Um Sistema de Reconhecimento Ótico de Caracteres Usando Características Topológicas e Algoritmo de Aprendizado por Máquina.

Marcos Alberto Lopes
Ogê Marques Filho

Centro Federal de Educação Tecnológica do Paraná - CEFET-PR
Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial - CPGEI
Av. Sete de setembro 3.165
80230 Curitiba - PR
cefetpr@brfapesp.bitnet

Abstract. This paper describes an Optical Character Recognition system that employs geometrical characteristics and a machine learning algorithm (CLS). Our method is suitable to text written in English or Portuguese. The main parts of the system are discussed and the results are used to evaluate the adopted techniques.

1 Introdução

Este trabalho apresenta uma técnica de extração de características, treinamento e classificação, para um sistema OCR aplicável a textos impressos em língua inglesa e portuguesa. As características são dos tipos geométrica e topológica [Suen (1980)], extraídas do contorno e dos pontos extremos dos traços que formam o caractere, após seu afinamento.

As características são determinísticas e permitem o uso de uma técnica de IA (o algoritmo CLS) para o aprendizado dos padrões de letras. As particularidades do idioma português, como os acentos, são tratadas a fim de permitir o reconhecimento de textos neste idioma.

O objetivo deste algoritmo é procurar diminuir as variações entre amostras diferentes de uma mesma classe, sem contudo sacrificar as diferenças entre classes, de modo a encurtar o processo de aprendizagem de novos padrões de letras e diminuir as taxas de erros do sistema.

2 Afinamento

O sistema prevê o uso de uma etapa de afinamento que reduz a espessura de cada traço do caractere a apenas um pixel de espessura. O algoritmo usado é o descrito em Zhang (1984), que efetua remoções sucessivas de "camadas" de pixels nas bordas do caractere, baseado na configuração de vizinhança de cada pixel.

A escolha deste algoritmo baseou-se no esqueleto de muito boa qualidade gerado, sendo que outros algoritmos testados tendiam a gerar "ramos" indesejados em virtude de pequenos ruídos nas bordas do caractere. Entretanto, para tornar o esqueleto ainda melhor, acrescentou-se uma etapa de remoção de pixels que formam "escadas" e de pixels supérfluos em pontos de cruzamentos de traços cuja remoção não cause perda de conexão do esqueleto.

3 Extração de características

As características dividem-se em dois grupos: as extraídas do caractere antes e as extraídas depois da etapa de afinamento.

3.1 Características pré-afinamento

As características obtidas antes do processo de afinamento visam distinguir classes muito parecidas, mas com alturas diferentes, como "O" maiúsculo e "o" minúsculo, ou de mesma altura e posicões das linhas de base diferentes como "P" maiúsculo e "p" minúsculo. Estas características referem-se a diferenças da posição relativa das linhas de base e de topo entre caracteres próximos e pertencentes à mesma linha de texto.

Segundo Kahan (1987), para textos impressos nota-se claramente que o topo dos caracteres apresenta duas posições predominantes, o mesmo acontecendo com suas bases, como pode ser visto na figura 1. No escopo deste trabalho, adota-se a seguinte nomenclatura:

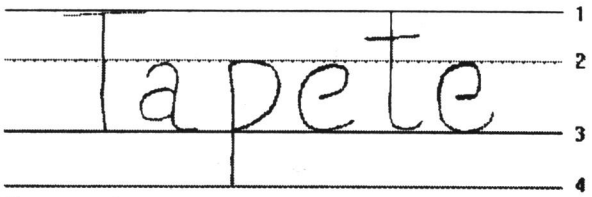

Figura 1: Comportamento típico dos topos e bases dos caracteres.

- Os caracteres com topo mais pronunciado são chamados **altos**. Ex: todas as letras maiúsculas e as minúsculas com altura comparável, como "b, "d", "f", "h", "i", "j", "k", "l" e"t";
- Os caracteres que não são altos são chamados de **baixos**;
- Os caracteres com base pronunciada são chamados **profundos**. Ex: "g", "j", "p", "q", "y";

minúsculas com altura comparável, como "b, "d", "f", "h", "i", "j", "k", "l" e"t";

- Os caracteres que não são altos são chamados de **baixos**;

- Os caracteres com base pronunciada são chamados **profundos**. Ex: "g", "j", "p", "q", "y";

- Os caracteres que não são profundos são chamados de **rasos**.

