EAMOMAC, un environnement d'aide à la modélisation à base de modèles de Markov cachés

Mawloud OMAR [(1)], Mâamar HAMADOUCHE [(2)] et Mourad LOUKAM [(3)]

(1) Université Abderahmane Mira de Bejaïa
Département d'Informatique (ReSyD), Targa Ouzemour, Bejaïa, Algérie.
mawloud.omar@gmail.com

(2) Université Saad Dahlab de Blida
Département d'Informatique, BP 270, Blida, Algérie.
hamadouchemaamar@yahoo.fr

(3) Université Hassiba Ben Bouali de Chlef
Département d'Informatique, Hay Essalem, Chlef, Algérie.
Loukam@hotmail.com

Résumé – Abstract

Les modèles de Markov cachés (ou MMC) suscitent depuis plusieurs années un vif intérêt dans le domaine de la modélisation et se sont rapidement imposés comme une référence dans plusieurs applications, comme : la reconnaissance de la parole, la reconnaissance optique, la classification automatique, l'extraction de données textuelles, la reconnaissance des séquences d'ADN, etc. Dans cet article, nous présentons notre système EAMOMAC (Environnement d'aide à la Modélisation à base de modèles de Markov cachés). EAMOMAC est un système offrant un environnement interactif qui permet de résoudre les problèmes essentiels des MMC : l'évaluation, le décodage et l'apprentissage.

The hidden Markov models (or HMM) have aroused since many years a great interest in the modeling fields. They are useful in several applications, like voice/optical recognition, automatic classification, textual data extraction, ADN sequences recognition, etc. In this paper, we present our system named EAMOMAC. It is an integrated system based upon the hidden Markov models. This system offers an interactive environment including the fundamental HMM algorithms: evaluation, decoding and training.

Mots Clés – Keywords

MMC, Forward–Backward, Viterbi, Baum–welch.

HMM, Forward–Backward, Viterbi, Baum–welch.

1 Introduction

Les modèles de Markov cachés (ou MMC) ont été introduits par Baum et *al.* dans les années 1960. Ce modèle est fortement apparenté aux automates probabilistes (F. Casacuberta, 1990). Un automate probabiliste est défini par une structure composée d'états et de transitions, et par une distribution de probabilités sur les transitions. À chaque transition est associé un symbole d'un alphabet fini. Ce symbole est généré à chaque fois que la transition est empruntée. Un MMC se définit également par une structure composée d'états et de transitions et par une distribution de probabilités sur les transitions. La différence qui existe par rapport aux automates probabilistes est que la génération des symboles s'effectue sur les états et non pas sur les transitions. De plus, on associe à chaque état non pas un symbole, mais une distribution de probabilités sur les symboles de l'alphabet (N. Abe, M. Warmuth, 1992).

Les MMC sont utilisés pour modéliser des séquences d'observations. Ces observations peuvent être de nature discrète (par exemple, les caractères d'un alphabet fini) ou continue (par exemple, la fréquence d'un signal, la température, etc.). Le premier domaine auquel les MMC ont été appliqués est le traitement de la parole au début des années 1970 (J.K. Baker, 1975). Dans ce domaine, les MMC se sont rapidement imposés comme le modèle de référence et une majeure partie des techniques d'implémentation a été développée dans le cadre de ces applications. Ces techniques furent ensuite appliquées et adaptées avec succès aux problèmes de la reconnaissance de textes manuscrits, et d'analyse de séquences biologiques (D. Haussler et *al.*, 1992). Les MMC ont également été appliqués à d'autres domaines comme la reconnaissance d'images, la modélisation d'un signal acoustique ou la génération de séquences de tests pour circuits microélectroniques. Depuis récemment, avec l'augmentation importante de la masse des données électroniques, une nouvelle application prometteuse semble être l'extraction d'informations à partir de données textuelles (K. Seymore et *al.*, 1999).

