# Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models

Alessandro Vinciarelli, Samy Bengio and Horst Bunke

**Abstract**

This paper presents a system for the offline recognition of large vocabulary unconstrained handwritten texts. The only assumption made about the data is that it is written in English. This allows the application of Statistical Language Models in order to improve the performance of our system. Several experiments have been performed using both single and multiple writer data. Lexica of variable size (from 10,000 to 50,000 words) have been used. The use of language models is shown to improve the accuracy of the system (when the lexicon contains 50,000 words, error rate is reduced by ∼50% for single writer data and by ∼25% for multiple writer data). Our approach is described in detail and compared with other methods presented in the literature to deal with the same problem. An experimental setup to correctly deal with unconstrained text recognition is proposed.

**Keywords**

Offline Cursive Handwriting Recognition, Statistical Language Models, $N$-grams, continuous density Hidden Markov Models.

## I. Introduction

OFFLINE cursive handwriting recognition systems presented in the literature deal, almost without exception, with single words [1][2][3]. This happened, in our opinion, because research focused on application domains (handwritten address recognition and bank check reading) where only single words are involved. Moreover, only recently datasets allowing experiments on the recognition of texts were made available [4][5][6]. In few works (see section II) postal addresses or legal amounts have been recognized as phrases. The solution of such problems relies on important information coming from the application environment (zip code, courtesy amount, etc.). Moreover, the address lines and the courtesy amounts have a structure that is not very variable. In both cases the data is made of few

A.Vinciarelli and S.Bengio are with IDIAP (Dalle Molle Institute for Perceptual Artificial Intelligence), Rue du Simplon 4, 1920 Martigny, Switzerland. E-mail: {vincia,bengio}@idiap.ch. H.Bunke is with University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland. E-mail: bunke@iam.unibe.ch.

recurring elements that change at most their order in the phrase. The lexica involved are relatively small (20-30 words for bank checks and 10-1,000 for address lines) and, even more important, are closed. This means that all words appearing in the data are represented in the lexicon.

This work presents an offline cursive recognition system dealing with large vocabulary unconstrained handwritten texts. This application is different from the above mentioned ones under several aspects. The variability in the texts is much higher in terms of phrase structure. The only hypothesis that can be made about an unconstrained text is that it is written in a certain language. No other constraints can be applied to make the problem easier. An unconstrained text can have any kind of content and this makes it very difficult to select a lexicon. The dictionary must be extracted from data independent from the test set and, regardless of its size, it will never contain all the words represented in the data to be recognized. In other words, the lexicon cannot be *closed* and the data contains *Out Of Vocabulary* words (OOV's).

The texts we recognize are the transcription of documents belonging to corpora assumed to reproduce the statistics of average English. This allows us to apply Statistical Language Models (SLM's) in order to improve the performance of our system [7]. We used $N$-gram models (the most successful SLM applied until now [8]) of order 1, 2 and 3 (called unigrams, bigrams and trigrams respectively).

Previous works typically preferred the application of syntax based postprocessing in order to include linguistic knowledge in the recognition. This approach has the important limit of separating the recognition of handwritten data from the application of language modeling. For this reason, this work applies a different approach where handwriting recognition and language modeling are tightly integrated and performed at the same time. This represents an important element of novelty with respect to previous works in the literature (see section II) and has several advantages (see section V-C).

When passing from single word to text line recognition, several changes in the experimental setup are necessary. The most important one concerns the selection of the lexicon (see above). In order to deal with unconstrained texts, the lexicon must be large enough to have a high probability of containing the words appearing in the text. On the other hand, if there are too many dictionary entries, the recognition problem becomes difficult. A good trade-off between these two effects must be found. Another important difference is in the way the performance of the system is measured. In single word recognition, a sample is correctly or incorrectly recognized (there is a single kind of error). In text recognition, there are several kinds of error. A word can be not only misclassified, but also deleted. Moreover, words not appearing in the text can be inserted during the decoding. This leads to different performance metrics that must be selected depending on the task the system is used for. Several experiments (changing the size of the lexicon from 10,000 to 50,000 and the order of the language model from 1 to 3) were performed over both single and multiple writer data. Part of the data sets used are publicly available in order to allow a comparison with other systems.

To our knowledge, the recognition of unconstrained texts has been addressed in very few works [9][10]. The approach used in [9] is significantly different from ours (see section II for more details). The system described in [10] does not apply trigrams and uses a closed vocabulary which leads to an overestimation of the performance. Part of the results presented in this work, have been shown in [11].

The rest of this paper is organized as follows: Section II gives an overview of the works dealing with word sentences, Section III provides the statistical foundations of our approach, Section IV presents SLMs and $N$-gram models, Section V describes the recognition system used in our work, Section VI reports experiments and results obtained and Section VII draws some conclusions.

## II. Previous Work

This section presents a survey of the works dedicated to the recognition of handwritten word sequences or phrases. They can be grouped into three classes depending on the kind of data involved: postal addresses, courtesy amounts and unconstrained texts.

The recognition of legal amounts on bank checks involves small lexica (20-30 words) and a very low variability. Even if the amounts change, their structure is always the same. Moreover, the courtesy amount (amount written in digits) can be used to support and validate the recognition. In most cases, the recognition systems simply transcribe the different parts of the amount separately (e.g. *three hundred fifty five*), but in few works the fact that the amount can be considered as a phrase is used. In [12][13] the amount is first segmented into single words that are recognized separately, then a syntactical postprocessing is applied in order to discard solutions that are not plausible from a linguistic point of view. Such an approach is effective in this problem because the grammars modeling the amounts are relatively simple. The use of the same approach in problems involving more variable data (especially unconstrained texts) is less effective or requires the use of tags grouping the words into few classes (e.g. *verbs*, *adjectives*, etc.).

The goal of postal address recognition is the selection of a delivery point, i.e. of a postal plant where a certain mail piece must be routed in order to reach its recipient. An address line contains several elements: a topographic name (e.g. *street* or *road*), a street name and a number. Different versions of the same element (e.g. *Street*, *St.* and *Str.*) and different ordering of the elements are the main sources of variability. On the other hand, only the street name is relevant to the delivery point selection. This is thus the only information that needs to be correctly transcribed. Two important constraints help to solve the problem. The first is that the number of elements in an address line is very limited (three or four, rarely more) and that always the same elements are present. The second is that the dictionaries are closed, i.e. no OOV's are expected.

Several approaches have been proposed to the problem. In [14][15][16], a system that segments the lines into words and then recognizes them separately is presented. In this way, the phrase recognition is reduced to a sequence of single word recognition steps. The main limit of the approach is the segmentation of a line into words. This is a difficult process and requires a large effort. Moreover, the segmentation process is not perfect and represents an important source of error: the method described in [15] incorrectly segments around 10% of the lines.

An alternative approach is proposed in [17][18], where a keyword spotting technique is used to detect the name of the street. The important advantage of such approach is that no segmentation of the line into words is required. The keyword spotting step obtains in fact the segmentation as a byproduct. The use of keyword spotting is possible because it is not necessary to recognize other words than the street name. The topographic name can be recognized only to apply lexicon reduction mechanisms: once the topographic name is recognized (e.g. *Road*), all street names associated with different topographic entities (*Street*, *Avenue*, etc.) are discarded.

Few works were dedicated to the recognition of unconstrained texts. Such a problem assumes that no hypothesis can be made about the content of the data except for the fact that they are written in a certain language. This allows the application of language models. Also in this case, several approaches have been proposed to solve the problem.

An approach based on segmentation and recognition of single words is proposed in [19][20][21]. After the words of a line have been recognized, a linguistic postprocessing is applied in order to use context information. The words are tagged following gramatical categories and a model of the transition between different categories is used to reweight the recognition scores. Since the segmentation is not a perfect process, several segmentation hypotheses must be considered. The segmentation poses the same problems described above and it is an important source of error. It is not clear in the cited works how the lexica were selected. It is especially important to know whether the lexicon is closed (no OOVs expected) or not. A closed dictionary is a simplifying condition, but is a strong constraint on the texts that can be recognized. Moreover, in order to have a closed dictionary, it is necessary to have information about the test set when building the system. This is not realistic when working on unconstrained texts.

