HMM Word Graph Based Keyword Spotting in Handwritten Document Images

Alejandro Héctor Toselli[a,*], Enrique Vidal[a], Verónica Romero[a], Volkmar Frinken[b]

[a] *Universitat Politècnica de València, Camino de Vera S/N - 46022 Valencia, Spain*
[b] *Faculty of Information Science and Electrical Engineering, Kyushu University, Japan*
Electrical and Computer Engineering, University of California, Davis, CA, USA
ONU Technology Inc., San Jose, CA, USA

Abstract

Line-level keyword spotting (KWS) is presented on the basis of frame-level word posterior probabilities. These posteriors are obtained using word graphs derived from the recognition process of a full-fledged handwritten text recognizer based on hidden Markov models and N-gram language models. This approach has several advantages. First, since it uses a holistic, segmentation-free technology, it does not require any kind of word or character segmentation. Second, the use of language models allows the context of each spotted word to be taken into account, thereby considerably increasing ~~the~~ KWS accuracy. And third, the proposed KWS scores are based on true posterior probabilities, ~~computed~~ taking into account all (or most) possible word segmentations of the input image. These scores are properly bounded and normalized. This mathematically clean formulation lends itself to smooth, threshold-based keyword queries which, in turn, permit comfortable trade-offs between search precision and recall. Experiments are carried out on several historic collections of handwritten text images, as well as ~~with~~ a well-known dataset of modern English handwritten text. According to the empirical results, the proposed approach achieves KWS results comparable to those obtained with the recently-introduced "BLSTM neural networks KWS" approach and clearly outperform the popular, state-of-the-art "Filler HMM" KWS method. Overall, the results clearly support all the above-claimed advantages of the proposed approach.

Keywords:
keyword spotting, handwritten text recognition, word graph, posterior probability, confidence score.

*Corresponding author
Email address: `ahector@prhlt.upv.es` (Alejandro Héctor Toselli)

1. Introduction

Large amounts of handwritten text have been produced over the centuries. ~~In fact it has been claimed that the total accumulated amount of handwritten text is most likely larger than the nowadays available amount of (original) printed text available today, including modern, digitally-born text.~~ In recent years, large quantities of these documents have been scanned and made available as digital images via web pages of libraries and archives all over the world. Despite these efforts to make the documents accessible, raw images are largely useless for their primary purpose, ~~; namely~~ exploiting the wealth of information given in the text of the document images. Consequently, there is a fast-growing interest in automated methods that allow ~~the~~ users to search these images for arbitrary textual information.

In order to use conventional text information retrieval approaches, a first necessary step would be to convert the text images into digital text. However, OCR technology is completely useless for typical handwritten text images, and current handwritten text recognition (HTR) technology is still far ~~away~~ from offering sufficiently-accurate transcripts for the type of (historical) document considered in this work. This renders exact searches impossible.

~~This situation has brought about the need for search approaches that are specifically-designed for large text image collections. Obviously exact searching is not possible in this case.~~

Approximate keyword searching should follow a *precision-recall trade-off model*, based on word confidence measures. For each query, users should be able to specify whether they need *high precision*, to ensure that most of the spotting results do correspond to the query keyword, or a *high recall*, to make sure that most of the instances of the query keyword are retrieved. This entails the use of confidence thresholds, which can be specified more or less explicitly, depending on the application. For instance, in cases where the spotting results are provided in the form of ranked lists, the threshold is indirectly defined by the size of the list.

This confidence-based query model cannot be properly implemented straight-forwardly using conventional textual information retrieval methods on the noisy output of an automatic HTR system. Instead, recognition techniques ~~are needed~~ which assign confidence scores to alternative word recognition hypotheses are needed. Keyword spotting (KWS) [32, 45, 6, 53, 46, 13, 16, 14, 17, 72] has long been seen as an adequate way to obtain the required word confidence scores, and is ~~thereby~~ often considered an alternative to full HTR. However, this view of KWS and HTR as *competing* technologies is disputable. As we will see in Section 3, any word recognition process relies more or less explicitly on computing word confidence scores (posterior probabilities) and, equally, any method that computes KWS scores always entails some form of word recognition. Accordingly, HTR and KWS should be seen as *complementary* rather than competing technologies. This is ~~in fact~~ the viewpoint adopted in this paper.

KWS and information retrieval (IR) are also related technologies. Both terms are used more or less interchangeably in many papers in the document image literature, while others propose different, often subtly opposite, points of view on this relationship. For instance, in [32] KWS is presented as a technique for "obtaining the index" needed for IR, while in [6] KWS is considered to be the process of "keyword retrieval in document images". Moreover, some authors define KWS as a *way* to do things, rather than as a *process* or ~~a~~ *function* to be accomplished. For instance, in [45] KWS is understood as a way to compute confidence scores "by matching word images with each other". Here we ~~rather~~ prefer a *functional* definition, where KWS is any process which provides confidence scores suitable for searching and retrieving words in document images. That is, we in part share the view of [32] in that document indexing is based on the scores obtained by KWS, but diverge from that presented in [45], in seeing KWS as a function or process, rather than as a ~~specific~~ way to accomplish the intended goal of obtaining the word confidence scores needed in the *precision-recall trade-off model*.

In the field of automatic speech recognition (ASR), a field closely-related to HTR~~)~~, early works [10, 22, 48, 30] also considered KWS as an alternative to full ASR. The close relationship between full recognition and KWS soon became commonly acknowledged, and more recent works [56, 7, 8, 9] do indeed capitalize on the complementarity of ASR and KWS. On the other hand, as a result of work inspired by a series of competitions organized by ~~the~~ NIST [7, 8, 9, 69], the relationship between IR and KWS[1] is also becoming increasingly clear and commonly acknowledged in the field of ASR. Our standpoints on the relationships between HTR, KWS, and IR match the trends reflected in the ASR papers mentioned above.

Based on the above discussion, we propose to obtain KWS confidence scores through word

[1]After NIST competitions, ASR KWS is often referred to as "Spoken Document Retrieval" (SDR) and/or "Spoken Term Detection" (STD).

graphs (WG) produced by a full-fledged statistical HTR system, based on stochastic *optical character models*, as well as probabilistic *lexicon models* and *language models*. This entails several important advantages: a) The proposed formal framework is mathematically sound and computationally tractable and does not rely on heuristics; b) While most KWS techniques proposed so far are based on a *single* (given or computed) word segmentation of the given text image, our proposed framework is holistic and does not require any kind of word or character segmentation. In fact, our KWS scores are true posterior probabilities, computed by taking into account *all* (or *most*) of ~~)~~ the possible word segmentations; c) Since these KWS scores are well-defined and properly normalized, they do not need further normalization heuristics to achieve smooth precision-recall trade-offs in practical use; and d) ~~and~~ most importantly, it allows us to take advantage of word contexts, which significantly boosts KWS performance with respect to systems which are linguistically and contextually agnostic.

These convenient features require linguistic resources, in particular a lexicon which, for smaller applications, may constitute a limitation of the proposed approach. Yet, our target applications are those involving large handwritten collections, where the effort or cost to produce these resources will be more than offset by the benefits of accurately making the textual contents of these collections available for exploration and retrieval.

Traditional work on handwritten KWS assumed prior segmentation of the text images into words (c.f. Sec. 3). However, this word pre-segmentation is ~~plainly~~ impossible for the millions of historical handwritten documents and, even in favorable cases, it is quite prone to errors [40, 33, 31] which tend to significantly hinder overall KWS performance [1]. To overcome this considerable drawback, recent works [58, 26, 17, 16, 14, 72, 52] assume the ~~(word-unsegmented)~~ *line image* as the lowest search level, without any further segmentation into words. This is a convenient setting because, in most cases of interest, text images can be fully-automatically segmented into lines with fair accuracy [40, 5], and lines are sufficiently precise ~~target image positions~~ for most practical document image search and retrieval applications.

In this paper we also consider ~~(word-unsegmented)~~ line level KWS and present experiments to compare our proposed approach with other recently proposed approaches ~~which also work~~ at the line level. Results obtained with four different data sets ~~do~~ support the interest of WG-based KWS, where the WGs are provided by holistic HTR using language models to leverage contextual word information.

The rest of this paper is organized as follows: Section 2 outlines two reference systems with which we compare our results. Section 3 proposes *a general probabilistic KWS framework*, which is the basis of the KWS approach proposed in this paper, and properly accounts for most of the other KWS methods proposed so far. Section 4 quickly reviews the basic concepts of HTR based on hidden Markov models. Section 5 provides *a simple and compact formulation of word graph concepts, properties and computational procedures* ~~(~~needed for WG-based KWS~~)~~. To our knowledge, this is the first time such a review, short but comprehensive, is given. Section 6 presents *the proposed line-level KWS approaches based on WGs*. Section 7 reports *empirical results showing the interest and capabilities of these approaches,* and in Section 8 we draw some main conclusions and discuss related future lines of research.

The main contributions of this paper, emphasized in *italics* above, are outlined in Sections 3, 5, 6 and 7.

2. Reference Systems

In this paper we compare our results with those achieved using the approach presented in [14]. Since it is based on exactly the same optical character HMMs as in the approaches proposed here, accurate comparisons can be made. We also consider the less comparable ~~neural-networks based~~ work presented in [17], which is based on neural networks and has been proven to yield excellent results on various datasets.

2.1. HMM Filler

In this approach, character HMMs are used to build both a *"filler" model* and a *word-specific* model. The idea of using word-specific and a filler – or "garbage" – models was proposed a long time ago for KWS in the field of ASR and ~~it~~ has been in use for many years in this field [24, 30, 56]. More recently, the same idea was applied to KWS in handwritten images [14, 59, 72].

In this work, we have adopted the system proposed in [14] as a reference for the following reasons: a) over recent decades, the filler model has become ~~a~~ consolidated as a commonly-accepted, state-of-the-art KWS technique both for spoken signals and text images; b) as in our approach, it is based on trained character HMMs, which allows a fair comparison using identical pre-processing, feature extraction, and HMM training procedures; and c) again as in our approach, it performs KWS at the text line level.

Following [14], let F be the filler HMM and K_v be a keyword HMM for the word v. F is built by arranging all the trained character HMMs in parallel with a back-loop, enabling recognition of any unrestricted sequence of characters. In addition, for each individual keyword v to be spotted, K_v is built to model the exact character sequence of v, surrounded by the space character and the same unrestricted character sequences modeled by F.

In a *preparatory* phase, F is used to perform a unique handwriting decoding process, for each text line image $\mathbf{x}$ (cf. Sect. 4.1), yielding a decoding log-likelihood score $\log p_f(\mathbf{x})$. In the ~~actual~~ *search* phase ~~itself~~, for each keyword v, K_v is similarly used for each text line image to compute the decoding log-likelihood score $\log p_v(\mathbf{x})$. The spotting score $S'(v, \mathbf{x})$ is then defined as:

$$S'(v, \mathbf{x}) \ = \ \frac{\log p_v(\mathbf{x}) - \log p_f(\mathbf{x})}{L_v} \tag{1}$$

where L_v is the length of v in number of frames between the detected word borders. The restricted decoding score $\log p_v(\mathbf{x})$ is bounded above by the unrestricted one $\log p_f(\mathbf{x})$, hence $S'(v, \mathbf{x})$ is non-positive, and higher values are expected if the spotted word v is contained in $\mathbf{x}$.

The preparatory and search ~~decoding~~ processes are done using the Viterbi algorithm [23]. The computational cost of Viterbi decoding with both the filler and the keyword-specific HMMs is $\gamma' \cdot n$, where n is the length of $\mathbf{x}$, and γ' is a constant which depends on the total number of HMM states and on the square of the number of character models in F [23]. Therefore, the total cost of the *preparatory phase* is $\gamma' \cdot n \cdot N$, where N is the number of line images in the collection. In the *search phase*, the asymptotic complexity of each query is again $\gamma' \cdot n \cdot N$, leading to a total cost of $\gamma' \cdot n \cdot N \cdot M$, where M is the number of spotted keywords.

Finally, character HMMs are trained using the Baum-Welch re-estimation algorithm [23]. The computational cost of this algorithm is essentially linear in the number and ~~the~~ length of training lines, with a relatively low constant which depends on the number of character HMMs and ~~on~~ their topology.

2.2. *Neural Network Based BLSTM*

We also consider the neural network-based keyword spotting method proposed in [17]. It, too, performs KWS at the text line level and ~~it~~ has been shown to outperform the HMM filler approach and many other KWS methods on a number of datasets. There are important differences between the BLSTM-based and the HMM-based approaches, including training computational costs. In this paper, BLSTM is considered as an upper reference, assuming training computational costs are not taken into account.

BLSTM-based KWS is a two-step process. In the first *preparatory* step, a recurrent neural network transforms a sequence of input features into a sequence of character posterior probability vectors called "character activations". Then, dynamic programming is used to compute the spotting score for a given keyword , represented as a character sequence. In step one, the system uses the Bidirectional Long-Short Term Memory ~~(BLSTM)~~ neural network introduced for handwriting recognition in [21]. The sequence is processed in both directions, through separate hidden layers into a common output layer. The hidden layer is recurrent and made up of so-called LSTM nodes, which can be seen as differentiable memory cells. By having a recurrent connection weight of 1, and separate nodes to control the information flow, this architecture does not suffer from the vanishing gradient problem often encountered when training recurrent neural networks [18].

