Electronic Letters on Computer Vision and Image Analysis 5(2):87-97, 2005

Self-supervised adaptation for on-line script text recognition

Lionel Prevost Loïc Oudot

Universit Pierre et Marie Curie
LISIF / PARC BC 252
4 Place Jussieu, 75252 Paris Cedex 05, France

Received 16 July 2004; accepted 28 July 2005

Abstract

We have recently developed in our lab a text recognizer for on-line texts written on a touch-terminal. We present in this paper several strategies to adapt this recognizer in a self-supervised way to a given writer and compare them to the supervised adaptation scheme. The baseline system is based on the activation-verification cognitive model. We have designed this recognizer to be writer-independent but it may be adapted to be writer-dependent in order to increase the recognition speed and rate. The classification expert can be iteratively modified in order to learn the particularities of a writer. The best self-supervised adaptation strategy is called prototype dynamic management and gets good results, close to those of the supervised methods. The combination of supervised and self-supervised strategies increases accuracy again. Results, presented on a large database of 90 texts (5,400 words) written by 38 different writers are very encouraging with an error rate lower than 10 %.

Key Words: handwriting recognition, supervised adaptation, self-supervised adaptation, model-based classifier.

1 Introduction

Recently, handheld devices like PDAs, mobiles phones, e-books or tablet PC have became very popular. In opposition to classical personal computers, they are small, keyboard-less and mouse-less. Therefore, electronic pen is very attractive as pointing and handwriting device. Such a device is at the frontier of two research fields: man-machine interface and handwriting recognition.

In this paper, we focus on the problem of handwriting recognition for handheld devices with large screen on which we can write texts. For such an application, recognition rate should be very high otherwise it should discourage all the possible users. With the last handwriting recognizers on the market (Microsoft Windows XP Tablet Edition, Apple Ink, myScript. . . ,) the recognition rate has became acceptable but is not high enough. The major problem for these recognizers is the vast variation in personal writing style. Updating the parameters of a writer-independent recognizer to transform it into a writer-dependent recognizer with a higher accuracy can solve this difficulty. The systems listed above are not able to adapt themselves to a given writer. We can get better recognition rates if we adapt a writer-independent recognizer with an adequate architecture and

Correspondence to: <lionel.prevost@lis.jussieu.fr>

Recommended for acceptance by <J.M. Ogier, T. Paquet, G.Sanchez <>
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

transform it quickly in a writer-dependent system. However, it should not be forgotten that the use of a pen as input modality has to be user friendly. So, the training step must be as shorter as possible or - better - totally hidden for the user.

Traditional adaptation technics require the writer intervention (the so-called supervised adaptation). We propose in this article several self-supervised adaptation scheme that we compare to the already existing techniques like supervised adaptation.

The article is organized as follows. In section 2, we present a review of the various techniques of adaptation. In section 3, we describe the writer-independent baseline system. In section 4, we describe the different adaptation strategies. In section 5, we present a combination between self-supervised and supervised methods to achieve very good results. Finally, conclusions and prospects are given in section 6.

2 Literature review

The idea of writer adaptation was revealed by researches in the field of perceptive psychology. It has been shown that, in the case of a hardly readable writer, it is easier to read a word if we have already read other words written by the same person. This phenomenon is called the graphemic priming effect [13]. Thus, we learn the user writing characteristics from the words we can read, and then, we use this new knowledge to read the remaining words.

In the literature, we consider two adaptation strategies: systems where the adaptation step takes place once first before use (called off-line) and systems with continuous adaptation (on-line).

Most systems [5, 12, 1, 2] using an off-line adaptation scheme need a labeled database of the writer. These examples are use to make a supervised training of the system. Thus, the system learns the characteristics of this particular writer before being used.

On the other hand, the following systems evolve continuously during use.

The on-line handwriting recognition and adaptation system of [10] uses a supervised incremental adaptation strategy. The baseline system uses a single MLP with 72 outputs (62 letters and 10 punctuation marks). An adaptation module, at the output of the MLP modifies its output vector. This adaptation module is a RBF (*Radial Basis Function*) network. The user informs the system of the classification error, giving the letter label, and the RBF is re-trained (modification of the existing kernels or addition of a new one).

