Two Methods to Improve Confidence Scores for Lexicon-free Word Spotting in Handwritten Text

Alejandro Héctor Toselli
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
Email: ahector@prhlt.upv.es

Joan Puigcerver
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
Email: joapuipe@prhlt.upv.es

Enrique Vidal
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
Email: evidal@prhlt.upv.es

***Abstract*—Two methods are presented to improve word confidence scores for Line-Level Query-by-String Lexicon-Free Keyword Spotting (KWS) in handwritten text images. The first one approaches true relevance probabilities by means of computations directly carried out on character lattices obtained from the lines images considered. The second method uses the same character lattices, but it obtains relevance scores by first computing frame-level character sequence scores which resemble the word posteriorgrams used in previous approaches for lexicon-based KWS. The first method results from a formal probabilistic derivation, which allow us to better understand and further develop the underlying ideas. The second one is less formal but, according with experiments presented in the paper, it obtains almost identical results with much lower computational cost. Moreover, in contrast with the first method, the second one allows to directly obtain accurate bounding boxes for the spotted words.**

I. Introduction

There are large quantities of digitized historical handwritten documents owned by libraries and archives. As most of such documents lack transcripts, the information conveyed by the text contained in them remain practically inaccessible. Thus, to exploit and make profit of such a mass-digitization efforts, fast information retrieval approaches are required to allow the users to search accurately and efficiently for textual contents in such material.

Aiming at this goal, *Keyword spotting* (KWS) on handwritten documents arises as a good alternative for making the information available to the users. Word spotting can be considered as a special case of image retrieval, whose goal is to find all instances of a given query word in the document collection. Among the noteworthy KWS paradigms aiming to fulfill the above mentioned goal, here we adopt the segmentation-free, query-by-string, training-based framework. Within this paradigm, there are the so-called lexicon-free methods, which as their name indicate, don't relay on a predefined vocabulary and therefore allow users to search for any wanted query. As examples of lexicon-free KWS approaches for handwritten documents, we can mention the popular ones based on *Hidden Markov Model filler* (HMM-Filler) [1], [2], [3] or those based on *recurrent neural networks* (RNNs) [4], [5].

In [6], a probabilistic framework for lexicon-free KWS was proposed, of which the traditional HMM-Filler emerges as a particular case. Furthermore in that work a efficient method is presented to compute the confidence scores required by HMM-Filler using character lattices. These lattices are obtained as a byproduct of the decoding process of text images by a recognition system using the HMM character filler model. The distribution of the log-likelihoods of paths in these character lattices is generally very sharp-peaked. This is because the decoding parameters are generally optimized to achieve the least error rate for the first-best hypotheses of the recognizer. As a result, most of the probability mass is concentrated practically on the first-best path, therefore making negligible the contribution of the remaining paths. It is generally acknowledged (see e.g., [7]) that, by changing adequately the logarithm base of the log-likelihood scores of word-graph arcs, a wider-spread distribution of path log-likelihoods can be obtained – thereby improving the effectiveness of resulting word confidence scores and the KWS accuracy.

In this work we explore this idea to see whether resulting confidence scores computed according to the method proposed in [6] (from now on named *forward* method) can also improve KWS accuracy. In addition, we present an alternative method for computing confidence scores, which resembles the posteriorgram-based methodology for KWS at word level proposed in [7]. Actually this new method, hereafter called *posteriorgram-like*, uses a generalization of the algorithm already proposed in [8]. We show empirically that, as compared with the *forward* method [6], the *posteriorgram-like* method has two main advantages: first it produces practically identical confidence scores, leading to the same KWS accuracy, but requiring much less computing time; and second, in contrast with the *forward* method it provides accurate image locations of spotted words.

The rest of the paper is organized as follows. The next section overviews the statistical framework for KWS. Sec. III provides some background of HMM-based recognition and essential details of character lattices. Sec. IV explains how confidence scores of words represented by character sequences can be directly computed using CLs. Sec. V describes the more efficient alternative to compute word confidence scores based on posteriorgram-like functions. Evaluation measures, corpora, experimental set-up and results are presented/reported in section VI. Finally, section VII summarizes the work presented and draws conclusions.

