

TP 10

HMM POS TAGGING

ET L'ALGORITHME DE VITERBI

G. A. Musillo
musillo4@etu.unige.ch

23 janvier 2007

Tagger ou étiquetter une phrase, c'est assigner à chacun des mots qui la constitue sa partie du discours ou sa catégorie (en anglais, *part-of-speech tag*). Cette tâche d'étiquetage n'est pas triviale, car les mots d'une langue naturelle sont massivement ambigus et peuvent appartenir à plus d'une catégorie. Par exemple, en anglais, le mot *like* peut appartenir non seulement à la catégorie des prépositions, qu'on note *IN*, mais également à la catégorie des formes verbales qu'on note *VB*. Il résulte de telles ambiguïtés catégorielles qu'une phrase peut être étiquetée de plus d'une façon. Considérez, par exemple, la phrase anglaise *time flies like an arrow*. Il est possible d'assigner au moins deux étiquetages à cette phrase :

- i) *time/NN flies/VB like/IN an/DET arrow/NN*
- ii) *time/NN flies/NN like/VB an/DET arrow/NN*

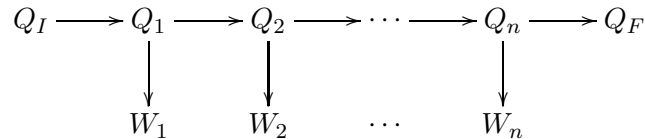
Selon le premier étiquetage, le mot *like* est une préposition et le mot *flies* un verbe. Selon le deuxième étiquetage, le mot *like* est un verbe et le mot *flies* un nom.

Afin de désambiguïser l'étiquetage d'une phrase telle que *time flies like an arrow*, on peut définir un modèle stochastique du tagging qui permet de calculer, étant donné une phrase, la séquence de tags la plus probable de cette phrase. Ce modèle est appelé *modèle caché de Markov* (en anglais, *h(idden) M(arkov) m(odell)*). Ce modèle, efficace et performant, décrit un processus qui génère les mots d'une phrase donnée ainsi que leurs tags respectifs. Pour des raisons théoriques qui ne seront pas justifiées ici, on stipule que chaque phrase taggée est précédée du tag initial est Q_I et suivie du tag final est Q_F .

Voici comment la phrase étiquetée Q_I *time/NN flies/VB like/IN an/DET arrow/NN* Q_F est générée par un processus de Markov caché. Initialement, le processus est dans l'état Q_I . Connaissant l'état initial, le tag *NN* du premier mot est généré aléatoirement. Le premier mot *time* est alors généré

aléatoirement étant donné son tag NN . C'est ensuite le tag VB en deuxième position qui est généré sur la base du tag précédent. Le deuxième mot *flies* est alors généré aléatoirement étant donné son tag VB . Un tel processus se termine après que tous les mots et tous les tags d'une phrase ont été générés de cette manière, c'est-à-dire après que l'état final Q_F a été visité.

Plus généralement, on peut représenter un tel processus par le modèle graphique suivant :



La séquence $W_1 W_2 \cdots W_n$ représente les éléments de la phrase et la séquence $Q_I Q_1 Q_2 \cdots Q_n Q_F$ décrit les tags associés aux mots de la phrase donnée. Les arêtes dirigées qui relient deux sommets de ce modèle graphique indiquent une relation de dépendance. Ainsi, les arêtes qui relient un tag de départ à un tag d'arrivée indiquent que la probabilité d'un tag ne dépend que de la probabilité du tag qui le précède. Les arêtes qui relient un tag Q_i à un mot W_i indiquent que la probabilité d'un mot ne dépend que de son tag. Un modèle de Markov caché stipule donc deux hypothèses d'indépendance : la génération aléatoire d'un tag n'est basée que sur la connaissance du tag qui le précède et la génération aléatoire d'un mot n'est basée que sur la connaissance du tag qui lui est assigné. Plus formellement, dans un tel modèle, la probabilité jointe d'une phrase de n mots et de ses tags est factorisée de la façon suivante :

$$\begin{aligned}
 P(Q_I, Q_1, W_1, \dots, Q_n, W_n, Q_F) &= P(Q_1|Q_I)P(W_1|Q_1) \\
 &\times \left(\prod_{t=1}^{n-1} P(Q_{t+1}|Q_t)P(W_{t+1}|Q_{t+1}) \right) \\
 &\times P(Q_F|Q_n)
 \end{aligned}$$

Notez que chacun des facteurs du produit est une probabilité conditionnelle qui correspond à une arête du modèle graphique.

