempre com bom humor.

Enfim, a todos que passaram e acrescentaram algo em minha vida, meu muito obrigado!

# RESUMO

O reconhecimento de palavras manuscritas, processo automático de identificar palavras escritas manualmente, possui diversas aplicações no mundo real, como em reconhecimento de formulários, textos manuscritos, entre outros. Para melhorar o desempenho, vários sistemas estão sendo desenvolvidos em domínios específicos, que tenham algum tipo de restrição, ao invés de domínios genéricos.

Este trabalho tem como objetivo discutir diversas questões a respeito de sistemas de domínios restritos, como quais são os tipos de restrição e suas vantagens.

O trabalho ainda objetiva demonstrar um estudo de caso com reconhecimento das palavras dos meses do ano, que traz, entre os desafios, sub-strings em comum entre as palavras dos meses. O domínio de meses, em particular, possui diversas aplicações, sendo de grande interesse os cheques bancários, para automaticamente identificar os meses escritos nos cheques, a fim de evitar fraudes e enganos.

**Palavras-chave: reconhecimento de padrões, domínio, reconhecimento local, sistemas OCR.**

# ABSTRACT

Handwritten word recognition, the process to identify handwritten words automatically, has several applications in the real world, such as in the recognition of handwritten texts and formularies. In order to improve the performance of these systems, some applications are being developed in specific domains, which have some kind of restriction, instead of generic domains.

This work discusses many questions about restricted domain systems, as what are the kinds of restriction and their advantages.

This work also aims to show a study case with the recognition of words of the months of the year, in Portuguese language, which brings some challenges, as the recognition of common sub-strings among words of the months. The domain of months of the year has several applications, such as bank checks, to automatically identify the months written on the checks, for the purpose of avoiding frauds and miswriting.

**Key-words: pattern recognition, domain, local recognition, OCR systems.**

# SUMÁRIO

1. INTRODUÇÃO ..... 1
1.1 Objetivos ..... 1
1.2 Estrutura do trabalho ..... 2
2. RESTRIÇÕES DE DOMÍNIOS ..... 3
2.1 Conceitos básicos ..... 3
2.1.1 Domínio ..... 3
2.1.2 Alfabeto e Palavras ..... 3
2.1.3 Dicionário e Vocabulário ..... 4
2.2 Linguagem e sub-linguagem ..... 4
2.2.1 Linguagem ..... 4
2.2.2 Sub-Linguagem ..... 4
2.2.3 Linguagem como um conjunto de sub-linguagens ..... 5
2.3 Domínios abertos e fechados ..... 6
2.4 Outras formas de restrição ..... 6
2.4.1 Tamanho da base ..... 7
2.4.2 Estilo de escrita ..... 7
2.5 Considerações finais de restrições de domínio ..... 8
3. SISTEMAS DE RECONHECIMENTO ..... 9
3.1 Obtenção *online versus off-line* ..... 9
3.2 Reconhecimento global *versus* local ..... 10
3.3 Reconhecimento *online versus off-line* ..... 10
3.4 Reconhecimento ótico de caracteres ..... 11
3.5 Etapas do reconhecimento de palavras manuscritas ..... 11
4. SISTEMA DE RECONHECIMENTO DAS PALAVRAS DOS MESES DO ANO. 14
4.1 Pré-processamento ..... 15
4.1.1 Binarização ..... 15
Binarização de Adamek - O' Connor ..... 15
Binarização de Otsu ..... 16
4.1.2 Eliminação de ruídos ..... 16
Cálculo de *x-height* ..... 16
4.1.3 Pré-processamento após Segmentação ..... 17
4.2 Segmentação ..... 18
4.2.1 Re-união ..... 19
4.2.2 Re-segmentação ..... 19

4.3 Extração de Características ..... 20
4.3.1 Extração *Edge Maps* ..... 20
4.3.2 Extração em Zonas ..... 21
4.4 Casamento da palavra ..... 21
2.4.3 Distância de Edição ..... 22
Algoritmo da Distância de Edição ..... 23
Algoritmo Combinacional da Distância de Edição ..... 24
5. EXPERIMENTOS E RESULTADOS ..... 27
5.1 Base de dados ..... 27
5.2 Classificadores das letras ..... 28
5.2.1 Redes Neurais Perceptron Multi-camadas ..... 28
5.2.2 Support Vector Machine ..... 29
5.3 Classificadores das palavras ..... 29
5.4 Descrição dos Experimentos ..... 30
5.4.1 Segmentação ..... 30
5.4.2 Extração de *Edge Maps* ..... 31
5.4.3 Extração em Zonas ..... 34
5.5 Sumário ..... 37
6. CONCLUSÕES ..... 38
REFERÊNCIAS ..... 39
APÊNDICE A – Protótipo desenvolvido ..... 42

# LISTA DE FIGURAS

Figura 2.1 - Exemplo de problema estruturado. ....6
Figura 2.2 - As cinco categorias de escrita [14]....8
Figura 3.1 – Aparelho eletrônico de assinatura digital [19]....9
Figura 3.2 – Módulos de um sistema de reconhecimento [3]. ....12
Figura 4.1 – Fases do sistema. ....14
Figura 4.2 - Binarização ....16
Figura 4.3 – Valores de $y_l$, $y_u$ e $y_{max}$ no cálculo do *x-height*....17
Figura 4.4 – Eliminação de Ruídos ....17
Figura 4.5 – Etapas da segmentação....19
Figura 4.6 – Segmentação ....19
Figura 4.7 –Extração de *Edge Maps*....21
Figura 4.8 - Extração de características em zonas ....21
Figura 4.9 - Erros encontrados nos padrões ....22
Figura 4.10 – Passos da distância de edição. ....23
Figura 4.11 – Exemplo de execução do algoritmo combinacional....24
Figura 5.1 – Exemplos de padrões utilizados. ....27
Figura 5.2 –Desempenho da rede de *Edge Maps*....31
Figura 5.3 – Desempenho da rede de extração em zonas. ....34

# LISTA DE TABELAS

Tabela 5.1 - Taxa de erro da segmentação.....30
Tabela 5.2 - Parâmetros da Rede Neural de *Edge Maps*.....31
Tabela 5.3 - Taxas de acerto e de DE Mínima de *Edge Maps* e RN.....32
Tabela 5.4 - Matriz de Confusão de *Edge Maps* e RN. .....32
Tabela 5.5 - Taxas de acerto e de DE Mínima de *Edge Maps* e SVM. .....33
Tabela 5.6 - Matriz de Confusão de *Edge Maps* e SVM. .....33
Tabela 5.7 - Parâmetros da Rede Neural da Extração em Zonas.....34
Tabela 5.8 - Taxas de acerto e de DE Mínima de Extração em Zonas e RN. .....35
Tabela 5.9 - Matriz de Confusão de Extração em Zonas e RN. .....35
Tabela 5.10 - Taxas de acerto e de DE Mínima de Extração em Zonas e SVM. .....36
Tabela 5.11 - Matriz de Confusão de Extração em Zonas e SVM.....36
Tabela 5.12 - Taxas de Acerto de cada experimento.....37

# 1. INTRODUÇÃO

Reconhecimento de padrões consiste em classificar padrões a partir de informações extraídas *a priori*. Encontrar um rosto numa imagem, reconhecimento de fala e classificação de documentos são alguns de seus exemplos.

Entre as aplicações de reconhecimento de padrões, encontram-se aquelas que reconhecem padrões em palavras, como reconhecimento de caligrafia e reconhecimento de palavras. A primeira refere-se em classificar o autor da escrita a partir da maneira como ele escreve, num campo conhecido como biometria (*i.e.*, estudo de características individuais únicas, capazes de identificar um indivíduo). Reconhecimento de palavras, no entanto, refere-se a classificar uma palavra, seja ela digitada ou escrita por uma pessoa.

O reconhecimento de palavras manuscritas, processo de identificar palavras escritas manualmente, possui diversas aplicações no mundo real, tais como processamento automático de cheques bancários, envelopes postais, formulários, textos manuscritos, entre outros. Seu estudo, além de aplicabilidade, possui muitos desafios, o que tem atraído muitos pesquisadores para a área.

Em busca de melhores taxas de acerto, vários pesquisadores restringiram o domínio e desenvolveram técnicas para o tipo de restrição escolhido. Apesar de tais restrições limitarem a aplicabilidade dos sistemas, seu uso vem crescendo, justificado pela modulação de sistemas complexos em vários mais simples, restritos, o que resulta numa melhora global do desempenho.

Uma vez definido o tipo de domínio que se queira, o projetista deve planejar como será seu sistema, é necessário fazer certas escolhas, antes mesmo de coletar os dados, *e.g.*, o tipo de reconhecimento (local ou global, *online* ou *off-line*), se o problema será estruturado, entre outras. Além disso, algumas técnicas podem ser específicas para determinado tipo de restrição.

## 1.1 Objetivos

O objetivo deste trabalho é fazer um estudo sobre algumas escolhas concernentes a domínios e a etapas do sistema, que devem ser feitas antes de desenvolver um sistema.

Em seguida, um sistema como estudo de caso foi desenvolvido, dentro do domínio das palavras dos meses do ano na língua portuguesa. As várias etapas do trabalho são descritas, bem como o porquê de certas escolhas do sistema.

Os objetos de estudo são as palavras dos 12 meses do ano, escritos na língua portuguesa. Um dos grandes desafios do sistema foi tratar palavras com sub-strings em comum: Jan**eiro**, Fever**eiro**, **Ma**rço, Abril, **Ma**io, **Ju**n**ho**, **Ju**l**ho**, Agosto, Set**embro**, Outu**bro**, Nov**embro** e Dez**embro**.

## 1.2 Estrutura do trabalho

O trabalho foi dividido primeiro no estudo dos domínios e depois no estudo de caso. O capítulo 2 descreve os diversos tipos de restrição que se pode escolher para um sistema, algumas questões relacionadas aos mesmos, como vantagens e problemas que podem ser encontrados. O capítulo 3 dedica-se a descrever o sistema em si, os tipos de sistemas utilizados e suas fases. O capítulo 4 apresenta o estudo de caso dos meses do ano, e algumas implicações que puderam ser feitas da escolha do domínio escolhido. Os experimentos e resultados do sistema desenvolvido são descritos no capítulo 5. Finalmente, o capítulo 6 faz as últimas considerações e propõe trabalhos futuros.

# 2. RESTRIÇÕES DE DOMÍNIOS

*"By studying the properties of natural languages... we may hope to gain some understanding of the specific characteristics of human intelligence."*
Chomsky

Antes do desenvolvimento de um sistema de reconhecimento de palavras manuscritas, é necessário fazer certas escolhas relacionadas a domínios. Assim, um projetista dirá se seu sistema é voltado a um tema específico ou será mais abrangente, se permitirá que outras palavras que não estejam na base de dados possam ser classificadas ou serão descartadas, entre outras decisões. Portanto, um bom estudo dos tipos de domínios ajudará a desenvolver um sistema mais apropriado à necessidade da aplicação.

## 2.1 Conceitos básicos

### 2.1.1 Domínio

**Domínio** significa um conjunto de elementos que possuem uma ou mais propriedades em comum. Por exemplo, em matemática, tem-se o domínio dos números naturais, cuja propriedade em comum é ser um inteiro não-negativo. Em sistemas de reconhecimento, o contexto no qual um problema está inserido pode ser chamado de domínio (*e.g.*, nomes de países, cores, números, são alguns dos domínios que podem ser tratados por tais sistemas).

### 2.1.2 Alfabeto e Palavras

**Alfabeto** (também representado por ∑) é um conjunto finito de símbolos [1] que se pode encontrar nos elementos de um domínio. A esses elementos, dá-se o nome de **palavras**. O alfabeto do conjunto dos números naturais são os algarismos de '0' a '9'. As palavras são cada um dos números naturais. Vale ressaltar que palavra não é apenas uma seqüência de símbolos do alfabeto, a palavra "*funny*", por exemplo, não é válida para o domínio das palavras em português, apesar de ser em inglês.

### 2.1.3 Dicionário e Vocabulário

**Dicionário** é definido como sendo um conjunto de todos os elementos (palavras) de uma linguagem [2]. Assim, o dicionário dos números naturais são todos os números naturais, portanto de tamanho infinito. O dicionário da língua portuguesa são todas as palavras em português, mas apesar de possuir um alfabeto parecido com a língua inglesa, os dicionários das duas linguagens são diferentes.

**Vocabulário** é um conjunto de palavras num determinado domínio. A diferença entre dicionário e vocabulário é que o segundo é um subconjunto do primeiro. Pode-se falar em dicionário ou vocabulário da língua portuguesa, mas não se pode falar em dicionário dos nomes de países, mas sim o vocabulário dos nomes de países. Alguns autores ainda definem léxico como o vocabulário da base de dados [3, 4].

## 2.2 Linguagem e sub-linguagem

### 2.2.1 Linguagem

Chomsky define **linguagem** como sendo "*um conjunto (finito ou infinito) de sentenças*", e que "*o fundamental propósito na análise de uma linguagem L é separar as seqüências gramaticais, que pertencem a L, das seqüências não gramaticais, que não são seqüências de L*". Uma linguagem pode pertencer a vários domínios: inglês, português, e até mesmo um sistema matemático formalizado. Por algumas linguagens serem infinitas, é necessário um artifício que possa classificar as seqüências em gramaticais ou não gramaticais a L. A esse artifício dá-se o nome de **gramática** [5].

Apesar de os estudos de Chomsky terem revolucionado a lingüística, e servirem como princípio para alguns conceitos da teoria da computação, e até mesmo o surgimento das linguagens de programação, eles não serão aprofundados neste documento, apesar de sistemas de reconhecimento de palavras os utilizarem para ganhar desempenho [6, 7]. Na verdade, o estudo da lingüística está mais relacionado com extração de informação e processamento em linguagem natural, que fogem ao escopo do trabalho.

### 2.2.2 Sub-Linguagem

Grishman *et al*. [8] definem **sub-linguagem**, como o próprio nome sugere, a uma linguagem usada por um grupo particular de locutores, ou uma linguagem com um tema ou um vocabulário específico, mas que estejam inseridos numa linguagem mais abrangente.

Ao conjunto de palavras da linguagem, dá-se o nome de dicionário, e da sub-linguagem de vocabulário. Apesar de linguagem e sub-linguagem serem bem parecidas, muitos sistemas preferem a segunda por ser mais simples e ajudar a entender problemas maiores. Por essa razão, foram utilizados sistemas com sub-linguagem no início do desenvolvimento de sistemas inteligentes [9], e em várias aplicações atualmente, como o sistema TESSA [10], voltado para surdos, cujo domínio são as agências de correios, mas com a finalidade de primeiro validar um pequeno protótipo para partir para sistemas mais complexos.

### 2.2.3 Linguagem como um conjunto de sub-linguagens

Lendaris *et al.* [11] propuseram um sistema de redes neurais (RN) modulares, decompondo o domínio em pequenas sub-tarefas, uma vez que uma RN tende a piorar seu desempenho à medida que aumenta o tempo de treinamento e a quantidade de pesos na rede. RN's com cerca de mil entradas são boas candidatas à modulação, as sub-tarefas geram redes com topologias menores, e consequentemente mais eficientes.

