HMM para o Reconhecimento de Caracteres Manuscritos

Lin Yu Han
Universidade Federal do Paraná
Av. Coronel Francisco Heráclito dos Santos, 210, Curitiba - PR, 81531-970
lyhan@inf.ufpr.br

Abstract

Técnicas de reconhecimento de escritas online têm sido estudadas para diminuir a taxa de erro na questão de reconhecer os traços de palavras manuscritas. Baseado nisso, este trabalho apresenta a análise de dois estados da arte no reconhecimento de manuscritos, fazendo uma comparação entre eles.

1. Introdução

Reconhecimento de manuscritos ainda é um grande desafio a ser explorado. Pesquisadores estudam várias técnicas e algoritmos para aumentar a taxa de acerto no reconhecimento de manuscritos tanto nas abordagens *online* quanto *offiline*. O reconhecimento *online* é dito quando o processo é feito em tempo real, ou quando durante o reconhecimento, dados espaço-temporais são disponibilizados (e.g. coordenadas espaciais); o reconhecimento *offline* ocorre quando apenas imagens são disponíveis para o reconhecimento.

Grande parte dos *smartphones* possui o reconhecimento de escritas *online*, pois sabendo que estes dispositivos não possuem teclados físicos, apenas um teclado virtual sobre a tela é disponibilizado para a entrada de caracteres.

Neste trabalho, estuda-se o reconhecimento *online* com a entrada de dados realizada por dispositivos móveis, utilizando o procedimento do HMM (*Hidden Markov Model*) usado por Liwicki *et al* [5], proposto em Marti *et al* [7] para a realização do reconhecimento de caracteres.

A organização do trabalho segue abaixo. Na subseção 1.1 citam-se os trabalhos relacionados, o qual contém os estados da arte com relação ao reconhecimento de manuscritos *online*. Na seção 2 apresenta-se as técnicas que serão utilizadas do trabalho Liwicki *et al* [5] e também o método para geração da base. Na seção 3 apresenta-se os resultados experimentais e a seção 4 conclusões e considerações finais.

1.1. Trabalhos relacionados

Na literatura existe uma grande quantidade de trabalhos com relação ao reconhecimento *online* de manuscritos.

Dutta *et al.* 2012 [2] desenvolve um sistema focado em plataformas Android que reconhece caracteres não cursivos de maneira rápida e de leve processamento, sendo este último requisito bastante necessário já que os smartphones possuem uma capacidade de processamento menor comparada a um computador. Em um trabalho futuro, Dutta *et al.* 2014 [3] visando no requisito de que o reconhecedor online trabalha com a ordem com que se escreve a palavra cursiva, trabalha com o classificador DTW (*Dynamic Time Wrapping*) apresentando um reconhecimento online de ordem livre.

Gao *et al* [4] apresentam um novo *framework* para o reconhecimento de caracteres, usando a plataforma de computação em nuvem como serviço; diferente do trabalho citado anteriormente, Gao *et al* usa os *smartphones* apenas como entrada de dados, os dados espaço-temporais recolhidos são então enviados para o servidor via internet que realiza a classificação e reconhecimento e retorna para o dispositivo móvel caracter; o resultado deste método propõem uma abordagem de reconhecimento que suporta diferentes tipos de sistemas operacionais de *smartphones*.

O sistema desenvolvido por Prassad *et al* [10] sobre dispositivos móveis estuda duas abordagens de reconhecimento: PCA (*Principal Component Analysis*) e DTW, a análise detalhada dos resultados mostrou que o PCA é mais adequado para aplicações de tempo real, pois atingiu uma taxa de acerto mais alta em um tempo muito mais curto. Wang *et al* [16] por outro lado apresenta em seu trabalho o algoritmo DTW com taxas de acertos consideravelmente aceitáveis, desenvolvendo uma caneta como dispositivo para a entrada de dados.

Namboodiri *et al* [8] apresenta um estudo bastante amplo utilizando cerca de oito classificadores para o reconhecimento de seis *scripts* diferentes. Para cada *script*, foram usadas onze características, tendo o classificador que apresentou a menor taxa de erro o SVM (*Support Vector Machine*).

Em contraste do constante uso do HMM para o reconhecimento de manuscritos de outros trabalhos, Shivram *et al* [11] soluciona o problema utilizando o CRF (*Conditional Random Field*) combinando com um *framework* de segmentação e reconhecimento, apresentando resultados melhores em comparação ao

HMM.