Dans cet article, nous présentons notre système EAMOMAC, un système offrant un environnement interactif qui permet de résoudre les problèmes essentiels des MMC.[1]

2 Etat de l'Art

2.1 Structure des MMC

Un MMC est un modèle stochastique particulier, il représente un objet donné par deux suites de variables aléatoires. L'une dite cachée et l'autre observable. La suite cachée correspond à la suite d'états $Q_T = q_1, q_2, ..., q_T$, où les q_i puisent leur valeur parmi l'ensemble des N états du modèle. La suite observable correspond à la suite d'observations $O_T = o_1, o_2, ..., o_T$, où les o_i sont en fonctions du temps et se réalisent parmi un ensemble de M symboles observables (L. Remaki, J.G. Meumier, 2000). Formellement, un MMC est caractérisé par ce que l'on appelle les paramètres complets du modèle $\lambda = (\pi, A, B)$, tel que :

- $\pi = \{\pi_i\}$ représente la distribution de l'état initial : $\pi_i = P(q_1 = S_i)$.

[1] EAMOMAC est disponible sur : http://membres.lycos.fr/omarmawloud/eamomac.rar.

- $A = \{a_{ij}\}$ représente la distribution de probabilités sur les transitions :

  $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$.

- $B = \{b_j(k)\}$ représente la distribution de probabilités de générations à l'état j :

  $b_j(k) = P(V_k \; aut | q_t = S_j)$

2.2 Les Problèmes Fondamentaux des MMC

2.2.1 Problème d'Evaluation

Etant donnée une séquence d'observations $O = o_1, o_2, ..., o_T$ et un modèle λ. Le problème d'évaluation consiste à calculer $P(O|\lambda)$.

2.2.2 Problème de Décodage

Etant donnée une séquence d'observations $O = o_1, o_2, ..., o_T$ et un modèle λ. Le problème de décodage consiste à trouver la séquence d'états $Q = q_1, q_2, ..., q_T$ qui maximise $P(O|\lambda)$.

2.2.3 Problème d'Apprentissage

Etant donnée une séquence d'observations $O = o_1, o_2, ..., o_T$ et un modèle λ. Le problème consiste à re-estimer les paramètres du modèle afin de maximiser $P(O|\lambda)$. La séquence O est appelée aussi séquence d'apprentissage.

2.3 Travaux Antérieurs

Plusieurs travaux ont été réalisés dans le cadre de mise en œuvre d'outils d'aide à la modélisation à base de modèles de Markov cachés. Nous citons notamment :

- GHMM, General Hidden Markov Model library (A. Schliep et al., 2004).
- HTK, *Hidden Markov Model Toolkit* (S. Yang et *al.*, 2002).
- *Hidden Markov Model Toolbox* (K. Murphy, 2002).
- HMM mini–toolkit (A. Venkataraman, 2000).
- HMM *Software* (A. Kornai, 1999).
- Meta–MEME (W. Nobel et *al.*, 1997).
- DHMM, *Discrete HMM toolkit* (J. Deller et *al.*, 1993).

La plupart de ces outils proposent des librairies de programmes utilisant les HMM et pouvant être utilisées dans un domaine particulier. Notre système EAMOMAC s'inscrit lui aussi dans

cette lignée, mais il offre en plus un aspect que nous considérons important : l'**interactivité**. En effet, avec EAMOMAC, l'utilisateur utilise un environnement graphique intégré qui l'assiste dans toutes les étapes de manipulation des HMM : introduction et choix des paramètres, lancement des calculs, vérification des modèles, visualisation de la trace des résultats, etc.

3 Architecture de EAMOMAC

EAMOMAC se compose de deux modules, qui traitent chacun des fonctionnalités spécifiques. Les interactions entre les deux couches sont assurées par des liens fonctionnels (cf. Figure 1).