Apesar de redes modulares servirem para um problema genérico, eles exemplificam bem a prática de "*dividir para conquistar*". Por ser mais simples tratar domínios de sub-linguagem, mesmo em problemas de linguagem, muitos sistemas procuram modular o problema em sub-problemas de mais fácil resolução. O desenvolvedor tem que conhecer bem a respeito do problema em questão, mas se faz necessário que seja um **problema estruturado**.

Para melhor ilustrar, a Figura 2.1 mostra um exemplo de problema estruturado. Em um cheque, na seção de "cidade" e "data", reconhecer cada uma das palavras usando um sistema único é mais complicado. Portanto, o problema pode ser simplificado segmentando a sentença em cidade, dia, mês e por fim ano, e utilizar um sistema específico para cada sub-linguagem.

Problemas de formulários são bons candidatos a serem modulados, desde que sejam pensados assim antes de coletarem os dados. Um problema inicialmente grande pode ser reduzido a vários problemas menores, *e.g.*, reconhecer idade, CEP, país, sexo, nome, entre outros. Daí a necessidade do projetista do sistema ter noção de domínios, para melhor planejar seu sistema.


**Figura 2.1 - Exemplo de problema estruturado.**

Mas infelizmente, nem todos os problemas são estruturáveis e reduzíveis a problemas menores. Nestes casos, uma grande base única de dados é necessária, embora outros problemas surjam, o que será discutido na seção 2.4.

## 2.3 Domínios abertos e fechados

Contreras [12] define **estruturas fechadas** como "*aquelas que não aceitam elementos que não sejam interpretáveis*". No caso de sistemas de reconhecimento de palavras, uma palavra (um elemento) que não seja interpretável é uma palavra que não esteja no vocabulário, ou base de dados. Assim, um sistema fechado que tenha sido treinado com palavras de animais não pode aceitar uma palavra de objeto, por exemplo.

De forma análoga, define-se **domínio aberto** como aquele capaz de aceitar outras palavras que não estejam na base de dados [13]. O reconhecimento se dá através de comparações com o padrão a ser classificado com padrões da base e uma vez que é pouco viável uma base conter todas as palavras do dicionário de linguagens complexas, nesses casos, os padrões que devem estar na base são os símbolos (as letras). Isso só será possível se o reconhecimento da palavra for local, *i.e.*, se a palavra puder ser segmentada em caracteres (ou símbolos do alfabeto), classificar cada caractere, e em seguida classificar a palavra. Apesar do vocabulário da base ser limitado, é preciso que se encontre a palavra no dicionário, a fim de validá-la ou rejeitá-la.

Sistemas de domínio aberto possuem taxas de acerto bem mais baixas que seus equivalentes de domínios limitados [3]. Uma das formas de se simular um domínio aberto utilizando domínio fechado é possuir léxicos grandes.

## 2.4 Outras formas de restrição

Um domínio que possui algum tipo de restrição é conhecido como **domínio restrito**. Sub-linguagens e domínios fechados são exemplos de domínios restritos. Sistemas de domínio restrito são, no geral, mais eficientes em relação aos não restritos equivalentes, já que são sistemas mais específicos e conhecem mais a respeito de seu domínio e de seus problemas típicos.

### 2.4.1 Tamanho da base

Uma das formas de restringir um domínio é limitar seu tamanho. Grande parte dos sistemas compara o padrão a ser classificado com os padrões da base de treinamento. Portanto, quanto maior a base, tende-se a levar mais tempo para efetuar a classificação. Além disso, maiores as chances de palavras parecidas estarem presentes, e consequentemente, maiores as taxas de erros de classificação. Koerich *et al.* [3] usam a seguinte classificação para o tamanho do vocabulário:

- Vocabulário pequeno – dezenas de palavras;
- Vocabulário médio – centenas de palavras;
- Vocabulário grande – milhares de palavras;
- Vocabulário muito grande – dezenas de milhares de palavras.

Devido aos problemas apresentados por vocabulários muito grandes, a maioria dos pesquisadores dedica-se a vocabulários restritos de dez a mil palavras, e para sistemas pequenos, não é dada muita atenção à complexidade computacional e velocidade de classificação [3], que são bem mais críticos em domínios maiores.

Apesar do grande uso de vocabulários não muito grandes, eles apresentam alguns problemas, como domínios fechados e eventos raros [14]. O primeiro refere-se ao fato de muitas palavras não poderem ser classificadas simplesmente porque não se encontram na base. O segundo, à frequência de palavras na base, enquanto umas aparecem frequentemente, outras aparecem de forma escassa. Eventos raros são bem mais difíceis de serem escolhidos por classificadores em que a frequência dos padrões na base influencia a classificação, como redes neurais e KNN (os k vizinhos mais próximos).

### 2.4.2 Estilo de escrita

Uma das grandes dificuldades de sistemas de reconhecimento de palavras manuscritas é a grande variabilidade na escrita. Assim, pessoas escrevem as letras em um estilo diferente entre si e até mesmo a mesma pessoa raramente escreve duas letras de forma idêntica. Por isso, vários sistemas existem para palavras digitadas, cuja variabilidade é menor.

De acordo com Tappert *et al.* [15], a variação pode ocorrer em propriedades estáticas e dinâmicas. As variações estáticas que podem ocorrer são na forma e no tamanho. As variações dinâmicas são no número e na ordem dos riscos. A segunda é

capturada apenas em sistemas de obtenção online (discutidos no capítulo 3), os quais, além das coordenadas x e y, há uma terceira em relação ao tempo. O grau de variação depende tanto do estilo quanto da velocidade da escrita. Ainda de acordo com Tappert *et al.*, há cinco graus de variação de escrita, como pode ser visto na Figura 2.2.

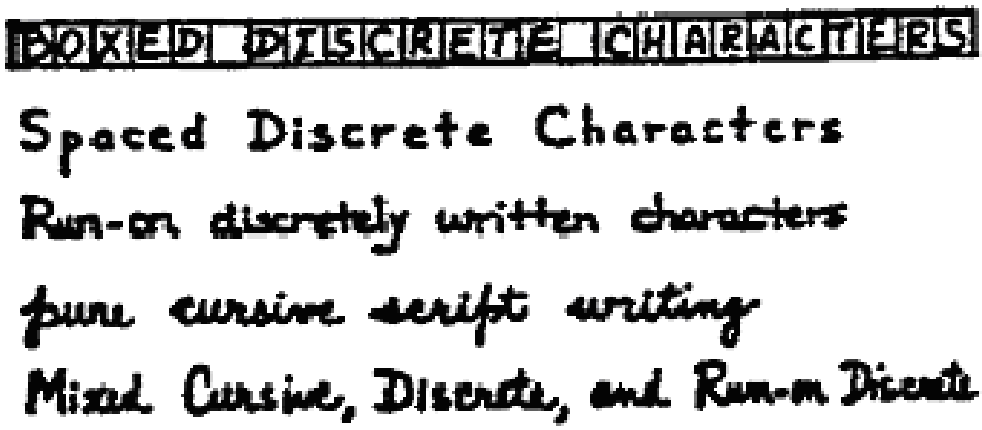



**Figura 2.2 - As cinco categorias de escrita [14].**

Muitos sistemas utilizam vocabulário irrestrito para estilo de letra, como o desenvolvido por Nathan *et al.* [16], permitindo que o usuário escreva tanto palavras em letra de fôrma, quanto cursiva, ou combinação das duas formas. No entanto, a fim de melhorar o desempenho, alguns sistemas impõem mais esse tipo de restrição.

## 2.5 Considerações finais de restrições de domínio

Apesar da melhora nas taxas de acerto, restrições limitam a aplicabilidade dos sistemas. No entanto, em muitos casos, muitos problemas podem ser estruturados para serem divididos em subproblemas em que cada um deles é tratado por um sistema específico. Além disso, pode haver combinação de sistemas, por exemplo, um sistema específico para letras de fôrma e outro para letras cursivas combinados podem tratar de palavras em ambos os formatos. Modular é uma prática bastante comum em computação, o que justifica restringir os sistemas de reconhecimentos.

No estudo de caso deste relatório, o sistema está restringido a palavras dos meses do ano, cujo tamanho do léxico é doze (vocabulário pequeno), não aceitando qualquer outra palavra (fechado), e restringido a letras de fôrma (ver Figura 2.3).

O próximo capítulo irá tratar sobre sistemas de reconhecimento, e de algumas decisões que devem ser feitas no sistema que mais se adéqüe à restrição do domínio.

# 3. SISTEMAS DE RECONHECIMENTO

Um sistema de reconhecimento deve ser desenvolvido a partir das necessidades do problema, e isso envolve naturalmente a espécie de domínio a ser tratada. Este capítulo visa a estudar os tipos de sistemas existentes e relacioná-los com os domínios.

## 3.1 Obtenção *online versus off-line*

Para uma palavra ser reconhecida automaticamente, é necessário digitalizá-la de alguma maneira. Há duas formas de fazê-lo, digitalização através de *scanners* da palavra escrita num papel, ou através de uma caneta ou superfície eletrônica, como um digitalizador [17]. Na Figura 3.1, pode-se ver um exemplo de aparelho utilizado para assinaturas digitais.

As duas formas são conhecidas como digitalização *online* e digitalização *off-line*, respectivamente. Na abordagem *online*, mais informações da dinâmica do processo de escrita, como ordem de escrita, pressão e velocidade, podem ser obtidos, e, portanto, resulta em melhores taxas de reconhecimento [18].

**Figura 3.1 – Aparelho eletrônico de assinatura digital [19].**

Com o uso crescente de aparelhos eletrônicos, como *personal digital assistants* (PDA's) e celulares digitais, que permitem a escrita digital, métodos de obtenção de dados não baseados em teclado, *e.g.*, através de canetas e voz, estão cada vez mais presentes [20]. No início, boa parte de suas aplicações envolvia poucas palavras e em domínios bastante restritos, além de uma menor capacidade de processamento desses aparelhos, não sendo ainda muito comum o uso de vocabulários extensos. No entanto, com a melhora dos processamentos de tais aparelhos e a possibilidade de uso de servidores de maior capacidade, já que tais aparelhos estão conectados à Internet, as aplicações tendem a crescer em complexidade.

Por outro lado, na abordagem *off-line*, apenas a imagem da palavra está disponível [21], e, portanto, apenas informações referentes a espaço e à luminosidade podem ser extraídas. Apesar disso, sistemas *off-line* são muito mais utilizados por serem mais universais, *i.e.*, podem ser usados em qualquer lugar, além de ser de mais fácil obtenção. Ainda há muitas aplicações que permitem apenas a obtenção *off-line*, como em cheques bancários.

## 3.2 Reconhecimento global *versus* local

Ainda há uma classificação que diz respeito à forma de extração de características: global, o padrão é toda a palavra, visa-se extrair características de toda a palavra; e local, visa-se segmentar a palavra em letras, para posteriormente classificar individualmente cada letra [4].

A abordagem global é possível apenas caso a palavra a ser reconhecida esteja na base de treinamento, *i.e.*, se a linguagem for pequena ou o domínio for fechado, uma vez que, para classificar um padrão, suas informações precisam estar na base de dados, e como toda a palavra é um padrão, não dá para classificar uma palavra que não esteja na base.

A abordagem local, por outro lado, permite ambos os tipos de domínio (aberto e fechado), uma vez que os padrões que se quer reconhecer são as letras, e a base é todo o alfabeto. Desde que a palavra contenha apenas letras do alfabeto, uma palavra fora do vocabulário de treinamento pode ser reconhecida [22]. Apesar disso, é recomendável o uso de um dicionário, a fim de corrigir possíveis erros de classificação da classificação das letras, e evitar gerar palavras ininteligíveis.

A procura pela palavra no dicionário é justamente o pós-processamento, comumente utilizado em sistemas de reconhecimento local. Várias técnicas podem ser utilizadas para tal fim. A utilizada no trabalho descrito deste documento é uma versão adaptada da distância de edição.

## 3.3 Reconhecimento *online versus off-line*

Tappert *et al*. [15] diferenciam reconhecimento *online* do *off-line* (diferentemente de obtenção *online* e *off-line*). O primeiro é o reconhecimento realizado durante a escrita da palavra, ou seja, reconhecimento em tempo real. As considerações a serem feitas são de tempo, da capacidade de processamento do computador e de algoritmos eficientes. Tappert *et al*. afirmam que o reconhecimento deve ser rápido o suficiente para acompanhar a escrita

do usuário, e que a maioria dos sistemas comerciais disponíveis conseguem uma taxa de um a dois caracteres reconhecidos por segundo. Como o reconhecimento é letra por letra (local), domínios abertos são possíveis, mas por serem mais complexos, e para manter a eficiência, aplicam-se alguns tipos de restrição, como limitar o vocabulário, por exemplo.

Já em sistemas de reconhecimento *off-line*, o reconhecimento só é feito depois que a palavra já foi escrita, portanto, são menos críticos no quesito de tempo. Pela sua natureza, sistemas de reconhecimento *online* também possuem obtenção *online*, já os de reconhecimento *off-line* podem ter obtenção tanto *online* quanto *off-line*.

## 3.4 Reconhecimento ótico de caracteres

Alguns sistemas, conhecidos como reconhecimento ótico de caracteres (*Optical character recognition* - OCR), são umas das aplicações mais bem sucedidas de reconhecimento de padrões.

Mori *et al*. [23] descrevem a evolução de sistemas OCR, bem como a grande quantidade de produtos comerciais disponíveis no mercado, mostrando quão bem estabelecidos tais sistemas se tornaram. A ideia básica desses sistemas é, a partir de arquivos de imagem, gerar documentos que possam ser editáveis, além de simplificar e agilizar a recuperação de informação desses documentos.

Muitos dos sistemas comerciais são de domínio aberto, ou semi-aberto, por isso possuem reconhecimento local, embora alguns sejam restritos a uma linguagem.

## 3.5 Etapas do reconhecimento de palavras manuscritas

Um sistema de reconhecimento de palavras manuscritas pode ter diferentes etapas, de acordo com o tipo de sistema que se escolha. Assim, um sistema de reconhecimento global, por exemplo, não possui segmentação.

Koerich *et al*. [3] fazem uma descrição completa dos principais módulos que pode haver num sistema.

- Obtenção da imagem: como visto na seção 3.1, pode ser *online* ou *off-line*;
- Pré-processamento: objetiva reduzir variabilidades indesejáveis na imagem. Operações como *slant* e *slope*, normalização e limpeza de ruído fazem parte do pré-processamento;
- Segmentação: presente apenas em sistemas de reconhecimento local, objetiva quebrar a palavra em letras;

- Extração de características: tem por finalidade extrair características discriminantes das palavras para que possam servir como representação das mesmas;
- Treinamento: consiste em treinar o sistema com exemplos para que o mesmo possa "aprender" com o treinamento e seja capaz de classificar um novo padrão;
- Reconhecimento: das palavras (ou das letras, caso seja local);
- Pós-processamento: por exemplo, se o sistema for local, tenta reconhecer as palavras formadas pelas letras reconhecidas;
- Mecanismo de rejeição: caso o grau de incerteza da classificação seja alto, pode-se rejeitar a classificação de uma palavra, a fim de reduzir a taxa de erro do sistema.

