Context-dependent HMM modeling using tree-based clustering for the recognition of handwritten words

Anne-Laure Bianne[a,b], Christopher Kermorvant[a], Laurence Likforman-Sulem[b]

[a]A2iA SA, Artificial Intelligence and Image Analysis
40 bis rue Fabert, 75007 Paris, France
`http ://www.a2ia.com`
[b]Telecom ParisTech/TSI and CNRS LTCI
46 rue Barrault, 75013 Paris, France

ABSTRACT

This paper presents an HMM-based recognizer for the off-line recognition of handwritten words. Word models are the concatenation of context-dependent character models : the trigraphs. Due to the large number of possible context-dependent models to compute, a top-down clustering is applied on each state position of all models associated with a particular character. This clustering uses decision trees, based on rhetorical questions we designed. Decision trees have the advantage to model untrained triphones. Our system is shown to perform better than a baseline context independent system, and reaches an accuracy higher than 74% on the publicly available Rimes database.

1. INTRODUCTION

We present in this paper an HMM-based recognizer for the recognition of unconstrained handwritten words. Target applications consist in reading large classes of handwritten documents : archive documents, letters sent to companies or public offices. For achieving such tasks, there is a need for multi-writer systems which can cope with unconstrained handwriting and which can recognize words not present in the training data. HMM systems based on the analytical strategy are particularly suited for this problem since additional word models can be built from the concatenation of character models without training from new word images.

Several HMM-based systems have been proposed for recognizing words. A number of them use the sliding window approach for extracting features over word images[1–4] . These systems differ from the type of features extracted and the shape of the window. Recognition results for unconstrained Latin handwriting are generally reported from two large publicly available databases, namely the IAM database[5] and the recently released Rimes database[6,7] .

The HMM framework, originally applied to speech recognition, can be refined in several ways. One popular refinement in speech recognition consists in building context-dependent phone models modeling the co-articulation effect. A few works have also applied this approach to handwriting recognition[8–10] by replacing phone units by letters. This allows a system to include more accurate character models, depending on adjacent letters. This ability is very useful for modeling the deformation that can occur when drawing successive characters. In such systems the number of character models would necessitate huge amount of training data to model letters in all their context. This is why practical implementations use model and/or state clustering. Context-dependent systems differ in the way they cluster models. Our approach differs from data-driven clustering methods in that we cluster models through a decision tree using human knowledge on how characters are drawn. In addition our approach allows the system to proceed unseen contexts. This capability is useful for our recognition problem since there are a number of contexts not present in training data.

To assess the effectiveness of our approach, two systems are compared : a context-independent system (the baseline system) and a context-dependent system based on trigraphs.

[a]alb,ck@a2ia.com
[b]bianne,likforman@telecom-paristech.fr

The paper is organized as follows. Section 2.1 presents our preprocessing and feature extraction steps using the sliding window approach. Section 2.2 describes the baseline system, using context-independent models while Section 3 introduces the enhanced system with context-dependent models where states are clustered with a decision tree. We compare the enhanced system to the baseline system and to related works in Section 4. Experiments are reported on the Rimes database whose lexicon contains more than 1600 words.

2. CONTEXT-INDEPENDENT HANDWRITING RECOGNITION SYSTEM

2.1 Preprocessing and feature extraction

The first steps of a recognition system consist in preprocessing the input images and extracting features. We limit the preprocessing to deslanting images since we have observed slant in the Rimes database and negligible skew. The features extracted are independent of the height of the word image and the HMM modeling can cope with sequences of variable length, so that it can cope with variable word width. In addition, we search for the main baselines around the word core zone to capture features related to descending and ascending strokes. Deslant and baseline extraction are based on the work of Vinciarelli[1] .

Feature extraction is based on the work of El-Hajj et al.[3,11] . A sequence of feature vectors are extracted from left to right through overlapping windows. The window is a frame divided into a fixed number of cells. Within each window a set of 28 features is extracted : 13 features related to pixel densities, 2 features related to background/foreground transitions, 12 features related to pixel configurations in order to capture stroke concavities, and a derivative feature. A number of features are baseline dependent. The grey-level images are binarized by the Otsu method in order to compute features related to foreground/background transitions and pixel configurations.

