Word Extraction Associated with a Confidence Index for On-Line Handwritten Sentence Recognition

Solen Quiniou, François Bouteruche, Eric Anquetil

International Journal of Pattern Recognition and Artificial Intelligence

WORD EXTRACTION ASSOCIATED WITH A CONFIDENCE INDEX FOR ON-LINE HANDWRITTEN SENTENCE RECOGNITION

SOLEN QUINIOU, FRANÇOIS BOUTERUCHE and ERIC ANQUETIL

IRISA - INSA de Rennes,
Campus universitaire de Beaulieu
35042 Rennes, France
{*Solen.Quiniou, Eric.Anquetil*}*@irisa.fr*
Francois.Bouteruche@evodia.fr
http://www.irisa.fr/imadoc

This paper presents a word extraction approach based on the use of a confidence index to limit the total number of segmentation hypotheses in order to further extend our on-line sentence recognition system to perform "on-the-fly" recognition. Our initial word extraction task is based on the characterization of the gap between each couple of consecutive strokes from the on-line signal of the handwritten sentence. A confidence index is associated to the gap classification result in order to evaluate its reliability. A reconsideration process is then performed to create additional segmentation hypotheses to ensure the presence of the correct segmentation among the hypotheses. In this process, we control the total number of segmentation hypotheses to limit the complexity of the recognition process and thus the execution time. This approach is evaluated on a test set of 425 English sentences written by 17 writers, using different metrics to analyze the impact of the word extraction task on the whole sentence recognition system's performances. The word extraction task using the best reconsideration strategy achieves a 97.94 % word extraction rate and a 84.85 % word recognition rate which represents a 33.1 % word error rate decrease relatively to the initial word extraction task (with no segmentation hypothesis reconsideration).

Keywords: Word extraction, confidence index, segmentation hypothesis generation, word graph, on-line sentence recognition.

1. Introduction

Handwriting recognition has been a subject of intensive research for many years. Whereas the recognition of isolated characters and words already achieves high recognition rates [29], handwritten text recognition is still a challenging task involving open issues. Among these problems are the integration of syntactic and semantic information during the recognition process or the segmentation problem since the number of words is unknown as well as their position in the text.

In the context of on-line handwriting recognition on pen-based devices such as personal digital assistants (PDAs) or Tablet PCs, input methods are needed. Handwriting input methods are designed to enter text on these pen-based devices, using a pen to write on the sensitive screen. The recognition of the strokes[a] corresponding to the text is thus

[a] A *stroke* is a list of chronologically sorted points, captured between a pen-down and a pen-up.

performed "on-the-fly" *i.e.* as they are written. Fig. 1 shows an example of an "on-the-fly" recognition on a PDA, using our input method DIGIME [5] which allows the input of isolated characters in a word context. The interface is designed to be as user friendly as

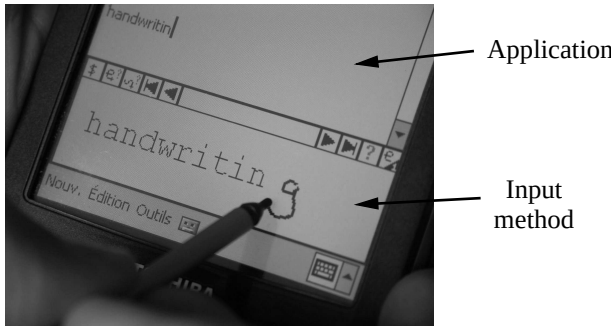


Fig. 1. Example of an "on-the-fly" recognition with the DIGIME input method.

possible. That is why the recognized characters are directly displayed in the input area, in addition to being transmitted to the target application. Our goal is actually to further extend our on-line handwritten sentence recognition system [30] to an "on-the-fly" recognition process to allow the input of text on TabletPCs. Fig. 2 shows an example of such a sentence input method based on the principles of the DIGIME method. The input area

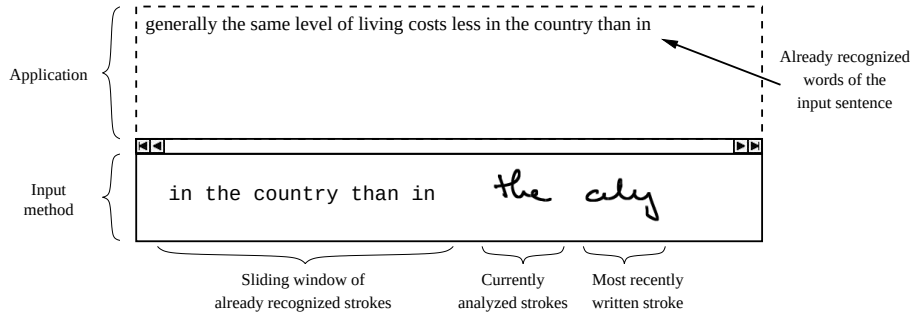


Fig. 2. Illustration of an "on-the-fly" sentence input method.

displays a sliding window of the most recently recognized words as well as strokes not yet recognized and whose recognition depends on the most recently written stroke. The will to further extend our sentence recognition system to an "on-the-fly" recognition process sets some constraints on the word extraction task in terms of the complexity of the methods used (which has an impact on the execution time) as well as toward the available information (not all the words of a text line are available but only the previously written ones).

In most of the works, lines are firstly detected before being segmented into words. In off-line recognition systems, the classic approaches are based on connected components [14,22] or on y-axis projection profiles [13,37]; recent works on the identification of regions and text lines on pages of on-line freeform handwritten notes also make use of the projection technique. [12,25,31] Other approaches to detect regions and text lines on on-line freeform handwritten notes are based on Probabilistic Feature Grammars [4], on classifiers [36] or on dynamic programming [33] but they need whole handwritten note pages and the detection algorithm is often a multi-pass one: this is not suited for an "on-the-fly" segmentation task. In off-line handwritten text recognition [23] as well as in on-line handwritten text recognition [27], rather simple methods using learnt thresholds on gaps between consecutive lines can also be applied.

Word extraction approaches are generally based on the following scheme. From a decomposition of a text line into components (in on-line recognition, the components generally correspond to the strokes whereas in off-line recognition they correspond to connected components), a classification of the gaps between these components is done in order to identify the words, by using a gap metric. Generally, the gap metrics used are the bounding box distance, the convex hull distance or the white run-length distance. Using the chosen gap metric, inter-component gaps are then classified into either inter-word gaps or intra-word gaps. This characterization can be done by comparing the gaps to a threshold [13,18,20,23,28,32,35]. Most of the thresholds used are not fixed but are computed from some features of the text line, like the median gap size or the line width: the whole text line is thus needed. Other methods use hierarchical clustering [15,21] or classifiers to discriminate intra-word and inter-word gaps. In Ref. 19, Hidden Markov Models are used to model space and non-space characters whereas a neural network classify the type of each inter-component gap in Refs. 9, 14, 27 (in Ref. 9, semantic information are also used in addition to the image features).