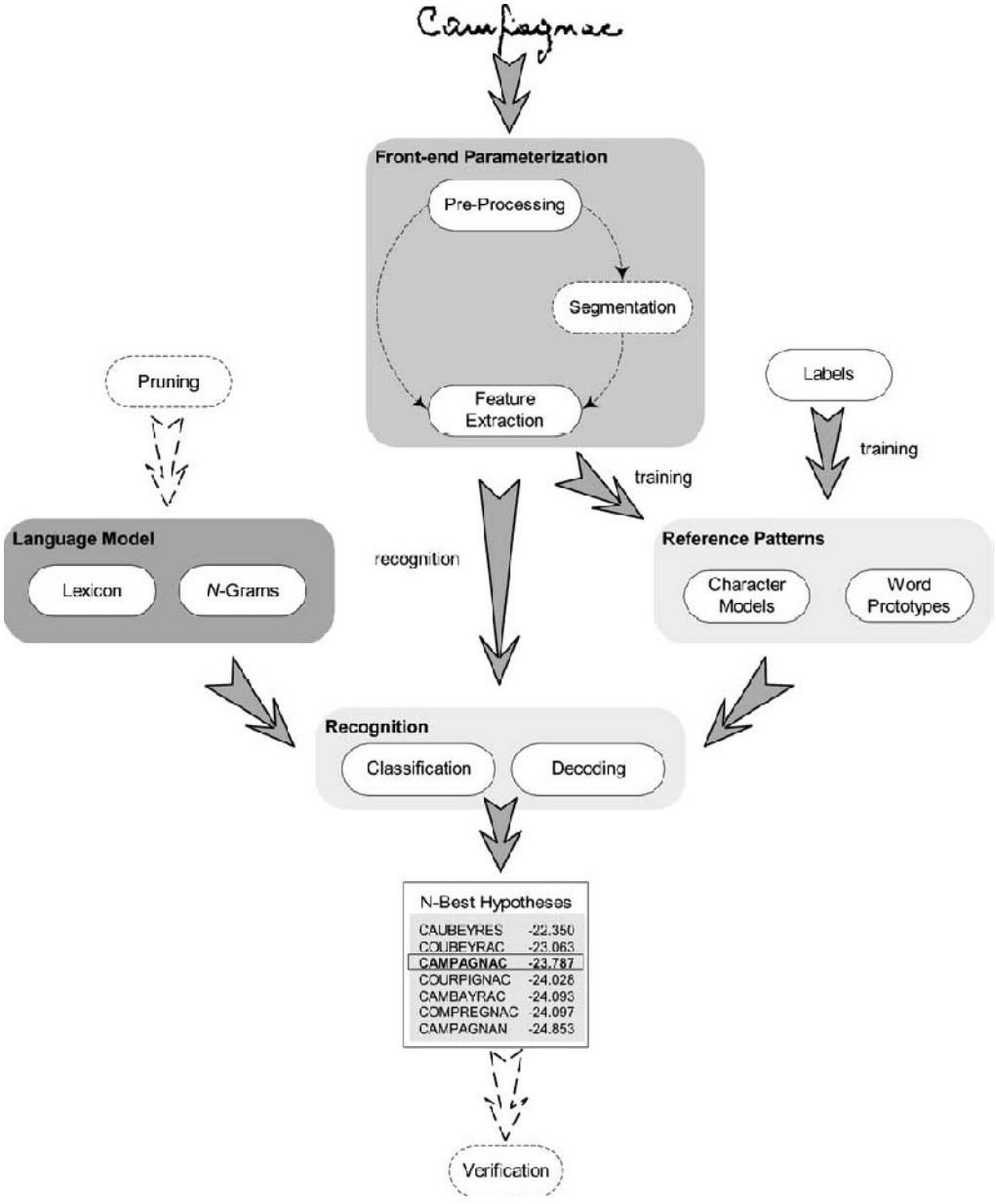



**Figura 3.2 – Módulos de um sistema de reconhecimento [3].**

O sistema desenvolvido possui obtenção *off-line* e reconhecimento *off-line* e local. O próximo capítulo dedica-se a descrever as seguintes etapas desenvolvidas: pré-processamento, segmentação, extração de características, treinamento, reconhecimento das letras, e pós-processamento.

# 4. SISTEMA DE RECONHECIMENTO DAS PALAVRAS DOS MESES DO ANO

Este capítulo mostra na prática os impactos da escolha do domínio e de suas restrições num sistema de reconhecimento de palavras manuscritas. O léxico possui 12 palavras, as palavras estão em letra de fôrma e é fechado a meses do ano.

Meses do ano estão presentes em vários documentos utilizados no dia-a-dia, como em documentos oficiais, em formulários e em cheques bancários. No caso de cheques bancários, a avaliação de cheques é manual, e devido ao grande volume de cheques utilizados, isso demanda muito tempo e dinheiro, além de entediante e sujeito a erros humanos. Com a automação, há um grande ganho de eficiência em produção e em custo [17].

O problema possui um pequeno vocabulário, mas algumas peculiaridades. Algumas palavras são muito parecidas, como *"Março"* e *"Maio"*, e *"Junho"* e *"Julho"*. Além disso, muitas palavras possuem sub-strings em comum, como *"Janeiro"* e *"Fevereiro"*.

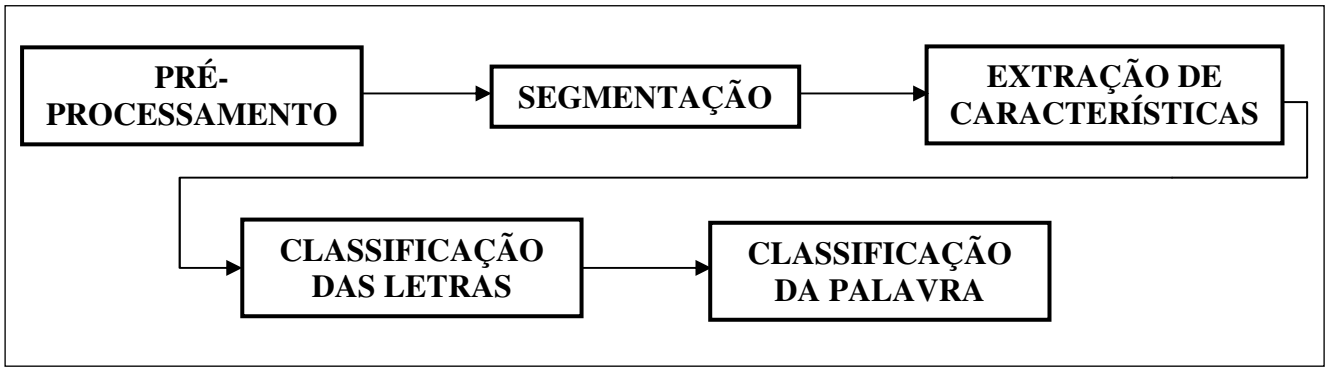



**Figura 4.1 – Fases do Sistema.**

O sistema também é local, o que implica numa fase de segmentação e outra de busca da palavra no dicionário. A Figura 4.1 mostra as fases do sistema. Primeiro, o pré-processamento, que elimina fatores que não contribuem para a classificação e deixa os padrões menos heterogêneos. Em seguida a segmentação, que divide a palavra em letras. De cada letra, são extraídas as características, que por sua vez são classificadas. Ao fim, a palavra formada é comparada às que estão no dicionário e a classe final é retornada.

## 4.1 Pré-processamento

O pré-processamento envolve vários passos com o objetivo de melhorar a segmentação e a classificação dos padrões. Tais passos podem acontecer antes, durante ou mesmo depois da segmentação.

O primeiro passo, conhecido como binarização, é transformar as imagens em dois valores (0 – representando papel e 1 – representando tinta). No entanto, foram utilizadas duas técnicas diferentes para binarização, uma para facilitar a segmentação e outra a classificação. Em seguida, a eliminação de ruídos da imagem e os últimos passos, após a segmentação, são corrigir o *slant* da letra e encontrar o *bound-box* da imagem.

### 4.1.1 Binarização

Como exposto, há duas técnicas usadas para binarização. A primeira produz palavras mais suaves, o que facilita a classificação das letras. A segunda produz uma binarização mais parecida com a imagem original, e, portanto, tem um papel importante na segmentação do sistema. Ambos os métodos, no entanto, comparam o valor de cada pixel a um limiar calculado para definir se este pixel representa papel ou tinta.

#### Binarização de Adamek - O' Connor

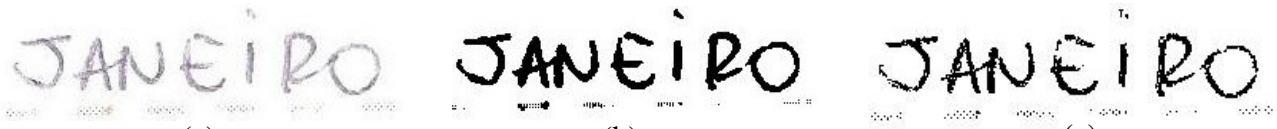
Adamek *et al.* [24] apresenta um método, baseado no algoritmo de Niblack [25], para definir, a partir de uma imagem em escala de cinza, se um pixel é um ou zero de acordo com seu valor e de seus vizinhos. Devido a diferenças em pressão durante a escrita e dependendo do instrumento utilizado, definir apenas um limiar para toda a imagem pode não ser o desejável. Por isso o limiar é definido dinamicamente pela fórmula de Sauvola *et al.* [26] para cada pixel:

$$L = \mu(1 - \mathrm{k}\left(1 - \frac{\sigma}{\mathrm{R}}\right))$$

Onde $\mathrm{k}$ e $\mathrm{R}$ são constantes cujos valores definidos são 0.02 e 128 respectivamente, definidos por Adamek *et al.*. $\mathrm{R}$ representa a dinâmica do desvio-padrão (σ). A média (μ) e o desvio-padrão (σ) são calculados para a janela, cujo tamanho escolhido foi 3x3, no qual o pixel é o centro. O valor de $L$ é o limiar que definirá se o pixel será um ou zero. No entanto, como essa fórmula resulta numa imagem grossa, e difícil de ser segmentada, uma pequena modificação foi feita para torná-la mais fina:

$$L = \mu(1 - \mathrm{k}\left(\frac{\sigma}{\mathrm{R}}\right))$$

O resultado final pode ser visto na Figura 4.2.b.



(a) (b) (c)

**Figura 4.2 - Binarização. (a) Imagem Original. (b) Adamek - O' Connor. (c) Otsu.**

**Binarização de Otsu**

Otsu [27] apresenta uma binarização com limiar único para a imagem completa. Os pixels são comparados por este valor para definir se são um ou zero. Os resultados podem ser vistos na Figura 4.2.c.

A imagem gerada, no entanto, não é suave como a gerada pelo método de Adamek - O' Connor. No entanto, como durante a escrita, quando se passa de uma letra para outra, a pressão feita é menor, pelo método de Otsu, essa passagem terá menos pixels pretos, e, portanto, será mais fácil reconhecer um ponto de corte na palavra.

### 4.1.2 Eliminação de ruídos

A Figura 4.2.b revela que há certas manchas na imagem que, por não fazerem parte da palavra, pioram o desempenho do sistema. Para eliminar esses ruídos, o método utilizado foi o de Adamek *et al.* [24].

A imagem a ser limpa foi a gerada pela binarização de Adamek - O' Connor. Primeiramente, a palavra foi dividida em componentes conexos-8. A limpeza é baseada na altura das letras minúsculas da palavra (*x-height*), e o limiar para definir se um componente é ruído, e, portanto, ser eliminado, é se o número de pixels do componente é menor que 5% do valor de *x-height*$^2$ (o valor definido pelo autor foi 10%, mas nos testes realizados, 5% obteve uma limpeza mais apropriada). Para calcular o valor de *x-height*, foi utilizado o método apresentado por Adamek *et al.* [24].

**Cálculo de *x-height***

Define-se $y_l$ e $y_u$ os pontos na coordenada y que limitam as letras minúsculas da palavra. As posições desses valores na figura podem ser vistos na Figura 4.3.

Para calcular tais valores, define-se a função $P_v(y)$, que retorna o número de pixels pretos na linha y da coordenada vertical e $P_h(x)$ seu equivalente na coluna x da coordenada horizontal. Define-se ainda $y_{max}$ como sendo a coordenada y no qual se encontra a maior

quantidade de pixels pretos, *i.e.*, possui o máximo valor de $P_v$. Os valores de $y_u$ e $y_l$ são encontrados pelas linhas mais próximas de $y_{max}$ que satisfaçam as seguintes condições:

$y_u$: $y > y_{max}$ e $P_v(y) < \alpha_u P_v(y_{max})$
$y_l$: $y < y_{max}$ e $P_v(y) < \alpha_l P_v(y_{max})$

onde a $\alpha_u$ e $\alpha_l$ são atribuídos por Adamek *et al.* os valores 0.5 e 0.3 respectivamente. O valor de *x-height* é a diferença do topo e da base das letras minúsculas, em outras palavras ($y_u - y_l$).

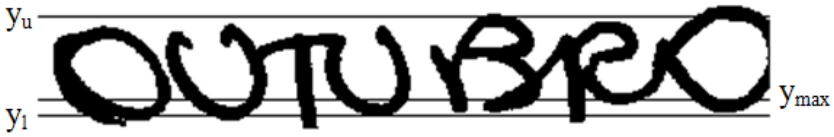



**Figura 4.3 – Valores de $y_l$, $y_u$ e $y_{max}$ no cálculo do *x-height*.**

O resultado final da eliminação de ruídos pode ser visto na Figura 4.4.b. Riscos do formulário e o ponto do 'i' foram eliminados. Percebe-se que pontuações e acentos são pequenos o suficiente para serem eliminados na limpeza, o que não é crítico no sistema em questão.


(a) (b)

**Figura 4.4 – Eliminação de Ruídos. (a) Binarização de Adamek - O' Connor. (b) Imagem limpa.**

### 4.1.3 Pré-processamento após Segmentação

Duas técnicas bastante utilizadas em reconhecimento de palavras manuscritas foram utilizadas logo após a segmentação a fim de normalizar os padrões, *bounding box* e correção de *slant*.

As letras continuam com muitos espaços livres, o que dificulta a classificação. Portanto, o tamanho da imagem precisa ser reduzido o suficiente para conter a letra, e ao espaço que limita a letra dá-se o nome de *bound box*. O *bound box* é limitado pelas primeiras colunas da esquerda à direita e da direita à esquerda que possuam pixel preto, e pelas primeiras linhas com pixel preto de cima para baixo e de baixo para cima.

Palavras escritas manualmente comumente apresentam uma inclinação (*slant*), o método utilizado para corrigi-la foi o proposto por Perantonis *et al.* [28].

## 4.2 Segmentação

Segmentação possui um papel importante em sistemas de reconhecimento local por ser crítico e ter um impacto grande no resultado final. A segmentação apresentada possui uma estratégia clássica de acordo com Casey *et al.* [22],entre três:

- clássica: que divide a imagem, através de operações chamadas de dissecação, em sub-imagens, que são então classificadas;
- por classificação: explícita através de janelas predefinidas, ou implícita através de características de toda a palavra;
- híbrida: que usa dissecação juntamente com classificação.