II. Probabilistic Framework for KWS

As in [6], we define a binary random variable $\mathbf{R}$ to denote whether an image region (e.g., a line) $\mathbf{x}$ is relevant for the query keyword v formed by the concatenation of L characters $c_1, c_2 \ldots, c_L \stackrel{\text{def}}{=} \mathbf{c}_\text{v}$. An image is considered to be "relevant" if $\mathbf{c}_\text{v}$ is actually written in it. Let us also consider the probability distribution $P(\mathbf{R} \mid \mathbf{V}, \mathbf{X})$, where $\mathbf{V}$ and $\mathbf{X}$ are random variables over all possible query keywords and all possible image regions respectively. Note that, the probability we seek to compute is $P(\mathbf{R} = 1 \mid \mathbf{V} = \mathbf{c}_\text{v}, \mathbf{X} = \mathbf{x})$. From now on, for sake of clarity, we will omit the random variable names, except for $\mathbf{R} = 1$, which we will write simply as $\mathbf{R}$.

The probability $P(\mathbf{R} \mid \mathbf{c}_\text{v}, \mathbf{x})$ is somewhat related to the distribution of the possible transcripts of the image, $P(\mathbf{c} \mid \mathbf{x})$. If we marginalize the distribution $P(\mathbf{R} \mid \mathbf{c}_\text{v}, \mathbf{x})$ over all possible transcripts which include the keyword $\mathbf{c}_\text{v}$, we obtain:

$$P(\mathbf{R} \mid \mathbf{c}_\text{v}, \mathbf{x}) \;=\; \sum_{\mathbf{c}' \in \Sigma^*} P(\mathbf{R}, \mathbf{c}' \mid \mathbf{c}_\text{v}, \mathbf{x}) \;=\; \sum_{\mathbf{c} \in \Sigma^* \, \mathbf{c}_\text{v} \, \Sigma^*} P(\mathbf{c} \mid \mathbf{c}_\text{v}, \mathbf{x})$$

For all addends in the last sum, $\mathbf{c}$ contains $\mathbf{c}_\text{v}$ and $P(\mathbf{c} \mid \mathbf{c}_\text{v}, \mathbf{x})$ is therefore independent of $\mathbf{c}_\text{v}$, resulting in:

$$P(\mathbf{R} \mid \mathbf{c}_\text{v}, \mathbf{x}) \;=\; \sum_{\mathbf{c} \in \Sigma^* \, \mathbf{c}_\text{v} \, \Sigma^*} P(\mathbf{c} \mid \mathbf{x}) \;=\; \sum_{\mathbf{c} \in \Sigma^* \, \mathbf{c}_\text{v} \, \Sigma^*} \frac{p(\mathbf{c}, \mathbf{x})}{p(\mathbf{x})} \qquad (1)$$

This equation provides an expression of the probability that an image $\mathbf{x}$ is relevant for a keyword $\mathbf{c}_\text{v}$, as the sum of the likelihoods of all transcripts $\mathbf{c}$ of line image $\mathbf{x}$ containing the keyword $\mathbf{c}_\text{v}$, divided by the total likelihood of the line image.

III. HMM-based Recognition and Character Lattices

To better understand the processes involved in the computation of confidence scores using character lattices, the next sections review some basic concepts of HMM-based recognition and the associated character lattices.

A. HMM-based Handwriting Recognition

Segmentation-free character-level HMM recognition is considered here. Recognizers of this kind accept a given handwritten text line image, represented as a sequence of D-dimensional feature vectors $\mathbf{x} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_M$, $\vec{x}_i \in \Re^D$, as input and find a most likely character sequence, $\hat{\mathbf{c}} \in \Sigma^*$, where Σ is the character alphabet. Moreover, rather than obtaining just a single-best, most likely character sequence, a set of k best character sequences can be obtained as:

$$\begin{aligned} \left\{\widehat{\mathbf{c}}^1, \widehat{\mathbf{c}}^2, \ldots, \widehat{\mathbf{c}}^k\right\} &= \underset{\mathbf{c}^1, \ldots, \mathbf{c}^k}{\text{k-best}} P(\mathbf{c} \mid \mathbf{x}) = \underset{\mathbf{c}^1, \ldots, \mathbf{c}^k}{\text{k-best}} p(\mathbf{c}, \mathbf{x}) \\ &= \underset{\mathbf{c}^1, \ldots, \mathbf{c}^k}{\text{k-best}} p(\mathbf{x} \mid \mathbf{c}) \cdot P(\mathbf{c}) \qquad (2) \end{aligned}$$

The conditional density $p(\mathbf{x} \mid \mathbf{c})$ is approximated by previously trained optical character HMMs, while the prior distribution, $P(\mathbf{c})$, is given by a character-level N-gram model. Eq. (2) is commonly solved by means of Viterbi decoding [9].

For very large values of k, the set of the most likely character sequences can be approximately represented in a compact way in the form of a *character-lattice* (CL). This kind of lattices are better known in the literature as *"word graphs"* [10].

