

N-Gram and N-Class Models for On line Handwriting Recognition

Freddy Perraud * — Christian Viard-Gaudin** — Emmanuel Morin *** — Pierre-Michel Lallican *

* *Vision Objects - 9,rue du Pavillon, 44980 Sainte Luce sur Loire*

{*freddy.perraud, pmlallican*}@*visionobjects.com*

** *Institut de Recherche en Communications et Cybernétique de Nantes - UMR CNRS 6597*

christian.viard-gaudin@polytech.univ-nantes.fr

*** *Institut de Recherche en Informatique de Nantes*

morin@irin.univ-nantes.fr

Abstract

This paper highlights the interest of a language model in increasing the performances of on-line handwriting recognition systems. Models based on statistical approaches, trained on written corpora, have been investigated. Two kinds of models have been studied: *n*-gram models and *n*-class models. In the latter case, the classes result either from a syntactic criteria or a contextual criteria. In order to integrate it into small capacity systems (mobile device), an *n*-class model has been designed by combining these criteria. It outperforms bulkier models based on *n*-gram. Integration into an on-line handwriting recognition system demonstrates a substantial performance improvement due to the language model.

1. Introduction

Recently, new devices such as Tablet PCs and digital pens have emerged. They allow to write large handwritten text, and not only short expressions as with PDA where the user only has a very small input window. Consequently, the handwritten recognition algorithms used to convert the handwriting into electronic text should be more complex as those developed until now. Most of the time, they were working either at the character or at the word level only. We propose here to introduce explicitly a language model to guide the output of the word recogniser.

Figure 1 describes the global structure of the proposed approach. If the duration of words were fixed *a priori*, sentence recognition could be done in much the same way as word recognition by assigning class labels independently to each fixed length handwritten text segment. However, the time-boundaries between words are not known *a priori* during recognition and it is necessary somehow to align hypothesized word sequences to the observed handwritten signal, i.e., to search for those segments of the handwritten signal that in some sense optimally represent the hypothesized words.

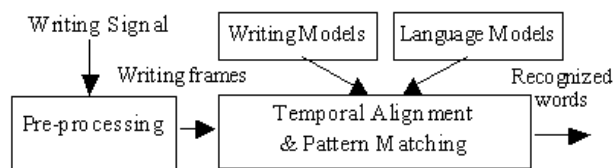


Figure 1. Structure of the on-line handwriting recognition system

When the recognition system works at the character level, the language model is restricted to the alphabet symbol set. When it is extended to the word level, it has to incorporate the lexicon content, and, at last, in order to take into account the structure of a sentence, a more complex language model has to be defined. It should be able to predict the sequence of words in a sentence. This is the purpose of this work. We want to study how to model natural language in order to take advantage of this model for on-line recognition. An important additional constraint is that the proposed model should be suited for real time and low memory systems.

More formally, the ultimate goal is to determine the most likely word sequence given the signal, a language model and handwriting models. These various elements, namely the writing signal x , the sentence to be recognized s , the language model and the handwriting models being linked by the well known Bayes relation:

$$p(s|x) = \frac{p(x|s)p(s)}{p(x)} \quad (1)$$

Within this relation, the language model accounts for the *a priori* probability $p(s)$ for a given sentence s . Whereas handwriting models compute the likelihood of s , given the observed signal x . Consequently, the result of the recognition system will consist in maximizing the *a posteriori* probability $p(s|x)$. Therefore, by using the MAP (Maximum A Posteriori) criteria, the selected sentence will be:

$$\hat{s} = \arg \max_s \{p(x|s)p(s)\} \quad (2)$$

Formula 2 requires to compute $p(s)$ for every possible sentence. It is the main challenge for the language model

to provide reliable estimates for the probability of every possible sentence, the number of possible sentences for unconstrained text being enormous. The proposed approach learns the model parameters from large written text corpora, and includes smoothing algorithms to deal with unseen sentences.

Secondly, the computation of $p(x|s)$ has to be done on-line as signal x is produced. So, it is mandatory to restrain the set of possible sentences to only a small subset. To achieve that, we use dynamic programming techniques based on Viterbi and Beam Search algorithms.

In the remaining part of the paper, we will focus on the developed language models since the recognition is presented in details in [1].