De acordo com Bunke *et al.* [29], há três questões principais a serem consideradas em relação à segmentação,:

- é muito difícil extrair caracteres da palavra, simplesmente porque há caracteres que se assemelham à combinação de outros, como ‘d’ e ‘cl’, ‘m’ e ‘rn’;
- uma má segmentação pode levar a um reconhecimento errôneo, até mesmo a palavras inexistentes, e portanto, um pós-processamento para encontrar a melhor palavra no dicionário é necessário;
- nem sempre o léxico está disponível, como por exemplo uma sequência de números.

A segmentação utilizada usa ambas as imagens binárias, mas as operações de dissecação são realizadas apenas na binarizada pelo método de Adamek *et al.*. A imagem binarizada pelo método de Otsu serve apenas para encontrar um bom ponto de corte. Os valores apresentados foram todos obtidos empiricamente.

Por se ter restringido o sistema apenas a palavras em letras de fôrma, a segmentação é específica para esse formato. Como algumas letras já estão separadas, o primeiro passo foi dividir a imagem em componentes conexos-8, como já foi feito durante a limpeza. O resultado pode ser visto na Figura 4.5.a, bem como alguns problemas típicos, que precisam de duas estratégias diferentes:

- re-segmentação: quando duas ou mais letras ainda estão conectadas (caso de ‘R’ com ‘O’ na Figura 4.5.a);
- re-união: quando uma letra possui partes desconexas (caso de ‘E’ e ‘B’ na Figura 4.5.a);


**Figura 4.5 – Etapas da Segmentação - (a) Divisão em componentes. (b) Re-união. (c) Re-segmentação.**

### 4.2.1 Re-união

Compara-se cada componente com todos os outros. Aqueles que possuem uma interseção maior que 60% da largura de um dos componentes são reunidos (ver Figura 4.6.a). Essa estratégia é capaz de unir, na maioria das vezes, as partes desconexas da letra 'E' e também o ponto do 'I' ao resto da letra, quando esse ponto é grande o suficiente para passar pela limpeza. O resultado da re-união pode ser visto na Figura 4.5.b.

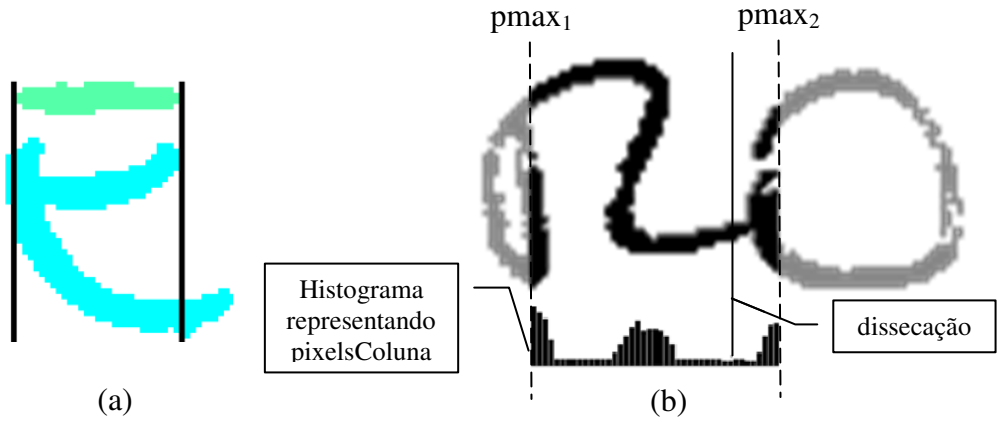



**Figura 4.6 – Segmentação - (a) Re-união. (b) Re-segmentação.**

### 4.2.2 Re-segmentação

Primeiro, seja *mHeight* a média das alturas dos componentes conexos-8. Aproximadamente, essa é a largura de uma letra. Portanto, todos os componentes 35% mais largos que o valor de *mHeight* são candidatos à re-segmentação.

O bloco é dividido ao meio e, de cada metade, é calculado o pixelsColuna como sendo a sequência da quantidade de pixels pretos em uma coluna na imagem binarizada pelo método de Otsu. Sejam $pmax_1$ e $pmax_2$ as colunas com mais pixels pretos para o primeiro e o segundo blocos respectivamente. A ideia é que $pmax_1$ e $pmax_2$ sejam os

"centros de massa" de cada letra. De $pmax_1$ a $pmax_2$, é onde se dará a dissecação (ver Figura 4.6.b), encontrando o ponto onde a conexão é mínima, percorrendo o pixelsColuna em uma janela de tamanho quatro. No centro da janela que possua média mínima e que intersecte o mínimo possível na palavra será feito o corte. Uma vez que um componente pode possuir mais de uma letra, este processo é repetido até que não haja mais subcomponentes 35% mais largos que o valor de *mHeight*.

Esse processo pode causar uma supersegmentação. Portanto, uma re-união pode ser necessária, mas numa estratégia diferente da anterior. Se no componente original houver sub-componentes contínuos não 35% mais largos que *mHeight* (isso pode acontecer caso haja mais de duas letras conexas), eles são reunidos. Ainda, se houver subcomponentes contínuos no qual um deles tem o número de pixels menor que 20% do número de pixels do outro, eles também são reunidos. Isso pode acontecer quando, por exemplo, uma letra é muito grande e é dividida em dois.

## 4.3 Extração de Características

Para classificar as letras segmentadas, elas precisam ser transformadas em uma representação diferente, como um conjunto de características. Dois métodos de extração de características foram utilizados.

### 4.3.1 Extração *Edge Maps*

Cruz *et al.* [30] propuseram uma técnica de extração de características baseada em *Edge Maps*. Primeiramente, a imagem é redimensionada para 25x25 pixels e o método de Zhang-Suen para thinning (afinamento) é utilizado [31]. Os segmentos de linha horizontal, vertical, diagonais esquerdo e direito são encontrados na imagem através de quatro operadores de Sobel [32], respectivamente:

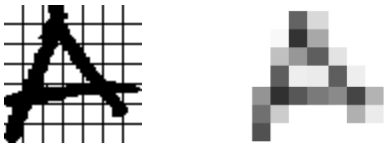
$$\begin{bmatrix} -1 & -1 & -1 \\ +2 & +2 & +2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & +2 & -1 \\ -1 & +2 & -1 \\ -1 & +2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & +2 \\ -1 & +2 & -1 \\ +2 & -1 & \end{bmatrix} \quad \begin{bmatrix} +2 & -1 & -1 \\ -1 & +2 & -1 \\ -1 & -1 & +2 \end{bmatrix}$$

A aplicação dos quatro operadores na imagem afinada gera quatro imagens, que podem ser vistas na Figura 4.7. Todas as quatro imagens são divididas em 25 quadrados de mesmo tamanho, e para cada um, a porcentagem de pixels pretos é calculada, o que gera 25 características para cada uma das imagens. O mesmo é feito para a imagem afinada, o que resulta num total de 125 características por padrão.

**Figura 4.7 –Extração de Edge Maps - Da esquerda à direita: letra redimensionada, edge maps horizontal, vertical, diagonal direita e diagonal esquerda.**

### 4.3.2 Extração em Zonas

A extração das características, baseadas em zonas e apresentadas por Perantonis *et al.* [28] consiste em dividir a imagem em pequenos quadrados de tamanho 10x10 e calcular a quantidade de pixels pretos dividida pela quantidade de pixels em cada um desses quadrados. A imagem foi redimensionada para 70x70, o que gera um total de 49 características. O resultado pode ser visto na Figura 4.8.



(a) (b)

**Figura 4.8 - Extração de características em zonas. (a) Divisão em quadrados. (b) Características extraídas**

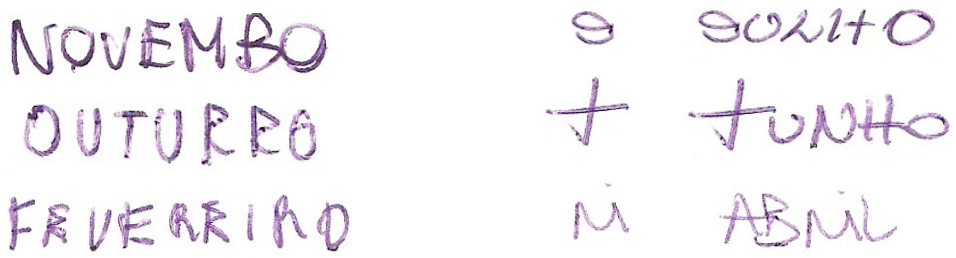
## 4.4 Casamento da palavra

Após a segmentação e a classificação das letras, a palavra resultante precisa ser comparada com as do dicionário, uma vez que muitas dessas palavras não casam perfeitamente com nenhuma das classes. Além disso, a segmentação pode falhar, causando um número diferente de letras, e o usuário pode escrever a palavra erroneamente.

Figura 4.9.a mostra exemplos de palavras do padrão que foram escritos erroneamente. Assumindo uma classificação perfeita das letras, as palavras resultantes seriam "*Novembo*", "*Outurro*" e "*Feverriro*". Além disso, algumas letras se assemelham mais a outras, como pode ser visto na Figura 4.9.b, na qual os "J" de "*Julho*" e de "*Junho*" e o "RI" de "*Abril*" se assemelham mais a "O", "T" e "M", respectivamente.

Há várias alternativas para busca da palavra, como Modelo Escondido de Markov [33, 34] e Distância de Edição. Casey *et al.* [22] apresentam várias abordagens para

classificar a palavra (que não necessariamente precisam classificar primeiro todas as letras, o que agiliza o processo em sistemas de vocabulários grandes). O sistema desenvolvido utiliza Distância de Edição.




(a) (b)

**Figura 4.9 - Erros encontrados nos padrões. (a) Falta ou substituição de uma letra. (b) Letras mais parecidas com outras.**

### 2.4.3 Distância de Edição

Distância de edição (DE) é definida como a menor distância para transformar uma string *s* em outra *t* [35]. Por exemplo, para transformar a palavra "*house"* em "*horses",* primeiro transforma-se "*u"* em *"r"* e acrescenta-se a letra *"s"* no final. O custo total dessa transformação é a soma dos custos de cada operação (substituir uma letra e acrescentar outra). Portanto, calculando a palavra do dicionário *t* mais próxima da palavra formada pelas letras classificadas *s* (a que tem a menor DE), encontra-se a classe referida.

No entanto, como alguns classificadores retornam mais que simplesmente um resultado por letra, essa abordagem pode falhar em desempenho. Seja *n* o número de resultados por letra. Por exemplo, se *t* for "*Fevereiro"*, que possui nove letras, e assumindo uma boa segmentação, se, para cada letra, o classificador retornar três opções (*n = 3*), seria necessário, para cada combinação dessas letras, executar uma vez o algoritmo de DE. Uma vez que há $3^9$ combinações, a execução completa é exponencial, e, portanto, intratável.

Uma alteração no algoritmo original é utilizada, chamada de Algoritmo Combinacional de Distância de Edição, que reduz a complexidade do problema anterior para *n* vezes o custo de execução do algoritmo original de DE. O algoritmo é capaz de encontrar a combinação *s* mais próxima de cada classe *t*, e a DE entre ambas. Para definir o custo da transformação de *s* em *t*, é necessário definir os custos de cada operação possível:

- Inserção: quando um caractere de *s* é adicionado;
- Deleção: quando um caractere de *t* é removido;
- Substituição: quando um caractere de *s* é substituído por outro de *t*;
- *Matching*: quando os caracteres de *s* e de *t* são iguais.

Para os propósitos do sistema, os custos das três primeiras operações são um, enquanto o custo da última é zero. Então, a DE entre duas strings é a mínima soma dos custos das operações aplicadas na transformação.

**Algoritmo da Distância de Edição**

O algoritmo de DE usa programação dinâmica para resolver o problema, cuja ideia básica está no Teorema 1.

**Teorema 1:** a menor distância para ir do ponto A ao C passando por B é a menor distância de A a B mais a menor distância de B a C.

O algoritmo cria uma matriz onde as linhas representam a string *s* e as colunas a string *t*. Assumindo *cost(I,J)* a DE para transformar *s* de tamanho *I* em *t* de tamanho *J*, e *cost(i,j)* a DE para transformar os *i* primeiros caracteres de *s* nos *j* primeiros caracteres de *t*; toda célula *(i,j)* contém o valor de *cost(i,j)*. Assume-se ainda que a célula *(0,0) = 0* é o ponto de partida *A*.

Inicialização

| | | H | O | U | S | E |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| H | 1 | | | | | |
| O | 2 | | | | | |
| R | 3 | | | | | |
| S | 4 | | | | | |
| E | 5 | | | | | |
| S | 6 | | | | | |

Passo intermediário

| | | H | O | U | S | E |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| H | 1 | 0 | 1 | 2 | 3 | 4 |
| O | 2 | 1 | 0 | x | | |
| R | 3 | | | | | |
| S | 4 | | | | | |
| E | 5 | | | | | |
| S | 6 | | | | | |

Conclusão

| | | H | O | U | S | E |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| H | 1 | 0 | 1 | 2 | 3 | 4 |
| O | 2 | 1 | 0 | 1 | 2 | 3 |
| R | 3 | 2 | 1 | 1 | 2 | 3 |
| S | 4 | 3 | 2 | 2 | 1 | 2 |
| E | 5 | 4 | 3 | 3 | 2 | 1 |
| S | 6 | 5 | 4 | 4 | 3 | 2 |

**Figura 4.10 – Passos da Distância de Edição.**

O primeiro passo do algoritmo é completar a primeira linha com a soma cumulativa dos custos de inserção e a primeira coluna com a soma cumulativa dos custos de deleção, como é mostrado na primeira matriz da Figura 4.10.

Para preencher cada uma das células *(i,j)* corretamente, baseado no Teorema 1, escolhe-se a soma mínima do custo para atingir cada uma das células imediatamente anteriores a partir de A e o custo de atingir a célula *(i,j)*. Felizmente, de apenas três células é possível atingir a célula *(i,j)* – a saber: *(i-1, j)*, *(i, j-1)* e *(i-1,j-1)*. Na segunda matriz da Figura 4.10, vê-se que o valor de *x* é min(2+1, 0+1, 1+1) = 1.

Portanto, para calcular *célula (i,j)*:

- *I* = *custo (i, j-1)* + (custo de inserção);
- *D*: *custo (i-1, j)* + (custo de deleção);
- *S*: *custo (i-1, j-1)* + (custo de substituição ou *matching*);
- *custo (i,j)* = *min (I, D, S)*;

O algoritmo preenche linha por linha (como na segunda matriz da Figura 4.10) ou coluna por coluna, até que a *célula (I, J)* seja preenchida.

**Algoritmo Combinacional da Distância de Edição**

O algoritmo combinacional de DE também usa programação dinâmica baseada no Teorema 1 e nas fórmulas para calcular *célula (i, j)*. A primeira diferença é que as células são preenchidas apenas coluna por coluna.

Cada coluna *j* é associada a *n* letras, exceto a coluna usada na inicialização (que representa o caractere vazio). O que o algoritmo faz é criar *n* cópias, cada uma associada a uma letra, para preencher a coluna *j* utilizando o algoritmo regular de DE. A matriz com *custo (I, j)* menor é escolhida.

