Script Independent Word Spotting in Offline Handwritten Documents Based on Hidden Markov Models

Safwan Wshah, Gaurav Kumar and Venu Govindaraju
Department of Computer Science and Engineering
University at Buffalo
srwshah, gauravku, govind@buffalo.edu

Abstract

Keyword spotting aims to retrieve all instances of a given keyword from a document in any language. In this paper, we propose a novel script independent line based word spotting framework for offline handwritten documents based on Hidden Markov Models. The methodology simulates the keywords in model space as a sequence of character models and uses the filler models for better representation of background or non-keyword text. We propose a two stage spotting framework where the candidate keywords are further pruned using the character based background and lexicon based background model. The system deals with large vocabulary without the need for word or character segmentation. The system has been evaluated on many public dataset from several languages such as IAM for English, AMA for Arabic and LAW for Devanagari. The system outperforms the modern line based approach on the English, Arabic and Devanagari Datasets.

1. Introduction

Despite the great progress made in handwriting recognition systems during the last decade, it still remains a challenging problem due to different writing styles and large vocabulary[2][11]. As a result, several word spotting techniques have been proposed instead of complete recognition systems to retrieve keywords from document images. The optimum trend in word spotting systems is to propose methods that show high accuracy, high speed and work on any language with minimum prepossessing steps such as preparing the query format or word segmentation. We categorize the word spotting approaches into three main categories: template based, word based and line based. In the template based approach, input image is matched to a set of template keyword images and the outputs are the images most similar to the query image. The image is represented as a sequence of features and usually compared with dynamic time warping (DTW) technique [9][13][16]. The main advantage of this approach is that there is minimum learning involved. However, we need at least one keyword sample in the training dataset. Moreover, the text line images have to be segmented into words and there are limitations of dealing with wide variety of unknown writers [6].

In word based spotting such as [14] and [15], the HMM model for each keyword is trained separately. The score of each HMM is normalized with respect to the score of the same topology HMM trained for all non-keywords. This approach relies heavily on perfect word segmentation and requires several samples for each keyword in the training set.

In the line based approach, the word or character segmentation step is done during the spotting process. In [4] and [5], character HMMs are trained from manually segmented templates assuming small variation in data. In [6], a line level approach is presented using HMM character models under the assumption that no more than one keyword can be spotted in a given line. Their approach outperformed the template based methods for single writer with few training samples and multi-writers with many training samples. A major drawback in their approach is the dependency on the white space to separate keywords from rest of the text. This not only has a large influence on the spotting results but also prevents the system from being scalable over other languages such as Arabic in which the space could be within or between the words revealing little information about the word boundaries [4]. Besides, the lexicon free approach to model the non-keyword has large negative effect on their system performance as well. For more details about their algorithm refer to [6].

There has not been much research focused on multi-

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.264

lingual word spotting systems. Srihari [16] and Bhardwaj [3] proposed work based on template matching for Arabic, English, and Hindi using GSC, and moment features. These approaches suffer from the same limitations of template based approaches.

In this paper we propose a novel methodology for word spotting in offline handwritten documents based on Hidden Markov Models (HMM). We learn HMMs of trained characters and combine them to simulate any keyword, even those unseen in the training corpus. We use filler models for better representation of non-keyword image regions avoiding the limitations of line-based keyword spotting technique, which largely relies on lexicon free score normalization and white space separation. Our system is capable of dealing with large vocabulary without the need for word or character segmentation and scalable over many languages such as English, Arabic and Devanagari. The main characteristic of the proposed approach is utilizing script-independent methods for feature extraction, training and recognition. The system has been evaluated on public datasets of different languages such as IAM [10] for English, AMA [1] for Arabic and LAW [7] for Devanagari or Indian languages. The system outperforms the modern line based approach presented in [6] for all the languages as shown in the experimental results.

2. Preprocessing and Feature Extraction

We assume that the document lines have been segmented and the image height resized to a fixed value maintaining the aspect ratio.The input line image is cleaned, deskewed and slant corrected. For each document line, features are computed from a sequence of overlapped windows, also called a frame, and gradient and intensity features are extracted for each frame. The selected frame width is 20 pixels with frame overlap of 85%. Each frame is divided into two vertical cells known as bins. The division is based on center of mass of its black pixels. For each bin, the gradient features are calculated from a normalized histogram of 8-directions. The intensity features were extracted by dividing each bin into four equal horizontal strips and count of black to white pixels was normalized for each strip. As a result, 24 features are extracted for each frame as shown in figure 1.