The computational time needed to generate the character activations from an input sequence of length n is $\Gamma''\cdot n$, where Γ'' is a constant proportional to $I \cdot H + H^2 + H \cdot O$ and I, H, and O are the sizes of the input, the hidden and the output layers, respectively. Therefore, for a collection of N text line images, the computational time needed by the *preparatory* step is $\Gamma''\cdot n \cdot N$.

For keyword spotting, the character probability sequence is extended by an additional entry with a constant value of 1. By adding this symbol at the beginning and at the end of the keyword, a dynamic programming procedure similar to Viterbi decoding is able to efficiently find the best path through the output matrix, passing through the symbol added at the beginning, ~~then~~ through all the characters of the keyword itself, and then through the symbol added at the end. In other words, the path traverses ~~through~~ the letters of the keyword at their most likely position, while the rest of the text line has no influence. This way, we get a keyword spotting score, $s_c(v, \mathbf{x})$, that reflects the product of all character probabilities at the optimal sub-sequence that starts with the space before the first character of the keyword and ends with the space after its last character. Likewise, to obtain a normalized value which can be thresholded, the logarithm of $s_c(v, \mathbf{x})$ is taken and divided by the length L''_v of the spotted word v in number of characters:

$$S''(v, \mathbf{x}) \;=\; \frac{\log s_c(v, \mathbf{x})}{L''_v} \qquad (2)$$

The runtime of the dynamic programming step to compute $s_c(v, \mathbf{x})$ is $\gamma'' \cdot n$, where n is the length of the sequence and γ'' is proportional to the number of characters of the keyword. Therefore, in the *search* phase, the computational time needed to perform M queries on a collection of N text line images is $\gamma'' \cdot n \cdot N \cdot M$.

Finally, BLSTM models are ~~essentially~~ trained with back-propagation [21] which, as in the case of HMMs, entails a computational cost which is essentially linear in the number and length of training text lines. However, in this case the constant is very large, which makes the BLSTM training cost very much larger than that of other more traditional approaches.

3. A General KWS Probabilistic Framework

Since the KWS approach presented here is *line-based*, our search domain consists of a set of text line images. The goal is to determine whether a given keyword is or is not present in each text line, regardless of the number of occurrences.

We start by defining a *word posterior probability at frame level* to account for the degree of uncertainty about the presence of a given keyword in a specific horizontal position within a text line image, represented by ~~a~~ feature vector sequence $\mathbf{x} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ (see Sec. 4.1). Given $\mathbf{x}$ and a horizontal position i, ~~(~~the index of a feature vector or *"frame"* of $\mathbf{x}$~~)~~, we define $P(v \mid i, \mathbf{x})$ as the probability that the word v appears in some horizontal segment of $\mathbf{x}$ that contains i. Considering the contribution of all intervals containing i, this probability can be computed as:

$$P(v \mid i, \mathbf{x}) \;=\; \sum_{\substack{k,l:\\ 1\leq k\leq i\leq l\leq n}} P(v, k, l \mid i, \mathbf{x}) \tag{3}$$

where $[k, l] \subseteq [1, n]$ is a frame interval of $\mathbf{x}$ and $P(v, k, l \mid i, \mathbf{x})$ is the probability that $[k, l]$ contains the frame i *and* the segment of $\mathbf{x}$ defined by this interval corresponds to the word v. By rearranging the sum and applying Bayes' rule, we can write:

$$P(v \mid i, \mathbf{x}) \;= \sum_{k=1}^{i} \sum_{l=i}^{n} P(k, l \mid i, \mathbf{x}) \cdot P(v \mid k, l, i, \mathbf{x}) \tag{4}$$

And assuming, for simplicity, that $P(k, l \mid i, \mathbf{x})$ is uniform among all the intervals $[k, l]$ containing i:

$$P(v \mid i, \mathbf{x}) \;\approx\; \mathcal{K} \sum_{k=1}^{i} \sum_{l=i}^{n} P(v \mid k, l, \mathbf{x})\,; \qquad \mathcal{K} \;=\; \frac{1}{i \cdot (n - i + 1)} \tag{5}$$

Given a keyword v to be spotted, KWS could be naively addressed ~~in a naive way~~ by computing ~~the~~ frame-level word posterior probabilities $P(v \mid i, \mathbf{x})$ for each $\mathbf{x}$ and i, and marking as candidate hits those positions for which $P(v \mid i, \mathbf{x})$ is greater than a given threshold τ. By varying τ, adequate *precision-recall* trade-offs could be achieved.

However, since we aim for line-level KWS, we need a line-level global measure that scores the degree of presence of a keyword in each text line, without considering any specific position within the line image. To this end, we adopt a *confidence score* $S(v, \mathbf{x})$, based on $P(v \mid i, \mathbf{x})$, as:

$$S(v, \mathbf{x}) \overset{\text{def}}{=} \max_i P(v \mid i, \mathbf{x}) \ . \tag{6}$$

Line-level KWS is then performed by computing $S(v, \mathbf{x})$ for each line image, $\mathbf{x}$, and marking as candidate hits those lines whose scores are greater than a given threshold τ.

It can be argued that this way of combining frame-level word posteriors is overly simplistic and can fail to adequately cope with several instances of the same word ~~appearing~~ in a line image. However, it has the advantage of being well normalized and bounded in the $[0, 1]$ range and can be properly interpreted in probabilistic terms. Moreover, even in the (uncommon) case[2]of repeated word instances, it can be shown that it is in fact a good approximation to the true probability that v is written in $\mathbf{x}$. Similar score definitions have already been proposed and used as good heuristics to obtain word and sentence recognition confidence measures in ASR [70, 54], machine translation [67] and HTR [57].

It is worth noting that $P(v \mid k, l, \mathbf{x})$ in Eq. (5) is the exact word posterior probability ~~which is~~ implicitly or explicitly used by any system capable of recognizing a presegmented word image, i.e. a segment of $\mathbf{x}$ between fixed positions k and l~~)~~. Therefore, the computation of Eq. (5) essentially entails a simple *sliding window* process, where the required word posteriors, $P(v \mid k, l, \mathbf{x})$, can be provided by any isolated word recognizer. Of course, the better the recognizer, the better the resulting frame-level word posterior estimate. We should consider, however, that the computational complexity of directly computing Eq. (5) for all the frames of $\mathbf{x}$ is exceedingly high: at least $\Omega(n^4)$ ~~or quartic~~ with the length of $\mathbf{x}$.

All of the early KWS techniques relying on pre-segmented word images circumvented this prohibitive cost by reducing the summation in Eq. (5) to just one fixed ~~, given~~ interval. Clearly, this makes it possible to reduce the computational cost to $O(n \cdot m)$, where m is the average

[2]Repetitions of interesting words within a line are rare. For example, in the IAM dataset (see Sec. 7.2), the distribution of counts of repeated keywords (excluding stop words and punctuation marks) per line has a mean of 1.02 ± 0.13; and only about 1.5% of the lines contain one or more repeated words.

length of the word image segments, at the expense of ~~previously~~ requiring precise image word segmentation in advance, which is generally difficult. More recent works, such as [46], rely on automatic over-segmentation of the text images to mitigate the impact of word segmentation errors. Such techniques can be seen as heuristic approximations to the marginalization in Eq. (5). Finally, in the case of KWS approaches which work with completely unsegmented line images, the summation in Eq. (5) is more or less implicitly approximated by the dominating addend only, ~~(~~ which is typically a good approximation ~~)~~. Then, dynamic programming techniques are used to avoid repeated computations during the sliding window process. This is specifically the case of segmentation-free dynamic time warping KWS methods such as [58, 26], as well as all the modern techniques based on HMMs [59, 14, 72] and recurrent neural networks [17].

In this work, we show (in Sec. 6) that the full summation over all (or most) segment intervals required to accurately obtain $P(v \mid i, \mathbf{x})$ as in Eq. (5), can be easily and efficiently computed ~~by~~ using state-of-the-art segmentation-free HTR technology based on optical HMMs and N-gram language models. This kind of HTR recognizer can obtain not only the best word sequence and its corresponding unique segmentation hypothesis for a given line image $\mathbf{x}$, but also a word graph (WG) representing a huge set of best hypotheses, including the corresponding segmentations and likelihoods. As we will see, such a WG can be considered ~~as~~ a fair representation of the text image itself, in that it contains all the probabilistic and segmental information required for the fast computation of Eq. (5). One remarkable advantage of this approach is that it allows the use of multiple contextual knowledge sources (optical shape, lexicon, syntax, etc.) in a fully holistic, albeit simple, way. This generally results in more discriminating word posterior probabilities than those provided by a context-agnostic word recognizer, thereby allowing us to obtain what are ultimately better KWS scores.

The following two sections review the fundamental concepts of HTR based on HMMs and N-grams, as well as all ~~the~~ details on the WGs ~~which are~~ needed for the proposed KWS approach.

4. HTR based on HMMs and N-Grams

The fundamentals of HTR based on HMMs and N-grams were originally presented in [2] and further developed in [68, 63], among other works. Recognizers of this kind accept a given handwritten text line image, represented as a sequence of feature vectors, $\mathbf{x}$, and find a most likely word sequence, $\widehat{\mathbf{w}} = \widehat{w}_1 \widehat{w}_2 \ldots \widehat{w}_l$, according to

$$\widehat{\mathbf{w}} \;=\; \arg\max_{\mathbf{w}} P(\mathbf{w} \mid \mathbf{x}) \;=\; \arg\max_{\mathbf{w}} p(\mathbf{x}, \mathbf{w}) \;=\; \arg\max_{\mathbf{w}} p(\mathbf{x} \mid \mathbf{w}) \cdot P(\mathbf{w}) \,. \qquad (7)$$

The conditional density $p(\mathbf{x} \mid \mathbf{w})$ is approximated by optical word models, built by concatenating character HMMs, and the prior $P(\mathbf{w})$ is approximated using an N-gram language model [23].[3]

The decoding problem of Eq. (7) can be adequately approached with the Viterbi algorithm [23]. Several variations of the basic algorithm and acceleration techniques have been developed which allow ~~to implement~~ the decoding process to be implemented in a very time- and memory-efficient way, typically entailing computational costs linear in the length of $\mathbf{x}$. Moreover, rather than obtaining just one single-best solution to Eq. (7), a huge set of almost-best solutions can be obtained in the form of a WG as a byproduct of this process, as discussed in Sec. 5.

4.1. System Architecture, Preprocessing, Feature Extraction, Training and Model Integration

Text line images constitute the basic input of the HTR process. They can be obtained from each document image by means of conventional text line detection and segmentation techniques [29].

Two main modules make up the HTR process [63]: preprocessing and feature extraction, and decoding. Preprocessing performs style attribute normalizations, such as slant and slope correction, and size normalization. For details we refer to [63, 49]. Feature extraction, on the other hand, processes each normalized line image and represents it as a sequence $\mathbf{x}$ of feature vectors, where each vector $\vec{x}_i$, $1 \leq i \leq n = |\mathbf{x}|$ describes grey-level and/or other geometric features of a narrow vertical window at uniformly spaced horizontal line positions. Details of the different feature extraction techniques will be given in Sec. 7. The modeling scheme encompasses three levels. *Character* shapes are represented by optical character HMMs. Each *lexical* entry (*word*) is modeled by an adequate composition of character HMMs which represents the possible concatenations of individual characters to form the word. Finally, text line *word sequences* are modeled using an N-gram *language model*.

Character HMM parameters are trained from unsegmented handwritten text images represented by feature vector sequences, along with the transcription of these images into the corresponding sequence of characters, according to the corresponding lexical models of the words. This

[3] In practice, these two models are not used directly; $P(\mathbf{w})$ is affected by an exponent ϕ, called *grammar scale factor*, and a length-dependent factor $|\mathbf{w}|^{\lambda}$ is included in the product of Eq. (7), where λ is called *"word insertion penalty"*. Both ϕ and λ are tuned empirically to better balance the contribution of these two models.

training process is carried out using a well-known instance of the EM algorithm called forward-backward or Baum-Welch re-estimation [23]. As previously commented in Sec. 2.1, this training process is exactly the same as that needed for the Filler-HMM approach. ~~discussed in Sec.2.1, with identical (relatively low) computational cost, which essentially grows linearly in the number and length of training line images.~~ *Lexical models* are straightforwardly built according to word spellings obtained from the transcripts of the training text images and/or from an adequate dictionary. Finally, N-gram *language model* parameters are estimated from the training transcripts ~~(~~and/or from adequate external texts~~)~~ and usually smoothed using the Kneser-Ney back-off technique [25]. The computational costs of lexicon building and N-gram training are completely negligible with respect to HMM training costs. All ~~these (~~character, lexicon, and language~~)~~ finite-state models can be easily *integrated* into a single *global* model on which the search process of Eq. (7) is performed to decode $\mathbf{x}$ into either an optimal output word sequence $\widehat{\mathbf{w}}$, or a huge set of optimal sequences represented as a WG.

5. Word Graphs

Word graphs were first proposed by several authors some decades ago during the development of ASR technology [37]. A WG is a labeled, weighted directed acyclic graph (WDAG) in which the edges are labeled with words and weighted with scores, and the nodes hold word segmentation boundaries. These scores and boundaries are derived from probabilities and alignments computed during the line image decoding process. This data structure allows the representation of a huge number of most-likely word sequence and segmentation hypotheses in a compact and convenient way. Fig. 1 shows a small, illustrative example of a *normalized* (see below) WG obtained from the decoding of a text line image, also shown in this figure.