An approach based on HMMs and $N$-grams has been proposed in [10]. This approach has the important advantage of avoiding the segmentation (which is a byproduct of the recognition). The main limit of this work is the use of a closed vocabulary of small size (less than 10,000 words). The lexicon has been extracted from the data used to test the system and this is not correct because no information coming from such set should be used when building the system. Moreover, the resulting system is overfitted to the test set leading to an overestimation of the performance. In such a configuration the system is also not robust with respect to a change of data. When moving to data having a different topic, most of the words will not be represented in the lexicon and the recognition performance will be dramatically decreased.

## III. Statistical Foundations

This section describes the problem of handwritten text recognition from a statistical point of view. The handwritten pages are split into lines before recognition and each line is decoded separately. Since the system is supposed to recognize unconstrained texts, no information is available about the content of the data. The only hypothesis made about the lines is that they are written in English. They are thus supposed to respect, on average, the statistics of any fragment of English text. The above aspects will be shown to have an important influence on the recognition results.

When the recognition is performed offline, only the image of the handwritten data is available. The image is converted into a sequence $O = (o_1, o_2, \ldots, o_m)$ of observation vectors and the recognition task can be thought of as finding a word sequence $\hat{W}$ maximizing the a-posteriori probability:

$$\hat{W} = \arg\max_W p(W|O) \tag{1}$$

where $W = (w_1, w_2, \ldots, w_n)$ is a sequence of words belonging to a fixed vocabulary $V$. By applying Bayes theorem, Equation (1) can be rewritten as follows:

$$\hat{W} = \arg\max_W \frac{p(O|W)p(W)}{p(O)} \tag{2}$$

and since $O$ is constant during recognition:

$$\hat{W} = \arg\max_W p(O|W)p(W). \tag{3}$$

The right side of Equation (3) shows the role of the different sources of information in the recognition problem. The term $p(O|W)$ is the probability of the observation sequence $O$ being generated by a model of sentence $W$. This probability is estimated with HMMs.

If $W$ is composed of $n$ words and the size of the dictionary is $|V|$, then the number of possible word sequences is $|V|^n$. Even for small values of $|V|$ and $n$, this amounts to a huge number, making the task of the recognizer difficult. Moreover, $n$ is not known in advance and such amount must be thus summed over all possible values. The term $p(W)$ provides an a-priori probability of the word sequence $W$ being written and it is often estimated using a *Statistical Language Model*. A good SLM can significantly constrain the search space so that all the sentences that are unlikely to be written (from a linguistic point of view) have a low probability.

In the single word recognition problem, $p(W)$ is typically a uniform distribution over the words of the lexicon. This means that the recognition relies only on the HMMs and Equation (3) corresponds to:

$$\hat{w} = \arg\max_{w} p(O|w) \tag{4}$$

where $w$ is a word belonging to the lexicon. In the next section, SLMs and $N$-gram models in particuar are described in detail.

## IV. Statistical Language Models

Statistical Language Modeling involves attempts to capture regularities of natural language in order to improve the performance of various natural language applications, e.g. Information Retrieval, Machine Translation and Document Classification [7]. This section is focused on the use of SLMs in our specific problem, i.e. the decoding of handwritten texts. As shown in Equation (3), the SLM is supposed to give the a priori probability of a certain sentence to be written [8][22]. If $W$ contains $n$ words, $p(W)$ can be decomposed as follows:

$$p(W) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}) = \prod_{i=1}^{n} p(w_i|h_i) \tag{5}$$

where $w_1^{i-1} = (w_1, \ldots, w_{i-1})$ and $h_i$ is referred to as *history* of word $i$.

The fact that the probability of word $w_i$ being written depends only on the previous words of the sentence makes decomposition in Equation (5) especially suitable for Viterbi decoding [23]. However, Equation (5) poses an important problem: for a vocabulary of reasonable dimension (in the order of tens of thousands), most of the possible histories appear too few times or even never in a reasonable training set. This does not allow the application of a statistical approach. The solution to such problem is to group the histories into a tractable number of equivalence classes. Equation (5) can then be rewritten as follows:

$$p(W) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}) = \prod_{i=1}^{n} p(w_i|\Phi(h_i)) \tag{6}$$

where $\Phi : \{h\} \rightarrow C$ associates a history $h$ to an equivalence class belonging to a finite set $C$.

The nature of $\Phi(h)$ allows one to distinguish between the different SLM techniques presented in the literature (see [7] for an extensive survey). In most cases, $\Phi(h)$ incorporates some linguistic knowledge. Attempts were made to replace the words $w_i$ with corresponding classes $C(w_i)$ obtained through word clustering. Decision Trees, Classification and Regression Trees and different kinds of grammars were applied. The $N$-grams have an important advantage: they can be easily included in a decoding tehnique that integrates recognition and language modeling (see section V-C). The integration of other language models into decoding algorithms has been tried in speech recognition and it leads to poor performance improvements with respect to the increase of complexity it requires [8]. For this reason, $N$-grams are preferred in this work.

In the next three subsections, $N$-gram models, smoothing techniques and an SLM performance measure are analyzed.

### *A. N-gram Language Models*

An $N$-gram model makes an equivalence class out of all the histories ending with the same $N-1$ words:

$$p(W) = \prod_{i=1}^{n} p(w_i|w_{i-N+1}^{i-1}) \tag{7}$$

where $N$ is called the *order* of the model. Even for low orders, the number of equivalence classes quickly becomes intractable. In practice, only unigrams, bigrams and trigrams are used. The probabilities $p(w_i|w_{i-N+1}^{i-1})$ are estimated by simply counting the number of times a certain sequence of $N$ words appears in a corpus of training texts:

$$p(w_i|w_{i-N+1}^{i-1}) = \frac{C(w_{i-N+1}^{i})}{C(w_{i-N+1}^{i-1})} \tag{8}$$

where $C(.)$ is the number of times the argument is counted.

This corresponds to a Maximum Likelihood (ML) estimation of the probabilities $p(w_i|w_{i-N+1}^{i-1})$. In other words, the model resulting from Equation (8) maximizes the likelihood of the training set of the corpus used to obtain the Language Models. This leaves an important problem open: all the sequences of $N$ words not appearing in the training text have zero probability. Moreover, many $N$-grams appear too few times (a frequence threshold $a$ is set empirically) to allow a good statistical estimation of their probability $p(w_i|w_{i-N+1}^{i-1})$. This is problematic because, in the case of unconstrained texts, no corpus is wide enough to contain all possible $N$-grams. Evaluations performed over a patent description dataset (around 1.8 million words) showed that, when splitting the data into training (1.5 million words) and test sets (0.3 million words), only 77% of the trigrams represented in the training set were also represented in the test set [22] (although the database is homogeneous). In order to solve this problem, the models are smoothed, i.e. the probability mass estimated over $N$-grams appearing more than $a$ times with eq. (8) is redistributed across all possible sequences of $N$ words. In such a way a model trained over a certain corpus can be used for any other text, but $N$-grams that are not possible from a linguistic point of view will also have non-zero probability. This is the main limitation of $N$-gram models.

### *B. Smoothing*

Smoothing is supposed to redistribute the probability mass estimated with ML across all possible sequences of $N$ words. Many smoothing techniques have been proposed in the literature, but none of them seems to be systematically superior to the others [24]. In this work we selected a combination of *discounting* and *backing-off* following the schema proposed in [25].

Discounting is necessary because ML overestimates the frequences of the represented $N$-grams and correspondingly underestimates the frequences of the non-represented ones. After discounting is applied, the frequence $r$ of a certain word sequence is changed to $r^*$ according to:

$$r^* = \frac{n_{r+1}}{n_r}(r+1) \tag{9}$$

where $n_r$ is the number of events with frequence $r$. This is called Good-Turing discounting (originally proposed in [26]) and is based on the inverse proportionality between $r$ and $n_r$. This property of many natural phenomena is known as Zipf Law and, in the case of texts, means that words appearing few times are more numerous than those appearing many times. The Good-Turing algorithm was selected because, being based on a natural law, it makes the discounting strategy more robust with respect to a change of data. Several discounting strategies are available in the literature and their performances are essentially equivalent [24].

