Extraction d'information dans des Textes

Université Lyon 2
M2 Data Mining
Advanced Supervised Learning

Réalisé par :

Bich-Ngoc HOANG
Pierre PRABLANC
Yang YANG

Année : 2018-2019

Professeur référent :

Julien Ah-Pine (**julien.ah-pine@univ-lyon2.fr**)

Table des matières

Chapitre 1

Introduction

1.1 Introduction Générale

Dans ce projet, nous nous proposons de traiter la tâche de reconnaissances d'entités nommées qui est une sous-tâche de l'extraction d'informations. La reconnaissance d'entités nommées consiste à détecter une entité textuelle (un mot, ou un groupe de mots) et à la classer dans une catégorie pouvant être des noms de personnes, de lieux, d'organisations, ou d'autres catégories plus spécifiques. Dans notre cas, nous nous intéressons à la reconnaissance de noms communs issus de la littérature scientifique médicale.

La quantité ainsi que la production de littérature dans les domaines scientifiques connaît une telle croissance qu'il est devenu humainement difficile d'analyser les articles pertinents noyés dans la masse de documents. La reconnaissance d'entité nommées permet donc d'aider à sélectionner les documents contenant des noms appartenant à une certaine entité (comme la catégorie des protéines par exemple). Les systèmes d'extractions de connaissance, qui servent à réunir et mettre à disposition de manière structurée l'information disponible dans la littérature, sont également alimentés par la reconnaissance d'entités nommées.

1.2 Problématique

Le problème considéré est une tâche de catégorisation pour laquelle il faut attribuer une classe à un mot. Nous disposons pour cela d'un corpus de données annotées dont une partie doit servir à alimenter le système de reconnaissance tandis que l'autre sert à l'évaluer.
Pour résoudre ce problème, on pourrait penser que la solution triviale consiste à comparer le mot à classifier à des dictionnaires de mots obtenus dans le jeu de données labellisées. Chaque dictionnaire correspondant à une classe, si le mot existe dans un dictionnaire alors il serait attribué à la classe correspondante. Une telle solution trouve malheureusement rapidement de nombreuses limitations. Comment classifier un mot qui n'existe dans aucun des dictionnaires (c'est à dire jamais observé dans le jeu d'entraînement) ? Par ailleurs, un même mot peut recouvrir plusieurs sens et donc appartenir à une classe différente selon le contexte. Il s'agit ici d'un problème d'ambiguïté. Le problème n'étant pas si simple, plusieurs méthodes ont été proposées.

1.3 Différentes Approches

Méthodes Rules-Based

Historiquement, les premières méthodes étaient basées sur des règles ("Rules-Based") élaborées soit de manière automatique, soit écrites "à la main". Le principe se base sur des paires $(pattern, action)$ où un *pattern* correspond généralement à un une expression régulière. Lorsqu'un token (ou une séquence de tokens) correspond à un *pattern*, une *action* associée est exécutée. Cette *action* correspond à l'étiquetage des tokens

(entité, début ou fin de l'entité par exemple). Bien que très efficaces, ces approches nécessitent, dans le cas des règles écrites à la main, un expert à la fois de la langue et du domaine, rendant la technique très spécifique et très coûteuse en temps. Pour le cas des règles obtenues de manière automatique, elles souffrent d'un manque de précision [**?**]. On trouve également d'autres inconvénients impactant par exemple la robustesse du système de reconnaissance. En effet, lorsque de nouvelles données nécessitent de nouvelles règles, il faut alors mettre à jour la table de règles.

Méthodes Statistiques

Les approches statistiques permettent de pallier certains inconvénients des méthodes Rules-Based car les "règles" sont apprises sur les données et non à la main. Pour la reconnaissance d'entité nommées, on distingue dans ces méthodes d'apprentissage statistique principalement 2 types d'approches : semi-supervisées et supervisées. Les deux types d'approches font usage des données étiquetées. Une partie de ces données sert à entraîner le modèle de classification, puis les données restantes servent à évaluer les performances du modèle.
Dans les méthodes semi-supervisées, le jeu d'entraînement est constitué de données étiquetées et non-étiquetées. Nous ne traiterons pas le cas des méthodes semi-supervisées dans ce rapport. On peut néanmoins citer les approches par Boostrapping [**?**] et par Co-training [**?**].
Pour le cas des méthodes supervisées, les données d'entraînement contiennent uniquement des données étiquetées. De nombreuses approches en apprentissage supervisé ont été développées basées sur les arbres de décision, modèles de Maximum d'Entropie, SVM, Réseaux de neurones, Bayésien Naïf, HMM, CRF ... (la liste n'est pas exhaustive).
Dans ce rapport, nous ne présenterons que les méthodes Bayésien Naïf (NB), Modèles de Markov Cachés (HMM) et Conditional Random Field (CRF).

1.4 Organisation du Rapport

Nous présentons dans ce rapport ainsi 3 méthodes de reconnaissance d'entité nommée qui sont appliquées sur le corpus GENIA. Dans le chapitre 2, nous présentons une vue générale du système. Chaque élément du système est ensuite détaillé (pré-traitements, extraction des descripteurs) ainsi que les 3 méthodes de classification (NB, HMM et CRF). Les expériences sont présentées dans le chapitre 3. Dans ce chapitre, nous rappelons brièvement le contenu et la structure du corpus GENIA. Les tests ne sont appliqués que sur une partie des entités nommées du corpus. Nous expliquons donc quelles entités nommées ont été sélectionnées pour l'évaluation du système. Enfin, nous présentons les différentes configurations des systèmes testés ainsi que leur résultats. Finalement, en dernière partie, on expose les comparaisons et critiques des configurations testées.