2. Stochastic language modeling

It is possible to describe a sentence, s , by a sequence of L words $w_i : s = w_1 \dots w_i \dots w_L = w_{1,L}$. A specific sentence being interpreted as a realization of a discrete stochastic process. Moreover, if it is assumed that the evolution of the process – the next word – depends only on its present state – the already written text –, the general framework of Markov processes is applicable. In addition, if the set of states is discrete, here it is defined by the lexicon, then the system can be modeled by a Markov chain [2].

With this assumption, and knowing the state transition probabilities, it is possible to compute the likelihood of any specific chain, i.e. of any specific sentence. The result is obtained by applying the Chapman-Kolmogorov formula:

$$p(s) = p(w_{1,L}) = p(w_1)p(w_2|w_1)p(w_3|w_1 w_2) \dots p(w_L|w_1 \dots w_{L-1})$$

$$= \prod_{i=1}^L p(w_i|w_1 \dots w_{i-1}) \quad (3)$$

In other words, for every position in the sentence $w_1 \dots w_i \dots w_L$, the language model is able to compute a probability value $p(w_i|w_1 \dots w_{i-1})$ for every possible next word w_i , satisfying the normalization constraint:

$$\sum_{j=1}^V p(w_i = w^j | w_1 \dots w_{i-1}) = 1 \quad (4)$$

where w^j stands for the j^{th} word of the lexicon.

Of course, the longer the past word sequence, the less reliable the estimate of the conditional probability value since such a sequence is not likely to have been encountered during learning. To overcome this limitation, a reduction in the order of the Markov chain is used. It reduces the influence of the past words to only the $n-1$ previous words. This allows to define the language model by using the so-called n -gram probabilities:

$$p(w_i | w_{i-n+1} \dots w_{i-1}) \approx p(w_i | w_1 \dots w_{i-1}) \quad (5)$$

By replacing in equation (3), we get:

$$p(s) = p(w_{1,L}) = \prod_{i=1}^L p(w_i | w_{i-n+1} \dots w_{i-1}) \quad (6)$$

The computation of the model parameters requires very large learning corpora. With such corpora, it is possible with a basic counting approach to estimate these n -gram probabilities:

$$p(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{c(w_{i-n+1} \dots w_i)}{c(w_{i-n+1} \dots w_{i-1})} \quad (7)$$

where $c()$ stands for the number of corresponding events.

From a practical point of view, it is essential to incorporate smoothing techniques in order to have more robust estimates and to deal with unseen n -grams. After many experiences, which can not be reported here because of the lack of space, we have selected the absolute discounting backing-off algorithm [3].

It is essential to be able to evaluate the performance of a given language model. It allows to optimize the setting of the parameters of the model and to compare different models. Two main criteria have been considered here. The first one is the precision of the model, it will be evaluated by the measurement of the perplexity. The second one is the memory requirement of the model.

The perplexity measurement determines the average precision of the model [4]. It corresponds to the inverse of the geometrical mean of the *a priori* probabilities $p(w_i|w_{i-n+1} \dots w_{i-1})$ computed on a test set, T_{test} , different from the training set. The smaller the perplexity, the better is the model.

$$PP_M(T_{\text{test}}) = \left[\prod_{i=1}^L p(w_i | w_{i-n+1} \dots w_{i-1}) \right]^{-\frac{1}{L}} \quad (8)$$

One severe limitation of the n -gram models lies in the size of the models. Indeed, the total number of parameters for these models is basically V^n , V being the size of the lexicon. For large size vocabulary, i.e. several thousands of words, even with $n = 2$, there are already too many parameters to store in a mobile device. To overcome this problem, words will be clustered in classes, and the probability for the next word will be estimated based on the classes of the previous words. The issues raised by this approach are discussed in the next section.

3. N-class Model

The main issue with n -class models is to define a criterion for the clustering process. In this work, we have compared two different clustering approaches [5]:

- Contextual criteria: words which share the same lexical context will be grouped into the same class. For example, the days of the week, the colors, ... are very likely to be in the same cluster, since most of the time, in the corpora they will appear in the same context. The

corresponding models will be termed “statistical n-class models”.

- Grammatical criteria: in that case, classes will be built from the grammatical nature of the words as they are defined with labels corresponding to the so-called Part Of Speech (POS). For example, the label “DTN:m:s” corresponds to Determinant/masculine singular; another label “PREP” is for preposition ... These labels are automatically set on the training corpora using the Brill tagger [6]. The corresponding models will be called “syntactical n-class models”.

