Context-Aware Lattice based Filler approach for Key Word Spotting in Handwritten Documents

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

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

Enrique Vidal
Universitat Politècnica de València
Valencia, Spain
Email: evidal@iti.upv.es

***Abstract*—The so-called filler or garbage Hidden Markov Models (HMM-Filler) are among the most widely used models for lexicon-free, query by string key word spotting (KWS) in the fields of speech recognition and (lately) handwritten text recognition. However, it has important drawbacks. First, the keyword-specific HMM Viterbi decoding process needed to obtain the confidence scores of each spotted word involves a large computational cost. Second, in its traditional conception, the model does not take into account any context information – and more recent works where simple character bi-gram context is used show that not only the computational cost becomes even larger, but also the required keyword-specific language model becomes quite intricate to build. In a previous work we introduced KWS methods based on character lattices which proved very much simpler and faster than the traditional HMM-Filler, while providing practically identical results. Here we extend our previous work by using context-aware character lattices obtained by means of Viterbi decoding with high-order character N-gram models. Experimental results show that, as compared with a direct 2-gram HMM-filler implementation, the proposed approach requires between one and two orders of magnitude less query computing time. Moreover, for the first time in the field of handwritten text KWS, Filler-based results for N-grams up to $N = 6$ are reported, clearly showing a great impact of context on precision-recall performance.**

I. Introduction

In recent years, large quantities of historical handwritten documents are being scanned into digital images, which are then made available through web sites of libraries and archives all over the world. Despite this, the wealth of information conveyed by the text captured in these images remains largely inaccessible (no plain text, difficult to read even for researchers). To exploit and make profit of such a mass-digitization, information retrieval approaches are needed to allow the users to search accurately and efficiently for textual contents of such handwritten documents.

Aiming at this goal, *Keyword spotting* (KWS) methods are being proposed, from which one of the most popular is known as "HMM-Filler" [1], [2]. In this approach, a *filler* (or "garbage") model, along with a *word-specific* model for each query word, are built using character *hidden Markov models* (HMMs). HMM-Filler is used for *line-oriented* KWS, where whole text line images, without any kind of segmentation into words or characters, are analyzed to determine the degree of confidence that a given keyword appears in the image. One of the attractive feature of this approach is that it is "lexicon-free"; that is, it does not require any predefined lexicon as, for example, the case of the index-based method described in [3]. On the other side, the large computing time entailed by *word-specific* Viterbi decoding, which is needed for each query, becomes an important issue. Another major drawback is undoubtedly the lack of context-information for driving the search process, which significantly hinders its performance with respect to context-aware approaches such as [3].

In order to overcome the large computing time problem, an alternative method based on *character lattices* (CLs) was introduced in [4] to compute the (highly-expensive) word-specific probabilities. The resulting system was very much faster than the traditional HMM-Filler, while preserving almost identical precision-recall performance. On the other hand, the lack of context-information of the HMM-Filler was addressed in [5] by using character 1-gram and 2-gram models; however, the high computing cost problem persists. Moreover, *character* 2-grams can only provide very limited context (as compared for instance with *word* 2-grams) and using larger context only makes computing issues even worse, since cost can asymptotically grow exponentially with the N-gram order. It is also worth nothing that building *word-specific* character N-gram models becomes quite a complex task for $N > 1$ [5].

This paper follows the work presented in [5], but using the CL-based approach introduced in [4]. To this end, we generate CLs for increasing N-gram orders, on which we perform the query search according to [4]. A major contribution of the paper, which has been possible thanks to the great efficiency of CL-based computing, is the use of context beyond that provided by 2-grams. Results achieved with up to 6-grams clearly confirm the expected high positive impact of increasing context-information on the KWS precision-recall performance.

The rest of the paper is organized as follows. The next section overviews basic concepts of HMM-based handwritten text recognition, CLs, HMM-Filler and its CL-based approach. Evaluation measures, dataset, experimental set-up and results are presented/reported in section III. Finally, section IV summarizes the work presented and draws conclusions.