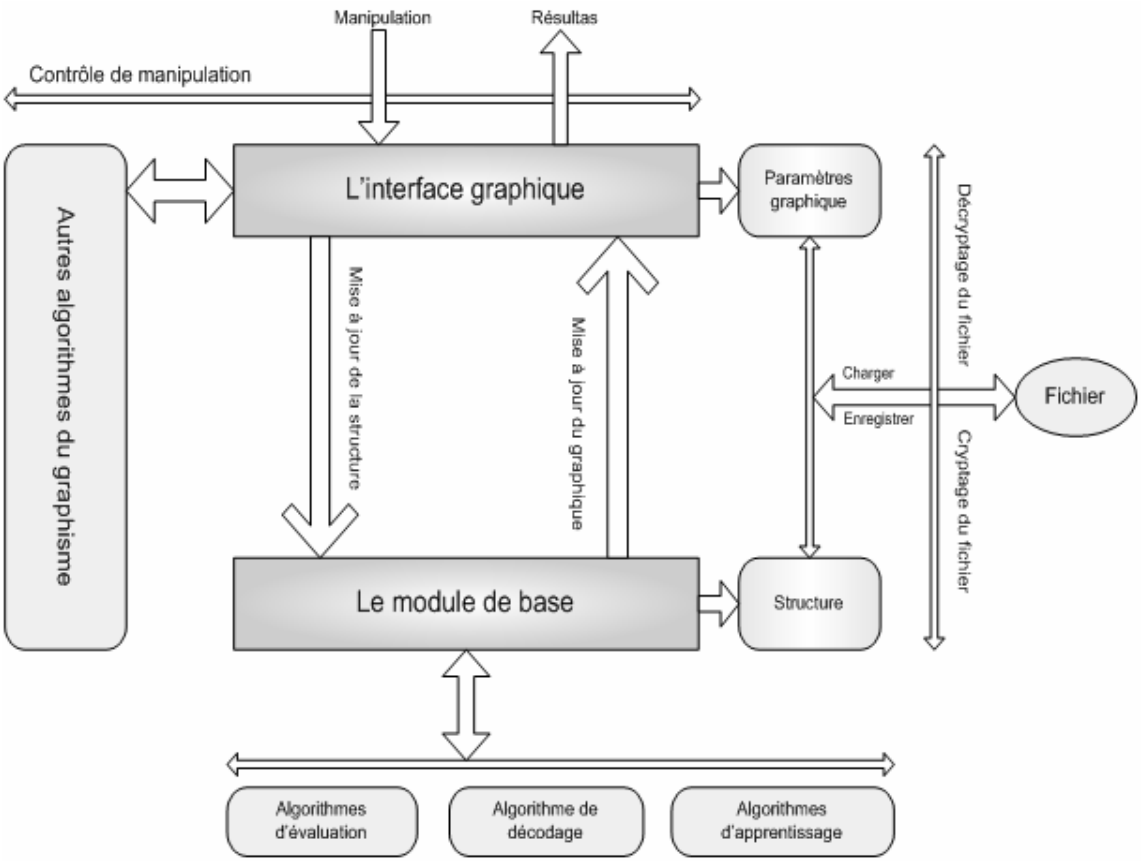

Figure 1: Architecture de EAMOMAC.

3.1 Le Module de l'Interface Graphique

Cette couche, représente le module par lequel l'utilisateur manipule des objets graphiques. Les objets ne sont rien d'autres que des états et des transitions. Ce module assure les différentes interactions de l'utilisateur (Création, suppression, édition, etc.).

3.2 Le Module de Base

Ce module fournit la résolution des trois problèmes des MMC, par l'implémentation des algorithmes d'évaluation *Forward–Backward*, l'algorithme de décodage de *Viterbi*, et les algorithmes d'apprentissages de *Viterbi* et *Baum–welch* (L.R. Rabiner, 1989).

