Word-Graph-based Handwriting Keyword Spotting of Out-of-Vocabulary Queries

Joan Puigcerver
PRHLT Research Center
Universitat Politècnica de València
Camí de Vera s/n
46022 - València - Spain
Email: jpuigcerver@dsic.upv.es

Alejandro Héctor Toselli
PRHLT Research Center
Universitat Politècnica de València
Camí de Vera s/n
46022 - València - Spain
Email: ahector@dsic.upv.es

Enrique Vidal
PRHLT Research Center
Universitat Politècnica de València
Camí de Vera s/n
46022 - València - Spain
Email: evidal@dsic.upv.es

***Abstract*—Thanks to the use of lexical and syntactic information, Word Graphs (WG) have shown to provide a competitive Precision-Recall performance, along with fast lookup times, in comparison to other techniques used for Key-Word Spotting (KWS) in handwritten text images. However, a problem of WG approaches is that they assign a null score to any keyword that was not part of the training data, i.e. Out-of-Vocabulary (OOV) keywords, whereas other techniques are able to estimate a reasonable score even for these kind of keywords. We present a smoothing technique which estimates the score of an OOV keyword based on the scores of similar keywords. This makes the WG-based KWS as flexible as other techniques with the benefit of having much faster lookup times.**

I. Introduction

Handwritten Keyword Spotting (KWS) aims to determine, with a pre-specified confidence level, whether a given keyword is written in a document image or image region, or it is not. Recently, Word-Graphs (WG) have been used for KWS in handwritten text images [1], [2]. The WGs used in this approach are produced by Viterbi-decoding text line images using morphological, lexical and language models previously trained for the handwriting task in hand. WG-based KWS spotting scores can be easily used to create an index which allows for extremely fast, confidence-level controlled look-up for indexed words. Moreover, the Precision-Recall performance of this KWS model has been shown to be competitive in comparison with other lexicon-agnostic methods, such as the HMM-*filler* approach [3], [4], and comparable with that of BLSTM KWS, which is perhaps the best handwritten KWS method currently, if its very high training costs are not taken into account.

One disadvantage of WG-based KWS is that any keyword that it is not included in a WG will be given a null score, and thus, it becomes completely useless for out-of-vocabulary (OOV) keywords (that is, words that were not included in the trained models). On the other hand, the density or *maximum node input degree* (NID, which specifies the amount of information retained at each node during the WG generation process) also affects the performance of this method – even though, as shown in [5], a reasonably good performance can be achieved with relatively small WGs with a minor loss of the useful retained information.

Lexicon-free approaches such as HMM-*filler* or BLSTM, however, do not suffer from this problem. But, lacking a lexicon, they do not lend themselves adequate for direct word indexing. Therefore, the search time needed by these approaches often becomes prohibitive for large collections of handwritten images (for one million images, for instance, a single keyword query could require days or weeks of intensive computing).

According to this state of affairs, practical applications involving large image collections call for using a hierarchy of spotting methods. In the first level, a lexicon-based index (obtained by means of WG KWS) provides fast and accurate spotting results for usual queries. Then, for non-indexed keywords, an affordable method is needed which provides at least some reasonable spotting results, even at the cost of producing many false alarms (i.e., ensure a high recall even though the precision is low). Finally lexicon-free methods could be used in exceptional cases, when the user is highly interested in spotting a specific (non-indexed) keyword and can afford waiting for days or weeks to obtain reasonably accurate KWS results with not too many false alarms.

This paper explores smoothing techniques to deal with the intermediate level discussed above. The smoothed score for an OOV keyword is based on the word-posterior probabilities of other words which are included in the WG, as well as on the similarity between these keywords and the query word. The similarity is modeled by means of a stochastic error-correcting analysis, which provides a *similarity probability* of an (unknown) word, given another (known) keyword.

The support of OOV keywords for a real KWS system is fundamental, since it is very likely that, over time, a considerable amount of queries will fall into this category. Therefore, WG-based KWS approaches must provide some solution for these keywords.