In this paper, we present a word extraction task based on the characterization of the gap between each couple of consecutive strokes, using a Radial Basis Function Network (RBFN). Unlike most of the works on word extraction, our proposed approach also detects the text lines simultaneously and doesn't perform any pre-processing steps typically used to normalize lines with respect to slant, skew or character height. Moreover, this approach is designed to be further used in an "on-the-fly" segmentation and recognition context, which limits the available knowledge to information about the strokes on the left of the most recently written stroke. In order to ensure the presence of the correct segmentation, we generate multiple segmentation hypotheses. Nevertheless, we want to limit the overall execution time which makes it important to choose the added segmentation hypotheses to only keep the most pertinent ones. To do this selection, we use a confidence index on the inter-stroke gap characterization results: this confidence index is then compared to a learnt threshold which enables us to control the total number of segmentation hypotheses by focusing on the less reliable segmentation points. All the segmentation hypotheses are finally organized into a word graph on which the sentence recognition is performed, using our word recognition system RESIFMot [8] as well as a language model. It should be noted that the word extraction task is independent of the sentence recognition step so any other

word recognition system may be used. Our word extraction task is evaluated according to different reconsideration thresholds as well as towards a straightforward "oversegment & merge" approach (each stroke is considered to be a word and up to $nmax_{strokes}$ strokes can be merged to form a word) which shows the efficiency of our approach.

In Sec. 2, the whole sentence recognition system is introduced. Then, our initial word extraction task is first described in Sec. 3 and its extension for the generation of additional segmentation hypotheses based on a confidence index is presented in Sec. 4. Finally, experimental results are discussed in Sec. 5 while Sec. 6 draws some conclusions.

2. Overview of the Sentence Recognition System

The whole recognition system illustrated by Fig. 3 extends our on-line sentence recognition system previously presented in Ref. 30. This system consists of three parts: segmentation of the sentence into its words, recognition of the extracted words and recognition of the sentence. We introduce these three steps and we will present the word extraction part in further details in Sections 3 and 4, since this is the object of this article.

2.1. *Word extraction*

As an on-line handwritten text is represented by a list of chronologically sorted strokes, the word extraction task consists of extracting each sublist of strokes corresponding to the words. Thus, the first sub-module of the word extraction task aims at gathering the parts of the input handwritten sentence that correspond to each handwritten word, to initialize the word graph (see Sec. 3). The nodes of this graph represent the segmentation frontiers between two consecutive words and the edges stand for the hypothetical handwritten words (initial edges are shown in bold in the word graph of Fig. 3). The task of the following sub-module is to generate additional segmentation hypotheses (represented by dotted edges in Fig. 3). These edges and nodes are created in the word graph in order to ensure the correct segmentation to appear in the word graph, thus dealing with potential under- and over-segmentation problems that may appear in the initial word graph (see Sec. 4).

2.2. *Word recognition*

Our word recognition system RESIFMot [8] takes each previously extracted word as an input and outputs a list of 20 candidate words: this list is then associated to each edge of the word graph, as shown in figure Fig. 3.

RESIFMot is a word recognition system based on an analytic approach. Words are segmented according to different hypotheses of letter allographs[b], based on specific knowledge of the structure of handwritten characters. The fundamental structure corresponds to *pertinent downstrokes*: one character is made of one to three pertinent downstrokes (independently of the writing style). The allograph hypotheses are organized into a segmentation graph and an adapted version of our character recognition system RESIFCar [2] is used to

[b] *Allographs* are variant shapes of the same letter.

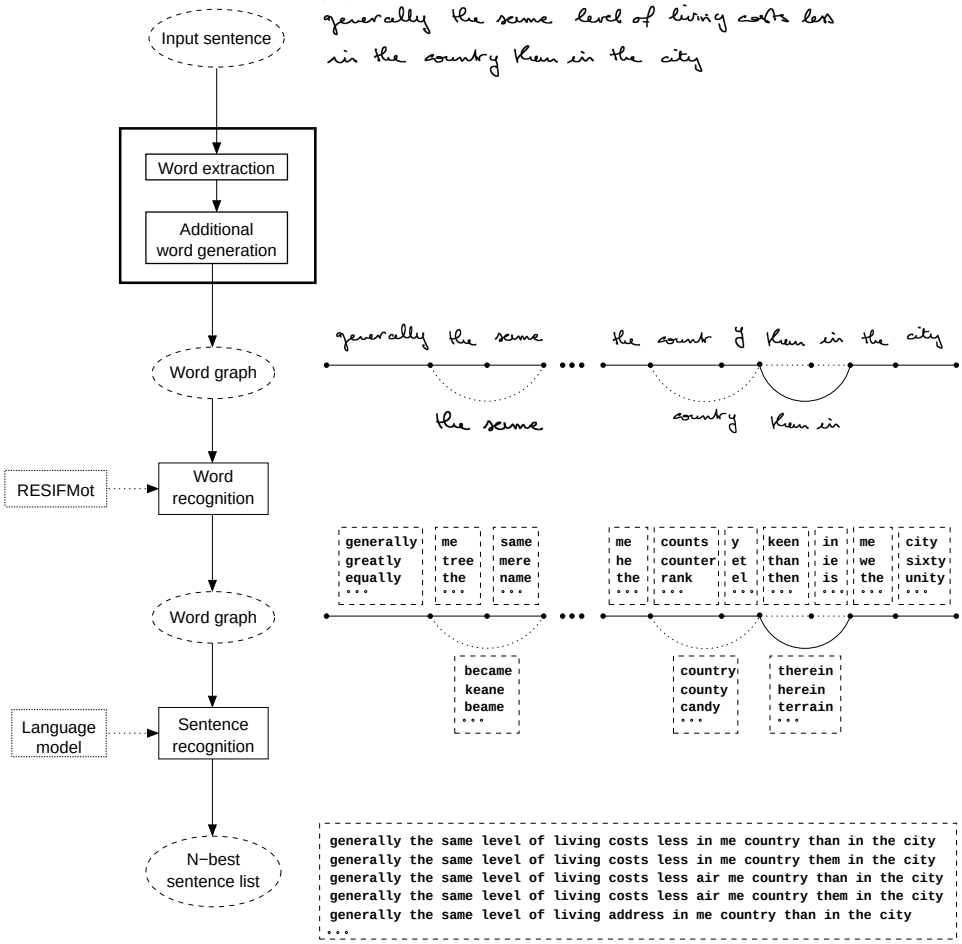


Fig. 3. Sentence recognition system.

validate the correct segmentation hypotheses and to generate each allograph hypothesis. Then, the exploration of this graph produces a list of character strings. Finally, a lexical post-processing step is performed to retrieve the nearest word of each string hypothesis, according to a dictionary. The word recognition system thus outputs a list of candidate words, ranked according to a score depending on edit operations used to transform the character string into its corresponding word [7].