Chapitre 2

Reconnaissance d'Entité Nommées

2.1 Vue d'Ensemble

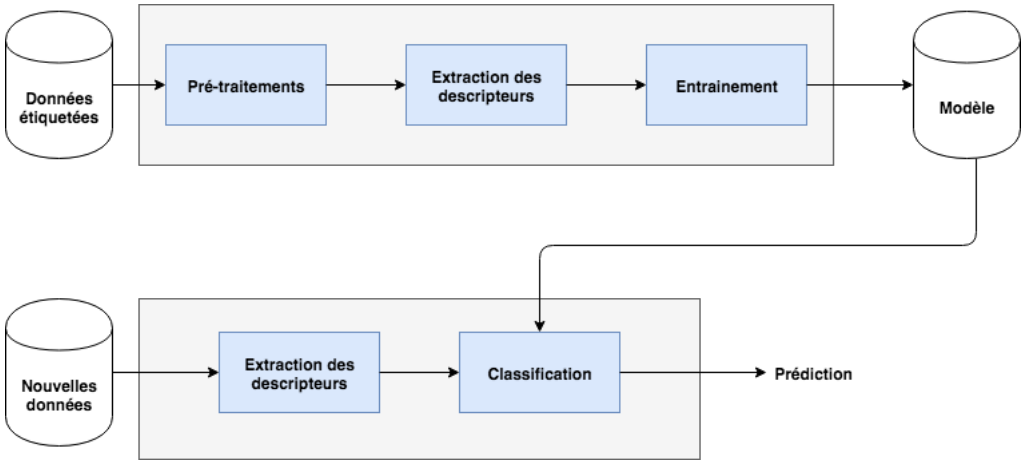

FIGURE 2.1 – Vue d'ensemble du système de reconnaissance d'entité nommée. Partie supérieur (entraînement), partie inférieur (test).

Les sytèmes de reconnaissance d'entité nommée dans le cas de l'apprentissage supervisé fonctionnent en deux temps (voir 2.1). D'abord le modèle est appris à partir de données d'entrainement comportant la vérité de terrain (données étiquetées ou labélisées). Dans notre cas, cette vérité de terrain correspond à l'association mot / entité nommées qui a été réalisée à la main. Ces données d'entrainement sont mises en forme dans le module de pré-traitement. Une fois après avoir obtenu les données sous forme de paires $(mot, label)$, on procède à l'extraction des features. Les features sont sélectionnées et envoyées avec leur label correspondant en entrée du module d'entrainement pour estimer les paramètres du modèle. Dans un deuxième temps, on peut procéder à la tache de classification de nouvelles données en procédant à l'extraction des features de chaque mot afin de leur attribuer une classe d'entité nommée. C'est la phase de prédiction.

2.2 Pré-Traitements

Le corpus de données annotées n'est pas dans un format directement utilisable. Il s'agit généralement d'un fichier structuré (XML dans notre cas). Il faut donc extraire les mots ainsi que leurs labels associés en utilisant le format balisé. Pour cela, nous utilisons la bibilothèque `xml2` de R. Par ailleurs, nous n'effectuons

pas la reconnaissance d'entité nommées sur la totalité des entités nommées disponibles dans le corpus mais uniquement sur un nombre limité d'entités nommées regroupées selon leur ontologie. Tout cela ainsi que d'autres pré-traitement est décrit dans la présentation du corpus GENIA dans le chapitre 3.

2.3 Extraction des Descripteurs

Inspiré par l'article de [**?**] et [**?**] qui travaillent sur le même corpus, nous avons décidé d'utiliser les descripteurs Word Formation Pattern (WFP), Special Verb Trigger (SVT), Morphological Pattern (Affixes), Part-Of-Speech (POS) avec les descripteurs connus en Text Mining : tokens et formes lemmatisées.

Tokens et formes lemmatisées

Les mots tokennisés par le package `quanteda` de R selon le vocabulaire extrait à partir du corpus ou les ponctuations. La lemmatisation permet de regrouper un terme prenant différentes formes (plurielle/singulier, conjugué/infinitive, etc. ...). Les lemmes sont obtenus à partir de la même bibliothèque.

Word Formation Pattern

L'objectif de ce descripteur est de capturer les capitalisations, digitalisations et les informations des autres formes de mots. Ce descripteur est très connu et utilisé souvent dans le domaine biomédical. Il aide à distinguer les différentes entités du domaines avec les autres. Par exemple : *"H2A"* peut indiquer une entité du domain biomédical [**?**]. La table ci-dessous liste tous les WFP que nous avons utilisé (Tableau 2.1).

TABLE 2.1 – F_{WFP} : Word Formation Pattern

F_{WFP}	e.g.	F_{WFP}	e.g.
Comma	,	OneCap	T
Dot	.	AllCaps	CSF
Parenthesis	( ) [ ]	CapLowAlpha	All
RomanDigit	II	CapMixAlpha	IgM
GreekLetter	Beta	LowMixAlpha	kDa
StopWord	in,at	AlphaDigitAlpha	H2A
ATCGsequence	ACAG	AlphaDigit	T4
OneDigit	5	DigitAlphaDigit	6C2
AllDigits	60	DigitAlpha	19D
DigitCommaDigit	1,25	Others	other
DigitDotDigit	0.5		

Morphological Pattern (Affixes)

Les informations morphologiques comme préfixes et suffixes sont importantes pour identifier les termes et sont souvent utilisées dans le domaine de biomédicale [**?**] [**?**]. Dans notre projet, nous trouvons les préfixes/suffixes les plus fréquents dans les données d'entraînement et les considérons comme candidats pour les évaluer :

$$Weight(Candidate_i) = \frac{\#IN_i - \#OUT_i}{\#IN_i + \#OUT_i}$$

$\#IN_i$ est le nombre du ième candidat ayant une entité et $\#OUT_i$ est le nombre du ième candidat n'ayant pas d'entité. Les candidats présentant un poids supérieur à un seuil donné sont choisis. Ici, nous avons utilisé un seuil égal à 0.7.

