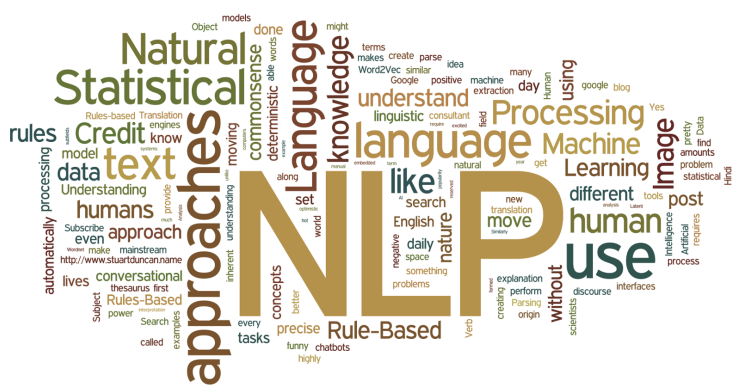
Institut Polytechnique de Paris

ENSTA Paris

MA202 - Modèles de Markov

TP 1

Le modèle HMC pour la segmentation de texte



Auteur :

Elie Azeraf

Professeur :

Wojciech Pieczynski

Année scolaire 2020-2021

1

Qu'est ce que la segmentation de texte ?

Parmi les très nombreuses tâches liées au Traitement du Langage Naturel (abrégé NLP pour Natural Language Processing), les modèles de chaînes de Markov cachées (Hidden Markov Chain, HMC) sont particulièrement adaptées pour celles concernant la segmentation de texte, également appelés *sequence labelling* ou *text segmentation* chez nos amis anglophones.

En quoi cela consiste t-il ?
Segmenter un texte consiste à labéliser chaque mot de celui-ci selon la tâche voulue. Nous avons trois types principales de segmentation que nous décrivons ci-après : Part-Of-Speech Tagging, Chunking, et Named-Entity-Recognition. La compréhension détaillée de ces trois tâches n'est pas nécessaires pour la suite du TP, est brève, non exhaustive, et consiste juste à vous familiariser avec le sujet.

1.1 Part-Of-Speech Tagging

Le Part-Of-Speech (POS) Tagging consiste à labéliser chaque mot d'une phrase avec sa fonction grammaticale tel que : NOM, ADJECTIF, DÉTERMINANT, etc...
Par exemple :

- *(Bruce, Wayne, is, the, main, vigilante, of, Gotham, .)* a les labels suivants : *(NOUN, NOUN, VERB, DET, ADJ, NOUN, DET, NOUN, PUNCT)*.
- *(ENSTA, is, one, of, the, most, famous, school, in, France, .)* a les labels *(NOUN, VERB, NUM, PREP, DET, ADV, ADJ, NOUN, PREP, NOUN, PUNCT)*

Pour évaluer un modèle de POS tagging, on calcule le taux de réponses corrects (ou de réponses fausses).

Pour ce TP, nous utiliserons le dataset Universal Dependencies English pour le POS Tagging (tous les datasets sont disponibles sur le Github, dont le lien sera disponible plus loin).

1.2 Chunking

Le chunking consiste à décomposer une phrase de manière syntaxique, nous allons capter la fonction d'un groupe de mot plutôt que les mots individuellement. Nous sommes un niveau au dessus du POS Tagging. Ainsi, une phrase peut se décomposer en différents groupes de mots : ceux liés à un nom, à un verbe, etc..
Par exemple :

— *(Mr., Carlucci, „ 59, years, old, „ served, as, defense, secretary, in, the, Reagan, administration, .)* a les labels *(NP, NP, O, NP, NP, ADJP, O, VP, PP, NP, NP, PP, NP, NP, NP, O)*.

— *(Rockwell, said, the, agreement, calls, for, it, to, supply, 200, additional, so-called, shipsets, for, the, planes, .)* a les labels *(NP, VP, NP, NP, VP, SBAR, NP, VP, VP, NP, NP, NP, NP, PP, NP, NP, O)*

Le Chunking peut être évalué selon le score F1 (https ://en.wikipedia.org/wiki/F1_score), avec le label O comme étant en référence.
Dans ce TP, nous utiliserons le dataset CoNLL 2000 pour le Chunking.

