# Entropie relative et algorithmes d'inférence grammaticale probabiliste

Franck Thollard et Pierre Dupont

EURISE, Université Jean Monnet
23 rue Dr P Michelon, 42023 Saint-Etienne

{thollard,pdupont}@univ-st-etienne.fr

**Résumé**

Nous nous intéressons aux algorithmes d'inférence de langages probabilistes par fusions d'états. Nous proposons un critère qui mesure l'influence d'une fusion entre deux états sur la qualité globale d'un automate probabiliste. Ce critère est basé sur la divergence de Kullback-Leibler. Nous étudions le calcul de la divergence entre les distributions générées par deux automates quelconques. Nous montrons les limitations de ce calcul et nous proposons un calcul exact et efficace de la divergence dans le cas particulier où les distributions sont générées par des automates obtenus par fusions d'états.

## 1 Introduction

En inférence grammaticale stochastique, une certaine classe d'algorithmes ([1], [6], [7]) travaillent par généralisation d'un automate, l'arbre des préfixes, qui représente exactement les données d'apprentissage. La technique de généralisation consiste à fusionner les états considérés comme similaires. Le test de similarité s'appuie sur une estimation statistique de la distance entre deux états. Plus précisément, les états sont associés aux distributions de probabilités définies sur l'ensemble des suffixes de ces états. Le test de similarité est alors une estimation de distance entre ces deux distributions.

Nous proposons une étude de l'entropie relative. Cette étude nous permet, sous certaines conditions, de calculer l'entropie relative entre deux automates stochastiques. Nous disposons alors d'une mesure de similarité entre automates et non entre états.

Après un rappel des définitions, nous donnons un algorithme d'inférence grammaticale stochastique générique. Nous montrons ensuite comment l'entropie relative peut être utilisée pour guider l'inférence. Nous détaillons enfin le cas particulier dans lequel l'entropie peut être calculée efficacement et exprimons explicitement son calcul.

## 2 Définitions et notations

### 2.1 Les SDFA

- Soit $\Sigma$ un alphabet fini, et $\Sigma^*$ l'ensemble des chaînes construites sur $\Sigma$. Soit $\lambda$ la chaîne vide. Pour $x \in \Sigma^*$, on note $x\Sigma^*$ l'ensemble des préfixes des mots de $\Sigma^*$ ayant $x$ comme préfixe.
- $\mathrm{C}_E(x)$ désigne le nombre d'occurrences de $x$ dans l'ensemble de chaînes $E$.
- Un *Stochastic Deterministic Finite Automaton* (SDFA) A est un quintuplet $(\Sigma, Q_A, q_I, \delta_A, \gamma_A)$ où :

  $\Sigma$ est un alphabet fini

  $Q_A$ est l'ensemble des états de A

  $q_I \in Q_A$ est l'état initial de A

  $\delta_A$: $Q_A \times \Sigma \to Q_A$ est la table de transition de A.

  $\gamma_A$ : $Q_A \times \Sigma \cup \{\lambda\} \to ]0,1]$ est la fonction de probabilités de A.

  Lorsqu'il n'y aura pas d'ambiguïté, on notera simplement $(\Sigma, Q, q_I, \delta, \gamma)$

  Les SDFA ainsi définis sont des générateurs de mots de $\Sigma^*$. On note $P_A(x)$ la probabilité que l'automate $A$ génère la chaîne $x$. Cette probabilité se calcule en utilisant les probabilités des symboles qui composent $x$. Pour $a \in \Sigma$ et $q_i \in Q_A$, la fonction de probabilités $\gamma_A$ définit la probabilité de générer le symbole $a$ lorsque l'on est dans l'état $q_i$. On la note $P_A(a|q_i)$. Pour $a = \lambda$, la probabilité $P_A(a|q_i)$ est la probabilité d'arrêter le processus de génération. Plus formellement, pour $x = x_0x_1...x_n \in \Sigma^*$, le langage $L_A$ engendré par le SDFA $A$ s'écrit:

$$
\begin{aligned}
L_A &= \{x \in \Sigma^* | \exists q_{i_1}, q_{i_2}, ...q_{i_{n+1}} \in Q_A, \\
&\quad \delta(q_{i_j}, x_j) = q_{i_{j+1}},\ 0 \leq j \leq n, \\
&\quad q_{i_0} = q_I \text{ et } \gamma(q_{i_{n+1}}, \lambda) \neq 0\} \qquad (1)
\end{aligned}
$$