II. CL-based KWS and Related Framework

This section reviews the basics of HMM-based handwriting recognition and the CLs generated as a byproduct. It includes a brief overview of lattice re-scoring using higher order N-gram, which is employed to obtain CLs for context beyond that provided by 2-grams. Finally, the HMM-Filler framework and its CL-based counterpart are outlined.

A. HMM-based Handwriting Recognition

Both the HMM-Filler and its CL-based approach rely on a segmentation-free, HMM-based character recognizer. Recognizers of this kind accept as input a given handwritten text line image, represented as a sequence of D-dimensional feature vectors $\mathbf{x} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$, $\vec{x}_i \in \Re^D$ and find a most likely character sequence, $\widehat{\mathbf{c}} = \widehat{c}_1 \widehat{c}_2 \ldots \widehat{c}_L$, $\widehat{c}_i \in \Sigma$ (the character set):

$$\widehat{\mathbf{c}} = \arg\max_{\mathbf{c}} P(\mathbf{c} \mid \mathbf{x}) = \arg\max_{\mathbf{c}} p(\mathbf{x} \mid \mathbf{c})\, P(\mathbf{c}) \tag{1}$$

The score associated with $\widehat{\mathbf{c}}$ (called "Viterbi score") is:

$$S(\mathbf{x}) = \max_{\mathbf{c}}\, p(\mathbf{x} \mid \mathbf{c})\, P(\mathbf{c}) \tag{2}$$

The conditional density $p(\mathbf{x} \mid \mathbf{c})$ is approximated by previously trained morphological character HMMs, while the prior $P(\mathbf{c})$ is given by a character-level N-gram language model.

Eq. (1–2) are commonly solved by means of Viterbi decoding [6]. As a byproduct, a huge set of most likely character sequence hypotheses, along with the corresponding segmentation and probabilistic information, can be obtained and represented in the form of a CL. CLs are the character-level versions of the better known *"word-graphs"* [7].

B. Character-Lattice Basics

A CL of a vector sequence $\mathbf{x}$ is a weighted directed acyclic graph with a finite set of nodes Q, including an initial node $q_I \in Q$ and a set of final nodes $F \subseteq (Q - q_I)$. Each node q is associated with a horizontal position of $\mathbf{x}$, given by $t(q) \in [0, n]$, where n is the length of $\mathbf{x}$. For an edge $(q', q) \in E$ ($q' \neq q, q' \notin F, q \neq q_I$), $c = \omega(q', q)$ is its associated character and $s(q', q)$ its score. The score is the product of the N-gram probability for the character c, $P(c)$, and the likelihood that the segment $\vec{x}_{t(q')+1}, \ldots, \vec{x}_{t(q)}$, represents an image of the character $\omega(q', q)$, as computed during Viterbi decoding of $\mathbf{x}$.

A *complete path*, $\mathcal{P}$, of a CL is a sequence of edges $(q'_1, q_1), (q'_2, q_2), \ldots, (q'_L, q_L)$ such that $q'_1 = q_I$, $q_i = q'_{i+1}$, $1 \leq i < L$, $q_L \in F$. A complete path corresponds to a whole line decoding hypothesis and its score is the product of the scores of all its edges: $S(\mathcal{P}, \mathbf{x}) = \prod_{i=1}^{L} s(q'_i, q_i)$.

The *Viterbi score* of $\mathbf{x}$, $S(\mathbf{x})$, is the maximum of the scores of all complete CL paths. It can be easily and efficiently obtained by *Dynamic Programming*:

$$S(\mathbf{x}) \overset{\text{def}}{=} \max_{\mathcal{P}} S(\mathcal{P}, \mathbf{x}) \;=\; \Phi(q_I) \;=\; \max_{q \in F} \Psi(q)$$

where the *forward* (Ψ) and *backward* (Φ) partial path scores are recursively defined as:

$$\Psi(q) \;=\; \max_{i:(q_i,q) \in E} \Psi(q_i)\, s(q_i, q) \quad \text{if } q \neq q_I \tag{3}$$

$$\Phi(q) \;=\; \max_{j:(q,q_j) \in E} s(q, q_j)\, \Phi(q_j) \quad \text{if } q \notin F \tag{4}$$

with $\Psi(q_I) = 1$, and $\Phi(q) = 1 \;\; \forall q \in F$. These equations are similar to those used in the standard backward-forward computations in word-graphs (see [7]).