(a)

| | | M | H | J | O |
|---|---|---|---|---|---|
| | | A | A | I | U |
| | | V | F | L | G |
| | 0 | | | | |
| M | 1 | | | | |
| A | 2 | | | | |
| I | 3 | | | | |
| O | 4 | | | | |

(b)

| | | M |
|---|---|---|
| | 0 | 1 |
| M | 1 | 0 |
| A | 2 | 1 |
| I | 3 | 2 |
| O | 4 | 3 |

(c)

| | | A |
|---|---|---|
| | 0 | 1 |
| M | 1 | 1 |
| A | 2 | 1 |
| I | 3 | 2 |
| O | 4 | 3 |

(d)

| | | V |
|---|---|---|
| | 0 | 1 |
| M | 1 | 1 |
| A | 2 | 2 |
| I | 3 | 3 |
| O | 4 | 4 |

**Figura 4.11 – Exemplo de execução do algoritmo combinacional.**

No entanto, Figura 4.11 apresenta um exemplo onde há custos iguais para mais de uma letra. Isto se deve ao fato de mais de uma letra ter números de match iguais, mesmo que em regiões diferentes. Em alguns casos, escolher uma em preferência a outra não faz diferença. No entanto, como no caso da Figura 4.11, se alguma letra apresentar um match em passos futuros, o resultado pode ser diferente e não representar um DE melhor.

A estratégia utilizada é escolher a letra com o match mais recente. Isso leva a custos menores em passos futuros. O artifício para encontrar a coluna com match mais recente mostrado no Algoritmo 4.1 foi escolher a matriz na qual a soma da coluna *j* é a menor.

**Algoritmo 4.1 – *Distância de Edição Combinacional***

**Entrada:**

*N ← quantidade de resultados por letra*
*s ← matriz com a combinação das letras retornadas pelo classificador (de tamanho N x I)*
*t ← vetor com a palavra que se quer encontrar a distância (de tamanho J)*

**Saída:**

*word ← a melhor combinação de s encontrada (a que apresenta menor distância em relação a t)*
*d[I,J] ← distância de word a t.*

**Pseudo-código:**

```
declare d[0..I, 0..J]
declare palavra

para i de 0 a I
        d[i, 0] ← i //deleção
para j de 0 a J
    d[0, j] ← j //inserção

para j de 1 a J
{
    declare m[0..I, 0..J, 1..N]

        para n de 1 a N
        {
            m[0..I, 0..J, n] ← d[0..I, 0..J]

            para i de 1 a I
            {
                        se s[n, i] = t[j] então
                                m[i, j, n] ← d[i-1, j-1, n]
                    senão
                        ins ← d[i, j-1, n]   + 1
                                del ← d[i-1, j, n]   + 1
                        sub ← d[i-1, j-1, n] + 1
                        m[i, j, n] ← min(del, ins, sub)
            }
        }

        [minVal, índice] ← min( soma(m(I,0..J,1..N)) );
        d ← m(0..I, 0..J, índice);
        palavra ← palavra + s(índice, j);

}

return palavra, d[I,J];
```

No caso de *n = 1*, a execução do algoritmo combinacional é igual à do algoritmo regular coluna por coluna. A prova de corretude (*i.e.*, o valor retornado pelo algoritmo é o menor, o correto) do combinacional e do regular está na Prova 1.

**Prova 1:** ambos os algoritmos são corretos – retornam a menor distância de *s* (no caso do segundo algoritmo, a melhor combinação *s*) a *t*. O primeiro passo é igual para ambos os algoritmos. Todas as demais células são preenchidas pelo cálculo da célula (i, j).

Por indução:

**Hipótese:** a célula (i, j) está preenchida com o menor custo a partir da origem (0,0)

**Caso Base:** a célula (0, 0) está preenchida com o menor custo a partir da origem.

Por definição, custo (0, 0) = 0, e como a distância deve possuir um número n ≥ 0, a célula (0, 0) está preenchida com o menor custo.

**Caso Indutivo:** assumindo que as células (k-1, l-1), (k-1, l), (k, l-1), a célula (k, l) possui a menor distância em relação à origem.

Como já foi visto, só há três pontos imediatamente anteriores da célula (k, l), justamente (k-1, l-1), (k-1, l), (k, l-1), e a partir das fórmulas para o cálculo da célula (i, j):

- *I* = *custo (k, l-1)* + (custo de inserção) = menor custo (k, l) passando por (k, l-*1*);
- *D*: *custo (k-1, l)* + (custo de deleção) = menor custo (k, l) passando por (k-*1*, l);
- *S*: *custo (k-1, l-1)* + (custo de substituição ou matching) = menor custo (k, l) passando por (k-1, l-1).

Portanto, só há três valores possíveis para célula (k, l), a saber, *I*, *D* e *S*. Escolhendo o mínimo dos três valores encontra-se o menor custo (k, l).

Como a prova é indutiva, conclui-se que é verdade para toda célula (i, j) até (I, J), que é a menor distância da palavra *s* à *t*.

O próximo capítulo visa a descrever os experimentos realizados com as técnicas apresentadas neste capítulo, bem como avaliar e relacionar os resultados obtidos.

# 5. EXPERIMENTOS E RESULTADOS

Os experimentos foram divididos em etapas, primeiro o desenvolvimento de classificadores para as letras e depois a classificação das palavras. Foram utilizados dois tipos de classificadores: redes neurais e *Support Vector Machine* (SVM), treinados utilizando uma base de letras. Em seguida, cada palavra da base foi testada utilizando os classificadores gerados e as técnicas descritas neste documento.

Todo o sistema foi desenvolvido através do MATLAB, e do toolbox para manipulação de imagens.

## 5.1 Base de dados

Para a base de dados dos meses, foram coletadas 2995 assinaturas de meses do ano, escritas por diferentes pessoas e em formato de letra de fôrma. Em média, foram cerca de 250 padrões coletados para cada mês do ano, digitalizados a 300dpi. A Figura 5.1 mostra exemplos de padrões coletados para cada classe.

Para a classificação das letras, no entanto, foi utilizado um subconjunto do banco de letras de Camastra *et al*. [36]. Por se tratar de um banco com letras cursivas, apenas as letras que se assemelham a letras de fôrma foram selecionadas. Além disso, apenas vinte letras aparecem pelo menos uma vez nas classes, as demais letras, portanto, foram igualmente descartadas do banco de letras.

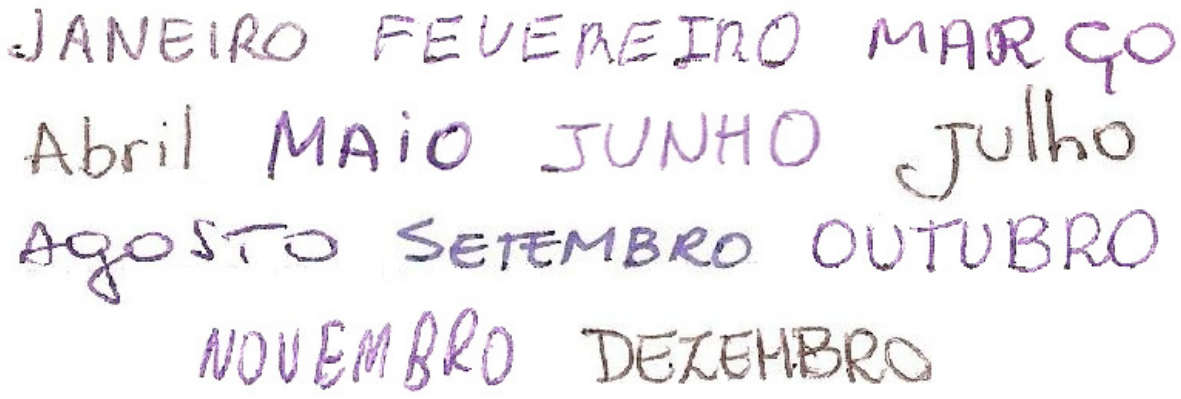



**Figura 5.1 – Exemplos de padrões utilizados.**

## 5.2 Classificadores das letras

As bases foram geradas a partir do banco de letras. O problema de classificar as letras foi restringido em letras de fôrmas, e em apenas vinte classes (‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘J’, ‘L’, ‘M’, ‘N’, ‘O’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’, ‘Z’), já que esse é o tamanho do alfabeto efetivamente utilizado no problema de classificação dos meses. Foram geradas duas bases, pela extração em zonas e de *Edge Maps*.

Cada conjunto de dados foi balanceado, para não haver classes com mais padrões que outras. Redes neurais são particularmente bastante sensíveis a bases não balanceadas. Cada classe passou a possuir 2233 padrões, totalizando em 44660 padrões.

### 5.2.1 Redes Neurais Perceptron Multi-camadas

Baseados no sistema neurológico humano, redes neurais artificiais são ótimos classificadores para resolver problemas complexos, como classificar letras, por exemplo.

Para encontrar os parâmetros, as bases das letras foram divididas em 50% treinamento, 25% teste e 25% validação. Os parâmetros que foram alterados em busca do melhor classificador foram o número de neurônios escondidos, a taxa de aprendizagem, o algoritmo de aprendizagem e de treinamento, e a função de ativação. O critério para definir o melhor classificador foi a taxa de acerto (fração de padrões certos sobre total de padrões). Após encontrar bons parâmetros através dos resultados obtidos nas bases de teste, os classificadores foram novamente treinados, mas dessa vez com a base de teste fazendo parte do conjunto de treinamento. Com mais padrões sendo apresentados aos classificadores, eles tendem a generalizar mais. Finalmente, os classificadores puderam ser utilizados na etapa seguinte.

Uma segunda opção era encontrar os bons parâmetros gerando o classificador e testando na base de palavras, mas como isso requer muito tempo (tempo de treinamento + tempo de teste da base), essa opção foi descartada.

O toolbox de redes neurais do MATLAB foi utilizado para gerar os classificadores.

### 5.2.2 Support Vector Machine

SVMs criam hiperplanos que representam bem as classes, baseados na base de treinamento. É um dos classificadores mais utilizados para resolver problemas de grande complexidade.

Toda a base foi utilizada para treinar o classificador, e para classificar um padrão, foi utilizado comparação um contra um (constrói um SVM para cada par de padrões [37]), e o vencedor é o padrão classificado. O toolbox de SVM de Canu *et al*. [38] foi utilizado para gerar os classificadores. A função de *kernel* escolhida foi a polinomial homogênea, por ser de treinamento mais rápido.

## 5.3 Classificadores das palavras

Depois do treinamento dos classificadores, a base inteira foi testada seguindo o processo mostrado na Figura 4.1. Para cada bloco segmentado, o classificador da rede neural retorna as três classes de letras com maiores *scores*. Portanto, para encontrar a melhor combinação, o algoritmo combinacional de DE foi utilizado, com $n = 3$. Já pela natureza do classificador SVM, apenas uma letra é retornada, e, portanto, o algoritmo combinacional com $n = 1$ foi utilizado, que é equivalente ao regular.

Depois de encontrar a melhor combinação da string *s* para cada classe de mês, a classe que apresenta a menor DE em relação à sua combinação é escolhida. No entanto, em muitos casos, houve empate em várias classes. Portanto, dois valores foram calculados, a taxa de acerto, que é a taxa de padrões corretos pelo total, e a taxa de DE Mínima, como a taxa de padrões que apresentam DE mínima para a classe correta, mesmo que ela não tenha sido escolhida (em caso de empate com a classe vencedora). Em outras palavras, se o critério de desempate fosse ótimo, a segunda taxa seria igual à taxa de acerto, ou ainda, a taxa de DE Mínima é o limite superior da taxa de acerto dos classificadores gerados.

Para a rede neural, o critério de desempate foi a combinação que apresenta a maior soma dos scores gerados pelo classificador. Já para o SVM, a primeira classe com a distância de edição mínima.

## 5.4 Descrição dos Experimentos

### 5.4.1 Segmentação

A técnica proposta para segmentar os padrões conseguiu segmentar corretamente a maior parte das letras. Uma letra é considerada mal segmentada quando se torna ilegível. Mesmo que o corte não seja ideal para separar duas letras, se uma delas for identificável e outra não, contabiliza-se apenas uma letra mal segmentada. A Tabela 5.1 mostra a quantidade de letras segmentadas erroneamente. No total, 13,97% das letras se tornaram ilegíveis após a segmentação. A contabilidade foi feita manualmente, analisando cada padrão.

**Tabela 5.1 - Taxa de erro da segmentação**

| Meses do ano | Número de Padrões | Número total de letras | Letras Mal Segmentadas | |
|---|---|---|---|---|
| | | | Absoluto | Relativo (%) |
| Jan | 253 | 1771 | 193 | 10,89 |
| Fev | 245 | 2205 | 217 | 9,84 |
| Mar | 250 | 1250 | 203 | 16,24 |
| Abr | 246 | 1230 | 113 | 9,18 |
| Mai | 247 | 988 | 166 | 16,8 |
| Jun | 267 | 1335 | 167 | 12,5 |
| Jul | 232 | 1160 | 150 | 12,93 |
| Ago | 248 | 1488 | 243 | 16,33 |
| Set | 248 | 1984 | 268 | 13,5 |
| Out | 248 | 1736 | 291 | 16,76 |
| Nov | 248 | 1984 | 326 | 16,43 |
| Dez | 263 | 2104 | 351 | 16,68 |
| Total | 2995 | 19235 | 2688 | 13,97 |

Algumas letras apresentaram muita dificuldade de serem segmentadas, como o “M”, que foi identificado como duas letras, pelo seu tamanho, e foi segmentado. Duplas de letras com “l”, como “El” e “Al”, que por serem pequenas, foram identificadas como uma única letra, e não foram segmentadas em muitos casos.

### 5.4.2 Extração de *Edge Maps*

#### Redes Neurais

A configuração da melhor rede com *back-propagation* encontrada pode ser vista na Tabela 5.2. A taxa de acerto encontrada para a classificação das letras foi de 87,84% com essa configuração. Os erros médio-quadrados para o conjunto de treinamento, validação e teste foram respectivamente 0,00617, 0,01149 e 0,011.

O treinamento foi bastante rápido, atingindo convergência em 197 épocas. O gráfico relacionando os erros médio-quadrados de treinamento e de validação com as épocas pode ser visto na Figura 5.2.