On notera $q_{i_{n+1}} = \delta(q_I, x)$ si le chemin de génération de $x$ finit en $q_{i_{n+1}}$.
La probabilité de générer $x$ s'écrit alors :

$$P_A(x) = (\prod_{k=0}^{n} \gamma(q_{i_k}, x_k)) \cdot \gamma(q_{i_{n+1}}, \lambda) \quad \text{avec } q_{i_0} = q_I$$

$L_A$ peut également se voir comme : $L_A = \{x \in \Sigma^* | P_A(x) \neq 0\}$. Pour que $P_A(x)$ définisse une distribution de probabilités sur $\Sigma^*$, il faut que les probabilités des transitions sortantes de chaque état de l'automate somment à 1. Plus formellement, on doit avoir $\sum_{a \in \Sigma \cup \{\lambda\}} \gamma_A(q_i, a) = 1$.

- Soit $q_i \in Q_A$, on note $Pref_A(q_i)$ l'ensemble des préfixes menant à l'état $q_i$ dans $A$. Formellement, $Pref_A(q_i) = \{x \in \Sigma^* | \delta(q_I, x) = q_i\}$. Lorsqu'il n'y aura pas d'ambiguïté on notera simplement $Pref(q_i)$.

## 2.2 Des automates particuliers

Nous présentons dans ce paragraphe deux types particuliers d'automates que l'on utilise dans les algorithmes d'inférence par fusion d'états.

**L'arbre accepteur des préfixes (PTA)** est un SDFA dont la structure sous-jacente est un arbre et qui représente un apprentissage par coeur de l'ensemble d'apprentissage $EA$. Nous le notons $PTA_{EA}$. Dans ce cas, la fonction $\gamma$ est calculée en utilisant les fréquences d'apparition d'un symbole relativement à son préfixe. Pour exemple la figure 1 nous montre $PTA_{EA}$ avec
EA = {$\lambda$, aac, aac, abc, aac, aac, abc, abc, $\lambda$, a, ab }.

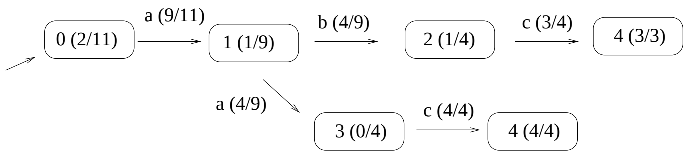


FIG. 1 – *Un arbre accepteur des préfixes*

La représentation graphique adoptée pour les automates est la suivante : chaque état de l'automate est étiqueté par son numéro et par la probabilité (représentée entre parenthèses) d'arrêter le processus de génération. La probabilité de générer un symbole de $\Sigma$ est représentée par une transition étiquetée par ce symbole, sa valeur est définie entre parenthèses. Par convention l'état de numéro 0 est l'état initial.

**Automate quotient :** A partir d'un SDFA $A$, on peut réaliser des fusions d'états. L'automate résultant peut devenir non déterministe[1]. On fusionne alors les états qui créent ce non déterminisme. On obtient alors un SDFA $A'$ déterministe. On a la propriété suivante : $L_A \subseteq L_{A'}$. On note alors $A' = A/\pi_{AA'}$.

On dira que $A'$ dérive de $A$ via la partition $\pi_{AA'}$ ;

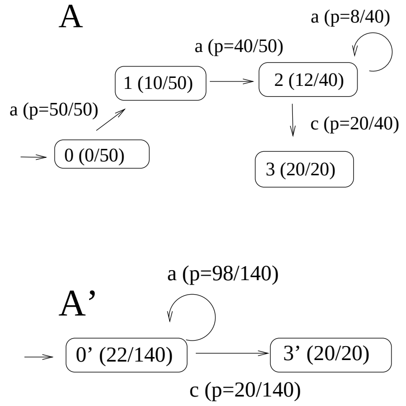


FIG. 2 – *Automate quotient*

Sur la figure 2, l'automate $A'$ dérive de l'automate $A$ via la partition $\pi_{AA'} = \{\{0, 1, 2\}, \{3\}\}$. On a $\pi_{AA'}(0') = \{0, 1, 2\}$, $\pi_{AA'}(3') = \{3\}$.

On note $B_{AA'}(q_i)$ l'état de $A'$ correspondant à l'état $q_i$ de $A$ relativement à $\pi_{AA'}$. Plus formellement, $B_{AA'}(q_i) = q'_j \in Q_{A'}$ tel que $q_i \in \pi_{AA'}(q'_j)$.
Sur la figure 2, $B_{AA'}(0) = B_{AA'}(1) = B_{AA'}(2) = 0'$. On dit que $0'$ est le bloc de l'état 0 (respectivement de 1 et de 2).

