Noname manuscript No.
(will be inserted by the editor)

Querying Out-of-Vocabulary Words in Lexicon-based Keyword Spotting

Joan Puigcerver · **Alejandro H. Toselli** · **Enrique Vidal**

Received: date / Accepted: date

Abstract Lexicon-based handwritten text keyword spotting (KWS) has proven to be a faster and more accurate alternative to lexicon-free methods. Nevertheless, since lexicon-based KWS relies on a predefined vocabulary, fixed in the training phase, it does not support queries involving out-of-vocabulary (OOV) keywords. In this paper, we outline previous work aimed at solving this problem and present a new approach based on smoothing the (null) scores of OOV keywords by means of the information provided by "similar" in-vocabulary words. Good results achieved using this approach are compared with previously published alternatives on different data sets.

Keywords keyword spotting, lexicon-based, smoothing, out-of-vocabulary, handwritten text recognition

1 Introduction

The aim of handwritten text keyword spotting (KWS) is to determine in which document images or image regions a given keyword appears with some likelihood. This work focuses on the case where queries are presented to the system as strings typed by the user (known as Query-by-String) [21,4,6,29,1,26,12,16,17,25], although an alternative formulation of KWS, where queries are presented as example images (known as Query-by-Example) is also very popular in the literature [10,19,11,7,22, 5,3,27]. Query-by-Example approaches are typically training-free and are based on template (image) matching between the query (example image) and word-sized image regions of the documents.

In this work, we focus on Query-by-String since it presents certain advantages over Query-by-Example: a) the users do not need any example image to specify the queries, b) they can use a very similar (or identical) interface to that of conventional web searching, and c) the indexing schemes typically used (and particularly the one

Joan Puigcerver, Alejandro H. Toselli and Enrique Vidal
PRHLT Research Centre, Universitat Politècnica de València,
Camino de Vera, s/n - 46022 Valencia - Spain
E-mail: {joapuipe,ahector,evidal}@prhlt.upv.es

we use) allow for very fast lookup times. Notice, however, that the statistical pattern recognition, training-based KWS approach underlying the work presented in this paper has also been used in Query-by-Example scenarios [27], providing much better performance than other state-of-the-art Query-by-Example approaches.

In the recent past, methods based on holistic Handwritten Text Recognition (HTR) technologies have been proposed for Query-by-String KWS in handwritten text images [4,6,29,26,12,16,17,25]. These methods typically rely on the use of optical, lexical, language and other statistical models, trained using transcribed handwritten text images of the considered task.

Particularly, this work follows the approach introduced in [26,25], which uses a word-level language model, a lexicon and a set of character optical models. For each word in the lexicon, these models are used to determine how likely is that a query word is written in given image regions. Using this probabilistic information, image regions such as lines or pages can be easily indexed to allow simple and accurate search under the so called *precision-recall trade-off* model. In this model, a confidence threshold is more or less explicitly specified by the user in order to trade accuracy of the spotting results (precision) for some guarantee that no regions containing the keyword are missed (recall).

This method provides very much faster searches than traditional lexicon-agnostic approaches, such as the HMM-Filler [4] or KWS using character-level bi-directional recurrent neural networks (BLSTM) [6]. Moreover, the Precision–Recall performance provided by lexicon-based methods is generally much better than that of the traditional HMM-Filler and comparable with that of BLSTM based KWS [26,25].

An important problem of lexicon-based methods is how to deal with out-of-vocabulary (OOV) queries: since these methods rely on a fixed vocabulary, determined during the model training phase, they will give a null score for any keyword not included in this lexicon.

Here we outline our previous work to deal with this problem and present a new technique for smoothing lexicon-based KWS scores. We show that this proposal improves the ones presented in [17]. In both cases, a similarity metric between OOV and indexed keywords is used to approximate OOV KWS scores. We also compare our new method with [16], which is a combination of lexicon-based and HMM-Filler systems and provides excellent performance results. However, as shown in our experiments, using the HMM-Filler method leads to much slower query processing, even when the fast HMM-Filler search technique introduced in [24] is used.

This paper extends our IbPRIA-2015 proceedings paper [18] by providing more complete outlines of the previous methods [17,16] and a more detailed account of the new one [18]. It also adds new empirical results to help better understanding the differences between the smoothing methods described in the paper and the basic approaches. The paper is organized as follows: Section 2 introduces basic concepts of the KWS framework used in this work, Section 3 briefly describes previous smoothing methods and presents the proposed method in detail, Section 4 reports the experiments conducted to evaluate this work and compare the results with those obtained with previously published approaches. Finally, conclusions are drawn in Section 5.

2 Review of Lexicon-Based Keyword Spotting

The smoothing method proposed in this work (c.f., Sec. 3.3) can be applied to any lexicon-based KWS approach which provides KWS scores that can be adequatelly interpreted as *relevance probabilities*, as discussed below. In order to allow for proper comparison with results of many of our previous works on KWS, we have chosen to use the method presented in [26,25].

For each query keyword v and each text line image region, denoted as $\mathbf{x}$, a score $S(\mathbf{x},v)$ is needed to measure how likely is the event "keyword v is written in $\mathbf{x}$", or re-phrased as "text line $\mathbf{x}$ is relevant for keyword v". In [26,25] it is computed as:

$$S(\mathbf{x},v) \stackrel{\text{def}}{=} \max_{1\leq i\leq m} P(v \mid \mathbf{x},i) \tag{1}$$

where m is the length of $\mathbf{x}$ and $P(v \mid \mathbf{x},i)$, called *frame-level word posterior*, is the probability that the word v is present in the line image $\mathbf{x}$ at horizontal position i. This simple computation tends to provide in practice better KWS scores than other possible approximations to the true relevance likelihood aimed at.