B. Character-Lattices

A CL of a line image represented by its feature vector sequence $\mathbf{x}$ is a weighted *directed acyclic graph* (DAG) with finite sets of nodes Q and edges E. Q includes an initial node $q_I \in Q$ and a set of final nodes $F \subseteq (Q - q_I)$. Each node q is associated with a horizontal position ("frame") of $\mathbf{x}$, given by $t(q) \in [0, M]$, where M is the length of $\mathbf{x}$. For an edge $(q', q) \in E$ ($q' \neq q, q' \notin F, q \neq q_I$), $c = \omega(q', q) \in \Sigma$ is its associated character and $r(q', q)$ is its score, corresponding to the HMM likelihood that the character $\omega(q', q)$ appears between frames $t(q')+1$ and $t(q)$, computed during the Viterbi decoding of $\mathbf{x}$. This score is typically weighted by the so-called *log-base factor* parameter, b, as follows:

$$s(q', q) \;=\; r(q', q)^{1/b}$$

In what follows we assume the weight of each CL edge (q', q) to be given by $s(q', q)$. The parameter b affects the impact of the different CL path scores in the computation of $S(\mathbf{x})$ (see below). However, these changes do not affect the best (or the k-best order of) character sequence(s) in the CL. Actually the goal of the parameter b is to obtain a wider-spread distribution of the CL path likelihoods as was commented in Sec. I. In practice, by tuning b adequately, significant KWS performance improvements can be achieved. Fig. 1 shows an illustrative example for a CL resulting from decoding an image containing the handwritten text "*to be for*".

A *sub-path* of a CL is an ordered sequence of edges $\mathbf{e} \equiv (q'_1, q_1), (q'_2, q_2), \ldots, (q'_L, q_L)$ such that $q_i = q'_{i+1}, 1 \leq i < L, L \leq M$. A *complete path* of a CL is a sub-path such that $q'_1 = q_I$, $q_M \in F$ and $L = M$. Complete paths correspond to whole line decoding hypotheses. The score of a complete path, $\mathbf{e}$, is the product of the scores of all the edges of $\mathbf{e}$:

$$S(\mathbf{e}, \mathbf{x}) = \prod_{i=1}^{M} s(q'_i, q_i)$$

Let $\Omega : E^* \rightarrow \Sigma^*$ be an extension of $\omega(\cdot)$, which denotes the character sequence associated with a given sub-path $\mathbf{e}$. The likelihood score of a character sequence $\mathbf{c} \in \Sigma^*$ is the sum of $S(\mathbf{e}, \mathbf{x})$ for all complete paths $\mathbf{e}$ such that $\Omega(\mathbf{e}) = \mathbf{c}$. That is;

$$S(\mathbf{c}, \mathbf{x}) = \sum_{\mathbf{e}: \Omega(\mathbf{e}) = \mathbf{c}} S(\mathbf{e}, \mathbf{x}) \qquad (3)$$

And the total likelihood score of $\mathbf{x}$, $S(\mathbf{x})$, is the sum of the scores of all complete CL paths. If a CL is large enough, the scores $S(\mathbf{c}, \mathbf{x})$ and $S(\mathbf{x})$ are good approximations to the probabilities $p(\mathbf{c}, \mathbf{x})$ and $p(\mathbf{x})$ which appear in Eq. (1) and (2).

As discussed in [7], $S(\mathbf{x})$ can be very efficiently computed by means of *forward* (α) or *backward* (β) accumulated path

scores, recursively computed by Dynamic Programming [10]:

$$S(\mathbf{x}) = \sum_{\mathbf{e}} S(\mathbf{e}, \mathbf{x}) = \beta(q_I) = \sum_{q \in F} \alpha(q) \tag{4}$$

where:

$$\alpha(q) \;\; = \sum_{i:(q_i,q)\in E} \alpha(q_i)\, s(q_i, q) \quad \text{if } q \neq q_I \tag{5}$$

$$\beta(q) \;\; = \sum_{j:(q,q_j)\in E} s(q, q_j)\, \beta(q_j) \quad \text{if } q \notin F \tag{6}$$

with $\alpha(q_I) = 1$, and $\beta(q\,) = 1 \;\; \forall q \in F$.

The overall *forward-backward* computing time is negligible with respect to that of CL generation, $O(\Gamma \cdot n)$, where Γ is a (generally large) constant which depends on the CL size [7].