For our experiments, the following parameters are used : window width of 8 pixels, overlap between two sliding windows = 4 pixels, number of cells per window = 20. These parameters happened to be the best for our deslanted images.

2.2 Context-independent character modeling

Our baseline system is generic and context-independent. It uses the analytical strategy and is segmentation-free. A word is modeled by the concatenation of its compound characters. All character models share the same HMM topology : 8 emitting states, left-right transitions with one skip allowed. The observation probability density for each state is a mixture of 20 Gaussian distributions. HMM models are trained thanks to the Baum-Welch algorithm and decoding is made with the Viterbi algorithm. We use the Hidden Markov Model Toolkit (HTK) for training and testing.

Since the baseline system is context-independent, a single model corresponds to each letter. In the following we call these context-independent models 'monographs'. We define a total of 66 (40) different case sensitive (resp. case insensitive) monographs, which can be letters, numerals, or symbols (hyphen, apostrophe, etc.).

We built two baseline systems, with an identical training process, but a different number of models to build. One is case insensitive, that is to say a and A share the same model. It enables the system to have a very large number of examples for training. The other is case insensitive, that is to say a and A have different models, which enables this system to train more specific models. However, the drawback is that there is less data for training, and thus some models might not be trained correctly. This issue is discussed in Section 4.

3. CONTEXT-DEPENDENT HANDWRITING RECOGNITION SYSTEM

Hidden Markov Models are widely used for speech recognition, and modeling has been refined to build more accurate models. As the parallel between speech and handwriting recognition has always been very interesting, we tried to apply a speech-specific tool to our problem : context-dependent models[12] . In speech recognition, the idea is that a phone has a different pronunciation when the previous and next sounds change. Hence, specific models for look-alike contexts of phones are created, as well as models for the different pronunciations of a same phone. If we transpose this to handwriting recognition, and say that a letter is equivalent to a phone, then we can exploit the fact that a letter within a word adapts its shape to its adjacent characters. A drawback of building

context-dependent models is that the number of HMM parameters related to all letter contexts becomes huge and there may be a lack of training data. This number of parameters can be reduced by sharing them between several models.

In this paper, we propose to build trigraph models and to share parameters between these trigraphs. Two types of parameters are shared : transition matrices are tied, and state models are clustered with an original decision tree-based clustering method.

["i" and "n" within the word "Monsieur"]   ["i" and "n" within the word "distinguée"]

FIGURE 1. Illustration of the different shapes a character can take for various contexts : "i" and "n" have a different shape each time they appear in the two words.

3.1 Context-dependent character modeling

The principle of context-dependent modeling of characters is quite natural, as it is obvious that our way of writing a character within a word evolves with its adjacent letters. An example of contexts influencing shape of letters can be seen on figure 1 for the lowercase characters "n" and "i". Considering that the two words have been written by the same person, we can easily imagine that the shape of letters is highly variable in a large database.

The idea of context-dependent modeling is that, instead of defining a word as a succession of characters, we define a word as a succession of characters and their contexts. The monographs of Section 2.2 are replaced by trigraphs. As we did our experiments on HTK[13] , we chose to keep the syntax of the toolkit, which is very clear. Now, a left context is defined with a minus "-" sign, and a right context with a plus "+" sign. For example, on figure 1.a, the letter "i" surrounded by a "s" and an "e" is now written : $s - i + e$. Similarly, on figure 1.b , the letter "d" surrounded by a silence and an "i" is now written : $\emptyset - d + i$.

Once the models to be trained have been defined, the next step is the training phase. Training could be done directly on the trigraphs previously defined. However, some trigraphs which don't have enough training data would be badly trained. Moreover, the amount of computations to be made would be extraordinary if we try to increase the number of Gaussian distributions : in a vocabulary of 1500 words, more than 4500 different trigraphs exist. Each final model having 8 emitting states and 20 Gaussian distribution, the number of distributions to compute would be nearly one million ! That is not realistic. This is why clustering and parameter tying is now considered, solving both problems of lack of training data and too large number of models.

3.2 HMM Parameters Tying and Clustering

3.2.1 Clustering organization

There are many ways to tie the parameters of trigraph models, as well as many ways to cluster models. In this section we will present our 3-pass tying system, illustrated on figure 2.