5.1. WG Definition and Properties

Let $\mathbf{x}$ be a feature vector sequence of length n. A WG of $\mathbf{x}$ is a direct acyclic graph defined by a finite set of *nodes* Q and another finite set of *edges* $E \subset (Q - F) \times (Q - q_I)$, where $q_I \in Q$ is a special *initial node* and $F \subseteq (Q - q_I)$ is a set of *final nodes*. An edge is denoted by its departing and ending nodes $(q', q) \in E$. A *position function*, $t : Q \to \{0, 1, \dots, n\}$, associates each node with a frame of the input sequence $\mathbf{x}$. It must fulfill: $t(q_I) = 0$, $t(q') < t(q) \;\; \forall (q', q) \in E$,

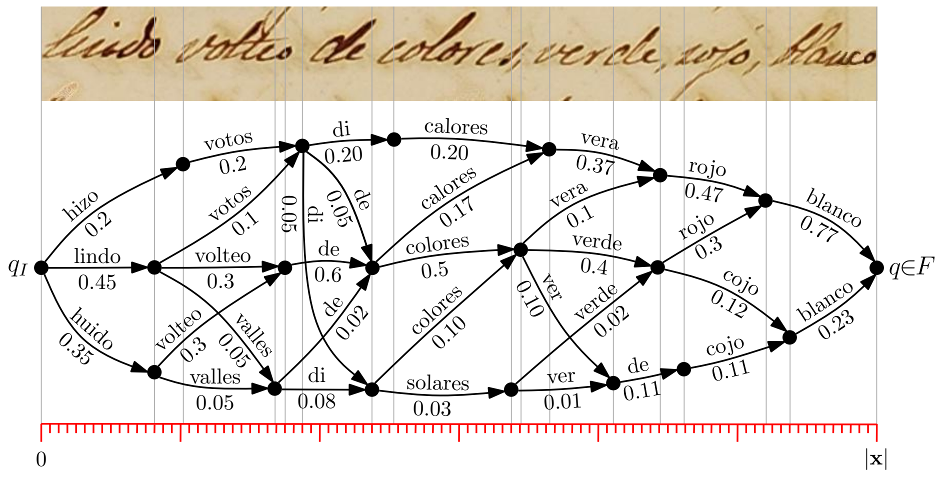


Figure 1: Illustrative, simplified example of a *normalized* WG that would be obtained by decoding a line image, $\mathbf{x}$, with the Spanish handwritten text: "lindo volteo de colores, verde, rojo, blanco", represented by its sequence of feature vectors of length $n = |\mathbf{x}|$. Each edge (q', q) is labeled with the corresponding word, $\omega(q', q)$, and weighted with its "edge posterior", $\varphi(q', q)$. Node positions, $t(q)$, delimiting word alignments, are also indicated on the bottom ruler.

$t(q) = n \; \forall q \in F$. Each edge (q', q) has an associated ~~a~~ *word*, denoted as $\omega(q', q)$, and a *score*, denoted as $s(q', q)$. See Fig. 1 for an illustration of these concepts.

The score $s(q', q)$ represents the likelihood of the hypothesis that the word $\omega(q', q)$ appears between frames $t(q') + 1$ and $t(q)$. More specifically, it is the product of the two elementary, i.e. word-level, optical and LM probabilities[4] needed to compute the respective factors of Eq. (7).

A *complete path* of a WG is a sequence of nodes starting with q_I and ending with a node in F. Complete paths correspond to whole -line decoding hypotheses. The WGs considered in this work are unambiguous; that is, no two complete paths may correspond to the same word sequence $\mathbf{w}$. A path's score is the product of the scores of all the edges making up the path. Thus, for a given word sequence $\mathbf{w}$, the joint probability $p(\mathbf{x}, \mathbf{w})$ can be approximately computed from a WG

[4]In practice, log-probabilities are used with an arbitrary logarithm base. Changing this base for a different one, b, affects the impact of the different WG path scores in the computation of $p_G(\mathbf{x})$ in Eq. (9). It also affects the actual values of $P_G(\mathbf{w} \mid \mathbf{x})$ computed in Eq. (13). While these changes do not affect the best (or the N-best order of) the word sequence(s) in the WG, some KWS performance improvements can be achieved by adequately tuning b empirically.

G as:

$$p(\mathbf{x}, \mathbf{w}) \;\approx \prod_{(q',q)\in\psi(\mathbf{w})} s(q', q) \;\stackrel{\text{def}}{=}\; p_G(\mathbf{x}, \mathbf{w}) \tag{8}$$

where $\psi(\mathbf{w})$ denotes the set of edges of the (unique) complete path in G, such that $\mathbf{w}$ is the sequence of words associated with the edges of this path. ~~On the other hand, a~~ A WG typically contains the majority of the most probable decoding hypotheses considered in the maximization of Eq. (7), including the best hypothesis. Therefore, the unconditional likelihood of $\mathbf{x}$ can be approximated by the total accumulated score (joint probability) of the paths corresponding to all possible word sequence hypotheses represented in G:

$$p(\mathbf{x}) \;\approx\; \sum_{\mathbf{w}} p_G(\mathbf{x}, \mathbf{w}) \;\stackrel{\text{def}}{=}\; p_G(\mathbf{x}) \tag{9}$$

Although, for useful WGs, this sum typically has a prohibitively large number of addends, it can be very efficiently computed using dynamic programming by means of recursively-computed *forward* (α) or *backward* (β) accumulated path scores [70]:

$$p_G(\mathbf{x}) \;=\; \beta(q_I) \;=\; \sum_{q\in F} \alpha(q) \tag{10}$$

where:

$$\alpha(q) \;=\; \sum_{q':(q',q)\in E} \alpha(q') \cdot s(q', q) \quad \text{if } q \neq q_I; \qquad \alpha(q_I) \;=\; 1 \tag{11}$$

$$\beta(q) \;=\; \sum_{q'':(q,q'')\in E} s(q, q'') \cdot \beta(q'') \quad \text{if } q \notin F; \qquad \beta(q) \;=\; 1 \quad \forall q \in F \tag{12}$$

Finally, word sequence posteriors $P(\mathbf{w} \mid \mathbf{x})$ can be approximately computed as:

$$P(\mathbf{w} \mid \mathbf{x}) \;\approx\; \frac{p_G(\mathbf{x}, \mathbf{w})}{p_G(\mathbf{x})} \;\stackrel{\text{def}}{=}\; P_G(\mathbf{w} \mid \mathbf{x}) \tag{13}$$

Most WG extracting algorithms, ~~(~~such as that of the *Hidden Markov Model Toolkit (HTK)* [74], used in this work~~)~~, have a parameter to specify the maximum input degree of each WG node. This allows control over the amount of information retained on which words end at each node [71] ~~is cotrolled~~. In addition, pruning techniques, such as *beam search*, can be applied to accelerate the Viterbi search. Using these pruning criteria, the size of the generated WGs and, therefore, the amount of best hypotheses finally represented can be adequately tuned.

5.2. *WG Normalization and "Edge Posteriors"*

The scores, or likelihoods, of the edges of a WG can be "normalized" in several ways. For the purposes of this paper, we need edge scores to be normalized in such a way that they are useful to obtain the frame-level word posterior probabilities $P(v \mid i, \mathbf{x})$ discussed in Sec. 3. To this end, for a given edge (q', q), its normalized score $\varphi(q', q)$, often called "edge posterior" [70], is defined as the sum of the posterior probabilities of all the complete decoding hypotheses whose paths pass through (q', q). That is:

$$\varphi(q', q) \overset{\text{def}}{=} \sum_{\mathbf{w}:(q',q)\in\psi(\mathbf{w})} P_G(\mathbf{w} \mid \mathbf{x}) \tag{14}$$

For a WG normalized in this way, the following properties can be shown (see proofs in Appendix I and illustrations in Fig. 1):

Efficient, dynamic programming computation [70]:

$$\varphi(q', q) = \frac{\alpha(q') \cdot s(q', q) \cdot \beta(q)}{\beta(q_I)} \tag{15}$$

Nodes are flow preserving:

$$\sum_{q':(q',q)\in E} \varphi(q', q) \;=\; \sum_{q'':(q,q'')\in E} \varphi(q, q''), \quad \forall q \in Q \tag{16}$$

Edge posteriors are consistent at frame-level:

$$\sum_{\substack{(q',q)\in E:\\ t(q')<i\le t(q)}} \varphi(q', q) \;=\; 1, \quad 1 \le i \le n \tag{17}$$

5.3. *Computational Cost of WG Generation and Normalization*

WGs are obtained as a by-product of Viterbi decoding by storing the best HMM and language model scores for a number of partial hypotheses. Then, all the paths starting from initial states and reaching a final state with sufficiently large total score are added to the graph [39, 70].

It is well-known that the computational complexity of the Viterbi algorithm is linear in the length of $\mathbf{x}$, and the cost can be made independent of the lexicon size and the overall size of the models used by means of pruning techniques [23]. However, when WG generation is also included in this process, the computational complexity of the whole decoding process is observed to grow very fast with the size of the resulting WG (exponentially in the WG *density*, according to [55]). Overall, the asymptotic cost of generating a WG for a line image of length n can be expressed as $\Gamma \cdot n$, where Γ is a (generally large) constant which is dependent upon the WG size. Nevertheless, we should remember that this process is carried out only once and that, by choosing adequate WG sizes, reasonable processing times ~~for WG generation~~ can be achieved for WG generation in practice.

The WG normalization process, on the other hand, entails an extra computational cost, which is also linear in n. However, according to [39], as well as ~~to~~ our own observations (c.f. Sec. 6.3), this cost is negligible.

6. Line-Level Keyword Spotting Using Word Graphs

We continue now with the presentation, begun in Sec. 3, of our proposed approach to line-level KWS based on WGs. Following Eq. (6), we explain below how very accurate frame-level word posteriors can be easily and efficiently obtained from WGs.

6.1. Computing Frame-Level Word Posteriors from WGs

Following essentially the same arguments as in the derivation of Eqs. (3–5) in Sect. 3, the frame-level word posterior $P(v \mid i, \mathbf{x})$ can be obtained by considering the contribution of all the WG edges labeled with v which correspond to segmentation boundaries that include the frame i:

$$P(v \mid i, \mathbf{x}) \;\approx\; \sum_{\substack{(q',q)\in E:\\ v=\omega(q',q),\\ t(q')<i\leq t(q)}} \varphi(q',q) \;\overset{\text{def}}{=}\; P_G(v \mid i, \mathbf{x}) \tag{18}$$

From Eq. (17), this is a properly defined word posterior; that is, $\sum_v P_G(v \mid i, \mathbf{x}) = 1$ for all i.

This computation can be straightforwardly carried out by ~~sequentially~~ visiting the WG edges in sequence and updating, for each edge (q', q), the values ~~of~~ $P_G(v \mid i, \mathbf{x})$ for $v = \omega(q', q)$ and for all i comprised between $t(q')$ and $t(q)$. Obviously, the computational time required for this process is proportional to the number of edges of the WG and to the average length $t(q') - t(q)$ of the segment of $\mathbf{x}$ defined by an edge (q', q). This cost can also be more conveniently given as $\Theta(\kappa \cdot n)$, where n is the length of $\mathbf{x}$. This follows from the observation that, for each WG word v

and frame i, the value of $P_G(v \mid i, \mathbf{x})$ has to be updated for a relatively small number κ_i of edges (q', q), such that $t(q') < i \leq t(q)$. The average value of this number, κ, depends on the size of the WG.

6.2. Computing Line Confidence Scores from WGs

According to the definition given by Eq. (6) and ~~to~~ the WG-based computation of frame-level word posteriors given by Eq. (18), the confidence score $S(v, \mathbf{x})$ is finally computed as:

$$S(v, \mathbf{x}) = \max_i P_G(v \mid i, \mathbf{x}) \tag{19}$$

6.3. Overall Computational Cost of the Proposed KWS Approach

As in the case of the reference systems discussed in Sec. 2, two processing phases are distinguished: the *preparatory* phase, which involves all the processes which do not depend on specific query words, and the *search* phase, which corresponds to actually locating each query word v in the plain text line-level word score lists computed from Eq. (19).

In the *preparatory* phase, WGs are generated and normalized. As discussed in Sec. 5.3, this requires $\Gamma \cdot n$ computational time for each text line image of length n, where Γ is a (generally large) constant which is dependent upon the average WG size. In addition, the line-level scores $S(v, \mathbf{x})$ have to be computed for each line image, according to Eqs. (18–19). As discussed above, the computational time of Eq. (18), and hence Eq. (19), is also linear in n, with a relatively small constant which is typically negligible with respect to Γ. Therefore, for a collection of N line images, the overall preparatory computational time is $\Gamma \cdot n \cdot N$. As we have already commented, this cost is heavily dominated by the cost of generating the WGs[5].

In the *search* phase, if line-level confidence score lists are used directly, the search cost is obviously $\gamma \cdot N$, where γ is ~~a~~ typically a small constant dependent upon the average sizes of the confidence score lists (which ultimately depend on ~~the~~ average WG sizes). Overall, to spot M query words on the whole collection of N line images, $\gamma \cdot N \cdot M$ computational time is required.

[5]For example, the time needed to generate a typical, large WG of the IAMDB corpus (see Table 1) was about 6.5 minutes. In contrast, the corresponding combined time needed for a) WG normalization and computing the *edge posteriors* (Eqs. 10–15), b) computing the *frame-level posteriors* (Eq. 18) and c) computing the final *line-level confidence scores* (Eq. 19), was about 0.3 seconds, less than 0.1% of the time needed to generate the WG.