C. Re-scoring CLs with higher order N-grams

In automatic speech and handwriting text recognition, it is well known that the use of models with richer contextual information generally leads to better decoding performance. But also it is fairly certain that the corresponding decoding process becomes computationally more expensive as the search space grows exponentially with the context length (acoustic triphones, N-gram language models with $N > 2$, etc.).

"Lattice re-scoring" (LR) [8] is one way to overcome this computational bottleneck, provided that we want to keep the decoding likelihoods obtained in the previous decoding process (i.e. $p(\mathbf{x} \mid \mathbf{c})$) and use a higher order N-gram ($P(\mathbf{c})$ in Eq. (1)). LR has proved to be a good approximation to the actual output of a full decoding process with such a higher order N-gram model.

In short, given a lattice obtained with a N'-gram model and a higher order N-gram, LR performs node duplication to guarantee the uniqueness of the N-word contexts before placing probabilities on transitions. For this reason, the final re-scored lattice can be considerably larger than the original one, depending mainly on the order of the N-gram model used for re-scoring. In order to limit the size of the resulting lattice, beam-search-like pruning techniques can be applied [9].

D. HMM-Filler Approach

In the HMM-Filler approach, character HMMs are used to build both a *"filler" model*, F, and a *keyword-specific* model, K_v, for each individual keyword v to be spotted. For the *classical* HMM-Filler presented in [1], F is built by arranging all the trained character HMMs in parallel with a loop-back and the prior probabilities $P(\mathbf{c})$ are not used. This model accounts for any unrestricted sequence of characters. On the other hand, K_v is constructed to model the exact character sequence of v, surrounded by the space character and the same unrestricted character sequences modeled by F.

For the *contextual* HMM-Filler [5], $P(\mathbf{c})$ is given by an N-gram model, F (see a 3-gram example in Fig. 1 top). Building K_v for general N-gram is quite intrincate. As in the classical approach, the topology has to deal with the query word character sequence, v, delimited by optional space characters and by the N-gram SFSAs, F. In addition, this topology should also consider all query word sequence starting and ending possibilities given by $[1, N-1]$-gram SFSAs (see Fig. 1 bottom).

In a *preprocessing phase*, F can be used once for all to compute the Viterbi decoding score $s_f(\mathbf{x})$ (Eq. (2)) for each line image $\mathbf{x}$. Similarly, in the *searching phase*, for each keyword v to be spotted and for each line image $\mathbf{x}$, the Viterbi score $s_v(\mathbf{x})$ is computed using the keyword-specific model K_v. A spotting score $S(v, \mathbf{x})$ is then defined as:

$$S(v, \mathbf{x}) \overset{\text{def}}{=} \frac{\log s_v(\mathbf{x}) - \log s_f(\mathbf{x})}{L_v} \tag{5}$$

where L_v is the length of v in number of characters. If v is actually in the line image $\mathbf{x}$, a matching is expected with high (negative) values of $S(v, \mathbf{x})$, upper bounded by $S(v, \mathbf{x}) = 0$. Note that unlike it has been done here, in [1], [5] L_v is given directly in number of frames between the

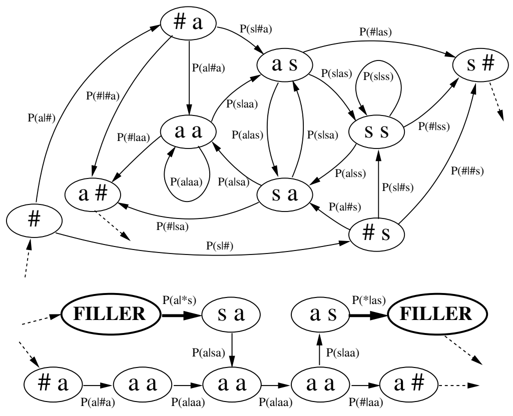

Fig. 1. Top: Simple character 3-gram filler model, F, for characters "a" and "s" (blank space). Bottom: 3-gram keyword model, K_v, for $v =$"aaa".

word detected borders, resulting in a better spotting score for the classical HMM-Filler (and the CL-based approach [4]: AP=0.36), instead of the one reported in Fig. 2 (AP=0.35). However, the proposed way to define L_v has proved to provide better spotting results when using N-gram models.