Special Verb Trigger

Comme le prouvent queleques recherches, les verbes sont utiles pour l'extraction des interactions entre les entités biomédicales [**?**]. Par exemple, le verbe *"bind"* est utilisé souvent pour indiquer les interactions entre les protéines. Pour la reconnaissance d'entités nommées dans le domaine de biomédical, il est certain que quelques verbes particuliers peuvent fournir des informations sur les entités nommées. Dans notre système, nous gardons les 60 premiers verbes les plus fréquents.

TABLE 2.2 – Tableau occurrence non exhaustif des SVT

Verbe	Nombre d'occurrences
bind	808
induce	432
activate	399
contain	439
...	...

Part-of-speech

Selon les recherches précédentes [**?**], les part-of-speech (POS) ne semblent pas utiles et n'aident pas beaucoup dans NER de biomédical car le fait d'utiliser un POS tagger (annotateur) d'un domaine différent du domaine de biomédical n'est pas adaptable. Dans notre projet, nous utilisons un POS tagger provenant d'un package de R `udpipe`. Cet annotateur de POS nous a aidé à annoter tous les tokens du corpus. Bien qu'il soit un annotateur du domain général et comme indiqué dans les articles, cela pourrait affecter les résultats du classifieurs. Cependant, nous n'avons pas d'accès à d'autres annotateurs du domaine biomédical. Nous décidons donc d'entraîner les modèles sans et avec les POS afin de vérifier les propositions des articles que nous avons consultés.

TABLE 2.3 – Liste des POS

Nom	Définition
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

2.4 Méthodes de Classification

2.4.1 Bayésien Naïf

Les modèles bayésiens sont des modèles de classification qui se basent sur la règle de Bayes. Pour prédire l'appartenance d'une observation à une classe, on cherche à maximiser la probabilité à posteriori.
On note $X = (x_1, x_2, ..., x_n)$ une observation composée de n descripteurs. On appelle C_k la $k^{ième}$ classe. La probabilité que l'observation X apartienne à la classe C_k sachant l'observation est :

$$P(C_k|X) = \frac{P(C_k)\, P(X|C_k)}{P(X)}$$

Pour la tache de classification, checher à maximiser la probabilité a posteriori est équivalent à maximiser la probabilité jointe $P(C_k, X)$ car la maximisation se fait selon la classe. Par conséquent, $P(X)$ n'intervient pas. Cela permet de faciliter le calcul car $P(X)$ est une quantité difficile à estimer. On a donc :

$$\underset{k}{\text{argmax}}\, P(C_k|X) = \underset{k}{\text{argmax}}\, P(C_k)\, P(X|C_k)$$

qui correspond au produit de la vraisemblance par la probabilité a priori de C_k. On peut appliquer successivement la règle des probabilités conditionnelles :

$$P(X|C_k) = P(x_1|C_k)\, P(x_2|C_k, x_1)\, P(x_3|C_k, x_2, x_1) \, ... \, P(x_n|C_k, x_{n-1}, x_{n-2}...x_1)$$

Si l'on fait l'hypothèse que les variables des observations sont indépendantes entre elles, on obtient la formulation du classifieur bayésien naïf :

$$\underset{k}{\text{argmax}}\, P(C_k|X) = \underset{k}{\text{argmax}}\, P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Ainsi, maximiser la probabilité a posteriori revient à maximiser le produit des probabilités conditionnelles $P(x_i|C_k)$ et la probabilité a priori des classes $P(C_k)$. Dans ce modèle, il suffit donc de calculer sans délicatesse les produits des $P(C_k)$ et $P(X_i|C_k)$ pour tous les i et tous les k et de retenir la classe qui maximise ce produit. On obtient ainsi un estimateur du Maximum A Posteriori (MAP).

Il reste donc à estimer les paramètres $P(C_k)$ et $P(x_i|C_k)$ du modèle. Pour les paramètres $P(C_k)$, si on ne dispose pas d'information a priori, on peut obtenir une estimation en calculant les proportions de chaque classe sur l'ensemble du jeu d'entrainement. Concernant les paramètres $P(x_i|C_k)$, on utilise généralement, dans le cas catégoriel, soit une distribution de Bernoulli multivariée, soit une distribution multinomiale.
Dans le cas d'une utilisation d'une distribution de Bernoulli multivariée, les probabilités conditionnelles sont définies par :

$$P(x_i|C_k) = P(i|C_k)\, x_i\, (1 - P(i|C_k))\, (1 - x_i)$$

où chaque x_i est une variable binaire prenant les valeurs 0 ou 1 et $P(i|C_k)$ est la probabilité d'occurence de x_i dans l'entrainement.
Si on utilise une distribution multinomiale de paramètre θ, les probabilités conditionnelles peuvent être estimés par :

$$P(x_i|C_k, \theta) = \frac{N_{ik} + \alpha}{N_k + \alpha n}$$

où N_{ik} désigne le nombre d'occurence de x_i sachant qu'il appartient à la classe C_k et N_k désigne le nombre total d'observations appartenant à la classe C_k. On introduit directement l'hyperparamètre α qui est un paramètre de lissage. Il permet d'obtenir d'éviter d'obtenir une probabilité nulle pour des variables qui n'auraient pas été observées. Lorsque $\alpha = 1$, on parle de lissage de Laplace.

2.4.2 Hidden Markov Model

Avant de passer à la description d'un modèle de HMM (Hidden Markov Model), nous introduisons les modèles de Markov. Le modèle de Markov, aussi appelé Chaîne de Markov, est un modèle statistique composé d'états et de transitions. Une transition matérialise la possibilité de passer d'un état à un autre.