The *amount of frequence* $r - r^*$ (summed up over all events) is redistributed across all possible $N$-grams not detected (or appearing less than a certain number of times) in the training text. In the most simple case, the redistribution is performed uniformly. Such a solution is not optimal because it does not take into account information present in the training corpus. For this reason, the redistribution is typically made through the so-called *back-off*. The trigram version of the back-off can be represented as follows:

$$p(w_i|w_{i-2}^{i-1}) = \begin{cases} p(w_i|w_{i-2}^{i-1}) & \text{if} \quad C(w_{i-2}^{i}) > a \\ \alpha_1 p(w_i|w_{i-1}) & \text{if} \quad C(w_{i-2}^{i}) \leq a \\ & \text{and} \quad C(w_{i-1}^{i}) > b \\ \alpha_2 p(w_i) & \text{otherwise.} \end{cases} \tag{10}$$

This means that, when an $N$-gram is not represented, lower order models are used. The constants $a$ and $b$ must be set empirically. The coefficients $\alpha_i$ ensure that the probabilities of the non represented trigrams sum up to the amount of frequence to be redistributed. By using bigram and trigram statistics, the probability is no longer redistributed uniformly, but according to the information extracted from the corpus.

### *C. Perplexity*

The perplexity (PP) is the measure most commonly used to evaluate the performance of a language model. The PP is estimated as follows:

$$PP = 2^H \tag{11}$$

where $H = \frac{1}{n}\sum_i \log p(w_i|h_i)$ is an estimate of the entropy of the model (measured over a text containing $n$ words). The PP is the average branching factor of the model, i.e. the average number of words having a probability significantly higher than zero at each step of the decoding [8]. In a problem where all the words have the same probability of being written, the PP corresponds to the size of the lexicon. For this reason, the PP is often interpreted as the dictionary size in the case of single word recognition. Such interpretations show that the lower the perplexity the better the model.

Unfortunately, the relationship between the PP of an SLM and the recognition rate of the system using it cannot be modeled clearly [27]. A decrease of the PP does not necessarily lead to a better recognition rate for a system (it can even have negative effects) and vice-versa.

## V. The recognition system

This section presents the recognition system used in this paper. The recognizer was originally developed to work on single words [28][29], but no modifications were necessary to use it for handwritten texts. The system is based on a sliding window approach: a fixed width window shifts column by column from left to right and at each position a feature vector is extracted. The sequence of vectors so obtained is modeled with continuous density Hidden Markov Models. In the next two subsections, the single steps of the recognition process are shown in detail.

### *A. Normalization and Feature Extraction*

The first step of the recognition is the normalization, i.e. the removal of slant (the angle between the vertical direction and the direction of the strokes supposed to be vertical in an ideal model of handwriting) and slope (the angle between the horizontal direction and the direction of the line the words are aligned on). The normalization technique we applied is described in detail in [30].

The slope removal method gives first a rough approximation of the core region. The stroke minima closer to its lower limit are used to fit the *baseline*, i.e. the line on which the words are aligned. The image is rotated until the estimated baseline is horizontal and the resulting image is desloped. The deslanting is based on the hypothesis that, when the word is deslanted, the number of columns containing a continuous stroke is maximum [30]. A shear transform corresponding to all angles in a reasonable interval is applied. The following histogram is collected from each shear transformed image:

$$H_\alpha(i) = \frac{v(i)}{\Delta y(i)} \tag{12}$$

where $v(i)$ is the vertical density in column $i$ (number of foreground pixels in column $i$), and $\Delta y(i)$ is the distance between the highest and lowest pixel in the same column. When the column contains a continuous stroke, $H_\alpha(i) = 1$, otherwise $H_\alpha(i) \in [0, 1[$. The *deslantedness* of the shear transformed image corresponding to angle $\alpha$ is measured with the following function:

$$S(\alpha) = \sum_{\{i:H_\alpha(i)=1\}} v(i)^2. \tag{13}$$

The use of $v(i)^2$ rather than $v(i)$ enhances the contribution of the longer strokes. The slant angle is estimated as follows:

$$\alpha^* = \arg\max_\alpha S(\alpha). \tag{14}$$

The normalization method described above has the important advantage of avoiding the use of any parameter to be set manually. This allows one to avoid the heavy experimental effort needed to find the optimal parameter set. Moreover, the method is shown to improve the recognition rate of the system with respect to more traditional techniques [30].

After the normalization, a window shifts column by column from left to right and, at each position, a feature vector is extracted. This approach avoids the segmentation (a difficult and error prone process) and allows the system to convert the text image into a sequence of vectors. The feature extraction is described in [28][29]: the area in the window content actually containing pixels is first isolated, then partitioned into cells regularly arranged in a $4 \times 4$ grid. The number $n_i$ of foreground pixels in each cell is counted and a feature vector is obtained as follows:

$$\mathbf{f} = (\frac{n_1}{N}, \frac{n_2}{N}, \ldots, \frac{n_{16}}{N}) \tag{15}$$

where $N$ is the total number of foreground pixels in the window. This feature extraction method is based on averaging rather than on exact reconstruction of the pattern in order to be more robust with respect to noise and cursive variability.

### *B. Recognition*

HMMs are probability density functions over the space of vector sequences. This makes them suitable to model the feature vectors extracted from the handwritten data. We use continuous density HMMs (for a detailed introduction see [8][31]) where the emission probability is expressed with mixtures of Gaussians. A different model is obtained for each character and models for strings can be obtained by concatenating single character models.

Several approximations are applied: the first one is that, although certain characters are longer than others, all the letter models have the same number $S$ of states and $G$ of Gaussians per state. The parameters $S$ and $G$ are selected through a validation procedure described in section VI-C. The second one is that the same model is used for both upper and lower case letters. Such approximation is made because the number of available capital letters in our datasets is not sufficient for a reliable training. Moreover, the upper and lower case versions of the same letter are often similar and differ only for their dimension (e.g. $p$ and $P$). If a scale invariant feature set is used (as in our case), this allows one to make a single class out of the two letter versions. The third one is that only the letters are modeled, the other symbols appearing in the text lines (punctuation marks, parentheses, digits, etc.) are considered as noise. This approximation is applied for two reasons: the first one is that not enough of these symbols samples are available for a good model training. The second one is that, by simply recognizing the words, the system does not give a perfect transcription of the text, but allows the application of content based techniques (Topic Detection, Document Categorization, Information Retrieval) where only the words play a role.

The models are trained using the Baum-Welch algorithm (a specific case of Expectation-Maximization) that maximizes the likelihood of the training set given the models [32][33][34]. The recognition is performed using the Viterbi algorithm [8][23]. This gives the best likelihood $\lambda$ that can be obtained by following a unique state sequence with the vectors extracted from the handwritten data. The state sequence giving the highest value of $\lambda$ is selected as transcription of the handwritten data.

### C. Decoding

The approaches used until now to recognize word sequences never integrate handwriting recognition and language modeling (see section II). The recognition process is typically split into two parts: first the handwritten data is transcribed, then a language model is applied to estimate how likely (from a linguistic point of view) the recognition result is. This approach has several problems: typically, it requires a segmentation into single words, a difficult and error prone process. Moreover, (especially when using grammar based models) the top scoring transcription is often not enough. The $N$-best list is necessary in order to identify the transcriptions most likely from a linguistic point of view and this requires a significant amount of computation.

This work proposes a decoding technique where handwriting recognition and language modeling are tightly integrated. This has several advantages: first, no segmentation is necessary because the likelihood of a certain point being the separation between two words can be estimated during the decoding. In this way, the segmentation is a byproduct of the recognition and not a process to be performed independently. Moreover, the sentence having the best likelihood is the one being at the same time the most likely from both handwriting and language point of view.

The decoding can be thought of as the search of the path optimally matching the observation sequence in the so-called *search space*. In our case, the search space is a network of word HMMs where the transition between different words is modeled by the language model $P(W)$. The search must be performed at both state and word levels. At each observation vector, the state-word pair belonging to the globally most likely path must be found. This task is performed with the Viterbi Algorithm. The algorithm is based on a recursion that exploits the Viterbi approximation in equation (3) and Bellman's Principle of Optimality [35].