As with the basic HMM-Filler model, both the *preprocessing phase* and the *search phase* associated with each query require a Viterbi decoding process with a trellis size proportional to the size of the filler (N-gram) model, F [4]. Therefore both costs can grow (very) fast with N (asymptotically as $|\Sigma|^N$).

E. CL-based KWS Score for Character Sequences

Let a *sub-path* $\mathbf{e}$ of a CL be defined as a sequence of connected edges) of the form: $\mathbf{e} = (q'_1, q_1), (q'_2, q_2), \ldots, (q'_L, q_L)$: $q_i = q'_{i+1}, 1 \leq i < L$. The maximum score of a complete path $\mathcal{P}$ which contains a sub-path $\mathbf{e}$ is denoted as $\varphi(\mathbf{e}, \mathbf{x})$. It can be efficiently computed using the backward and forward partial scores of Eq. (3–4):

$$\varphi(\mathbf{e}, \mathbf{x}) \overset{\text{def}}{=} \max_{\mathcal{P}:\mathbf{e}\in\mathcal{P}} S(\mathcal{P}, \mathbf{x}) \;=\; \Psi(q'_1) \cdot \prod_{i=1}^{L} s(q'_i, q_i) \cdot \Phi(q_L) \quad (6)$$

where $\mathbf{e} \in \mathcal{P}$ means that $\mathbf{e}$ is a sub-path of $\mathcal{P}$. Given a character sequence $\tilde{\mathbf{c}}$, let $S'(\tilde{\mathbf{c}}, \mathbf{x})$ be the maximum score of a complete path containing a sub-path $\mathbf{e}$ associated with $\tilde{\mathbf{c}}$:

$$S'(\tilde{\mathbf{c}}, \mathbf{x}) \;=\; \max_{\mathbf{e}:\Omega(\mathbf{e})=\mathbf{c}} \varphi(\mathbf{e}, \mathbf{x}) \quad (7)$$

where $\Omega(\mathbf{e})$ is the character sequence $\tilde{\mathbf{c}} \equiv \tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_L$ associated with $\mathbf{e}$; that is, $\mathbf{e} = (q'_1, q_1), (q'_2, q_2), \ldots, (q'_L, q_L)$, $\omega(q'_i, q_i) = \tilde{c}_i, 1 \leq i \leq L$.

Assuming the CL is complete, if $\tilde{\mathbf{c}}$ is the character sequence of a word v to be spotted (including blank spaces) it can be readily seen that $S(\mathbf{x}) = s_f(\mathbf{x})$ and $S'(\tilde{\mathbf{c}}, \mathbf{x}) = s_v(\mathbf{x})$ (c.f. Eq. (5)). Therefore,

$$S(\tilde{\mathbf{c}}, \mathbf{x}) \;=\; \frac{\log S'(\tilde{\mathbf{c}}, \mathbf{x}) - \log S(\mathbf{x})}{L_v} \quad (8)$$

where L_v is the same length normalization as in Eq. (5).

As discussed in [4], the computational cost of this approach is largely dominated by the cost of generating the filler (N-gram) CL as a byproduct of Viterbi decoding each test line image with the the filler (N-gram) model, F. As with the traditional Filler-HMM approach, This is carried out in the *preprocessing* phase; however, the extra work required to producing the CLs makes this preprocessing cost significantly larger than that of the traditional Viterbi-only approach. This is largely compensated by the very light CL computing needed in the *search phase*, which no longer requires dealing with long line-image feature-vector sequences or large N-gram models.

III. Experiments

To assess effectiveness and efficiency of the CL-based KWS approach with higher order N-grams, several experiments were carried out. Evaluation measures, dataset, experimental setup and the results are presented next.