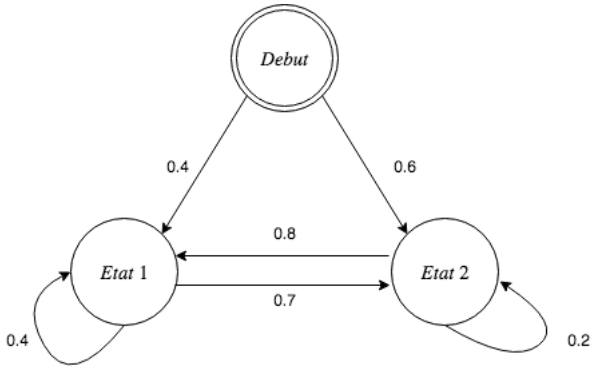

FIGURE 2.2 – Exemple d'un modèle de Markov simple avec 2 états

Dans le modèle de Markov, les transitions sont unidirectionnelles : une transition de l'état A vers état B ne permet pas d'aller de l'état B vers l'état A (Figure 2.2). Tous les états ont des transitions vers tous les autres états, y compris vers eux-mêmes. On associe à chaque transition une probabilité pouvant être éventuellement nulle. La somme des probabilités des transitions partant d'un état est toujours égale à 1. L'utilisation du modèle se fait en rapport avec le temps. À chaque "unité de temps", on opère une transition, cela génère finalement une séquence d'états.

Un modèle HMM est basé sur un modèle de Markov dans lequel on ne peut pas observer directement la séquence d'états car les états sont "cachés". Chaque état émet des "observations" qui, elles, sont observables. On ne travaille donc pas sur la séquence d'états, mais sur la séquence d'observations générées par les états. Soient les paramètres d'une modèle HMM :

- S (States) : l'ensemble des états cachés
- O (Observations) : l'ensemble d'observations
- π(startprobs) : la matrice des probabilités de départ [1xK], c'est à dire les probabilités de démarrer dans chacun des N états
- A (transprobs) : la matrice des probabilités de transition [KxK], c'est à dire les probabilités de passer d'un état à l'autre
- B (émissionprobs) : la matrice des probabilités d'émission [KxN], c'est à dire les probabilités pour chaque état d'émettre chacune des observations possibles

avec K le nombre de classes et N le nombre total d'observations. Les trois matrices ont la particularité d'avoir des lignes "stochastiques". Cela signifie que chaque élément d'une ligne est une probabilité et que la somme des éléments de la ligne est égale à 1. Dans le cas d'états observés discrets, on utilise généralement

une distribution multinomiale pour modéliser les probabilités d'émission B. Un modèle de HMM repose sur plusieurs hypothèses :

— L'état au temps t ne dépend que de l'état au temps précédent $t-1$. C'est l'hypothèse de Markov.
— Les probabilités de transitions sont indépendantes du temps auquel la transition se produit. C'est l'hypothèse de stationnarité.
— L'indépendance conditionnelle d'une observation par rapport aux observations précédentesCette dernière hypothèse est très restrictive.

Il y a trois questions majeures que nous pouvons poser à un HMM, étant donnée une séquence d'observations :

— Quelle est la probabilité d'apparition de cette séquence ?
— Quelle est la séquence d'états la plus probable qui a généré cette séquence ?
— Comment modifier le HMM pour que la probabilité d'apparition de cette séquence soit maximale ?

On appelle la première question le problème d'évaluation. Connaissant les paramètres de HMM et une sequence d'observations, il y a deux façons résoudre le problème. La plus simple méthode est de calculer directement selon la formule de probabilité. C'est-à-dire en listant toutes les possibilités de sequences de transitions l'état $P(\mathbf{O}, \mathbf{I}; \lambda)$, puis en calculant la somme $P(\mathbf{O}); \lambda)$. Cette méthode est rarement utilisée étant donnée la complexité $O(T \text{ x } Q^T)$ des calculs. Une autre manière basé sur l'algorithme du Forward-backward. On a la formule :

$$P(O;) = \sum_{i=1}^{Q} \sum_{j=1}^{Q} \alpha_t(i) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j), t = 1, 2, ..., T-1$$

Quand $t = 1$, c'est backward, et quand $t = T-1$, c'est forward.
$\alpha_t(i)$ est probabilité d'une séquence d'observations $o_1, o_2, ..., o_t$ sachant l'état caché S_i au temps t.
$\alpha_t(i)a_{i,j}$ est la probabilité de la séquence d'observations $o_1, o_2, ..., o_t$ au temps t sachant l'état caché S_i au temps t et l'état caché S_j au temps $t+1$.
$\alpha_t(i)a_{i,j}b_j(o_{t+1})$ est la probabilité de la séquence d'observations $o_1, o_2, ..., o_t$ au temps $t+1$ sachant que l'état caché S_i au temps t et l'état caché S_j au temps $t+1$
$\alpha_t(i)a_{i,j}b_j(o_{t+1})\beta_{t+1}(j)$ est la probabilité de la séquence d'observations $o_1, o_2, ..., o_t$ sachant l'état caché S_i au temps t et l'état caché Sj au temps $t+1$
On note Q, l'ensemble des possibilités d'états cachés.

La deuxième question correspond au problème du décodage. On connaît les paramètres de HMM et une sequence d'observations. On a également deux façons pour le résoudre. La première est un algorithme d'approximation. On choisit la plus grande possibilité de S_i à chaque temps t, on a une sequence de transitions. Cette méthode est localement optimale (optimale pour chaque temps) mais non optimal globalement car prédire on peut prédire un état caché qui n'a pas lieu dans le cas réel. Une autre méthode basé sur l'algorithme de Viterbi.

Enfin, la troisième question est liée au problème d'apprentissage. Selon le jeu de données d'entraînement, on a deux façons de traiter le problème :

— Lorsque le jeu de données d'entraînement est composé des séquences d'observations et des séquences d'états associées, on utilise la méthode maximum de vraisemblance. C'est la méthode employée dans notre travail.
— Lorsque le jeu de données d'entraînement ne présente aucun état associé aux séquences d'observations, on peut résoudre le problème par algorithme de Baum-Welch. Il s'agit d'une méthode apprentissage non-supervisé. Le jeu de données entraînement est défini par $D = (\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_N)$ où les $\mathbf{O}$ sont les variables d'observations. La séquence d'états cachés est notée I. HMM peut être vu comme un modèle de probabilités avec des variables cachées :

$P(\mathbf{O};\lambda) = \sum_{\mathbf{I}} P(\mathbf{O}|\mathbf{I};\lambda)P(\mathbf{I};\lambda)$, Ici, λ est un paramètre du HMM qui peut être appris avec l'algorithme EM.