2.3. *Sentence recognition*

The sentence recognition is finally performed to retrieve the N-best sentences $\left\{\hat{W}\right\}$ that maximize the *a posteriori* probability among all the sentences $W_k = w_{k,1} \dots w_{k,n_k}$, given

the handwritten signal S:

$$\left\{\hat{W}\right\} = \left\{\arg\max_{W_k} \ score(W_k)\right\} = \left\{\arg\max_{W_k} \ score(S|W_k) + \gamma \ \log P(W_k) + \delta \, n_k\right\} \quad (1)$$

where $score(S|W_k)$ is estimated by the word recognition system RESIFMot and $P(W_k)$ is estimated thanks to a language model (since our word recognition system is not probabilistic, the output is not a probability but this output score can be interpreted as a likelihood and then combined to a language model probability). To optimize the integration of the language model into the sentence recognition system, two additional parameters γ and δ are introduced. The parameter γ (called the *Grammar Scale Factor*) is used to balance the relative impact of the language model toward the word recognition system whereas δ (called the *Word Insertion Penalty*) is used to deal with the whole sentence recognition system tendency to make over- or under-segmentations. When the recognition system has a tendency to under-segmentation, setting $\delta < 0$ can compensate it whereas setting $\delta > 0$ can deal with a tendency to over-segmentation.

A beam search algorithm [26] is actually performed on the word graph to efficiently retrieve the N-best sentences $\left\{\hat{W}\right\}$ (corresponding to paths in the graph), by combining the graphic and linguistic information as given by Eq. (1). This search strategy is time-synchronous and can be viewed as an approximation of the Viterbi algorithm [10] where, at each node of the word graph, only the sentence hypotheses whose scores within a fixed radius (called the *beam*) are kept: this enables the pruning of the most unlikely hypotheses. The size of the beam is set empirically (at most 5 000 hypotheses are kept in our system). This recognition scheme can be performed during the writing of the sentence, which is suitable for an "on-the-fly" recognition. Therefore, the recognized sentence can be displayed as the sentence is being written and will be updated each time the most recently written strokes have been analyzed and recognized (as shown by Fig. 2).

After presenting the whole sentence recognition system, we will focus on the word extraction task, in the two following sections.

3. Initial Word Extraction

Our word extraction scheme consists of a characterization of the inter-stroke gaps of the on-line signal representing the considered handwritten sentence. Since our data are English handwritten sentences without any punctuation mark, we consider three kinds of gaps:

- *intra-word gap*: gap between two strokes from the same word
- *inter-word gap*: gap between two strokes from two consecutive words
- *inter-line gap*: gap between two strokes from two consecutive words written on two consecutive lines.

Fig. 4 gives an example of a handwritten sentence to recognize where intra-word, inter-word and inter-line gaps are shown.

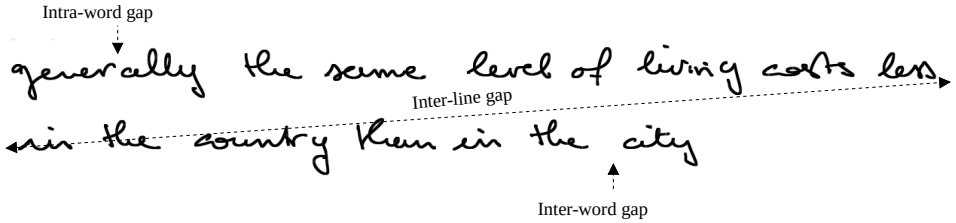


Fig. 4. Handwritten sentence with examples of the three kinds of inter-stroke gaps.

3.1. *Inter-stroke gap characterization*

The inter-stroke gap characterization algorithm process the strokes as they are written (see Algorithm 1). For each newly written stroke S_{new} the gap between this stroke and the

Algorithm 1: Algorithm for the characterization of inter-stroke gaps.

Input: strokes of the handwritten sentence to recognize;

Output: inter-stroke gaps characterized as intra-word, inter-word or inter-line gaps;

begin
- **Initialization**
- initialization of BRG with the firstly written strokes (temporally sorted);
- **while** *a stroke* S_{new} *is written* **do**
  - detection of the writing baselines;
  - computation of the distance Δx_{ref}^{new} between S_{ref} and S_{new};
  - classification of the gap E_{ref}^{new} between the strokes S_{ref} and S_{new};
  - update of BRG;

end

previously written one S_{ref} (also called the *reference stroke*) is characterized into one of the three considered kinds of gaps. This gap characterization is based on the computation of the distance Δx_{ref}^{new} between these two strokes. In fact, for this distance computation, not only the previously written stroke S_{ref} is used but also a group of more previously written strokes called the *Baseline Reference Group* (BRG): this group of strokes represents a relative spatial context toward which the distance Δx_{ref}^{new} can be compared. Fig. 5 illustrates the distance Δx_{ref}^{new} computation as well as the corresponding BRG used. A classifier is finally used to characterize the inter-stroke gap, based on the computed distance Δx_{ref}^{new} as well as other information.

In the following subsections, we detail the steps of our gap characterization algorithm.

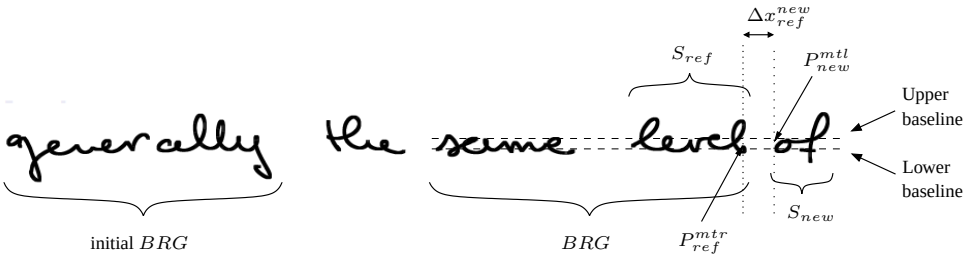


Fig. 5. Example of a distance Δx_{ref}^{new} computation.

3.1.1. *Initialization*

At the beginning of the word extraction task, the Baseline Reference Group BRG is initialized with the firstly written strokes of the sentence (see Sec. 3.1.5 for greater details on the number of strokes used to initialize the BRG).

3.1.2. *Lower and upper baseline detection*

The lower and upper baselines we detect delimit a more restricted zone than the text body classically used. This zone is defined as the vertical zone covering the maximal number of pertinent downstrokes. This maximal covering is computed as in our word recognition system [1]: an histogram distribution of the projection of the text on the y-axis is used but where only the pertinent downstrokes are being projected (instead of considering all the points of all the characters). This zone actually corresponds to the one where letters like *a* or *e* are written: this zone is the most stable zone of the writing and is less sensitive to skew problems.

To detect the lower and upper baselines, we run the algorithm also used in our word recognition system. The baseline detection is thus carried out on a group of strokes representing a "virtual word": it corresponds to the BRG previously presented. To ensure the robustness of the baseline detection, this "virtual word" has to contain three or four characters. Thus, since a character is composed of one to three pertinent downstrokes (see Sec. 2.2), we keep enough strokes in the BRG to ensure it contains at least 10 pertinent downstrokes. We could have kept more previously written strokes in the BRG to make the baseline detection the most reliable possible but this detection would have been too sensitive to skew problems. Indeed, since the gap characterization algorithm is designed to be further extended to an "on-the-fly" recognition, we want to avoid costly pre-processing steps that are generally performed on the signal to correct slant, skew or character size problems. That's why we limit the number of strokes in the BRG.