3.1. Statistical n-class models

The objective with statistical classification is to gather in the same class words which share the same lexical context. For instance, consider the next expressions: “Every Sunday morning”, “Every Tuesday morning”, ... and let us define a class <day> = {Sunday, Monday, ..., Saturday}. Then consider the three events:

$a: (w_{i-1}=Every); b:(g(w_i)=day); c:(w_i=Tuesday);$

From the Bayes relation, we have:

$$p(c|a) = p(a|c).p(c)/p(a) \text{ and } p(c|b) = p(b|c).p(c)/p(b) \quad (9)$$

but $p(b|c) = 1$ (Tuesday is actually a day), consequently :

$$p(c|a) = p(a|c).p(c|b).p(b)/p(a) \quad (10)$$

This relation can be written:

$$p(c|a) = p(a|c).p(c|b).p(b|a)/p(a|b) \quad (11)$$

If we assume that $p(a|b) \approx p(a|c)$, i.e. $p(Every|<day>) \approx p(Every|Tuesday)$ which means that we have the same frequency for the events “Every Sunday”, “Every Tuesday”, ..., then we get:

$$p(c|a) \approx p(c|b).p(b|a) \quad (12)$$

Which means here:

$$p(Tuesday|Every) \approx p(Tuesday|<day>).p(<day>|Every) \quad (13)$$

The number of parameters required to compute the right-hand side of formula 13 is only $V+K \times V$ compared to V^2 for the left-hand side, V being the size of the lexicon, K the number of classes.

In fact, this concept is generalized without semantic meaning associated to the classes. They are automatically built during a learning stage based on the similarity of the context.

Every word w_i is associated to a class $g(w_i) = g_k$, with $k \in [1..K]$. The conditional probability $p(w_i|w_{i-n+1}...w_{i-1})$ is then given by:

$$p(w_i|w_{i-n+1}...w_{i-1}) = p(w_i|g(w_i)) \cdot p(g(w_i)|g(w_{i-n+1})...g(w_{i-1})) \quad (14)$$

For a bi-class model, it remains only:

$$p(w_i|w_{i-1}) = p(w_i|g(w_i)) \cdot p(g(w_i)|g(w_{i-1})) \quad (15)$$

In that case, only $V+K^2$ parameters are required to define the language model. To define the $g()$ function which assigns a class to a word we have used an iterative

procedure based on the Expectation-Maximization algorithm which minimizes as a cost function the global perplexity on the learning database [7].

3.2. Syntactical n-class models

The syntactic clustering is based on the Parts of Speech of the language (POS). Model parameters are easily computed from a learning database which should be tagged with the POS labels, the number of classes being the number of POS.

Contrary to the preceding clustering method, in the case of syntactic clustering, a given word can share several POS. For instance the word “can” is possibly a verb, a name, ... Consequently, for a given word sequence, several different class sequences are possible. Figure 2 displays the oriented graph which defines all the possible class sequences for the sentence “La souris court vers le fromage” (The mouse is running towards the cheese).

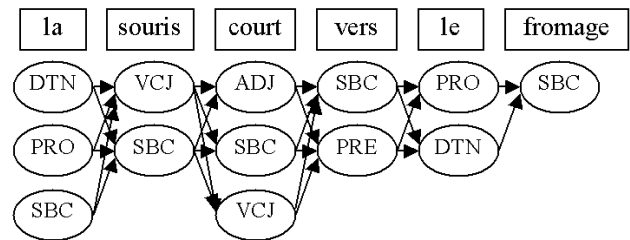


Figure 2. POS Sequence Graph for the sentence « la souris court vers le fromage »

For every word, the set of possible POS is defined by a morpho-syntactic lexicon. The likelihood of a path, is computed in the same way as for the statistical classes, using formula (14). Since there are several different valid paths, the overall likelihood is summed over all valid paths. This computation is carried out in an efficient way using the forward-backward algorithm.

4. Language model performances

In this section, we first, introduce the text corpora which are used in the experiments, then several experiments are described to study the properties of the language models that we have described previously.

Table 1. Training databases

Name	Size	Lexicon size	Text domain
ABU	4.1 M words	90 K words	literature
ECI	4.2 M words	100 K words	journalistic
TEST	1.6 M words	64 K Words	mixed

Within the framework of our experiments, two distinct corpora, namely ABU and ECI have been used to train the language models. The main features of these databases are described in table 1. Both databases have about 4 millions words, they cover different fields of texts: ABU is composed of novels from XIX and XXth centuries, ECI is composed of articles from the French newspaper “Le Monde”. A distinct database, mainly issued from newspapers (TEST) is used for test.