Two other systems use a TDNN (*Time Delay Neural Network*) as classifier instead of the MLP. This TDNN is trained on an omni-writer database and the output layer of this network is replaced either by a k-nn classifier in [3] or by a discriminating classifier in [6]. During the adaptation step, the TDNN is fixed and the output classifier is trained, in order to learn mis-recognized characters.

The system described in [15] is very close to our system but is dedicated to isolated alphanumeric character recognition. The k-nn classifier uses the Dynamic Time Warping algorithm to compare the unknown characters to a prototype database. The writer adaptation consists in adding the mis-classified characters in this database. Moreover, useless prototypes can be removed from the database to avoid an excessive growth of this latter.

There are also a lot of works on adaptation in off-line character recognition and other pattern recognition fields including speech recognition [16]. For example, in [14], the authors adapt the Hidden Markov Models (HMM) first trained on a large database with a small database of the particular writer.

Based on the results of all these studies, we can notice that model-based classifier (MBC) like k-nn have better ability to learn particular patterns than machine learning classifier (MLC) like HMM, MLP or GMM (Gaussian Mixture Model). MBC need very few samples to learn a new pattern (sometime one sample is enough) and, as this learning consists in adding the new sample in the classifier database, they are not time consuming. But the database size tends to increase significantly, so the classification time and the memory needed, increase linearly with this size. On the other hand, MLC need more samples and are time consuming to re-estimate their parameters. But after the training, the size and the classification time remain the same.

3 Writer independent baseline system

For the experiments, we collected a large text database written by 38 different writers. Each writer wrote an average of 150 words for a total of 5,400 words and 26,000 letters. A human expert labeled all the texts. We present in this paper some iterative adaptation strategies: the performances of the system improve continuously with the amount of data. Thus, we will study the evolution of the recognition rate on three ranges corresponding respectively to 50, 100 and 150 words used for the adaptation. Some other writers who have written less than 50 words are kept to constitute the text training database for the tuning of the writer independent system [8].

We use for adaptation a lexicon containing the 8,000 most frequent words of the French language. Our system is also able to handle very large lexicons (some 200,000 words) as shown in the following. The complete analysis speed is about 6 words per second (P4 1,8GHz Matlab) and a small amount of memory is required (about 500Ko including the system program, the 8K lexicon and the database).