IV. Computing Confidence Scores directly from Character-Lattice

Eq. 1 can be directly rewritten in terms of CL approximations (see Sec. III-B) as:

$$P(\mathbf{R} \mid \mathbf{c}_{\mathrm{v}}, \mathbf{x}) \;\approx\; \frac{1}{S(\mathbf{x})} \sum_{\mathbf{c} \in \Sigma^* \,\mathbf{c}_{\mathrm{v}}\, \Sigma^*} S(\mathbf{c}, \mathbf{x}) \tag{7}$$

Finally we can apply the same heuristics used in the computation of HMM-Filler scores [1], [6], in order to mitigate the problem of the exponential decay of the likelihood $p(\mathbf{c}, \mathbf{x})$ with the length of $\mathbf{x}$ and $\mathbf{c}$; that is:

$$P(\mathbf{R} \mid \mathbf{c}_{\mathrm{v}}, \mathbf{x}) \;\approx\; \left(\frac{1}{S(\mathbf{x})} \sum_{\mathbf{c} \in \Sigma^* \,\mathbf{c}_{\mathrm{v}}\, \Sigma^*} S(\mathbf{c}, \mathbf{x}) \right)^{1/L^{\sigma}} \tag{8}$$

where L is the number of characters of $\mathbf{c}_{\mathrm{v}}$ and σ is weighting parameter to be tuned empirically.

The sum in Eq. (8) can be obtained by means of the *forward* computation (Eq. (5)) applied to a special DAG obtained from the original CL and the keyword $\mathbf{c}_{\mathrm{v}}$. To do that, an algorithm based on the well-known Knuth–Morris–Pratt algorithm [11] for string matching, is used to build a *Deterministic Finite Automaton* (DFA) that accepts all the sequence of characters in Σ^* that contain the keyword $\mathbf{c}_{\mathrm{v}}$, that is the regular language $\Sigma^*\, \mathbf{c}_{\mathrm{v}}\, \Sigma^*$. Fig. 2 shows this automaton for the exemplar keyword "*soup*". Then, the previous DFA is simply intersected with the CL, obtaining a new weighted DAG which encodes all the possible paths from the original CL containing the keyword $\mathbf{c}_{\mathrm{v}}$. The *forward* likelihood in the final node of the intersection graph is equal to the sum in Eq. 8.

Moreover, both the step of intersecting and the *forward* computation can be directly tackled using an embedded dynamic-programming algorithm, which avoids creating the keyword DFA and intersecting the graphs explicitly. This way has the same asymptotic running time, $O((|Q| + |E|) \cdot L)$, where Q and E are the set of nodes and edges of the resulting graph, respectively.

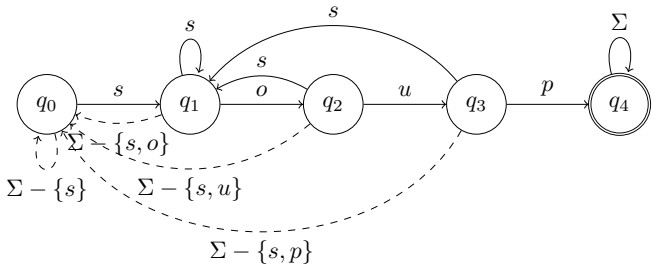

Fig. 2. Complete DFA accepting all strings containing the sequence of characters: *s o u p* .

V. Posteriorgram-like based Computation of Confidence Scores

We introduce here an alternative approach for computing the confidence scores, $P(\mathbf{R} \mid \mathbf{c}_{\mathrm{v}}, \mathbf{x})$, which in comparison with the previous one, has a much lower computational cost and, according to the experiments to be presented below, dose not affect the KWS accuracy achieved.

Let $\mathbf{e}'$ be a *sub-path*. The score $\varphi(\mathbf{e}')$, named henceforward *edge sequence normalized score*, is the normalized sum over all complete paths $\mathbf{e}$ which contain the sub-path $\mathbf{e}'$ (denoted $\mathbf{e}' \subseteq \mathbf{e}$, for simplicity). This sum can be efficiently computed by means of the *backward* and *forward* accumulated scores of Eq. (5–6):

$$\varphi(\mathbf{e}') \;\stackrel{\text{def}}{=}\; \frac{\displaystyle\sum_{\mathbf{e}:\mathbf{e}'\subseteq\mathbf{e}} S(\mathbf{e}, \mathbf{x})}{S(\mathbf{x})} \;=\; \frac{\alpha(q'_1) \displaystyle\prod_{i=1}^{L} s(q'_i, q_i)\, \beta(q_L)}{\beta(q_I)} \tag{9}$$

This score can be seen as a multi-edge extension of what is often called "edge posterior" (see [7], e.g.).