To improve readability, some shorthand notations are introduced : BGM : BiGram Model ; BCSyntM-X : BiClass Syntactic Model using label set X, with X={Short, Long} ; BCStatM-X : BiClass Statistical Model with X classes, X={10, 50, 100, 500, 1000}

4.1. Bi-gram model

First experiments have been conducted on the bi-gram model. The results obtained can be considered as target results for smaller bi-class models. For the considered applications -small mobile devices- such a language model cannot be a satisfying solution since its memory requirement is by far too big. Perplexity values obtained with this model (BGM) on the test set with two different training sets are displayed in Figure 3.

4.2. Statistical bi-class models

For the statistical models, the important parameter is the number of classes. A small value defines a very compact model whereas a bigger one allows a better modeling of the language. Nevertheless, even if the memory limitation was not a constraint, a very large value for the number of classes will require an enormous learning database otherwise the generalization capability of the model will be poor (empirical versus structural risk minimization). This behavior is confirmed by the curves presented in Figure 3. Up to 500 classes the perplexity values decrease, i.e. the model behaves better, then no more improvement is obtained for 1.000 classes. It is worth to notice that with 500 classes, the results are very close to those obtained with the bi-gram model.

4.3. Syntactical bi-class models

With syntactical models, the number of classes is ruled by the number of POS of the morpho-syntactic lexicon. POS, which are established by the Brill tagger, are defined by a set of attributes which allow several possible groupings. We have defined two different sets: one set with 210 classes and a smaller second one, with only 57 classes, which correspond to the main syntactic labels. They are respectively referred as BCSyntM-Long or BCSyntM-Short in Figure 3. Not surprisingly, the bigger model outperforms the smaller one. With respect to

statistical models, syntactical models are a little bit less powerful with about the same number of classes.

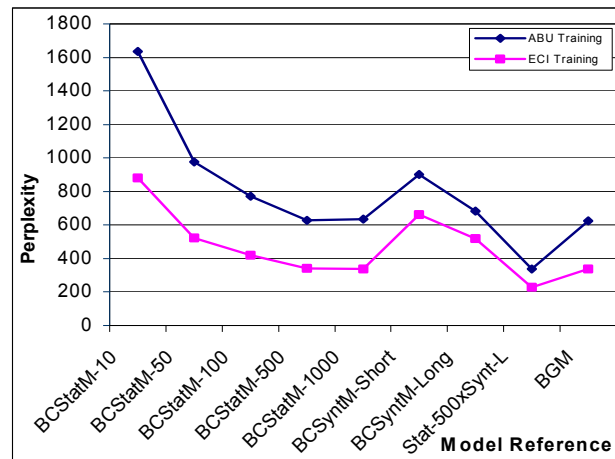


Figure 3. Model performances on the test set

4.4. Syntactical and statistical models combination

It is possible to combine several language models by simply weighting their contribution :

$$P_{combined}(w_i|w_{i-n+1} \dots w_{i-1}) = \sum_{m=1}^M \lambda_m \cdot P_m(w_i|w_{i-n+1} \dots w_{i-1}) \quad (16)$$

We have experimented such a combination with the statistical and the syntactical models (M = 2) to test the complementarities of these two models. Actually, it allows to increase the precision of the model since the perplexity reaches its minimum value as it is displayed in Figure 3, and it outperforms the bigram model.

5. Language and handwriting models combination

The proposed recognition system is based on a hybrid solution combining a neural network, which computes the character posterior probabilities, and a dynamic programming scheme which allows to handle several different segmentation hypotheses. For each hypothesis a likelihood recognition score is available (N-best Viterbi algorithm).

Two different combination techniques for the handwriting model and the language model are possible. One is to integrate the language model *a priori* in the search algorithm for the best path in the trellis of solutions. The other one, that we have implemented here, is to first deal with the handwriting model alone and then, *a posteriori*, to reorder the N-best solutions by taking into account the output of the language model (Equ. 6). In that case, the probability product of both models defines the overall likelihood function of a sentence.