As compared with the reference systems outlined in Sec. 2, the proposed approach has a similar asymptotic computational times trend; namely, linear in the number of line images N and queries M. Since the relative actual cost of the different approaches will depend on constants which are difficult to compare analytically, real computational times, including training times, have been determined empirically and will be reported in Sec. 7.8.

7. Experiments and Results

The empirical work carried out to assess the effectiveness of the proposed approach is presented in this section.

7.1. Experimental Framework

Three main sets of experiments were carried out. In the first set, baseline results were established using the KWS reference system described in Sec. 2.1, which will be referred to as "Filler-HMM". In addition, results using the BLSTM approach outlined in Sec. 2.2 are also reported. KWS BLSTM is known to significantly outperform Filler-HMM (and many other KWS techniques) [17]. However, the computational cost for training and validating BLSTM models can be orders of magnitude higher than for HMMs. Because of this, among other important differences between BLSTM- and HMM-based approaches, BLSTM is considered here only as an upper reference, assuming model training costs are ignored.

In the second set, the proposed WG-based KWS approach was tested *without* using a language model for WG generation. Since this setting uses the same character-level knowledge sources as in Filler-HMM (trained character HMMs), the benefits of the proposed confidence scores, based on frame-level word posteriors computed with the help of a lexicon for most possible word intervals, can be fairly assessed. This setting will be referred to as "Plain-WG".

In the the third set, finally, the proposed approach was tested using WGs produced by means of a full-fledged HTR system, including a bi-gram language model. This setting, named "LM-WG", allows us to study the KWS performance gain that can be achieved by adding word-contextual information.

7.2. Corpora

Two main data sets, referred to as IAMDB and CS were used in the experiments. Basic information about these data sets, along with the corresponding standard partitions for empirical evaluation, are given hereafter. Details about the use of these data in this work are given in Sec. 7.4.

IAMDB is a publicly available, well-known modern English handwritten text corpus compiled by the FKI-IAM Research Group on the basis of the Lancaster-Oslo/Bergen electronic text corpus (LOB). The last released version (3.0) consists 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 [36] is used here. Statistics of the IAMDB corpus appear in Table 1. The "text data" information for this corpus refers to three external text corpora (LOB, Brown and Wellington, collectively called "LBW") which were used to compile the lexicon and ~~for~~ train the IAMDB bi-gram language model [3] used in the LM-Plain and LM-WG settings. The OOV (out-of-vocabulary) row shows the number~~s~~ of words of the test and validation partitions which do not appear in the training partition.

CS ("Cristo Salvador") is a 19th-century Spanish manuscript, made up of 50 color images of relevant text pages written by a single writer. Line images were extracted from the original page images in previous works [50]. The CS corpus, along with directions for its use in HTR experiments, is publicly available for research purposes[6]. The first 29 pages (675 lines) are used for training (and validation) and the test set comprises the remaining 21 pages (497 lines). The text ground truth of CS is case-insensitive. Statistics of this dataset are also shown in Table 1.

Both IAMDB and CS were used in the main experiments reported in Sec. 7.6. In addition, in Sec. 7.9 we will report the results obtained using the techniques proposed here with two other corpora, Parzival [15] and George Washington (GW) [28], also used frequently for KWS experimentation.

7.3. *Keyword Selection*

Several criteria can be adopted for the selection of the keywords to be used in KWS assessment experiments. Clearly, any given KWS system may perform better or worse depending on the query words it is tested with and how these words are distributed in the test set. In general, the larger the set of keywords, the more reliable the empirical results. In accordance with these observations, in this work we adopt the same criterion as in [17], where all the words seen in the training partition

[6]The CS corpus can be downloaded from `https://prhlt.iti.upv.es/page/data`

Table 1: The IAMDB and CS datasets and their corresponding partitions. The IAMDB Running Words, Lexicon and OOV (out of vocabulary) figures labeled "Text data" correspond to the external text corpora (LBW) used to train the bi-gram language model. For CS, these figures correspond only to the reference transcripts of the CS text images.

		IAMDB				CS		
		Training	Validation	Test	Total	Training	Test	Total
Image data	Total number of chars	269 270	39 318	39 130	347 718	35 176	25 819	60 995
	Number of diff. chars	72	69	65	81	53	52	54
	Running words	53 765	8 599	8 315	70 679	6 227	4 691	10 918
	Lines	6 161	920	929	8 010	675	497	1 172
Text data	Running words	3 128 155	8 599	8 315	3 145 069	6 227	4 691	10 918
	Lexicon size	19 892	2 450	2 492	20 773	2 474	1 879	3 805
	OOV running words (%)	0.00	6.12	6.27	0.00	0.00	29.03	0.00

are tested as keywords. This is exactly true for CS but, following [17], no *stop words* are included in the IAMDB keyword list. Table 2 shows basic information about the keyword sets used in the two corpora considered (see more details in [66]).

Note that, by selecting the keywords in this way, there may be many keywords which do not actually appear in any of the test images. We say that these keywords are *non-relevant*, while the remaining ones are *relevant*. In both datasets, the amount of relevant words is less than one third of the total number of selected keywords. Clearly, spotting non-relevant words is also challenging, since the system may erroneously find other similar words, thereby leading to important precision degradations. Overall, the selected keywords constitute rather challenging sets.

In line-level KWS experiments, rather than the number of line images (N) or keywords (M), it is more informative to consider the total number *query events*; that is, the number $N \cdot M$ of pairs composed of an image and a keyword. A query event $(\mathbf{x}, v)$ is *relevant* if v is relevant for $\mathbf{x}$, i.e. the keyword v is actually written in the image $\mathbf{x}$. According to Table 2, from the large amount of query events in each dataset, only a few are actually *relevant*, even though most of the IAMDB test images, and all of the CS test images, are relevant to at least one of the selected keywords.

7.4. Systems Setup

In general, the basic system architecture outlined in Sec. 4.1 is used in all ~~the~~ experiments using character HMMs, viz. *HMM-Filler, Plain-WG,* and *LM-WG*. However, ~~some~~ details of line image pre-processing, writing style attribute normalization, and feature extraction usually adopted

Table 2: Sizes and other related information for the keyword sets selected for IAMDB and CS. A line image is considered "relevant" if it contains at least one of the M words of the corresponding Keywords set.

	IAMDB		CS	
	Total	Relevant	Total	Relevant
Line images (N)	929	855	497	497
Keywords (M)	3 421	1 098	2 236	620
Query events ($N \cdot M$)	3 178 109	1 916	1 111 292	3 005

for each of the two corpora are different.

The features extracted for the IAMDB consist of 9-dimensional geometric features computed for each pixel column of the line images [35]. On the other hand, the 60-dimensional features extracted for CS ~~which are also well-known,~~ consist of raw grey-level values, and grey-level horizontal and vertical gradient components, averaged over a horizontal sliding image window [60].

Regarding the optical models, character HMMs were trained from the corresponding training partitions of each corpus. A left-to-right HMM was trained for each of the elements appearing in the training images, such as lowercase and uppercase letters, punctuation marks, special symbols, possible inter-word spaces, etc. In the case of CS, a case-less topology was trained, that is, each character was modeled "in parallel" by two HMM, one for each corresponding lower- and uppercase glyph. The same character HMMs were used for both, the Filler-HMM reference system and ~~for~~ the WG-based approaches proposed here.

~~On the other hand,~~ For the *Plain-WG* and *LM-WG* approaches, the training partition transcripts of CS were used to build the lexicon and ~~to~~ train the language model, ignoring letter cases, punctuation signs and diacritics. The lexicon and bi-gram language model of IAMDB were directly obtained using the training partition of the external LBW text data [3].

The HTK toolkit [74] was used to obtain two kinds of WG for each test line image, one using the bi-gram LM for the *LM-WG* and one without any language model but the same vocabulary for the *Plain-WG* approach. The WGs were normalized as explained in Sec. 5 and then frame-level word posterior probabilities $P_G(v \mid i, \mathbf{x})$ were computed. Finally, the line-level word confidence scores, $S(v, \mathbf{x})$, were obtained as described in Sec. 6.2. Table 3 shows some statistics of the resulting WGs.

In the case of *LM-WG*, three different sets of WGs were produced, both for CS and IAMDB, by setting the HTK parameter which specifies the maximum node input degree (NID) [74] to 40, 5

and 1. The latter value yields degenerated WGs which represent solely the single best recognized hypotheses (see Tab. 3).

Table 3: IAMDB and CS statistics of the Plain-WG and bi-gram LM-WG word graphs for different node input degree values (NID). All the figures are numbers of elements, averaged over all the generated WGs.

Corpus	Type	NID	Nodes	Edges	Words	Paths
IAMDB	Plain-WG (no LM)	40	745	23 061	410	$\sim 10^{26}$
	LM-WG (bi-grams)	40	876	22 080	493	$\sim 10^{21}$
		5	76	297	61	$\sim 10^{8}$
		1	9	8	7	1
CS	Plain WG (no LM)	40	7 060	252 958	473	$\sim 10^{30}$
	LM-WG (bi-grams)	40	7 230	258 243	498	$\sim 10^{28}$
		5	169	765	56	$\sim 10^{8}$
		1	10	9	8	1

Meta-parameters of the line-image preprocessing, feature extraction, HMMs, and N-grams, as well as the log-base parameter, b, for WG normalization (Sec. 5.1), were optimized through cross-validation on the training data for CS and on the validation data for IAMDB. More details about all these settings can be found in [66, 17].

For the BLSTM system, exactly the same setup and networks as in [17] were adopted here for the IAMDB corpus. Using the feature vector sequences of the training text line images and their corresponding transcripts, 75 neural networks were trained with 100 LSTM nodes each in both hidden layers, using a learning rate of 10^{-4} and a momentum of 0.9. Training regularization implicitly consisted in initializing all weights to random values, with a mean of 0 and a standard deviation of 0.1, and restricting the error gradient for each LSTM weight to ~~be~~ within the interval $[-1 : 1]$. These fixed parameters were tested in previous experiments and found to be optimal for the task of text line transcription. The stopping criterion was the label error rate of the text lines of the validation set.

~~Essentially identical,~~ The training procedure ~~was~~ carried out for the CS corpus was essentially identical. However, since CS data are smaller and more regular, only 10 neural networks were trained in this case. Finally, to single out the best network for each corpus, BLSTM keyword spotting was performed on the IAMDB validation set and on the CS training set. These two networks were then used to obtain results on the corresponding test partitions.

According to Eqs. (1) and (2), the line spotting scores $S'(v,\mathbf{x})$ and $S''(v,\mathbf{x})$, cannot be directly interpreted in probabilistic terms and their (negative) ranges are unbounded. While this is not a problem for obtaining the precision and recall results needed to be able to compare these systems with ours, it may raise practical issues when trying to regulate the score threshold, τ, ~~in order~~ to meet real-life search requirements. In the results presented below, it is clear that this problem ~~clearly~~ shows up whenever performance is plotted as a function of τ (Fig. 3). In order to mitigate this problem, and to allow better comparison of the different approaches in Fig. 3, the ranges of the original scores $S'(v,\mathbf{x})$ and $S''(v,\mathbf{x})$ were mapped to the $[0,1]$ interval as follows:

$$S_F(v,\mathbf{x}) \;=\; \exp\left(\eta' \cdot S'(v,\mathbf{x})\right), \qquad S_B(v,\mathbf{x}) \;=\; \exp\left(\eta'' \cdot S''(v,\mathbf{x})\right) \tag{20}$$

where the scale parameters η' and η'' were tuned in order to make $F_1(\tau)$, the F_1-measure (see Eq. (23)) as a function of τ in a wide interval around $\tau \approx 0.5$ as flat as possible, (shown in Fig. 3). Of course, since these mappings increase monotonically, they do not affect the precision-recall or max-F_1 results. ~~, which are in fact all independent of η' or η''.~~

7.5. *KWS Evaluation Measures*

KWS effectiveness is assessed by means of the standard *recall* and *precision* measures. Let r be the total number of *relevant events* (c.f. Sec. 7.3) and, for a fixed *search threshold*, τ, let $d(\tau)$ be the number of events *detected*, or retrieved as relevant by the system and $h(\tau)$ be the number of *hits*, or correctly detected events. *Recall*, $\rho(\tau)$, and *precision*, $\pi(\tau)$, are defined as:

$$\rho(\tau) \;=\; \frac{h(\tau)}{r}, \qquad \pi(\tau) \;=\; \frac{h(\tau)}{d(\tau)} \tag{21}$$

The ~~interrelated~~ trade-off between recall and precision can be conveniently displayed as a ~~so-called~~ *recall-precision* (R-P) curve, $\pi(\rho)$ [11]. Any KWS system should allow users to (more or less explicitly) regulate the search threshold in order to choose the precision-recall operating point which is most appropriate for each query. Of course, good systems should achieve both high precision and high recall for a wide range of values of τ.

In Eq. (21), precision can become undefined and, moreover, ~~it~~ can fail to exhibit the ~~typicaly expected~~ concavely-decreasing curve typically expected for increasing recall values [11]. To overcome these problems, the ~~so-called~~ *interpolated precision*, π', is often used. In [34], it is defined as the highest precision value found for any recall ~~value~~ $\rho' \geq \rho$:

$$\pi'(\rho) = \max_{\rho' : \rho' \geq \rho} \pi(\rho') \tag{22}$$

This definition makes $\pi'(\rho)$ a well-defined, monotonically-decreasing function, even for $d(\tau) = 0$ (and null recall).