Pour spécifier un modèle de Markov caché, il est suffisant de spécifier quatre composantes :

- un ensemble fini Q d'états qui contient les états Q_I et Q_F et quelques tags
- un ensemble fini W de mots
- un ensemble de probabilités $t_{i,j}$ qui, pour chaque paire ordonnée d'états i et j , définit la probabilité conditionnelle de l'état j étant donné l'état précédent i
- un ensemble de probabilités $e_i(w)$ qui, pour chaque paire d'états i et de mots w , définit la probabilité conditionnelle du mot w connaissant son tag qui est l'état i

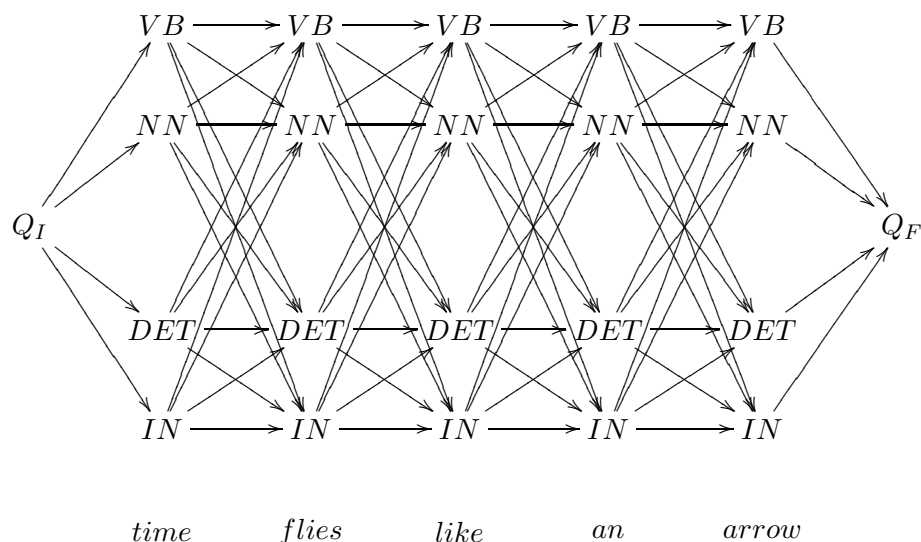
Considérez le modèle spécifié ci-dessous :

- $Q = \{Q_I, Q_F, VB, NN, DET, IN\}$
- $W = \{time, flies, like, an, arrow\}$

$t_{i,j}$	Q_I	VB	NN	DET	IN	Q_F
Q_I	0	.1	.3	.2	.4	0
VB	0	.01	.3	.3	.3	.09
- NN	0	.25	.15	.0091	.3909	.2
DET	0	.0003	.9985	.0005	.0005	.0002
IN	0	.1	.49	.399	.01	.001
Q_F	0	0	0	0	0	0
$e_i(w)$	$time$	$flies$	$like$	an	$arrow$	
VB	.001	.45	.545	.001	.003	
- NN	.31	.19	0.103	.007	.39	
DET	.075	.075	.075	.97	.075	
IN	.075	.075	.97	.075	.075	

Ce modèle génère 4^5 séquences de tags pour la phrase donnée *time flies like*

an arrow. Le treillis suivant représente toutes ces séquences possibles :



Plus généralement, si l'ensemble Q contient r tags, alors un modèle de Markov caché générera, pour une phrase de longueur n , r^n étiquetages possibles.