The paper is organized as follows: Sec. II introduces basic concepts of HMM-based Handwriting Text Recognition (HTR) and WGs, the basis framework used by the WG-based KWS approach. Sec. III presents the WG-based KWS approach used in this work. The proposed smoothing methods are presented in Sec. IV. Performance assessment metrics, corpora, experimental setup and results are reported in Sec. V. Finally, Sec. VI summarizes this work and draws conclusions and future lines of research.

II. HTR FRAMEWORK USING HMMs AND N-GRAMS

This work is framed in a line-level KWS scenario, and thus, a line-level HTR framework is used as basis. Text line images can be obtained from each document image using well-known text line detection and segmentation algorithms [6]. A sequence of feature vectors is obtained after applying different normalization and feature extraction techniques. Fig. 1 shows an overview of the HTR decoding process. More details about the different modules can be found in [7].

HMMs and N-grams are widely used models in the HTR field, which were originally introduced in [8] and developed with additional details in [9], [10], among other works. They are used to estimate the most likely word sequence, $\widehat{\mathbf{w}} = \widehat{w}_1\widehat{w}_2\ldots\widehat{w}_l$, represented by a given text line image, encoded as a sequence of d-dimensional feature vectors $\mathbf{x} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n, \vec{x}_i \in \mathbb{R}^d$, according to Eq. (1).

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} P(\mathbf{w} \mid \mathbf{x}) = \arg\max_{\mathbf{w}} p(\mathbf{x} \mid \mathbf{w}) \cdot P(\mathbf{w}) \qquad (1)$$

The conditional distribution $p(\mathbf{x} \mid \mathbf{w})$ is modeled by the concatenation of several character HMMs [11], [12], and the prior $P(\mathbf{w})$ is modeled using a N-gram language model [11].

The Viterbi algorithm [11] can be used to solve Eq. (1). A huge set of best hypotheses for $\widehat{\mathbf{w}}$, along with the corresponding word segmentations and likelihoods, can be obtained as a byproduct of the Viterbi decoding as well, in form of a WG.

A WG of a vector sequence $\mathbf{x}$ is a weighted directed acyclic graph $G(\mathbf{x}) = (Q, q_0, F, \tau, \omega, \delta)$, with initial node $q_0 \in Q$ and a set of final nodes $F \subseteq (Q - \{q_0\})$. Each node q has associated an integer given by $\tau(q_i) \in [0, n]$ (where n is the length of the sequence $\mathbf{x}$). For every edge $(q, q') \in E$ $(q \neq q', q \notin F, q' \neq q_0)$, the application $\omega(q, q') = v$ associates a word v to the edge and $\delta(q, q')$ is a score, corresponding to the likelihood that the word $\omega(q, q')$ is written in the *image segment* delimited by frames $x_{\tau(q)+1}, \ldots, x_{\tau(q')}$. These likelihoods are given by the decoding process. Fig. 2 shows an illustrative example of a normalized WG.

As mentioned above, WGs contain a huge set of possible decoding hypotheses for $\mathbf{x}$. Each of these hypotheses is encoded as a *complete path* in the WG, which is any sequence of connected nodes starting with the initial node q_0 and ending with one of the final nodes $q_F \in F$.

III. WG-BASED KWS FOR HANDWRITTEN TEXT LINES

This work uses the KWS approach for handwritten text line images presented in [1]. For each keyword v and each text line represented by $\mathbf{x}$, a score $S(v, \mathbf{x})$ is computed according to Eq. (2), which measures the confidence of the system about the statement "keyword v is in line $\mathbf{x}$" (scores are given in the range [0,1], with 1 being a high confidence and 0 a low confidence).

$$S(v, \mathbf{x}) \overset{\text{def}}{=} \max_{1 \leq i \leq n} P(v \mid \mathbf{x}, i) \qquad (2)$$

In the previous equation, $P(v \mid \mathbf{x}, i)$ is referred as the *frame-level word posterior*, which is the probability that the word v is present in the line image at position i (the index of the feature vector $\mathbf{x}$). This probability can be directly approximated from $G(\mathbf{x})$, as shown in [1], [2], considering the contribution of all the edges labeled with the keyword v, i.e. all the segmentation hypotheses that include the frame i. Eq. (3) shows how to compute this probability, where $\alpha(\cdot)$ is the *forward* and $\beta(\cdot)$ *backward* accumulated path scores which can be efficiently computed on the WGs using dynamic programming [13], [1].