The first tied parameters we consider are the transition matrices. We have observed that once we chose the topology of the model (left-right, state skipping), modifications on the transition matrix coefficients had very little influence on the recognition step. Considering this, we impose that all trigraph roots ($* - a + *$, $* - b + *$, $* - c + *$, etc.) shall have the same transition matrix. Hence, the number of matrices to compute is reduced to the number of initial monographs. Our first step is then to list all the existing trigraphs in the training lexicon, copy the initialized monographs (with 8 emitting states and 1 Gaussian distribution per state) in the trigraphs, and train the trigraph models with the Baum-Welch algorithm, keeping a tying constraint on the transition matrices. This first step is illustrated as the "duplication + tie of transition matrix" part of figure 2.

The second step is a state-level clustering, applied on trigraphs. Fink et al.[10] and Natarajan et al.[9] worked with state tying, and established it as a very effective method. We differ from them by the use of decision trees to do our clustering. This will be explained in Section 3.2.2. The principle of state tying is that, for a given trigraph root, all states corresponding to the same position in an HMM model are subject to agglomerative clustering. For a given character c, N_c different trigraphs with c as central letter exist, hence for each state position i of the $* - c + *$ trigraphs, $1 \leq i \leq 8$, N_c different models of states are to be computed. The tree-based clustering of states enables the reduction of the number of different state models. It results in pools of states, so that the ith

state of each $* - c + *$ trigraph takes its value within the $s_{i,n}$ different models defined after the clustering, where $1 \leq n \leq n_{c,i}$ and $n_{c,i} < N_c$.

This step is illustrated in the "tree-based clustering" part of figure 2 for the character b. It enables us to reduce the total number of states definitions from nearly $40,000$ to only $2,500$.

FIGURE 2. Illustration of the clustering steps for the context-dependents models of the character b.

Finally, a gathering of identical models is done. Actually, state clustering meaning less state models, many trigraphs find themselves sharing exactly the same states than other ones. This enables us to reduce the number of different trigraph models nearly by a third.

3.2.2 Decision trees for trigraph clustering

Decision-tree-based clustering is an alternative to data driven clustering, based on another distance measure between models (maximization of the log-likelihood), and on a more controlled split of clusters. The merging or splitting of state clusters is driven by a binary tree whose nodes correspond to rhetorical questions on the characteristics of the models.

In our case, decision trees are based on a set of questions on the behaviour of left and right contexts, and are applied to states. One tree is built for every state position of all trigraphs with the same central letter. Starting at the root node, all the states corresponding to the same position and the same central letter are gathered in a single cluster. Then, the binary question which maximizes the likelihood of the two children clusters it would create is chosen, and the split is made, creating two new nodes. This splitting continues until the increase in likelihood falls below a threshold or no questions are available which do not pass a chosen threshold. The change in log-likelihood is calculated using the means, variances, and state occupation statistics of the single Gaussian distribution defining the states of the considered nodes :

$$d(node_1, node_2) = N_1.log|\Sigma_1| + N_2.log|\Sigma_2| - (N_1 + N_2).log|\Sigma_0| \tag{1}$$

where N_i $(i = 1, 2)$ is the number of frames used by the Baum-Welch algorithm to evaluate node i, $|\Sigma_i|$ $(i = 1, 2)$ is the determinant of node i 's covariance matrix, and $|\Sigma_0|$ is the one of the parent.

An example of decision tree is given on figure 3. It represents the tree calculated for the second state of all the trigraphs centered on the lowercase character b.

Such decision trees have been designed for speech recognition at the phone level by experts[14] . To our knowledge, there is no work using such trees for handwriting recognition. For this purpose, we defined a set of questions in order to gather look-alike character shapes in different contexts. The set of questions includes questions yielding to large clusters of models, and more precise ones yielding smaller clusters. Questions are only functions of the left and right context. Here are some examples :
  – (global question) is the left or right context uppercase, lowercase ? Has it an ascender, descender, or is it a small letter ?
  – (question on the main shape of the context) does it contain a loop ("o", "a", but also "b", "d", etc.) ? Or a bar ("t", "p", etc.) ?
  – (precise question on the shape of the context) is the link with previous/next letter on the upper baseline ("v", "w", etc.) or on the lower baseline ("a", "c", etc.) ?
The full set of questions we propose can be found on Appendix.