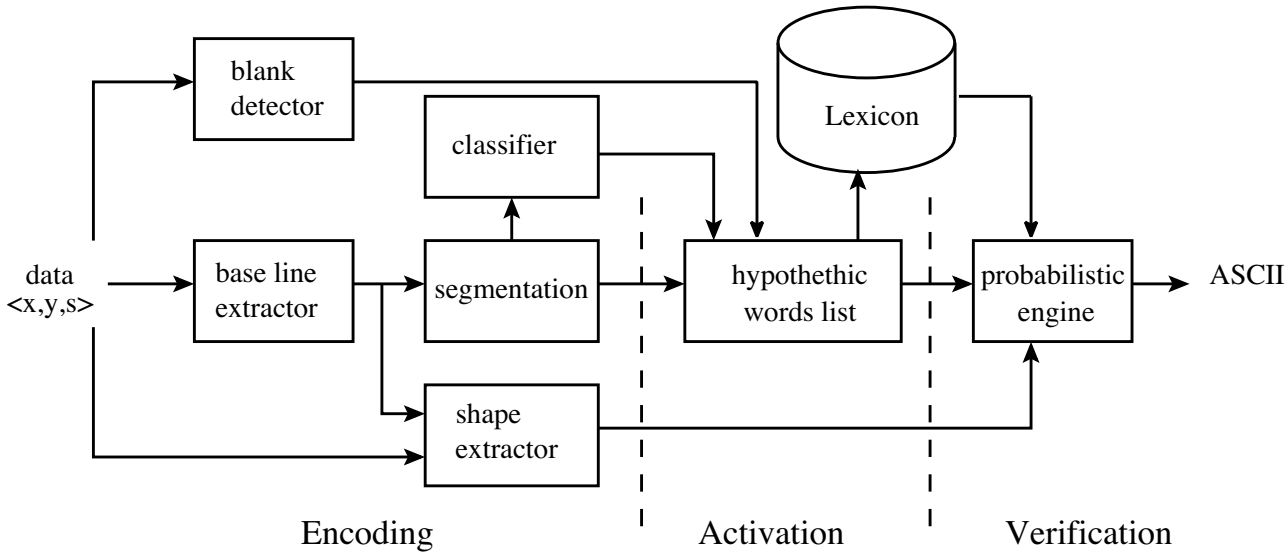

Figure 1: Baseline system.

The writer independent baseline system is presented in figure 1. It is based on the activation-verification cognitive model described by Paap in 1982 [9]. The system consists of a set of three neural encoding experts [8] that extract geometrical and morphological informations in the input data (*i.e.* strokes)

The first expert gives informations about the **shape** of the strokes (size of ascender and descender...). We compare the bounding box of the stroke with the estimated height and positioning of medium letters in the line.

The second expert gives us **segmentation** informations like between-letter, within-letter and within-word separation between two consecutive strokes. The input of the neural network is a 32 features vector composed of absolute and relative measurement of the two strokes. We use a *forward backward sequential selection* (FBSS algorithm described in [7]) to keep the most relevant features.

The last expert is the **character classifier**. It is a k-nn classifier and it uses an omni-writer prototype database. This database was created by using an automatic clustering algorithm [11] starting from the 60,000 samples of UNIPEN database [4] (corpus Train-R01/V07). This algorithm is well fitted to heterogeneous character classes with highly variable densities. It overcomes the classical problems of clustering (prototype optimal number, initialization ...). It works on labeled examples of a given class and try to optimize the within-class variance by combining two stages: a sub-optimal unsupervised research of prototypes followed by an adaptation stage using vector quantization. After clustering, the prototype database contains some 3,000 stroke prototypes for the 62 classes (26 upper-case letters, 26 lower-case letters and 10 digits). Each sample represents a given character allograph (for single-stroke characters) or a part of the allograph (for multi-stroke characters). An allograph

is a specific handwriting feature. It includes on the one hand characters with the same static representation (i.e. the same image) but written with variable dynamics (number of strokes, senses, direction ...) and on the other hand, the different handwriting model for a given character : cursive, hand-printed, mixed ... When an unknown character has to be classified, it is first divided into strokes. Then, each stroke is compared with a prototype subset producing a distance vector. The distance of the unknown data to each character class is the sum of all the distance vectors (over the number of strokes). The nearest-neighbor criterion is then applied to find the winning class.

All these experts provide probabilistic information at the stroke level. For each expert, we also compute a confusion matrix on the training data, in order to evaluate prior probabilities. We use the Bayesian rule to re-estimate posterior probabilities by combining this latter with prior knowledge. The segmentation probabilities are used to construct the smallest and most relevant segmentation tree of a line of text. The classifier probabilities are used to activate a list of hypothetical words in the lexicon for each segmentation in the tree. A probabilistic engine that combines all the available probabilities evaluates the likelihood of each hypothetic word in this list. We call this information the *probability of lexical reliability* (PLR). We used dynamic programming in the segmentation tree where each node has a PLR in order to get the best re-transcription of the line.

We evaluate this lexicon driven recognizer on differently lexicon size on the whole text database used for adaptation (figure 2, graph Omni). We also add some allographes from the text database into the classifier prototype database to turn the system into a multi-writer recognizer (figure 2, graph Multi). Even if the recognition rate is not so high, we can notice the very good ability to manipulate very big lexicon. We loose less than 5 % of the recognition rate when we use a 187,000 words lexicon comparing with a 400 words lexicon (4675 times smaller). Finally, we achieve a word error rate of 25 % in a writer-independent frame with a 8,0000 words lexicon.