$$P(v \mid i, \mathbf{x}) \approx \sum_{\substack{(q,q') \in E: \\ v = \omega(q,q'), \\ \tau(q) < i \leq \tau(q')}} \frac{\alpha(q) \cdot s(q, q') \cdot \beta(q')}{\beta(q_0)} \qquad (3)$$

IV. WORD-GRAPH SMOOTHING FOR OOV KEYWORDS

A. Score Smoothing with Levenshtein Distance

First, we tried to give a score for an OOV keyword directly from the scores of the other keywords which are included in $G(\mathbf{x})$, using the Levenshtein distance as a similarity metric between the OOV keyword and the keywords included in the WG (the set $V_{G(\mathbf{x})}$). Thus, the score of an OOV keyword u for a line image represented by $\mathbf{x}$ is computed as:

$$S(u, \mathbf{x}) \overset{\text{def}}{=} \max_{v \in V_{G(\mathbf{x})}} S(v, \mathbf{x})^{1-\alpha} e^{-\alpha d(u,v)} \qquad (4)$$

In the previous equation, $d(u, v)$ refers to the Levenshtein distance between keywords u, v. The parameter α is used to fine-tune the importance of the similarity measure, in the same way that the *grammar scale factor* is used to tune the importance of the LM in HTR and Automatic Speech Recognition (ASR). We use this method as a baseline to compare the proposed smoothing method, introduced below.

B. Frame-level Word Posterior Smoothing

Consider the frame-level word posterior probability $P(u \mid \mathbf{x}, i)$ of a keyword u which is not present in any edge of $G(\mathbf{x})$, e.g. an OOV query. Then, $P(u \mid \mathbf{x}, i) = 0$ for any frame i, and thus $S(u, \mathbf{x}) = 0$. However, we would like to give some non-zero score to this keyword. The posterior probability $P(u \mid \mathbf{x}, i)$ can be marginalized among all keywords in $V_{G(\mathbf{x})}$ as Eq. (5) indicates.

$$P(u \mid \mathbf{x}, i) = \sum_{v \in V_{G(\mathbf{x})}} P(u, v \mid \mathbf{x}, i) = \sum_{v \in V_{G(\mathbf{x})}} P(v \mid \mathbf{x}, i) P(u \mid v, \mathbf{x}, i) \qquad (5)$$

By assuming that u is conditionally independent of $\mathbf{x}$ and i, given v, Eq. (5) can be approximated as:

$$P(u \mid \mathbf{x}, i) \approx \sum_{v \in V_{G(\mathbf{x})}} P(v \mid \mathbf{x}, i) P(u \mid v) \qquad (6)$$

Observe that $P(u \mid \mathbf{x}, i)$ and $P(v \mid \mathbf{x}, i)$ are actually different distributions, since keyword u and v come from different random variables: $u \in \Sigma^*$ and $v \in V_{G(\mathbf{x})}$, where Σ is the set of symbols of the language and Σ^* is the set of any number of concatenations of symbols in Σ (i.e. any string formed by symbols from Σ plus the empty string). Thus, $P(u \mid \mathbf{x}, i) + \sum_{v \in V_{G(\mathbf{x})}} P(v \mid \mathbf{x}, i) \geq 1$. Equality holds if and only if $P(u \mid \mathbf{x}, i) = 0$. However, we combine $P(u \mid \mathbf{x}, i)$ and $P(v \mid \mathbf{x}, i)$ into a single distribution by adding the event u to the set of possible outcomes from $V_{G(\mathbf{x})}$ and re-normalizing

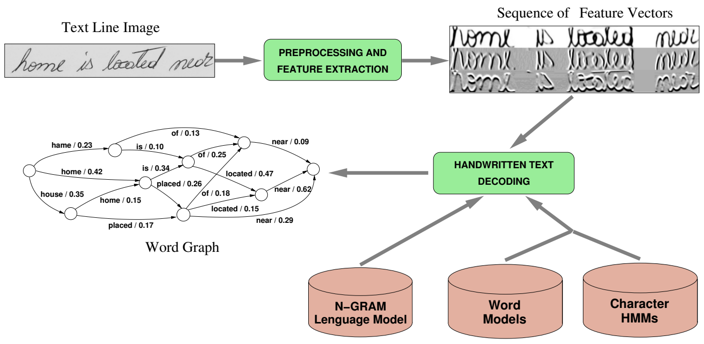


Fig. 1. Diagram of the HTR decoding process. The input image containing the text "home is located near" is processed and different features are obtained: average grey-level and horizontal & vertical components of the grey-level gradient. The decoding step uses character HMMs, lexicon word models and N-gram language models to build the corresponding WG.