Estimation des paramètres du modèle

A (transProbs) : A_{ij} est la fréquence du label I au temps précédent transfere à label j au temps suivant, par le different donnés entraînement , ce matrice va changer. Parce qu'on note label de punctuation "," et "." est "O" , ils ne sont pas un bonne état caché pour la sequence d'états. Chaque fois je compte un phrase terminant à punctuation "," et "." ,c'est-à-dire après c'est deux punctuation, une nouvelle phrase commence, il y a aussi une nouvelle séquence d'états.

$$\hat{a}_{i,j} = \frac{A_{i,j}}{\sum_{u=1}^{Q} A_{i,u}}$$

$$i = 1, 2, ..., Q; j = 1, 2, ..., Q$$

B (émissionProbs)B_{jk} est la fréquence de l'observation k à l'état jIci on calcule toutes les features dans notre jeu de données(entraînement test) car il y a des mots qui existent dans le jeu de données de test, mais n'existent pas dans l'entraînement.

$$\hat{b}_j(k) = \frac{B_{j,k}}{\sum_{v=1}^{V} B_{j,v}}$$

$$j = 1, 2, ..., Q; k = 1, 2, ..., V$$

π_i (startprobs) : π est la fréquence du label i qui est le premier mot dans chaque phrase :

$$\hat{\pi}_i = \frac{C_i}{sum_{j-1}^{Q} C_j}$$

$$i = 1, 2, ..., Q$$

2.4.3 Conditional Random Field

Conditional Random Field (Champs Aléatoires Conditionnels) est présenté en 2001 par Lafferty et al. Ce modèle nous a aidé à découvrir en plus le domaine de l'analyse de séquences. Dans les années précédentes, les modèles les plus utilisés sont les Modèles de Markov Cachés (HMM) pour l'analyse de séquences et des autres approches dans des différents domaines comme la segmentation, la classifiction, la détection, etc.

En géneral, CRF est vu comme un modèle discriminant (au même titre que la régression logistique) qui modélise les limites décisionnelles entre les différentes classes. En comparant avec les modèles génératifs comme le classifieur Bayésien Naïf ou HMM, CRF ne repose pas sur l'hypothèse forte d'indépendance des observations entres elles conditionnellement aux états associés. Ce modèle aide à résoudre le problème de biais de label des MEMM (Maximum Entropy Markow Model) ou les MMC pour l'analyse de la langue.

Avant de présenter les propriétés de CRF, nous définissons quelques notations principales [**?**] :
$X = x_1, x_2, ..., x_T$ est une séquence de T observations (séquence de T mots)
$Y = y_1, y_2, ..., y_T$ est une séquence des T étiquettes associés aux observations de X (séquence d'entités pour chaque mot dans X)

L est l'ensemble des étiquettes possibles (les valeurs d'entité possibles pour les y_t)
O est l'ensemble des observations (les valeurs possible pour x_t)

Le modèle CRF est définit par cette formulation :

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, x, t)\right) = \frac{1}{Z(X)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, x_t)\right)$$

La probabilité qu'une séquence d'étiquette particulière Y sachant la séquence d'observation X est obtenue par une combinaison linéaire de poids (weight) λ_k associés à des fonctions f_k. Les poids λ_k peuvent être interprétés comme l'importance ou bien comme la fiabilité de l'information apportée par la fonction f_k. Ces fonctions s'appellent les fonctions des caractéristiques (Features functions) : $f(y_{t-1}, y_t, x, t)$ qui expriment les caractéristiques de la séquence de l'observation présente. Par exemple, dans le cas de l'annotation Part-of-speech (POS tagging) :

$$f(y = \text{NOUN}, x_t) = \begin{cases} 1 & \text{si } y_{t-1} = \text{NOUN et } y_t = \text{VERB} \\ 0 & \text{sinon} \end{cases}$$

Les fonctions de caractéristiques sont définies par les utilisateurs car elles prennent en compte les séquences de variables de caractéristiques (Features), ceux qui reflètent la connaissance de l'utilisateur dans le domaine étudié. Comme CRF ne se base pas sur l'hypothèse d'indépendance des x_t entre elles conditionnellement à leurs y_t, il permet de définir tout un ensemble de caractéristique contextuelles.

Finalement, notons que la constante de normalisation $Z(x)$ - un terme de normalisation pour que la somme des proba somme à 1 :

$$Z(x) = \sum_{y} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_{t-1}, y_t, x_t)\right)$$

Les paramètres $\theta = \{\lambda_k\}$ du modèle CRF sont estimés par la maximisation de la vraisemblance. En principe, ceci peut être réalisé de la même manière que pour la régression logistique, ce qui n'est pas surprenant étant donné la relation entre ces 2 modèles. Cependant, CRF tend à avoir plus de paramètres et une structure plus complexe que celui d'un simple classifieur, donc il est plus coûteux à entraîner. Avec un jeu de données iid training data $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ où chaque $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, ...x_T^{(i)}\}$ est une séquence d'inputs et chaque $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, ...y_T^{(i)}\}$ est une séquence de prédictions désirée.
La Log-vraisemblance conditionnelle est présentée :

$$l(\theta) = \sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}$$

Une manière de comprendre la vraisemblance conditionnelle $p(y|x;\theta)$ est d'imaginer de la combiner avec $p(y|x;\theta')$ pour former $p(y|x)$. Ensuite, lorsqu'on optimise la log-vraisemblance de $p(y, x)$:

$$\log p(y, x) = \log p(y|x;\theta) + \log p(x;\theta')$$

La valeur de θ' n'affecte pas l'optimisation de θ. Si on ne veut pas estimer $p(x)$, on peut simplement abandonner la deuxième terme à droite, ce qui nous donne l'expression de la-log vraisemblance conditionnelle.