Para cada linha de texto a ser tratada, a detecção de caracteres profundos prevê a localização inicial da linha de base correspondente aos caracteres rasos (linha 3 da figura 1). Isto feito, será considerado profundo todo o caractere cuja base tenha um determinado deslocamento para baixo em relação a esta linha. Um processo semelhante é utilizado para a detecção de caracteres altos, havendo inicialmente a localização da linha 2 e verificando-se quais caracteres são mais altos que ela.

3.2 Características pós-afinamento

Após o caractere ter sido afinado, as características são obtidas percorrendo-se o esqueleto resultante.

O esqueleto do caractere passa a ser representado em forma de um grafo, onde os nodos representam pontos de cruzamento de traços ou de término dos mesmos e os arcos não-direcionados representam os traços do esqueleto, ou segmentos, que unem estes nodos (figura 2).

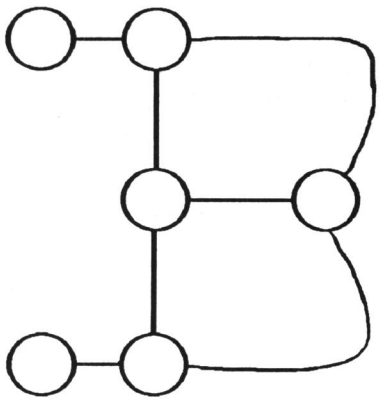

Figura 2: Grafo que representa o caractere.

3.2.1 Detecção de *loops*

Após a obtenção do grafo, o mesmo é pesquisado quanto à ocorrência de *loops*, onde *loop* representa um caminho que partindo de um nodo permite que se chegue ao mesmo através de um ou mais arcos passando ou não por outros nodos. No caso do caractere possuir um *loop* que envolva completamente outros *loops*, apenas o mais abrangente é considerado. Como exemplo, observando-se a figura 2 nota-se a presença de três *loops*: um acima do traço horizontal do meio, um abaixo e outro que engloba estes dois, que será o escolhido.

A detecção de *loops* visa diminuir as variações do número de segmentos que o caractere sofre, em função de serifas e ruído, uma vez que o processo de extração de características interrompe os segmentos nos pontos de cruzamento. Após o *loop* ser detectado ocorre a junção destes segmentos, tornando caracteres com serifas bem mais parecidos com suas versões *sans-serif*. Como exemplo, o "D" romano maiúsculo será particionado inicialmente em quatro segmentos enquanto o mesmo caractere no tipo *sans-serif* possuirá apenas um segmento. Após detectado o *loop* e reemendados os segmentos do tipo romano, este possuirá um total de três, sendo um deles idêntico ao *sans-serif*. As duas serifas terão grande chance de serem desprezadas durante a classificação, como será explanado mais adiante.

3.2.2 Marcação de pontos de referência

Cada segmento passa por uma etapa de marcação de pontos de referência baseada nas posições de máximos e mínimos verticais e horizontais, num processo semelhante ao descrito em Ward (1988), proposto para caracteres manuscritos. O processo começa em qualquer dos pontos extremos do segmento que é imediatamente marcado e procede a uma verificação do sentido tomado horizontalmente (esquerda ou direita) e verticalmente (acima ou abaixo). No momento que houver alteração de um dos sentidos, o ponto em que isto ocorre é marcado como ponto de referência (figura 3). A partir deste ponto o processo recomeça e repete-se até o último pixel do segmento, que também é marcado como ponto de referência.