3.1.3. *Distance Δx_{ref}^{new} computation*

Since our data are English handwritten sentences, the first stroke of a new word is more likely to be on the right of the last stroke of the previous word. To compute the distance Δx_{ref}^{new} (see Fig. 5) between the most recently written stroke S_{ref} and the newly written

stroke S_{new}, we use the distance in x-coordinate between the most-on-the-right point of S_{ref} and the most-on-the-left point of S_{new}:

$$\Delta x_{ref}^{new} = x_{P_{new}^{mtl}} - x_{P_{ref}^{mtr}} \tag{2}$$

where P_{new}^{mtl} is the most-on-the-left point of S_{new} and P_{ref}^{mtr} is the most-on-the-right point of S_{ref}. These points are chosen among the points between the lower and the upper baselines in order to deal with slant problems. Indeed, this kind of problem is not corrected by pre-processing steps performed on the sentence before the extraction of its words. In the example given by Fig. 6, we can see that if these points weren't chosen between the baselines, the computed distance would have been the bounding box distance Δx_{BB} and an intra-word gap would have been detected between the words *said* and *again* whereas our restriction allows the correct detection of the inter-word gap between these words.

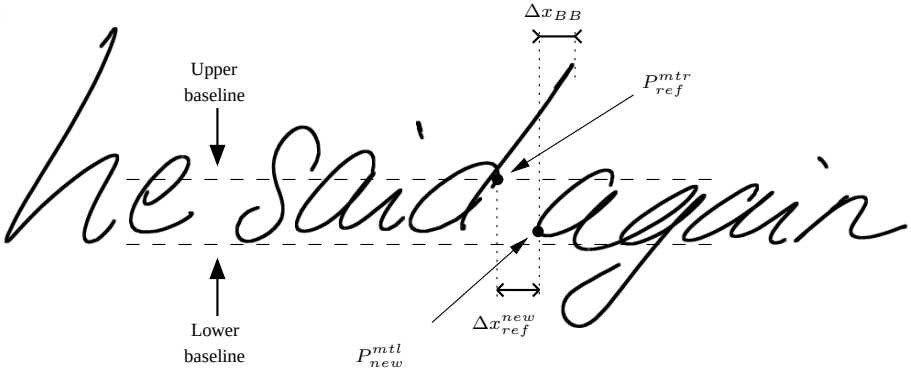


Fig. 6. Illustration of the choice of the points P_{new}^{mtl} and P_{ref}^{mtr} to deal with slant problems.

3.1.4. *Gap E_{ref}^{new} classification*

Once the distance Δx_{ref}^{new} has been computed, we use a Radial Basis Function Network (RBFN) to classify the inter-stroke gap. The radial basis functions are learnt for each class separately using the fuzzy C-means algorithm [16] and the weights in the output layer are trained with the least-mean square algorithm [3].

The inputs of this classifier are the following:

- the size of the current inter-stroke gap, relatively to the previous strokes: it corresponds to the distance Δx_{ref}^{new};
- information on the relative sizes of the previous inter-stroke gaps: this is given by the maximum and the median Δx_{ref}^{new} in the BRG;
- information for inter-line gaps identification: this is given by the distance between the top of the S_{new} bounding box and the lower baseline.

The outputs of the RBFN are the scores associated with each of the three classes corresponding to the intra-word gap, the inter-word gap and the inter-line gap. Finally, the type of the considered inter-stroke gap corresponds to the class with the highest score.

3.1.5. BRG *update*

Once the gap E_{ref}^{new} is classified, the stroke S_{new} is added to the BRG. The temporally oldest stroke of the BRG can then be deleted but only if at least 10 pertinent downstrokes still remain in the BRG. The BRG can be seen as a sliding window, moving with the writing and the processing of the strokes: this is particularly adapted for an "on-the-fly" word extraction.

To start the inter-stroke gap characterization, the baseline detection needs to have enough written strokes to ensure the first BRG to contain at least 10 pertinent downstrokes. This initial BRG is also used to characterize the first inter-stroke gaps even if the considered strokes belong to this BRG. As can be seen in Fig. 5, the BRG used to characterize the gap between the strokes *gener* and *ally* contains these two strokes.

3.2. *Word graph creation*

The initial word graph is built from the results of the inter-stroke gap classification task. For each inter-line or inter-word gap detected, a node is created: each node thus represents a segmentation frontier between two words. Edges are created by gathering strokes such that each inter-stroke gap between two consecutive strokes of an edge is an intra-word gap. They thus represent the extracted handwritten words.

Fig. 7 shows an example of a word graph built from the characterization of the inter-stroke gaps of the sentence *"generally the same level of living costs less in the country than in the city"*. We can see that the words *living* and *country* are over-segmented whereas the words *"than in"* are under-segmented.

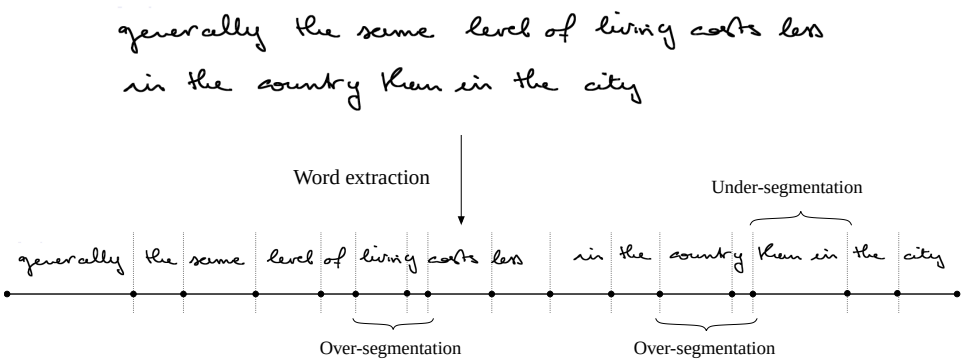


Fig. 7. Example of a word graph, built from the initial extraction of the words of a handwritten sentence.

We now extend our inter-stroke gap characterization algorithm to deal with potential

under- and over-segmentations, by creating additional segmentation hypotheses while controlling the size of the word graph.

4. Confidence Index Based Word Extraction

To ensure the presence of the correct segmentation of the sentence, one straightforward solution would be to consider all the possible segmentation hypotheses (it corresponds to an "oversegment & merge" approach). This is not reasonable since it would dramatically increase the size of the word graph and thus considerably increase the recognition time. Furthermore, it would mostly introduce some noise *i.e.* incorrect segmentation hypotheses. This could then bring errors during the sentence recognition step performed on the word graph. The experiments presented in Sec. 5 will confirm this idea.

The key aspect of the generation of additional segmentation hypotheses lies in the choice of these additional hypotheses which should be the most pertinent. To create these hypotheses, we rely on an index evaluating the confidence in the result of the inter-stroke gap classification.

In the following subsections, we present the computation of this confidence index as well as its use to deal with potential under- and over-segmentations.

4.1. *Confidence index on the inter-stroke gap characterization*

In order to evaluate the reliability in the first answer of the inter-stroke gap classification, we associate a *confidence index* to the result of the RBFN. If this confidence is too low, the second answer of the classifier is also considered.