Après avoir substitué la formulation du modèle CRF dans la vraisemblance, on obtient :

$$l(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N} \log Z(x^{(i)})$$

C'est souvent le cas où on a un grand nombre de paramètre. Pour éviter le sur-apprentissage, on utilise la régularisation qui est un pénalty sur les vecteurs de poids dont la norme est trop large. Nous utilisons dans notre projet, le package `CRFsuite` qui s'implémente 2 algorithmes d'entraînement "$lbfgs$" (descente de gradient utilisant le méthode de L-BFGS) et "$l2sgd$" (descente de gradient stochastique pour la régularisation L2).

Chapitre 3

Expériences

Dans ce chapitre, nous concentrons à présenter les étapes importantes de pré-traitement du corpus GENIA à fin d'extraire les informations nécessaires pour apprendre un classifieur capable de détecter automatiquement des termes (des entités) appartenant à certaines catégories :

— Pré-traiter des documents du corpus xml de façon à enlever les méta-données et à représenter les textes sous forme numérique selon les besoins des classifieurs.
— Les données possèdent parfois plusieurs labels, dans notre tâche de classification, on fait un choix sur le nombre de classes. On regroupe des classes entre elle (ontologie).
— Les méthodes de classifications testées (Les méthodes testées, les features choisis, les options de paramètres).
— Les résultats obtenus.

3.1 Corpus GENIA

Corpus GENIA : corpus développé pour le projet GENIA dont l'objectif est d'explorer les méthodes d'extraction d'information et de text mining spécifiques au domaine de la science médicale. Le corpus peut ainsi servir de référence pour la communauté scientifique dans les tâches citées ci-dessus. Le corpus GENIA est composé d'un sous-ensemble d'éléments d'articles de la base de données Medline spécifiques aux réactions biologiques impliquées dans les « transcriptions factors in human bloods cells ». Ainsi pour chaque article, seuls les titres et résumés ont été collectés à partir de requêtes sur l'interface web de PubMed.

Table 3.1 – Statistiques sur le corpus GENIA

	Nombre		Moyenne
Abstract	2000		
Phrase	18545	9.27s/a	
Mot	436967	218.48 w/a	23.56 w/s

Le corpus GENIA est important pour 2 raisons majeures : premièrement, il fournit la plus large source d'annotation des données d'entraînement pour la tâche NE dans la domain de biologie moléculaire et deuxièmement, pour la classification.

La valeur principale du corpus GENIA provient des annotations : tous les résumés et même ses titres sont annotés par deux experts de la domaine pour les termes biologies importantes. Ces termes sont annotées sémantiquement avec les descripteurs présentés dans Figure 3.2.

Principalement, les termes peuvent être annotées simplement en insérant les balises XML comme dans l'exemple Figure 3.1. Dans cet exemple, les 3 termes *IL-2 gene*, *IL-2 gene transcription* et *T cells* apparaissent dans la phrase et sont encadrées dans les balises par les tags **cons**. Cependant, il existe des cas particuliers,

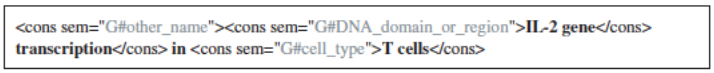

FIGURE 3.1 – Exemple d'annotation.

par exemple, le text *"CD2 and CD25 receptors"* désigne 2 termes *CD2 receptors* et *CD25 receptors* mais le terme *CD2 receptors* n'apparaît pas dans le text. Dans ces cas, l'annotation devient plus complexe.

La table 3.2 présente les statistiques sur les annotations du corpus, notons qu'il existe pas mal de tags **cons** qui n'a pas de sens, c'est à dire des tags **cons** sans valeurs **sem** qui indique la classe du terme annoté. Nous avons décidé d'ignorer ces types d'annotations dans notre projet car il ne nous apporte pas d'information sur la classe du terme et il sera impossible les classifier.

TABLE 3.2 – Statistiques sur les annotations

	Nb de cons	Nb de termes
Simple	89862	89862
Complexe	1583	3431
Sans sens	5137	0
Total	96582	93293

Nous avons utilisé le pakage **xml2** de R de façon à extraire tous les informations dans les tag **cons** : les valeurs des `lex` (le terme avec l'espace remplacé par underscore) et `sem` (la classe du terme) et les stocker dans un dataframe. Par exemple, le text comme Exemple 3.1 sera traité pour former un tableau comme Table 3.3

Listing 3.1 – Exemple de XML

```
<cons lex="IL-2_gene_expression" sem="G#other_name">
        <cons lex="IL-2_gene" sem="G#DNA_domain_or_region">IL-2 gene</cons>
        expression
</cons> and
<cons lex="NF-kappa_B_activation" sem="G#other_name">
        <cons lex="NF-kappa_B" sem="G#protein_molecule">NF-kappa B</cons>activation
</cons> through
<cons lex="CD28" sem="G#protein_molecule">CD28</cons>
requires reactive oxygen production by
<cons lex="5-lipoxygenase" sem="G#protein_molecule">5-lipoxygenase</cons>.
```

TABLE 3.3 – Exemple de tableau après l'extraction

lex	sem
IL-2_gene_expression	G#other_name
NF-kappa_B_activation	G#other_name
CD28	G#protein_molecule
5-lipoxygenase	G#protein_molecule

3.2 Sélection des Catégories d'Entités Nommées

GENIA Ontologie : l'ontologie de GENIA contient 47 catégories regroupés par 3 grands concepts : **Biological source**, biological substance et other.