L'algorithme de Viterbi permet de déterminer, pour une phrase donnée, lequel de ces étiquetages est le plus probable. Il calcule donc un chemin optimal dans le treillis des étiquetages possibles. Plus formellement, l'algorithme de Viterbi calcule pour une phrase W_1, \dots, W_n de n mots

$$\max_{Q_1, \dots, Q_n} P(Q_I, Q_1, \dots, Q_n, Q_F | W_1, \dots, W_n) = \max_{Q_1, \dots, Q_n} P(Q_I, Q_1, \dots, Q_n, Q_F, W_1, \dots, W_n)$$

L'algorithme de Viterbi n'est pas un algorithme banal qui énumère chacune des r^n séquences de tags possibles et retourne la séquence dont la probabilité est la plus grande. Cet algorithme subtil est basé sur la programmation dynamique et exploite les hypothèses d'indépendance stipulées par un modèle de Markov caché afin de calculer efficacement l'étiquetage le plus probable.

Dénotons par $\delta(i, l)$ la séquence de tags la plus probable Q_I, Q_1, \dots, Q_l qui génère le préfixe W_1, \dots, W_l de la phrase donnée en entrée et dont le dernier tag Q_l vaut i . Si on parvenait à calculer, pour une phrase de longueur n , $\delta(i, n)$ pour chacun des tags i possibles, alors on aurait résolu le problème de l'étiquetage le plus probable. En effet, si $\delta(i, n)$ est connu pour tout i , alors l'étiquetage le plus probable de la phrase W_1, \dots, W_n n'est rien d'autre que

$$\max_i \delta(i, n) P(Q_F | i)$$

Intuitivement, pour calculer une valeur de $\delta(i, l)$, il suffit de calculer l'état k qui maximise le produit de

**la probabilité d'une transition de l'état k vers l'état i (i.e. $t_{k,i}$)
et $\delta(k, l - 1)$, autrement dit la séquence de tags la plus probable
qui a généré W_1, \dots, W_{l-1} dont le tag Q_{l-1} est k .**

Connaissant $\delta(k, l - 1)$ pour tous les tags k , on peut calculer le tag k qui maximise le produit $\delta(k, l - 1) \times t_{k,i}$. Il suffit donc de calculer $\delta(k, l - 1)$ pour déterminer la valeur de $\delta(i, l)$.

Plus formellement, $\delta(i, l)$ est défini par les clauses de base et d'induction suivantes :

$$\begin{aligned}\delta(i, 1) &= t_{Q_1, i} \times e_i(W_1) \\ \delta(i, t + 1) &= \left(\max_{k \in Q} \delta(k, t) \times t_{k, i} \right) \times e_i(W_{t+1})\end{aligned}$$