Lors de la fusion, la fonction de probabilité $\gamma$ est mise à jour par manipulation des fréquences de passage pour chaque transition et pour chaque état fusionné.

Si l'on note $freq_A(q_i)$ (respectivement $freq_A(q_i, a)$) la fréquence de passage sur l'état $q_i$ de l'automate $A$ (respectivement de la transition sortante de $q_i$ étiquetée par la lettre $a$), la mise à jour de la fonction de probabilités s'écrit pour chaque état $q'_i$ de $A'$ :
$\gamma_{A'}(q'_i, a) = \sum_{q_i \in B_{AA'}(q_i)} freq_A(q_i, a) \ / \sum_{q_i \in B_{AA'}(q_i)} freq_A(q_i)$

L'automate ainsi généré définit toujours une distribution de probabilités sur $\Sigma^*$.

Sur l'exemple de la figure 2 la transition sortante de $0'$ étiquetée par la lettre "a" aura pour probabilité 0.7 ce qui correspond à $(50 + 40 + 8)/(50 + 50 + 40)$.

1. Un automate non déterministe est un automate dans lequel il existe au moins un état qui a plusieurs transitions sortantes étiquetées par un même symbole.

### 2.3 L'entropie relative

Nous présentons ici la définition de l'entropie relative ainsi que son interprétation en terme de théorie de l'information.

Soit $P_C$ et $P_A$ deux distributions de probabilités sur $\Sigma^*$. Soit $P_C(x)$ (respectivement $P_A(x)$) la probabilité de $x$ relativement à la distribution $P_C$ (respectivement $P_A$). La divergence de Kullback-Leibler s'écrit :

$$D(P_C||P_A) \stackrel{def}{=} \sum_{x \in \Sigma^*} P_C(x) \ \log \frac{P_C(x)}{P_A(x)} \qquad (2)$$

Nous allons maintenant présenter une analyse informelle de cette quantité. La définition 2 peut se réécrire :

$$\begin{aligned} D(P_C||P_A) &= \underbrace{- \sum_{x \in \Sigma^*} P_C(x) \ \log(P_A(x))}_{\text{Entropie croisée entre } P_A \text{ et } P_C} \\ &\quad - \underbrace{\left[ - \sum_{x \in \Sigma^*} P_C(x) \ \log(P_C(x)) \right]}_{H_{P_C}} \quad (3) \end{aligned}$$

Si $\Sigma^*$ est distribué selon $P_C$ et que le logarithme est pris en base 2, la théorie de l'information nous dit que le terme $H_{P_C}$ est la taille moyenne, en nombre de bits, du codage optimum d'une chaîne de $\Sigma^*$ tirée conformément à la distribution $P_C$. On pondère la taille du codage de chaque $x$ (terme $log(P_C(x))$) par sa probabilité d'apparition conformément à la distribution $P_C$. On suppose dans la suite que la base du log est 2.

Supposons $\Sigma^*$ distribué selon $P_C$. Si nous estimons $P_C$ par la distribution $P_A$, le code d'une chaîne $x$ de $\Sigma^*$ sera de longueur $log(P_A(x))$. Mais la probabilité réelle d'apparition de $x$ est $P_C(x)$. La taille moyenne du codage d'une chaîne $x$ de $\Sigma^*$ que nous allons proposer sera donc $P_C(x) \ log(P_A(x))$. En d'autres termes, l'équation 3 nous permet de voir la divergence comme une mesure du nombre de bits additionnels pour coder des chaînes distribuées selon $P_C$ lorsque l'on choisi un code optimal selon $P_A$.

L'entropie relative peut donc être utilisée comme critère pour estimer une distance entre une distribution et son estimation. L'équation 3 nous permet de déduire que plus la valeur de la divergence est faible, plus grande est l'adéquation entre la distribution cible (représentée par $P_C$) et son estimation (représentée par $P_A$).

# 3 Motivations

Les algorithmes d'inférence d'automates probabilistes par fusion d'états construisent d'abord l'arbre des préfixes. Ils le généralisent ensuite en fusionnant certains de ses états. L'algorithme 1 donne un schéma général de ces algorithmes :

```
INFÉRENCE GRAMMATICALE PROBABILISTE PAR FUSION D'ÉTATS

Données : Un ensemble d'apprentissage EA, un réel α
Résultat : Un automate stochastique
début
    Construire_PTA (EA) ;
    A ← PTA ;
    tant que états à fusionner faire
        (q_i,q_j) ← Choisir_Une_Paire_d_Etat (A) ;
        si Distance (A,q_i,q_j) < α alors
            A ← Fusion (A,q_i,q_j) ;
        fin
    fin
    Retourner A ;
fin
```