In order to compute $P(v \mid \mathbf{x},i)$ for each word in a given lexicon or vocabulary, V, and line image, $\mathbf{x}$, a language model and a set of character optical models are needed. In most previous works, n-grams and Hidden Markov Models (HMM) have been used for language and character optical modeling, respectively (nevertheless, all the problems and techniques introduced in this paper also apply to any other lexicon-based systems using different types of optical character models, such as neural networks). Using these models, $P(v \mid \mathbf{x},i)$ can be accurately computed through an extension of the conventional process used to decode $\mathbf{x}$ [26,25].

The required models are trained from moderate amounts of training images, accompanied by the corresponding transcripts, using well known statistical estimation techniques [8]. The lexicon V, on the other hand, is also obtained from the training transcripts and possibly expanded with additional words obtained from other relevant texts, if they are available.

In some of the coming sections, rather than an arbitrary score, a probabilistic interpretation of $S(\mathbf{x},v)$ will be needed. To this end, we observe that, since $P(v \mid \mathbf{x},i)$ is a well-defined discrete probability mass function, it is bounded between 0 and 1 and thus it can be properly used to define the following Bernoulli distribution:

$$P(R \mid \mathbf{x},v) \stackrel{\text{def}}{=} \begin{cases} S(\mathbf{x},v) & R=1 \\ 1-S(\mathbf{x},v) & R=0 \end{cases} \tag{2}$$

where the random variable R represents the event "text line $\mathbf{x}$ is relevant for keyword v". In order to explicitly assume this probabilistic meaning of $S(\mathbf{x},v)$, from now on we will refer to it as $P(R=1 \mid \mathbf{x},v)$, or simply $P(R \mid \mathbf{x},v)$.

It is worth pointing out that, for the given lexicon V, $P(v \mid \mathbf{x},i)$ and thereby $P(R \mid \mathbf{x},v)$, can be easily computed during an off-line indexing phase for all v in V (or for all v for which $P(R \mid \mathbf{x},v)$ is not negligible). This avoids having to carry out heavy computations during user's query lookup and permits extremely fast query processing (see experiments in Section 4).

3 Out-of-Vocabulary Queries

As briefly mentioned in the introduction, the main drawback of lexicon-based KWS is its inability to deal with OOV keywords. The underlying problem is that $P(R \mid \mathbf{x}, v)$, computed using a word-level LM, is null for any word not included in the lexicon V.

In this section, different techniques to mitigate this problem will be explained. In all the cases, alternative techniques are used to estimate the relevance probabilities only for keywords $u \notin V$. That is, the approaches to compute $P(R \mid \mathbf{x}, u)$ discussed in the coming subsections are only applied for OOV queries. In general, for an arbitrary keyword w, relevance probabilities are computed as:

$$P(R \mid \mathbf{x}, w) \;=\; \begin{cases} P_V(R \mid \mathbf{x}, v) & w = v \in V \\ \tilde{P}(R \mid \mathbf{x}, u) & w = u \notin V \end{cases} \tag{3}$$

where $P_V(R \mid \mathbf{x}, v)$ is the original lexicon-based probability (i.e. Eq. (2)) and $\tilde{P}(R \mid \mathbf{x}, u)$ is the specific solution presented in each subsection.

Two types of approaches to estimate $\tilde{P}(R \mid \mathbf{x}, u)$ will be considered: One is to rely on relevance probabilities available for words $v \in V$ which are "similar" to $u \notin V$. The other is to consider u in terms of its character spelling and use a lexicon-free, character-based KWS method.

In the following equations we will always refer to in-vocabulary keywords with the variable v, and OOV keywords with u.

3.1 Heuristic Line-level Smoothing

A first attempt to solve the OOV keyword problem was presented in [17], by directly defining the probability $\tilde{P}_1(R \mid \mathbf{x}, u)$ of an OOV word $u \notin V$, based on the probability $P_V(R \mid \mathbf{x}, v)$ of other words $v \in V$ and the Levenshtein distance [13] between the character-sequence representation of keywords u and v:

$$\tilde{P}_1(R \mid \mathbf{x}, u) \stackrel{\text{def}}{=} \max_{v \in V} P_V(R \mid \mathbf{x}, v)^{1-\alpha} \exp(-d(u, v))^{\alpha} \tag{4}$$

where $d(u, v)$ refers to the Levenshtein distance [13] between character representations of keywords u and v. The parameter $\alpha \in [0, 1]$ is used to fine-tune the importance of the similarity measure, in the same way that the grammar scale factor is used to balance the importance of optical and language models in HTR and Automatic Speech Recognition (ASR). In this case, when u and v are similar, the probability of the in-vocabulary word $v \in V$, $P_V(R \mid \mathbf{x}, v)$, has a greater effect in the resulting relevance probability. This is achieved by scaling $P_V(R \mid \mathbf{x}, v)$ with a factor that exponentially decays with $d(u, v)$.

Obviously, Eq. (4) can not be properly interpreted in probabilistic terms and is thus presented here just as a intuitively derived heuristic.