FIGURE 3. Example of a decision tree for state clustering : questions and clusters are shown for the 2nd state of all -*b+* trigraphs.

3.2.3 Context-Dependent Recognition system

For this system, trigraphs are modeled instead of monographs. The first step of the training phase is the clustering of models to reduce the dimension of the problem, explained in Section 3.2. Once the processes of transition matrix and state tying are completed, the system is ready to finish the training of the models with a high level of refinement. Using the Baum-Welch algorithm for model re-estimation, we increase the number of Gaussian distributions per state up to 20. We finally reach a topology of 8 emitting states with 20 Gaussian distributions per state for all the trigraphs. This enables us to compare directly our baseline system to the context-dependent one.

Apart from the nice human-driven clustering part, we chose to use trees for clustering because it is essential for decoding. Actually, had we chosen data-driven clustering, the state models would be clustered in a different way (data-driven clustering is based on euclidean distances between states), and the resulting clusters might be different. But we couldn't model new trigraphs that we had not seen in the training phase. Nonetheless, this is an essential property, as test lexicons often differ from train lexicons, so new unseen trigraphs have to be recognized.

The modeling of new trigraphs with decision trees is very simple : each state of the trigraph is positioned at the root node of the tree corresponding to the same state position and the same central letter. Then each state descends its belonging tree, answering questions on the trigraph contexts, until it reaches a node where a cluster is positioned. The state model representing the cluster will be the model assigned to the considered state number of the trigraph for its recognition. The ability of trees to model any unseen trigraph with existing ones was very helpful for our experiments on the Rimes database, as the test and train dictionaries were different, and hundreds of new trigraphs had to be modeled.

4. EXPERIMENTS

4.1 The Rimes DB

The Rimes database[6, 7] was launched in 2006 and gathers more than 12500 handwritten documents written by about 1300 volunteers. It was created to answer the recurrent issue of lack of large clean and complete databases. No constraint was imposed on writers so the panel of documents offers a large variability and makes the database a difficult though very interesting one. Based on it, it is possible to proceed logo recognition, document structure retrieval, word and character recognition for example. In our work, we focus on isolated word recognition.

Since 2007, evaluation campaigns have occurred, which will enable us to compare our results to the state of the art. We use the Rimes2008 Evaluation Campaign[15] data and evaluation procedure. 44197 images of words are given for train, 7542 images for test. Examples of what can be found are shown on figure 4. The words for the training phase are based on a lexicon of 4000 words. The test was set on a 1501 words case insensitive dictionary (1636 if case sensitive). More than 200 words are present in the test dictionary and not in the training one.

FIGURE 4. Some images of the Rimes database. Examples for the words "Je", "suite" and "plus" are shown.

The error rate is calculated case insensitive, that is to say if the system outputs a character or a word in the wrong case, the error is not counted.

4.2 Results

We have built three different systems, based on the baseline procedure or on the tied-graphs procedure.

system 1 : CSB (case sensitive baseline) This system is based on the baseline procedure. The number of models is 66 as it is case sensitive (66 = 26 uppercase characters + 26 lowercase characters + 10 numerals + 4 other type). Each model of character and numeral has 8 emitting states and 20 Gaussian distributions per state. The 4 other models (apostrophe, slash, hyphen, and silence) have 2 emitting states and 1 Gaussian distribution per state.

system 2 : CIB (case insensitive baseline)   This system is based on the baseline procedure too. The number of models is 40 (26 + 10 + 4) as it is case insensitive. Each model of character and numeral has 8 emitting states and 20 Gaussian distributions per state. The 4 other models are as in the previous system.

system 3 : CST (case sensitive with tied states)   This last system is based on the context-dependent and tied states procedure and is case sensitive. The number of different trigraphs in the training database is 4233. After state clustering, they are reduced to 1765 different trigraph models, based on 2322 different state models, each containing 20 Gaussian distributions. Each trigraph model (lowercase, uppercase and numeral) has 8 emitting states. The models based on the hyphen, the slash, or the apostrophe as central letter have 2 emitting states and 1 Gaussian distribution per state. The silence is not considered context-dependent, and is modeled like in the other systems. The addition of new words in the test dictionary compared to the training lexicon added 364 unseen trigraphs, that were modeled with the decision trees.

