Point de départ : le corpus

Construison un HMM depuis un corpus d'apprentissage[3] :

Y	=	DET	→	NC	→	VERB
		↓		↓		↓
X	=	la		bonne		soupe
Y	=	DET	→	ADJ	→	NC
		↓		↓		↓
X	=	la		bonne		soupe
Y	=	DET	→	NC	→	VERB
		↓		↓		↓
X	=	la		soupe		fume

[3]exemple repris de : https://lattice.cnrs.fr/sites/itellier/cours-HMM-CRF.pdf

Statistiques

Initiales :

- DET = 1

Transitions :

- ADJ $\implies$ NC = 1
- DET $\implies$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\implies$ VERB = 1

Émissions :

- ADJ $\implies$ bonne = 1
- DET $\implies$ la = 1
- NC $\implies$
  - bonne = $\frac{1}{3}$
  - soupe = $\frac{2}{3}$
- VERB $\implies$
  - soupe = $\frac{1}{2}$
  - fume = $\frac{1}{2}$

Algorithme de Viterbi

L'algorithme de Viterbi utilise la propriété de Markov pour donner la meilleure séquence. À chaque instant, on évalue quel est le meilleur "pas" à faire, la meilleure séquence étant simplement la suite des meilleurs pas (la séquence la plus probable est la suite des transitions les plus probables).

Algorithme de Viterbi

L'algorithme de Viterbi utilise la propriété de Markov pour donner la meilleure séquence. À chaque instant, on évalue quel est le meilleur "pas" à faire, la meilleure séquence étant simplement la suite des meilleurs pas (la séquence la plus probable est la suite des transitions les plus probables).

L'algorithme fonctionne de la façon suivante :

1. pour le 1er mot : on calcule, pour chaque étiquette, sa probabilité initiale étant donné le mot.
2. pour les autres : on calcule, pour chaque étiquette, les meilleures transitions depuis le mot précédent, on garde une trace de là où l'on vient (*back pointer*).
3. récupération la meilleure séquence en partant de la fin et en rebroussant chemin (*backtrack*).

HMM et morphosyntaxe : Algorithme de Viterbi (1.)

On commence avec une matrice "vide" (cellule vide = proba de 0).

On calcule la probabilité du premier mot en multipliant la probabilité initiale et la probabilité d'émission...

	la	*bonne*	*fume*
ADJ	0		
DET	1		
NC	0		
VERB	0		

Initiales :

- DET = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.a)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0		
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\implies$ NC = 1
- DET $\implies$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\implies$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.b)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0 →	0	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\implies$ NC = 1
- DET $\implies$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\implies$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.c)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$1 \times \frac{1}{3}$	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\implies$ NC = 1
- DET $\implies$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\implies$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.c)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\implies$ NC = 1
- DET $\implies$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\implies$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.c)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.c)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.c)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1		
NC	0		
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.d)

Une fois la meilleure transition trouvée, on la multiplie avec la probabilité d'émission.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}\times 1$	
DET	1		
NC	0		
VERB	0		

Émissions :

- ADJ $\Longrightarrow$ bonne = 1
- DET $\Longrightarrow$ la = 1
- NC $\Longrightarrow$
  - bonne = $\frac{1}{3}$
  - soupe = $\frac{2}{3}$
- VERB $\Longrightarrow$
  - soupe = $\frac{1}{2}$
  - fume = $\frac{1}{2}$

HMM et morphosyntaxe : Algorithme de Viterbi (2.e)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0		
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.f)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0	$1\times\frac{2}{3}$	
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.f)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0	$\frac{2}{3}$	
VERB	0		

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.g)

Une fois la meilleure transition trouvée, on la multiplie avec la probabilité d'émission.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0	$\frac{2}{3}\times\frac{1}{3}$	
VERB	0		

Émissions :

- ADJ $\Longrightarrow$ bonne = 1
- DET $\Longrightarrow$ la = 1
- NC $\Longrightarrow$
  - bonne = $\frac{1}{3}$
  - soupe = $\frac{2}{3}$
- VERB $\Longrightarrow$
  - soupe = $\frac{1}{2}$
  - fume = $\frac{1}{2}$

HMM et morphosyntaxe : Algorithme de Viterbi (2.g)

Une fois la meilleure transition trouvée, on la multiplie avec la probabilité d'émission.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0	$\frac{2}{9}$	
VERB	0		

Émissions :

- ADJ $\Longrightarrow$ bonne = 1
- DET $\Longrightarrow$ la = 1
- NC $\Longrightarrow$
  - bonne = $\frac{1}{3}$
  - soupe = $\frac{2}{3}$
- VERB $\Longrightarrow$
  - soupe = $\frac{1}{2}$
  - fume = $\frac{1}{2}$

HMM et morphosyntaxe : Algorithme de Viterbi (2.h)

On calcule la meilleure transition pour chaque étiquette du mot suivant : "bonne".

Pour chaque étiquette à un instant t (étiquette$_t$) possible : on cherche la meilleure transition depuis l'instant $t-1$.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	
DET	1	0	
NC	0	$\frac{2}{9}$	
VERB	0	0	

Transitions :

- ADJ $\Longrightarrow$ NC = 1
- DET $\Longrightarrow$
  - NC = $\frac{2}{3}$
  - ADJ = $\frac{1}{3}$
- NC $\Longrightarrow$ VERB = 1

HMM et morphosyntaxe : Algorithme de Viterbi (2.i)

On continue jusqu'à remplir la matrice... On peut (enfin!) passer à la dernière étape.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	0
DET	1	0	0
NC	0	$\frac{2}{9}$	0
VERB	0	0	$\frac{1}{9}$

HMM et morphosyntaxe : Algorithme de Viterbi (3.)