3.2 Heuristic Frame-level Smoothing

In [17], a more formal approach was studied, consisting in smoothing the frame-level posterior probabilities used in [26,25]. As in the previous technique, the smoothing here is based on word similarities between character representations of in-vocabulary words, v, and the OOV keyword, u. A main difference here is that smoothing is carried out at the frame level; that is, the distribution $P(v \mid \mathbf{x}, i), v \in V$, introduced in Sect. 2 is smoothed in order to give some probability mass to any keyword $u \notin V$. Before, we defined $P(v \mid \mathbf{x}, i)$ as the probability that (part of) word v is written at frame i in the image line $\mathbf{x}$. However, the modeling of this probability distribution was biased since, being restricted to the given vocabulary V, it is not defined over all sequence of characters. Thus, a new probability distribution, similar to the previous one, is introduced instead: $P(u \mid \mathbf{x}, i)$, which is defined $\forall u \in \Sigma^*$. The proposed method marginalizes $P(u \mid \mathbf{x}, i)$ among all keywords in V:

$$P(u \mid \mathbf{x}, i) \;=\; \sum_{v \in V} P(u, v \mid \mathbf{x}, i) \;=\; \sum_{v \in V} P(v \mid \mathbf{x}, i) \cdot P(u \mid \mathbf{x}, i, v) \tag{5}$$

It should be realized that the u and v in $P(u, v \mid \mathbf{x}, i)$ come from two different random variables, thus this probability should not be interpreted as "probability that word u and word v are written at the i-th frame of $\mathbf{x}$", but as "probability that the sequence of characters u is written and the word v was recognized at the i-th frame of $\mathbf{x}$".

It can be assumed that the probability of u is conditionally independent of $\mathbf{x}$ and i, given v. This is because the *similarity* between a sequence of characters u and a word v does not depend on how v is rendered in the image $\mathbf{x}$. While this might not be strictly true, it simplifies the complexity of the formulation. Then, Eq. (5) reduces to:

$$P(u \mid \mathbf{x}, i) \;\approx\; \sum_{v \in V} P(v \mid \mathbf{x}, i) \cdot P(u \mid v) \tag{6}$$

The distribution $P(u \mid v)$ is a probabilistic model of string similarity. Here we follow a traditional stochastic error correction model (see e.g. [2]) using the classical definition of symbol insertion, deletion and substitution operations, as in the Levenshtein distance. For each word $v \in V$, let $v = v_1, \cdots, v_m$, $v_i \in \Sigma, 1 \leq i \leq m$, be its character sequence representation, where Σ is the character alphabet. A stochastic finite-state machine (FSM) can be built with states $q_0, q_1, \cdots, q_m$, where q_0 is the initial state and q_m is the final state. There are edges joining consecutive states, which represent the operations of substitution and deletion of a symbol v_i, and loops for each state representing the insertion of new symbols. Observe that this FSM can accept any string in Σ^*. Figure 1 shows the FSM built for the string $v =$"aab".

Weights are associated to edges in the FSM which allow us to compute $P(u \mid v)$ by means of a forward-like algorithm. These weights, interpreted as transition probabilities, are estimated from the frequencies of the corresponding edit operations given by the minimum Levenshtein distance alignment [13] of a training set of pairs of strings.

As discussed in [17], this modeling for $P(u \mid v)$ presents some problems, the most notable one is that the value of $P(u \mid v)$ decays with the length of the strings u and v, since it is a discrete probability mass function defined over an infinite space. In

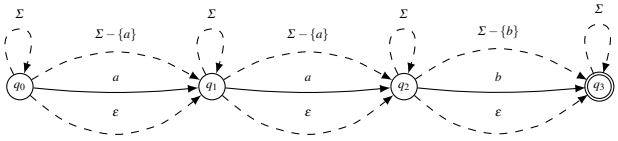

Fig. 1: FSM used to compute $P(u \mid \text{“aab”}), \forall u \in \Sigma^*$. Erroneous edit operations are represented by dashed lines.

order to alleviate this problem, in [17] a heuristic modification of Eq. (6) is proposed as follows:

$$\tilde{P}(u \mid \mathbf{x}, i) \stackrel{\text{def}}{=} \frac{\sum_{v \in V} P(v \mid \mathbf{x}, i) \cdot f(u, v)^{\alpha}}{1 + \sum_{v \in V} P(v \mid \mathbf{x}, i) \cdot f(u, v)^{\alpha}} \tag{7}$$

where $\alpha \in [0, \infty)$ is again an hyperparameter introduced to weight the importance of the similarity between u and v, and $f(u, v)$ is a similarity factor (it is not a probability mass function anymore), defined as:

$$f(u, v) = \frac{P(u \mid v)}{\max_{u' \in \Sigma^*} P(u' \mid v)} \tag{8}$$

It is worth noting that these heuristics are *not* applied because $P(u \mid v)$ or $P(u \mid \mathbf{x}, i)$ are not well normalized. In fact, it can be proven that $P(u \mid v)$ modeled by a FSM is a well-defined probability mass function [2]. Once proven that, it is trivial to show that $P(u \mid \mathbf{x}, i)$ is so as well. However the probability assigned to long sequences by $P(u \mid v)$ diminishes exponentially with length, as discussed before, and this ends up damaging significantly the performance of long keywords in the KWS experiments. The previous heuristics are introduced explicitly to alleviate this problem and ensure that $\tilde{P}(u \mid \mathbf{x}, i) \leq 1, \forall u \in \Sigma^*$, since this is the only requirement to make $\tilde{P}_2(R \mid \mathbf{x}, u)$ a valid binary probability distribution, given the definition in Eq. (3).