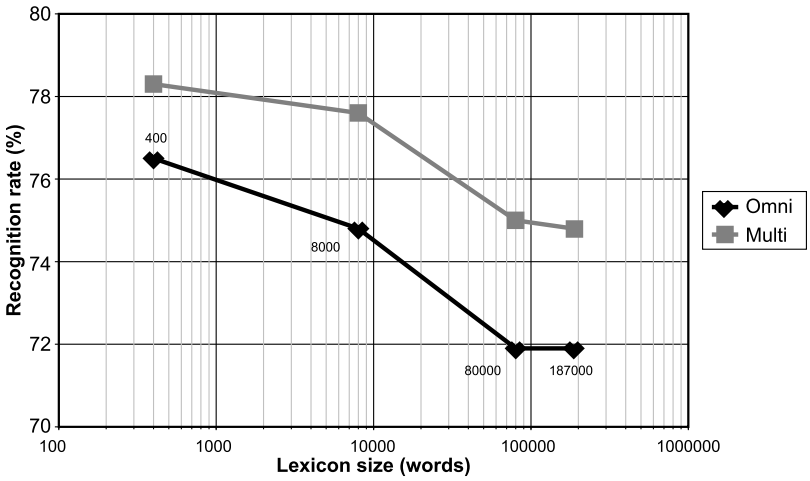

Figure 2: Recognition rate vs lexicon size.

4 Writer adaptation strategies

The baseline system recognition is writer-independent. Its prototype dataset (the so-called WI database) should cover all the writing styles. Each prototype corresponds to a particular shape of a whole letter (*i.e.* allograph). Experimental results show that it covers at least the most common writing styles. We also remark that storing character samples taken from the text database in the prototypes database (multi-writer system) improves greatly the recognition rate. There are, at least, two situations that reduce the recognition rate.

- Missing allograph: the allograph is missing in the prototype database and it must be stored (added) in this set.
- Confusing allograph: for a given writer, the prototype is confusing or erroneous and it must be removed from the prototype database.

Model-based classifier can be adapted very easily and quickly to new writing styles, just by storing new character samples in the writer dependent (WD) database (when these latter miss) and, if needed, by inactivating existing prototypes (when they are confusing). The system specialization on a given user – by registration of his personal features – makes it writer-dependent and increases its accuracy. The comparison of classification hypothesis with either the labeled data (supervised adaptation) or the lexical hypothesis (self-supervised adaptation) detects classification errors. The misclassified characters can be stored in the writer-dependent (WD) database, using the lexical hypothesis as a label.

4.1 Supervised adaptation

Before comparing the accuracy of self-supervised adaptation strategies, we start by studying supervised technics. We use the labels of the text database and the real text segmentation to carry out supervised adaptation. Note that when we know the text segmentation, our writer-independent recognizer does not have to build a segmentation tree and so the word error rate is about 5 %. The supervised adaptation acts as follow. Characters of the text are classified one after the other. The classification hypothesis (the best answer, top_1, of the character classifier) is compared with the label. If they do not match, the mis-recognized character is stored in the user personal database (figure 3). We consider two approaches: the *text* approach where the characters are added at the end of the analysis of the text and the *line* approach where the characters are added at the end of the analysis of each line. The results (table 1) show the improvement of the recognition rate due to the writer adaptation of the handwriting recognition system when the segmentation of the text in words and letters is known. We present the word error rate (WER) after 50, 100, and 150 analyzed words.

Figure 3: Supervised addition of prototypes in the user database.