Now, for a given character sequence $\mathbf{c}_{\mathrm{v}}$ to be spotted, we define a *frame-level character sequence score* for each horizontal position or *frame*, i, of the given image $\mathbf{x}$. It considers the contribution of the edge sequence normalized scores of all the CL sub-paths labeled with $\mathbf{c}_{\mathrm{v}}$, which correspond to segmentation hypotheses that include the frame i; that is:

$$S(\mathbf{c}_{\mathrm{v}}, \mathbf{x}, i) \;\stackrel{\text{def}}{=} \sum_{\substack{\mathbf{c}_{\mathrm{v}} = \Omega(\mathbf{e}),\\ \mathbf{e}=(q'_1,q_1),\ldots,(q'_L,q_L),\\ t(q'_1) < i \leq t(q_L)}} \varphi(\mathbf{e}) \tag{10}$$

Alg. 1 shows how to compute this sum efficiently.

$S(\mathbf{c}_{\mathrm{v}}, \mathbf{x}, i)$ resembles what is often referred to as *posteriorgram* [12], [7]. Fig. 1 shows an example of posteriorgram-like function computed for the query "*to*" from the CL produced during the decoding of the line image "*to be for*".

As discussed in [7], the word confidence score $P(\mathbf{R} \mid \mathbf{c}_{\mathrm{v}}, \mathbf{x})$ is simply computed as the maximum over i of $S(\mathbf{c}_{\mathrm{v}}, \mathbf{x}, i)$. In addition, we also apply here the same heuristic used in Sec. IV to mitigate the exponential decay of scores with the length of $\mathbf{x}$ and $\mathbf{c}$. This results in:

$$P(\mathbf{R} \mid \mathbf{c}_{\mathrm{v}}, \mathbf{x}) \;\approx\; \max_i S(\mathbf{c}_{\mathrm{v}}, \mathbf{x}, i)^{\frac{1}{L^{\sigma}}} \tag{11}$$

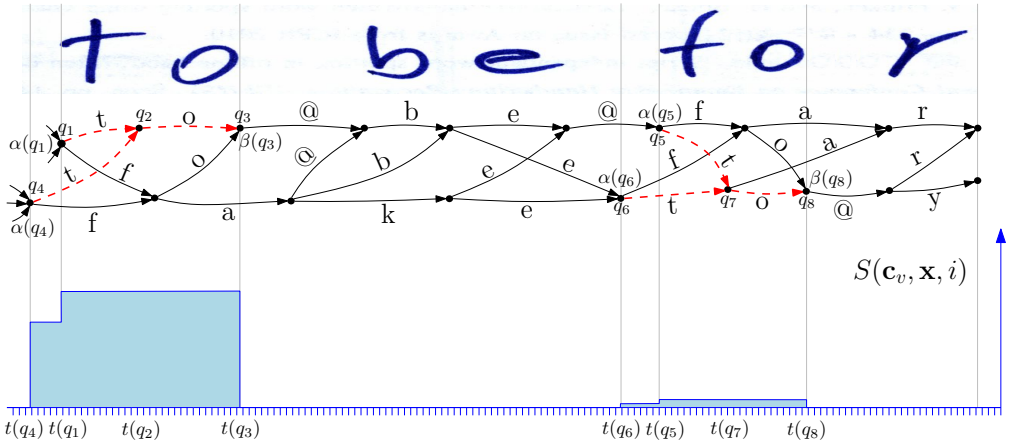

Fig. 1. Top: A small, partial example of CL for a handwritten text "**to be for**". The edge sequence paths: $\{(q_1, q_2), (q_2, q_3)\}$, $\{(q_4, q_2), (q_2, q_3)\}$, $\{(q_5, q_7), (q_7, q_8)\}$ and $\{(q_6, q_7), (q_7, q_8)\}$ (in dashed red lines) correspond to the query character sequence: $\mathbf{c}_v$="**to**". The "@" symbol stands for the white space character. Bottom: Corresponding posteriorgram-like frame-level function $S(\mathbf{c}_v, \mathbf{x}, i)$. Note that the score $S($"to"$, \mathbf{x}, i)$ is not negligible in the interval where the word "**for**" appears in the image. However is much lower than in the correct interval where "**to**" is actually written.

Algorithm 1 Comput. of the frame-level score: $S(\mathbf{c}_v, \mathbf{x}, i)$

Input: query word: $\mathbf{c}_v \equiv c_1, c_2, \ldots, c_L$.
CL generated from decoding an input image of length n. $\alpha(q_j), \beta(q_j) \ : \ j = 1, 2, \ldots, \mid Q \mid$ computed for all CL nodes.