3.2.1 Algorithme d'Evaluation Forward

- Initialisation : pour tout i∈ [1, N]

$$\alpha_1(i) = \pi_i b_i(o_1)$$

- Induction : pour tout t∈ [1, T-1] et pour tout j∈ [1, N]

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i).a_{ij}\right).b_j(o_{t+1})$$

- Terminaison :

$$p(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

3.2.2 Algorithme d'Evaluation Backward

- Initialisation : pour tout i∈ [1, N]

$$\beta_T(i) = 1$$

- Induction : pour tout t∈ [1, T-1] et pour tout j∈ [1, N]

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}.b_j(o_{t+1}).\beta_{t+1}(j)$$

- Terminaison :

$$p(O|\lambda) = \sum_{i=1}^{N} \pi_i.\beta_1(i).b_i(o_1)$$

3.2.3 Algorithme de Décodage de Viterbi

- Initialisation :

$$\delta_1(i) = \pi_i.b_i(o_1)$$
$$\Phi_1(i) = 0$$

- Induction :

$$\delta_t(j) = \max_{1\le i\le N}\left(\delta_{t-1}(i).a_{ij}\right)b_j(o_t) \qquad 2 \le t \le T$$
$$\Phi_t(j) = \arg\max_{1\le i\le N}\left(\delta_{t-1}(i).a_{ij}\right) \qquad 1 \le j \le N$$

- Terminaison :

$$\hat{p} = \max_{1\le i\le N} \delta_T(i)$$
$$\hat{q}_T = \arg\max_{1\le i\le N} \delta_T(i)$$

- Rétro propagation :

$$\hat{q}_t = \Phi_{t+1}(\hat{q}_{t+1}), \qquad t = T-1, T-2, \ldots, 1$$

3.2.4 Algorithme d'Entraînement de Viterbi

Répéter

- Calculer le chemin de Viterbi de O suivant λ ;
- Calculer les variables n_s, $n_{s,s'}$ et n_s^o de O ;
  - n_s : le nombre de fois où l'état s est utilisé.
  - $n_{s,s'}$: le nombre de fois où la transition s→s' est utilisée.
  - n_s^o : le nombre de fois où le symbole o est généré par s.
- Ré estimer les paramètres λ de H en utilisant les formules des estimateurs :

$$\overline{P}(s \to s') = \frac{n_{s \to s'}}{n_s} \qquad \overline{P}(o|s) = \frac{n_s^o}{n_s}$$

Jusqu'à la stabilité du paramétrage;

3.2.5 Algorithme d'Entraînement de Baum–welch

Répéter

- Calculer $\alpha_t(i)$ à l'aide de l'algorithme Forward;
- Calculer $\beta_t(i)$ à l'aide de l'algorithme Backward;
- Calculer $\xi_t(i, j)$;
- Calculer $\gamma_t(i)$;

$$\zeta_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \qquad \gamma_t(i) = \sum_{j=1}^{N} \zeta_t(i,j)$$

- Ré estimer les paramètres λ de H en utilisant les formules suivantes :

$$\overline{\pi}_i = \gamma_1(i)$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \overline{b}_j(k) = \frac{\sum_{t=1;o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Jusqu'à critère d'arrêt;

On peut prouver que la probabilité de génération des séquences $P(O|\lambda)$ augmente à chaque itération et que l'algorithme converge vers un optimum local. Néanmoins, dans ce cas là, l'optimum local n'est jamais atteint et il est nécessaire de définir un critère d'arrêt, par exemple en stoppant la procédure lorsque l'augmentation de $P(O|\lambda)$ est inférieure à un certain seuil, ou lorsque le nombre d'itérations effectuées est jugé suffisant.

EAMOMAC offre le choix du critère d'arrêt : soit en introduisant le nombre d'itérations ou bien le seuil. A chaque itération EAMOMAC vérifie la stabilité du paramétrage, afin de garantir la terminaison de l'algorithme. Les deux figures suivantes nous montrent un exemple d'entraînement sur la séquence d'apprentissage : "*VVRV*", respectivement avec l'algorithme de *Viterbi* et l'algorithme de *Baum–welch*.