À présent, on parcourt la matrice en sens inverse pour reconstituer la meilleure séquence.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	0
DET	1	0	0
NC	0	$\frac{2}{9}$	0
VERB	0	0	$\frac{1}{9}$

HMM et morphosyntaxe : Algorithme de Viterbi (3.)

À présent, on parcourt la matrice en sens inverse pour reconstituer la meilleure séquence.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	0
DET	1	0	0
NC	0	$\frac{2}{9}$	0
VERB	0	0	$\frac{1}{9}$

HMM et morphosyntaxe : Algorithme de Viterbi (3.)

À présent, on parcourt la matrice en sens inverse pour reconstituer la meilleure séquence.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	0
DET	1	0	0
NC	0	$\frac{2}{9}$	0
VERB	0	0	$\frac{1}{9}$

HMM et morphosyntaxe : Algorithme de Viterbi (3.)

À présent, on parcourt la matrice en sens inverse pour reconstituer la meilleure séquence.

	la	*bonne*	*fume*
ADJ	0	$\frac{1}{3}$	0
DET	1	0	0
NC	0	$\frac{2}{9}$	0
VERB	0	0	$\frac{1}{9}$

On a donc la meilleure annotation :

- la/DET bonne/NC fume/VERB
- probabilité = $\frac{1}{9}$

Alors que cette séquence n'existe pas dans nos exemples!

Features

On peut bien sûr améliorer ce modèle en prenant en compte plus de features que simplement les formes.

En supposant encore que les features sont indépendantes, ça revient à considérer des probabilités d'émission composites :

$$\mathrm{P}([0,0]{=}\text{petit}, [-1,0]{=}\text{Le}|\mathrm{ADJ}) = \mathrm{P}([0,0]{=}\text{petit}|\mathrm{ADJ}) \times \mathrm{P}([-1,0]{=}\text{Le}|\mathrm{ADJ})$$

Cependant ce n'est pas le cadre le plus sympathique pour travailler.

- L'hypothèse d'indépendance est vite gênante

→ On aimerait pouvoir utiliser des features composites

HMM

On peut penser aux modèles de Markov cachés comme un équivalent de Naïve Bayes pour l'étiquetage, avec les mêmes avantages

- Faciles et rapides pour l'entraînement et l'inférence
- Étonnamment efficace même pour peu de données

HMM

On peut penser aux modèles de Markov cachés comme un équivalent de Naïve Bayes pour l'étiquetage, avec les mêmes avantages

- Faciles et rapides pour l'entraînement et l'inférence
- Étonnamment efficace même pour peu de données

Et les mêmes inconvénients

- Interprétabilité limitée avec beaucoup de features
- Les hypothèses d'indépendance simplifient parfois trop le problème.

Modèles d'étiquetage

MEMM et CRF

MEMM

Les *Maximum Entropy Markov Models* sont une reformulation des HMM où on modélise directement la probabilité $\mathrm{P}(\mathrm{Y}|\mathrm{X})$ en définissant manuellement ses composantes.

MEMM

Les *Maximum Entropy Markov Models* sont une reformulation des HMM où on modélise directement la probabilité $\mathrm{P}(\mathrm{Y}|\mathrm{X})$ en définissant manuellement ses composantes.

Comme pour les HMM, on travaille sous l'hypothèse que les étiquettes sont émises suivant un processus Markovien, c'est à dire qu'une étiquette ne dépend directement que de l'étiquette précédente.

Forme du modèle

On écrit comme précédemment

$$\mathrm{P}(y_1, \ldots y_n|\mathrm{X}) = \prod \mathrm{P}(y_i|y_1, \ldots, y_{i-1}, \mathrm{X})$$

Forme du modèle

On écrit comme précédemment

$$\mathrm{P}(y_1, \ldots y_n|\mathrm{X}) = \prod \mathrm{P}(y_i|y_1, \ldots, y_{i-1}, \mathrm{X})$$

Soit, avec l'hypothèse de Markov

$$\mathrm{P}(y_1, \ldots y_n|\mathrm{X}) = \prod \mathrm{P}(y_i|y_{i-1}, \mathrm{X})$$

Forme des probabilités

On modélise les probabilités comme

$$\mathrm{P}(y_i|y_{i-1}, \mathrm{X}) = \frac{e^{\phi(y_i, y_{i-1}, \mathrm{X})}}{\sum_c e^{\phi(c, y_{i-1}, \mathrm{X})}}$$

Forme des probabilités

On modélise les probabilités comme

$$\mathrm{P}(y_i|y_{i-1}, \mathrm{X}) = \frac{e^{\phi(y_i, y_{i-1}, \mathrm{X})}}{\sum_c e^{\phi(c, y_{i-1}, \mathrm{X})}}$$

Avec

$$\phi(y_i, y_{i-1}, \mathrm{X}) = \sum_k w_k \phi_k(y_i, y_{i-1}, \mathrm{X})$$

Forme des probabilités

On modélise les probabilités comme

$$\mathrm{P}(y_i|y_{i-1}, \mathrm{X}) = \frac{e^{\phi(y_i, y_{i-1}, \mathrm{X})}}{\sum_c e^{\phi(c, y_{i-1}, \mathrm{X})}}$$

Avec

$$\phi(y_i, y_{i-1}, \mathrm{X}) = \sum_k w_k \phi_k(y_i, y_{i-1}, \mathrm{X})$$

Où

- Les ϕ_k sont des features définies manuellement (souvent binaires)
- Les w_k sont des poids appris

On parle de modèle log-linéaire.

Inférence

L'inférence avec un MEMM se fait exactement comme avec un HMM : on a des probas de transition, donc on peut soit appliquer une heuristique rapide soit Viterbi.

L'apprentissage, c'est une autre paire de manches…