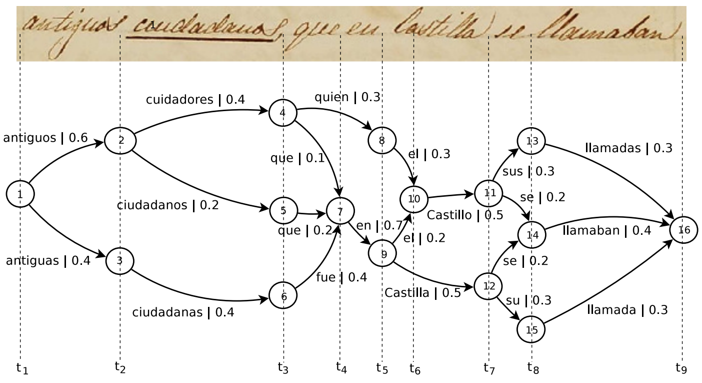


Fig. 2. Illustrative, simplified example of a *normalized* WG that would be obtained from the decoding of a line image, $\mathbf{x}$, of the Spanish handwritten text: "antiguos ciudadanos, que en Castilla se llamaban", represented by its sequence of feature vectors of length $n = |\mathbf{x}|$. Each edge (q, q') is labeled with the corresponding word, $\omega(q, q')$, and weighted with its "edge posterior", $\delta(q, q')$. Node positions, $t_q \equiv \tau(q)$, corresponding to different word segmentations, are also shown on the bottom and in the image itself.

the distribution so that $\sum_{v \in V_{G(\mathbf{x})} \cup \{u\}} \tilde{P}(v|\mathbf{x}, i) = 1$. In the case of the smoothed word posterior for an OOV keyword u, this results in Eq. (7).

$$\tilde{P}(u|\mathbf{x}, i) = \frac{\sum_{v \in V_{G(\mathbf{x})}} P(v|\mathbf{x}, i) P(u|v)^{\alpha}}{1 + \sum_{v \in V_{G(\mathbf{x})}} P(v|\mathbf{x}, i) P(u|v)^{\alpha}} \tag{7}$$

As before, α is used to fine-tune the importance of the similarity measure.

Finally, the score for an OOV keyword u and the text line image $\mathbf{x}$ is obtained using Eq. (7) in Eq. (2).

C. Similarity Probability

In order to compute the smoothed frame-level word posterior $\tilde{P}(u|i, \mathbf{x})$ the conditional probability $P(u|v)$ (referred to as *similarity probability*), is needed. This conditional probability is modeled using the traditional approach in stochastic error correction, presented in [14]. This approach uses the classical definition of insertion, deletion and substitution operations between two string. For each string $v = v_1 \cdots v_m$, a stochastic finite-state machine (FSM) is built with states $q_0, q_1, \ldots, q_m$, 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^*. Fig. 3 shows the FSM defined for the string aab.

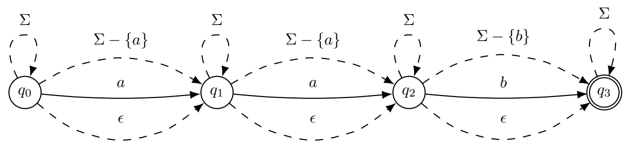


Fig. 3. FSM used to compute $P(u|\text{"aab"}), \forall u \in \Sigma^*$. Erroneous edit operations are represented by dashed lines.

Edges in the FSM have associated weights which allow us to compute $P(u|v)$ by means of a forward-like algorithm. Instead of using the originally proposed Baum-Welch algorithm to estimate the weights of the edges in the FSM, the frequencies of the corresponding edit operations given by the minimum Levenshtein distance alignment [15] of a training set of pairs of strings are used (a confusion matrix is obtained with counts of how many times symbol a was substituted by b, including insertions and deletions).

