République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université des Sciences et Technologies Houari Boumediene.

Rapport de projet du module

Traitement Automatique du Langage Naturel :

Développement d'un outil pour l'extraction des entités nommées

(Named Entity Extraction) pour la langue arabe.

Réalisé par :

ABOULKACEM Hania

DJEHICHE Manel

SAHARI Hadjer

Année scolaire : 2019 - 2020

Introduction :

Lors de ce projet, nous avons développé un extracteur des entités nommées pour la langue arabe sous Python en utilisant les techniques de Traitement Automatique du Langage Naturel et en nous basant sur l'extracteur de SAIE (Al Shamsi et Guessoum, 2005).

En premier lieu, nous allons présenter brièvement la bibliothèque logicielle NLTK. Ensuite, nous allons expliquer qu'est-ce que l'extraction des entités nommées et à quoi cela peut servir. Par la suite, nous allons présenter le corpus sur lequel nous avons effectué notre travail et expliquer les différentes étiquettes utilisées. Après cela, nous expliquons en détail le travail effectué, étape par étape. Enfin, la dernière partie sera la présentation de l'interface de notre application avec des captures d'écran et explication de fonctionnement.

I. Natural Langage Toolkit :

Natural Langage Toolkit (NLTK) est une bibliothèque logicielle en Python. Elle a été développée par Steven Bird et Edward Loper du département informatique de l'Université de Pennsylvanie. Elle permet un traitement automatique des langues, fournit des démonstrations graphiques, des données échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API).

II. Extraction des entités nommées :

L'extraction des entités nommées est une sous-tâche de l'activité d'extraction d'information dans ces corpus documentaires.

L'extraction des entités nommées consiste à parcourir un texte en entier mot par mot et de séparer les mots qui représentent des entités nommées du reste du texte et les classifier d'après les différents types. Les entités nommées sont des noms propres : Des noms de personnes, noms d'organisations, noms de lieu, des dates… Etc.

III. Pourquoi l'extraction des entités nommées ?

Cette tâche est une composante essentielle dans plusieurs domaines du Traitement Automatique du Langage Naturel : Analyse syntaxique, traduction automatique, recherche d'informations.

La reconnaissance des entités nommées peut aider à améliorer les performances de diverses applications de traitement de langage naturel, tel que l'extraction d'information, la récupération d'information, ainsi que les réponses aux questions.

IV. Pourquoi l'arabe ?

La motivation est due au manque de ressources pour les entités nommées en arabe et pour améliorer la précision atteinte dans les précédents systèmes.

V. Le corpus :

Un corpus est un ensemble de documents, artistique ou non, regroupés dans une optique précise. [1]

Pour notre projet, nous avons utilisé un corpus que nous avons téléchargé à partir d'internet. Il s'agit du corpus « Arabic Named Entity Recognition Corpus » (ANERCorp) que nous avons trouvé dans un Github. [2]

Ce corpus contient au total X mots. Les étiquettes utilisées sont : Other (O), B-PERS pour désigner le début d'une entité de type PERSON, I-PERS pour désigner la suite d'une entité de type PERSON. Lorsque l'entité PERSON n'est composée que d'un seul mot, on retrouve l'étiquette B-PERS seulement. Lorsque l'entité PERSON qui composé de 2 ou plusieurs mots, on retrouve B-PERS comme étiquette du premier mot, puis I-PERS pour les étiquettes des mots suivants. Il y a également les étiquettes, B-LOC et I-LOC qui représentent Location, B-ORG et I-ORG qui représentent Organisation, B-MISC et I-MISC pour représenter les entités monétaires.

VI. Les étapes du traitement :

Pour avoir une meilleure reconnaissance des entités nommées, nous avons décidé d'exécuter pour commencer une tokenization ainsi que le POS Tagging sur le texte de notre corpus. La raison étant qu'avoir des étiquettes bien spécifiques des entités non-nommées (OTHER) serait plus efficace pour la détection des entités nommées. Donc, nous avons exécuté le POS Tagging sur le texte de notre corpus afin de récupérer le type de chacun des mots portants l'étiquette OTHER.