As we know the labels and the text segmentation (it is not realistic just an interesting case study), we achieve an awesome word recognition rate of 99 % that proves the necessity of applying adaptation strategies to recognition systems. The WDDBS show the amount of prototypes added in the WD database regarding to the WI database size. The *line* approach allows a faster improvement of the recognition rate and adds fewer prototypes to the user database than the *text* approach. When we add characters after a full text analysis, we can add several similar prototypes (and the average number of added prototypes increases). On the other hand, the *line* approach, adds the first prototype of a mis-recognized character. Thanks to this new sample, the following

	WER			WDDBS
Words	50	100	150	
Baseline system	5 %			100 %
Text appr.: min	0 %	0 %	0 %	+3 %
mean	1.3 %	1.1 %	0.6 %	+6 %
max	10 %	5.1 %	4.5 %	+9 %
Line Appr.: min	0 %	0 %	0 %	+2 %
mean	**1.1 %**	**0.7 %**	**0.4 %**	**+4 %**
max	6.2 %	5.2 %	3.7 %	+8 %

Table 1: Supervised adaptation: Word error rate WER and WD database size WDDBS (known segmentation). *min* is the result on the best writer, *max* is the result on the worst writer and *mean* is the result on the overall text database (8k lexicon).

similar characters are correctly classified, so they do not need to be stored in the prototypes database. So, the number of added prototypes is smaller in the *line* approach than in the *text* approach and we select the first strategy for the following works. Due to the architecture of the recognition system, it is not possible to study a *word* approach, where we made the adaptation after each analyzed words. It seems logical to think that a *word* approach should perform better than the *line* approach but the difference should not be enough to change completely the results obtained with the *line* approach.

From a perceptive point of view, the prototype storing imitates – at the letter level – the priming repetition effect noticed at the word level: the initial presentation of a word reduces the amount of information necessary to its future identification and this identification is performed faster. Nevertheless, activating WD prototypes is not sufficient to perform perfect classification, even with a great amount of labeled data. Some added characters will generate mis-classification and new errors will appear. It seems necessary to inactivate – or even delete – some WI prototypes.

4.2 Self-supervised adaptation

In self-supervised adaptation, we use the recognizer in a real framework, *i.e.* the data labels and the text segmentation are not known (our reference system achieve a word error rate of 25 % on a 8,000 words lexicon, see figure 2). Moreover, self-supervised adaptation must be completely hidden to the writer which should not be solicited by the system. Now, the classifier hypothesis and the lexical hypothesis are compared to find which prototypes must be stored in the user database.

Figure 4: Self-Supervised adaptation method. Addition of prototypes in the user database.

4.2.1 Systematical activation (SA)

In the systematical activation strategy, we consider that the lexical analyzer is "perfect". Therefore, when an error (difference between the classification hypothesis and the lexical hypothesis) occurs, the corresponding character is stored in the user personal database. Due to the lexical analyzer errors cumulated with the segmentation errors, some prototypes are stored in bad classes (figure 4). These errors introduce many new classification errors. The performances of the recognition system after adaptation is just a little bit better than those of the baseline system (table 2).

4.2.2 Conditional activation (CA)

As the previous strategy is not really accurate, it seems necessary to study the behavior of the lexical analyzer in order to store only useful prototypes. We saw that the recognition engine estimates for each word a *probability of lexical reliability (PLR, section 3)*. This PLR reflects the probability of error of the lexical analyzer for this word. The conditional activation strategy is described in the following. If, for a given word, the PLR is greater than α (*i.e.* we have good confidence in this word), then the mis-classified characters of this word are added to the user database. We determined the α parameter on the text training database by minimizing the Bayesian error between the PLR distributions of well-corrected words and words which were not well corrected by lexical analysis. We obtained an α of 0.015 and we show in table 2 the result of the conditional activation.

	WER			WDDBS
Words	50	100	150	
Baseline system	25 %			100 %
SA strategy: min	0 %	1.9 %	2 %	+2 %
mean	25 %	23 %	23 %	+6 %
max	53 %	73 %	51 %	+14 %
CA strategy: min	0 %	0 %	2 %	+1 %
mean	**22 %**	**20 %**	**17 %**	**+2 %**
max	71 %	58 %	43 %	+3 %

Table 2: Systematic and conditional activation: Word error rate WER and WD database size WDDBS (8k lexicon).