3. Character Models

Feature extraction procedure converts the text line into a set of features F as shown in figure 1. Given an Observation vector O, the i-th observation value O_i

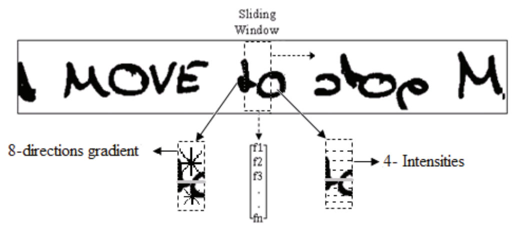

Figure 1: Feature Extraction

corresponds to the ith dimension in the feature vector f_i. For each character, a 14-state linear topology HMM is learnt. For each state S_i, the observation probability distribution is modeled by a continuous probability density function of Gaussian mixture. Each line is constructed from words that are formed by concatenating the corresponding character HMM models which we refer as recognition network. The recognizer finds the highest probability path through the network using Viterbi algorithm [12].

4. Proposed Model

We propose a novel script independent line based spotting framework. Given a line image, the goal is to locate and recognize one or more occurences of a keyword. The algorithm starts by detecting the candidate keywords using a recognizer that searches for the keywords in a line. Keyword models consist of all keywords built by concatenating their HMM character models. The HMM based recognizer uses the Viterbi beam search decoder [12] to parse the entire line finding the maximum probability path between the keywords and filler models as shown in figure 2. Each keyword candidate is processed by extracting it from the line using the start and end position. Then the score of each candidate keyword is normalized with respect to the score of word background models. Our approach utilises both the filler and background models. The distinction between the models is explained below.

4.1. Filler Models

Filler models are used to model the non-keywords without explicitly defining them. They allow separation of keyword text from non-keyword background. While the proper modeling of non-keywords reduces the rate of false positives, the proper modeling of the keywords increases the true positive rate. We investigate several filler models such as sub-words including characters or sequence of characters. Table 1 contains the summary of filler models that we propose and evaluate, including

Table 1: Filler models

Model	Description
Characters	Group of character models used as fillers.
Bigram context dependent Character models	All bigram context dependent character not found in the keyword list used as filler models.
Bigram context independent sub-words	All bigram characters sequence not found in the keyword list is modeled used as filler models.
Trigram context dependent Character models	All Trigram context dependent character not found in the keyword list used as filler models.
Trigram context independent sub-words	All Trigram characters sequence not found in the keyword list is modeled used as filler models.

characters, bigram-subwords or trigram-subwords. All these models are compared based on the accuracy, computation complexity and simplicity of the implementation.

In case of context-dependent bigram and trigram models, all character models are trained based on their position in the context. All context-dependent characters not appearing in keyword models are used as fillers. This technique requires an exceptionally large training data for the purpose of training a large number of context dependent character models. In the case of context independent filler model all the non-keyword character sequences not appearing in the keywords are used as filler models. Since the number of non-keywords sequences is huge, it adds more complexity to the system.

Character filler models (CFM) can significantly reduce the computational complexity making it more attractive for real applications due to fewer models and high efficiency. Each CFM is an HMM model that has exactly same implementation of the character models but trained on different classes.It is expected that the number of CFMs will affect the performance, and thus different numbers of CFMs are evaluated for each language. The clustering of these CFMs is implemented as described in algorithm 1. The candidate keywords from the filler models are pruned using the word background models to efficiently reduce the false positive rate.

Algorithm 1

```
INPUT :HMM characters models, testing data, number of the required filler models.
OUTPUT: Character filler models.
Initialization: INPUT ← HMM character models. OUTPUT ← ”
Step 1:
for all character model in INPUT do
    for other character models in INPUT do
        Merge the characters pair models.
        PAIRS[Accuracy, characters pair]← Evaluate the testing set accuracy after merging.
    end for
    MaxPair← Pick maximum accuracy from PAIRS.
    Merge the corresponding pair (MaxPair)and store it in OUTPUT array.
    Delete pair from INPUT.
end for
Step 2:
Label the testing data according to the new models.
Step 3:
if OUTPUT size == Number of filler models then
    end
else
    INPUT ← OUTPUT, OUTPUT ← ”, go to step 1.
end if
```

4.2. Background Model

Score normalization is an effective method to enhance accuracy. It is applied as a rejection strategy to decrease the false positive rate [14]. In this paper we present two novel methods for score normalization. The first method is based on score normalization between the keyword candidate and non-keywords scores as shown in figure 3. We refer to it as Lexicon Based Background Model. The other method is based on the character filler models as shown in figure 4 referred as Character Based Background Model.