D. Length-independent correction

A well known problem when modeling $P(u|v)$ with a FSM is that the greater the length of v, the lower $\max_{u \in \Sigma^*} P(u|v)$. This behavior is not desirable, since the score $S(v, \mathbf{x})$ happens to be highly dependent of the length of both u and v, no matter which line is considered. We observed that long keywords tended to have much lower scores systematically, and thus, we decided to change the way of smoothing $\tilde{P}(u|\,\mathbf{x},i)$, so that the length of the keyword does not affect that much. We define a factor $f(u,v)$ as follows and we substitute Eq. (7) with Eq. (9).

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

$$\tilde{P}(u|\,\mathbf{x},i) = \frac{\sum_{v \in V_{G(\mathbf{x})}} P(v|\,\mathbf{x},i) f(u,v)^{\alpha}}{1 + \sum_{v \in V_{G(\mathbf{x})}} P(v|\,\mathbf{x},i) f(u,v)^{\alpha}} \tag{9}$$

V. Experiments

Several experiments were conducted to assess the performance of the proposed smoothing method when it is used to search for OOV keywords, using different WG sizes. Performance assessment, corpora, experimental setup and results are presented below.

A. Performance assessment

We use the *average precision* (AP) [16] metric, based on the standard *recall* and *interpolated precision* measures. Interpolated precision is widely used to avoid cases where plain precision can be ill-defined [17]. Precision and recall are functions of a threshold used to determine whether the score $S(v,\mathbf{x})$ is high enough to assume that v is relevant in $\mathbf{x}$ (i.e. v appears in $\mathbf{x}$). This function is summarized into a single scalar measure, the area under the *recall-precision curve*, which takes into account all possible thresholds.

In addition, we use the *mean average precision* (MAP), which is also very often adopted in the KWS literature. It is computed by averaging the *average precision* of each keyword. While it is often claimed to better reflect the real user search experience, it has the problem that it is not well defined in the case of non-relevant queries, thus the AP is here preferred, in particular to fine-tune KWS hyper-parameters.

B. Corpora

The "Cristo-Salvador" (CS) corpus was used to carry out the experiments. This is a XIX century Spanish manuscript kindly provided by the *Biblioteca Valenciana Digital*[1] (BiVaLDi). It is composed by 50 color images of 5.1×7.2 inches of relevant text pages, written by a single writer and scanned at 300dpi. The CS corpus and directions for its usage in HTR experiments is publicly available for research purposes[2].

The first 29 pages (675 lines) are used as a training set and the test set contains the remaining 21 pages (497 lines), many of which are formed by notes and short comments written in a style rather different from that of the training pages. Tab. I summarizes this information. Word statistics were computed ignoring capitalization and punctuation marks.

[1]http://bv2.gva.es

[2]https://prhlt.iti.upv.es/page/data

TABLE I. Basic statistics of the Cristo-Salvador dataset

	Training	Test	Total
Running Chars	35,863	26,353	62,216
Running Words	6,223	4,637	10,860
Running OOV(%)	–	29.03	–
# Lines	675	497	1,172
# Pages	29	21	50
Char Lex. Size	78	78	78
Word Lex. Size	2,236	1,671 (1,051 OOV)	3,287
OOV Lex. Size	–	1,051	–
Events (Lex. $\times$ # Lines)	–	830,487	–
OOV Events (OOV $\times$ # Lines)	–	522,347	–
Relevant Events	–	4,346	–
Relevant OOV Events	–	1,341	–

C. Experimental setup

The CS line images of the training partition were used to train the character HMMs using the standard embedded Baum-Welch training algorithm [11], [18]. A left-to-right HMM was trained for each of the 78 elements appearing in the training text images, such as lowercase and uppercase letters, symbols, special abbreviations, possible spacing between words and characters, crossed-out words, etc. On the other hand, a lexicon and a bi-gram language model (LM) was trained from the transcripts of the line images of this training partition.

Since CS does not have a standard validation set, a 10-fold cross-validation of the training set was adopted to tune all the system hyper-paramentes. The same scheme was used to estimate the character ins-subs-del probabilities of the proposed error model, $P(u|v)$.