Using the smoothed frame-level posteriors $\tilde{P}(u \mid \mathbf{x}, i)$, a relevance probability for OOV keywords is computed as in Eq. (1):

$$\tilde{P}_2(R = 1 \mid \mathbf{x}, u) \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} \tilde{P}(u \mid \mathbf{x}, i) \tag{9}$$

It is important to realize that a significant computational load is entailed by Eq. (6-9) which, in practice, makes this method only suitable for small or moderately sized document image collections.

3.3 Probabilistic Line-Level Smoothing

In this subsection, we present a new probabilistic line-level smoothing method, preliminary introduced in [18]. It is based on the same intuition as in the heuristic approach discussed in Sect. 3.1, but it follows a sound probabilistic formulation. Moreover, this method is as fast as that of Sect. 3.1 and does not suffer from the string-length problems discussed in Sect. 3.2.

The complex definition of Eq. (7) was introduced mainly to mitigate the decay of $P(u \mid v)$ with the lengths of u, v. Here, we follow a less fine-grained smoothing approach by directly marginalizing $P(R \mid \mathbf{x}, u)$ among all $v \in V$:

$$P(R \mid \mathbf{x}, u) \;=\; \sum_{v \in V} P(R, v \mid \mathbf{x}, u) \;=\; \sum_{v \in V} P(R \mid \mathbf{x}, u, v) \cdot P(v \mid \mathbf{x}, u) \tag{10}$$

Similarly as in the previous section, observe that u and v come from different random variables. While u is defined among all sequence of characters ($u \in \Sigma^*$), v is defined only among all words in the lexicon ($v \in V$). Taking this into account, $P(R, v \mid \mathbf{x}, u)$ can be interpreted as the "probability that line $\mathbf{x}$ is relevant for the sequence of characters u and the word v was recognized in it". Likewise, $P(R \mid \mathbf{x}, u, v)$, would be defined then as "probability that line $\mathbf{x}$ is relevant for the sequence of characters u, given that the word v was recognized in it".

Now, we make two fairly natural independence assumptions. First, assume that v is conditionally independent of $\mathbf{x}$, given u, i.e. $P(v \mid \mathbf{x}, u) \approx P(v \mid u)$. Similarly, we assume that $P(R \mid \mathbf{x}, u, v) \approx P(R \mid \mathbf{x}, v)$. This way, the relevance probability in Eq. (10) is approximated as:

$$\tilde{P}_3(R \mid \mathbf{x}, u) \stackrel{\text{def}}{=} \sum_{v \in V} P_V(R \mid \mathbf{x}, v) \cdot P(v \mid u) \tag{11}$$

It should be noted that the similarity probability distribution $P(v \mid u)$ is the converse of the one used in Section 3.2. Now it must be normalized across all $v \in V$, rather than $\forall u \in \Sigma^*$. Since the distribution is over a finite set of elements, it can be defined in arbitrary ways that do not exhibit the probability vanishing problems explained before. In particular, we choose the following distribution based on the Levenshtein distance $d(u, v)$:

$$P(v \mid u) \stackrel{\text{def}}{=} \frac{\exp(-\alpha\, d(u, v))}{\sum_{v' \in V} \exp(-\alpha\, d(u, v'))} \tag{12}$$

In the same way that the previous smoothing methods did, we introduce a parameter $\alpha \in [0, \infty)$ to tune the contribution of the similarity measure.

3.4 Combining Lexicon-based and Lexicon-free KWS

This approach, originally presented in [16], simply computes the probability scores of OOV keywords, $u \notin V$, by using an alternative lexicon-free, character-level KWS model; namely, the popular HMM-Filler approach [4]. It uses two different character HMMs: a generic "filler" or "garbage" model, f, and a *keyword-specific* model, k_u, for the keyword u, as depicted in Figure 2.

Both models are composed of optical character HMMs, which are exactly the same trained for the lexicon-based KWS approach outlined in Sect. 2. Using these models, a KWS confidence score $S_F(\mathbf{x}, v)$ is computed by means of the following equation:

$$S_F(\mathbf{x}, v) \stackrel{\text{def}}{=} \frac{\max_{\mathbf{q}} \log p_{k_v}(\mathbf{q}, \mathbf{x}) - \max_{\mathbf{q}'} \log p_f(\mathbf{q}', \mathbf{x})}{|v|} \tag{13}$$

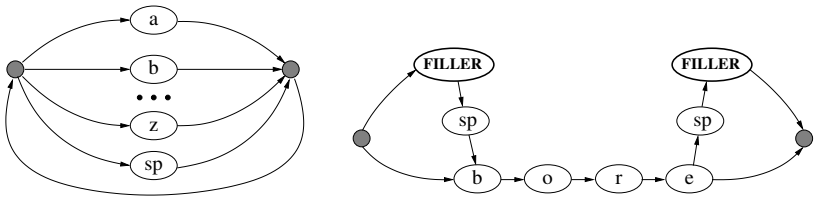

Fig. 2: "Filler HMM" (f) on the left and "keyword HMM" (k_v) on the right, built for the keyword v = "bore".

where $|v|$ is the length of v, $\mathbf{q}$ and $\mathbf{q}'$ are HMM state sequences, and expressions like $\max_{\mathbf{q}} \log P(\mathbf{q},\mathbf{x})$ refer to the standard Viterbi decoding log-likelihoods. The subscript in the P notation refers to the modeling of the likelihood: P_{k_v} is the likelihood given by the keyword-specific model and P_f is the likelihood according to the "filler".