To assess the overall behavior of a search and retrieval system as an explicit function of the search threshold τ, the *harmonic mean* of precision and recall, called F_1*-measure,* is often used:

$$F_1(\tau) = 2 \cdot \frac{\pi'(\tau) \cdot \rho(\tau)}{\pi'(\tau) + \rho(\tau)} \tag{23}$$

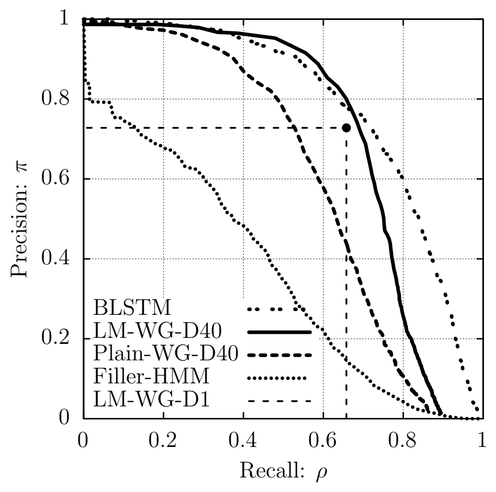
Finally, to summarize how well a KWS system can perform without referring to any specific value of τ, some scalar assessment measures are often used. The most simple is what is known as *R-precision* (RP), which is the precision (or recall) such that $\pi'(\rho) = \rho$. A better scalar KWS measure is based on the commonly-accepted fact that the better the KWS algorithm performs, the larger is the area under the corresponding R-P curve, which is expressed by the *average precision* (AP) [75].

AP should not be confused with the *mean AP* (mAP), which is the average over all the queries of the individual AP value computed for each query. While mAP is often reported in KWS papers, in this work it can not be computed because it becomes undefined if non-relevant keywords are used, as it is the case in our choice for keyword selection (see Sec. 7.3).

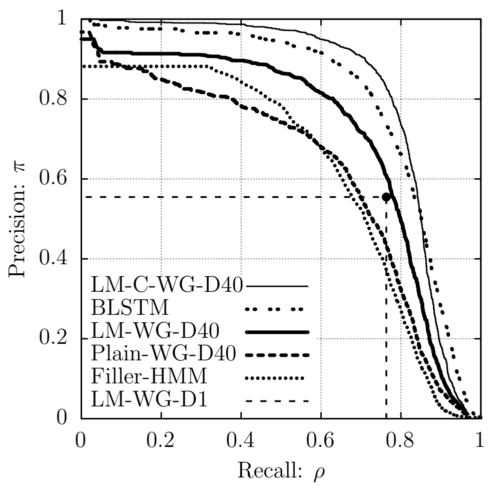
7.6. Main Results

Interpolated R-P curves were obtained for both the IAMDB and CS corpora presented in Sec 7.2. Results are shown in Fig. 2 for the settings discussed in Sec. 7.1.

In IAMDB, both WG-based KWS approaches very significantly outperform the Filler-HMM reference system. On the other hand, by leveraging word contexts by means of a bi-gram LM, the LM-WG-D40 approach performs significantly better than Plain-WG-D40,~~(~~ which uses no LM~~)~~. For a wide, useful range of recall ($0.3 \lesssim \rho \lesssim 0.7$), LM-WG-D40 also achieves precision values slightly higher than those of the reference BLSTM system, yet precision is significantly lower in the high-recall range. The *recall-precision* point corresponding to LM-WG-D1 was also included in the plots of Fig. 2 for comparison purposes. In fact, this is equivalent to directly searching the query words ~~just~~ in the HTR single-best ASCII transcription only.



(a) IAMDB Corpus



(b) CS Corpus

Figure 2: KWS *Recall-Precision* curves (with interpolated precision) using BLSTM, LM-WG-D40 (with bi-grams and NID = 40), LM-C-WG-D40 (with closed-vocabulary bi-grams and NID = 40), Plain-WG-D40 (NID = 40), Filler-HMM, and LM-WG-D1 (with bi-grams and NID = 1) for the IAMBD and CS datasets (LM-C-WG-D40 only for CS).

For the CS dataset, LM-WG-D40 again outperforms Filler-HMM significantly but the precision achieved by LM-WG-D40 falls short of that obtained by the BLSTM system for the whole recall range. In our interpretation, this relative degradation with respect to BLSTM KWS is mainly caused by the large OOV rate of 29.0% in CS, compared with 6.3% in IAMDB (see Tab. 1).

To support this interpretation, an additional experimental setting was considered, called LM-C-WG-D40. It is identical to LM-WG-D40, except a *closed vocabulary* was used. While the bi-gram LM was trained only on the training partition text (as in LM-WG-D40), this time all the test-set OOV words were added to the lexicon. This yielded the thin-line curve shown in Fig. 2(b). Clearly, while a huge external text dataset was used to build the IAMDB lexicon, no such a resource was readily available for (the kind of historical text in) CS and only the very few words seen in the few transcripts of the training set images were used in this case. On the other hand, since the BLSTM approach does not use a lexicon, this problem is not encountered. For a real-life application, it would not be difficult to collect electronic text corresponding to a language usage similar to that found in CS (19th-century Spanish) and we expect this would bring LM-WG-D40 results closer to those of LM-C-WG-D40.

Fig. 3 shows the $F_1(\tau)$ curves for the different KWS approaches and corpora. The relative performance levels are similar to those shown in the precision-recall curves of Fig. 2. As in Fig. 2(b), the thin curve of Fig. 3(b) corresponds to the use of a closed vocabulary in LM-WG-D40 and illustrates how well LM-WG-D40 might behave if a very good lexicon were available.

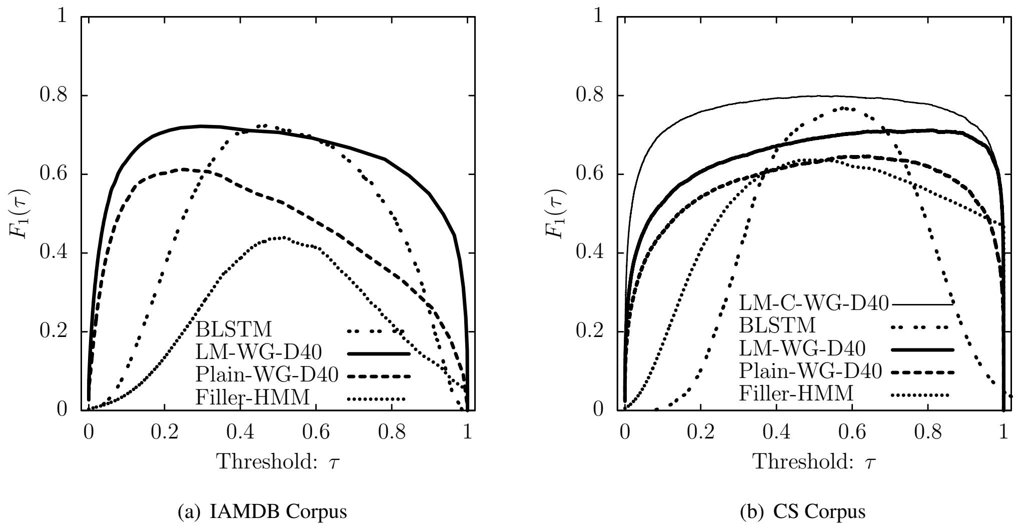


Figure 3: KWS F_1-*measure* (F_1) curves using BLSTM, LM-WG-D40, LM-C-WG-D40 (closed-vocabulary), Plain-WG-D40 (NID=40) and Filler-HMM for the IAMBD and CS datasets (LM-C-WG-D40 only for CS). All LM-WG curves correspond to WGs obtained with bi-grams and NID=40. To obtain the Filler-HMM and BLSTM curves, ~~respectively,~~ the following scaling factors were used (see Eq. (20)): $\eta' = 1$ and $\eta'' = 3$ for IAMDB; $\eta' = 1/3$ and $\eta'' = 1/3$ for CS.

Table 4 summarizes the overall KWS performance of all the approaches tested in this work, including the closed vocabulary LM-C-WG-D40 for CS, expressed in terms of *average-precision* (AP), *R-precision* (RP) and maximum F_1-*measure* (F_1^*). Results using WGs with NID values of 40, 5 and 1 are included for the LM-WG setting. 95% confidence intervals, computed for all the AP results (except those of BLSTM) using the bootstrap method proposed in [4], were all smaller than ± 0.03. It is worth noting that no significant differences in KWS performance are observed between NID values 40 and 5, though AP results degrade dramatically for the degenerate WGs with NID=1. However, as shown in Tab. 3, WGs with NID=5 are almost two orders of magnitude smaller than those with NID=40, leading to total computing costs for NID=5 that are less than half of those incurred with NID=40 (see Tab. 5). In [62, 61], we study in depth how ~~the~~ KWS performance and computational costs are affected by varying NID values.

Table 4: Average Precision (AP), R-Precision (RP) and maximum F$_1$-measure (F_1^*), considering different input degree values (IND), corresponding to the (bi-gram) LM-WG, Plain-WG, Filler-HMM and BLSTM KWS approaches for the IAMDB and CS corpora. Results for the closed-vocabulary bi-gram LM (LM-C-WG) are also shown for CS. 95% confidence intervals, computed for the AP results (except for those of IAMDB), were all smaller than ± 0.03.

	IND	IAMDB			CS		
		AP	RP	F_1^*	AP	RP	F_1^*
BLSTM (best on validation set)	–	0.78	0.72	0.73	0.80	0.75	0.77
LM-C-WG (closed-voc bi-grams)	40	–	–	–	0.82	0.78	0.80
LM-WG (bi-grams)	40	0.72	0.70	0.72	0.71	0.70	0.71
	5	0.71	0.69	0.72	0.71	0.70	0.71
	1	0.48	0.66	0.69	0.42	0.76	0.64
Plain-WG (no LM)	40	0.60	0.59	0.61	0.62	0.64	0.65
Filler-HMM	–	0.36	0.44	0.44	0.62	0.63	0.64

7.7. *Qualitative Analysis of Spotting Results*

It is worth noting the relatively low precision achieved by Filler-HMM for low recall, which can be clearly observed in the R-P curves in Fig. 2. An analysis of this behavior reveals that it is mainly caused by *false positives*, most of which correspond to short query words that are substrings of other possible query words. Fig. 4-left shows one of these false positives, where the query word “ways” is incorrectly spotted by Filler-HMM, with confidence 1, in a text line image which instead contains the word “always”. In contrast, both WG-based confidence scores are significantly lower than 1. In particular, LM-WG would never spot the keyword “ways” in this line image unless an extremely low confidence threshold $\tau < 3 \cdot 10^{-9}$ is specified.

On the other hand, Fig. 4-right illustrates another (less frequent) example where Filler-HMM (and also Plain-WG) work as expected, but LM-WG fails to produce the desired results. The uncapitalized keyword $v =$“senior” is spotted by LM-WG with $S_L(v, \mathbf{x}) = 0.89$ in a line image which contains the capitalized word “Senior” instead.

It is illustrative to analyze how the bi-gram LM leads to the above LM-WG behaviors. In the first example, $P($“been” | “always”$) = 0.08$ and $P($“always” | “has”$) = 0.01$. In contrast, both $P($“been” | “ways”$)$ and the bi-gram probabilities of plausible word prefixes for “ways”, such as “at” or “all”, are all lower than 10^{-4}. In the second example, both “Senior” and “senior” have similarly high HMM likelihoods, and $P($“Officers” | “senior”$) = 0.005$, $P($“senior” | “squadron”$) = 0.014$.

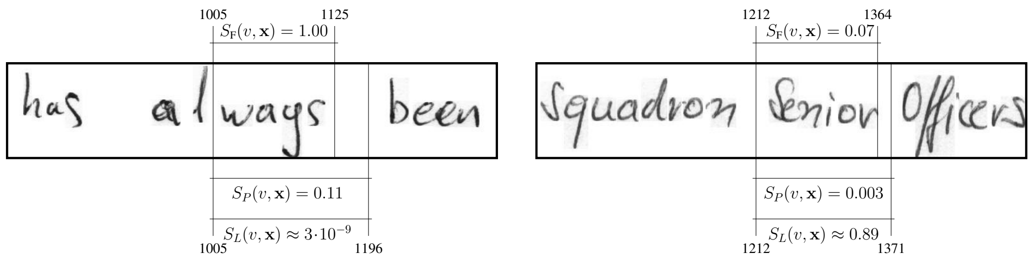

1005
1125
$S_F(v,\mathbf{x}) = 1.00$
$S_P(v,\mathbf{x}) = 0.11$
$S_L(v,\mathbf{x}) \approx 3{\cdot}10^{-9}$
1005
1196
1212
1364
$S_F(v,\mathbf{x}) = 0.07$
$S_P(v,\mathbf{x}) = 0.003$
$S_L(v,\mathbf{x}) \approx 0.89$
1212
1371

Figure 4: Left: Example of a Filler-HMM *false positive* keyword detection where the query word v =“ways” is incorrectly spotted with full confidence ($S_F(v, \mathbf{x}) = 1$). The confidence scores $S_L(v, \mathbf{x})$ and $S_P(v, \mathbf{x})$ of ~~to~~ the two WG-based approaches (with and without using a bi-gram LM, respectively), as well as the corresponding spotting boundaries, are also shown. Right: example of a LM-WG *false positive* keyword detection where the uncapitalized query word v =“senior” is incorrectly spotted with high confidence ($S_L(v, \mathbf{x}) = 0.89$).

However all the (smoothed) LM probabilities involving “Senior” are very small.