A. Evaluation Measures

The standard *recall* and *interpolated precision* measures [10] are used here. *Interpolated precision* is widely used in the literature to avoid cases in which *average precision* (AP, defined next) can be ill-defined. In addition, we employ another popular scalar KWS assessment measure called *average precision* (AP) [11] which, actually, is the area under the Recall-Precision curve.

B. Dataset

Experiments were carried out with the IAMDB dataset. *IAMDB* is a publicly available, well known modern English handwritten text corpus, compiled by the FKI-IAM Research Group on the base of the Lancaster-Oslo/Bergen Corpus (LOB). The last released version (3.0) is composed of 1 539 scanned text pages, handwritten by 657 different writers and partitioned into writer-independent training, validation and test sets. The line segmentation provided with the corpus [12] is used here. Basic statistics appear in Table I.

TABLE I. Basic statistics of the IAMDB database and its partition.

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

C. Set of Keywords to be Spotted

In this work the same set of query words defined and used in [1] is adopted. The main criterion was to consider as keywords all the vocabulary words appearing in the IAMDB training partition with exception of *stop words* which were all discarded. It is important to remark that this query set differs considerably from the one used in [5], a subset of it containing only queries appearing at least once in any of the test line images.

Thereby, the IAMDB query set consists of $V = 3\,421$ words, including numeric expressions and a few symbols. It is worth mentioning that, from these keywords, only $1\,098$ actually appear in the test line GTs. We say that these query words are *"pertinent"*, whereas the remaining $2\,323$ words are *"not pertinent"*. Clearly, spotting the non-pertinent words of this query set make also things more challenging compared with [5], since the system may erroneously find other similar words and thereby, leading to important precision degradations. On the other hand, some of the pertinent keywords appear more than once in the GT of the test partition. Concretely, there are 1 925 pertinent keyword occurrences, representing around 26% of the total number of running words in the GT of the test images. Each of the 3 421 keywords is to be tested against all the $M = 929$ test lines. So the total number of query-line events is $M \cdot V = 3\,178\,109$, from which only 1 916 are pertinent *and* relevant. All this information is summed up in Table II.

TABLE II. FEATURES OF SELECTED KEYWORD SET

	Total	Rel/Pert
Line images (M)	929	855
Keywords (V)	3 421	1 098
Running pertinent keywords	—	1 925
Line-query events ($M \cdot V$)	3 178 109	1 916

D. Experimental Set-up

IAMDB character HMMs were trained from the training partition. In general, a left-to-right HMM was trained for each of the elements appearing in the training text images, such as lowercase and uppercase letters, symbols, blank spaces, etc. Specific details of the image preprocessing, feature extraction and the HMM training setup usually adopted for IAMDB are described in [1]. The character HMM related parameters were optimized on the validation data: number of states and Gaussian densities per state.

In the *preprocessing phase* of the CL-based approach, the required N-gram models for generating the corresponding CLs, were trained from adequate external texts and smoothed using the Kneser-Ney back-off technique [13]. In this case as in [5], we have used the Lancaster-Oslo/Bergen corpus (LOB) [14] to train the N-gram models. This text corpus contains a variety of printed British English texts which, in total, encompasses more than one million words. Since some of these texts were employed to create the IAMDB itself, they have been removed from the validation and test sets.

Once all the N-gram models had been trained, the CLs of the validation and test lines were obtained using the HTK Toolkit [9] for each N-gram model with $1 \leq N \leq 4$, and also for the traditional single "filler" model (without prior probabilities). Since HTK can only decode directly using up to 2-gram models, we have resorted to *re-scoring* (see Sec. II-C to obtain CLs for $3-$ and 4-grams. That is, a 3-gram CL is obtained by re-scoreing the previous 2-gram CL with the 3-gram model and, in turn, a 4-gram CL by re-scoring the previous 3-gram CL with the 4-gram model. However, by applying HTK re-scoring beyond 4-grams, it was observed that the *character accuracy rate* (CAR)[1] computed on the decoded hypotheses of the validation set began to degrade. Therefore, in order to obtain results for 5- and 6-grams (Fig. 2) we used CLs produced by another recognizer, iATROS [15], capable of dealing with N-grams of order greater than 4^2.