The confidence index we use corresponds to the relative difference $diff_{top2}$ between the scores of the two best classes. The confidence index $diff_{top2}$ is then compared to a learnt threshold $\sigma_{reconsider}$: if the confidence index is below the threshold, the first answer of the RBFN can be reconsidered and additional segmentation hypotheses may then be created.

To set the threshold $\sigma_{reconsider}$ on these confidence indexes, we use an *ambiguity reject* [24]. The use of this kind of reject allows us to learn the threshold by setting the percentage of elements that have to be rejected *i.e.* the percentage of inter-stroke gap classifications that have to be reconsidered, in our case.

4.2. *Generation of additional segmentation hypotheses*

Confidence indexes associated to each of the inter-stroke gaps now allow the creation of additional segmentation hypotheses (represented by edges in the word graph).

To control the size of the final word graph (in terms of number of edges), we set some limitations. Firstly, inter-line gaps are supposed to be correctly identified and will thus not be reconsidered. The other restrictions aim at dealing with either under-segmentations or over-segmentations and will be further explained in the following subsections. In Sec. 5, we will see that these limitations are not too restrictive since almost all the segmentations can be retrieved, in the test set.

4.2.1. *Under-segmentation processing*

To limit the creation of additional edges, we consider that at most $nmax_{under}$ words may be grouped into a same word (thus represented by one edge), after the initial word extraction. An initial edge could then be further separated into $nmax_{under}$ parts, at most.

In order to chose the points of the potential separations of an edge, its intra-word gaps with a positive distance Δx_{ref}^{new} and an associated confidence index $diff_{top2}$ below the reconsideration threshold $\sigma_{reconsider}$ are sorted according to their ascending confidence indexes. Then, at most the $nmax_{under} - 1$ first intra-word gaps are considered as potentially misrecognized inter-word gaps. Additional edges and nodes are then created from the current edge, as shown in Fig. 8 by dotted edges and their corresponding nodes (here, $nmax_{under} = 3$).

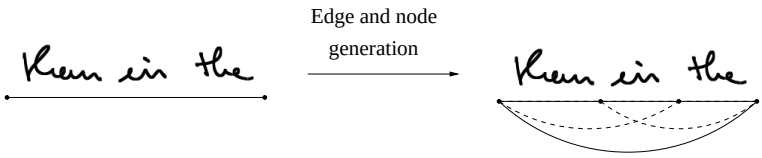


Fig. 8. Example of additional edge and node creation, to deal with under-segmentations.

4.2.2. *Over-segmentation processing*

In the same way as the processing of under-segmentations, we consider that a word may be over-segmented into at most $nmax_{over}$ parts. At most $nmax_{over}$ initial edges could then be further gathered into additional edges.

To deal with these potential over-segmentations, we consider each group of $nmax_{over}$ consecutive edges. At most $nmax_{over} - 1$ inter-word gaps with a confidence index $diff_{top2}$ below the reconsideration threshold $\sigma_{reconsider}$ can be considered as potentially misrecognized intra-word gaps: additional edges are then created as shown in Fig. 9 (here, $nmax_{over} = 3$).

Moreover, to limit the length of edges in terms of the number of dowstrokes, we consider that additional edges can only be created if their total number of downstrokes is at most $nmax_{downstr}$.

4.2.3. *Creation of the final word graph*

The final word graph is created after the processing of every potential under- and over-segmentations. Fig. 10 shows the full word graph resulting in the whole word extraction task based on the use of confidence indexes, for the sentence already presented in Fig. 7. Edges and nodes in dotted lines were created to deal with the initial under- and over-segmentations. We can see that a path corresponding to the correct segmentation now appears in the word graph.

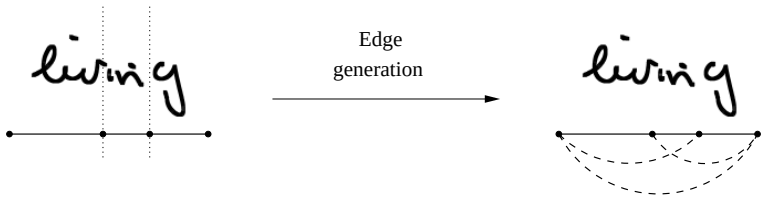


Fig. 9. Example of additional edge creation, to deal with over-segmentations.

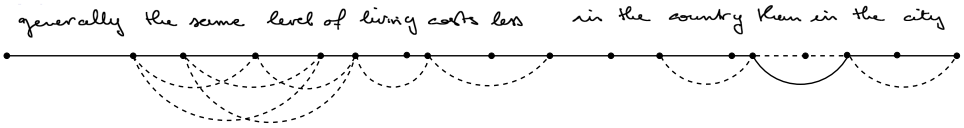


Fig. 10. Example of a full word graph (additional edges and nodes are illustrated by dotted lines).

5. Experiments and Results

In this section, we present the experiments conducted to evaluate our word extraction approach. We first describe the databases we use and then discuss the parameter optimization in Sec. 5.2 before presenting the results in Sec. 5.3.

5.1. *Databases*

The handwritten material consists of English sentences without any punctuation marks, written from 2,598 sentences of the Brown corpus [11]. In these first experiments on word extraction, we wanted to evaluate our approach without adding the difficulty of processing punctuation marks, as this is the case in the work presented in Ref. 18 where punctuation marks were removed from the data using some heuristics (the processing of punctuation marks will be investigated in future works, which will be further discussed in Sec. 6). The training set includes 517 sentences (8,047 words) written by 25 writers whereas the test set includes 425 sentences (6,362 words) written by 17 writers. The training set is used to learn the RFBN for the inter-stroke gap classification task as well as the reconsideration thresholds $\sigma_{reconsider}$. The writers of the test set are different from those of the training set.

The language model used during the sentence recognition step (see section 2) is a bigram model. A bigram model gives the probability of a word w_i by only considering its previous word w_{i-1}: the probability $P(w_{k,i}|w_{k,i-n+1}^{i-1})$ is thus approximated by $P(w_{k,i}|w_{k,i-1})$. The bigram language model used in our experiments is built from the Brown corpus, with the SRILM toolkit [34]. The Brown corpus contains 52,954 English sentences (1,002,675 words) where 46,836 sentences (900,108 words) were actually kept to learn the language model (the 2,598 remaining ones are used for the acquirement of the handwritten material). During the lexical post-processing [8], we use an associated lexicon including 13,748 words.

5.2. *Parameter optimization*

Since we don't have a validation set, the training set is also used to optimize the integration of the language model (*i.e.* the values of the grammar scale factor, and of the word insertion penalty) as well as to tune the parameters for our complete word extraction approach based on confidence indexes. In the following sub-sections, we present the results of the optimization of the parameters used in the confidence index based word extraction approach, to generate additional segmentation hypotheses.