To finish this section, Fig. 5 illustrates how a normalized WG is used to obtain the word confidence scores for a line image from CS containing the handwritten text “tiempos modernos a las generaciones de soldados ...” ~~(“modern times to the generations of soldiers ...” in English)~~. It also shows another example of how the LM-WG approach steps on the linguistic context to boost its discriminative behaviour. The best confidence score is $S(v_1, \mathbf{x}) = 0.99$ and corresponds to the word v_1 = “generaciones”. The next best scores are for the (also plausible) words v_2 =“generacion”, v_3 =“general” and v_4 =“racion” ~~(“portion” in English)~~: $S(v_2, \mathbf{x}) = 2.72 \cdot 10^{-6}$, $S(v_3, \mathbf{x}) = 1.69 \cdot 10^{-10}$ and $S(v_5, \mathbf{x}) = 8.73 \cdot 10^{-16}$, all of which are orders of magnitude lower than the first (and correct) one. It is interesting to note that both v_2 and v_4 are (close to) sub-words of “generaciones”. However, since they have very low confidence scores, they will not be spotted in this line ~~(as an image part of “generaciones”)~~, unless extremely low confidence thresholds are used. Such a behaviour is clearly achieved thanks to the lingustic context captured in the bi-gram LM. The probabilities of the bi-grams “las *generaciones*” and “*generaciones* de” are high. In contrast, the probabilities of other possible bi-grams involving the words “*general*” and “*racion*”, such as “las *general*”, “*general* cimas”, “*general* cosas”, “gema *racion*”, “genio *racion*”, “*racion* es”, etc., are low or very low. We see the discriminating behaviour achieved in this way as an important advantage of the proposed, context-aware approach with respect to many other KWS methods based on pattern matching which do not take context into account.

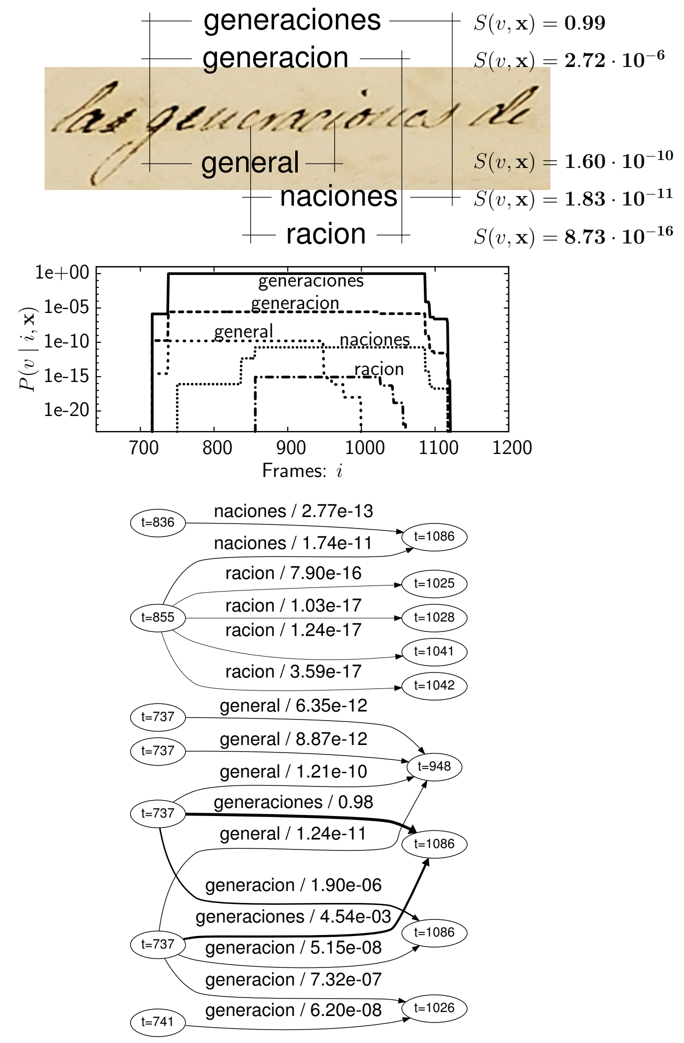


Figure 5: The word confidence score, $S(v, \mathbf{x})$, for several words. Bottom: a pruned, small part of a normalized WG of the ~~given~~ image (arrows thickness represents edge posterior levels). Middle: a plot of the frame-level word posteriors, $P_G(v \mid i, \mathbf{x})$, computed from the WG. Top: some spotting hypotheses, along with their corresponding confidence scores and spotting boundaries. The correct word is highly discriminated with respect to competing hypotheses, thanks in part to the bi-gram context provided by the LM used to obtain the WG.

7.8. *Computing Efficiency*

Table 5 presents the computing times for BLSTM, Filler-HMM, LM-WG-D40, and LM-WG-D5 KWS on the IAMDB corpus. Although experiments were run on several different computers, reported times are approximately scaled ~~so as~~ to correspond to the *elapsed* times ~~which would~~

~~be needed~~ on a single dedicated core of a 64-bit Intel® Core Quad® CPU running at 2.83GHz. Each approach has its ~~own~~ bottleneck or dominating computational cost. First, the training and validation costs for BLSTM are by far the highest, at more than 6 years. As we have mentioned (Sec. 7.1), this can be prohibitive for many applications, unless previously trained models can be used or adapted [17]. Second, LM-WG has the highest preparatory time. According to Sec. 5.3, this time is predominantly accounted for by the cost of generating the WGs. Finally, Filler-HMM is by far the slowest approach in terms of query time.

Table 5: Approximate training, preparation and average query times for IAMDB using BLSTM, Filler-HMM and LM-WG for two NID values. The *Total Indexing Time* is the sum of the Training & Validation time, the Preparatory time and the *Average Time per Query* multiplied by the number of queries (3 421).

	BLSTM	Filler-HMM	LM-WG-D40	LM-WG-D5
Training & Validation Time (days)	2 258	0.35	0.35	0.35
Preparatory Time (hours)	0.95	1.12	102	38
Average Time per Query (seconds)	1.90	4 080	0.003	0.003
Total Indexing Time (days)	2 258	162	4.6	1.9
Average Precision (AP)	0.78	0.36	0.72	0.71

The extremely fast query performance of LM-WG is worth commenting on. Clearly, this is possible since the use of a lexicon allows the vast majority of the KWS work to be completed in the preparatory phase. Of course, if the query words are available beforehand, the line-level scores $S_B(v, \mathbf{x})$ and $S_F(v, \mathbf{x})$ can also be easily precomputed using BLSTM and Filler-HMM, respectively, much the same as $S_L(v, \mathbf{x})$ is pre-computed using LM-WG. Therefore, to allow for better, and fairer comparisons, the overall time needed to train the models, prepare the document image collection and perform all the required queries is also reported in the "Total Indexing Time" row of Tab. 5.

According to these overall computing times, and taking into account the corresponding KWS performance (AP), the LM-WG models clearly achieve the best cost/performance trade-off, even more so with the much smaller WGs produced with NID=5.

We should point out, however, that the bottleneck computational costs of the other approaches can be reduced to some extent. The dominant ~~query~~ cost of Filler-HMM (queries) can be easily reduced by more than one order of magnitude using a character-graph based technique we ~~have~~

introduced recently [65]. This would bring the overall computational cost (though not KWS accuracy) of Filler-HMM closer to that of the approaches proposed in this paper. Similarly, the very high training cost of BLSTM can be substantially reduced (perhaps by a factor of 5 or more) simply by reducing the number of networks actually trained. In these experiments, 75 randomly initialized networks were trained using back-propagation while observing the label error rate on the validation set as a stopping criterion. However, only the best network is used for KWS. This means, fewer networks can be trained, or poorly performing networks can be aborted at an early training stage, with an increased risk of not finding a very good one. Also, the back-propagation iterations can be limited at the cost of a less thoroughly trained network.

Finally, the space complexity is analyzed assuming the use of KWS for indexing purposes. In this case, the required storage space is similar for all the approaches: for each test line image, a list of words which might appear in the image, along with the corresponding KWS scores, must be stored[7] ~~perhaps using appropriate data structures for the sake of efficiency~~. For the WG-based approaches, using very large WGs affects the memory space temporally used in the preparatory step, but not significantly the storage required for indexing.

7.9. Additional Results and Comparisons

Supplementary KWS experiments have been carried out using the proposed approach on two additional corpora, PARZIVAL and GEORGE WASHINGTON (GW), on which several previous KWS studies have been reported.

PARZIVAL contains 45 digital images of a ~~medieval~~ manuscript from the 13th century written in Middle High German language [15]. Although written by several writers, all the writing styles found in this dataset are very similar. GW consists of 20 pages of letters written by George Washington and his associates in the year 1755. These 20 relatively clean pages, all of which exhibit a very similar writing style, have been selected from a larger collection of images [28]. Given the small size of the data set, and in line with previous works on this corpus, a four-fold cross validation is adopted for empirical evaluation.

The experimental setup established here for PARZIVAL and GW is essentially the same as that outlined in Sec. 7.4 for the IAMDB and CS corpora, with the exception of feature extraction,

[7]The time needed to build these lists is essentially the preparatory time, plus the time per query multiplied by the size of the indexing vocabulary..

and some ~~details of~~ training and parameter optimization details. Feature extraction for PARZIVAL and GW were carried out as described in [41]. The PARZIVAL bi-gram LM was trained using only the transcripts from its training and validation partitions. On the other hand, following [17], the GW bi-gram was obtained by combining a bi-gram trained from the GW training and validation transcripts with another bi-gram trained on the external LOB text corpus. Details about these settings, including ground-truth, corpus partitions, keyword list selection[8], etc., can be seen in [14, 41]. Finally, PARZIVAL and GW WGs were generated with a maximum node input degree of 40.

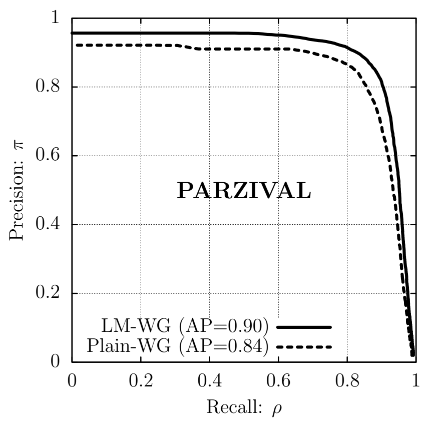
Our experimental results for these two datasets using the Plain-WG and (bi-gram) LM-WG KWS approaches are presented in Fig. 6, which shows the *Recall-Precision* curves and the corresponding AP figures.

By way of a rough comparison, we also outline here recently reported KWS results using the same IAM data set as in the main experiments reported in Sec. 7.6, but with different experimental setups. Only works following the same challenging *segmentation-free* and *query-by-string* (QbS) paradigms adopted here have been considered. In contrast with the main empirical work presented in Sec. 7.6, here we have not been able to re-run ~~ourselves~~ the experiments carried out by other authors. Therefore the experimental conditions may vary ~~(very)~~ significantly across the different works reported and care should be taken in any comparison.

Tab. 6 shows these results for the following approaches: *Bag of Features HMMs* (BoF HMMs) [51], *Dynamic Time Warping* (DTW) [17, 14, 47], *relevance-based language model* (RBLM) [44], semi-continuous HMMs (SC-HMMs) [47], *pyramidal histogram of characters* (PHOC) [20], *variational dynamic background model* (VDBM) [27] and *script-independent line-based spotting framework* (SILSF) [73]. The results can be loosely compared with those shown in Fig. 6. The entries of Table 6, marked with † ~~(~~ or ‡~~)~~ indicate that KWS performance is reported in terms of *mean average precision* (mAP), rather than the overall AP used in our results. Since mAP is only computed for *relevant* queries, mAP values tend to be optimistic with respect to AP, which significantly drops when false positives are produced for non-relevant queries.

Even taking into account the large variability in experimental conditions and evaluation pro-

[8]Standard PARZIVAL and GW partitions and keyword lists can be found here: `http://www.iam.unibe.ch/fki/databases/iam-historical-document-database`



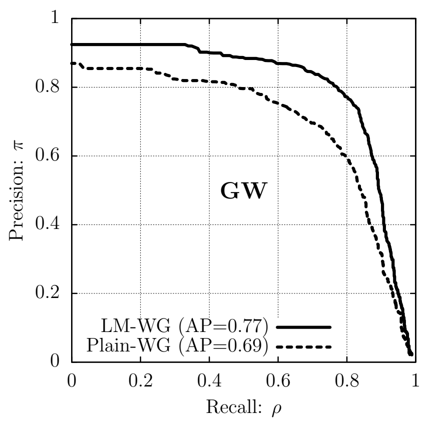


Figure 6: *Recall-Precision* curves and AP figures using bi-gram LM-WG and Plain-WG. The GW curves correspond to the second cross-validation partition.

Table 6: AP figures for different approaches on IAMDB, PARZIVAL and GW under very varied empirical setups. Values marked with † are *mean average precision* (mAP) figures, rather than plain overall AP, while the entry marked with ‡ is an average calculated from a range of computed precision values.

Ref.	Approach	IAMDB	PARZ	GW
[12]	**Classic Filler-HMM**	—	—	0.72
	2-gram Filler-HMM	0.55	—	0.74
[51]	**BoF HMM**	—	—	0.80 †
[19]	**Haar-Like-Features**	—	—	0.77 ‡
[17]	**BLSTM**	—	0.94	0.84
	Classic Filler-HMM	—	0.83	0.60
	DTW	—	0.37	0.48
[44]	**RBLM**	—	—	0.54 †
[47]	**SC-HMM**	—	—	0.53 †
	DTW	—	—	0.50 †
[14]	**Classic Filler-HMM**	—	0.86	0.62
	DTW	—	0.39	0.44
[20]	**PHOC**	0.39 †	—	0.64 †
[27]	**VDBM**	0.49 †	—	—
[73]	**SILSF**	0.58 †	—	—

tocols, ~~of~~ the results in Tab. 6 ~~they~~ clearly show that the approaches proposed in this paper can provide KWS performances that are much better than that of most other techniques proposed in ~~the~~ recent years, some of which are currently considered state-of-the-art.