Cette définition récursive est implémentée par le pseudo-code suivant :

```

input : un ensemble de  $K$  tags, un ensemble de probabilités de
          transition  $t_{i,j}$ , un ensemble de probabilités d'émission  $e_i(w)$ ,
          et une phrase  $W_1, \dots, W_n$ 
output: un ensemble de variables  $\delta(i, t)$ 
for  $i \leftarrow 1$  to  $K$  do
  |  $\delta(i, 1) \leftarrow t_{Q_1, i} \times e_i(W_1)$ 
end
for  $t \leftarrow 1$  to  $n - 1$  do
  | for  $i \leftarrow 1$  to  $K$  do
  | | for  $k \leftarrow 1$  to  $K$  do
  | | |  $\delta \leftarrow \delta(k, t) \times t_{k, i} \times e_i(W_{t+1})$ 
  | | | if  $\delta > \delta(i, t + 1)$  then
  | | | |  $\delta(i, t + 1) \leftarrow \delta$ 
  | | | end
  | | end
  | end
end

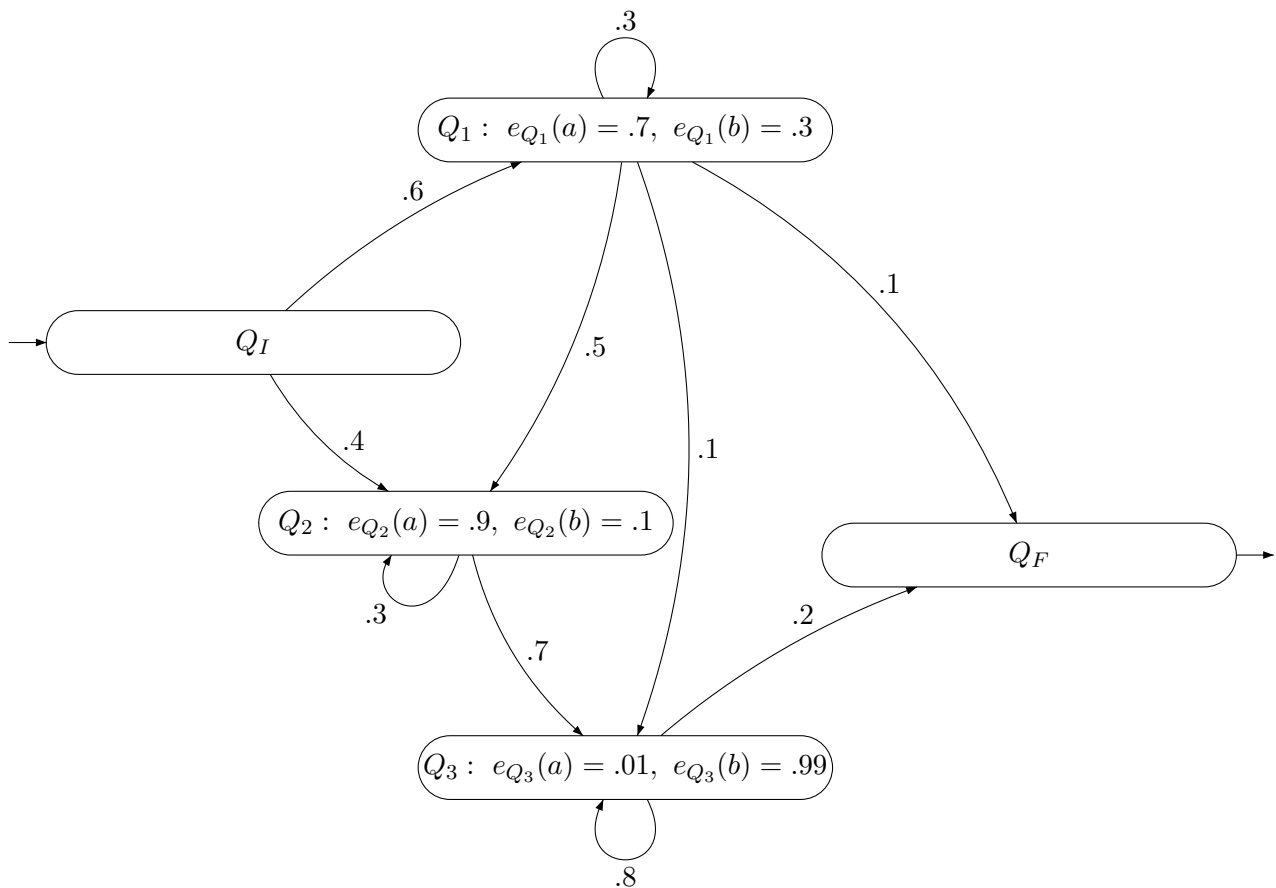
```

Remarquez que ce code ne retourne pas explicitement la séquence de tags la plus probable, il ne fait que calculer la probabilité de cette séquence. De plus, ce code ne calcule pas la séquence de tags la plus probable qui se termine par l'état final Q_F . On vous demande de modifier le code de façon telle que la séquence la plus probable qui se termine par l'état final Q_F soit retournée. Testez votre code sur le modèle défini plus haut et calculez l'étiquetage le plus probable de la phrase *time flies like an arrow* étant

donné ce modèle.

Considérez enfin l'automate de la page suivante qui représente un modèle de Markov caché. A chaque arête est associée une probabilité de transition $t_{i,j}$ d'un état i à vers état j et à chaque sommet sont associés un état i et une distribution conditionnelle $e_i(\cdot)$ sur les symboles générés par l'état i . Etant le modèle défini par cette automate, calculez à la main les valeurs $\delta(i,l)$ du tableau ci-dessous pour la séquence de symboles $aabab$ et retournez la séquence d'états la plus probable :

	a	a	b	a	b
Q_I					
Q_1					
Q_2					
Q_3					
Q_F					



Faites-moi parvenir vos solutions au plus tard le mercredi 24 janvier.