5.2.1. *Number of segmentation parts: setting $nmax_{under}$ and $nmax_{over}$*

The generation of additional segmentation hypotheses aims at dealing with over- and under-segmentation errors that could appear when using the initial word extraction approach. To do so, the number of parts in which a word may be over-segmented has to be set as well as the number of words considered as one word: it corresponds to the parameters $nmax_{under}$ and $nmax_{over}$.

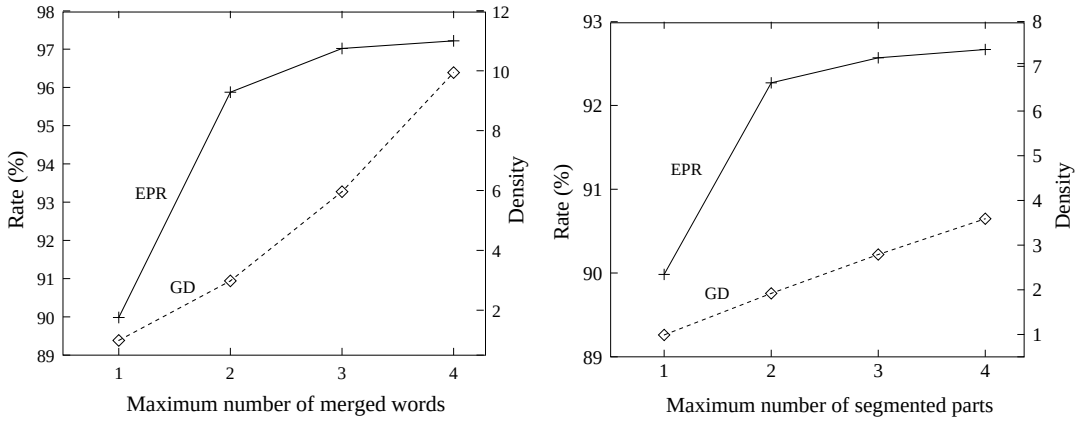
Fig. 11(a) and Fig. 11(b) show the impact of $nmax_{under}$ and $nmax_{over}$ respectively, according to two criteria:

- *Edge presence rate*:

$$EPR = \frac{\text{number of edges corresponding to correctly extracted words}}{\text{number of words}}$$

- *Graph density*:

$$GD = \frac{\text{number of edges}}{\text{number of words}}$$



(a) Optimization of $nmax_{under}$ (b) Optimization of $nmax_{over}$

Fig. 11. Optimization of the segmentation parameters $nmax_{under}$ and $nmax_{over}$, on the training set.

The aim is thus to maximize the presence of the whole correct segmentation hypothesis (*i.e.* EPR) while minimizing the total number of segmentation hypotheses (*i.e.* GD). The values of the EPR and the GD are maximal bounds since we consider here that each inter-stroke gap characterization can be reconsidered to generate additional segmentation hypotheses. In fact, these values will be lower because an optimized reconsideration threshold $\sigma_{reconsider}$ is used to further limit the number of generated hypotheses (it will be discussed in the following sub-section). Therefore, we consider that the best trade-off between the considered indicators is achieved for $nmax_{under} = 3$ and $nmax_{over} = 3$. With these values, the generation of additional segmentation hypotheses could allow the correct extraction of words over-segmented into at most three parts as well as of words under-segmented with at most three words considered as one word.

Moreover, the maximal number of downstrokes contained in a word has to be set: its computation on the training set sets the value of $nmax_{downstr}$ to 25. Using the optimized values of these three segmentation parameters, the maximal EPR we can achieve on the test set is 99.76 %.

5.2.2. *Reconsideration threshold: optimizing $\sigma_{reconsider}$*

To generate additional segmentation hypotheses, a reconsideration threshold $\sigma_{reconsider}$ also needs to be chosen: the value of this threshold is automatically learnt from the percentage of inter-stroke gaps whose characterization has to be reconsidered (see Sec. 4.1).

Fig. 12 shows the impact of different reconsideration thresholds on the creation of additional segmentation hypotheses, according to the edge presence rate EPR and the graph density GD defined in Sec. 5.2.1. The different segmentation strategies are identified by the

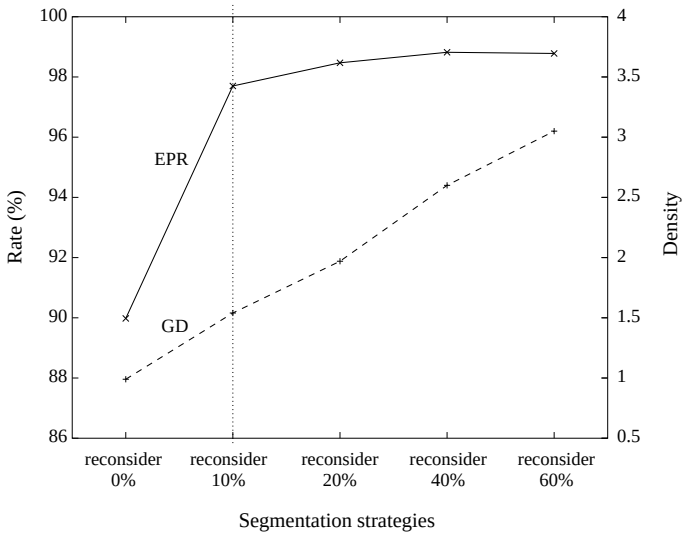


Fig. 12. Optimization of the reconsideration threshold $\sigma_{reconsider}$, on the training set.

percentage of inter-stroke gap classifications that are reconsidered when the corresponding threshold is learnt. As previously, the aim is thus to minimize the GD while maximizing the EPR. The reconsideration threshold that leads to the best trade-off between these indicators is the one reconsidering 10 % of the inter-stroke gap classifications (the corresponding value is $\sigma_{reconsider} = 0.71$): the graph density increases a little (from 0.99 to 1.54) whereas the edge presence rate rises greatly (by 8 %), as compared to the initial word extraction approach. Whereas the improvement from no reconsideration threshold to the 10 % reconsideration threshold is statistically significant, the small increase in the edge presence rate when going from the 10 % reconsideration threshold to the 20 % reconsideration threshold is not statistically significant (measured at the word level using a paired t-test at the 0.01 significance level).

5.3. *Experimental results*

The aim of these experiments is to show the efficiency of our word extraction approach based on the characterization of inter-stroke gaps as well as of our extended approach where additional segmentation hypotheses are generated using the optimized reconsideration threshold (see Sec. 5.2.2). Our approaches are also compared to a manual word extraction task (representing the ground truth) as well as to a straightforward "oversegment & merge" approach. In the latest approach, each stroke is considered to be a word and words can also be created from up to $nmax_{strokes}$ consecutive strokes.

Four metrics are used to compare these approaches. We use the edge presence rate EPR and the graph density GD defined in Sec. 5.2.1 as well as the word extraction rate and the word recognition rate defined as follows:

- *Word extraction rate*:

$$WER = \frac{\text{number of correctly extracted words}}{\text{number of words}}$$

- *Word recognition rate*:

$$WRR = \frac{\text{number of correctly recognized words} - \text{inserted words}}{\text{number of words}}$$

The word extraction rate WER and the word recognition rate WRR are computed on the recognized sentences whereas the edge presence rate EPR and the graph density GD are computed on the word graphs. The graph density allows the comparison between the number of words that have to be extracted (*i.e.* the number of words to recognize) and the number of words extracted by the considered approach. Since the word recognition system is called on each edge and the recognition of words represents the main part of the overall processing time of a sentence, the graph density gives an indication on the increase of the overall recognition time towards a correct segmentation approach (*i.e.* that only generates segmentation hypotheses corresponding to words to recognize). Indeed giving real execution time wouldn't be meaningful since our word recognition system is not yet optimized to really perform "on-the-fly".