The emission probability densities give the likelihood of the $t^{th}$ observation vector $\mathbf{o_t}$ given a state $s$. In our system, the words are modeled as letter sequences and the number of states to check is given by $S \times M$, where $S$ is the number of states per letter model (see section VI) and $M$ is the total number of models (in our case $M = 27$, 26 models for letters and 1 for blank space). Since we are recognizing words, we need to know not only the probability that a vector has been emitted by a certain letter state, but also that the letter belongs to a word $w$. This probability is defined by $Q(t,(s;w))$ and is estimated as follows:

$$Q(t,(s;w)) = \max_{s'}\{p(\mathbf{o_t}|(s;w))p((s;w)|(s';w)) \cdot Q(t-1,(s',w))\} \tag{16}$$

where $s' \neq 0$, $p(\mathbf{o_t}|(s;w))$ is the emission probability of state $s$ in word $w$ and $p((s;w)|(s';w))$ is the transition probability between states $s'$ and $s$ in word $w$. This equation applies to all intra-word states, but when the transition is between the last state of the last letter of a word $w$ and the first state of the first letter of the next word, then the language model term must be added:

$$Q(t,(s=0;w)) = max_{\mathbf{h}}\{p(w|\mathbf{h}) \cdot Q(t,(S,\mathbf{h}))\} \tag{17}$$

where $s(0;w)$ and $(S;\mathbf{h})$ denote the first state of word $w$ and last state of history $\mathbf{h}$ respectively. The history is made of $N-1$ words when an $N$-gram model is being applied. The above equation corresponds to equation (3) where $Q(t,(s;w))$ is the likelihood of a subset of the observation sequence being produced by word model $w$ and $p(w|\mathbf{h})$ is the language model.

The two above equations define the Dynamic Programming recurrence equations for searching through the state network. The difference with respect to other approaches (see section II) is that the hypothesis of being at the transition between a word and the following one is measured at each observation vector. For this reason, it is not necessary to segment the sentence into words. Moreover, the score of the word boundary hypothesis is not based on a local measure, but on the score of the whole path leading to that specific node of the network.

At each node of the network, the information about the best path leading to it is retained. In this way, when the last observation is reached and the most likely state it corresponds to is found, it is possible to backtrack the sequence of states leading to it. The sentence corresponding to the sequence of states is the transcription of the handwritten line.

One of the most important aspects of this approach is that the best path is identified by taking into account the whole observation sequence. No decision is taken before reaching the last observation vector. This is important because any error determined by local problems (e.g. a letter written ambiguously or a small scratch) can be recovered by using the rest of the line. On the other hand, this forces the system to use the whole sequence to build the search space network. Its dimension (i.e. the number of nodes it contains) grows with the number of vectors in the sequence and, when the sequence is too long, the computational burden becomes too high.

This is a limit for systems producing, as in our case, many observations per unit length: if the number of vectors per line is high, it is not possible to concatenate several lines and this limits the performance of $N$-grams with higher order $N$. In fact, a line contains on average around ten words and only the words after the $N-1^{th}$ one can take an actual advantage from the $N$-grams. This results in a saturation of the improvement given by the $N$-grams when $N$ increases. A posible solution is the inclusion of the previous line only at the linguistic level. The simplest solution is to keep memory of the last $N-1$ words in the previous line, but this can have a negative influence. In fact, if the last words of the previous line are recognized incorrectly, they can lead to very low probabilities for the actually written words. The error can then propagate and decrease the recognition performance. On the other hand, the development of more sophisticated techniques to include the previous line at the linguistic level can be difficult and has an important drawback. The optimal path would no longer be the one maximizing the probability in equation (3) because the term $p(O|W)$ would not include the last observations of the previous line.

## VI. Experiments and Results

Experiments were performed using three different data sets. The first is composed of a text written by a single person and is publicly available on the web[1]. The second is composed of pages written by many persons and can be obtained at University of Bern [4][6]. The third is composed of documents extracted from the the Reuters-21578 text database [36]. These three data sets will be referred to as *Cambridge*, *IAM* and *Reuters* databases respectively. The Cambridge database was originally presented in [5] and contains 353 handwritten text lines split into training (153 lines), validation (83 lines) and test (117 lines) sets. The lines are kept in the same order as they were written. This reproduces a realistic situation where the data available at a certain time is used to obtain a system capable of recognizing the data that will be written in the future.

A subset was randomly selected from the IAM database. It was split into training, validation and test sets containing 416, 206 and 306 lines respectively. The data is split in such a way that the writers represented in the training set are not represented in the test set. This is assumed to reproduce a realistic situation where the data produced by a certain number of writers is used to recognize the data written by other persons.

The Reuters database contains single writer data and is composed of 70 documents. The documents are ordered temporally since they are extracted from a news bulletin and they have a date. For this

[1] The data can be downloaded at the following ftp address: `ftp.eng.cam.ac.uk/pub/data`.

reason we selected the first 30 documents for training and the last 40 for testing. The number of lines in training, validation and test set is 255, 101 and 447 respectively. The language models for Cambridge and IAM databases were obtained using the TDT-2 Corpus [37], a collection of news transcriptions obtained from several journals and broadcasting companies. The corpus is completely independent of the texts in the handwriting data sets. In this way, the language models are not fitted to the specific texts they have to model and the experimental setup reflects the actual condition of unconstrained text recognition. The language models for the Reuters database were obtained using the Reuters-21578 corpus [36]. In order to have language models independent of the specific texts being recognized (see above), the documents belonging to our handwriting data set have been eliminated from it.

In the next subsections we show in more detail how the lexicon was selected (section VI-A), how the language models were trained and tested (section VI-B) and the results obtained over the Cambridge (section VI-C), IAM (section VI-D) and Reuters (section VI-E) databases.

### *A. Lexicon Selection*

In single word recognition, the lexicon is always implicitly assumed to cover 100% of the data. Every handwritten word is supposed to be in the dictionary and this happens because the lexicon is determined from information coming from the application environment (e.g. the zip code in postal address recognition). This is not the case for the recognition of unconstrained texts. The presence of proper names, technical terms and morphological variations of a single stem makes it impossible to define a *universal* lexicon.

The only available source of information at the linguistic level is the text corpus we use to train the language models. It is therefore reasonable to extract the lexicon from it. The TDT-2 corpus is 20,407,827 words in length. It is composed using 196,209 unique words. In order to build a lexicon of size $M$, we simply selected the $M$ most frequent words in the lexicon. The same technique has been used for the Reuters database which is 1,730,515 words long (for a total of 34,574 unique terms). This approach is based on the hypothesis that the most frequent words in a text are typically *functional words* (prepositions, articles, conjunctions, verbs of common use, etc.), hence we can expect to find them in any other text. Moreover, the use of the most frequent words allows to obtain more reliable estimates of the $N$-gram probabilities. In this way, five lexica have been obtained corresponding to $M$ values ranging from 10,000 to 50,000 (30,000 for the Reuters database) step 10,000 (5,000 for the Reuters database). Such lexica are used in our experiments to show the effect of the dictionary size on the recognition rate. The plot in figure 1 shows the coverage (percentage of text words actually represented in the dictionary) of the lexica obtained with the above mentioned criterion in function of their size. The coverage is measured on the test set of the Cambridge, IAM and Reuters data sets. The coverage represents, for a given lexicon size, the upper limit of system recognition performance using the corresponding lexicon.

### B. $N$-gram Model Training

The $N$-gram models for the IAM and Cambridge databases were trained over the TDT-2 corpus [37], a collection of transcriptions from several broadcast and newswire sources (ABC, CNN, NBC, MSNBC, Associated Press, New York Times, Voice of America, Public Radio International). The language models for the Reuters database were obtained using the Reuters-21578 corpus [36], a newswire economic bulletin. For each one of the lexica described in section VI-A, three models (based on unigrams, bigrams and trigrams respectively) were created. The plots in figure 2, 3 and 4 show the perplexities of the SLMs as a function of the lexicon size. The perplexity is estimated over the part of the text covered by the lexicon, without taking into account OOV words. This happens because the only part of the text where the language model can be effective is the one covered by the lexicon. For this reason, we are interested to know the language model performance only over such part of the text. In practice, when an OOV word is encountered, the history is reset. The first term after the OOV word can be modeled only with unigrams, the second one at most with bigrams and only the third one can make use of trigrams. This increases significantly the perplexity and simulates the fact that the language model can only guess wrong words in correspondence of OOV words. The perplexity measured including the OOV words is lower, but less realistic with respect to the recognition task. In fact, during the recognition, the information about the presence of an OOV word is not available and the model can only guess a wrong word (independently of its perplexity).