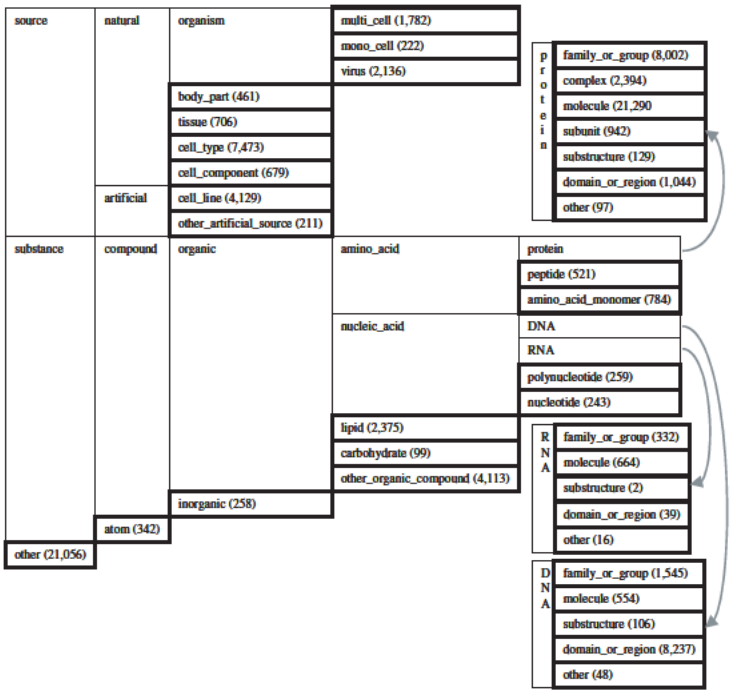

FIGURE 3.2 – Ontologie du corpus GENIA et les statistiques.

Les concepts encadrés en noir sont des concepts terminaux et qui forment des tags pour les anntations sémantiques (`sem`). Après avoir construit un dataframe comme Table 3.3, nous observons les tags `sem` qui apparaissent les plus fréquents dans le corpus.

TABLE 3.4 – Exemple de tableau après l'extraction

sem	Nb	sem	Nb	sem	Nb
G#other_name	11353	G#protein_molecule	4528	G#DNA_domain_or_region	4259
G#protein_family_or_group	2951	G#cell_line	2262	G#cell_type	2008
G#other_organic_compound	1186	G#DNA_family_or_group	890	G#protein_complex	710
G#protein_domain_or_region	659	G#multi_cell	512	G#DNA_molecule	408
G#virus	391	G#tissue	384	G#protein_subunit	358
G#RNA_molecule	346	G#lipid	331	G#peptide	266
G#cell_component	232	G#body_part	194	G#polynucleotide	185
G#amino_acid_monomer	177	G#RNA_family_or_group	138	G#other_artificial_source	126
G#DNA_substructure	88	G#mono_cell	87	G#protein_substructure	86
G#protein_N/A	74	G#atom	68	G#nucleotide	66
G#inorganic	62	G#carbohydrate	47	G#DNA_N/A	34
G#RNA_domain_or_region	31	G#RNA_N/A	9	G#RNA_substructure	2

Dans le cadre du projet, nous décidons de simplifier ces 36 classes en n'utilisant que 5 classes : DNA, PROTEIN, RNA, CELL TYPE, CELL LINE. La raison pour la quelle nous avons choisi ces 5 classes au lieu d'autres classes,c'est parce que, en observant table 3.4, trouvons que ces 5 classes sont les plus fréquentes classes dans les articles du corpus. Nous regroupons tous les tags qui appartiennent dans les

classes qu'on a choisi et changeons leurs noms vers leurs nom de classe, par exemple `G#protein_molecule` ou `G#protein_family_or_group` seront transformés comme `PROTEIN`.

3.3 Configurations et Méthodes d'Evalutions

Dans cette partie, nous décrivons les différents paramètres liés aux méthodes employées ainsi que les mesures d'évaluations.

Naive Bayes

Les paramètres qui entrent en jeu dans classifieur bayésien naïf sont le choix de la distribution des probabilités conditionnelles et les éventuels paramètres associés à cette distribution. Dans notre cas, nous utilisons une loi multinomiale et jouons sur le paramètre de lissage *alpha* expliqué en (2.4.1).

Hidden Markov Model

La méthode HMM a été ré-implémenté car la librarie HMM prennait beaucoup trop de temps (ou malcompréhension de la documentation). L'algorithme de viterbi a été utilisé pour trouver la séquence d'états cachés.

Conditional Random Field

Avec CRF, le package `crfsuite` de R nous propose les algorithmes d'entraînement "lbfgs" (Gradient descent using the L-BFGS method) et "l2sgd" (Stochastic Gradient Descent with L2 regularization term). Pour chaque algorithme, il nécessite des différents paramètres à fin de "tuner" le modèle.

- `max_iterations` (integer) : Le nombre maximum d'itérations pour optimiser l'algorithme. La valeur par défault dépend d'algorithme d'entraînement : Illimité pour "lbfgs" et 1000 pour "l2sgd".
- `c1` (double) : Le coefficient pour L1 régularisation. Par défaut, la valeur est 0 (pas de L1 régularisation). Si il est une valeur non-nulle, le package changera vers une autre méthode : Orthant-Wise Limited-memory QuasiNewton (OWL-QN). Ce paramètre supporte l'algorithme de "lbfgs".
- `c2` (double) : Le coefficient pour L2 régularisation. Par défaut, la valeur est 1.0. Ce paramètre supporte tous les deux algorithmes "lbfgs" et "l2sgd".