system	word recognition rate
CSB	$67.5\% \pm 1.1$
CIB	$68.4\% \pm 1.1$
CST	$74.1\% \pm 1.0$

TABLE 1. Comparison of the proposed context-dependent system with baseline context-independent systems

The results calculated with the Rimes2008 procedure are shown on Table 1. The word recognition rate for each system is reported together with its binomial confidence interval at 95%. The first remark we can make is that the system based on context-dependent models clearly outperforms the baseline systems (p-value $< 2.2e-16$ for a two-sided binomial test at 95%). We can also see that, for the baseline systems, the case insensitive system gives slightly better results than the case sensitive one, although this difference is not statistically significant (p-value=0.09 for a two-sided binomial test at 95%). However, for some monographs, very few training data were available : for instance, the uppercase character K had only 5 examples. This is why the CIB system could be better, as it could benefit a larger set of training data for its models. Moreover, the CIB system can be seen as a simple clustering system : models of lowercase and uppercase same letters are tied.

For the context-dependent system, the issue of lack of training data for some trigraphs is the same than for the baseline systems : even if the states of the trigraphs corresponding to the five K are gathered in a single cluster, there are still only five images to compute their models. But for a wide majority of trigraphs, an improvement is brought by the tree-based clustering of states, which gives a very high accuracy to state models. In our experiments, the improvement is of nearly 6% in absolute value compared to the CIB system, which means than 450 more images are well decoded.

As we used the Rimes2008 competition procedure and data, it is interesting to compare our results to the ones obtained by other systems, recently reported by Grosicki et al.[15] . Results are shown on Table 2. The two systems presented for comparison are also using HMMs. The one proposed by Litis[16] is based on a multi-stream segmentation free HMM, and uses two feature vector sequences. The one proposed by Itesoft is based on the Non-Symmetric Half-Plane Hidden Markov Model (NSHP-HMM). Our proposed context-dependent state-clustering system outperforms the best system of the competition by 2.2% absolute percent of recognition rate and the difference is statistically significative with a two-sided binomial test at 95% (p-value =0.002).

system	word recognition rate
Itesoft	64.0%
Litis	72.5%
proposed CST system	74.1%

TABLE 2. Comparison of the proposed context-dependent system with the state of the art

4.3 Related work

Context dependent modeling has been proposed for speech recognition. This refinement in HMM modeling has been first applied to on-line handwriting recognition[17–19] . To our knowledge, context-dependent models for

off-line handwriting recognition have been reported in 4 works[8–10, 20] . The main concerns of the literature are the amount of training data required, and the very large number of models to be created. This can be treated using semi-continuous models, or doing model clustering.

Schussler et al.[8] describe a context-dependent system using HMMs, where all sub-word units (from monographs to the whole word) are modeled within a word hierarchy. Models with not enough training samples are eliminated. They test their system on a small and dynamical lexicon.

Natarajan et al.[9] use parameter tying for each state position. A pool of 128 Gaussian distributions for each state of each character is defined. This is quite similar to our state clustering, however is it performed with data-driven clustering.

More recently, Fink et al.[10] also proposed a system based on context dependent models and a state by state data-driven clustering. They managed to improve their results (word error rate from 24.0% to 22.7%), but they judged that the performance gains were not worth the additional computations made.

El Hajj et al.[20] , for arabic handwriting, clusters models at hand into broad classes : trigraphs preceded by a descending character are clustered.

Our CST system proposes parameter tying for the HMM states, as Fink et al.[10] and Natarajan et al.[9] did. But we perform this tying using decision trees, whose questions are designed thanks to human knowledge on how characters are drawn. This approach improves performances in higher proportion than for the above approaches : compared to our CSB baseline system, accuracy is increased by 6.6% in absolute value, which corresponds to a 20.4% relative reduction in error rate.

5. CONCLUSION

We have presented a context-dependent system based on Hidden Markov Models and state clustering for off-line handwritten word recognition. HMMs are not designed for single characters, but for trigraphs, which increases the number of models to compute. This number is reduced thanks to a clustering of states based on an original decision tree. The decision tree clustering also enables us to deal with unseen trigraphs in the test dictionary.