Malik *et al* [6] foca o estudo em um sistema de reconhecimento de manuscrito de Urdu utilizando como classificador o *Tree Based Dictionary Searching*: para cada caracter/palavra do dicionário é assimilado uma estrutura d dados baseando no tipo do componente e o seu tamanho, caracteres que possuem a mesma estrutura tem o mesmo conjunto de componentes e os mesmos tamanhos. Isto mostra que os algoritmos considerados eficientes para o reconhecimento de manuscritos pode ser diferente quando aplicadas em outros idiomas.

Trabalhando com classificadores, Sridhar *et al* [13] propõem uma abordagem diferente juntando *Active Shape Models* com *Elastic Matching*, descrito como *Active-DTW*.

Alguns trabalhos envolvem também a área de Interação Humano-Computador, como é o caso de Aggarwal *et al* [1], que utiliza um sensor de profundidade para a captura de dados, utilizando DTW e SVM para a classificação e reconhecimento, e também Liwicki *et al* [5] que utiliza a captura de escritas em quadros brancos.

Conceitos e descrições sobre o reconhecimento *online* e *offline* podem ser encontrados no trabalho de Wakahara *et al* [15] e Plamondon *et al* [9].

2. Metodologia

Nesta seção serão apresentadas as metodologias utilizadas de acordo com o trabalho estudado com seu respectivo algoritmo.

2.1. Método HMM

Em um trabalho anterior, Marti *et al* [14] descreve o HMM para o reconhecimento offline. Através desse trabalho, Liwicki *et al* [2] continua a pesquisa incrementando um modelo de entrada de dados online. Então ele utiliza um modelo online para a entrada de dados e realiza o reconhecimento offline com o HMM. Liwicki *et al* [2] explica dois motivos do uso do reconhecedor offline em vez de um online. Primeiramente um reconhecedor desenvolvido por eles é offline e por segundo, a conversão de dados online para offline é direta Liwicki *et al*[2].

Neste trabalho, para este algoritmo, utilizaremos a metodologia usada por Liwicki *et al* [2], ou seja, capturando a entrada de modo online, transformando os dados online em offline para então realizar o processo de reconhecimento. Foi feito isso para simular de maneira mais próxima possível as etapas de Liwicki *et al* [2].

A visão geral do sistema é apresentada na Figura 1 O sistema consiste de módulos principais: a transformação, onde os dados online são convertidos em offline; pré-processamento offline, o qual passos de normalizações são realizados (correção da angulação/redimensionamento); a extração de características, onde a imagem normalizada é transformada em vetores de características; e o reconhecimento, onde a classificação via HMM gera os n melhores sequencias de caracters.

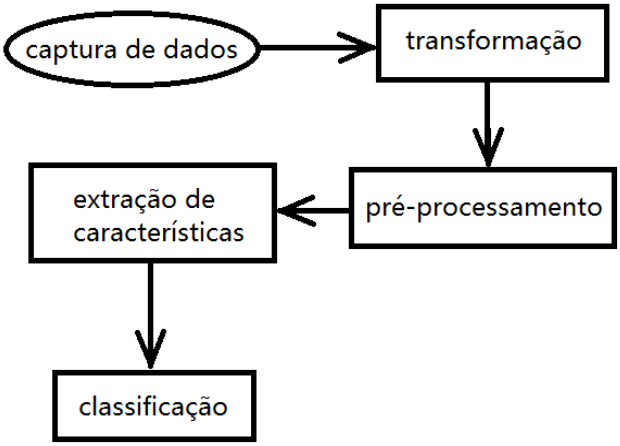

Figura 1: Visão geral do sistema.

Processos de segmentação e alguns processamentos de imagem (como binarização e remoção de ruídos) foram ignorados neste trabalho por dois motivos: a entrada de dados será de apenas caracteres cursivos individuais (caracteres singulares); a escrita é realizada com a cor preta tendo o plano de fundo branco; de modo empírico, não foram observados ruídos (pontos/riscos que fogem dos traços dos caracteres) nas imagens obtidas.

A extração de características é realizada pela análise da imagem obtida após o pré-processamento. São computadas nove características como descrito em [14]. A primeira representa a quantidade de pixels pretos presentes na janela; a segunda é o centro de gravidade da janela; a terceira é o momento de segunda ordem da janela (o quanto a estrutura toda se deforma); a quarta característica representa a posição mais alta da janela (pixel mais alto); a quinta representa a posição mais baixa da janela (pixel mais baixo); a sexta e sétima características representam respectivamente a orientação dos pixeis mais altos e mais baixos da janela; a oitava característica mostra o número de transições de pixel preto/branco na orientação vertical; a nona e última característica nos dá a quantidade de pixels pretos entre o pixel mais alto e o mais baixo.