Para exemplificar este processo, tomando-se a figura 3a e começando-se pelo ponto P1 verifica-se que o sentido vertical tomado é ascendente e o horizontal indefinido. Após percorrido o lado L1 e iniciado o L2, verica-se o sentido horizontal para a direita. Quando os primeiros pixels de L3 tiverem sido percorridos, o sentido ascendente anteriormente verificado será violado e um ponto de referência caracterizado, que, neste caso, representa a mudança de sentido ascendente para descendente. Para a localização deste ponto, os pixels desde o último ponto de referência marcado até o último pixel pesquisado são verificados para a determinação daquele com a maior coordenada y (o mais alto). Em seguida, define-se uma região Δy centrada neste ponto (figura 3b) e define-se o primeiro e o último pixels da seqüência que pertencem a esta

região. O pixel escolhido como ponto de referência será aquele com coordenada x intermediária entre a do primeiro e a do último pixels que pertencem à região Δ y. Processos semelhantes são usados quando a direção e/ou o sentido violado for(em) diferente(s).

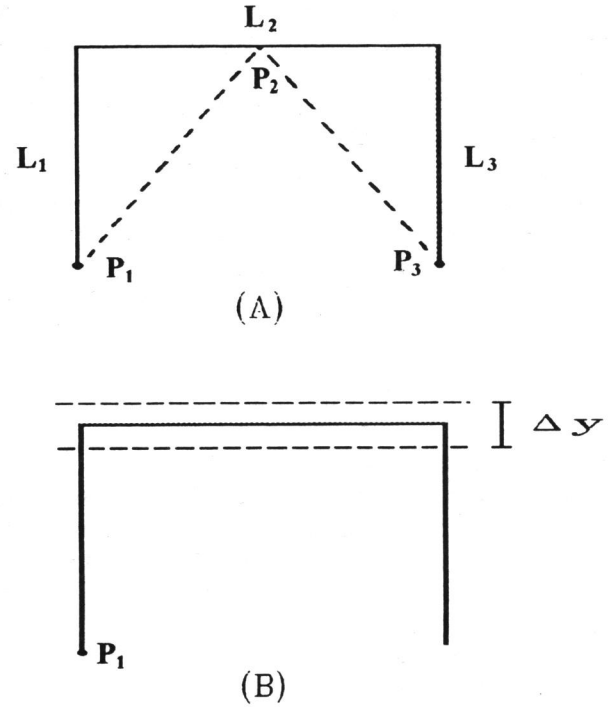

Figura 3: Marcação de pontos de referência:
(A): Posição dos pontos marcados;
(B): Região Δy usada para estabelecer a posição de um ponto em trecho horizontal.

4 Codificação

A etapa de codificação é encarregada de representar cada uma das características detectadas nas etapas anteriores em uma forma adequada às etapas subseqüentes. A forma de representação escolhida é a de um vetor de caracteres (*string*), de tamanho variável para cada segmento, sendo que um caractere possuirá necessariamente um *string* para cada segmento que dele faça parte e mais um para cada característica extra que represente o caractere como um todo (alto, profundo, etc).

Para a representação de um segmento, inicialmente uma grade de quatro linhas por quatro colunas é traçada sobre o caractere e os pontos extremos são nela localizados (figura 4). Para fins de simplificação, cada coordenada na grade é representada por um único caractere hexadecimal. Os dois primeiros caracteres do *string* de um segmento serão as coordenadas do ponto inicial (por onde é iniciada a

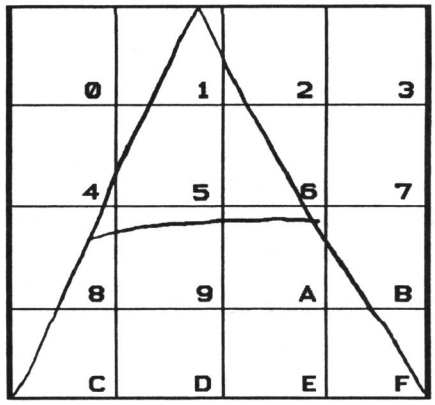

Figura 4: Grade de localização de pontos extremos.

marcação dos pontos de referência) e do ponto final, respectivamente.

No próximo passo, a seqüência dos pontos de referência marcados é codificada usando códigos de Freeman [Freeman (1961)] de oito direções (figura 5). Cada código obtido é transformado em representação ASCII e concatenado ao *string*. Os segmentos que não possuirem nenhum ponto de referência entre o primeiro e o último pixels receberão o código 8 representando um código de Freeman fictício.