For each of the validation partitions, HMMs and LMs where trained independently and tested in order to tune the hyper-parameters of the HTR feature extraction module (e.g. frame width and height) and the HMMs models parameters (number of states for the HMMs, grammar scale factor, word-insertion penalty, etc). Bi-gram LMs where trained converting uppercase character to lowercase and eliminating punctuation symbols and diacritics. Finally, the standard Kneser-Ney back-off technique was used to smooth the probabilities of unseen bi-grams of the LM [19].

The best HTR accuracy was observed when the original line image was segmented into $m = 16$ rows and a number of columns, n, chosen so that $\frac{m}{n}$ was *three* times the original line image aspect ratio. Thus, each line image was represented as a sequence of n vectors (frames) of 48 dimensions, with an average number of frames equal to $\overline{n} = 1659$. The optimal number of states per HMM was 14, with 16 Gaussian densities per state.

For each validation set, the ins-del-sub probabilities were estimated by means of a confusion matrix obtained by computing the frequency of each character, a, being substituted by character, b, including the case of a or b being the empty-string symbol. These frequencies were determined from the Levenshtein-distance alignment between the 1-best decoding in each validation partition and the corresponding ground-truth transcript, as explained in Sec. IV-C.

In order to perform KWS experiments, the vocabulary in each of the validation partitions was used as the query set during cross-validation to tune the KWS smoothing hyper-parameters (α). Once these were tuned, the final performance was measured using the test vocabulary as the query set for KWS on the test lines. The proposed smoothing methods were only used for OOV keywords, whereas in-vocabulary keywords used the score given by the baseline approach.

During cross-validation, values of α in the range $[0,1]$ with increments of 0.1 units were tried for different WG sizes (NID values of $1, 3, 5, 10, 20, 40$). The AP was computed and averaged across the 10 validation partitions to obtain the best α value for each smoothing method and each NID value.

In principle, a different α value could be used for each NID value. However, since the confidence intervals were highly overlapping for some values, the α which provided the highest average improvement across all NID values was selected, for each smoothing method. In the case of Eq. (4), the chosen value of α was 0.8, and for both Eq. (7) and Eq. (9), it was 0.6.

Once all hyper-parameters were tuned by means of cross-validation over the training data, the character HMMs, the lexicon and the LM were trained again using the whole training set. The confusion matrix used to compute $P(u|v)$ in the test experiments, was built averaging the frequencies from the ten validation partitions.

Finally, for each test line image, six WGs were obtained for the following NID values: 1, 3, 5, 10, 20 and 40. All these WGs were generated using the HTK toolkit [18].

D. Results

Tab. II summarizes the empirical results. For each NID, Eq. (4), Eq. (7) and Eq. (9) where used to approximate $S(u, \mathbf{x})$ as described earlier, for OOV queries.

TABLE II. RESULTS ON THE TEST LINES USING DIFFERENT SMOOTHING METHODS AND INPUT DEGREE. THE USED QUERY SET IS THE WHOLE TEST VOCABULARY.

	Mth. / NID	1	3	5	10	20	40
AP (%)	None	38.9	53.6	55.0	55.4	55.6	55.6
	Eq. (4)	41.5	56.3	57.5	57.7	57.9	57.8
	Eq. (7)	43.0	56.6	57.7	57.9	58.0	58.0
	Eq. (9)	43.5	57.1	58.3	58.6	58.8	58.8
MAP (%)	None	19.8	27.5	28.1	28.7	29.0	29.0
	Eq. (4)	28.3	42.8	43.7	44.6	44.8	45.0
	Eq. (7)	32.0	44.7	45.7	46.2	46.6	46.7
	Eq. (9)	32.0	44.4	45.5	46.2	46.6	46.7

First, observe that the baseline MAP is much lower than the AP (29.0% vs. 55.6%, for a NID of 40). This can be explained because the percentage of OOV keywords is very large (almost 63%, see Tab. I) and the AP of all these is close to zero (i.e. the prior probability of relevant events, 0.5%), which results in a low MAP when averaging the individual AP across all keywords. On the other hand, the percentage of relevant events due to OOV is smaller (around 31%, see Tab. I), and thus, in this particular scenario, the AP is less affected by the null scores assigned to OOV keywords.