Table 1. Comparison of different word extraction strategies (on the test set).

Word extraction strategy	WRR	WER	EPR	GD
Manual (ground truth)	88.46 %	100 %	100 %	1.00
Initial word extraction (0 % reconsider.)	77.35 %	89.86 %	89.86 %	0.99
Confidence index based word extraction (10 % reconsider.)	84.85 %	96.01 %	97.94 %	1.58
"Oversegment & merge" ($nmax_{strokes} = 6$)	57.48 %	66.44 %	89.09 %	20.32
"Oversegment & merge" ($nmax_{strokes} = 10$)	73.42 %	78.58 %	98.29 %	32.53

The results on the comparison of the different word extraction strategies on the test set are summarized in Table 1. As already seen with the EPR on the training set, the performances are greatly improved when additional segmentation hypotheses are generated, based on confidence indexes. Indeed, with the optimal confidence index based word extraction approach (with the optimal reconsideration threshold presented in Sec. 5.2.2), the WRR is increased by 7.5 % (corresponding to a 33.1 % relative reduction of the word error rate), the WER by 6.2 % (corresponding to a 66.7 % relative reduction of the word extraction error rate) and the EPR by 8.1 %, as compared to the initial word extraction approach. These rates are also quite close to the ground truth (less than 4 % below). Moreover, the GD only rises a little and is about 1.5 times the GD of the initial extraction approach. The GD is also about 1.5 times the GD of the ground truth: it means that the overall execution time with the confidence index based word extraction approach doesn't increase too much as compared to the execution time when the segmentation is entirely correct, represented by the ground truth.

Two straightforward "oversegment & merge" approaches are considered, depending on the value of the parameter $nmax_{strokes}$. These values were chosen so as to have edge presence rates close to the one achieved with our initial word extraction approach (corresponding to $nmax_{strokes} = 6$) and close to the one for our confidence index based word extraction approach (corresponding to $nmax_{strokes} = 10$). The graph densities obtained with the "oversegment & merge" methods are much higher than the ones achieved with our approaches: when $nmax_{strokes} = 6$, the GD is increased by more than a 20 factor (relatively to the complexity for our initial word extraction method) and when $nmax_{strokes} = 10$, this factor is also a little more than 20 (relatively to the complexity achieved with our confidence index based word extraction approach). It shows the efficiency of our word extraction approaches since we achieve comparable EPR but with lower GD which implies lower execution times. Moreover, the WER and WRR are below the ones achieved with our approaches (more than 20 % and 15 % lower, respectively). Having lower WER and WRR whereas having close EPR can be explained by the fact that the size of the search space has been considerably increased with the "oversegment & merge" methods (as illustrated by their GD): it makes the recognition of the correct sentences harder (the WER and WRR are computed on the recognized sentences). Indeed, there is a very high number of incorrect segmentation hypotheses that are generated to ensure the correct one to appear in the word graph and it makes it more difficult to identify the correct sentence between all the possible sentences of the graph. This shows the importance of the selection of the generated segmentation hypotheses not only with respect to the execution time but also

towards the overall performance of the recognition system.

These results cannot be directly compared to existing works since the databases are different from each other. However, in Ref. 18, the WER achieved on a subset of the IAM-OnDB database [17] with a threshold based approach is 86.6 %. It can be compared to the WER achieved with our initial word extraction approach since no sentence recognition step is involved in the cited work whereas the WER for our confidence index based approach is computed on recognized sentences. Nonetheless, even if the handwritten texts of the subset of the IAM-OnDB database do not contain punctuation marks (that were manually removed), these data were written on a whiteboard which makes the segmentation process more difficult than using texts written on a TabletPC like our data. Future works will involve the evaluation of our word extraction approach on this IAM-OnDB database.

6. Conclusion and Future Work

In this paper, we have presented an automatic word extraction mechanism based on the classification of inter-stroke gaps, using a RBFN. In order to ensure the presence of the correct segmentation, a step was added to allow the generation of other segmentation hypotheses into the word graphs on which the sentence recognition is performed. Their creation relies on a confidence index associated with the inter-stroke gaps classification results. This confidence index is used to detect the less reliable segmentation points by comparing it to a learnt threshold. Thanks to it, we were able to limit the total number of segmentation hypotheses and thus the overall execution time since we want to further extend our sentence recognition system to perform "on-the-fly" recognition.

Experimental results have shown significant improvements on the performances of the recognition system when additional segmentation hypotheses were created, based on the use of confidence indexes. Indeed, when 10 % of the gap classification results may be reconsidered, the word recognition rate went from 77.35 % to 84.85 % whereas the word extraction rate went from 89.86 % to 96.01 %. Our word extraction approaches were also compared to a straightforward "oversegment & merge" method. The graph densities achieved with our approaches were thus shown to be more than 20 times lower than those of the considered "oversegment & merge" methods. It is important to take into account the size of the word graph generated by a word extraction approach since the overall recognition time is directly linked to it and one of our objectives was to limit this recognition time to further perform "on-the-fly" recognition. Furthermore, we have also enlightened the importance of the choice of the generated segmentation hypotheses. Indeed, the word recognition rates dropped significantly when a too large number of incorrect segmentation hypotheses were added to the word graphs.

In order to take advantage of segmentation information to further optimize the sentence recognition, future works will investigate the integration of the segmentation score associated with each inter-stroke gap (see Sec. 3.1.4) into the sentence score which already uses graphic and linguistic information (see Eq. (1)). Moreover, we will carry on our first experiments by evaluating our word extraction task on other on-line handwritten sentence databases, like the IAM-OnDB database [17]. In order to extend our word extraction task

to real use cases, we will need to process punctuation marks. In Ref. 6, we have already presented an approach which allows the recognition of punctuation marks in a character recognition context: the same approach can be used in our sentence recognition context. Finally, since our word extraction task was designed to be further used for "on-the-fly" recognition, it would be interesting to investigate its performance in a real "on-the-fly" recognition context. This will also require an optimization of our word recognition system.