**Tabela 5.2 - Parâmetros da Rede Neural de *Edge Maps*.**

| | |
|---|---|
| Número de neurônios escondidos | 100 |
| Taxa de Aprendizagem | 0,02 |
| Quantidade máxima de falhas na validação | 5 |
| Algoritmo de Aprendizagem | Gradiente Descendente |
| Algoritmo de Treinamento | *Backpropagation* Resiliente |
| Função de Ativação | Sigmoide Logística |

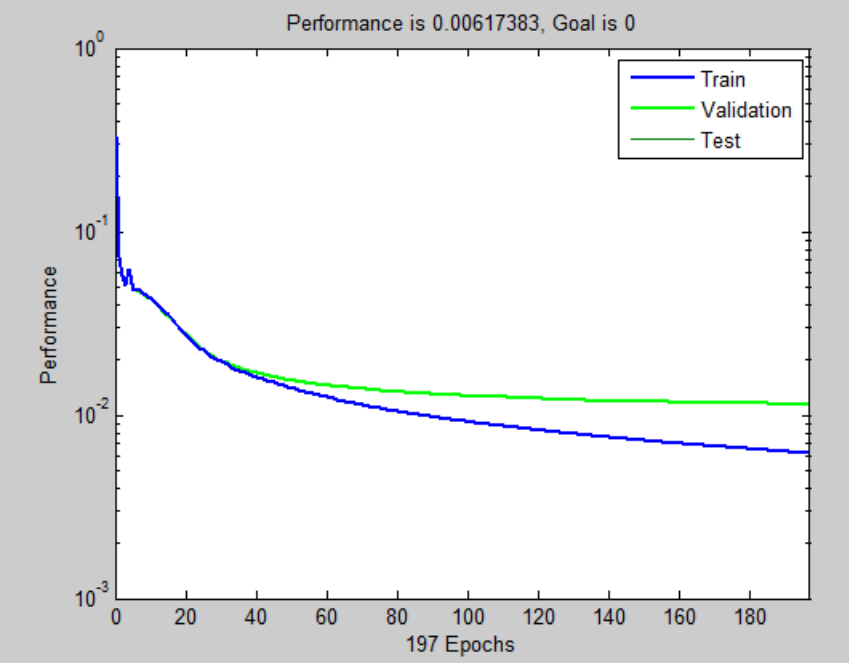



**Figura 5.2 –Desempenho da rede de *Edge Maps*.**

A tabela 5.3 mostra as taxas de acerto e de DE Mínima obtidas para cada mês. O experimento escolheu corretamente 93,28% ($\frac{86,31}{92,52}$) das vezes em que a classe correta apresentou DE mínima. O experimento conseguiu uma taxa de acerto global de 86,31%.

**Tabela 5.3 - Taxas de acerto e de DE Mínima de *Edge Maps* e RN.**

| Meses | Acerto (%) | DE Mínima (%) | Desempate correto (%) |
|---|---|---|---|
| Jan | 85,77 | 92,49 | 92,73 |
| Fev | 91,43 | 91,84 | 99,55 |
| Mar | 78,80 | 90,80 | 86,78 |
| Abr | 95,93 | 97,56 | 98,32 |
| Mai | 76,11 | 87,85 | 86,63 |
| Jun | 91,76 | 97,00 | 94,59 |
| Jul | 85,78 | 97,41 | 88,06 |
| Ago | 87,90 | 91,94 | 95,60 |
| Set | 90,32 | 93,55 | 96,54 |
| Out | 89,11 | 92,34 | 96,50 |
| Nov | 89,11 | 92,34 | 96,50 |
| Dez | 74,14 | 85,55 | 86,66 |
| Global | 86,31 | 92,52 | 93,28 |

**Tabela 5.4 - Matriz de Confusão de Edge Maps e RN.**

| | J | F | M | A | M | J | J | A | S | O | N | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Janeiro | 217 | 1 | 5 | 2 | 0 | 13 | 2 | 6 | 3 | 1 | 2 | 1 |
| Fevereiro | 2 | 224 | 5 | 5 | 1 | 0 | 1 | 1 | 2 | 2 | 1 | 1 |
| Março | 1 | 0 | 197 | 7 | 34 | 1 | 4 | 6 | 0 | 0 | 0 | 0 |
| Abril | 0 | 0 | 0 | 236 | 5 | 1 | 1 | 3 | 0 | 0 | 0 | 0 |
| Maio | 1 | 0 | 41 | 3 | 188 | 3 | 3 | 5 | 0 | 3 | 0 | 0 |
| Junho | 3 | 0 | 1 | 0 | 4 | 245 | 7 | 1 | 1 | 4 | 1 | 0 |
| Julho | 1 | 0 | 1 | 0 | 5 | 22 | 199 | 2 | 0 | 2 | 0 | 0 |
| Agosto | 0 | 0 | 8 | 16 | 4 | 1 | 0 | 218 | 1 | 0 | 0 | 0 |
| Setembro | 2 | 4 | 2 | 3 | 2 | 4 | 3 | 1 | 224 | 1 | 0 | 2 |
| Outubro | 2 | 0 | 2 | 1 | 4 | 5 | 2 | 1 | 5 | 221 | 4 | 1 |
| Novembro | 1 | 3 | 5 | 4 | 4 | 1 | 0 | 2 | 2 | 3 | 221 | 2 |
| Dezembro | 2 | 5 | 2 | 5 | 5 | 5 | 2 | 4 | 31 | 5 | 2 | 195 |

"Abril" obteve a maior taxa de acerto devido a pouca semelhança com outras classes e também porque apresentou boa segmentação. Já classes parecidas, como "Março" e "Maio", tiveram taxas menores, e muitos padrões de uma classe foram classificados erroneamente como sendo de outra, como se vê na matriz de confusão (Tabela 5.4), e na diferença das taxas de acerto e de DE mínima. Já "Junho" e "Julho" apresentaram boas taxas, apesar de serem também parecidas, mas apresentaram melhores segmentações.

***Support Vector Machine***

SVM apresentaram ótimos resultados, como se vê na coluna de DE Mínima da Tabela 5.5, mas devido ao critério de escolha, a taxa de acerto mostrou-se maior nas primeiras classes que nas últimas, resultando numa taxa de acerto global menor (86,11%) que por Redes Neurais (86,31%).

**Tabela 5.5 - Taxas de acerto e de DE Mínima de *Edge Maps* e SVM.**

| Meses | Acerto (%) | DE Mínima (%) | Desempate correto (%) |
|---|---|---|---|
| Jan | 90,12 | 90,12 | 100 |
| Fev | 88,57 | 91,02 | 97,30 |
| Mar | 87,20 | 88,40 | 98,64 |
| Abr | 97,15 | 97,97 | 99,16 |
| Mai | 84,21 | 94,33 | 89,27 |
| Jun | 96,25 | 98,13 | 98,08 |
| Jul | 81,47 | 94,83 | 85,91 |
| Ago | 82,66 | 88,71 | 93,18 |
| Set | 90,32 | 95,16 | 94,91 |
| Out | 86,29 | 96,77 | 89,17 |
| Nov | 82,26 | 93,55 | 87,93 |
| Dez | 66,92 | 88,59 | 75,53 |
| Global | 86,11 | 93,12 | 92,47 |

**Tabela 5.6 - Matriz de Confusão de *Edge Maps* e SVM.**

| | J | F | M | A | M | J | J | A | S | O | N | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Janeiro | 228 | 0 | 9 | 3 | 3 | 7 | 1 | 2 | 0 | 0 | 0 | 0 |
| Fevereiro | 11 | 217 | 3 | 5 | 1 | 1 | 0 | 1 | 3 | 0 | 3 | 0 |
| Março | 3 | 0 | 218 | 3 | 21 | 1 | 2 | 2 | 0 | 0 | 0 | 0 |
| Abril | 0 | 0 | 2 | 239 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Maio | 0 | 0 | 33 | 0 | 208 | 2 | 1 | 2 | 0 | 0 | 1 | 0 |
| Junho | 2 | 0 | 5 | 0 | 0 | 257 | 1 | 2 | 0 | 0 | 0 | 0 |
| Julho | 2 | 0 | 1 | 2 | 5 | 32 | 189 | 1 | 0 | 0 | 0 | 0 |
| Agosto | 0 | 0 | 13 | 27 | 2 | 1 | 0 | 205 | 0 | 0 | 0 | 0 |
| Setembro | 3 | 1 | 5 | 6 | 2 | 5 | 0 | 1 | 224 | 0 | 0 | 1 |
| Outubro | 6 | 1 | 3 | 3 | 2 | 16 | 1 | 0 | 2 | 214 | 0 | 0 |
| Novembro | 5 | 2 | 6 | 4 | 4 | 1 | 0 | 9 | 6 | 6 | 204 | 1 |
| Dezembro | 13 | 3 | 6 | 6 | 2 | 5 | 1 | 2 | 44 | 2 | 3 | 176 |

“Dezembro” apresentou uma taxa de acerto insatisfatória (66,92%) em parte por conta do critério de escolha. Qualquer outra classe que tivesse DE Mínima seria escolhida em detrimento de “Dezembro”, mesmo que esta também tivesse DE Mínima.

### 5.4.3 Extração em Zonas

#### Redes Neurais

A configuração da melhor rede com *back-propagation* encontrada pode ser vista na Tabela 5.7. A taxa de acerto encontrada para a base da extração em zonas foi de 85,94% com essa configuração. Os erros médio-quadrados para o conjunto de treinamento, validação e teste foram respectivamente 0,00563, 0,01278 e 0,01203.

O treinamento também foi bastante rápido, com convergência sendo atingida em apenas 167 épocas. O gráfico relacionando os erros médio-quadrados de treinamento e de validação com as épocas pode ser visto na Figura 5.4.

**Tabela 5.7 - Parâmetros da Rede Neural da Extração em Zonas.**

| | |
|---|---|
| Número de neurônios escondidos | 150 |
| Taxa de Aprendizagem | 0,1 |
| Quantidade máxima de falhas na validação | 5 |
| Algoritmo de Aprendizagem | Gradiente Descendente |
| Algoritmo de Treinamento | *Backpropagation* Resiliente |
| Função de Ativação | Sigmoide Logística |

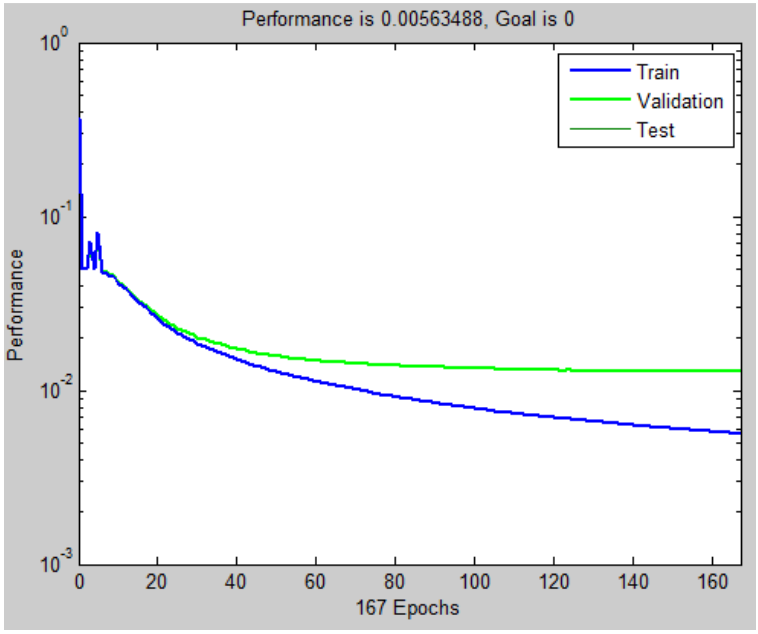



**Figura 5.3 – Desempenho da rede de Extração em Zonas.**

O sistema obteve 84,71% de acerto nesse experimento, escolhendo corretamente 92,89% ($\frac{84,71}{91,19}$) das vezes em que a classe correta apresentou DE mínima.

**Tabela 5.8 - Taxas de acerto e de DE Mínima de Extração em Zonas e RN.**

| Meses | Acerto (%) | DE Mínima (%) | Desempate correto (%) |
|---|---|---|---|
| Jan | 83,79 | 89,33 | 93,79 |
| Fev | 90,20 | 95,10 | 94,84 |
| Mar | 81,20 | 89,20 | 91,03 |
| Abr | 93,90 | 96,34 | 97,46 |
| Mai | 74,49 | 88,66 | 84,01 |
| Jun | 91,76 | 94,76 | 96,83 |
| Jul | 83,62 | 96,55 | 86,60 |
| Ago | 88,31 | 91,13 | 96,90 |
| Set | 87,50 | 91,53 | 95,59 |
| Out | 88,71 | 92,74 | 95,65 |
| Nov | 82,66 | 88,71 | 93,18 |
| Dez | 70,72 | 80,99 | 87,31 |
| Global | 84,71 | 91,19 | 92,89 |

**Tabela 5.9 - Matriz de Confusão de Extração em Zonas e RN.**

| | J | F | M | A | M | J | J | A | S | O | N | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Janeiro | 212 | 3 | 9 | 1 | 4 | 6 | 2 | 6 | 6 | 0 | 3 | 1 |
| Fevereiro | 3 | 221 | 4 | 2 | 3 | 2 | 0 | 0 | 2 | 0 | 8 | 0 |
| Março | 2 | 1 | 203 | 4 | 29 | 4 | 3 | 4 | 0 | 0 | 0 | 0 |
| Abril | 0 | 0 | 4 | 231 | 4 | 3 | 0 | 2 | 1 | 0 | 1 | 0 |
| Maio | 3 | 1 | 37 | 4 | 184 | 4 | 2 | 11 | 0 | 0 | 1 | 0 |
| Junho | 1 | 2 | 5 | 1 | 2 | 245 | 7 | 1 | 0 | 3 | 0 | 0 |
| Julho | 1 | 0 | 0 | 0 | 2 | 32 | 194 | 3 | 0 | 0 | 0 | 0 |
| Agosto | 0 | 0 | 8 | 14 | 4 | 1 | 2 | 219 | 0 | 0 | 0 | 0 |
| Setembro | 1 | 4 | 8 | 1 | 1 | 1 | 2 | 2 | 217 | 2 | 2 | 7 |
| Outubro | 2 | 1 | 4 | 3 | 4 | 6 | 1 | 3 | 1 | 220 | 3 | 0 |
| Novembro | 2 | 11 | 5 | 1 | 4 | 1 | 2 | 4 | 6 | 6 | 205 | 1 |
| Dezembro | 3 | 5 | 5 | 3 | 4 | 4 | 2 | 3 | 38 | 4 | 6 | 186 |

Mais uma vez “Abril” obteve a maior taxa de acerto. Muitos dos padrões de “Dezembro” foram classificados como “Setembro”, e de “Julho” como “Junho” (Tabela 5.9). “Maio” e “Março” também apresentaram muitos padrões classificados como da outra classe.

***Support Vector Machine***

SVM apresentaram ótimos resultados, como se vê na coluna de DE Mínima da Tabela 4.5, mas devido ao critério de escolha, a taxa de acerto mostrou-se maior nas primeiras classes que nas últimas, resultando numa taxa de acerto global menor (86,11%) que por Redes Neurais (86,31%).