The CA strategy is more accurate than the SA strategy as it reduces considerably the false additions of prototypes (see the small growth of the user database). Moreover, with the CA strategy the error rate decreases continuously over the time. After 150 words of adaptation, the error rate decreases of about 8 %.

4.2.3 Dynamic management (DM)

This method has two goals. As seen previously, using lexical hypothesis as a reference may add confusing or erroneous prototypes, even when conditional activation is applied. Dynamic management is used to recover from those prototypes that contribute more often to incorrect than correct classifications. Inactivation methods are also used to prune the prototype set and speed-up the classification [15]. Each prototype (of the WI database as of the WD database) has an initial adequacy ($Q_0 = 1000$). This adequacy is modified during the recognition of the text according to the usefulness of the prototype in the classification process, by comparing the classification hypothesis and the lexical hypothesis. Let us consider the prototype i of the class j, three parameters are necessary for the dynamic management:

- **G** : Rewards (+) the prototype i when it performs **G**ood classification (classification and lexical hypotheses are the same).

- **M** : Penalizes (-) the prototype i when it performs **M**is-classification (classification and lexical hypotheses are different).
- **U** : Penalizes (-) for all the **U**seless prototypes of the class j.

The three parameters act differently. The U parameter is used to reduce the adequacy of the useless prototypes for a given writer. As the baseline recognizer is writer-independent, it needs many prototypes (an average of 40 prototypes per class) to model a character class but only a few ones will be useful for a given user. This parameter eliminates the prototypes that are not used during a long time. The value of U defines this life "time". The M parameter is used to penalize strongly erroneous prototypes. The value of this parameter must be bigger than the value of U because erroneous prototypes are much more troublesome than useless prototypes. By preserving only these two parameters, all the prototypes should disappear. Thus, it is necessary to reward good prototypes. To achieve it, the G parameter is used to increase the adequacy of any prototype activated during the classification and validated by the lexical analyzer. The equation (1) describes the evolution of the prototype adequacy. Where F_j is the frequency of the class j in the French language. These three parameters are mutually exclusive *i.e.* on each occurrence, only one parameter is activated. When $Q_j^i = 0$, the prototype is removed from the database. If these parameters are finely tuned, the system should inactivate quickly erroneous prototypes while preserving only the useful writer prototypes. After an exhaustive search of the parameters (G, M, U) the optimal triplet is (30, 200, 8) and does not depend of the lexicon size used for the lexical analysis. Moreover, we can change their values by ± 20 % without changing the results. A complete analysis of these three parameters can be found on [7].

$$Q_j^i(n+1) = Q_j^i(n) + [G(n) - M(n) - U(n)]/F_j \tag{1}$$

The dynamic management combined with the conditional activation strategy is very efficient as it greatly reduces the size of the database while preserving the recognition rate of the conditional activation strategy (table 3). Even with a very large lexicon of more than 187,000 words, this self-supervised adaptation technique is very accurate and allows us to increase the recognition rate of about 7 %.

	WER		WDDBS
	8k words	187k words	
Baseline system	25 %	28 %	100 %
DM strategy	**17 %**	**21 %**	**-80 %**

Table 3: Dynamic management: Word error rate WER and WD database size WDDBS after 150 adaptation words for two different lexicon sizes.

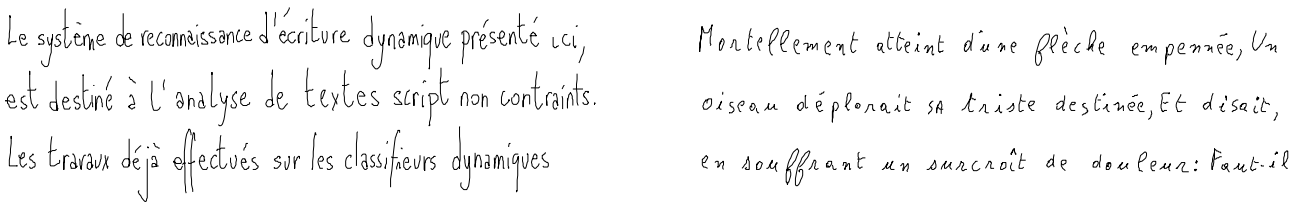

Figure 5: Best recognition rate writer (99 %) and worst writer (70 %).