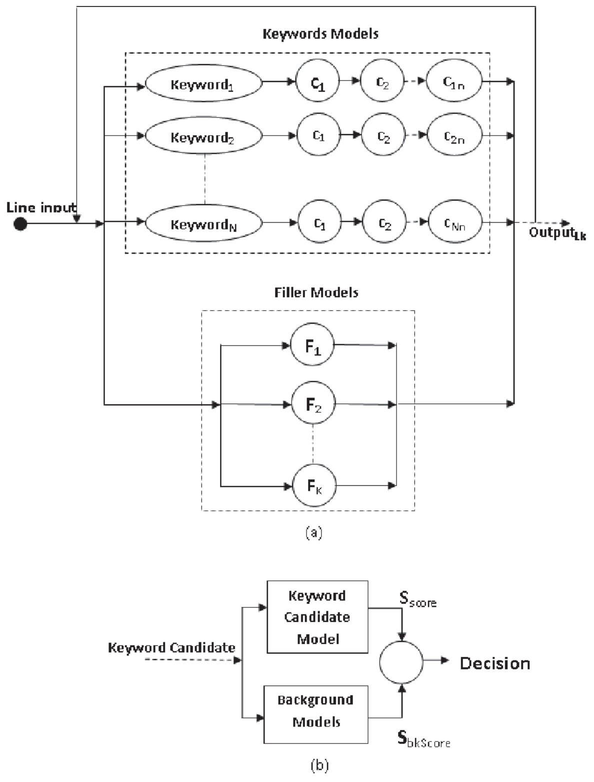

Figure 2: Main Model, (a) keywords and filler models, (b) score normalization using background models.

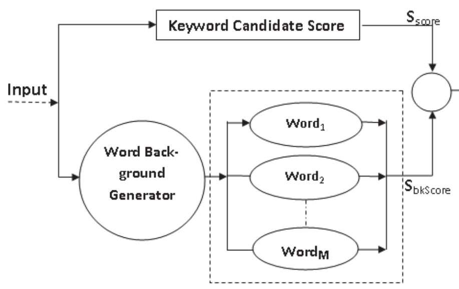

Figure 3: Lexicon based word background model

Lexicon Based Background Model: In this technique, Background model is represented by all or a subset of non-keywords.A reduced lexicon is used to overcome the high computational complexity which results from using all non-keywords. The candidate keyword recognized in filler models stage is either correct or similar to the keyword due to the fact that filler models represent the non-keywords regions. The reduction in size of the Background model is implemented based on the Levenshtein distance between all non-keywords and the candidate keyword text. The non-keyword is added to the reduced lexicon if its edit distance is less than a certain threshold. Reduced lexicon can be com-

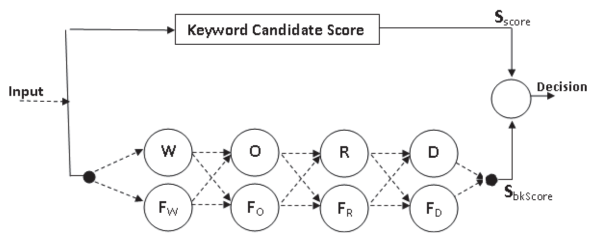

Figure 4: Word Background Model using Character Filler Models.

puted once for each keyword without adding more computation cost to the system. In general, for lexicon based background models the likelihood ratio, R between keyword candidate score (S_{score}) and the WBM scores ($S_{Lexicon\_score}$) is given by:

$$R = \frac{S_{score}}{S_{Lexicon\_score}} \tag{1}$$

If R is larger than 1, this means the candidate is most likely a keyword. The likelihood ratio R is normalized by the width of the keyword width. Positive match is considered if the normalized likelihood score is greater than a certain threshold T.