The models show similar behaviour over the different sets. A significant improvement is obtained when passing from unigrams to bigrams, but no further improvement is obtained when applying trigrams. This happens for several reasons. The first is that the handwritten text is split into lines and only the words after the third one can take some advantages from the trigram model. Since a line contains on average 10 words, this means that only $\sim$80% of the data can actually benefit from the trigram model (while 90% of the data can be modeled with bigrams).

A second problem is that the percentage of trigrams covered by the corpus in the test set of all databases is $\sim$40%. This further reduces the number of words where the trigram model can have a positive effect. The coverage in terms of bigrams is much higher (around 85%) and the percentage of words over which the model can have an effect is around 90%. On average, when trigrams are applied, $\sim$ 45% of the words in the test set are modeled with a trigram, $\sim$ 40% with a bigram and $\sim$ 15% with a unigram. This results in an average history length of 2.3. On the other hand, when the language is modeled with bigrams, $\sim$ 85% of the words are guessed with bigrams and $\sim$ 15% with unigrams. The resulting average history length is 1.8. For these reasons, the bigram and trigram models have a similar perplexity and do not make a big difference in terms of recognition performance.

The perplexity over the test set of IAM and Cambridge databases (see figures 2 and 3) is much higher than the perplexity over the test set of the Reuters database (see figure 4). The reason is that, for this database, the texts to be recognized have been produced by the same source that also produced the texts used to train the language models. In other words, the language model is better aligned with the data to recognize.

### *C. Cambridge Database Results*

This section reports the results obtained on the Cambridge database. Since it is not possible to set a priori the number of states $S$ and the number of Gaussians $G$ in the models, a validation phase is necessary. Models with $10 \leq S \leq 14$ and $10 \leq G \leq 15$ are trained over the training set and tested, without using SLMs, over the validation set. The system corresponding to the couple $(S, G)$ giving the best results (over the validation set) is selected as optimal. The system selected in the validation phase ($S = 12$ and $G = 12$) is retrained over the union of training and validation set and the resulting system is used in the actual recognition experiments.

For each one of the five lexica described in section VI-A, four versions of the system are tested over the test set. The first version (called *baseline*) makes no use of SLMs, the other ones use unigram, bigram and trigram models corresponding to the lexicon under consideration. The performance is measured using two different metrics: *accuracy* and *recognition rate*. In the recognition of a text, there are several sources of errors. A word can be misclassified and the corresponding error is called a *substitution*. Secondly, it can be split into two parts leading not only to an incorrect transcription, but also to the introduction of a word that does not exist in the original text. In this case the error is referred to as *insertion*. Finally, when the space between two words is missed, they are joined together giving rise to a substitution error and to the disappearance of a word existing in the original text.

The error in this case is called a *deletion*. The different performance metrics depend on the sources of error that are taken into account. The *accuracy* is obtained as $100 - s - d$ (where $s$ is the substitution rate and $d$ is the deletion rate). The accuracy is thus the percentage of words correctly classified. The *recognition* is estimated as $100 - d - s - i$ where $i$ is the insertion rate. The recognition is not a percentage and can be negative. Its highest value is 100. When the task is to correctly transcribe of the text, the recognition is the most appropriate performance measure. It takes into account insertions that represent important deviations with respect to the original text. When the task is related to content modeling (e.g. indexing), the accuracy is a good metric since the only important factor is how many words are correctly classified.

In our experiments we used both metrics: Figure 5 shows the recognition (left plot) and the accuracy (right plot) respectively as a function of the lexicon size. In both cases, the performance is the result of a tradeoff between the improvement of the test set coverage and the increase of the lexicon size. The application of the $N$-gram models has a significantly positive effect on both recognition rate and accuracy (the statistical confidence is higher than 90%). Moreover, the SLMs make the system more robust with respect to the increase of the lexicon size so that it is possible to maximize the benefit of the improved coverage.

The insertions have an important influence on the performance of the system. Sometimes, when part of a word corresponds to an entry in the lexicon (e.g. *unmentionable* is composed of the entries *un*, *mention* and *able*) the decoder favours the transcription splitting the bigger word, especially when the shorter words are more frequently represented in the training corpus. No deletion error is observed. This is due to the fact that the spaces between neighboring words are typically evident, and are never missed (condition necessary to observe a deletion).

The systems using unigrams, bigrams and trigrams are equivalent in terms of performance. This is due, in our opinion, to the fact that the handwriting model alone has a high performance. The space for improvement is thus reduced. Most content words are recognized without the help of the language models. $N$-grams are actually helpful only to recognize functional words that are an important source of error because they are typically short (two or three letters). On the other hand, the performance of the language models over the functional words is not significantly improved by increasing their order. For this reason, the use of bigrams and trigrams does not result in a higher recognition or accuracy.

The situation is different for multiple writer data where the handwriting model alone is weak. In this case, the HMMs have a low performance over the words where $N$-grams of different order have a significantly different effectiveness. This leads to an improvement when passing from unigrams to trigrams.

### D. IAM Database Results

This section describes the results obtained over the IAM database. The parameters $S$ and $G$ were set using the same method as described in the previous section for Cambridge database. Models with $19 \leq S \leq 23$ and $10 \leq G \leq 15$ are trained over the training set and tested, without using SLMs, over the validation set. The selected model ($S = 20$ and $G = 12$) is retrained over the union of training and validation set and it is used in the actual recognition experiments.

The dictionaries and the language models are the same as those used in the single writer experiments. The performance of the systems is measured in terms of accuracy and recognition (see previous section). For each dictionary, four recognizers are tested: the first (called *baseline*) makes no use of language models. The others use alternatively unigrams, bigrams and trigrams.

Also in this case, the use of $N$-grams has a two-fold positive effect: the performance is not only improved (independently of the metric used), but the system is also more robust with respect to an increase of the lexicon size. Figure 6 shows that the performance of the systems using the language models is stable when the lexicon size passes from 10,000 to 50,000, while accuracy and recognition of the baseline system are significantly lowered.

The increase of the language model order produces an improvement (statistical significance higher than 90%). The language models can play a role not only over the functional words (see previous section) but also over the content words where the difference of the order results in a better local perplexity. The error is mostly due to substitution (around 45%). Insertion and deletion rates are about 9% and 4% respectively. No other systems have been tested over the same data, thus no direct comparison is possible. The system presented in [10] has higher accuracy (∼60%), but the lexicon is smaller than ours (∼7,000 words) and it is closed.

### E. Reuters database results

This section shows the results obtained over the Reuters database. The values of the parameters $S$ and $G$ were obtained using the same method applied for the other databases. The models selected through the validation have $S = 11$ and $G = 11$. Since the corpus used to train the language models has only ∼35,000 unique words, lexica of size between 10,000 and 30,000 (step 5,000) were used. The results (see figure 7) show once again the improvement that can be obtained with the language models, but there are some important differences with respect to the previous experiments. The improvement when passing from the baseline system to the systems using language models is much higher. Moreover, the difference between the performance of 1-grams on one side and 2-grams and 3-grams on the other side is more evident. The reason of such differences is, in our opinion, the better alignment of the language model with the data to recognize. On the other hand, the difference between 2-grams and 3-grams remains small and it seems to favor bigrams (trigrams have slightly higher substitution rate). This is probably due to the line by line decoding (see end of section VI-B).

## VII. Conclusions

This work presented a system for the offline recognition of handwritten texts. The recognizer is based on continuous density Hidden Markov Models and Statistical Language Models (unigrams, bigrams and trigrams). Several experiments were performed using both single and multiple writer data. The experimetal setup reproduces the conditions of unconstrained text recognition: no information about the content of the documents to be transcribed is used, except for the fact that they are written in English. The differences with respect to the case of single word recognition are highlighted. Lexica of different sizes, (from 10,000 to 50,000 words) extracted on a statistical basis from the corpus used to train the language models, were used. The performance of the system was measured in terms of accuracy and recognition.