As características 4 ao 9 são facilmente calculadas. As três primeiras características são calculadas pela seguintes fórmulas:

$$(1) \quad f_1(x) = \sum_{y=1}^{m} p(x,y)$$

$$(2) \quad f_2(x) = \frac{1}{m} \sum_{y=1}^{m} y \,.\, p(x,y)$$

$$(3) \quad f_3(x) = \frac{1}{m^2} \sum_{y=1}^{m} y^2 \,.\, p(x,y)$$

O sistema HMM é construído para cara caracter, como descrito em Marti[14]. O uso deste modelo permite compartilhar os dados de treinamento; cada instância de uma letra no conjunto de treinamento tem um impacto no treinamento e conduz para uma melhor estimativa de parâmetro.

No modelo proposto, para cada modelo de caracter foi estabelecido 14 estados. Este número, explica Marti *et al* [14], foi estabelecido empiricamente, pelo fato de que a janela utilizada para a extração de características é de apenas um pixel de largura. A inicialização dos modelos é feita pelo algoritmo de *Viterbi*. A probabilidade de transição e a probabilidade de distribuição das saídas durante o treinamento é calculado através do algoritmo *Baum-Welch*.

O sistema de reconhecimento foi implementado no ambiente *Visual Studio Comunity 2015*, utilizando a linguagem C# com o *Accord.NET Framework* [12], que é uma estrutura de bibliotecas para sistemas de aprendizado de máquina desenvolvido em C#.

2.2. Geração da Base

Através de um dispositivo móvel Android, desenvolveu-se um aplicativo que capturasse a escrita do usuário. O aplicativo é simples e de fácil manuseio apenas com objetivo de coleta de dados para a base de testes e

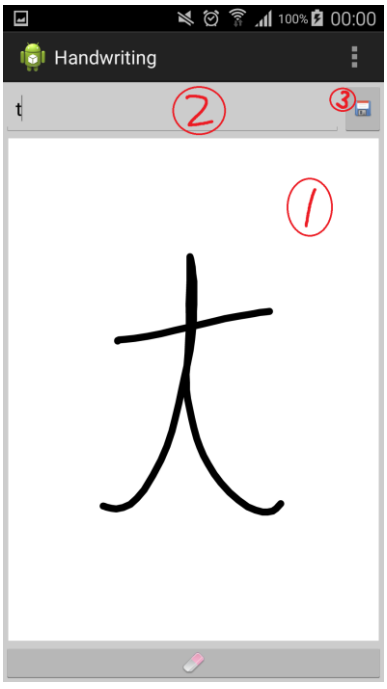

Figura 2: Aplicativo para geração das bases.

treinamento da pesquisa, como mostrado na Figura 2.

O usuário traça o caracter sobre o espaço em branco (1) e então informa no campo (2) a que classe (no caso qual letra) o traçado pertence. Após isso, clica-se no botão (3) para salvar.

Os dados salvos são as coordenadas dos pixels traçados (x_1, y_1; x_2, y_2; x_3, y_3; x_n, y_n) e sua respectiva classe (no caso 't', como mostra a Figura 2).

O aplicativo foi desenvolvido no ambiente Eclipse Java Mars com a ferramenta Android SDK em um ASUS TP500L 8GB RAM Intel Core i7-4510U 3,1Ghz. As bases foram obtidas através da instalação do aplicativo em um *smartphone* Samsung da série Galaxy S5 2GB RAM Quad-Core 2,5Ghz tela de 5,1 polegadas.

3. Experimentos e resultados

Nesta seção apresentamos a quantidade de dados da base e também os resultados do reconhecimento de caracteres.

Para a coleta de dados para a construção da base, foram convidadas cinco pessoas de idade entre 12-20 anos para escreverem as vinte e seis letras do alfabeto (A-Z) cinco vezes, totalizando 650 amostras.

Das 650 amostras, um total de 582K, sendo 520 (corresponde a 80%) foram utilizadas para treinamento e 130 (corresponde a 20%) para testes. A permutação foi realizada para que todos os dados pudessem fazer parte tanto da base de testes quanto da base de treinamento.