1- Traitement du Dataset :

Pour commencer le pré-traitement, nous avons effectué un traitement sur le dataset. Nous avons corrigé quelques erreurs de frappe présentes, éliminé le bruit. Nous avons aussi rajouté manuellement quelques étiquettes non-nommées (Conjonctions, prépositions, pronoms) afin que ça nous aide pour la déduction des entités nommées. Nous avons aussi rajouté l'étiquette DATE qui ne figurait pas dans les étiquettes de base du corpus. L'étiquette DATE englobe les années, les mois, les jours et l'heure.

Exemple :

25 سبتمبر 2019 => I-DATE / I-DATE / B-DATE

سبتمبر 2019 => I-DATE / B-DATE

الاثنين 2019 => I-DATE / B-DATE

2020 => B-DATE

الاثنين => B-DATE

2- Enrichissement du lexique :

Nous avons enrichi le lexique avec d'autres types d'étiquettes. Nous avons rajouté l'étiquette MESURE ('متر', 'ملليمتر', 'سنتيمتر', 'ديسيمتر', 'كيلومتر', 'بوصة', 'قدم', 'ميل', 'هكتار', 'لتر', 'ملليلتر', 'سنتيلتر', 'دسيليتر', 'غالون', 'درجة', 'ثانية', 'دقيقة', 'ساعة', 'يوم', 'عام', 'سنة', 'غرام', 'كيلوغرام', 'أوقية', 'رطل', 'طن', 'جول', 'أوم', 'فولت', 'كولون', 'أمبير', 'فهرنهايت', 'كيلووات', 'واط")

Et également l'étiquette B-MISC, I-MISC qui n'existait pas encore dans le corpus SAIE.

3- Transformation du dataset en un DataFrame:

Pour cette partie, nous avons juste transformé ledataset étiqueté en un tableau, plus précisément un DataFrame, tel que la première colonne contient les mots de notre corpus et la seconde contient l'étiquetage correspondant.

4- Modèle de Markov caché (Hidden Markov Model « HMM ») :

« C'est un modèle statistique dans lequel le système modélisé est supposé être processus Markovien de paramètres inconnus. Contrairement à une chaîne de Markov classique, où les transitions prises sont inconnues de l'utilisateur mais où les états d'une exécution sont connus, dans un modèle de Markov caché, les états d'une exécution sont inconnus de l'utilisateur.»[3]

Les modèles de Markov cachés sont basés sur une approche purement probabiliste très populaire. Il en résulte d'assez bons résultats généralement.

Selon Merialdo (1995: 10): « Les modèles de Markov constituent un des types de modèles probabilistes les plus utilisés, en raison de leur simplicité et de leur efficacité ».

Le modèle HMM détectera l'étiquette d'un mot en se basant sur le contexte précédent ce dernier. Ça consiste à déduire la séquence d'états cachés à partir des évènements observés.

Nous pouvons définir le modèle HMM de manière formelle, comme suit : $\lambda = (A, B, \pi)$

A représente la matrice de transition.

La matrice de transition est un ensemble de probabilité A = $a_{01}a_{02}a_{03}$ ... a_{0n} ... a_{nn}

a_{ij} représente la probabilité de passer de l'état i à l'état j. Ainsi, l'état actuel ne dépend que de l'état précédent.

B représente la matrice d'observations.

La matrice d'observation est un ensemble de probabilité

B = $b_0(1)b_0(1)b_0(2)$ ... $b_0(n)$ ... $b_n(m)$

$b_j(k)$ représente la probabilité d'observer le mot k en sachant que l'état actuel est l'état j.

π représente la matrice des probabilité initiales

$\pi = \pi_0 \pi_2 \dots \pi_n$

π_i représente la probabilité que l'état i soit en début de phrase.

Pour la matrice des probabilités initiales, nous l'avons remplie aléatoirement.

Nous avons calculé les matrices A et B à partir du DataFrame du dataset.

Pour la matrice de transition, nous avons calculé la probabilité d'avoir une étiquette 2 en sachant qu'on a l'étiquette 1 :

P(etiq2/etiq1) = Count(etiq1,etiq2) / Count(etiq1)

En ce qui concerne la matrice d'émission, nous avons calculé la probabilité d'avoir un mot en sachant qu'on a etiq :

P(mot/etiq) = Count(etiq,mot) / Count(etiq)

Nous avons stocké les calculs dans 2 DataFrame, un pour chaque matrice.