Compared to a baseline system with context independent HMMs, accuracy is increased by 6.6% in absolute value which corresponds to a 20.4% relative reduction in error rate. In addition, the performance of the context-dependent state-tying system is improved compared to the published results on the Rimes database in 2008[15] .

There are still a lot to explore in the decision-tree clustering, such as maximization steps, or cluster thresholds. We could also explore different ways to create questions for decision trees. New questions could be generated that would be more accurate and give a higher increase in likelihood when tree splitting.

REFERENCES

[1] Vinciarelli, A. and Luettin, J., "A new normalization technique for cursive handwritten words," *Pattern Recognition Letters* **22**(9), 1043–1050 (2001).

[2] Toselli, A. H., *Reconocimiento de Texto Manuscrito Continuo*, PhD thesis, Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia (2004).

[3] Al-Hajj Mohamad, R., Likforman-Sulem, L., and Mokbel, C., "Combining slanted-frame classifiers for improved hmm-based arabic handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31** (2009).

[4] Rodriguez, J. A. and Perronnin, F., "Local gradient histogram features for word spotting in unconstrained handwritten documents," in [*Proceedings of the 1st International Conference on Handwriting Recognition - ICFHR08*], (August 2008).

[5] Marti, U. and Bunke., H., "A full english sentence database for off-line handwriting recognition," in [*Proceedings of the 5th International Conference on Document Analysis and Recognition - ICDAR99*], 705–708 (1999).

[6] Augustin, E., Carre, M., Grosicki, E., Brodin, J.-M., Geoffrois, E., and Preteux, F., "Rimes evaluation campaign for handwritten mail processing," in [*Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition - IWFHR06*], 231–235 (October 2006).

[7] http ://rimes.it sudparis.eu/.

[8] Schüßler, M. and Niemann, H., "A hmm-based system for recognition of handwritten address words," in [*Proceedings of Sixth Int. Workshop on Frontiers in Handwriting Recognition - IWFHR98*], 505–514 (1998).

[9] Natarajan, P., Saleem, S., Prasad, R., MacRostie, E., and Subramanian, K., "Multi-lingual offline handwriting recognition using hidden markov models : A script-independent approach," in [*Summit on Arabic and Chinese Hanwriting - SACH06*], (2006).

[10] Fink, G. and Plotz, T., "On the use of context-dependent modeling units for HMM-based offline handwriting recognition," *International Conference on Document Analysis and Recognition - ICDAR07* **2**, 729–733 (2007).

[11] El-Hajj, R., Likforman-Sulem, L., and Mokbel, C., "Arabic handwriting recognition using baseline dependant features and hidden markov modeling," in [*Proceedings of the Eighth International Conference on Document Analysis and Recognition - ICDAR05*], 893–897 (2005).

[12] Lee, K.-F., "Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition," *Readings in speech recognition* , 347–366 (1990).

[13] Young, S. and al., [*The HTK Book V3.4*], Cambridge University Press, Cambridge, UK (2006).

[14] Chelba, C. and Morton, R., "Mutual information phone clustering for decision tree induction," in [*Proceedings of the International Conference on Spoken Language Processing -ICSLP02*], (2002).

[15] Grosicki, E., Carre, M., Brodin, J.-M., and Geoffrois, E., "Results of the second rimes evaluation campaign for handwritten mail processing," in [*Proceedings of the 10th International Conference on Document Analysis and Recognition ICDAR09*], (2009).

[16] Kessentini, Y., Paquet, T., and Benhamadou, A., "A multi-stream hmm-based approach for off-line multiscript handwritten word recognition," in [*11th International Conference on Frontiers in Handwriting Recognition - ICFHR08*], (2008).

[17] Kosmala, A., Rottland, J., and Rigoll, G., "Improved on-line handwriting recognition using context dependent hidden markov models," in [*Proc. Int. Conference on Document Analysis and Recognition - ICDAR97*], 641–644 (1997).

[18] Tokuno, J., Inami, N., Matsuda, S., Nakai, M., Shimodaira, H., and Sagayama, S., "Context-dependent substroke model for hmm-based on-line handwriting recognition," in [*Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition - IWFHR02*], (2002).

[19] G. Rigoll, A. K. and Willett, D., "An investigation of context-dependent and hybrid modeling techniques for very large vocabulary on-line cursive handwriting recognition," (1998).