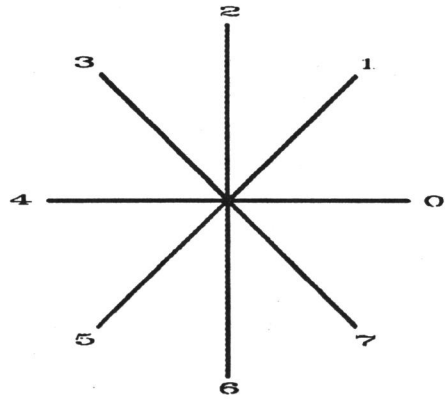

Figura 5: Códigos de Freeman

Após o *string* ter sido completado, faz-se necessário verificar se a escolha do ponto inicial de codificação foi correta. Será considerado ponto inicial aquele com a coordenada hexadecimal de valor menor. Se ambos os pontos extremos tiverem a mesma coordenada (*loop*), será escolhido aquele que, se a codificação começasse por ele, o primeiro código de Freeman produzido seria menor que o primeiro código produzido pelo outro extremo. Caso a escolha tomada inicialmente tenha sido errada, o *string* deve ser invertido de modo a acertar os códigos.

As características descritas até o momento não permitem distingüir entre partes quadradas e partes

arredondadas de caracteres, o que pode gerar confusão entre determinadas classes como "D" e "O", "J" e "/", "(" e "[". Para resolver este problema, uma outra característica, a presença de um traço vertical maior que 3/4 da altura do caractere, é usada. Para os caracteres que a possuem ("D", "J" e "[" do exemplo acima), ela é representada pelo *string* "G00" que é adicionado aos já existentes no caractere, como se representasse mais um segmento (segmento fictício). Da mesma maneira, se o caractere tiver sido considerado alto, o string "G01" é acrescentado e se tiver sido considerado profundo, o string "G02" também o será.

5 Treinamento e classificação

Um processo de treinamento é utilizado no sistema a fim de extrair, a partir de um conjunto de textos que lhe seja fornecido, um número suficiente de amostras de cada classe de caracteres e delas obter informações que permitam a posterior classificação de outros textos do mesmo estilo de letra.

O treinamento utiliza o algoritmo CLS com capacidade de aprendizado dos padrões de letras apresentados e de generalização dos mesmos para a montagem de uma árvore de decisão binária. Durante o processo as características essenciais para a caracterização de cada classe são automaticamente reconhecidas e deste modo a classificação tem sua confiabilidade incrementada, uma vez que as variações que eventualmente ocorram nas características não-essenciais são desprezadas.

5.1 O algoritmo CLS

A tarefa do algoritmo CLS é a montagem de uma árvore de decisão binária que será usada na etapa de classificação. A árvore foi escolhida binária por uma questão de simplicidade, onde os dois ramos de cada nodo representam simplesmente a presença ou ausência de uma dada característica, representada pelo *string* descrito anteriormente.

O algoritmo CLS monta a árvore de decisão a partir do conjunto de amostras obtidas durante o treinamento e que devem estar disponíveis na memória do computador. Cada nodo da árvore pode representar uma característica a ser testada ou um código relativo a uma dada classe. Cada nodo representando uma característica possui dois nodos filhos correspondentes à presença ou ausência da referida característica. Os nodos que representam classes são nodos folha.

Considera-se o conjunto de amostras de treinamento, **T**, sobre o qual uma característica será escolhida de cada vez, de modo a constituir um nodo da árvore. Começa-se com uma árvore vazia e vai-se adicionando recursivamente novos nodos através de refinamentos sucessivos até que todo o conjunto de treinamento possa ser corretamente classificado pela árvore obtida.

Passo 0: Cria-se um nodo **N** vazio.

Passo 1: Se todas as amostras em **T** pertencerem a uma mesma classe, coloca-se em **N** o código desta classe e finaliza-se. Caso contrário, seleciona-se uma característica **C** em **T**, usando o critério descrito mais adiante e coloca-se-a em **N**.

Passo 2: Cria-se dois conjuntos de amostras, **TS** e **TN**, colocando-se no primeiro o subconjunto de amostras em **T** que possuam a característica **C**, e no segundo as amostras que não a possuam. Cria-se dois nodos, filhos de **N**: **NS** representando **TS** e **NN** representando **TN**.