8. Concluding Remarks and Future Work

In this work we have explored a novel keyword spotting (KWS) framework designed for searching ~~in~~ (large) collections of handwritten text images. Interesting features of this framework include:

- It properly supports *fast word queries* controlled by user-specified *confidence thresholds*.
- *Confidence scores* are based on *frame-level word posterior probabilities*, which are computed by taking into account the contribution of all (or most) word segmentations of the input image. Since the confidence scores are properly bounded and normalized, they need no further heuristics to allow comfortable adjustment of precision-recall trade-offs.
- The core *frame-level word posteriors* are ~~very~~ efficiently obtained from *word graphs* produced as a byproduct of processing unsegmented line images with a *full-fledged handwritten image recognizer*. This is a very versatile and well-understood framework.
- Since the recognizer is *holistic*, it does *not* require any kind of *word or character image segmentation*. This is essential because accurate pre-segmentation of handwritten text images into characters has proved extremely elusive.
- The recognizer relies on *character hidden Markov models*, a *lexicon* and an N*-gram language model*, all of which can be trained or straightforwardly derived from moderate amounts of training data. This allows for a simple, cost-effective adaptation of the system to new writing styles, alphabets and languages.
- *Character hidden Markov model training* is also *holistic*, i.e. it requires only very simple training data annotation. Only a literal text transcript of the training images is needed, without any kind of costly ground truth such as coordinates of word or character bounding boxes.
- The use of a *language model* allows easy *leverage of the word context* of the spotted words, which significantly *boosts spotting performance*. According to various standard metrics, this leads to performance that is significantly better than that of the well-known Filler-HMM approach, which is considered one of the state-of-the-art KWS techniques, and ~~is~~ comparable to that of BLSTM KWS, which is perhaps the best HTR KWS method currently available if the very high training costs are not taken into account.
- The overall *computational cost* is *much lower* than that of other KWS approaches considered state-of-the-art.

Of course, these advantages come at a price. In particular, the accuracy level depends on the quality of the lexicon and the language model. The lexicon is needed in order to specify, in terms of character strings, the word forms that can be searched for. A basic lexicon can be straightforwardly derived from the training transcripts, but both coverage and accuracy can be significantly improved if it is expanded by including other words ~~which are~~ expected to appear in the handwritten image collection being considered [38]. These words can be derived from similar texts and available vocabularies of the language and historical period of the collection. Similarly, a basic language model is automatically learned from the training transcripts. In this case, ~~accoring to~~ our experiments show that results significantly improve just by using a very basic, largely under-trained bi-gram language model. But again, even larger improvements are possible by augmenting this training text with other similar texts, where available. The demands of these linguistic resources may be difficult to meet in some smaller applications, but the benefits will certainly pay off in projects involving large handwritten text image collections.

The results of this paper, and the above demands raise a number of issues for future research. One of the most important issues is to explore adequate techniques to avoid the need for a large or specific lexicon. As an initial idea, words ~~which are~~ not in the lexicon can be searched for by relying on the confidence scores of *"similar"* words which are in the lexicon, and therefore in the WGs. A word similarity can be computed in terms of character edit distances, possibly weighted by estimated optical dissimilarity between character pairs. Work exploring this idea, along with the use the Filler-HMM model as a back-off method, is presented in [43].

A different alternative for coping with the lexicon requirements is to ignore the words altogether and work directly with "character graphs" produced using a full-fledged character-level handwritten recognizer. Such a pure character-level setting would bring us closer to the setting assumed in the successful work of [17], with the added benefits of being able to easily leverage word-like contextual information by means of high-order character N-grams and much lower training computational costs. First steps towards this goal are presented in [42, 64].

In addition to these research plans, perhaps the single most important issue for future development is to further develop the ideas underlying the neural network based BLSTM approach in the context of the word posterior probability-based KWS confidence scores introduced in this work. The results reported in this paper show that, by taking advantage of word context, e.g. by means of an N-gram LM, these confidence scores have the potential to significantly improve KWS performance. As published in [17], BLSTM provides the best KWS performance for handwritten images known so far, but ~~it~~ is essentially word-context agnostic. However, using the basic BLSTM technology, it has been shown that a full-fledged HTR decoder ~~can be implemented~~ which incorporates a lexicon and an N-gram language model [21] can be implemented. Therefore, an obvious next step is to extend this kind of system to allow it to adequately produce word graphs as a byproduct of decoding. In this way, all the WG-based techniques developed in this paper could be straightforwardly applied to further improve the already excellent BLSTM KWS results reported in [17]. From a practical point of view, additional work will be also needed to reduce the BLSTM training computational demands.

To close this paper, we would like to mention that some demonstration prototypes, directly based on the approaches described and tested in this paper, have been implemented and are publicly available at the *tranScriptorium* project web site[9].

Acknowledgments

This work has been partially supported bythe 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] G. R. Ball, S. N. Srihari, H. Srinivasan, Segmentation-Based And Segmentation-Free Methods for Spotting Handwritten Arabic Words, in: G. Lorette (ed.), Tenth International Workshop on Frontiers in Handwriting Recognition, Université de Rennes 1, Suvisoft, La Baule (France), 2006, 6 pages.

[2] I. Bazzi, R. Schwartz, J. Makhoul, An Omnifont Open-Vocabulary OCR System for English and Arabic, IEEE Trans. on Pattern Analysis and Machine Intell. 21 (6) (1999) 495–504.

[3] R. Bertolami, H. Bunke, Including language model information in the combination of handwritten text line recognisers, in: Proc. of the Int. Conference on Frontiers in Handwriting Recognition (ICFHR'08), 2008, 6 pages.

[4] M. Bisani, H. Ney, Bootstrap estimates for confidence intervals in asr performance evaluation, in: Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. (ICASSP'04), vol. 1, IEEE, 2004, 4 pages.

[9] `http://transcriptorium.eu/indexing-and-searching-based-on-keyword-spotting`

[5] V. Bosch, A. H. Toselli, E. Vidal, Statistical text line analysis in handwritten documents, in: Int. Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2012, pp. 201–206.

[6] H. Cao, A. Bhardwaj, V. Govindaraju, A probabilistic method for keyword retrieval in handwritten document images, Pattern Recognition 42 (12) (2009) 3374–3382.

[7] C. Chelba, J. Silva, A. Acero, Soft indexing of speech content for search in spoken documents, Computer Speech & Language 21 (3) (2007) 458 – 478.

[8] T. K. Chia, H. Li, H. T. Ng, A Statistical Language Modeling Approach to Lattice-Based Spoken Document Retrieval, in: Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '07), Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 810–818.

[9] T. K. Chia, K. C. Sim, H. Li, H. T. Ng, Statistical lattice-based spoken document retrieval, ACM Trans. Inf. Syst. 28 (1) (2010) 2:1–2:30.

[10] R. Christiansen, C. Rushforth, Detecting and locating key words in continuous speech using linear predictive coding, IEEE Trans. on Acoustics, Speech and Signal Processing, 25 (5) (1977) 361–367.

[11] L. Egghe, The measures precision, recall, fallout and miss as a function of the number of retrieved documents and their mutual interrelations, Information Processing & Management 44 (2) (2008) 856–876.

[12] A. Fischer, V. Frinken, H. Bunke, C. Suen, Improving HMM-based keyword spotting with character language models, in: 12th Int. Conference on Document Analysis and Recognition (ICDAR), 2013, pp. 506–510.

[13] A. Fischer, A. Keller, V. Frinken, H. Bunke, HMM-based Word Spotting in Handwritten Documents Using Subword Models, in: 20th Int. Conf. on Pattern Recognition (ICPR '10), 2010, pp. 3416 –3419.

[14] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recognition Letters 33 (Special Issue on Awards from ICPR 2010) (7) (2012) 934 – 942.

[15] A. Fischer, M. Wuthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, M. Stolz, Automatic transcription of handwritten medieval documents, in: 15th Int. Conference on Virtual Systems and Multimedia. (VSMM'09), 2009, pp. 137–142.

[16] V. Frinken, A. Fischer, H. Bunke, A Novel Word Spotting Algorithm Using Bidirectional Long Short-Term Memory Neural Networks, in: F. Schwenker, N. El Gayar (eds.), Artificial Neural Networks in Pattern Recognition, vol. 5998 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 185–196.

[17] V. Frinken, A. Fischer, R. Manmatha, H. Bunke, A Novel Word Spotting Method Based on Recurrent Neural Networks, IEEE Trans. on Pattern Analysis and Machine Intelligence 34 (2) (2012) 211–224.

[18] F. Gers, N. N. Schraudolph, J. Schmidhuber, Learning Precise Timing with LSTM Recurrent Networks, Journal of Machine Learning Research 3 (2002) 115–143.

[19] A. Ghorbel, J.-M. Ogier, N. Vincent, A segmentation free word spotting for handwritten documents, in: Proc. of the 13th Intl. Conf. on Document Analysis and Recognition (ICDAR'15), 2015, pp. 346–350.

[20] S. K. Ghosh, E. Valveny, Query by string word spotting based on character bi-gram indexing, in: 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2015, pp. 881–885.

[21] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A Novel Connectionist System for Unconstrained Handwriting Recognition, IEEE Transaction on Pattern Analysis and Machine Intelligence 31 (5) (2009) 855–868.

[22] A. Higgins, R. Wohlford, Keyword recognition using template concatenation, in: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, (ICASSP '85), vol. 10, 1985, pp. 1233–1236.

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

[24] J. Keshet, D. Grangier, S. Bengio, Discriminative keyword spotting, Speech Communication 51 (4) (2009) 317–329.

[25] R. Kneser, H. Ney, Improved backing-off for N-gram language modeling, in: Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP '95), vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, 1995, pp. 181–184.

[26] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson, G. Viorel Popescu, A Line-Oriented Approach to Word Spotting in Handwritten Documents, Pattern Analysis & Applications 3 (2000) 153–168.

[27] G. Kumar, V. Govindaraju, A bayesian approach to script independent multilingual keyword spotting, in: 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2014, pp. 357–362.

[28] V. Lavrenko, T. M. Rath, R. Manmatha, Holistic word recognition for handwritten historical documents, in: Proceedings of the First Int. Workshop on Document Image Analysis for Libraries, 2004, pp. 278–287.

[29] L. Likforman-Sulem, A. Zahour, B. Taconet, Text line segmentation of historical documents: a survey, Int. Journal on Document Analysis and Recognition 9 (2007) 123–138.

[30] E. Lleida, J. B. Marino, J. M. Salavedra, A. Bonafonte, E. Monte, A. Martinez, Out-of-vocabulary word modelling and rejection for keyword spotting, in: Proc. of the Third European Conference on Speech Communication and Technology, Berlin, Germany, 1993, pp. 1265–1268.

[31] R. Mandal, P. P. Roy, U. Pal, M. Blumenstein, Multi-lingual date field extraction for automatic document retrieval by machine, Information Sciences 314 (2015) 277 – 292.

[32] R. Manmatha, C. Han, E. Riseman, Word Spotting: a New Approach to Indexing Handwriting, in: Int. Conference on Computer Vision and Pattern Recognition (ICPR '96), 1996, pp. 631–637.

[33] R. Manmatha, J. L. Rothfeder, A scale space approach for automatically segmenting words from historical handwritten documents, Pattern Analysis and Machine Intelligence, IEEE Trans. on 27 (8) (2005) 1212–1225.

[34] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

[35] U.-V. Marti, H. Bunke, Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System, Int. Journal of Pattern Recognition and Artificial Intelligence 15 (2001) 65–90.

[36] U.-V. Marti, H. Bunke, The iam-database: an english sentence database for offline handwriting recognition, Int. Journal on Document Analysis and Recognition 5 (2002) 39–46.

[37] M. Oerder, H. Ney, Word graphs: an efficient interface between continuous-speech recognition and language understanding, in: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, vol. 2, 1993, pp. 119 –122.

[38] C. Oprean, L. Likforman-Sulem, C. Mokbel, A. Popescu, Blstm-based handwritten text recognition using web resources, in: 13th International Conference on Document Analysis and Recognition (ICDAR), 2015, pp. 466–470.

[39] S. Ortmanns, H. Ney, X. Aubert, A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition, Computer Speech and Language 11 (1) (1997) 43–72.

[40] V. Papavassiliou, T. Stafylakis, V. Katsouros, G. Carayannis, Handwritten document image segmentation into text lines and words, Pattern Recognition 43 (1) (2010) 369–377.

[41] H. Pesch, M. Hamdani, J. Forster, H. Ney, Analysis of preprocessing techniques for latin handwriting recognition, in: International Conference on Frontiers in Handwriting Recognition (ICFHR), 2012, pp. 280–284.

[42] J. Puigcerver, A. H. Toselli, E. Vidal, Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting, in: Proc. of the 13th Intl. Conf. on Document Analysis and Recognition (ICDAR'15), 2015, pp. 731–735.

[43] J. Puigcerver, A. H. Toselli, E. Vidal, Querying out-of-vocabulary words in lexicon-based keyword spotting, Neural Computing and Applications (2016) 1–10.