Now, let us focus on the evolution of the adequacy of some prototypes (figure 6). For some writers, the WI prototypes are sufficient. For the class 'a', 2 prototypes are used and thus the adequacy of the 45 others decreases. For the class 's', 4 prototypes are useful (the writer has probably an unstable writing, see figure 5) and the 36 others are inactivated. For another writer (class 's' and 'e'), WD prototypes (in bold) are necessary. For the class 's', at the beginning, a WI prototype is used and after some 15 occurrences, a WD prototype is

added (the writer gets familiar with the handheld device and the pen). Another WD prototype is stored after some 35 occurrences (the user writes faster perhaps and changes his way of writing). After 150 adaptation words, the size of the prototype database was reduced by 80 %.

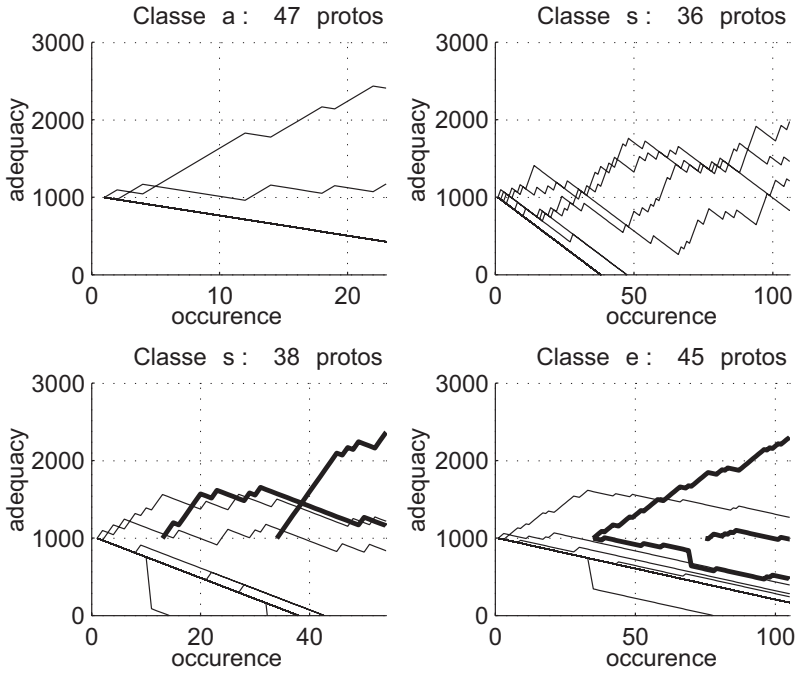

Figure 6: Prototypes adequacy evolution vs. occurrence. Thin lines are WI prototypes and bold lines are WD prototypes.

5 Supervised / self-supervised combination

We can simulate a perfect adaptation strategy if we use the prototype database determined in a supervised way in paragraph 4.1 in the reference system without knowing the text segmentation. In this case, the word error rate after 150 words of CA adaptation reaches 12 %. We just saw that the performances of the recognizer with a self-supervised CA adaptation are not far from the perfect adaptation (17 % against 12 %). It seems interesting to introduce some labelled data (*i.e.* soliciting the user to enter the real word) in the self-supervised adaptation scheme to achieve better results. So, it becomes a combination of supervised and self-supervised adaptation called semi-supervised strategy.

Soliciting the user for writing 150 words is much too constraining. On the other hand, asking him (her) to write some words is acceptable, especially if the recognition rate is largely improved. This last combination consists in carrying out a supervised adaptation of the system on some known words and then uses the self-supervised dynamic management adaptation strategy (table 4). Asking the user to write a sentence of 30 words decreases the error rate to 10 % which is even better than supervised adaptation performed alone (12 %)! We guess these very interesting results are due to the fact that, in supervised adaptation, we do not use the dynamic management of the prototypes.