$$\frac{R}{W} > T \tag{2}$$

Character Based Background Model: The second background model is based on the character filler models as shown in figure 4. The candidate keyword is evaluated over the background models as the best path between keyword candidate characters and their corresponding character filler models. Thus, obtaining the separation amount between the keyword and the background. The complexity of this technique is very low compared to the lexicon free and reduced lexicon techniques. The normalized likelihood score is the ratio, R between keyword candidate score (S_{score}) and the sum of background character scores ($S_{bkscore}$):

$$R = \frac{S_{score}}{\sum_i S_{bkscore}(i)} \tag{3}$$

If R is closer to 1, it is most likely to be a keyword. The likelihood ratio R is normalized with the width of the keyword (W) and a Positive match is considered if the normalized likelihood score is within certain thresholds.

$$T_1 > \frac{R}{W} > T_2 \tag{4}$$

5. Experimental Results

Three public dataset are used for the experimental evaluation: the public IAM dataset [10] for English, the public AMA [1] dataset for Arabic, and LAW [7] dataset for Devanagari.

IAM English dataset: A Modern English handwritten dataset consists of 1539 pages text from the LancasterOslo/Bergen corpus [8]. The dataset has been written by 657 writers. 3200 lines are used for training and 1000 line for testing. For more details about this dataset refer to[10].

AMA Arabic dataset: A Modern Arabic handwritten dataset of 200 unique documents consisting of 5000 documents transcribed by 25 writers using various writing utensils. 4200 lines are used for training and 1000 lines for testing. For more details about this dataset refer to [1].

LAW Devanagari dataset: The Devanagari lines are semi automatic and are formed by randomly concatenating up to 5 words from a dataset containing 26,720 handwritten words written in Hindi and Marathi languages (Devanagari script) separated by a random space size. We use 2000 lines for training and 1000 lines for testing. For more details about this dataset refer to [7].

The results are measured using precision, recall and mean average precision (MAP). Recall = $\frac{TP}{(TP+FN)}$ and Precision = $\frac{TP}{(TP+FP)}$, where TP: True positive, FN: False negative and FP: False positive. Mean Average Precision (MAP) is given by the area under curve of recall vs precision graph. To evaluate the filler types, an experiment for each type is implemented. All the non-keywords are used to represent the background models to study the effect of the filler types only. IAM dataset is used with three different numbers of keywords, 30, 100 and 500 respectively. The keywords are randomly selected with 20% not present in the testing set. Figure 5 shows that character filler models outperform all other filler types, making it the most attractive model due to its excellent performance and low complexity. The number of best character filler models found for English, Arabic and Devanagari are 4, 11 and 7 respectively. The main reason for using the lexicon reduction method is to reduce the computational complexity without affecting the performance. The reduction based on Levenshtein distance shows an effective method of large lexicon reduction with slight affect on the systems performance due to the similarity between the detected candidate and the keyword text. As a result, a huge reduction in the computational complexity is achieved. Figure 6 shows the performance of different background models including background character based method and lexicon based method with different reduction ratios.

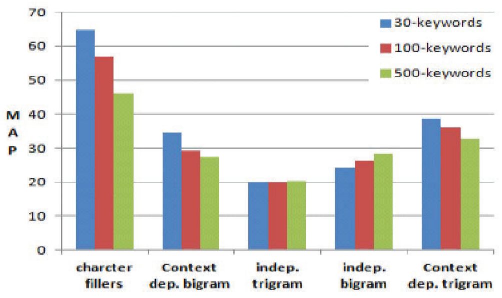

Figure 5: Filler types performance

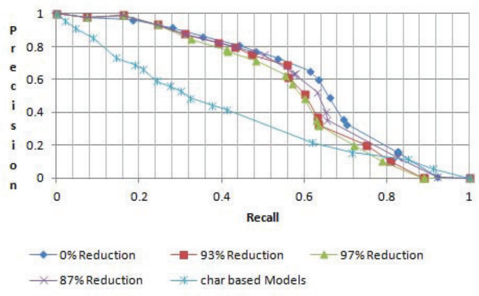

Figure 6: Background models performance

We compare the proposed system with similar line based approach proposed by Fischer[6]. We implemented their algorithm using the same HMM models topology. We used same features and data as used in our system for comparison.Thus both systems were compared in exactly same environment. Our proposed method outperforms their method on different size of keyword list as shown in figure 7.Fischer's algorithm [6] can detect up to one keyword per line. The best results for their algorithm can be obtained when each line contains only one keyword. There are many false positives detected in the lines that do not contain keywords due to the high error rate of the lexicon free technique. In addition their dependency on the white space model affect the performance to a high degree particularly for Arabic in which the space could be within or between the words revealing little information about the word boundaries as shown in figure 7.Proposed word spotting system is able to retrieve any number of keywords and of any size.