1.3 Named-Entity Recognition

La reconnaissance d'entités nommées (NER) consiste à trouver les entités au sein d'une phrase. Une entité peut être un nom, une ville, le nom d'une société, etc...

— *(IBM, is, a, company, based, in, New, York, .)* a les labels *(ORG, O, O, O, O, O, LOC, LOC, O)*

— *(Bruce, Wayne, is, the, secret, identity, of, Batman, .)* a les labels *(PER, PER, O, O, O, O, O, PER, O)*

La NER s'évalue avec le F1 Score basé sur le label O également. Pour ce TP, nous utiliserons le dataset CoNLL 2003 pour la NER.

La personne intéressée plus en détail par ces tâches et la NLP en général peut poursuivre son étude avec le NLTK Book : https ://www.nltk.org/book/ plus orienté pratique, et l'excellent livre de Jurafsky https ://web.stanford.edu/ jurafsky/slp3/ pour plus de théorie.

2

Rappels sur les chaînes de Markov cachées

Une chaîne de Markov cachées (HMC) est un double processus stochastique composé d'une séquence "cachée" et d'une séquence "observée". Le graphe de probabilité d'une HMC est donnée dans la figure ci-dessous.

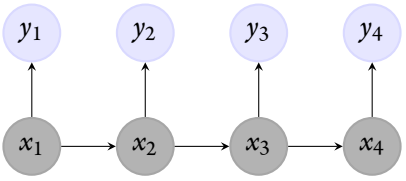


Figure 1 – Graphe de probabilité d'un HMC

Soit la séquence de taille T, nous notons $x_{1:T} = (x_1, ..., x_T)$ la séquence cachée, et $y_{1:T} = (y_1, ..., y_T)$ la séquence observée. $\forall t \in \{1, ..., T\}, x_t$ prend ses valeurs dans l'espace discret $\Lambda_X = \{\lambda_1, ..., \lambda_N\}$, et y_t dans Ω_Y. La loi de probabilité d'un HMC est la suivante :

$$p(x_{1:T}, y_{1:T}) = p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)...p(x_T|x_{T-1})p(y_T|x_T)$$

Ainsi, comme nous allons le voir, un HMC est capable de retrouver les variables cachées à partir des observations, en modélisant les variables comme ci-dessus.

Prenons l'exemple du POS Tagging, les variables observées sont les mots, et les variables cachées les fonctions grammaticales. Les réalisations des variables aléatoires sont notés en minuscules.

2.1 Les paramètres d'un HMC

Nous supposons prendre un HMC stationnaire. Il peut être défini avec les trois paramètres suivants, $\forall(\lambda_i, \lambda_j) \in \Lambda_X^2, y \in \Omega_Y$:

— Probabilité d'un état π : $\forall t \in \{1, ..., T\}$,

$$\pi(i) = p(x_t = \lambda_i)$$

— Probabilité de transition A : $\forall t \in \{1, ..., T-1\}$,

$$a_i(j) = P(x_{t+1} = j | x_t = \lambda_i)$$

— Probabilité d'émission B : $\forall t \in \{1, ..., T\}$,

$$b_i(y) = P(y_t = y | x_t = \lambda_i)$$

2.2 Calcul de $p(x_t = \lambda_i | y_{1:T})$: l'algorithme Forward-Backward

Supposons que nous observons les mots d'une phrase, nous avons les trois paramètres π, A, et B, nous allons voir comment retrouver nos labels grâce à l'algorithme Forward-Backward.

L'objectif de l'algorithme Forward-Backward est de calculer l'estimateur par MPM, cela consiste à calculer, $\forall t \in \{1, ..., T\}, \lambda_i \in \Lambda_X, p(x_t = \lambda_i | y_{1:T})$. Ainsi, il suffira de sélectionner λ_i tel que cette probabilité soit la plus grande pour restaurer notre séquence.
Mais alors, comment calculer cette probabilité ?

Nous pouvoir écrire cette probabilité de la manière suivante :

$$p(x_t = \lambda_i | y_{1:T}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j \in \Lambda_X} \alpha_t(j)\beta_t(j)}$$

avec $\alpha_t(i) = p(x_t = i, y_{1:t})$ et $\beta_t(i) = p(y_{t+1:T} | x_t = \lambda_i)$ appelées les fonctions forwards et backwards, respectivement. Elles sont calculées de manière récursive.