**Algorithme 1:** Algorithme générique d'inférence probabiliste

La fonction `Choisir_Une_Paire_d_Etat` est la fonction qui désigne les états candidats à une fusion. Dans l'algorithme *Alergia* [1] les états du PTA sont numérotés selon un parcours en largeur. Chaque état est comparé avec les états de rang inférieur selon cet ordre. Ron [6] propose une exploration par niveau du PTA. Chaque état est comparé avec les états du même niveau. Le récent concours d'inférence grammaticale non stochastique *Abbadingo* ([4]) a permis de mettre au point une heuristique de choix d'états à fusionner. Cette heuristique améliore l'algorithme *RPNI* [5] qui est la version non stochastique de *Alergia*. L'ordre des fusions est adapté dynamiquement en fonction des données d'apprentissage.

La fonction `Distance` est un test statistique qui permet de décider s'il faut fusionner deux états. Le paramètre $\alpha$ permet d'être plus ou moins tolérant sur ce test. Les algorithmes précités utilisent un critère local de similarité entre états basé sur l'inégalité de Hoeffding [3]. Notre but est de proposer un critère global capable, pour chaque automate intermédiaire, d'en estimer la qualité. La mesure que nous nous proposons d'utiliser ici est l'entropie relative entre la distribution engendrée par le PTA et celle engendrée par l'un de ses automates quotient. Plus précisément, nous verrons comment la fusion de deux états modifie la divergence entre la distribution associée à l'automate résultant de cette fusion et celle engendrée par le PTA. Notons que Ron a montré le lien entre une mesure de similarité locale entre états et la notion de divergence, dans le cas particulier d'automates acycliques [6]. Nous nous intéressons ici à la mesure de divergence entre deux automates contenant éventuellement des cycles.

# 4 Propriétés du calcul de la divergence

Nous allons montrer dans un premier temps la technique que Carrasco [2] propose pour calculer l'entropie relative entre deux automates. Nous verrons les problèmes que pose ce calcul et exhiberons des cas particuliers dans lesquels la divergence prend toujours une valeur finie d'une part et est calculable efficacement d'autre part.

## 4.1 La proposition de Carrasco

Carrasco calcule la divergence entre deux distributions de probabilités représentées par des SDFA en décomposant $\Sigma^*$ symbole par symbole. Les langages réguliers pouvant comporter de la récursivité, le calcul de la divergence impose, dans le cadre général, la sommation de probabilités sur des ensembles infinis. L'auteur propose une technique itérative pour approximer le calcul de cette somme.

Nous présentons maintenant le calcul de Carrasco.

Soient $L$ et $L'$ deux langages engendrés respectivement par les SDFA $A = (\Sigma, Q, q_I, \delta, \gamma)$ et $A' = (\Sigma, Q', q'_I, \delta', \gamma')$.

La décomposition de $\Sigma^*$ symbole par symbole permet de réécrire la divergence comme :

$$D(A||A') = \sum_{q_i \in Q} \sum_{q'_j \in Q'} \sum_{a \in \Sigma \cup \{\lambda\}} c_{ij} P_A(a|q_i) \log \frac{P_A(a|q_i)}{P_{A'}(a|q'_j)} \tag{4}$$

où $c_{ij}$ est la somme des probabilités des préfixes de $A$ commun à $A'$ et menant respectivement à $q_i$ dans $A$ et $q'_j$ dans $A'$. Plus formellement, $c_{ij} = \sum_{x \in L_{ij}} P_A(x\Sigma^*)$ avec

$$\begin{aligned} L_{ij} &= Pref_A(q_i) \bigcap Pref_{A'}(q'_j) \qquad (5)\\ &= \{x \in \Sigma^* : \delta(q_I, x) = q_i \wedge \delta'(q'_I, x) = q'_j\} \end{aligned}$$

Sur la figure 3 nous avons, $A' = A/\pi_{AA'}$ avec $\pi = \{\{0,1,2\},\{3\}\}$, $L_{33'} = aa^+c$ et $c_{33'} = \sum_{x \in L_{ij}} P_A(x\Sigma^*) = \sum_{n \geq 0} \frac{50}{50} \times \frac{40}{50} \times \frac{8}{40}^n \times \frac{20}{40}$.