In its traditional form, this technique is less accurate and more computationally expensive than lexicon-based methods such as the one outlined in Sect. (2) [24, 26]. However, we have recently proposed a much faster technique to obtain identical HMM-Filler scores by using character lattices obtained by Viterbi decoding the line images only with the filler model (i.e., without the expensive use of word-specific models) [24]. Using this accelerating technique, HMM-Filler can become actually useful in practice (at least for small and medium sized documents) to deal with (hopefully few) OOV queries.

According to Eq. (13), one should realize that $S_F(\mathbf{x},v)$ is actually a log-likelihood with a value in the range $(-\infty,0]$. In order to allow interpretation in terms of a relevance probability, as needed in Eq. (3), a exponentiation function has to be applied, leading to:

$$\tilde{P}_4(R \mid \mathbf{x},u) \stackrel{\text{def}}{=} \exp(\eta \cdot S_F(\mathbf{x},u)) \tag{14}$$

The parameter $\eta \in [0,\infty)$ is used to better adjust the conversion of the log-probability into a $[0,1]$ score, and its value is adjusted using a validation partition, as it was done with the hyperparameters of the other approaches.

4 Experiments

The experiments conducted to compare the approaches described in the previous section are presented now. Two different corpora were used and performance was assessed with two well known metrics which are widely used in the KWS community.

These two corpora vastly differ in terms of vocabulary size and, more importantly, in the amount of OOV query events. Therefore, the empirical results will allow us to draw more comprehensive conclusions about the relative capabilities of the proposed smoothing approaches in heterogeneous KWS scenarios.

4.1 Performance assessment

Assessment was carried out using the average precision (AP) metric [20], based on the recall and precision measures. Interpolated precision is adopted to avoid ill-defined cases which may appear with plain precision [14]. Both precision and recall are functions of a threshold used to determine whether the score $P(R=1 \mid \mathbf{x}, v)$ is high enough to actually assume that $\mathbf{x}$ is relevant to v. AP provides a single scalar quality measure taking into account all possible thresholds for all the keywords considered in an experiment:

$$\mathrm{AP} = \int_0^1 p(r)dr \tag{15}$$

where $p(r)$ is the global precision as a function of the recall r, considering all queries and documents.

Additionally, we report the mean average precision (mAP), which is also widely adopted in the literature. It is computed by averaging individual AP values of each individual keyword, $AP(q)$.

$$\mathrm{mAP} = \frac{1}{|Q|}\sum_{q\in Q}\mathrm{AP(q)} = \frac{1}{|Q|}\sum_{q\in Q}\int_0^1 p_q(r)dr \tag{16}$$

where Q is the query set and $p_q(r)$ is the precision of the query keyword q as a function of the recall r.

Since AP is computed globally across all keywords, it can be observed that systems with a good AP metric provide more consistent KWS score values across different keywords, which is not necessarily true if one looks only at the mAP metric. Thus we decided to optimize our hyperparameters based on the AP metric.

Finally, we also compared the average time required to compute the different KWS scores needed for an OOV query, in each corpus. We only considered the OOV keywords for this comparison, since the scores of in-vocabulary keywords can be precomputed and the lookup time is extremely small and asymptotically constant (if hash tables are used, for instance), as mentioned in Section 2.

4.2 Corpora and Query Sets

Two data sets were considered: "Cristo-Salvador" (CS)[1] and IAM[2] [15].

CS is a small XIX century single-writer Spanish manuscript. We used exactly the same partitioning as in [24,17]. Since the CS corpus is quite small, we ignored capitalization and diacritics to build the lexicon and the LM, and performed cross-validation to tune all parameters.

IAM, on the other hand, consists of English handwritten texts from many writers. We used the same data partitions used in previous KWS experiments [4,6,24,16]. In addition to the text in the line images, we used three external text corpora (LOB, Brown and Wellington), to build the lexicon and to train a LM (c.f. Sect. 4.3).

[1] `https://www.prhlt.upv.es/page/projects/multimodal/cs/index`

[2] `http://www.iam.unibe.ch/fki/databases/iam-handwriting-database`

Since all our experiments were aimed at assessing KWS performance for OOV keywords, in both datasets, we used the set of words written in the test lines as the *query set*. This makes our results not directly comparable with those previously published works in which only the the words included in the training set were considered as keywords (thereby implicitly ignoring the OOV problem). On the other hand, this allows us to properly compute mAP values, which are only defined for relevant queries. In the case of IAM, following previous works on this dataset, we subtracted from the query set all stop words. Table 1 summarizes the most important information of the corpora.

Table 1: Tables summarizing the corpora used for experimentation.

(a) Basic statistics of the selected databases.