Output: $\mathbf{S} \equiv S_1, \ldots, S_n$, frame-level score vector

Let P: stack of tuples $(\varphi \in \Re, \ k \in [0, n], \ q \in Q, \ p \in [1, L])$
P.clear(); $\mathbf{S}' \leftarrow \vec{0}$
for all $(q', q) : c_1 = \omega(q', q)$ **do**
  P.push$(\alpha(q') \cdot s(q', q), \ t(q'), \ q, \ 2)$
end for
while not P.empty() **do**
  $(\varphi, \ k, \ q, \ p) \leftarrow$ P.top(); P.pop()
  if $p \leq L$ **then**
    for all $q' : c_p = \omega(q, q')$ **do**
      P.push$(\varphi \cdot s(q, q'), \ k, \ q', \ p+1)$
    end for
  else
    $\varphi \leftarrow \dfrac{\varphi \cdot \beta(q)}{\beta(q_I)}$
    for $i \leftarrow k$ **to** $t(q)$ **do**
      $S_i \leftarrow S_i + \varphi$
    end for
  end if
end while
return $\mathbf{S}$

where, as in Eq. (8), L and σ are the number of characters of $\mathbf{c}_v$ and a weighting parameter respectively.

As commented, this method also provides the spotted word location associated to the maximum i used to compute $P(\mathbf{R} \mid \mathbf{c}_v, \mathbf{x})$; for instance, in the example shown by the Fig. 1, the best scores of $S(\mathbf{c}_v, \mathbf{x}, i)$ for the keyword "*to*" (whose maximum is $P(\mathbf{R} \mid \mathbf{c}_v, \mathbf{x})$) span positions $t(q_1)$ through $t(q_3)$.

The overall cost of computing the final confidence score given by Eq. (11), excluding CL generation and *forward-backward* computation, is determined by the costs of computing the frame-level character sequence score S for all i and the final maximization of Eq. (11). According to Alg. 1, the worst-case cost of computing $S(\mathbf{c}_v, \mathbf{x}, i), 1 \leq i \leq n$ is $O(n + l \cdot B^L)$, where l is the average length (in frames) of the sub-paths corresponding to $\mathbf{c}_v$, and $B = |E|/(|Q| \cdot |\Sigma|)$ is the CL average branching factor per character.

Real computing times of this KWS approach, compared with the one presented in Sec. IV, will be reported in Tab. II.

VI. Experiments

To compare the effectiveness and efficiency of both computing confidence score methods: the forward and the posteriorgram-like, several experiments were carried out. The evaluation measures, corpus, experimental setup and the results are presented next.

A. Evaluation Measures

The standard *recall* and *interpolated precision* measures [13] are used here. *Interpolated precision* is widely used to avoid cases in which plain precision can be ill-defined. Results are reported in terms of *average precision* (AP) [14], which is computed as the area under the Recall-Precision curve and is a popular scalar assessment measure for KWS.

On the other hand for efficiency assessment, real computing times are reported in terms of total elapsed times measured on a dedicated single-core of an Intel® Xeon™ running at 2.5GHz.

B. Corpora Description

Experiments were carried out with the IAMDB dataset. It is a publicly available, well known modern English handwritten text corpus, compiled by the FKI-IAM Research Group on the base of the Lancaster-Oslo/Bergen Corpus (LOB). The last released version (3.0) is composed of 1 539 scanned text pages, handwritten by 657 different writers and partitioned into writer-independent training, validation and test sets. The line segmentation provided with the corpus [15] is used here. Basic statistics appear in Table I.

TABLE I
BASIC STATISTICS OF THE IAMDB DATABASE AND ITS PARTITION.

	Training	Validation	Test	Total
Running chars	269 270	39 318	39 130	347 718
Running words	47 615	7 291	7 197	62 103
Lines	6 161	920	929	8 010
Char set size	72	69	65	81
Lexicon size	7 778	2 442	2 488	9 809

C. Query Set

In this work the same set of query words defined and used in [1] and [8] is adopted. It consists of all the words which appear in the IAMDB training partition, except very low frequency and *stop* words. The resulting query set contains 3 421 keywords, including numeric expressions and a few symbols.

D. Experimental Setup

The experimental setup is practically the same as in [6], [8]. Character HMMs were trained from the IAMDB training partition. A left-to-right HMM was trained for each of the elements appearing in the training text images, such as lower-case and uppercase letters, symbols, blank spaces, etc. Specific setup details of image preprocessing, feature extraction and HMM training usually adopted for IAMDB are described in [1]. Meta-parameters related to the character HMMs, such as number of states and Gaussian densities per state, were optimized on the validation data.

The N-gram models required for generating the CLs, were trained using text from the Lancaster-Oslo/Bergen corpus (LOB) [16] and smoothed using the Kneser-Ney back-off technique [17]. The text used contains a variety of printed British English texts which, in total, encompasses more than one million words. Since some of these texts were used to create the IAMDB itself, they were removed from the validation and test sets.