Notons deux propriétés pour $A$ et $A'$ quelconques :

- D'une part si $P_A(a|q_i) \neq 0$ et $P_{A'}(a|q'_j) = 0$ la valeur de l'expression $D(A||A')$ devient infinie. La divergence ne peut donc pas être utilisée directement pour comparer deux états d'un automate. Elle prendrait régulièrement une valeur infinie non significative.
- D'autre part, la taille de l'ensemble $L_{ij}$ devient infinie si $A$ et $A'$ possèdent un circuit en commun (Dans l'exemple de la figure 3, $L_{33'} = aa^+c$). Le calcul des $c_{ij}$ résulte d'une somme infinie de termes. Carrasco propose un calcul approché itératif des $c_{ij}$ : il stoppe la récursivité à un $n$ fixé.

Nous allons présenter des cas particuliers de relations entre automates qui permettent un calcul efficace de la divergence sans présenter les problèmes mentionnés ci dessus. Ces cas particuliers correspondent aux automates obtenus lors de l'application d'un algorithme d'inférence par fusion.

## 4.2 Divergence entre un automate et ses automates quotients

Nous présentons dans ce paragraphe le calcul de la divergence entre l'automate $A$ et l'automate $A'$ dans le cas particulier où l'automate $A'$ est quotient de l'automate $A$. Nous montrons que dans ce cas la divergence $D(A||A')$ ne peut être infinie.

Soit $A = (\Sigma, Q, q_I, \delta, \gamma)$ et $A' = (\Sigma, Q', q'_I, \delta', \gamma')$ tels qu'il existe une partition $\pi_{AA'}$ des états de $A$ telle que $A' = A/\pi_{AA'}$.

Notre but est de montrer que $c_{ij}$ est non nul si et seulement si l'état $q'_j$ de l'automate $A'$ correspond au bloc de l'état $q_i$ de l'automate $A$ (proposition 1). Nous allons dans un premier temps montrer la propriété suivante :

**Lemme 1**

$$Pref_{A'}(q'_j) \supseteq \Big( \bigcup_{\substack{q_k \in Q_A \\ q_k \in \pi_{AA'}(q'_j)}} Pref_A(q_k) \Big)$$

**preuve :**

L'automate $A$ (respectivement $A'$) est déterministe. Chaque mot de $L_A$ (respectivement $L_{A'}$) est reconnu via un unique chemin. Puisque il existe une partition $\pi_{AA'}$, on a $L_{A'} \supseteq L_A$. D'où $Pref_{A'}(q'_j) \supseteq Pref_A(q_k)$ pour chaque état $q_k$ de $\pi_{AA'}(q'_j)$.

D'où

$$Pref_{A'}(q'_j) \supseteq \Big( \bigcup_{\substack{q_k \in Q_A \\ \pi_{AA'}(q_k) = q'_j}} Pref_A(q_k) \Big)$$

□

**Lemme 2**

$c_{ij} = 0 \Leftrightarrow Pref_A(q_i) \bigcap Pref_{A'}(q'_j) = \emptyset$

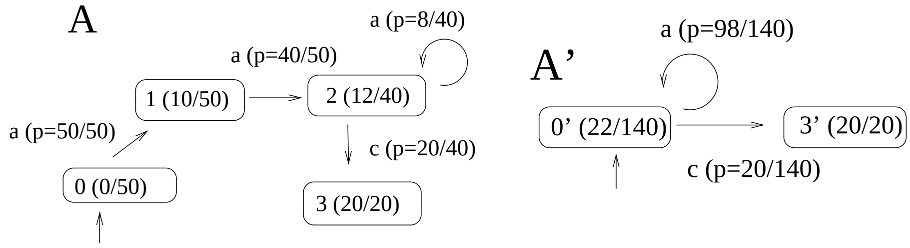


FIG. 3 – *automates exemples*

**preuve :**

$\Rightarrow$

On a $c_{ij} = 0$, d'où

$$c_{ij} = \sum_{\substack{x \in \Sigma^*, \\ \delta(q_I, x) = q_i, \\ \delta(q'_I, x) = q'_j}} P(x\Sigma^*) = \sum_{x \in Pref_A(q_i) \cap Pref_{A'}(q'_j)} P(x\Sigma^*) = 0$$

En tant que somme de probabilités non nulles, $P(x\Sigma^*) > 0$.

La somme $\sum_{x \in Pref_A(q_i) \cap Pref_{A'}(q'_j)} P(x\Sigma^*) = 0$ est nulle si elle ne s'applique sur aucun terme.