[20] El-Hajj, R., Mokbel, C., and Likforman-Sulem, L., "Recognition of arabic handwritten words using contextual character models," *Document Recognition and Retrieval XV - SPIE* (January 2008).

6. APPENDIX : QUESTIONS FOR DECISION TREES

This is the complete set of questions we propose for the tree-based clustering of states. Questions depend only on left and right contexts and have different levels of precision. Please note that :

- QS = question
- LC=lowercase, UC=uppercase
- sA = apostrophe, sB = slash and backslash, sT = hyphen
- *+character = right context (R_) and −*character = left context (L_)
- zero and five (0 and 5) do not exist as I suppose they are the same as O (letter o uppercase) and S (letter s uppercase) respectively

```
/is the right context of the trigraph an apostrophe?
QS "R_isSA" {*+sA}

/is the right context of the trigraph a slash?
QS "R_isSB" {*+sB}

/is the right context of the trigraph an hyphen symbol?
QS "R_isST" {*+sT}

/is the right context of the trigraph lowercase?
QS "R_lowercase" {*+1,*+8,*+a,*+b,*+c,*+d,*+e,*+f,*+g,*+h,*+i,*+j,*+k,*+l,*+m,*+n,*+o,*+p,*+q,*+r,*+s,
    *+t,*+u,*+v,*+w,*+x,*+y,*+z}

/is the right context of the trigraph uppercase?
QS "R_uppercase" {*+2,*+3,*+4,*+6,*+7,*+9,*+A,*+B,*+C,*+D,*+E,*+F,*+G,*+H,*+I,*+J,*+K,*+L,*+M,*+N,*+O,
    *+P,*+Q,*+R,*+S,*+T,*+U,*+V,*+W,*+X,*+Y,*+Z}

/has the right context of the trigraph a descender?
QS "R_LC_descender" {*+f,*+g,*+j,*+p,*+q,*+y,*+z}

/etc...
QS "R_LC_ascender" {*+1,*+8,*+b,*+d,*+f,*+h,*+k,*+l,*+t}
QS "R_LC_small" {*+a,*+c,*+d,*+e,*+i,*+m,*+n,*+o,*+q,*+r,*+s,*+u,*+v,*+w,*+x,*+z}
QS "R_LC_loopsmall" {*+8,*+a,*+c,*+d,*+g,*+e,*+o,*+q}
QS "R_LC_loopascender" {*+8,*+b,*+f,*+h,*+k,*+l}
QS "R_LC_loopdescender" {*+f,*+g,*+j,*+y,*+z}
QS "R_LC_barascender" {*+1,*+b,*+f,*+h,*+k,*+l,*+t}
QS "R_LC_bardescender" {*+f,*+p}
QS "R_LC_nushape" {*+m,*+n,*+r,*+u,*+v,*+w,*+y}
QS "R_LC_complexwithinbaseline" {*+a,*+b,*+e,*+k,*+o,*+r,*+s,*+x,*+z}
QS "R_LC_starthigh" {*+1,*+v,*+w,*+x,*+z}
QS "R_LC_point" {*+i,*+j}
QS "R_LC_e" {*+e}
QS "R_UC_verticalbar" {*+A,*+B,*+E,*+F,*+H,*+K,*+L,*+P,*+R,*+U}
QS "R_UC_middlehorizontalbar" {*+6,*+7,*+A,*+B,*+E,*+F,*+H,*+K,*+P,*+R,*+S}
QS "R_UC_uphorizontalbar" {*+7,*+E,*+F,*+I,*+J,*+M,*+T,*+Z}