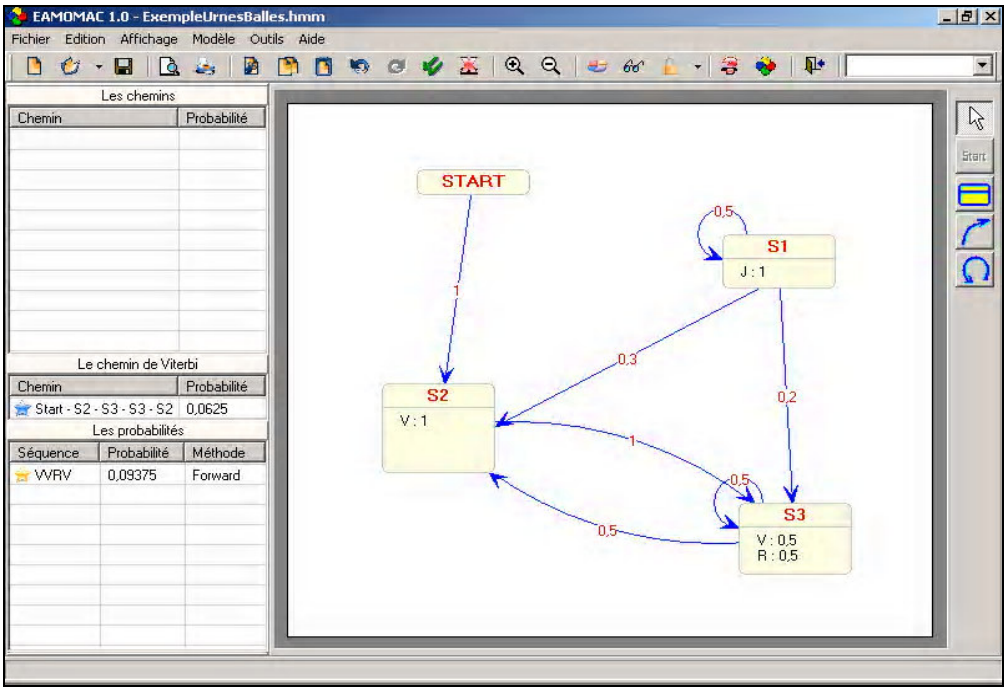

Figure 2 : Entraînement de *Viterbi*.