	CS		IAM		
	Train	Test	Train	Valid.	Test
Running Chars	35 863	26 353	269 270	39 318	39 130
Running Words	6 223	4 637	47 615	7 291	7 197
# Lines	675	497	6 161	920	929
Char Set Size	78	78	72	69	65
Word Lex. Size	2 236	1 671	20 000	2 442	2 488
OOV Lex. Size	—	1 051	—	435	437

(b) Details of the query sets used in each dataset.

	CS	IAM
Line images: N	497	929
Query words: M	1 671	2 209
Line-query events: $M \cdot N$	830 487	2 052 161
OOV Line-query events	522 347	405 973
Relevant line-query events	4 346	3 446
Relevant OOV line-query events	1 341	496

4.3 Experimental setup

In each corpus, we trained a left-to-right HMM model, with GMM distributions on the states, for each character included in the training set. The standard embedded Baum-Welch training algorithm was used [28]. Details about preprocessing, feature extraction, number of states and mixtures in the GMM, etc. can be found in [26, 25] for the CS dataset, and also [4] for IAM.

Bi-gram LMs with standard Kneser-Ney back-off smoothing [9] were used to build the lexicon-based systems. In the case of CS, only the transcripts of the training set were used to train the LM, converting lowercase characters to uppercase. This resulted in a small vocabulary of 2 236 words, and a very large proportion of OOV query words (close to 50%). For IAM, the LM was trained using the external LBW corpus,

restricted to the 20K most frequent words (transcripts of test lines were excluded from this text training set). Using this relatively large vocabulary, the proportion of OOV keywords was quite small (about 2%).

The computation of Eq. (1) for lexicon-based indexing was carried out with the help of word graphs (WGs), obtained as a byproduct of conventional Viterbi decoding [26, 25]. WGs were generated using the described language and optical models, with a maximum node input degree (NID) value equal to 40, without any decoding pruning technique. Further details about the lexicon-based method can be found in [17, 16].

On the other hand, in order to speedup the search needed for the HMM-Filler method, the approach described in [26] was used. In a *preparatory phase*, a character-lattice (CL) was obtained for each test line image using only the "filler" HMM. The maximum NID of these CLs was set to 30. Then, during the *search phase*, the scores of each query were computed using the CLs.

We adopted the same hyperparamenter values as in the reference papers [17, 16] for the existing methods. As for our new method (Section 3.3), we tuned the required hyperparameters using the validation set for IAM, or cross-validation for CS.

Experiments were conducted on a Intel Core 2 Quad Q9550 CPU at 2.83GHz, running Ubuntu Linux 14.04. All custom software was implemented in C++.

4.4 Results

Table 2 summarizes the main results obtained on the test set of each corpora. The first row displays the performance obtained using a trivial recognition-based KWS approach where input images are explicitly decoded by means of a traditional HTR engine (based on the same HMMs and n-grams used in the rest of the experiments). In this naive approach a keyword has score 1 if it appears in the HTR transcript, or 0 otherwise. The second row shows the performance of the plain, unsmoothed lexicon-based KWS approach briefly explained in Sect. 2. Finally, the remaining rows report the performance using the different smoothing methods.

Table 2: Line-level Average Precision (AP) and Mean Average Precision (mAP) provided by different spotting methods on the test set of each corpus. Results tagged with † and ‡ were presented in [17, 16], respectively. Query time (Qtime) is the average time required to serve an OOV query, expressed seconds.

	CS			IAM		
Method	AP	mAP	Qtime	AP	mAP	Qtime
Naive, unsmoothed recognition-based KWS	37.5	19.2	—	51.3	52.4	—
Unsmoothed Lexicon-based KWS (Sect. 2)	55.6	29.0	—	69.1	68.8	—
Heuristic Line-Level smoothing (Sect. 3.1)	† 57.8	† 45.0	0.44	69.8	76.0	8.78
Heuristic Frame-Level smoothing (Sect. 3.2)	† 58.8	† 46.7	27.21	70.2	76.1	42.48
Probabilistic Line-Level smoothing (Sect. 3.3)	59.5	46.0	0.52	71.3	76.0	9.96
Lexicon-based + Lexicon-free (Sect. 3.4)	72.5	76.6	39.11	‡ 76.9	‡ 82.2	58.16

On the one hand, observe the low performance of trivial HTR-based KWS applied to both tasks. This highlights the fact that naively indexing the results of automatic HTR transcripts is, indeed, not a good idea for handwritten documents.

On the other hand, the baseline unsmoothed lexicon-based KWS approach outlined in Sect. 2 does manage to improve the results over the naive approach very significantly. However, the results can clearly be improved when the problem entailed by out-of-vocabulary queries is addressed. This is specially notorious if one looks at the mAP increase in CS, where the number of relevant OOV events is very significant (31%). In general, the smoothing improvements are smaller in IAM, given the smaller number of relevant OOV events (14%). Clearly, the need of properly dealing with OOV queries drops as the amount of indexed words becomes large.

Regarding our new Probabilistic Line-Level proposal (see Sect. 3.3), Table 2 shows that it improves the AP results in both datasets (about 1.7 points of absolute improvement), when compared to the line-level method described in Sect. 3.1, while maintaining similarly low computational costs. This is because the new method successfully uses more information from the index: the contribution of all in-vocabulary keywords is considered, rather of only a maximum.