D'où $Pref_A(q_i) \bigcap Pref_{A'}(q'_j) = \emptyset$

$\Leftarrow$

Réciproquement, si $Pref_A(q_i) \bigcap Pref_{A'}(q'_j) = \emptyset$ alors $L_{ij} = \emptyset$ et donc $c_{ij} = 0$

$\square$

**Proposition 1**
$c_{ij} \neq 0 \Leftrightarrow B_{AA'}(q_i) = q'_j$

**preuve :**

$\Rightarrow$

contraposée : montrons que $B_{AA'}(q_i) \neq q'_j \Rightarrow c_{ij} = 0$

Si $B_{AA'}(q_i) \neq q'_j$ alors, $q_i \neq q_k \ \forall q_k \in \pi_{AA'}(q'_j)$.

Les automates étant deterministes, on a $Pref_{A'}(q'_j) \bigcap Pref_A(q_i) = \emptyset$.

Par le lemme 2 on a $c_{ij} = 0$.

$\Leftarrow$

Si $B_{AA'}(q_i) = q'_j$ alors $Pref_{A'}(q'_j) \supseteq Pref_A(q_i)$ (par le lemme 1).

On a donc $Pref_{A'}(q'_j) \bigcap Pref_A(q_i) \neq \emptyset \overset{lemme2}{\Rightarrow} c_{ij} \neq \emptyset$

$\square$

Dans la mesure où $c_{ij}$ est non nul pour un seul état $q'_j = B_{AA'}(q_i)$, nous noterons $c_{ij}$ par $c_{iB_{AA'}(q_i)}$.

**Corollaire 1**
*Si $c_{ij} \neq 0$ alors*
$P_A(a|q_i) \neq 0 \Rightarrow P_{A'}(a|q'_j) \neq 0$

**preuve :**

Si $c_{ij} \neq 0$ alors (proposition 1) $q'_j = B_{AA'}(q_i)$ et donc si $P_A(a|q_i) \neq 0$ alors $P_{A'}(a|B_{AA'}(q_i)) \neq 0$ et donc $P_{A'}(a|q'_j) \neq 0$.

$\square$

**Corollaire 2**
*Soient deux automates $A$ et $A'$ et $\pi_{AA'}$ tel que $A' = A/\pi_{AA'}$ (en posant $0 \log 0 = 0$)*

$$D(A||A') < \infty$$

**preuve :**

S'il existe une partition $\pi_{AA'}$ telle que $A' = A/\pi_{AA'}$ alors, $\forall\ q_i \in Q$, $\exists\ q'_j \in Q'$ tel que $B_{AA'}(q_i) = q'_j$ et donc, par la proposition 1, $c_{ij} \neq 0$ pout tout état $q_i$ de $A$ et tout état $q'_j$ de $A'$. Par le corollaire 1 on a donc que $P_{A'}(a|q'_j)$ est toujours défini lorsque $P_A(a|q_i) \neq 0$.

$\square$

Le corollaire 2 nous permet de dire que l'entropie relative est toujours définie entre un automate et ses automates quotients. Le problème de son calcul se pose maintenant. Si l'automate de référence est quelconque, on peut appliquer la technique d'approximation itérative de Carrasco. Si par contre il a une structure d'arbre, on a la propriété suivante :

**Proposition 2**
*Si l'automate $A$ est un arbre, $|Pref_A(q_i)| < \infty\ \forall q_i \in Q_A$*

**preuve :**

Evident : dans le cas d'un arbre, $|Pref_A(q_i)| = 1\ \forall q_i \in Q_A$.

$\square$

La conséquence de la proposition 2 est que si $A$ est un arbre, alors, pour chaque $q_i \in Q_A$ il existe un unique chemin menant de $q_I$ à $q_i$ ; le calcul des $c_{ij}$ pourra se faire par un simple parcours dans l'arbre $A$.

Avec les notations précédentes, l'équation de la divergence (équation 4) peut se réécrire :

$$D(A||A') = \sum_{q_i \in Q} \sum_{a \in \Sigma \cup \{\lambda\}} c_i P_A(a|q_i) \log \frac{P_A(a|q_i)}{P_{A'}(a|B_{AA'}(q_i))} \tag{6}$$

Avec $c_i = P_A(x_i)$ où $x_i$ est l'unique préfixe $x$ de $Pref_A(q_i)$ (*i.e.* tel que $\delta_A(q_I, x) = q_i$).