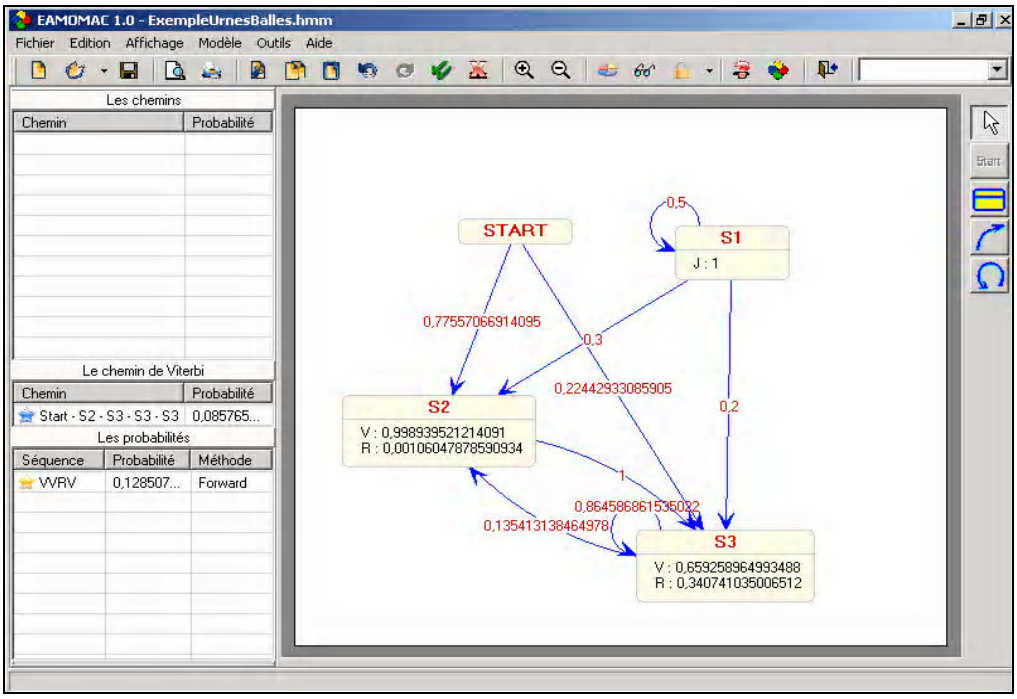

Figure 3 : Entraînement de *Baum–welch*.

4 Conclusion

Dans cet article, nous avons présenté EAMOMAC, notre système d'aide à la modélisation à base de modèles de Markov cachés. Ce système se distingue des produits existant actuellement par son environnement intégré. Il implémente la plupart des algorithmes s'appliquant aux MMC : calcul de probabilités de génération d'une séquence, le chemin optimal et l'apprentissage à structure discrète. Des fonctions de paramétrage et d'explications de résultats sont également fournies aux utilisateurs.

Plusieurs extensions peuvent être proposées. Nous pouvons citer entre autres : l'implémentation des méthodes d'apprentissage avec des structures inconnues (les probabilités de transition et de génération ne sont pas connues à l'avance); ce qui permettra la construction des MMC directement à partir de séquences d'apprentissage.

Références

A. Schliep, B. Georgi, W. Rungsarityotin, I.G. Costa, A. Schönhuth (2004), The General Hidden Markov Model Library: Analyzing Systems with Unobservable States, *in Proceedings of the Heinz-Billing-Price*, 121-136.

S. Yang, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollanson, D. Povey, V. Valtchev, P. Woodland (2002), The HTK book (for HTK version 3.2.1), *In Cambridge University Engineering Department*.

K. Murphy (2002), Hidden Markov Model Toolbox, http://www.cs.berkeley.edu/murphyk/Bayes/hmm.html.

A. Venkataraman (2000), The PFSA formulation of HMMs and adapted versions of some useful algorithms, *Computer Science, IIST Massey University*, software available in: http://popper.massey.ac.nz/~ARaman/hmm+pfsa.tar.gz.

L. Remaki, J.G. Meumier (2000), Un modèle HMM pour la détection des mots composés dans un corpus textuel.

K. Seymore, A. McCallum, R. Rosenfeld (1999), Learning hidden Markov model structure for Information Extraction, *In AAAI'99 Workshop – ML for Information Extraction*, pages 37–42.

A. Kornai (1999), UMDHMM: Hidden Markov Model Toolkit, in Extended Finite State Models of Language, *Cambridge University Press*.

W. Nobel, T. Bailey, C. Elkan, M. Baker (1997), Meta-MEME: Motif-based Hidden Markov Models of Biological Sequences, *Computer Applications in the Biosciences*, 13(4):397-406.

J. Deller, J. Proakis, J. Hansen (1993), Discrete-Time Processing of Speech Signals, *MacMillan*, ISBN: 0-02-328301-7.

D. Haussler, A. Krogh, S. Mian, K. Sjolander (1992), Protein modeling using hidden Markov models: analysis of globins, *Technical Report*.

N. Abe, M. Warmuth (1992), On the computational complexity of approximating distributions by probabilistic automata, *Machine Learning*, 9(2-3):205-260.

F. Casacuberta (1990), Some relations among stochastic finite state networks used in automatic speech recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7): 691-695.

L.R. Rabiner (1989), A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, 77(2): 257-285.

J.K. Baker (1975), The DRAGON system – An overview, *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23: 24-29.