The effect of the SLMs changes depending on the data. In all our experiments, the $N$-grams were able to improve significantly the performance of the system independently of the metric used. On the other hand, to pass from unigrams to trigrams results in an improvement of recognition and accuracy only for IAM and Reuters databases. This happens, in our opinion, because the models trained over Cambridge data have already a very high performance without SLMs. The space left for improvement is thus reduced and limited to a part of the text (namely the functional words) where models of different order have a similar perplexity (see sections VI-C and VI-D).

The improvement obtained with SLM's is especially high for the Reuters database. The reason is, in our opinion, that the language model is better aligned with the data to be recognized. The SLM's have been trained over a corpus of documents produced by the same source (the Reuters newswire bulletin) that created the data to recognize. This makes the system more constrained, but more effective. It must be taken into account the fact that SLM's are penalized in our experimental setup because of an important aspect: the lines are decoded separately, so the models of higher order are effective on a smaller part of the data (see end of section VI-B). A solution to this problem can be the concatenation of different lines before the recognition, but this requires the system to produce few observations per line in order to reduce the dimension of the search space (see section V-C).

In its current configuration, the system appears to be more suitable for the application of content modeling techniques (Indexing, Information Retrieval, Topic Detection, etc.) than for the actual transcription of texts. All symbols other than letters are in fact not modeled and treated as noise. This is due to the fact that only the letters are represented sufficiently for a reliable training. In order to obtain a system more oriented toward the exact transcription of documents, it is necessary to increase the amount of training material (especially for the less represented symbols).

At the present state-of-the-art in language modeling, the possibility of having models performing better than $N$-grams seems unlikely. The application of other language modeling techniques (e.g. grammars or decision trees) seems to be limited to very constrained problems, like bank check reading, date processing or postal address recognition. The possibility of adapting generic $N$-gram models to specific problems can be explored in the same way as writer independent HMMs can be adapted to writer dependent data.

Since very few works have been dedicated to the offline recognition of handwritten texts, the above list of possible future directions is far from being exhaustive. The availability of systems for text recognition can give rise to new applications that were not considered until now. Especially from this point of view, the recognition of handwritten texts represents an interesting and promising research domain.

## Acknowledgements

The authors wish to thank F.Camastra for useful discussions and D.Moore for his collaboration. This work is supported by the Swiss National Science Foundation through the National Center of Competence in Research on Interactive Multimodal Information Management (IM2).

## References

[1] Tal Steinherz, Ehud Rivlin, and Nathaniel Intrator, "Off-line cursive script word recognition - a survey," *International Journal of Document Analysis and Recognition*, vol. 2, no. 2, pp. 1–33, February 1999.

[2] R. Plamondon and S.N. Srihari, "On line and off-line handwriting recognition: A comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, january 2000.

[3] Alessandro Vinciarelli, "A survey on off-line cursive word recognition," *Pattern Recognition*, vol. 35, no. 7, pp. 1433–1446, june 2002.

[4] U.V. Marti and H. Bunke, "The IAM-database: an english sentence database for offline handwriting recognition," *International Journal of Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, january 2002.

[5] Andrew W. Senior and Anthony J. Robinson, "An off-line cursive handwriting recognition system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 309–321, March 1998.

[6] M. Zimmermann and H. Bunke, "Automatic segmentation of the IAM off-line database for handwritten english text," in *Proceedings of $16^{th}$ International Conference on Pattern Recognition*, 2002, vol. IV, pp. 35–39.

[7] R. Rosenfeld, "Two decades of statistical language modeling: where do we go from here?," *Proceedings of IEEE*, vol. 88, no. 8, pp. 1270–1278, august 2000.

[8] F. Jelinek, *Statistical Aspects of Speech Recognition*, MIT Press, 1998.

[9] G. Kim, V. Govindaraju, and S.N. Srihari, "An architecture for handwritten text recognition systems," *Pattern Recognition*, vol. 2, pp. 37–44, 1999.

[10] U.V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 1, pp. 65–90, 2001.

[11] A. Vinciarelli, S. Bengio, and H. Bunke, "Offline recognition of large vocabulary vursive handwritten text," in *Proceedings of International Conference on Document Analysis and Recognition*, 2003.

[12] T. Paquet and Y. Lecourtier, "Recognition of handwritten sentences using a restricted lexicon," *Pattern Recognition*, vol. 26, no. 3, pp. 391–407, march 1993.

[13] D. Guillevic and C.Y. Suen, "Recognition of legal amounts on bank cheques," *Pattern Analysis and Applications*, vol. 1, no. 1, 1998.

[14] E. Cohen, J.J. Hull, and S.N. Srihari, "Control structure for interpreting handwritten addresses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 10, pp. 1049–1055, october 1994.

[15] J. Park and V. Govindaraju, "Use of adaptive segmentation in handwritten phrase recognition," *Pattern Recognition*, vol. 35, pp. 245–252, 2002.

[16] G. Kim and V. Govindaraju, "Handwritten phrase recognition as applied to street name images," *Pattern Recognition*, vol. 31, no. 1, pp. 41–51, january 1998.

[17] M. El Yacoubi, M. Gilloux, and J.M. Bertille, "A statistical approach for phrase location and recognition within a text line: an application to street name recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 172–188, february 2002.

[18] A. El Yacoubi, J.M. Bertille, and M. Gilloux, "Conjoined location of street names within a postal address delivery line," in *Proceedings of International Conference on Document Analysis and Recognition*, 1995, vol. 2, pp. 1024–1027.

[19] G. Kim, V. Govindaraju, and S.N. Srihari, "An architecture for handwritten text recognition systems," *International Journal on Document Analysis and Recognition*, vol. 2, pp. 37–44, 1999.

[20] R.K. Srihari and C.M. Baltus, "Incorporating syntactic constraints in recognizing handwritten sentences," in *Proceedings of International Joint Conference on Artificial Intelligence*, Chambéry, 1993, pp. 1262–1267.

[21] R.K. Srihari, "Use of lexical and syntactic techniques in recognizing handwritten text," in *Proceedings of ARPA Workshop on Human Language Technology*, Princeton, 1994, pp. 403–407.

[22] F. Jelinek, "Self-organized language modeling for speech recognition," in *Readings in Speech Recognition*, A. Waibel and L. Kai-Fu, Eds., pp. 450–506. Morgan Kaufmann, Palo Alto, CA, 1989.

[23] A.J. Viterbi, "Error bounds for convolutionalcodes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, 1967.

[24] Stanley Chen and R. Rosenfeld, "A survey of smoothing techniques for ME models," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 37–50, january 2000.

[25] S.M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[26] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer Speech and Language*, vol. 10, pp. 187–228, 1996.

[27] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Communication*, vol. 38, pp. 19–28, 2002.

[28] A. Vinciarelli and J Lüttin, "Off-line cursive script recognition based on continuous density HMM," in *Proceedings of the $7^{th}$ International Workshop on Frontiers in Handwriting Recognition*. 2000, pp. 493–498, World Publishing.

[29] A. Vinciarelli and S. Bengio, "Offline cursive word recognition using continuous density HMMs trained with PCA or ICA features," in *Proceedings of the $16^{th}$ International Conference on Pattern Recognition*, 2002, pp. 493–498.

[30] A. Vinciarelli and J. Lüttin, "A new normalization technique for cursive handwritten words," *Pattern Recognition Letters*, vol. 22, no. 9, pp. 1043–1050, 2001.

[31] L. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," in *Readings in Speech Recognition*, A. Waibel and L. Kai-Fu, Eds., pp. 267–296. Morgan Kaufmann, Palo Alto, CA, 1989.

[32] L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[33] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[34] L.E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[35] Bellman. R., *Adaptive Control Processes: A Guided Tour*, Princeton University Press, 1991.

[36] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[37] D. Graff, C. Cieri, S. Strassel, and N. Martey, "The TDT-3 text and speech corpus," in *Proceedings of Topic Detection and Tracking Workshop*, 2000.