Passo 3: Repete-se recursivamente os passos 2 e 3, primeiro para **NS** e **TS** e depois para **NN** e **TN**.

5.2 O cálculo de entropia

O critério para escolha da característica no passo 1 do algoritmo é o de se usar aquela que apresente a máxima redução de entropia do conjunto de amostras considerado, ou seja, a característica que melhor discrimine as classes envolvidas.

A entropia em **T** para uma dada característica **C** é calculada pela fórmula:

$$\sum_i \sum_j \left[\frac{N_{cj}}{\sum_j N_{cj}} * \left(- \frac{N_{cj}(i)}{N_{cj}} \right) * \log_2 \left(\frac{N_{cj}(i)}{N_{cj}} \right) \right]$$

Onde:

j assume dois valores: 0 representando as amostras que não possuam a característica **C** e 1 as que possuam;

i percorre todas as classes existentes;

N_{cj} número de amostras cuja característica **C** assuma o valor **j** (presente ou ausente);

$N_{cj}(i)$ número de amostras da classe **i** cuja característica **C** assuma o valor **j**.

No passo 1 do algoritmo, todas as características têm sua entropia calculada, e aquela que apresentar o menor valor, ou seja, o maior ganho de informação, é escolhida para nó da árvore. A otimização da entropia em cada nodo conduz teoricamente à árvore com o menor número de características examinadas [Pao (1989)].

O que se espera com uso do cálculo de entropia é que, sendo a escolha da característica de cada nodo da árvore feito de forma ótima, somente as mais relevantes sejam escolhidas. Espera-se que as características provenientes de ruídos ou serifas sejam desprezadas,

em favor daquelas que conduzam a uma melhor separação entre classes. Isto tornará a etapa de classificação mais imune a textos de baixa qualidade de impresão.

6 Acentos

Em um caractere, cada conjunto de traços que possua caminho de conexão entre todos os componentes é chamado no escopo deste trabalho de **ilha**. Como exemplo, o caractere "=" possuirá duas ilhas.

O tratamento de caracteres com mais de uma ilha, os acentuados, "i", "j" e "=", por exemplo, é diferente dos caracteres normais. Durante o treinamento, cada nova ilha cadastrada pelo sistema recebe um código numérico único de identificação. Para a classificação de caracteres com mais de uma ilha, o sistema mantém uma lista com os códigos das ilhas que os formam. Ex: para o "õ" a entrada correspondente na lista conterá o código do mesmo e os códigos do "~" e do "o". Durante a classificação, quando um caractere com mais de uma ilha for detectado, elas devem ser classificadas separadamente e os códigos das classes obtidos usados para uma consulta à tabela a fim se obter o código do caractere (acentuado) resultante.

7 Resultados e conclusões

Durante o processo de treinamento observou-se que um número muito grande de variações de cada classe foi armazenado. Isto deve-se em parte ao posicionamento dos pontos extremos do caractere na grade 4x4. Em tipos que apresentam serifas, muitas vezes - quando estas são muito pequenas em relação ao tamanho do caractere - estas são eliminadas pelo processo de afinamento. A presença ou ausência de serifas ou outros traços pequenos altera o posicionamento do caractere sobre a grade, mudando as coordenadas dos pontos extremos dos segmentos. Existem ainda aqueles pontos extremos que aparecem naturalmente no limiar entre um quadrículo e outro(s), apresentando coordenadas diferentes em amostras diferentes.

A tabela 1 mostra o número de amostras de cada classe e de erros cometidos sobre os textos utilizados para os testes do sistema, ficando a taxa geral de erros de classificação em torno de 3,59%. Estes testes foram efetuados sobre textos extraídos de revistas impressas em *off-set*, com um *scanner* manual regulado para uma resolução de 400 dpi.

Nota-se uma taxa de erros significativamente maior em caracteres acentuados, devido ao fato dos acentos agudos serem freqüentemente confundidos com o apóstrofo ou o "pingo do i". A principal causa disto são os esqueletos, gerados pelo algoritmo de afinamento para ambas as classes, que são muitas vezes parecidos. Este problema, no entanto, parece ser de fácil solução.