5- L'algorithme Viterbi [4]:

a. Définition :

Il représente l'algorithme de décodage du modèle HMM, il sert à trouver la séquence optimale des étiquettes, étant donné une séquence d'observation, l'algorithme renvoie le chemin d'état à travers le HMM qui attribue la probabilité maximale à la séquence d'observation.

b. Pseudo-code :

Create a table `viterbi`[T][N+2]
for each state q from 1 to N **do**
    `viterbi`[1][q] $= p(q|q_0) \times p(o_1|q)$
    `backpointer`[1][q] $= 0$
end for
for $t = 2$ to T **do**
    for each state q from 1 to N **do**
        `viterbi`[t][q] $= p(o_t|q) \cdot \max_{q'=1}^{N}$ `viterbi`$[t-1][q'] \cdot p(q|q')$
        `backpointer`[t][q] $= \arg\max_{q'=1}^{N}$ `viterbi`$[t-1][q'] \cdot p(q|q')$
    end for
end for
`viterbi`[T][q_F] $= \max_{q=1}^{N}$ `viterbi`$[T][q] \cdot p(q_F|q)$
`backpointer`[T][q_F] $= \arg\max_{q=1}^{N}$ `viterbi`$[T][q] \cdot p(q_F|q)$
return the best path by following the backtrace of the backpointers

c. Fonctionnement :

- D'abord l'algorithme établit une matrice ou un réseau de probabilités avec:
  - Une colonne pour chaque observation o_t .
  - Une ligne pour chaque état.
  - Chaque colonne a donc une cellule pour chaque état q_i

La figure suivante montre une intuition de ce réseau pour la phrase « هاجر محمد الى مكة »

- Chaque cellule du réseau $v_t(j)$ représente la probabilité que HMM soit en état j après avoir vu les premières observations t et traversé la séquence (q_1 ... q_{t-1}) d'états la plus probable, et la valeur de chaque cellule $v_t(j)$ est calculée en empruntant récursivement le chemin le plus probable qui pourrait nous conduire à cette cellule.