QS "R_UC_bottomhorizontalbar" {*+4,*+2,*+E,*+I,*+L,*+Z}
QS "R_UC_upcurve" {*+3,*+2,*+F,*+H,*+K,*+P,*+R,*+U,*+V,*+W,*+X,*+Y}
QS "R_UC_bottomcurve" {*+3,*+A,*+F,*+H,*+I,*+J,*+K,*+M,*+N,*+P,*+R,*+S}
QS "R_UC_bottomloop" {*+2,*+B,*+D,*+L,*+Q,*+Z}
QS "R_UC_bigloop" {*+C,*+G,*+O,*+Q}
QS "R_UC_descender" {*+G,*+J,*+Y}
QS "R_UC_uplefttobottomrightbar" {*+V,*+W,*+X,*+Y}
QS "R_UC_bottomlefttouprightbar" {*+7,*+4,*+2,*+A,*+X,*+Z}
QS "R_NUM_is3" {*+3}
QS "R_NUM_is6" {*+6}
```

```
QS "R_NUM_is9" {*+9}
QS "L_isSA" {sA-*}
QS "L_isSB" {sB-*}
QS "L_isST" {sT-*}
QS "L_lowercase" {8-*,9-*,a-*,b-*,c-*,d-*,e-*,f-*,g-*,h-*,i-*,j-*,k-*,l-*,m-*,n-*,o-*,p-*,q-*,r-*,s-*,
    t-*,u-*,v-*,w-*,x-*,y-*,z-*}
QS "L_uppercase" {1-*,2-*,3-*,4-*,6-*,7-*,A-*,B-*,C-*,D-*,E-*,F-*,G-*,H-*,I-*,J-*,K-*,L-*,M-*,N-*,O-*,
    P-*,Q-*,R-*,S-*,T-*,U-*,V-*,W-*,X-*,Y-*,Z-*}
QS "L_LC_descender" {f-*,g-*,j-*,p-*,q-*,y-*,z-*}
QS "L_LC_ascender" {8-*,9-*,b-*,d-*,f-*,h-*,k-*,l-*,t-*}
QS "L_LC_small" {a-*,b-*,c-*,e-*,h-*,i-*,k-*,m-*,n-*,o-*,p-*,r-*,s-*,u-*,v-*,w-*,x-*,z-*}
QS "L_LC_loopsmall" {8-*,a-*,b-*,o-*,p-*}
QS "L_LC_loopascender" {8-*,b-*,f-*,h-*,k-*,l-*}
QS "L_LC_loopdescender" {f-*,g-*,j-*,y-*,z-*}
QS "L_LC_barascender" {9-*,d-*,f-*,l-*}
QS "L_LC_bardescender" {f-*,j-*,q-*,y-*}
QS "L_LC_nushape" {m-*,n-*,u-*}
QS "L_LC_complexwithinbaseline" {a-*,b-*,e-*,k-*,o-*,s-*,r-*,x-*,z-*}
QS "L_LC_linklow" {a-*,c-*,d-*,e-*,f-*,g-*,h-*,i-*,j-*,k-*,l-*,m-*,n-*,p-*,q-*,r-*,s-*,t-*,u-*,x-*,y-*}
QS "L_LC_linkhigh" {b-*,o-*,r-*,s-*,t-*,v-*,w-*,x-*,z-*}
QS "L_LC_point" {i-*,j-*}
QS "L_UC_middlehorizontalbar" {6-*,7-*,A-*,3-*,B-*,E-*,F-*,G-*,H-*,P-*,S-*}
QS "L_UC_uphorizontalbar" {7-*,E-*,F-*,I-*,J-*,M-*,T-*,Z-*}
QS "L_UC_bottomhorizotalbar" {2-*,4-*,E-*,I-*,L-*,Z-*}
QS "L_UC_bottomcurve" {A-*,H-*,K-*,L-*,N-*,M-*,R-*,U-*,Y-*,Z-*}
QS "L_UC_upcurve" {6-*,F-*,H-*,K-*,N-*,T-*,X-*}
QS "L_UC_uploopfromleft" {C-*,E-*,N-*,M-*,S-*,Z-*}
QS "L_UC_uploopfrombottom" {O-*,V-*,W-*}
QS "L_UC_halfcircleup" {2-*,3-*,B-*,P-*,R-*}
QS "L_UC_halfcirclebottom" {6-*,3-*,B-*,S-*}
QS "L_UC_uplefttobottomrightbar" {A-*,K-*,Q-*,R-*,X-*}
QS "L_UC_bottomlefttouprightbar" {7-*,K-*,V-*,W-*,X-*,Y-*,Z-*}
QS "L_UC_bigloop" {D-*,O-*,Q-*}
QS "L_UC_descender" {4-*,G-*,J-*,Y-*}
QS "L_NUM_is2" {2-*}
QS "L_NUM_is9" {9-*}
```