Once all the N-gram models had been trained, the CLs of the validation and test lines were obtained using the HTK Toolkit [18] for each N-gram model with $1 \leq N \leq 4$, and also for the 0-gram model (traditional character "filler" model without prior probabilities). Since HTK can only decode directly using up to 2-gram models, we have resorted to *re-scoring* [8] to obtain CLs for $3-$ and 4-grams. That is, a 3-gram CL is obtained by re-scoring the previous 2-gram CL with the 3-gram model and, in turn, a 4-gram CL by re-scoring the previous 3-gram CL with the 4-gram model. To obtain results for 5- and 6-grams we used CLs produced by another recognizer, iATROS [19][1], capable of dealing with N-grams of order greater than 4.

In addition, the two language model parameters *grammar scale factor* and *character insertion penalty* were also tuned empirically on the validation set for an optimum *character error rate*. After optimizing these two parameters, the CLs of the 929 test-set lines were generated. The sizes of the CLs were limited by setting the maximum *input degree* to 30 and applying beam-search pruning. Statistics and other details about the resulting CLs can be consulted in [8].

Once all CLs have been generated, the *forward-backward* scores were computed according to Eq. (5-6).

The parameters b and σ appearing in Eqs. (8), (9) and (11), were tuned empirically on the validation set to optimized the KWS accuracy.

Finally, the two methods of computing confidence scores were applied for each keyword, $\mathbf{c}_{\mathrm{v}}$, and test line image, $\mathbf{x}$ according according to what was described in Secs. IV and V.

E. Results

Experiments were carried out to compare both the effectiveness and the efficiency of the two proposed methods for KWS confidence score computation explained in previous sections.

Average Precision figures obtained using the *forward* and the *posteriorgram-like* methods are reported in Tab. II for CLs produced using N-gram models with $N \in [0, 6]$. As expected, for all the methods, the AP increases with the N-gram order. This confirms previous results [8], [6] and clearly shows the great importance of taking advantage of linguistic context for KWS. We can also see that the AP values obtained by both methods proposed in this work are identical.

For comparison purposes and to establish a baseline, the row labeled with "Filler" in Tab. II provides the AP figures obtained with the N-gram-based HMM-Filler model studied in [8]. These figures are somewhat better than those reported in [8] (and [6]) because, in order to allow fair comparison, here we have more carefully optimized all the empirical settings (cf. Sec. VI), exactly in the same way as for the here proposed approaches. The results clearly assess the superiority of the here proposed methods.

On the other hand, to assess the efficiency, real computing times (in milliseconds) are also reported in Tab. II for the two new confidence score computing methods. These results, clearly show that the posteriorgram-like method is very much faster than the forward one.

This can be understood if we take into account, for instance, that the average CL for $N = 4$ has about 125 K nodes and 557 K edges [8] and therefore the average branching factor is $B \approx 557/125) \approx 4.5$. For a typical line image of about 2 000

TABLE II
AVERAGE PRECISION (AP %) AND AVERAGE QUERY TIMES (MILLISECONDS) PER LINE AND KEYWORD IN IAMDB, USING CLS PRODUCED WITH N-GRAMS FOR INCREASING N AND DIFFERENT METHODS TO COMPUTE THE KWS SCORES.

	Approach	N-gram 0	1	2	3	4	5	6
AP	Forward	41.8	47.6	50.3	53.1	56.9	58.9	61.3
	Pstgram-like	41.8	47.6	50.3	53.1	56.9	58.9	61.3
	Filler	40.2	42.9	45.3	48.3	50.0	50.5	50.8
Time	Forward	533.8	522.2	560.0	355.1	383.7	441.5	191.9
	Pstgram-like	61.0	147.6	225.6	57.7	35.2	81.9	26.3

[1] iATROS is less efficient than the HTK decoder and, moreover, for $N < 5$ iATROS CLs tend to produce slightly worse results than those obtained with HTK. Therefore HTK+re-scoring was preferred for $N < 5$.

frames and an average keyword length of 5 characters and 50 frames, the approximate overall cost of the posteriorgram-like method would be roughly proportional to $2000 + 50 \cdot 4.5^5 \approx 9.5 \cdot 10^4$. In contrast, the cost of the forward method would be roughly proportional to $(125\,000 + 557\,000) \cdot 4 \approx 2.7 \cdot 10^6$.

VII. Remarks and Conclusions

Two methods have been have been proposed to improve word confidence scores for segmentation-free, line-level, query-by-string, lexicon-free keyword spotting in handwritten text images. The first method, which results from a formal probabilistic derivation, computes required confidence scores directly from character lattices obtained during HTR decoding of the line images considered. The less formal second method uses the same character lattices to compute frame-level character sequence scores (a posteriorgram-like function) from which the line-region confidence scores are determined by simple maximization.