**Tabela 5.10 - Taxas de acerto e de DE Mínima de Extração em Zonas e SVM.**

| Meses | Acerto (%) | DE Mínima (%) | Desempate correto (%) |
|---|---|---|---|
| Jan | 90,91 | 90,91 | 100 |
| Fev | 85,31 | 89,80 | 95,00 |
| Mar | 85,60 | 86,40 | 99,07 |
| Abr | 95,93 | 97,15 | 98,74 |
| Mai | 74,09 | 92,71 | 79,91 |
| Jun | 92,13 | 97,75 | 94,25 |
| Jul | 81,90 | 97,41 | 84,07 |
| Ago | 79,44 | 89,52 | 88,73 |
| Set | 88,31 | 93,95 | 93,99 |
| Out | 86,29 | 93,55 | 92,23 |
| Nov | 76,21 | 85,08 | 89,57 |
| Dez | 64,26 | 90,11 | 71,31 |
| Global | 83,34 | 92,02 | 92,48 |

**Tabela 5.11 - Matriz de Confusão de Extração em Zonas e SVM.**

| | J | F | M | A | M | J | J | A | S | O | N | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Janeiro | 230 | 1 | 9 | 1 | 2 | 7 | 0 | 0 | 1 | 1 | 1 | 0 |
| Fevereiro | 15 | 209 | 2 | 4 | 2 | 0 | 0 | 0 | 9 | 2 | 2 | 0 |
| Março | 3 | 0 | 214 | 8 | 21 | 2 | 1 | 1 | 0 | 0 | 0 | 0 |
| Abril | 0 | 0 | 3 | 236 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Maio | 2 | 0 | 49 | 7 | 183 | 3 | 1 | 2 | 0 | 0 | 0 | 0 |
| Junho | 3 | 0 | 10 | 3 | 3 | 246 | 1 | 0 | 0 | 1 | 0 | 0 |
| Julho | 1 | 0 | 3 | 4 | 5 | 28 | 190 | 1 | 0 | 0 | 0 | 0 |
| Agosto | 3 | 0 | 13 | 27 | 7 | 1 | 0 | 197 | 0 | 0 | 0 | 0 |
| Setembro | 3 | 1 | 8 | 5 | 2 | 2 | 1 | 1 | 219 | 1 | 1 | 4 |
| Outubro | 5 | 0 | 6 | 5 | 3 | 8 | 2 | 2 | 1 | 214 | 1 | 1 |
| Novembro | 3 | 2 | 7 | 2 | 6 | 5 | 0 | 5 | 12 | 16 | 189 | 1 |
| Dezembro | 6 | 4 | 4 | 6 | 1 | 6 | 0 | 1 | 62 | 2 | 2 | 169 |

“Dezembro” apresentou uma taxa de acerto insatisfatória (66,92%) em parte por conta do critério de escolha. Qualquer outra classe que tivesse DE Mínima seria escolhida em detrimento de “Dezembro”, mesmo que esta também tivesse DE Mínima.

## 5.5 Sumário

A Tabela 5.12 mostra as taxas de acerto para cada mês e a porcentagem de letras bem segmentadas. Observa-se que os classificadores de redes neurais apresentam um melhor resultado, devido ao critério utilizado para a escolha ser mais eficiente do que para SVM.

"Abril" apresentou melhor resultado, devido a sua singularidade em relação aos demais e à boa segmentação. Por outro lado, "Dezembro" apresentou a pior, por ter bastante semelhança com outros meses, como "Setembro" e "Novembro" e pelo resultado pior da segmentação. No entanto, "Junho" e "Julho" que possuem apenas uma letra em comum, apresentou resultados melhores. Isso mostra a importância de uma boa segmentação em sistemas de reconhecimento local.

**Tabela 5.12 - Taxas de Acerto de cada experimento.**

| Meses do Ano | Letras bem segmentadas (%) | Edge Maps | | Em Zonas | |
|---|---|---|---|---|---|
| | | RN (%) | SVM (%) | RN (%) | SVM (%) |
| Janeiro | 89,11 | 85,77 | 90,12 | 83,79 | 90,91 |
| Fevereiro | 90,16 | 91,43 | 88,57 | 90,20 | 85,31 |
| Março | 83,76 | 78,80 | 87,20 | 81,20 | 85,60 |
| Abril | 90,82 | 95,93 | 97,15 | 93,90 | 95,93 |
| Maio | 83,20 | 76,11 | 84,21 | 74,49 | 74,09 |
| Junho | 87,50 | 91,76 | 96,25 | 91,76 | 92,13 |
| Julho | 87,07 | 85,78 | 81,47 | 83,62 | 81,90 |
| Agosto | 83,67 | 87,90 | 82,66 | 88,31 | 79,44 |
| Setembro | 86,50 | 90,32 | 90,32 | 87,50 | 88,31 |
| Outubro | 83,24 | 89,11 | 86,29 | 88,71 | 86,29 |
| Novembro | 83,57 | 89,11 | 82,26 | 82,66 | 76,21 |
| Dezembro | 83,32 | 74,14 | 66,92 | 70,72 | 64,26 |
| Global | 86,03 | 86,31 | 86,11 | 84,71 | 83,34 |

Trabalhos semelhantes feitos por Kapp [4,17] tiveram uma taxa de acerto global de 81,75% utilizando redes neurais modulares, mas o sistema não estava restrito quanto à forma (cerca de 70% no estilo cursivo puro). Utilizando a mesma base de Kapp, Oliveira [39] obteve melhores resultados (90,4%) utilizando uma abordagem híbrida de HMM, NN-P e NN-D. Ambos os trabalhos utilizaram reconhecimento global.

# 6. CONCLUSÕES

As restrições do domínio visam criar sistemas mais especialistas, e que, portanto, tenham um desempenho melhor, possibilitando um desempenho global melhor, caso o especialista decida modular seu problema. Devem ser bem entendidas e planejadas pelo especialista, devido ao impacto delas no desenvolvimento do sistema.

Este trabalho estudou os diversos tipos de restrições que os sistemas de reconhecimento podem ter e o impacto deles em certas escolhas. No estudo de caso, algumas dessas decisões foram bem claras, como do estilo da forma para a segmentação, e do domínio ser fechado, para criar um sistema dedicado ao problema

Quanto ao sistema desenvolvido, teve dificuldade de diferenciar alguns meses que apresentam sub-strings em comum, principalmente “Março” e “Maio” e “Setembro” e “Dezembro”. No entanto, conseguiu bons resultados, mesmo utilizando técnicas simples de segmentação, extração e classificação. Os experimentos mostraram que uma segmentação eficaz melhora o desempenho e consegue lidar com o problema das sub-strings, como “Junho” e “Julho”.

O algoritmo Combinacional da Distância de Edição também se mostrou eficaz para o reconhecimento da palavra e para encontrar a melhor combinação entre as letras retornadas pelo classificador. No entanto, em muitos casos, houve empate entre mais de uma classe, e o sistema teve boa eficiência para escolher qual a melhor classe para classificadores de redes neurais, mas não foram tão eficientes para SVM.

Como trabalho futuro, estudar técnicas mais eficientes para segmentar palavras em letra de fôrma, que provaram ter forte impacto no resultado final. Testar combinações de técnicas de extração de características e de classificadores, e estudar uma forma mais eficaz para escolher uma classe em caso de empate em SVM, sendo uma das alternativas estudar combinar reconhecimentos local e global.

# REFERÊNCIAS

[1] – SIPSER, M. *Introduction to the Theory of Computation*. Course Technology, 1996.

[2] – SMITH, N. *Chomsky Ideas and ideals*. 2 ed., p. 48, 2004.

[3] – KOERICH, A. L.; SABOURIN, R.; SUEN, C. Y. *Large vocabulary off-line handwriting recognition:* A survey. Pattern Analysis and Applications, v. 6, p. 97-121, 2003.

[4] – KAPP, M.N.; FREITAS, C.O.A.; SABOURIN, R. *Handwritten Brazilian Month Recognition:* an Analysis of two NN Architectures and a Rejection Mechanism. 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR'04), p. 209-214, 2004.

[5] – CHOMSKY, N. *Syntactic structures.* Berlim: Walter de Gruyter & Co. KG, 2 ed., p. 13, 2002.

[6] – SEIDENBERG, M. S.; MCCLELLAND, J. L. *A Distributed, Developmental Model of Word Recognition and Naming*. Psychological Review, v. 96, p. 523-568, 1989.

[7] – FROST, R. *Toward a Strong Phonological Theory of Visual Word Recognition*: True Issues and False Trails. Psychological Bulletin, v. 123, n. 1, p. 71-99, 1998.

[8] – GRISHMAN, R.; KITTREDGE, R. *Analyzing language in restricted domains*: sublanguage description and processing. New Jersey: Lawrence Erlbaum Associates, Inc. p. 2-5, 1986.

[9] – BITTENCOURT, B. *Breve história da Inteligência Artificial.* Florianópolis, Brasil, 2004.

[10] – COX, S.; LINCOLN, M.; TRYGGVASON, J.; NAKISA, M.; WELLS, M.; TUTT, M.; ABBOTT, S. *TESSA, a system to aid communication with deaf people*. Proceedings of the fifth international ACM conference on Assistive technologies, p. 205–212, 2002.

[11] – LENDARIS, G. G.; TODD, D. N. *Use of a Structured Problem Domain to explore Development of Modularized Neural Networks*. Proceedings of the IJCNN'92, Baltimore, v. 3, p. 869-874, IEEE, jun. 1992.

[12] – CONTRERAS, H. *Closed Domains*, Probus 1, p. 163-180, 1989.

[13] – BAZZI, I.; SCHWARTZ, R.; MAKHOUL, J. *An Omnifont Open-Vocabulary OCR System for English and Arabic*. IEEE Trans Pattern Analysis and Machine Intelligence, jun. 1999.

[14] – MÖBIUS, B. *Rare Events and Closed Domains*: Two Delicate Concepts in speech Synthesis. International Journal of Speech Technology, p. 57-71, 2004.

[15] – TAPPERT, C. C.; SUEN, C. Y.; WAKAHARA, T. *The State of the Art in On-line Handwriting Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 12, n. 8, p. 787-808, ago. 1990.

[16] – NATHAN, K. S.; BEIGI, H. S. M.; SUBRAHMONIA, J.; CLARY, G. J.; MARUYAMA, H. *Real-time On-line Unconstrained Handwriting Recognition using Statistical Methods*. Proc. ICASSP'95, p. 2619-2622, jun. 1995.

[17] – KAPP, M. N. *Reconhecimento de Palavras Manuscritas Utilizando Redes Neurais artificiais*. Curitiba, Brasil, 116 p., mar. 2004.

[18] – LALLICAN, P.M.; VIARD-GAUDIN, C.; KNERR, S. *From Off-line to On-line Handwriting Recognition*. Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, p. 303-312, 2000.

[19] – NetChange, Consultores em Sistemas de Informação S.A. Disponível em: <http://www.netchange.pt/Products/Hardware/StepOver.aspx>. Acesso em 14 de mar. 2010.

[20] – NAMBOODIRI, A. M.; JAIN, A. K. *Online Handwritten Script Recognition*. IEEE Transactions on pattern analysis and machine intelligence, v. 26, n. 1, jan. 2004.

[21] – ARAÚJO, R. S. A.; CAVALCANTI, G. D. C.; FILHO, E. C. B. C. *On-line Verification for Signatures of Different Sizes*. Tenth International Workshop on Frontiers in Handwriting Recognition, p. 539-544, 2006.

[22] – CASEY, R. G.; LECOLINET, E. *A survey of methods and strategies in character segmentation*. IEEE Transactions on PAMI, jun. 1996.

[23] – MORI, S.; SUEN, C. Y.; YAMAMOTO, K. *Historical review of OCR research and development*. Proceedings of the IEEE, v. 80, p. 1029-1058, jul. 1992.

[24] – ADAMEK, T.; O' CONNOR, N. E. *Word matching using single closed contours for indexing handwritten historical documents*. International Journal on Document Analysis and Recognition, v. 9, n. 2-4, p. 153-165, abr. 2007.

[25] – NIBLACK, W. *An Introduction to Digital Image Processing*. Birkeroed, Dinamarca: Strandberg Publishing Company, 215 p., 1985.

[26] – SAUVOLA, J.; SEPPÄNEN, T.; HAAPAKOSKI, S.; PIETIKÄINEN, M. *Adaptive Document Binarization*. Proceedings International Conference on Document Analysis and Recognition, v. 1, p. 147-152., 1997.

[27] – OTSU, N. *A Threshold Selection Method from Gray-Level Histograms*. IEEE Transactions on Systems, Man, and Cybernetics, p. 62-66, jan. 1979.

[28] GATOS, B.; PRATIKAKIS, I.; KESIDIS, A.L.; PERANTONIS, S.J. *Efficient Off-Line Cursive Handwriting Word Recognition*. Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition, out. 2006.

[29] – BUNKE, H.; WANG, P. S. *Handbook of character recognition and document image analysis*. World Scientific Publishing Co. Pte. Ltd., p. 305-329, 1997.

[30] – CRUZ, R. M. O.; CAVALCANTI, G. D. C.; REN, T. I. *Análise de Técnicas de Extração de Características para o Reconhecimento de Dígitos Manuscritos*. Workshops of Sibgrapi 2009 - Undergraduate Works, Rio de Janeiro, Rio de Janeiro, Brasil, out. 2009.

[31] – ZHANG, T. Y.; SUEN, C. Y. *A fast parallel algorithm for thinning digital patterns*. Communications of the ACM, v. 27, n. 3, p. 236-239, 1982.

[32] – GONZALEZ, R. C.; WOODS, R. E. Woods. *Digital Image Processing*. New Jersey: Pearson Education, Inc. 3 ed, p. 709. 2008.

[33] – DEHGHAN, M.; FAEZ, K.; AHMADI, M. *A Hybrid Handwritten Word Recognition using Self-Organizing Feature Map, Discrete HMM, and Evolutionary Programming*. IEEE-INNS-ENNS International Joint Conference on Neural Networks, v. 5, p. 5515, jul. 2000.

[34] – LAVRENKO, V.; RATH, T. M.; MANMATHA, R. *Holistic Word Recognition for Handwritten Historical Documents*. Proceedings of the First International Workshop on Document Image Analysis for Libraries, p. 278, 2004.

[35] – GUSFIELD, D. *Algorithms on Strings, Trees, and Sequences*. The press Sundicate of the University of Cambridge, p. 215-245, 1997.

[36] – CAMASTRA, F.; SPINETTI, M.; VINCIARELLI, A. *Off-line Cursive Character Challenge*: a new Benchmark for Machine Learning and Pattern Recognition Algorithms. Proceedings of the 18th International Conference on Pattern Recognition, v. 2, p. 913-916, 2006.

[37] – MILGRAM, J.; CHERIET, M.; SABOURIN, R. *"One Against One" or "One Against All"*: Which One is Better for Handwriting Recognition with SVMs?. Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 2006.

[38] – CANU, S. ; GRANDVALET, Y. ; GUIGUE, V. ; RAKOTOMAMONJY, A. *SVM and Kernel Methods Matlab Toolbox*. Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.

[39] – JÚNIOR, J. J. O.; CARVALHO, J. M.; FREITAS, C. O. A.; SABOURIN, R. *Evaluating NN and HMM Classifiers for Handwritten Word Recognition*. Proceedings of SIBGRAPI'2002, p. 210-217, 2002.

# APÊNDICE A – Protótipo desenvolvido

Funcionamento do protótipo do sistema desenvolvido para reconhecimento dos meses. É possível classificar apenas uma imagem ou mostrar o resultado de toda a base (Figura 1).