To evaluate the benefits of the language model, a database of 5,000 different sentences corresponding to 37,710 words for a 8,800 words lexicon is used. About 400 writers have participated to the database. It is labeled only at the sentence level, therefore to access to the word recognition rate an edit distance (Levenstein distance) is computed between the true label sentence and the recognized one.

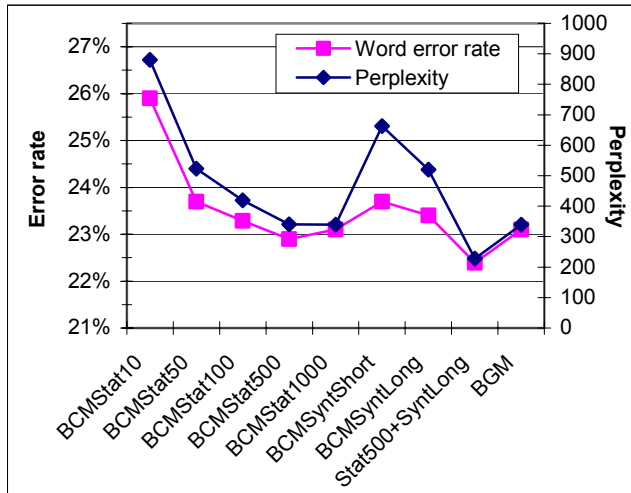


Figure 4. Language model performances on the handwriting test set

On this basis, the word error rate without any language model is 34 %. This relatively large value can be explained by additional difficulties compared to just word recognition systems. In addition to the errors made at the word recognition level, in the case of sentence processing, errors due to inter-word segmentation and punctuation recognition also affect the recognition rate.

When a basic language model consisting of only the monogram probabilities is incorporated in the recognition trellis, then the recognition rate dropped to 29 %. Then, if we combined a bi-class language model with an *a posteriori* product combination, we get the results displayed in Figure 4.

The decrease in the word rate is consistent with the results obtained with the model perplexity measurement. For a statistical model, the number of 500 classes appears to be the best trade-off between bias and variance of the model, it allows to decrease the error rate down to 23 % which is already better than the error rate corresponding to a full bigram model (right column). As we can see on the before last column, by combining this model with the BC Syntactic Model, it is possible to improve a little bit the recognition rate.

Figure 5 shows some examples which are corrected by the language model since the initial top recognition sentences were not correct. The first sentence was recognized by the handwriting recognition model as

« Mais jamais pour *tirs* longtemps. » and the second one « Ce n'est pas si *dû*. » In both cases, as well the BCStatM-500 and the BCSyntM-short are able to correct the sentence so that the true label is actually recognized.

Mais jamais pour très longtemps.
Ce n'est pas si sûr.

Figure 5. True labels « Mais jamais pour très longtemps. » and « Ce n'est pas si sûr. »

6. Conclusion

The language models that have been experimented in this work show that they allow to increase the performance of handwritten recognition systems. On a particularly difficult test database, the word error rate has been decreased from 34 % down to 22.5 %. To achieve that, a bi-class model has been defined which combines up to 500 statistical classes, and syntactical classes. With the experiments that have been conducted we have demonstrated that the perplexity measure is a good performance indicator since it is highly correlated to the error rate. Such bi-class models perform better than bigram models despite very little memory requirements. An extension to tri-class model is currently under investigation. It further improves the overall recognition rate with little computational.

7. References

- [1] Tay Y.H., Lallican P.M., Khalid M., Viard-Gaudin C., and Knerr S. "An Analytical Handwritten Word Recognition System with Word-level Discriminant Training". 6th ICDAR, Seattle, pp. 726-730, 2001.
- [2] Manning C., Scutze. H. *Foundation of Statistical Natural Language Processing*, The MIT Press, 2000.
- [3] Goodman J., Stanley F. Chen, "A empirical study of smoothing technique for language modelling". Vol. TR-10-98. 1998.
- [4] Marti U.-V. and Bunke H., "Unconstrained Handwriting Recognition: Language Models, Perplexity, and System Performance", IWFHR VII, Amsterdam, 2000.
- [5] Niesler T., "Category Based Statistical Language Model's", Ph. D. thesis, University of Cambridge, 1997.
- [6] Brill E., *Some Advances in Rule-Based Part of Speech Tagging*. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 94), pp 722-727, 1994
- [7] Martin S., Ney H., Liermann J. *Algorithms for bigram and trigram word clustering*, Speech Communication, pp. 19-37, 1998.