L'équation 6 se réécrit :

$$\begin{aligned} D(A||A') &= \sum_{q_i \in Q} \sum_{a \in \Sigma \cup \{\lambda\}} c_i P_A(a|q_i) \log P_A(a|q_i) \\ &- \sum_{q_i \in Q} \sum_{a \in \Sigma \cup \{\lambda\}} c_i P_A(a|q_i) \log P_{A'}(a|B_{AA'}(q_i)) \end{aligned} \tag{7}$$

Le premier terme ne dépend que de l'automate $A$, Nous le noterons $K_A$. En remarquant que $c_i \times P_A(a|q_i) = c_{\delta_A(q_i,a)}$ l'équation 7 devient :

$$\begin{aligned} D(A, A') &= K_A \\ &- \sum_{q_i \in Q} \sum_{a \in \Sigma \cup \{\lambda\}} c_{\delta_A(q_i,a)} \log P_{A'}(a|B_{AA'}(q_i)) \end{aligned} \tag{8}$$

avec

$$\begin{aligned} K_A &= \sum_{q_i \in Q} \sum_{a \in \Sigma \cup \{\lambda\}} c_{\delta_A(q_i,a)} \log P_A(a|q_i) \\ &\text{et} \\ c_{\delta(q_i,\lambda)} &= c_i \times P(\lambda|q_i) \end{aligned}$$

Sur la figure 3, on a :

$$\begin{aligned} D(A||A') &= K_A \\ &- \underbrace{c_0.\frac{0}{50}.\log(22/140)}_{\text{Etat 0, a=}\lambda} - \underbrace{c_0.\frac{50}{50}.\log(98/140)}_{\text{Etat 0, a='a'}} \\ &- \underbrace{c_1.\frac{10}{50}.\log(22/140)}_{\text{Etat 1, a=}\lambda} - \underbrace{c_1.\frac{40}{50}.\log(98/140)}_{\text{Etat 1, a='a'}} \\ &- \underbrace{c_2.\frac{12}{40}.\log(22/140)}_{\text{Etat 2, a=}\lambda} - \underbrace{c_2.\frac{8}{40}.\log(98/140)}_{\text{Etat 2, a='a'}} \\ &- \underbrace{c_2.\frac{20}{40}.\log(20/140)}_{\text{Etat 2, a='c'}} - \underbrace{c_3.\frac{20}{20}.\log(20/20)}_{\text{Etat 3, a=}\lambda} \end{aligned} \tag{9}$$

En conclusion, l'entropie relative entre un automate dont la structure est un arbre et ses automates quotients est non seulement toujours définie mais est de plus calculable efficacement (linéairement en fonction de la taille des automates).

Nous allons maintenant voir comment un calcul itératif de la divergence peut se faire lorsque l'on considère différents automates quotients du PTA et quotients entre eux.

### 4.3 Calcul itératif de la divergence

Soit $A_0$ un automate dont la structure d'automate est un arbre. $\pi_{01}$ une partition des états de $A_0$, $A_1$ l'automate tel que $A_1 = A/\pi_{01}$. Soit $\pi_{12}$ une partition des états de $A_1$. On a $\pi_{02} = \pi_{01} \circ \pi_{12}$. L'équation 8 nous permet d'écrire :

$$\begin{aligned} D(A_0||A_1) &= K_{A_0} \\ &- \sum_{q_i \in Q_0} \sum_{a \in \Sigma \cup \{\lambda\}} c_{\delta_{A_0}(q_i,a)} \log P_{A_1}(a|B_{01}(q_i)) \\ D(A_0||A_2) &= K_{A_0} \\ &- \sum_{q_i \in Q_0} \sum_{a \in \Sigma \cup \{\lambda\}} c_{\delta_{A_0}(q_i,a)} \log P_{A_2}(a|B_{02}(q_i)) \end{aligned}$$

D'où :

$$\begin{aligned} D(A_0||A_2) = D(A_0||A_1) \\ - \sum_{a \in \Sigma \cup \{\lambda\}} \sum_{q_i \in Q^{A_0}_{\pi_{01} \oplus \pi_{02}}} c_{\delta_{A_0}(q_i,a)} \log P_{A_2}(a|B_{02}(q_i)) \\ + \sum_{a \in \Sigma \cup \{\lambda\}} \sum_{q_i \in Q^{A_0}_{\pi_{01} \oplus \pi_{02}}} c_{\delta_{A_0}(q_i,a)} \log P_{A_1}(a|B_{01}(q_i)) \end{aligned} \tag{10}$$

Où $Q^{A_0}_{\pi_{01} \oplus \pi_{02}}$ est l'ensemble des états de $A_0$ qui sont intervenus dans le passage de l'automate $A_1$ à l'automate $A_2$. Plus formellement on a :

$$Q^{A_0}_{\pi_{01} \oplus \pi_{02}} = \{q_i \in Q_0 | \pi_{01}(B_{01}(q_i)) \neq \pi_{02}(B_{02}(q_i))\}$$

Sur la figure 3 on a $Q^{A_0}_{\pi_{01} \oplus \pi_{02}} = \{0, 1, 2\}$.