Intuitivamente, pode-se deduzir uma taxa de acerto aceitável no reconhecimento de caracteres de maneira empírica observando os dados contidos na base, mesmo com uma base de tamanho pequena. O motivo disso pode ser ocasionado por dois motivos: os caracteres foram "cuidadosamente" escritos sobre o aplicativo Android de maneira que houvesse poucos ruídos e distorções; a letra cursiva das pessoas que produziram a base é bastante caligrafada.

O tempo de treinamento médio foi de 3 horas e a taxa de acerto médio atingiu cerca de 62.85%. Este resultado se assemelha com os resultados obtidos por Liwicki *et al*[2], mesmo este sendo um reconhecimento de palavras e não de apenas caracteres.

4. Conclusão

Neste trabalho simulamos o estudo realizado por Liwicki[2] em uma escala menor. Os resultados obtidos foram de certo modo sastisfatórios.

Como trabalho futuro, realizar um experimento onde a etapa de pré-processamento seja realizada antes da transformação dos dados para imagem, ou seja, realizar o pré-processamento nos dados *online*, como uma forma de explorar melhor a área. Também deseja-se utilizar outros classificadores como SVM e PCA. Outro aspecto de estudo está em realizar pesquisas mais aprofundadas em reconhecedores *onlines*¸tanto diretamente para *smartphone* como apresentado por Dutta *et al.* 2012 [2], como um

reconhecedor em um servidor acessado via internet, como apresenta Gao *et al* [4].

Há ainda muito a se explorar na área de reconhecimento de manuscritos e este trabalho foi um pequeno passo para o avanço de estudos futuros.

Referências

[1] Aggarwal, Rajat, et al. "Online Handwriting Recognition using Depth Sensors." Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. IEEE, 2015.

[2] Dutta, D., et al. "Building a personal handwriting recognizer on an Android device." Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on. IEEE, 2012.

[3] Dutta, D., et al. "Stroke level user-adaptation for stroke order free online handwriting recognition." Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. IEEE, 2014.

[4] Gao, Yan, et al. "Handwriting character recognition as a service: A new handwriting recognition system based on cloud computing." Document Analysis and Recognition (ICDAR), 2011 International Conference on. IEEE, 2011.

[5] Liwicki, Marcus, and Horst Bunke. "IAM-OnDB-an on-line English sentence database acquired from handwritten text on a whiteboard." Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on. IEEE, 2005.

[6] Malik, Shumaila, and Shoab Khan. "Urdu online handwriting recognition." Emerging Technologies, 2005. Proceedings of the IEEE Symposium on. IEEE, 2005.

[7] Marti, U-V., and Horst Bunke. "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system." International journal of Pattern Recognition and Artificial intelligence 15.01 (2001): 65-90.

[8] Namboodiri, Anoop M., and Anil K. Jain. "Online handwritten script recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 26.1 (2004): 124-130.

[9] Plamondon, Réjean, and Sargur N. Srihari. "Online and off-line handwriting recognition: a comprehensive survey." Pattern Analysis and Machine Intelligence, IEEE Transactions on 22.1 (2000): 63-84.

[10] Prasad, G. Keerthi, Imran Khan, and Naveen Chanukotimath. "On-line Hindi handwritten character recognition for mobile devices." Proceedings of the International Conference on Advances in Computing, Communications and Informatics. ACM, 2012.

[11] Shivram, Arti, et al. "Segmentation based online word recognition: A conditional random field driven beam search strategy." Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, 2013.

[12] Souza, César Roberto de. "A Tutorial on Principal Component Analysis with the Accord. NET Framework." arXiv preprint arXiv:1210.7463 (2012).

[13] Sridha, Muralikrishna, Dinesh Mandalapu, and Mehul Patel. "Active-dtw: a generative classifier that combines elastic matching with active shape modeling for online handwritten character recognition." Tenth International Workshop on Frontiers in Handwriting Recognition. Suvisoft, 2006.

[14] Viard-Gaudin, Christian, et al. "The ireste on/off (ironoff) dual handwriting database." Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on. IEEE, 1999.

[15] Wakahara, Toru, Hiroshi Murase, and Kazumi Odaka. "On-line handwriting recognition." Proceedings of the IEEE 80.7 (1992): 1181-1194.

[16] Wang, Jeen-Shing, Yu-Liang Hsu, and Cheng-Ling Chu. "Online handwriting recognition using an accelerometer-based pen device." 2nd International Conference on Advances in Computer Science and Engineering. 2013.