$\forall \lambda_i \in \Omega_X, t \in \{1, ..., T-1\}$, nous calculons les fonctions forwards de la manière suivante :

— $\alpha_1(i) = \pi(i) b_i(y_1)$,

— $\alpha_{t+1}(i) = b_i(y_{t+1}) \sum_{j=1}^{N} \alpha_t(j) a_j(i)$;

et les fonctions backwards :

— $\beta_T(i) = 1$,

— $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_i(j) b_j(y_{t+1})$.

Attention Si, lors d'un calcul d'une fonction a_t ou β_t, il existe y_t tel que pour tout $\lambda_i \in \Lambda_X, b_i(y_t) = 0$, alors il faudra remplacer les valeurs de $b_i(y_t)$ par 1 pour tout i dans ce cas. Ainsi, nous évitons de casser la chaîne, l'observation en question étant considéré comme inutile.

Ainsi, nous avons rappelé comment restaurer une HMC avec l'algorithme Forward-Backward.

Question 1 Démontrez que les fonctions forwards et backwards se calculent bien avec les récursions ci-dessus.

2.3 Calcul de $p(x_{1:T} = \lambda_{1:T} | y_{1:T})$: l'algorithme de Viterbi

Avec l'algorithme Forward-Backward, vous avez vu comme restaurer une séquence cachée avec un HMC selon le critère du MPM. Il est également possible d'utiliser le critère du MAP, consistant à trouver la séquence $\hat{x}_{1:T} \in (\Lambda_X)^T$ tel que $\hat{x}_{1:T} = \arg\max p(x_{1:T} = \lambda_{1:T} | y_{1:T})$.

L'algorithme de Viterbi consiste à calculer $V_t(k) = p(x_1, \ldots, x_t = \lambda_k, y_1, \ldots, y_t)$ la séquence la plus probable des t premières observations ayant $x_t = \lambda_k$ de la manière suivante :

$$\begin{aligned} V_1(k) &= \pi(k) b_k(y_1) \\ V_t(k) &= \max_{\lambda_i \in \Lambda_X} b_k(y_t) a_i(k) V_{t-1,i} \end{aligned}$$

Puis, en se donnant la fonction $Ptr(\lambda_k, t)$ retournant la valeur de λ_i utilisé pour calculer $V_t(k)$ si $t > 1$ ou λ_k si $t = 1$, la séquence cachée est retrouvé avec un "chemin de Viterbi" :

$$\begin{aligned} x_T &= \arg\max_{\lambda_i \in \Lambda_X} V_T(i) \\ x_{t-1} &= Ptr(x_t, t) \end{aligned}$$

Attention Si, lors d'un calcul de $V_t(k)$, il existe y_t tel que pour tout $\lambda_i \in \Lambda_X, b_i(y_t) = 0$, alors il faudra remplacer les valeurs de $b_i(y_t)$ par 1 pour tout i dans ce cas.

Ainsi, nous avons rappelé l'utilisation de l'algorithme de Viterbi pour le HMC.

PS : le livre de Jurafsky, chapitre 8, donne plus de détails pour les personnes intéressés par l'algorithme de Viterbi.

3

Modéliser une phrase avec une chaîne de Markov : le principe

Nous allons, au cours de ce TP, appliquer notre modèle HMC pour un problème réel : la segmentation de texte.

Pour que tout soit clair avant d'attaquer ce TP, comment modélise-t-on ce problème ? Supposons que nous sommes dans le POS tagging, alors les variables observées sont les mots, et les variables cachées à retrouver sont les fonctions grammaticales de ces mots. Le graphe pour représenter cela est le suivant :

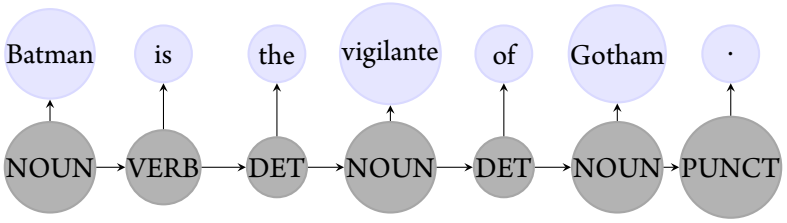


Figure 2 – Exemple d'une modélisation d'un HMC pour le POS tagging

Ainsi, nous allons appliquer notre segmentation de texte avec le modèle HMC à partir des questions suivantes.