[44] T. M. Rath, V. Lavrenko, R. Manmatha, A statistical approach to retrieving historical manuscript images without recognition, Tech. rep., Space and Naval Warfare Systems Center San Diego CA (2003).

[45] T. M. Rath, R. Manmatha, V. Lavrenko, A search engine for historical manuscript images, in: Proc. of the international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, 2004, pp. 369–376.

[46] J. A. Rodríguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden Markov models and universal vocabularies, Pattern Recognition 42 (2009) 2106–2116.

[47] J. A. Rodríguez-Serrano, F. Perronnin, A model-based sequence similarity with application to handwritten word spotting, IEEE Trans. on Pattern Analysis and Machine Intelligence 34 (11) (2012) 2108–2120.

[48] J. Rohlicek, W. Russell, S. Roukos, H. Gish, Continuous hidden markov modeling for speaker-independent word spotting, in: Int. Conf. on Acoustics, Speech, and Signal Processing. (ICASSP-89.), 1989, pp. 627–630 vol.1.

[49] V. Romero, M. Pastor, A. H. Toselli, E. Vidal, Criteria for handwritten off-line text size normalization, in: Procc. of the IASTED Int. Conf. on Visualization, Imaging, and Image Processing (VIIP 06), Palma de Mallorca, Spain, 2006, 6 pages.

[50] V. Romero, A. H. Toselli, L. Rodríguez, E. Vidal, Computer assisted transcription for ancient text images, in: Proc of the Int. Conf. on Image Analysis and Recognition (ICIAR '07), vol. 4633 of Lecture Notes in Computer Science (LNCS), Springer-Verlag, Montreal (Canada), 2007, pp. 1182–1193.

[51] L. Rothacker, G. A. Fink, Segmentation-free query-by-string word spotting with bag-of-features HMMs, in: Proc. of the 13th Intl. Conf. on Document Analysis and Recognition (ICDAR'15), 2015, pp. 661–665.

[52] P. P. Roy, F. Rayar, J.-Y. Ramel, Word spotting in historical documents using primitive codebook and dynamic programming, Image and Vision Computing 44 (2015) 15 – 28.

[53] M. Rusinol, D. Aldavert, R. Toledo, J. Lladós, Efficient segmentation-free keyword spotting in historical document collections, Pattern Recognition 48 (2) (2015) 545–555.

[54] A. Sanchis, A. Juan, E. Vidal, A word-based naïve bayes classifier for confidence estimation in speech recognition, Audio, Speech, and Language Processing, IEEE Transactions on 20 (2) (2012) 565–574.

[55] N. Ström, Generation and Minimization of Word Graphs in Continuous Speech Recognition, in: Proc. IEEE Workshop on ASR'95, Snowbird, Utah, 1995, pp. 125–126.

[56] I. Szõke, P. Schwarz, L. Burget, M. Karafiát, J. Cernocký, Phoneme based acoustics keyword spotting in informal continuous speech, in: Radioelektronika 2005, Faculty of Electrical Engineering and Communication BUT, 2005, pp. 195–198.

[57] L. Tarazón, D. Pérez, N. Serrano, V. Alabau, O. Ramos Terrades, A. Sanchis, A. Juan, Confidence Measures for Error Correction in Interactive Transcription Handwritten Text, in: Image Analysis and Processing (ICIAP '09), vol. 5716 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 567–574.

[58] K. Terasawa, Y. Tanak, Slit style hog feature for document image word spotting, in: Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on, 2009, pp. 116–120.

[59] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, Alpha-Numerical Sequences Extraction in Handwritten Documents, in: Proc. of the Int. Conference on Frontiers in Handwriting Recognition (ICFHR '10), IEEE Computer Society, Washington, DC, USA, 2010, pp. 232–237.

[60] A. Toselli, V. Romero, M. P. i Gadea, E. Vidal, Multimodal Interactive Transcription of Text Images, Pattern Recognition 43 (5) (2010) 1814–1825.

[61] A. Toselli, V. Romero, E. Vidal, Word-graph based applications for handwriting documents: Impact of word-graph size on their performances, in: R. Paredes, J. S. Cardoso, X. M. Pardo (eds.), Pattern Recognition and Image Analysis, vol. 9117 of Lecture Notes in Computer Science, Springer Int. Publishing, 2015, pp. 253–261.

[62] A. Toselli, E. Vidal, Word-graph based handwriting key-word spotting: Impact of word-graph size on performance, in: 11th IAPR Int. Workshop on Document Analysis Systems (DAS), 2014, pp. 176–180.

[63] A. H. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, F. Casacuberta, Integrated Handwriting Recognition and Interpretation using Finite-State Models, Int. Journal of Pattern Recog. and Artificial Intelligence 18 (4) (2004) 519–539.

[64] A. H. Toselli, J. Puigcerver, E. Vidal, Context-aware lattice based filler approach for key word spotting in handwritten documents, in: Proc. of the 13th Intl. Conf. on Document Analysis and Recognition (ICDAR'15), 2015, pp. 736–740.

[65] A. H. Toselli, E. Vidal, Fast HMM-Filler approach for Key Word Spotting in Handwritten Documents, in: Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR '13), IEEE Computer Society, Washington, DC, USA, 2013, pp. 501–505.

[66] A. H. Toselli, E. Vidal, V. Romero, V. Frinken, Word-graph based keyword spotting and indexing of handwritten document images, Tech. rep., Universitat Politècnica de València (2013).

[67] N. Ueffing, H. Ney, Word-Level Confidence Estimation for Machine Translation, Comput. Linguistic 33 (1) (2007) 9–40.

[68] A. Vinciarelli, S. Bengio, H. Bunke, Off-line recognition of unconstrained handwritten texts using HMMs and statistical language models, IEEE Trans. on Pattern Analysis and Machine Intelligence 26 (6) (2004) 709–720.

[69] C. Wang, P. Zhang, Optimization of spoken term detection system, Journal of Applied Mathematics 2012 (548341) (2012) 1–8.

[70] F. Wessel, R. Schluter, K. Macherey, H. Ney, Confidence measures for large vocabulary continuous speech recognition, IEEE Trans. on Speech and Audio Processing 9 (3) (2001) 288–298.

[71] P. Woodland, C. Leggetter, J. Odell, V. Valtchev, S. Young, The 1994 HTK large vocabulary speech recognition system, in: Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP '95), vol. 1, 1995, pp. 73 –76 vol.1.

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

[73] S. Wshah, G. Kumar, V. Govindaraju, Statistical script independent word spotting in offline handwritten documents, Pattern Recogn. 47 (3) (2014) 1039–1050.

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

[75] M. Zhu, Recall, Precision and Average Precision, Working Paper 2004-09 Department of Statistics & Actuarial Science - Univ. of Waterloo (August 26 2004).

Appendix I: Proofs of some Properties of the paper: Word-Graph Based Keyword Spotting in Handwritten Document Images

I - ***Efficient, Dynamic Programming computation of*** $\varphi(q', q)$ (Eqs. (14)-(15)):

$$\sum_{\mathbf{w}:(q',q)\in\psi(\mathbf{w})} P_G(\mathbf{w} \mid \mathbf{x}) \;=\; \frac{\alpha(q')\cdot s(q',q)\cdot\beta(q)}{\beta(q_I)}, \quad \forall (q',q)\in E$$

Proof: By applying successively (13), (10), (11), (12) and (15):

$$\sum_{\mathbf{w}:(q',q)\in\psi(\mathbf{w})} P_G(\mathbf{w} \mid \mathbf{x}) = \sum_{\mathbf{w}:(q',q)\in\psi(\mathbf{w})} \frac{P_G(\mathbf{w},\mathbf{x})}{P_G(\mathbf{x})} = \frac{1}{\beta(q_I)} \sum_{\mathbf{w}:(q',q)\in\psi(\mathbf{w})} P_G(\mathbf{w},\mathbf{x})$$

$$= \frac{1}{\beta(q_I)} \sum_{\substack{\mathbf{w}:\\(q',q)\in\psi(\mathbf{w})}} \prod_{(q'',q''')\in\psi(\mathbf{w})} s(q'',q''')$$

Let $\Phi(q_i, q_j)$ denote the set of paths in G defined by sequences of nodes, departing from node q_i and arriving to node q_j.

$$= \frac{1}{\beta(q_I)} \sum_{\substack{(q_1,q_2,\dots,q_i,q',q,q_{i+1},\\ \dots,q_{n-1},q_n)\in\Xi(q_I,q_n):\\ q_n\in F}} [\, s(q_1,q_2)\cdots s(q_i,q')\,]\cdot s(q',q)\cdot[\, s(q,q_{i+1})\cdots s(q_{n-1},q_n)\,]$$

$$= \frac{1}{\beta(q_I)} \sum_{\substack{(q_I,q_2,\dots,\\ q_i,q')\in\Phi(q_I,q')}} \; \sum_{\substack{(q,q_{i+1},\dots,\\ q_{n-1},q_n)\in\Phi(q,q_n):\\ q_n\in F}} [\, s(q_1,q_2)\cdots s(q_i,q')\,]\cdot s(q',q)\cdot[\, s(q,q_{i+1})\cdots s(q_{n-1},q_n)\,]$$

$$= \frac{1}{\beta(q_I)} \left[\sum_{\substack{(q_I,q_2,\dots,\\ q_i,q')\in\Phi(q_I,q')}} s(q_1,q_2)\cdots s(q_i,q') \right] \cdot s(q',q) \cdot \left[\sum_{\substack{(q,q_{i+1},\dots,\\ q_{n-1},q_n)\in\Phi(q,q_n):\\ q_n\in F}} s(q,q_{i+1})\cdots s(q_{n-1},q_n) \right]$$

$$= \frac{\alpha(q')\cdot s(q',q)\cdot\beta(q)}{\beta(q_I)} = \varphi(q',q)$$

II - ***Flow Preserving Node*** **Property** (Eq. (16)):

$$\forall q\in Q \quad \sum_{(q',q)\in E} \varphi(q',q) \;=\; \sum_{(q,q'')\in E} \varphi(q,q'')$$

Proof: By applying successively (15), (11), (12) and (15):

$$\sum_{q':(q',q)\in E} \varphi(q',q) = \sum_{q':(q',q)\in E} \frac{\alpha(q')\cdot s(q',q)\cdot\beta(q)}{\beta(q_I)} = \frac{\alpha(q)\cdot\beta(q)}{\beta(q_I)}$$

$$= \sum_{q'':(q,q'')\in E} \frac{\alpha(q)\cdot s(q,q'')\cdot\beta(q'')}{\beta(q_I)} = \sum_{q'':(q,q'')\in E} \varphi(q,q'')$$

III - *Frame-Level Edge Posterior Consistency* Property (Eq. (17)):

$$\sum_{\substack{(q',q)\in E:\\ t(q')<i\leq t(q)}} \varphi(q',q) \;=\; 1, \quad 1\leq i\leq n$$

Proof: We must first show that: $\displaystyle\sum_{q:(q_I,q)\in E} \varphi(q_I,q)=1$

$$\sum_{q:(q_I,q)\in E} \varphi(q_I,q)=\sum_{q:(q_I,q)\in E} \frac{\alpha(q_I)\cdot s(q_I,q)\cdot \beta(q)}{\beta(q_I)}=\sum_{q:(q_I,q)\in E} \frac{s(q_I,q)\cdot \beta(q)}{\beta(q_I)}=\frac{\beta(q_I)}{\beta(q_I)}=1$$

Now, using mathematical induction:

- For $i=1 \longrightarrow \displaystyle\sum_{\substack{(q_I,q)\in E:\\ t(q_I)<1\leq t(q)}} \varphi(q_I,q)=1, \quad t(q_I)=0$
- Assuming now for $i=n \longrightarrow \displaystyle\sum_{\substack{(q',q)\in E:\\ t(q')<n\leq t(q)}} \varphi(q',q)=1$
- We need to prove that for $i=n+1 \longrightarrow \displaystyle\sum_{\substack{(q',q)\in E:\\ t(q')<n+1\leq t(q)}} \varphi(q',q)=1$

Let be $E_n = \{(q',q)\in E : t(q')<n\leq t(q)\}$ and $Q_n = \{q\in Q : n=t(q)\}$. Thereby, for $i=n$:

$$\sum_{\substack{(q',q)\in E:\\ t(q')<n\leq t(q)}} \varphi(q',q)=\sum_{(q',q)\in E_n} \varphi(q',q)=\sum_{\substack{(q',q)\in E_n:\\ q\notin Q_n}} \varphi(q',q)+\sum_{\substack{(q',q)\in E_n:\\ q\in Q_n}} \varphi(q',q)=1$$

And, for $i=n+1$

$$\begin{aligned}\sum_{\substack{(q',q)\in E:\\ t(q')<n+1\leq t(q)}} \varphi(q',q) &= \sum_{\substack{(q',q'')\in E_n:\\ q''\notin Q_n,\\ t(q')<n+1\leq t(q'')}} \varphi(q',q'')+\sum_{\substack{(q'',q)\in E:\\ q''\in Q_n,\\ t(q'')<n+1\leq t(q)}} \varphi(q'',q)\\ &= \sum_{\substack{(q',q'')\in E_n:\\ q''\notin Q_n,\\ t(q')<n+1\leq t(q'')}} \varphi(q',q'')+\sum_{\substack{(q',q'')\in E_n:\\ q''\in Q_n,\\ t(q'')<n+1}} \varphi(q',q'')=1 \quad \text{(by property 1)}\end{aligned}$$