All proposed smoothing methods bring an improvement of the global performance over the baseline WG-based KWS system. Moreover, during cross-validation, the confidence intervals at 95% for the AP metric were around $\pm 1.1\%$ for all methods. In the case of the MAP metric, these confidence intervals were $\pm 0.8\%$. Thus, assuming that the confidence intervals are similar on the test set, the improvement brought by all the smoothing methods is statistically significant for both AP and MAP metrics.

The frame-level smoothing methods (Eq. (7) and Eq. (9)) offer a significant improvement over the simple line-level smoothing (Eq. (4)), when the MAP metric is considered. However, in terms of AP, the achieved improvement is in the borderline of statistical significance.

Observe that the proposed length-independence error model (Sec. IV-D) is critical to achieve the best possible results, in terms of AP, using the frame-level smoothing. For instance, in the case of the biggest word-graphs, the score-level smoothing offers an AP of 57.8% and the basic frame-level smoothing offers 58.0%. However, this correction boosts the AP to 58.8%. Also, it does not affect the MAP metric very much (46.7% in both frame-level cases).

This is due to the fact that the proposed stochastic similarity measure suffers from the problem stated in Sec. IV-D: the greater the length of a keyword v is, the lower its maximum *similarity probability* is, and so, the resulting KWS scores. Thus, when all events are considered together in order to compute the AP metric, relevant events of long keywords may have a lower score than other shorter keywords, which damages the AP. This effect is diminished when using the proposed correction. MAP is not affected by this, since it only considers events from a single keyword, and then averages the results.

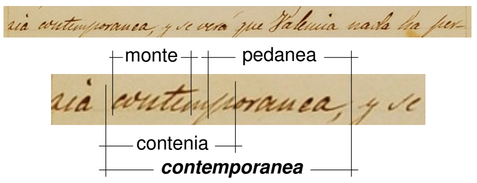
Finally, as a demonstration, Fig. 4(a) and Fig. 4(b) show two lines spotted as result of two different OOV queries. The line in Fig. 4(a) obtained the highest score for the keyword "contemporanea", which is the true word in the image. Other OOV keywords have much lower scores in that line, as expected. However, Fig. 4(b) shows an example of a false positive where the keyword "observar" got a higher score than the correct keyword "observo" (both OOV), but they are actually very close and much higher than other OOV keywords.

VI. Conclusions

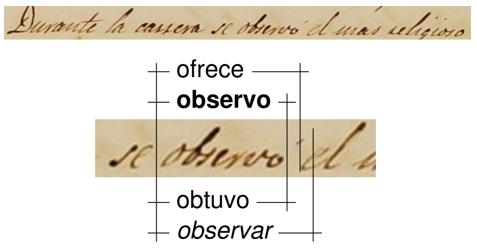
Three different methods for smoothing the score of OOV keywords in WG-based KWS have been presented. One of the methods is based on smoothing directly the final line-level scores given by the baseline WG-based method, while the other two methods go one step beyond and use the *frame-level word-posterior* of the words included in the WGs and a similarity between these words and the queried keyword, using a stochastic error correction model.

All three methods significantly improve the global performance over the baseline system. The line-level smoothing offers a relative improvement of 4% in AP and 55% in MAP, while the best frame-level smoothing method offers a relative improvement of 6% in AP and 61% in MAP, bringing the system to usable levels.

One of the benefits of this smoothing technique is that it is *free* for indexed keywords, in the sense that it does not affect the performance on them. Since the search engine stores a list of indexed keywords, it can decide whether to apply the



(a) $S(\text{“contemporanea”}, \mathbf{x}) = 0.0828$, $S(\text{“monte”}, \mathbf{x}) = 2.03 \cdot 10^{-7}$, $S(\text{“contenia”}, \mathbf{x}) = 8 \cdot 10^{-9}$, $S(\text{“pedanea”}, \mathbf{x}) = 10^{-9}$. AP = 100%.



(b) $S(\text{“observar”}, \mathbf{x}) = 0.1168$, $S(\text{“observo”}, \mathbf{x}) = 0.0719$, $S(\text{“ofrece”}, \mathbf{x}) = 3.86 \cdot 10^{-5}$, $S(\text{“obtuvo”}, \mathbf{x}) = 1.25 \cdot 10^{-6}$. AP = 50%.