4

Un petit mot sur le code !

Pour pouvoir mener à bien ce TP, l'utilisation de Python 3. avec la librairie NumPy est nécessaire. Les personnes n'ayant pas cette version de Python devront impérativement l'installer.

Le code est disponible sur le lien suivant :
https ://github.com/ElieAzeraf/TP_ENSTA

Vous avez à votre disposition un code contenant trois dossier.

— Dataset : ce dossier contient les datasets et comment les charger, vous n'aurez jamais à toucher ce dossier. Sachez qu'un dataset chargé se présente sous la forme d'une liste python. Chaque élément de la liste correspond à une phrase. Une phrase est de la forme $(x_1, y_1), (x_2, y_2), ..., (x_T, y_T)$.

— Learning_Parameters : contenant le fichier HMC_Learning_Parameters.py qui comporte la fonction permettant de calculer les paramètres π, A, et B pour un dataset, vous n'avez pas besoin de modifier ce fichier non plus

— HMC : dossier principale de notre TP, vous y trouverez :

  — HMC_Forward_Backward_Restorator.py : ce fichier conporte la classe permettant d'appliquer l'algorithme Forward-Backward

  — HMC_Viterbi_Restorator.py : ce fichier conporte la classe permettant d'appliquer l'algorithme de Viterbi

  — Le dossier Result_HMC_FB : contenant les trois fichiers pour appliquer l'algorithme Forward-Backward de HMC, vous n'aurez pas à modifier ces fichiers

  — Le dossier Result_HMC_Viterbi : contenant les trois fichiers pour appliquer l'algorithme de Viterbi pour le HMC, vous n'aurez pas à modifier ces fichiers

Question 2 Dans le fichier HMC_Learning_Parameters.py du dossier Learning_Parameters se trouve la fonction pour l'estimation de nos trois paramètres π, A, et B.

1. Décrivez en détail le fonctionnement de cette fonction, comment cette estimation est-elle réalisée ?
2. Vous remarquerez que les différents paramètres sont modélisés par des dictionnaires en python. Selon vous, pourquoi procède t-on ainsi, et non par des matrices ?

5

L'algorithme Forward-Backward pour la segmentation de texte

Question 3 Prenez le fichier HMC/HMC_Forward_Backward_Restorator.py, prenez connaissance des différentes fonctions en présence. Complétez, à partir du rappel des formules de l'algorithme Forward-Backward, les fonctions :

- compute_alpha_1
- compute_alpha_t_plus_1
- compute_beta_T
- compute_beta_t

Question 4 Vous pouvez maintenant compiler l'ensemble des fichiers du dossiers Result_FB_HMC. Présentez chacun des résultats sous forme d'un tableau, en notant bien les distinctions entre les mots connus et les mots inconnus. Que remarquez-vous ?

Question 5 Dans les quatre fonctions complétées lors de la question 2, à la fin de chaque fonction nous normalisons nos probabilités forwards et backwards que nous venons de calculer. Dans quelle but faisons-nous cela ? Pouvez-vous prouver pourquoi cela n'affecte pas les résultats de notre algorithme.

Félicitations ! A l'issue de cette partie vous êtes capables d'appliquer l'algorithme Forward-Backward avec le modèle HMC pour de la segmentation de textes !

6

L'algorithme de Viterbi pour la segmentation de texte

Question 6 De même, prenez le fichier HMC/HMC_Viterbi_Restorator.py, et prenez connaissance des différentes fonctions en présences. Complétez, à partir des formules de l'algorithmes de Viterbi ainsi que l'implémentation indiqué sur https ://en.wikipedia.org/wiki/Viterbi_algorithm, complétez les fonctions :

- compute_V1
- compute_V_t_plus_1
- viterbi_path

Question 7 Vous pouvez maintenant compiler l'ensemble des fichiers du dossiers Result_Viterbi_HMC. Présentez chacun des résultats sous forme d'un tableau. Comparez ces résultats avec les précédents.

Le rendu de ce TP doit ce faire en complétant les fichiers HMC_Forward_Backward_Restorator.py et HMC_Viterbi_Restorator.py. N'hésitez pas en cas de questions durant le TP, et par la suite par mail à azeraf.elie@telecom-sudparis.eu et/ou elie.azeraf@ibm.com

May the force be with you !