Moreover, the new approach slightly improves the AP of the method described in Sect. 3.2 (about 0.9 points of absolute improvement), while requiring much less computing time (it is about 50 times faster in CS and 4 times faster in IAM). It is important to notice that the running time of the method of Sect. 3.2 depends not only on the size of V but also on the average length of the line images. In contrast, the new method (Sect. 3.3) works directly with lexicon-based indexed relevance probabilities and do not need the frame-by-frame computation needed to apply Eq. (7-8) for each $i,\ 1 \leq i \leq m$, where m is the length (number of frames) of the test images. Finally, the much larger size of the IAM vocabulary (roughly 10 times larger than that of CS) and the larger amount of test line images (about twice as many as in CS) explain the relative computing times needed by the different smoothing methods on both datasets.

It is worth pointing out that the computation of Levenshtein distances has been done naively in this work, with an asymptotic cost of $O(|u| \cdot L \cdot |V|)$, where $|u|$ is the number of query string characters, and $L = \max_{v \in V} |v|$. Nevertheless, this computation can be dramatically accelerated by using *tries* (i.e. prefix trees) for indexing the vocabulary, or by limiting the maximum number of insertion deletions and substitutions allowed [23].

Finally, we confirm that the combination of a Lexicon-based and a Lexicon-free (HMM-Filler) methods gives better AP results than any of the other methods considered, but at a much higher computational cost, even when the fast technique proposed in [24]) is used. As compared with our new method, the HMM-Filler approach is more than 75 times slower in CS and 6 times slower in IAM.

Two interplaying factors explain the speed-up differences between the two datasets: first, the IAM test set is more than twice as large as that of CS, which linearly affects the query time of all methods. On the other hand, the vocabulary size of IAM is almost 9 times larger than that of CS, which affects also linearly the computational costs of the smoothing methods presented in this paper, and thereby increases the corresponding differences between CS and IAM. Finally, observe that the HMM-Filler method is not affected by the vocabulary size (it is a lexicon-agnostic model).

Figure 3 shows the recall-precision curves obtained for CS and using the plain lexicon-based KWS approach with no smoothing, and the new probabilistic line-level smoothing. It clearly shows the significant precision improvement achieved for high recall, especially for the CS corpus, which explains the corresponding overall AP improvements reported in Table 2. The lower relative improvement in the IAM corpus is clearly due to the much smaller proportion of OOV queries, which allows the unsmoothed lexicon-based approach to provide good results by itself, leaving little room for improvement by OOV query processing.

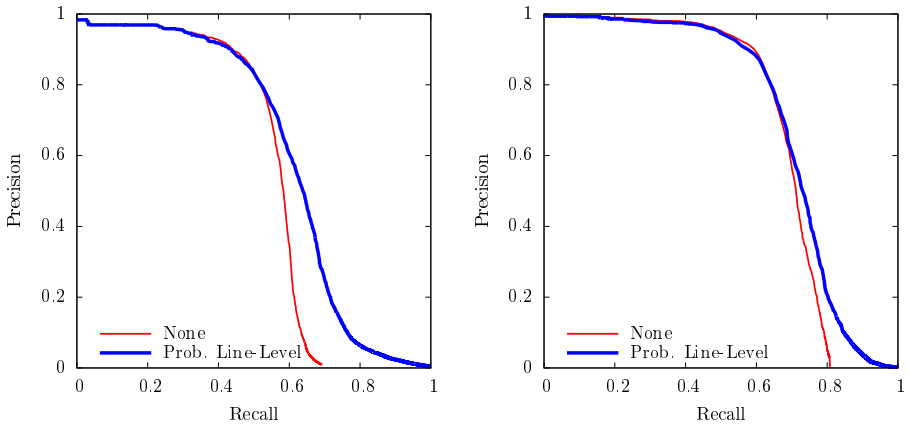

Fig. 3: Recall-Precision curves for CS (left) and IAM (right) using the unsmoothed lexicon based KWS approach, and the new probabilistic line-level smoothing.

5 Conclusions

We have presented a new method for smoothing the KWS scores given by lexicon-based systems for handwritten text images, based on the similarity between the OOV keyword and in-vocabulary words. A detailed comparison of different alternatives that aim to alleviate the OOV query problem has been also presented.

The new smoothing method gives better results than most of the compared alternatives, and it is comparable in speed to the fastest of the previous methods, also based on string similarity measures. A live demonstration of the KWS approach used in this paper can be found at `http://transcriptorium.eu/demots/demo_ncaa`, supporting the search of OOV keywords using the new technique presented in Sect. 3.3.

Only the combination of a lexicon-based system and a HMM-Filler surpasses our new proposal. Nevertheless, it comes at a time cost very much larger than that of our new proposal, which may result impractical in most real scenarios (even in those involving data sets as small as the CS corpus).

In the future, we plan to investigate ways of effectively indexing open-lexicon character-level KWS scores and combine them with lexicon-based approaches, thus allowing for faster and more accurate searches for both in-vocabulary and out-of-vocabulary queries.