La formule 10 se réécrit finalement :

$$\begin{aligned} D(A_0||A_2) = D(A_0||A_1) \\ + \sum_{a \in \Sigma \cup \{\lambda\}} \sum_{q_i \in Q^{A_0}_{\pi_{01} \oplus \pi_{02}}} c_{\delta_{A_0}(q_i,a)} \big(\log P_{A_1}(a|B_{01}(q_i)) \\ - \log P_{A_2}(a|B_{02}(q_i))\big) \end{aligned} \tag{11}$$

En conclusion, l'équation 11 nous permet de calculer efficacement l'accroissement de divergence en ne considérant que les états qui sont intervenus dans le passage de l'automate $A_1$ à l'automate $A_2$.

## 5 Vers un algorithme d'inférence

Nous rappelons que l'algorithme générique d'inférence probabiliste (algorithme 1 section 3) est caractérisé par deux paramètres : la méthode permettant de choisir deux états candidats à une fusion (réalisée par la fonction *Choisir_Une_Paire_d_Etat*) et celle qui décide si deux états doivent être fusionnés (réalisée par la fonction *Distance*).

Nous allons maintenant présenter comment utiliser les calculs précédents pour implémenter une fonction *Distance* basée sur la variation de la divergence.

Soit $A_0$ un PTA, $A_1$ un de ses automates dérivés et $D(A_0||A_1)$ leur divergence. Soit maintenant deux états de $A_0$ (choisis par la fonction *Choisir_Une_Paire_d_Etats*). La fusion de ces états définit une partition des états de $A_0$ (que nous notons $\pi_{A_0A_1}$). Le calcul de la divergence que nous avons présenté nous permet alors de calculer la divergence entre $A_0$ (l'arbre des préfixes) et $A_1 = A_0/\pi_{A_0A_1}$.

L'automate qui a une divergence minimale par rapport à l'arbre des préfixes est cet arbre lui-même puisque dans ce cas la divergence est nulle. Nous nous proposons d'utiliser l'accroissement de divergence comme critère pour choisir les fusions qui permettent de s'éloigner le moins possible (en terme de divergence) du PTA. Cette idée consiste à généraliser les données d'apprentissage, représentées par le PTA, seulement si l'automate résultant ne présente pas une divergence trop importante avec le PTA. Pour ce faire, nous introduisons un seuil $\alpha$ sur cette divergence et nous ne permettons la fusion que si la divergence entre le PTA et l'automate après fusion est inférieur au seuil. Ce seuil va donc nous permettre de contrôler le taux de généralisation de l'automate inféré.

## 6 Conclusions et perspectives

Nous avons proposé ici un calcul de la divergence entre deux distributions représentées par des SDFA. Nous avons montré dans un premier temps les limitations dans le cas général de ce calcul. Nous avons montré par ailleurs que la divergence est toujours définie et calculable efficacement quand elle est calculée entre un automate dont la structure est un arbre et un de ses automates quotients. Nous en avons enfin donné une technique de calcul incrémental.

Les prochains travaux vont d'abord consister en la réalisation d'expérimentations pour voir ce qu'apporte l'utilisation de la divergence comme critère de similarité entre états dans un algorithme d'inférence par fusions. Ces expérimentations permettront de plus de définir la fourchette de validité du paramètre $\alpha$ et éventuellement de déterminer dynamiquement et automatiquement sa valeur en fonction des données d'apprentissage. D'autre part nous envisageons d'établir les propriétés théoriques de convergence vers un automate cible en fonction par exemple du paramètre $\alpha$.

## Références


[1] R. Carrasco et J. Oncina. Learning stochastic reagular grammars by means of a state merging method. *ICGI-94*, Lecture Notes in Artificial Intelligence 862:139–152, 1994. Subseries of Lectures Notes in Computer Science.

[2] R. C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, vol. 31(num. 5):p 437–444, 1997.

[3] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[4] K. J. Lang, B. A. Pearlmutter et R. A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *Lecture Notes in Computer Science*, 1433:1–12, 1998.

[5] J. Oncina et P. García. Identifying regular languages in polynomial time. In H. Bunke, éditeur, *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 1992.

[6] D. Ron, Y. Singer et N. Tishby. On the learnability and usage of acyclic probabilistic automata. In *Computational Learning Theory, COLT'95*, pages 31–40, Santa Cruz, CA, USA, 1995.

[7] A. Stolke et S. Omohundro. Inducting probabilistic grammars by bayesian model merging. In Lecture Notes in Artifitial Intelligence, éditeur, *Proceedings of International Colloquium on Gramatical Inference*, number 862, pages 106–118. ICGI-94, 1994.