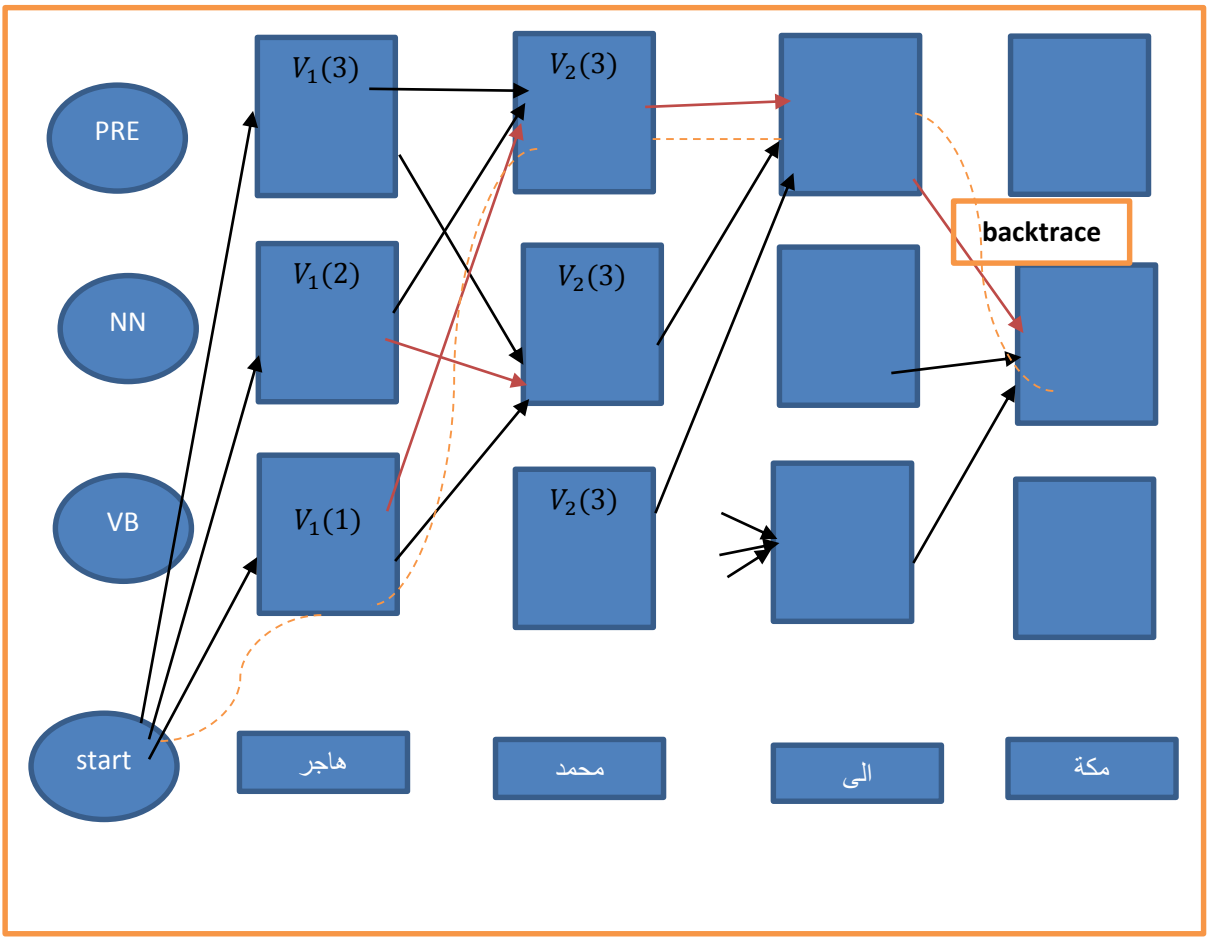


Figure 1 – Fonctionnement de l'algorithme Viterbi.

Si le programme rencontre un mot inconnu, c'est-à-dire qui n'existe pas dans le corpus, il lui attribue l'étiquette « OTHER ».

6- Modification d'un résultat sous demande de l'utilisateur :

Nous avons mis en place un bouton modification qui permet à l'utilisateur de modifier l'étiquette d'un ou plusieurs mots lorsque le résultat ne le satisfait pas.
L'utilisateur devra par ailleurs entrer la nouvelle étiquette et donc notre programme prend en considération cette modification en allant rajouter la phrase avec les nouvelles étiquettes à notre corpus puis, les étapes de calculs seront relancées afin que cette modification soit prise en compte.

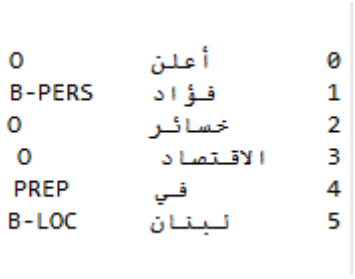
7- Quelques captures d’écran (Sans interface graphique) :

N’ayant pas encore fini l’interface graphique, voici quelques captures d’écran d’exécution du programme.

Nous donnons au programme en entrée la phrase ci-dessous :

```
['أعلن', 'فؤاد', 'خسائر', 'الاقتصاد', 'في', 'لبنان']
```

Il nous retourne en sortie les mots avec leurs étiquettes respectives comme le montre la capture ci-dessous :



O	أعلن	0
B-PERS	فؤاد	1
O	خسائر	2
O	الاقتصاد	3
PREP	في	4
B-LOC	لبنان	5

Répartition du travail :

DJEHICHE Manel : Preprocessing + Calcul des probabilités de transition.
ABOULKACEM Hania : Calcul des probabilités d'émission + Rédaction du rapport.
SAHARI Hadjer : Implémentation de l'algorithme Viterbi + Interface graphique.

Bibliographie :

[1] : https://fr.wikipedia.org/wiki/Corpus
[2] : https://github.com/EmnamoR/Arabic-named-entity-recognition
[3] : https://fr.wikipedia.org/wiki/Mod%C3%A8le_de_Markov_cach%C3%A9
[4] : https://web.stanford.edu/~jurafsky/slp3/8.pdf