Acknowledgments

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

References

1. Almazan, J., Gordo, A., Fornes, A., Valveny, E.: Handwritten word spotting with corrected attributes. In: Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 1017–1024 (2013). DOI 10.1109/ICCV.2013.130
2. Amengual, J.C., Vidal, E.: On the estimation of error-correcting parameters. In: Pattern Recognition, 2000. Proceedings. 15th International Conference on, vol. 2, p. 883?886 (2000)
3. Fernández, D., Lladós, J., Fornés, A.: Handwritten word spotting in old manuscript images using a pseudo-structural descriptor organized in a hash structure. In: Pattern Recognition and Image Analysis, pp. 628–635. Springer (2011)
4. Fischer, A., Keller, A., Frinken, V., Bunke, H.: Lexicon-free handwritten word spotting using character HMMs. Pattern Recognition Letters **33**(7), 934 – 942 (2012). DOI 10.1016/j.patrec.2011.09.009. Special Issue on Awards from ICPR 2010
5. Fornés, A., Frinken, V., Fischer, A., Almazán, J., Jackson, G., Bunke, H.: A keyword spotting approach using blurred shape model-based descriptors. In: Proceedings of the 2011 Workshop on Historical Document Imaging and Processing, pp. 83–90. ACM (2011)
6. Frinken, V., Fischer, A., Manmatha, R., Bunke, H.: A Novel Word Spotting Method Based on Recurrent Neural Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(2), 211 –224 (2012). DOI 10.1109/TPAMI.2011.113
7. Gatos, B., Pratikakis, I.: Segmentation-free word spotting in historical printed documents. In: 10th International Conference on Document Analysis and Recognition, 2009. ICDAR'09., pp. 271–275. IEEE (2009)
8. Jelinek, F.: Statistical Methods for Speech Recognition. MIT Press (1998)
9. Kneser, R., Ney, H.: Improved backing-off for N-gram language modeling. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP '95), vol. 1, pp. 181–184. IEEE Computer Society, Los Alamitos, CA, USA (1995). DOI http://doi.ieeecomputersociety.org/10.1109/ICASSP.1995.479394
10. Kolcz, A., Alspector, J., Augusteijn, M., Carlson, R., Viorel Popescu, G.: A Line-Oriented Approach to Word Spotting in Handwritten Documents. Pattern Analysis & Applications **3**, 153–168 (2000). 10.1007/s100440070020
11. Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., Perantonis, S.J.: Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. Int. Journal of Document Analysis and Recognition **9**(2-4), 167–177 (2007)
12. Kumar, G., Govindaraju, V.: Bayesian active learning for keyword spotting in handwritten documents. In: Pattern Recognition (ICPR), 2014 22nd International Conference on, pp. 2041–2046. IEEE (2014)
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. In: Soviet physics doklady, vol. 10, p. 707 (1966)
14. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)

15. Marti, U.V., Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition **5**(1), 39–46 (2002). DOI 10.1007/s100320200071. URL http://dx.doi.org/10.1007/s100320200071
16. Puigcerver, J., Toselli, A.H., Vidal, E.: Word-Graph and Character-Lattice Combination for KWS in Handwritten Documents. In: 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 181–186 (2014)
17. Puigcerver, J., Toselli, A.H., Vidal, E.: Word-Graph-based Handwriting Keyword Spotting of Out-of-Vocabulary Queries. In: 22nd International Conference on Pattern Recognition (ICPR), pp. 2035–2040 (2014)
18. Puigcerver, J., Toselli, A.H., Vidal, E.: A new smoothing method for lexicon-based handwritten text keyword spotting. In: 7th Iberian Conference on Pattern Recognition and Image Analysis. Springer (2015)
19. Rath, T., Manmatha, R.: Word spotting for historical documents. International Journal on Document Analysis and Recognition **9**, 139–152 (2007)
20. Robertson, S.: A new interpretation of average precision. In: Proc. of the Intl. ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '08), pp. 689–690. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1390334.1390453
21. Rodriguez-Serrano, J.A., Perronnin, F.: Handwritten word-spotting using hidden markov models and universal vocabularies. Pattern Recognition **42**(9), 2106–2116 (2009). DOI http://dx.doi.org/10.1016/j.patcog.2009.02.005. URL http://www.sciencedirect.com/science/article/pii/S0031320309000673
22. Rusinol, M., Aldavert, D., Toledo, R., Llados, J.: Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 63 –67 (2011). DOI 10.1109/ICDAR.2011.22
23. Shang, H., Merrettal, T.: Tries for approximate string matching. Knowledge and Data Engineering, IEEE Transactions on **8**(4), 540–547 (1996)
24. Toselli, A.H., Vidal, E.: Fast HMM-Filler approach for Key Word Spotting in Handwritten Documents. In: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 501–505 (2013)
25. Toselli, A.H., Vidal, E.: Word-Graph Based Handwriting Key-Word Spotting: Impact of Word-Graph Size on Performance. In: 11th IAPR International Workshop on Document Analysis Systems (DAS), pp. 176–180. IEEE (2014)
26. Toselli, A.H., Vidal, E., Romero, V., Frinken, V.: Word-Graph Based Keyword Spotting and Indexing of Handwritten Document Images. Tech. rep., Universitat Politécnica de Valéncia (2013)
27. Vidal, E., Toselli, A.H., Puigcerver, J.: High performance query-by-example keyword spotting using query-by-string techniques. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, pp. 741–745. IEEE (2015)
28. Woodland, P., Leggetter, C., Odell, J., Valtchev, V., Young, S.: The 1994 HTK large vocabulary speech recognition system. In: Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP '95), vol. 1, pp. 73 –76 vol.1 (1995). DOI 10.1109/ICASSP.1995.479276
29. Wshah, S., Kumar, G., Govindaraju, V.: Script independent word spotting in offline handwritten documents based on hidden markov models. In: Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on, pp. 14–19 (2012). DOI 10.1109/ICFHR.2012.264