[1]We assume that CAR is tight correlated with the KWS accuracy evaluation measure AP. Using CAR for the validation process of CLs generated with different N-gram orders (and related meta-parameters: GSF and CIP) results faster than employing directly AP.

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

In addition, the two related language model parameters *grammar scale factor* (GSF) and *character insertion penalty* (CIP) were also tuned empirically on the validation set for an optimum CAR value. Once these two parameter values have been found, the final CL generation of the corresponding 929 test lines was carried out. It is important to remark that in order to avoid procucing CLs of excessive size, this was limited by setting the maximum input branching factor to 30 and by applying beam-search pruning technique. Table III shows some statistics of the resulting CLs obtained for the test line set.

TABLE III. STATISTICS OF THE IAMDB CLS. ALL THE FIGURES ARE NUMBERS OF ELEMENTS, AVERAGED OVER ALL THE CLS. AVERAGE *branching factor* (BF) PER NODE AND CHARACTER AS WELL AS *character accuracy rate* (CAR(%)) ARE ALSO REPORTED.

RS	Ord	Nodes	Edges	BF	Paths	CAR(%)
HTK-recg	0	37 313	1 112 070	2.7	$\sim 10^{307}$	32.78
	1	35 477	1 056 820	3.0	$\sim 10^{306}$	57.31
	2	37 443	1 115 130	3.1	$\sim 10^{302}$	61.94
HTK-rscr	3	82 951	580 719	1.7	$\sim 10^{122}$	66.26
	4	125 280	557 420	1.4	$\sim 10^{81}$	70.43
iATROS	5	202 977	588 493	1.8	$\sim 10^{42}$	73.04
	6	111 235	228 739	1.4	$\sim 10^{26}$	74.54

Once all CLs have been generated, the *preprocessing phase* ends with the corresponding forward-backward score computation in each of them according to Eq. (3-4). In the *query phase*, the character sequence scores, $S'(\tilde{\mathbf{c}}, \mathbf{x})$, were computed for each keyword, $v \equiv \tilde{\mathbf{c}}$, and line image, $\mathbf{x}$. Finally, the KWS scores $S(\tilde{\mathbf{c}}, \mathbf{x})$ were determined for all the keywords, according to Eq. (8).

E. Results

Experiments were carried out with the IAMDB dataset using the CL-based approach described in Sec. II-E for different CL sets produced using N-gram models of increasing order: $N \in [0, 6]$. Fig. 2 plots the Average Precision (AP) in function of the order of N-gram models employed to generated CLs. As expected, the AP increases with the order of N-gram models allowing to hold more context information in the final obtained CLs. Furthermore, as can be seen in the plot of Fig.2, we also have included the language model perplexity (PPL) figure computed for each N-gram model order on the test set, which reflects the quality of the text constraints imposed by the different N-gram models. As can be observed, the improvement in spotting performance is clearly correlated with the reduction in perplexity, and thereby the used N-gram models are progressively becoming more restrictive [16].

Regarding the contextual HMM-Filler approach using 1- and 2-grams (employed as reference in Table IV), all experimentation was carried out according to [5]. As shown in [4], since both approaches share identical text-line image processing and character HMMs, performance results should be identical for both approaches, according to Eq. (8). However, since CL-based KWS (necessarily) relies on incomplete (pruned) CLs, some (negligible [4]) degradation is expected.