**A**lessandro Vinciarelli received the Laurea degree in Physics from University of Torino (Italy) in 1994 and the PhD degree from University of Bern (Switzerland) in 2003. He has worked with several companies and research laboratories in Italy and in the United States. In 2001 and 2002 he participated in the Summer Internship Program at IBM Watson Research Center (USA). He is currently research scientist at IDIAP (Dalle Molle Institute for Perceptual Artificial Intelligence) in Switzerland. His interests include Handwriting Recognition, Pattern Recognition, Information Retrieval, Text Categorization. He is author and coauthor of several papers published in conference proceedings and international journals.

**S**amy Bengio is a research director and the machine learning group leader at the Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP) since 1999, where he supervises PhD students and postdoctoral fellows working on many areas of machine learning such as support vector machines, time series prediction, mixture models, large-scale problems, speech and speaker recognition, multi-modal problems, brain computer interfaces, and many more. He has obtained his PhD in computer science from Universite de Montreal (1993), and spent three post-doctoral years at CNET, the research center of France Telecom, and INRS-Telecommunications (Montreal). He then worked as a researcher for CIRANO, an economic and financial academic research center, applying learning algorithms to finance. Before joining IDIAP, he was also research director at Microcell Labs, a private research center in mobile telecommunications. His current interests include all theoretical and applied aspects of learning algorithms.

**H**orst Bunke received his M.S. and Ph.D. degrees in Computer Science from the University of Erlangen, Germany. In 1984, he joined the University of Bern, Switzerland, where he is a professor in the Computer Science Department. He was Department Chairman from 1992 - 1996 and Dean of the Faculty of Science from 1997 to 1998. From 1998 to 2000 Horst Bunke was 1st Vice-President of the International Association for Pattern Recognition (IAPR). In 2000 he also was Acting President of this organization. Horst Bunke is a Fellow of the IAPR, former editor-in-charge of the International Journal of Pattern Recognition and Artificial Intelligence, editor-in-chief of the journal Electronic Letters of Computer Vision and Image Analysis, editor-in-chief of the book series on Machine Perception and Artificial Intelligence by World Scientific Publ. Co., associate editor of Acta Cybernetica, the International Journal of Document Analysis and Recognition, and Pattern Analysis and Applications. He was on the program and organization committee of many conferences and served as a referee for numerous journals and scentific organizations. He has more than 450 publications, including 28 books and special editions of journals.

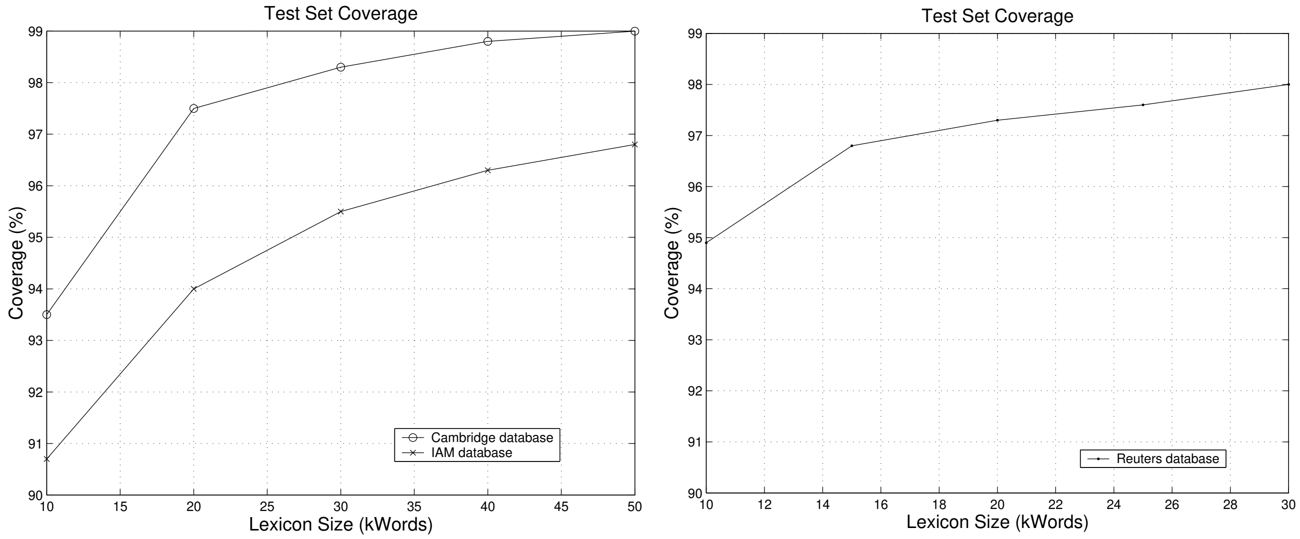


Fig. 1. Test set coverage vs lexicon size. The plots show the percentage of words covered by a lexicon in the test set as a function of the lexicon size. Cambridge and IAM databases are shown in the left plot (they share the same lexicon), Reuters database is shown in the right plot.

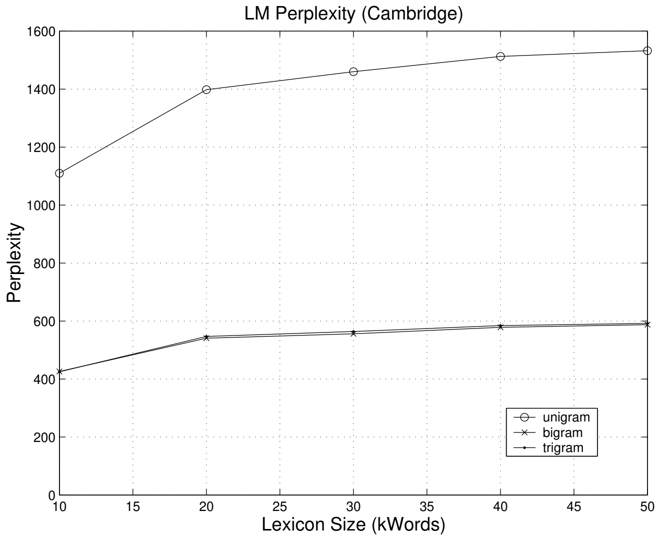


Fig. 2. Language Model perplexity. The plot shows the perplexity of the language models over the test set of the Cambridge database as a function of the dictionary size.

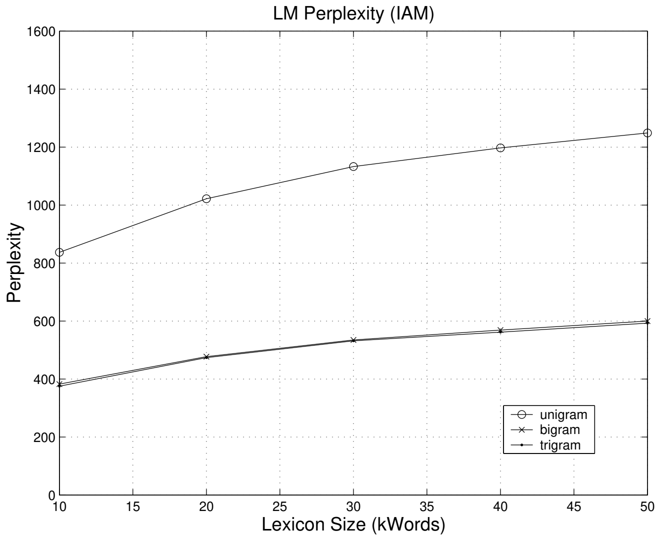


Fig. 3. Language Model perplexity. The plot shows the perplexity of the language models over the test set of the Bern database as a function of the dictionary size.

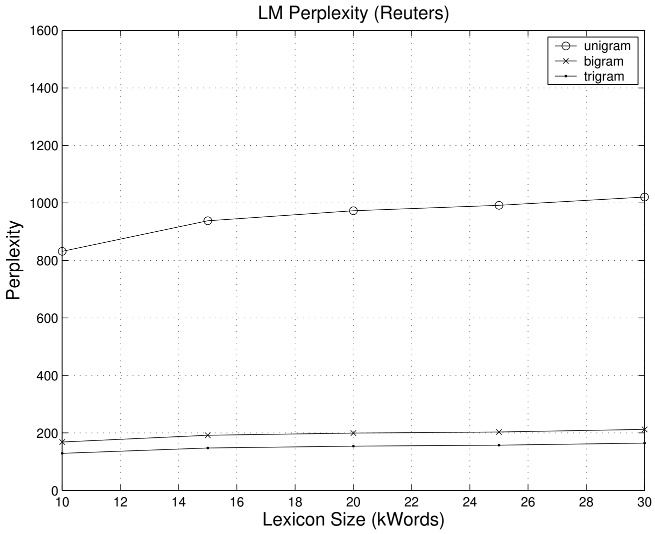


Fig. 4. Language Model perplexity. The plot on the left shows the perplexity of the language models over the test set of the Reuters database as a function of the dictionary size.