Fig. 4. Smoothed scores of different OOV keywords using Eq. (9). The query is marked in italic, and the ground-truth in bold. Fig. 4(a) shows an example where the smoothing worked as expected. In Fig. 4(b), the score of the query was over-estimated. The AP for each query is also shown.

proposed smoothing or not, so the gain in performance on OOV keywords is neat and independent from other keywords.

On the other hand, the introduced smoothing methods provide much faster lookup times than using a *filler* model. However the latter typically offers a better performance for OOV, suggesting that a mixed approach using WGs for regular keywords, and different smoothing techniques of WGs and the *filler* models for OOV could be used to provide the user with different degrees of speed and accuracy. Exploring such a KWS architecture constitutes an interesting line for further research.

Other lines of future work have arisen from these experiments, for instance, improving the training of the *similarity probability* $P(u|v)$ by using an EM training algorithm, as proposed in [14]. Also, a better modeling of this similarity probability should be investigated, since it is too sensitive to the length of both u and v, which causes undesirable effects on the KWS scores. The proposed solution is just a heuristic and a fundamentally better model should be pursued.

Finally, the proposed approach could be used to smooth not only the OOV keywords, but more generally, the score of any keyword which is not present in the WGs (for instance, when such keyword is pruned due to the value of the NID parameter). This could be very useful to reduce the size of WGs and thus, improving the indexing times of the proposed method, since the WG generation time is one of the main drawbacks of the WG-based KWS approach, which is mainly affected by the NID parameter, according to [5].

Acknowledgments

This work was partially supported by the Spanish MEC under the STraDA research project (TIN2012-37475-C02-01), by the Generalitat Valenciana under the grant Prometeo/2009/014, and through the EU 7th Framework Programme grant tranScriptorium (Ref:600707).

References

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

[2] ——, “Word-Graph Based Keyword Spotting in Handwritten Document Images,” 2013, under review.

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

[4] A. H. Toselli and E. Vidal, “Fast HMM-Filler approach for Key Word Spotting in Handwritten Documents,” in *International Conference on Document Analysis and Recognition (ICDAR'13)*, Aug. 2013, accepted for publication.

[5] ——, “Word-Graph based Handwriting Key-word Spotting: Impact of Word-Graph Size on Performance,” 2013, accepted to be presented in DAS.

[6] L. Likforman-Sulem, A. Zahour, and B. Taconet, “Text line segmentation of historical documents: a survey,” *International Journal on Document Analysis and Recognition*, vol. 9, pp. 123–138, April 2007.

[7] V. Romero, A. H. Toselli, and E. Vidal, *Multimodal Interactive Handwritten Text Transcription*, ser. Series in Machine Perception and Artificial Intelligence (MPAI). World Scientific Publishing, 2012, http://www.worldscientific.com/worldscibooks/10.1142/8394.

[8] I. Bazzi, R. Schwartz, and J. Makhoul, “An Omnifont Open-Vocabulary OCR System for English and Arabic,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 495–504, 1999.

[9] A. Vinciarelli, S. Bengio, and H. Bunke, “Off-line recognition of unconstrained handwritten texts using HMMs and statistical language models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 709–720, june 2004.

[10] A. H. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, and F. Casacuberta, “Integrated Handwriting Recognition and Interpretation using Finite-State Models,” *Internationa Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 4, pp. 519–539, 2004.

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

[12] L. Rabiner, “A Tutorial of Hidden Markov Models and Selected Application in Speech Recognition,” *Proceedings IEEE*, vol. 77, pp. 257–286, 1989.

[13] 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.

[14] J.-C. Amengual and E. Vidal, “On the estimation of error-correcting parameters,” in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2, 2000, p. 883?886.

[15] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” in *Soviet physics doklady*, vol. 10, 1966, p. 707.

[16] S. Robertson, “A new interpretation of average precision,” in *Proc. of the International ACM SIGIR conference on Research and development in information retrieval (SIGIR '08)*. New York, NY, USA: ACM, 2008, pp. 689–690.

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

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

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