According to the experimental results, both methods produce identical KWS results, but the computational cost of the second is very much lower. In addition, the second method provides accurate locations of spotted words in the image.

The KWS results of both proposed methods are very much better than those of the classical HMM-Filler approach [20], [1], [21] and significantly better than those of previous methods [22], [8], [6], more or less explicitly derived from the HMM-Filler idea, but using linguistic context in the form of character N-grams.

In addition, our second, posteriorgram-like approach, is orders of magnitude faster than the traditional implementations of HMM-Filler [1], [21], based on direct Viterbi decoding. It is also significantly faster than previous CL based implementations of HMM-Filler and contextual HMM-Filler [8], [6].

VIII. Acknowledgments

This work was partially supported by the Spanish MEC under FPU grant FPU13/06281, by the Generalitat Valenciana under the Prometeo/2009/014 project grant ALMAMATER, and through the EU projects: HIMANIS (JPICH programme, Spanish grant Ref. PCIN-2015-068) and READ (Horizon-2020 programme, grant Ref. 674943).

References

[1] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character HMMs," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 934 – 942, 2012, special Issue on Awards from ICPR 2010.

[2] S. Wshah, G. Kumar, and V. Govindaraju, "Script independent word spotting in offline handwritten documents based on hidden markov models," in *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, 2012, pp. 14–19.

[3] J. A. Rodríguez-Serrano and F. Perronnin, "Handwritten word-spotting using hidden Markov models and universal vocabularies," *Pattern Recognition*, vol. 42, pp. 2106–2116, September 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1542560.1542881

[4] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A Novel Word Spotting Method Based on Recurrent Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 211 –224, Feb. 2012.

[5] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, and Y. Kessentini, "A deep hmm model for multiple keywords spotting in handwritten documents," *Pattern Analysis and Applications*, vol. 18, no. 4, pp. 1003–1015, 2015.

[6] J. Puigcerver, A. H. Toselli, and E. Vidal, "Probabilistic interpretation and improvements to the hmm-filler for handwritten keyword spotting," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, Aug 2015, pp. 731–735.

[7] A. H. Toselli, E. Vidal, V. Romero, and V. Frinken, "Word-graph based keyword spotting and indexing of handwritten document images," Universidad Politcnica de Valencia, Tech. Rep., 2013.

[8] A. H. Toselli, J. Puigcerver, and E. Vidal, "Context-aware lattice based filler approach for key word spotting in handwritten documents," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, Aug 2015, pp. 736–740.

[9] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[10] F. Wessel, R. Schluter, K. Macherey, and H. Ney, "Confidence measures for large vocabulary continuous speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 288–298, mar 2001.

[11] D. Knuth, J. Morris, Jr., and V. Pratt, "Fast pattern matching in strings," *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977. [Online]. Available: http://dx.doi.org/10.1137/0206024

[12] T. J. Hazen, W. Shen, and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," in *Automatic Speech Recognition Understanding, 2009. ASRU 2009. IEEE Workshop on*, Nov 2009, pp. 421–426.

[13] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[14] S. Robertson, "A new interpretation of average precision," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR. New York, NY, USA: ACM, 2008, pp. 689–690. [Online]. Available: http://doi.acm.org/10.1145/1390334.1390453

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

[16] S. Johansson, G. N. Leech, and H. Goodluck, *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers*, Department of English, University of Oslo, 1978.

[17] R. Kneser and H. Ney, "Improved backing-off for N-gram language modeling," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 1995, pp. 181–184.

[18] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book: Hidden Markov Models Toolkit V2.1*, Cambridge Research Laboratory Ltd, Mar. 1997.

[19] D. Martn-Albo, V. Romero, and E. Vidal, "An experimental study of pruning techniques in handwritten text recognition systems," in *IbPRIA 2013: 6th Iberian Conference on Pattern Recognition and Image Analysis*. Springer Berlin Heidelberg, 2013, pp. 559–566, c.

[20] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "HMM-based Word Spotting in Handwritten Documents Using Subword Models," in *20th International Conference on Pattern Recognition (ICPR)*, Aug. 2010, pp. 3416 –3419.

[21] A. H. Toselli and E. Vidal, "Fast HMM-Filler approach for Key Word Spotting in Handwritten Documents," in *Proc. of the Intl. Conf. on Document Analysis and Recognition (ICDAR'13)*, Washington, DC, USA, Aug. 2013.

[22] A. Fischer, V. Frinken, H. Bunke, and C. Suen, "Improving HMM-Based Keyword Spotting with Character Language Models," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, Aug 2013, pp. 506–510.