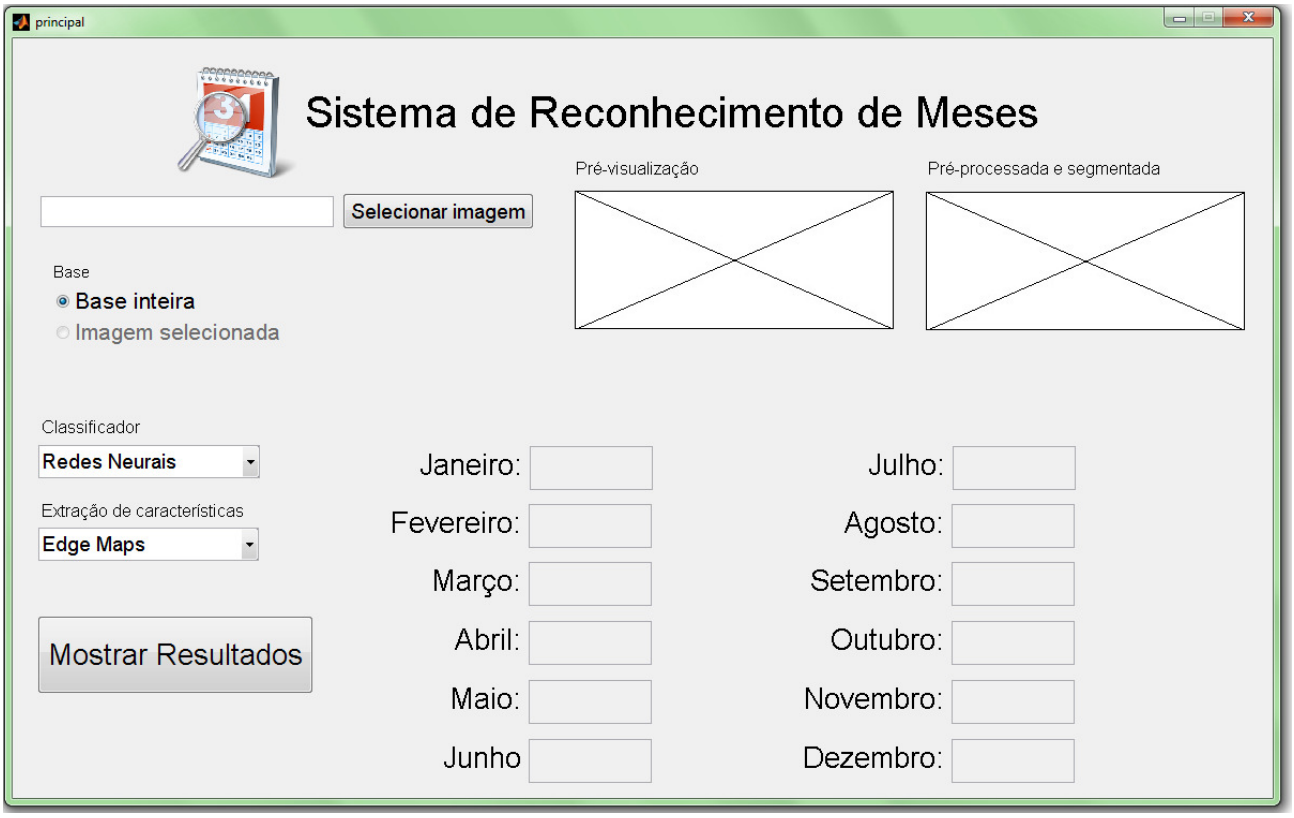



**Figura 1 – Tela inicial do protótipo.**

Selecionando a imagem desejada, é possível ver o resultado do pré-processamento em “Pré-processada e segmentada”, ao lado da imagem original em “Pré-visualização” (Figura 2).

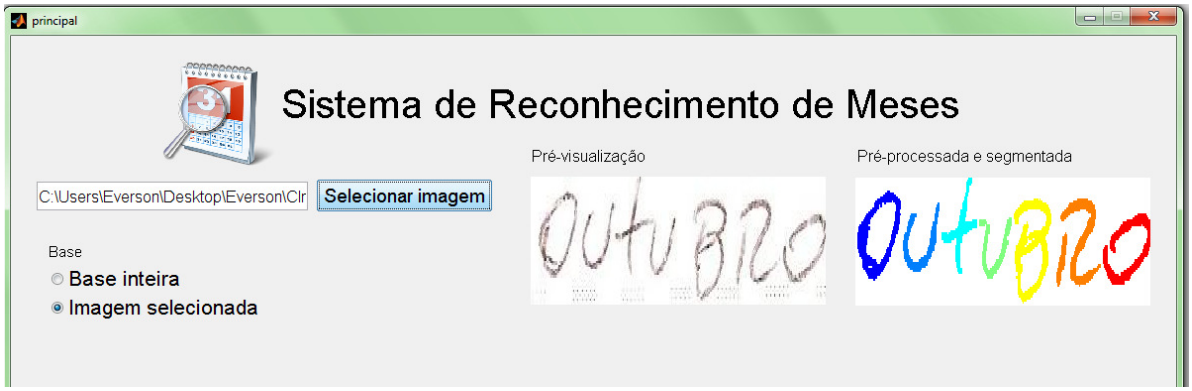



**Figura 2 – Resultado de pré-processamento da imagem selecionada.**

É possível escolher o classificador e o tipo de extração de características nas caixas ao lado, e após clicar em “Mostrar Resultados” para observar qual a melhor combinação dos resultados encontrados das letras para a classe escolhida, que se encontra em verde (Figura 3).

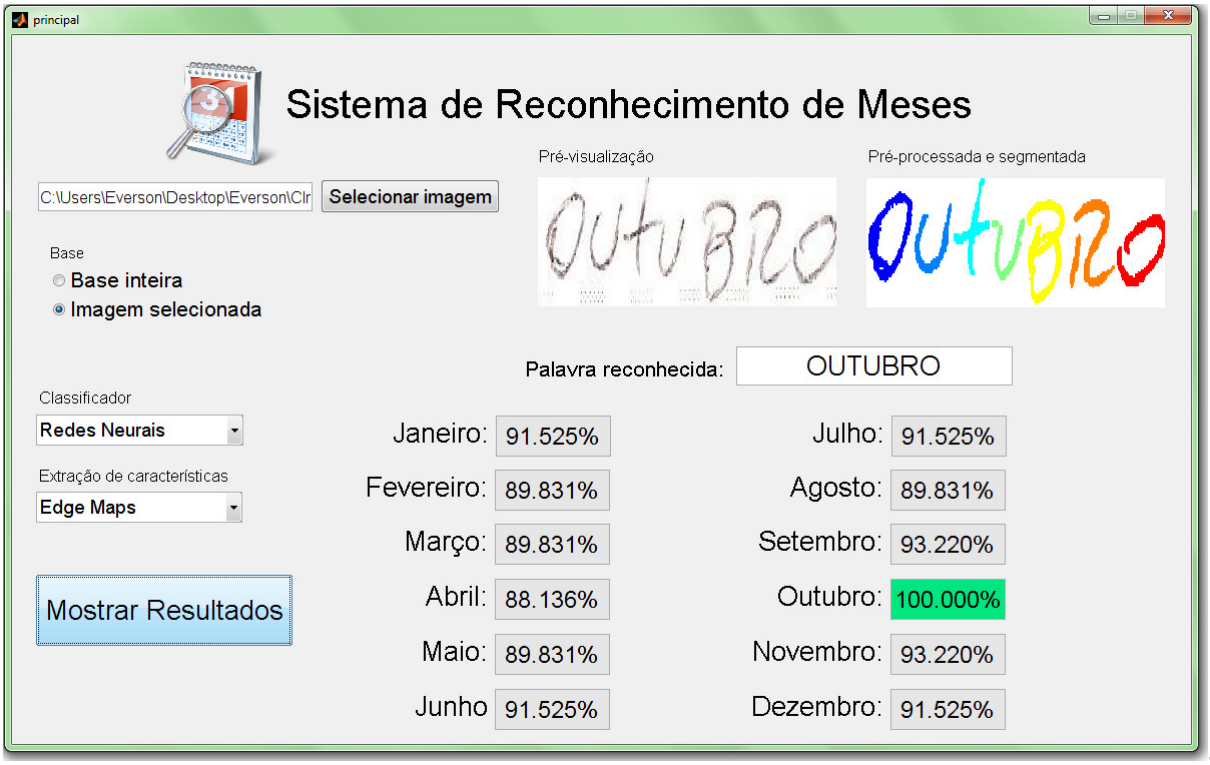



**Figura 3 – Resultado da classificação da palavra selecionada.**

Selecionando o campo “Base inteira”, as taxas de acerto de cada mês ficam visíveis nos respectivos campos. Basta selecionar o tipo de classificador e de extração de características, clicar em “Mostrar Resultados” e o resultado de toda a base é apresentado (Figura 4).

Clicando em algum dos botões “Ver erros”, é possível visualizar os padrões que foram classificados erroneamente da base de dados para aquele mês. Na Figura 5, por exemplo, é possível observar alguns dos padrões mal classificados do mês de “*Setembro*”, para o classificador de redes neurais e para a extração de características de *Edge Maps*.

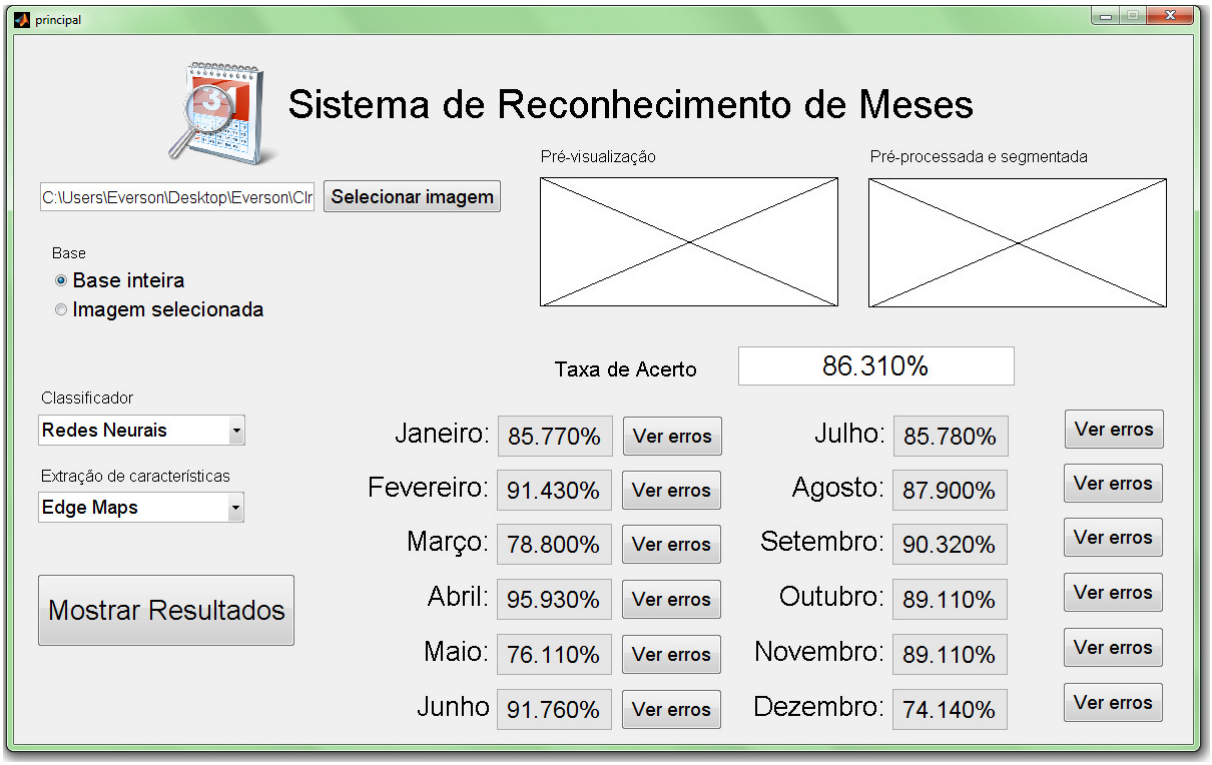



**Figura 4 – Taxas de acerto de toda a base.**

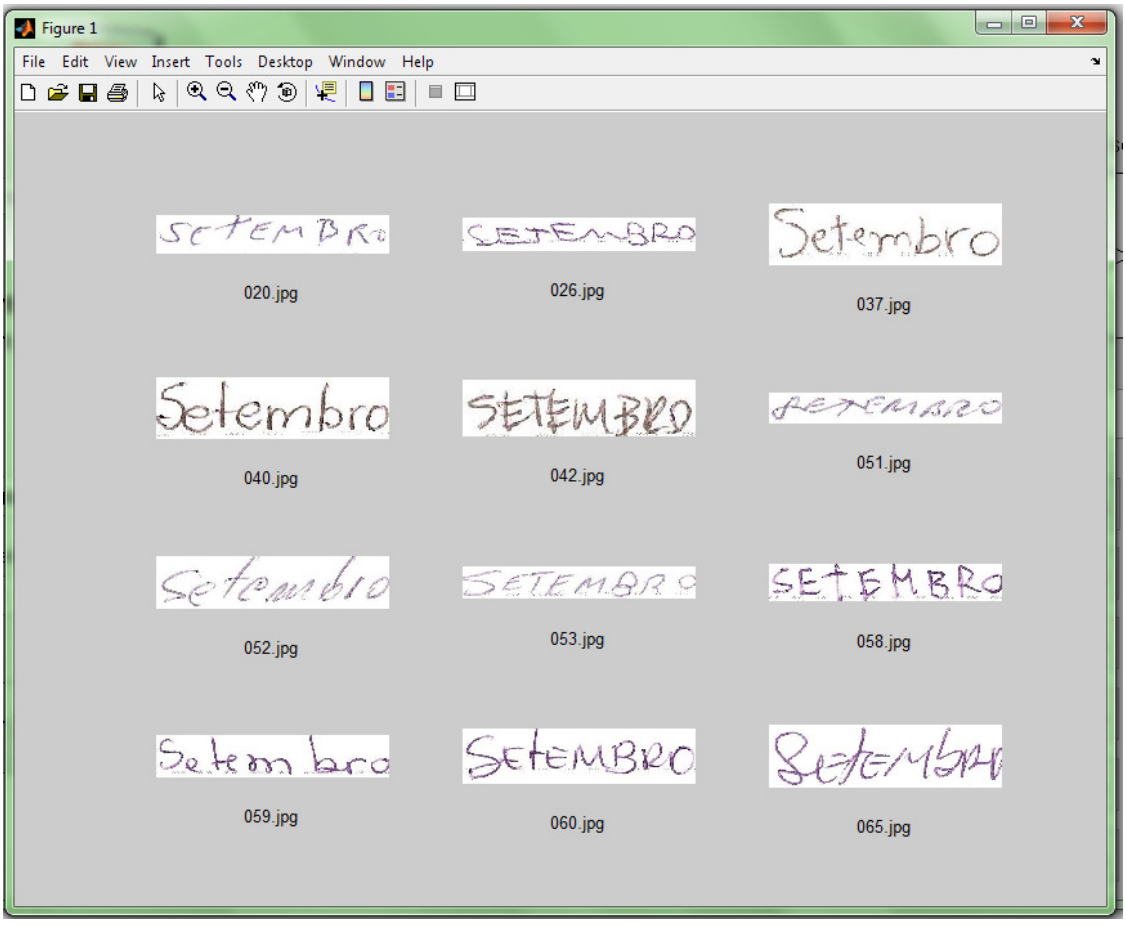



**Figura 5 – Palavras classificados erroneamente.**