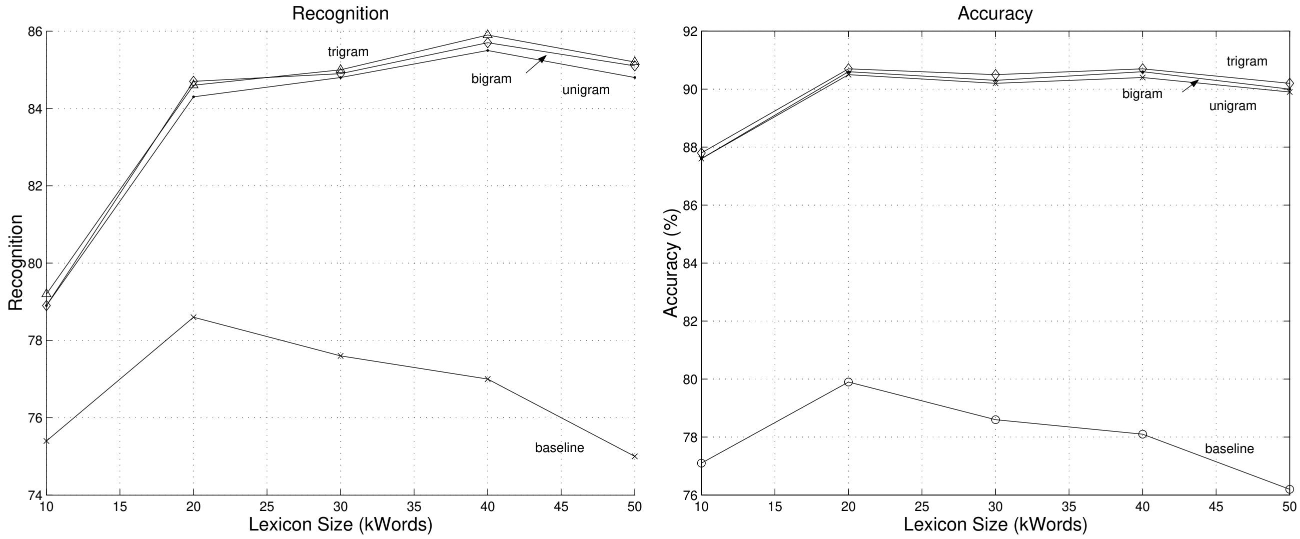


Fig. 5. System performance over Cambridge database. The plot on the left (right) shows the recognition rate (accuracy) of the system. The performance is measured over the test set for the four systems considered: baseline (no SLM), unigrams, bigrams and trigrams.

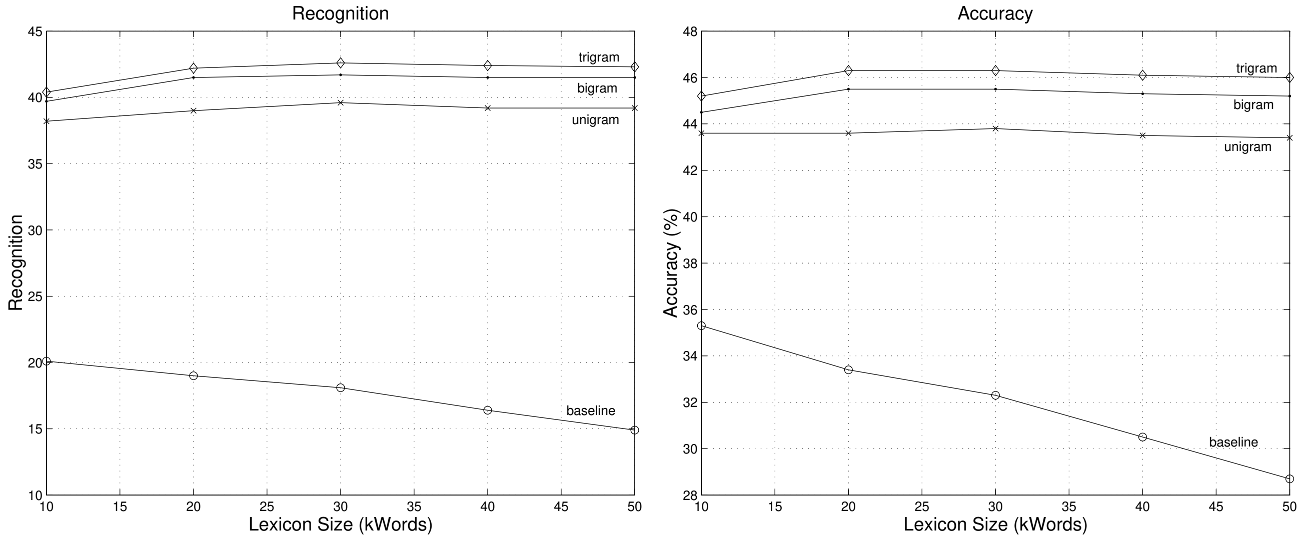


Fig. 6. System performance over IAM database. The plot on the left (right) shows the recognition (accuracy) of the system. The performance is measured over the test set for the four systems considered: baseline (no SLM), unigrams, bigrams and trigrams.

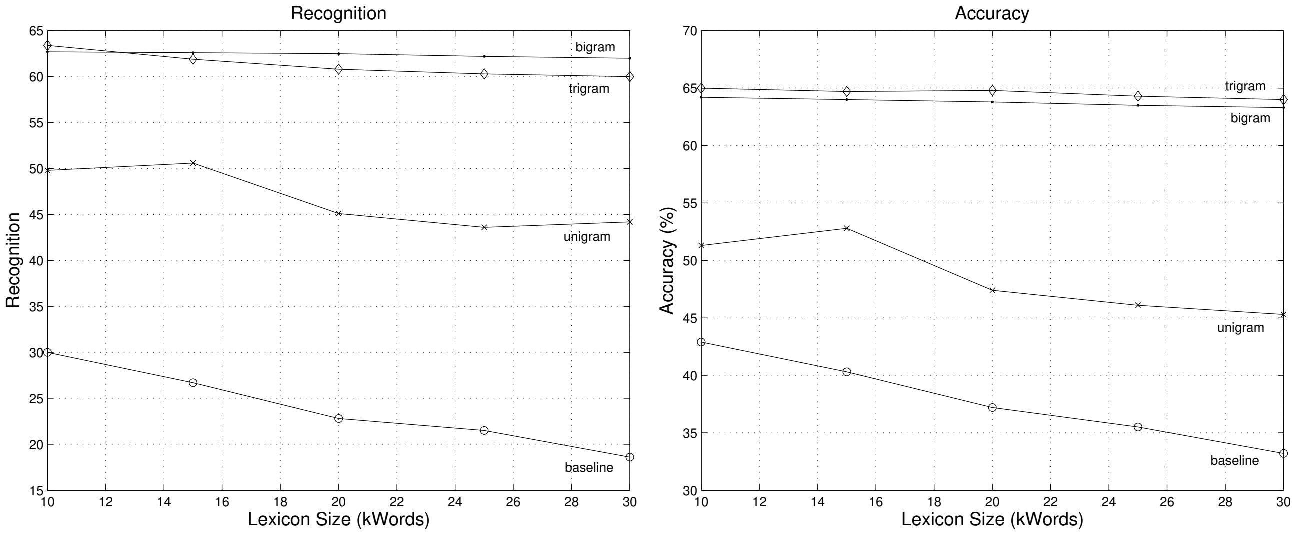


Fig. 7. System performance over Reuters database. The plot on the left (right) shows the recognition (accuracy) of the system. The performance is measured over the test set for the four systems considered: baseline (no SLM), unigrams, bigrams and trigrams.