A taxa de erros aparentemente alta, quando comparada com sistemas de alto desempenho, deve-se também ao fato acima exposto de haver muitas variações, sendo que nem todas as possibilidades são armazenadas durante o processo de treinamento. Deste modo, na classificação sempre chegam amostras diferentes das que o sistema conhece, sendo que algumas não podem ser classificadas corretamente, mesmo que apenas as características mais significativas sejam utilizadas.

As características utilizadas não permitiram diferenciar todas as amostras de "l" (éle) e "1" (numeral), bem como "O" (letra maiúscula) e "O" (zero). Neste sistema, isto não pode ser feito, uma vez que não é empregado pós-processamento.

Os resultados obtidos mostram que o sistema, da forma como se encontra, necessita de melhorias para ter aplicação comercial, devido à elevada taxa de erros e uma certa lentidão de execução em máquinas de baixo desempenho (3,6 cars./seg. em AT de 8 MHz). Esta lentidão se deve principalmente às etapas de segmentação e afinamento implementadas, havendo a possibilidade de melhorias neste sentido.

Como descrito anteriormente, cada classe apresentou um elevado número de amostras diferentes, mesmo usando-se um único tipo de letra. Isto torna o processo de aprendizado excessivamente longo (da ordem de horas) a fim de se obter uma taxa de acerto razoável durante a classificação.

A generalização de conceitos esperada do algoritmo CLS não pôde ser observada em conseqüência do elevado número de variações obtido.

Mesmo que o sistema apresente deficiências, o trabalho pode ter continuidade com a busca de novas e mais eficazes características e a adição de uma etapa de pós-processamento, que faria a verificação contextual do texto de saída, onde cada letra ou palavra seria verificada em função de suas vizinhas. Uma técnica como esta é essencial inclusive para a diferenciação entre certas letras e números.

Carac -ter	Amos -tras	Erros	Carac -ter	Amos- tras	Erros
A	25	6	a	890	1
B	13	0	b	70	0
C	16	0	c	260	1
D	6	0	d	413	0
E	19	1	e	839	4
F	11	0	f	45	1
G	5	5	g	93	0
H	8	1	h	50	5
I	8	0	i	498	12
J	10	0	j	17	13
K	3	0	k	9	5
L	5	0	l	213	3
M	14	5	m	349	3
N	14	3	n	360	14
O	15	0	o	726	3
P	19	3	p	208	3
Q	4	1	q	69	0
R	9	1	r	467	9
S	12	0	s	645	10
T	7	3	t	350	29
U	7	0	u	270	1
V	7	2	v	101	1
W	3	2	w	4	1
X	4	2	x	19	0
Y	3	1	y	12	0
Z	4	4	z	23	4
0	14	0	ã	51	1
1	11	2	õ	18	0
2	13	3	é	43	24
3	10	4	â	4	0
4	5	4	à	5	5
5	10	2	ç	50	1
6	5	0	ê	17	1
7	9	0	ô	3	0
8	8	1	á	35	21
9	6	0	í	26	20
			ó	13	6
			ú	8	5

Tabela 1: Número de amostras utilizadas e de erros cometidos para cada classe.

Referências

H. Freeman, On the encoding of arbitrary geometric configurations, *IEEE Trans. Elec. Computers*, 10 (1961), 260-268.

S. Kahan, T. Pavlidis, H. S. Baird, On the recognition of printed characters of any font and size, *IEEE Transactios on Parttern Analysis and Machine Intelligence*, 9 (1987), 274-288.

Y-H. Pao, Adaptative pattern recognition and neural networks, *Addison Wesley Publishing Company, Inc*, (1989).

C. Y. Suen, M. Berthod, S. Mori, Automatic recognition of handprinted characters - the state of the art, *Proceedings of the IEEE*, 68 (1980), 469-487.

J. R. Ward, T. Kuklinski, A model for variability effects in handprinting with implications for the design of handwriting character recognition systems, *IEEE Transactions on Systems, Man, and Cybernetics*, 18 (1988), 438-451

T. Y. Zhang, C. Y. Suen, A fast parallel algorithm for thinning digital patterns, *Communications of the ACM*, 27 (1984), 236-239.