Figure 7 shows the proposed system evaluation on other languages such as Arabic and Devanagari. The same feature extraction, character filler models and background models are used. We used 150 keywords for Arabic and English, and 30 keywords for Devanagari. Character filler models and 90% reduced lexicon background models are used. For each language the re-

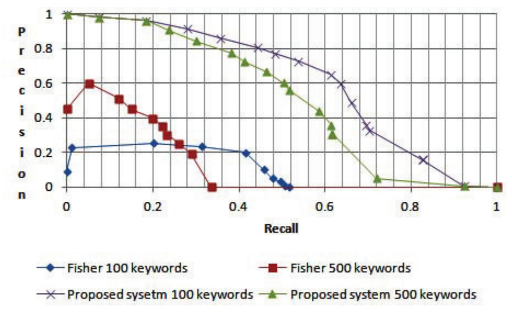

Figure 7: Comparison with the previous methods performance

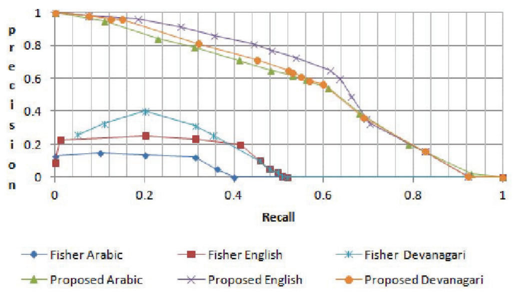

Figure 8: Performance on English, Arabic and Devanagari compared with line based approach.

sults are compared to the method presented in [6] showing our proposed system had better results over all the languages.

6. Conclusion

In this paper we proposed a learning based script independent word spotting system effectively capable of dealing with large vocabularies without the need for word or character segmentation. The high performance and low complexity of the character filler models make them attractable for real systems. Lexicon reduction using Levenshtein distance allows a high degree of lexicon reduction thus reducing the complexity which results from large lexicon size without affecting the performance. The proposed system has been evaluated on several languages including English, Arabic and Devanagari. The system shows high performance compared to the previous line based algorithms. In future work, the algorithm will be extended to work on multilingual documents and investigate the possibility of using language models to enhance the performance.

References

[1] Applied media analysis, arabic-handwritten-1.0. 2007.

[2] Alessandro and Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35(7):1433 – 1446, 2002.

[3] A. Bhardwaj, D. Jose, and V. Govindaraju. Script independent word spotting in multilingual documents. *2nd Intl Workshop on Cross Lingual Information Access*, pages 48–54, 2008.

[4] J. Chan, C. Ziftci, and D. Forsyth. Searching offline arabic documents. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1455–1462, Washington, DC, USA, 2006. IEEE Computer Society.

[5] J. Edwards, Y. W. Teh, D. Forsyth, R. Bock, and M. Maire. Making Latin manuscripts searchable using gHMMs. *NIPS*, 2004.

[6] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern Recogn. Lett.*, 33(7):934–942, May 2012.

[7] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal. Database development and recognition of handwritten devanagari legal amount words. *Document Analysis and Recognition, International Conference on*, 0:304–308, 2011.

[8] L.-G. G. H. Johansson, S. Manual of Information to Accompany the LancasterOslo/Bergen Corpus of British English, for Use with Digital.

[9] R. Manmatha, C. Han, and E. Riseman. Word spotting: a new approach to indexing handwriting. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 631 –637, jun 1996.

[10] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002. 10.1007/s100320200071.

[11] R. Plamondon and S. N. Srihari. On-line and offline handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):63–84, Jan. 2000.

[12] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[13] T. M. Rath and R. Manmatha. Word spotting for historical documents. *INTERNATIONAL JOURNAL ON DOCUMENT ANALYSIS AND RECOGNITION*, pages 139–152, 2007.

[14] J. A. Rodrguez-Serrano and F. Perronnin. Handwritten word-spotting using hidden markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106 – 2116, 2009.

[15] R. Saabni and J. El-Sana. Keyword searching for arabic handwritten documents. *11th International Conference on Frontiers in Handwriting recognition (ICFHR2008)*, pages 716–722, 2008.

[16] S. N. Srihari, H. Srinivasan, C. Huang, and S. Shetty. Spotting words in latin, devanagari and arabic scripts. *Artificial Intelligence*, page 2006.