In Table IV we observe a very large gain in efficiency for the CL-based KWS method with respect to the contextual HMM-Filler approach. In this case, we report computing times

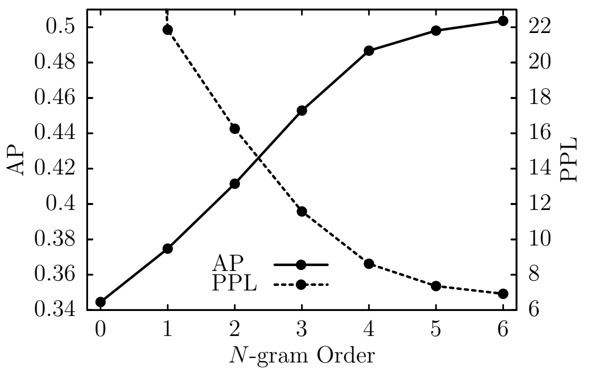

Fig. 2. Average precision (AP) and language model perplexity (PPL) both in function of the order of the N-gram used to generate the corresponding CLs.

only for the 1- and 2-gram, where no re-scoring (CLs were obtained directly from Viterbi decoding) or beam-search has been used to generate the CLs. The average query response time is reduced by a factor of 76 and 78 for 1- and 2-gram respectively. The corresponding overall time needed for fully indexing all the selected keywords is affordable using the proposed CL-based KWS method, while it becomes clearly prohibitive for the contextual HMM-Filler approach.

TABLE IV. EFFECTIVENESS (AP) AND EFFICIENCY IN TERMS OF PREPROCESSING AND AVERAGE QUERY TIMES AND TOTAL INDEXING TIMES, FOR CLASSICAL HMM-FILLER (REF) AND CL-BASED KWS (CL). CORRESPONDING FIGURES FROM PREVIOUS WORK [4] ARE INCLUDED.

	Work [4]	1-gram		2-gram	
	CL	REF	CL	REF	CL
Preproc. Time (min)	219.5	65.6	227.5	66.4	245.7
Av. Query Time (min)	0.9	68.3	0.9	71.0	0.9
Total Ind. Time (days)	2.3	162.3	2.3	168.7	2.3
Average Precision	0.36	0.37		0.41	

This table also contains the relevant computing times, split into *preprocessing* and *search* (query) times, the latter given in average minutes required for each single keyword search. The table also shows total time (in days) needed to fully index the corpus with the corresponding selected keywords (3 421 for IAMDB – c.f. Sec. III-C).

IV. REMARKS AND CONCLUSIONS

In this paper we have presented new spotting results on the IAMDB dataset using an extension of the CL-based KWS method presented in [4]. This extension consists in introducing context information on the character lattices employed for performing the query search, which are produced by a standard HMM-based handwritten character recognizer. The context information is provided by employing character N-gram models with the mentioned recognizer. Actually, for 1- and 2-grams, this method obtains practically the same spotting results as the original contextual HMM-Filler approach in [5], but with a drastic reduction of computational cost. Moreover, the proposed approach avoids the increasing difficulty of building the *word-specific* models when for increasing order of the filler N-gram models. Thanks to the simplicity and great computational efficiency of the CL-Filler approach, we have been able to report, for the first time, handwritten text recognition KWS Filler-HMM results with filler models beyond 2-grams.

It should be emphasized, that the IAMDB test set is extremely small, as compared with the size of the (historical) text image collections for which keyword indexing and search solutions are needed. For instance, medium sized book or bundle collection such as *Bentham* or (the full) *George Washington* are at least 1000 times larger than IAMDB. Since the computing times of Table IV scale linearly with the size of the text image collection considered, for collections of such a moderate size, a traditional bi-gram HMM-Filler system would require about 50 days to honor a single keyword query!. In contrast, the proposed CL-Filler approach would need only about half a day. Clearly, by resorting to (simple) parallelization and other acceleration techniques, the latter can become useful in practice, but the former can not.

REFERENCES

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

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

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

[4] A. Toselli and E. Vidal, "Fast hmm-filler approach for key word spotting in handwritten documents," in *12th Int. Conf. on Document Analysis and Recognition (ICDAR), 2013*, Aug 2013, pp. 501–505.

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

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

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

[8] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, no. 1, pp. 43–72, 1997.

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

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

[11] S. Robertson, "A new interpretation of average precision," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR. New York, NY, USA: ACM, 2008, pp. 689–690.

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

[13] R. Kneser and H. Ney, "Improved backing-off for N-gram language modeling," in *Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1. IEEE Computer Society, 1995, pp. 181–184.

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

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

[16] U.-V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 01, pp. 65–90, 2001.