References

1. E. Anquetil and G. Lorette, "Perceptual model of handwriting drawing, application to the handwriting segmentation problem", *Proc. 4th Int. Conf. Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 112–117.
2. E. Anquetil and H. Bouchereau, "Integration of an on-line hHandwriting recognition system in a smart phone device", *Proc. 16th Int. Conf. Pattern Recognition*, Québec, Canada, 2002, pp. 192–195.
3. C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
4. J. Blanchard and T. Artières, "On-line handwritten documents segmentation", *Proc. 9th Int. Workshop Frontiers in Handwriting Recognition*, Tokyo, Japan, 2004, pp. 148–153.
5. F. Bouteruche, G. Deconde, E. Anquetil, and E. Jamet, "Design and evaluation of handwriting input interfaces for small-size mobile devices", *Proc. 1st Workshop Improving and Assessing Pen-Based Input Techniques*, Edinburgh, Scotland, 2005, pp. 49–56.
6. F. Bouteruche and E. Anquetil, "Fuzzy point of view combination for contextual shape recognition: application to on-line graphic gesture recognition", *Proc. 18th Int. Conf. Pattern Recognition*, Hong-Kong, China, 2006, pp. 1088–1091.
7. S. Carbonnel E. Anquetil, "Lexicon organization and string edit distance learning for lexical post-processing in handwriting recognition", *Proc. 9th Int. Workshop Frontiers in Handwriting Recognition*, Tokyo, Japan, 2004, pp. 462–467.
8. S. Carbonnel E. Anquetil, "Lexical post-processing optimization for handwritten word recognition", *Proc. 7th Int. Conf. Document Analysis and Recognition*, Edinburgh, Scotland, 2003, pp. 477–481.
9. M. Feldbach and K.D. Tönnies, "Word segmentation of handwritten dates in historical documents by combining semantic a-priori-knowledge with local features", *Proc. 7th Int. Conf. Document Analysis and Recognition*, Edinburgh, Scotland, 2003, pp. 333–337.
10. G.D. Forney, The Viterbi Algorithm, *Proc. IEEE*, **61** (3) (1973) 268–278.
11. W.N. Francis and H. Kucera, *Brown Corpus Manual*, Brown University, 1979.
12. A.K. Jain and A.M. Namboodiri and J. Subrahmonia, "Structure in on-line documents", *Proc. 6th Int. Conf. Document Analysis and Recognition*, Seattle, WA, 2001, pp. 844–848.
13. E. Kavallieratou and N. Dromazou and N. Fakotakis and G.K. Kokkinakis, "An integrated system for handwritten document image processing", *Int. Journal Pattern Recognition and Artificial Intelligence*, **17** (4) (2003) 617–636.
14. G. Kim and V. Govindaraju and S.N. Srihari, "An architecture for handwritten text recognition systems", *Int. Journal Document Analysis and Recognition*, **2** (1) (1999) 37–44.
15. S.H. Kim and C.B. Jeong and H.K. Kwag and C.Y. Suen, "Word segmentation of printed text lines based on gap clustering and special symbol detection", *Proc. 16th Int. Conf. Pattern Recognition*, Québec City, Canada, 2002, pp. 320–323.
16. R. Krishnapuram and J.M. Keller, "The possibilistic C-means algorithm: insights and recommendations", *IEEE Trans. Fuzzy Systems*, **4** (3) (1996) 385–393.
17. M. Liwicki and H. Bunke, "Handwriting recognition of whiteboard notes – studying the influence of training set size and type", *Int. Journal Pattern Recognition and Artificial Intelligence*, **21** (1) (2007) 83–98.

18. M. Liwicki and M. Scherz and H. Bunke, "Word extraction from on-line handwritten text lines", *Proc. 18th Int. Conf. Pattern Recognition*, Hong-Kong, China, 2006, pp. 929–933.
19. F. Lüthy and T. Varga and H. Bunke, "Using hidden markov models as a tool for handwritten text line segmentation", *Proc. 9th Int. Conf. Document Analysis and Recognition*, Curitiba, Brazil, 2007, pp. 8–12.
20. U. Mahadevan and R.C. Nagabushnam, "Gap metrics for word separation in handwritten lines", *Proc. 3rd Int. Conf. Document Analysis and Recognition*, Montréal, Canada, 1995, pp. 124–127.
21. U. Mahadevan and S.N. Srihari, "Hypotheses generation for word separation in handwritten lines", *Proc. 5th Int. Workshop Frontiers in Handwriting Recognition*, Essex, England, 1996, pp. 453–456.
22. R. Manmatha and J. Rothfeder, "A scale space approach for automatically segmenting words from historical handwritten documents", *IEEE Trans. Pattern Analysis and Machine Intelligence*, **27** (8) (2005) 1212–1225.
23. U.-V. Marti and H. Bunke, "Text line segmentation and word recognition in a system for general writer independent handwriting recognition", *Proc. 6th Int. Conf. Document Analysis and Recognition*, Seattle, WA, 2001, pp. 159–163.
24. H. Mouchère and E. Anquetil, "A unified strategy to deal with different natures of reject", *Proc. 18th Int. Conf. Pattern Recognition*, Hong-Kong, China, 2006, pp. 792–795.
25. A.M. Namboodiri and A.K. Jain, "Online handwritten script recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, **26** (1) (2004) 124–130.
26. H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition", *IEEE Signal Processing*, **16** (5) (1999) 64–83.
27. L. Oudot and L. Prevost and M. Milgram, "An activation-verification model for on-line texts recognition", *Proc. 9th Int. Workshop Frontiers in Handwriting Recognition*, Tokyo, Japan, 2004, pp. 485–490.
28. J. Park and V. Govindaraju, "Use of adaptive segmentation in handwritten phrase recognition", *Pattern Recognition*, **35** (1) (2002) 245–252.
29. R. Plamondon and S.N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey", *IEEE Trans. Pattern Analysis and Machine Intelligence*, **22** (1) (2000) 63–84.
30. S. Quiniou and E. Anquetil, "A priori and a posteriori integration and combination of language models in an on-line handwritten sentence recognition system", *Proc. 10th Int. Workshop Frontiers in Handwriting Recognition*, La Baule, France, 2006, pp. 403–408.
31. E.H. Ratzlaff, "Inter-line distance estimation and text line extraction for unconstrained on-line handwriting", *Proc. 7th Int. Workshop Frontiers in Handwriting Recognition*, Amsterdam, North-Holland, 2000, pp. 33–42.
32. G. Seni and E. Cohen, "External word segmentation of off-line handwritten text lines", *Pattern Recognition*, **27** (1) (1994) 41–52.
33. M. Shilman and Z. Wei and S. Raghupathy and P. Simard and D. Jones, "Discerning structure from freeform handwritten notes", *Proc. 7th Int. Conf. Document Analysis and Recognition*, Edinburgh, Scotland, 2003, pp. 60–65.
34. A. Stolcke, "SRILM - an extensible language modeling toolkit", *Proc. 7th Int. Conf. Spoken Language Processing*, Denver, CO, 2002, pp. 901–904.
35. T. Varga and H. Bunke, "Tree structure for word extraction", *Proc. 8th Int. Conf. Document Analysis and Recognition*, Seoul, South Korea, 2005, pp. 352–356.
36. M. Ye and P. Viola and S. Raghupathy and H. Sutanto and C. Li, "Learning to group text lines and regions in freeform handwritten notes", *Proc. 9th Int. Conf. Document Analysis and Recognition*, Curitiba, Brazil, 2007, pp. 28–32.
37. B. Yu and A.K. Jain, "A robust and fast skew detection algorithm for generic documents", *Pattern Recognition*, **29** (10) (1996) 1599–1630.