6 Conclusions & Future works

In this paper, we have shown that model-based classifiers are easy to adapt. Thanks to their structure, they can learn new writings styles, by activating new prototypes and inactivating erroneous ones. We first present

Words for supervised adapt.	WER	
	After supervised adapt.	After 100 words more (DM)
0	25 %	20 %
10	24 %	17 %
20	24 %	12 %
30	**24 %**	**10 %**
50	23 %	9 %

Table 4: Word error rate (WER) in semi-supervised adaptation

a supervised adaptation strategy. It is very accurate but not user-friendly as it needs to be supervised by the writer. Then we try to hide the adaptation process and present several self-supervised strategies. The conditional activation scheme is the more accurate as it focuses on reliable words alone. The prototype dynamic management increases both recognition rate (from 75 % to 83 %) and classification speed (close to twice). This process automatically transforms a writer-independent database into a writer-dependent database of very high quality and compactness. Finally, combining supervised and self-supervised improves again the system accuracy (more than 90

It would be interesting to evaluate a semi-supervised strategy where the user is solicited only in the ambiguous cases. We have also to adapt the parameters of the segmentation expert, which actually is the biggest source of error.

References

[1] A. Brakensiek, A. Kosmala, and G. Rigoll. Comparing adaptation techniques for on-line handwriting recognition. *ICDAR*, 1:486–490, 2001.

[2] S. D. Connel and A. K. Jain. Writer adaptation of online handwriting models. *IEEE Transaction PAMI*, 24(3):329–346, 2002.

[3] I. Guyon, D. Henderson, P. Albrecht, Y. Le Cun, and J. Denker. *Writer independent and writer adaptative neural network for on-line character recognition*. S. Impedovo, From Pixels to Featuress III, Elsevier, 1992.

[4] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN project of on-line data exchange and recognizer benchmarks. *ICPR'94*, pages 29–33, 1994.

[5] H. Li. *Traitement de la variabilité et développement de systèmes robustes pour la reconnaissance de l'écriture manuscrite en-ligne*. PhD thesis, UPMC Paris 6, 2002.

[6] N. Matić, I. Guyon, J. Denker, and V. Vapnik. Writer-adaptation for on-line handwritten character recognition. *ICDAR*, 1:187–191, 1993.

[7] L. Oudot. *Fusion d'informations et adaptation pour la reconnaissance de textes manuscrits dynamiques*. PhD thesis, UPMC Paris 6, 2003.

[8] L. Oudot, L. Prevost, and M. Milgram. An activation-verification model for on-line texts recognition. *IWFHR*, 1:9–13, 2004.

[9] K. Paap, S. L. Newsome, J. E. McDonald, and R. W. Schvaneveldt. An activation-verification model for letter and word recognition: The word superiority effect. *Psychological Review*, 89:573–594, 1982.

[10] J. C. Platt and N. P. Matić. A constructive RBF network for writer adaptation. *Advances in Neural Information Processing Systems, 9*, 1:765–771, 1997.

[11] L. Prevost and M. Milgram. Modelizing character allographs in omni-scriptor frame: a new non-supervised algorithm. *Pattern Recognition Letters*, 21(4):295–302, 2000.

[12] L. Schomaker, E. H. Helsper, H.-L. Teulings, and G. H. Abbink. Adaptive recognition of online, cursive handwriting. *6th ICOHD*, 1:19–21, 1993.

[13] M. Taft. *Reading and mental lexicon*. Erlbaum Edition, 1991.

[14] A. Vinciarelli and S. Bengio. Writer adaptation techniques in HMM based off-line cursive script recognition. *Pattern Recognition Letters*, 23(8):905–916, 2002.

[15] V. Vuori, J. Laaksonen, and J. Kangas. Influence of erroneous learning samples on adaptation in on-line handwriting recognition. *Pattern Recogntion*, 35(4):915–926, 2002.

[16] L. Wang and P. Woodland. Mpe-based discriminative linear transform for speaker adaptation. *ICASSP*, 2005.