Méthode d'évaluation
Dans notre projet, nous évaluons les méthodes avec les mesures de rappel, précision et le F1-Score)

- *Précision* : Le rapport entre le nombre d'annotations correctement étiquetées et le nombre total d'annotations correctes du modèle d'apprentissage.

$$Précision = \frac{\text{Vraie Positive}}{\text{Vraie Positive} + \text{Fausse Positive}} = \frac{\text{Vraie Positive}}{\text{Total Positive Prédiction}}$$

- *Rappel* : Le rapport entre le nombre d'annotations correctement étiquetées et le nombre total d'annotations qui auraient dû être prédites.

$$Rappel = \frac{\text{Vraie Positive}}{\text{Vraie Positive} + \text{Fausse Négative}} = \frac{\text{Vraie Positive}}{\text{Total Positive Actuelle}}$$

- *F1-score* : Moyenne harmonique des valeurs de *Précision* et *Rappel*.

$$Rappel = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Nous utilisons une validation croisée pour mieux évaluer les méthodes en divisant le dataset par 5 (5 "folds"). 80% des données sont utilisées pour l'entrainement et les 20% restants sont utilisés pour le test. Nous nous sommes assurés que les données étaient iid pour chaque fold. Nous avons également vérifié que les proportions de classes entre folds soient équilibrées.

3.4 Résultats

Nous présentons les résultats des différents modèles pour différents ensembles de descripteurs dans le tableau 3.5. Nous n'avons pas eu le temps d'entraîner et de tester les modèles de HMM pour toutes les configurations en raison de la durée de calcul. Par ailleurs, les modèles HMM ne sont pas évalués en validation croisée 5-folds mais uniquement sur 1 fold en prenant 80 % de données d'entrainement et 20 % de données pour le test.
Globalement, les performances des différentes méthodes peuvent être classées par ordre croissant. La méthode bayésien naïf donne les moins bons résultats. Le modèle HMM bien que testé sur peu d'ensemble de descripteurs semble montrer de meilleurs résultats. Enfin, l'approche CRF dépasse dans presque tous les cas les résultats des deux autres méthodes.
La méthode Bayésien Naïf obtient la meilleure précision en utilisant les descripteurs *word* et *pos* en utilisant une fenêtre de 3 tokens. Le meilleur rappel est obtenu avec les descripteurs word + SVT. Le meilleur F1-Score est obtenu pour les descripteurs word obtenus sur une fenêtre de 3 tokens. Tous ces résultats sont obtenus en utilisant un lissage de Laplace égal à 1. Si ce paramètre est laissé à 0, donc sans lissage, les résultats chutent drastiquement.
Les HMM obtiennent le meilleur rappel en utilisant seulement les descripteurs word mais la meilleure précision en utilisant tous les descripteurs (sans fenêtre).
Enfin, c'est en utilisant tous les descripteurs sur une fenêtre de 3 qu'on obtient les meilleurs résultats pour CRF. Notons que les CRF ont été entrainés sur 100 itérations.

TABLE 3.5 – Les résultats par modèles

	Naive Bayes			HMM			CRF		
Test	Précision	Rappel	F1 score	Précision	Rappel	F1 score	Précision	Rappel	F1 score
word	53.10	66.43	59.02	71.43	**83.44**	76.97	78.94	82.85	80.85
lemma	50.16	62.71	55.73	X	X	X	73.38	78.67	75.93
word + pos	56.25	51.95	54.02	76.49	78.94	77.69	79.76	83.24	81.46
word + WFP	54.88	53.81	54.34	X	X	X	78.75	83.07	80.85
word + SVT	52.88	**66.92**	59.08	X	X	X	80.45	83.25	81.82
word + pref + suff	51.98	66.16	58.21	X	X	X	78.57	82.98	80.72
$\text{word}_k + word_{k-1} + word_{k+1}$	76.32	52.84	**62.44**	X	X	X	81.28	86.30	83.71
$\text{word}_k + word_{k-1} + word_{k+1} +$ $\text{pos}_k + pos_{k-1} + pos_{k+1}$	**74.73**	41.39	53.26	X	X	X	81.81	86.61	84.14
word + pos + pref + suff	55.87	47.23	51.19	X	X	X	79.57	83.37	81.42
word + pos + svt + WFP + prefixes + suffixes	55.42	51.97	53.64	**76.77**	79.39	**78.06**	80.20	83.34	81.74
Tous les features	74.13	39.09	51.11	X	X	X	**82.37**	**86.56**	**84.41**

3.5 Discussion

Les 3 méthodes testées ci-avant sont toutes basées sur des modèles bayésiens mais avec des différences sur leur complexité. Le modèle le plus simple est le modèle bayésien naïf qui fait une hypothèse très large que les variables des observations sont indépendantes entre elles. Or lorsqu'on analyse les descripteurs, l'intuition nous fait dire que cette hypothèse d'indépendance n'est généralement pas respectée. Le modèle n'est donc pas vraiment adapté aux données. Les modèles HMM et CRF apportent une plus grande complexité mais aussi plus de paramètres de modèles à estimer. Or disposant de beaucoup de données, on peut penser qu'un modèle plus complexe donnera de meilleurs résultats. C'est effectivement le cas ici.
On constate également que plus le modèle est simple et plus il est sensible au choix des descripteurs. Par exemple, le classifieur bayésien naïf est très instable selon les descripteurs qui sont utilisés et lorsqu'un critère augmente (précision par exemple), le critère complémentaire (rappel) diminue. Une remarque concernant le modèle bayésien naïf est que si l'on n'utilise pas de lissage, certaines observations qui ne sont pas apparues dans les données d'entrainement font tomber la probabilité à 0 et font chuter par la même occasion les performances du système.
Le modèle CRF est très dépendant du nombre d'itérations utilisées pour faire converger la solution. Lorsqu'on diminue le nombre d'itérations, on reçoit des résultats qui sont moins bons que ceux avec un nombre d'itérations supérieur. Le temps de calcul est plus élevé, mais on peut obtenir un gain significatif en performances. Les CRF sont, contrairement au classifieur bayésien naïf, plus stables selon les descripteurs utilisés. De plus, augmenter le nombre de descripteurs permet généralement d'augmenter les performances des CRF.

Conclusion

Pour ce projet, nous avons étudié des articles, sur les modèles comme NB, HMM, CRF et également ceux qui abordent le sujet de la reconnaissance des entités nommées pour le domaine de biomédical, por nous aider à comprendre le principe et de nous inspirer dans le processus de réaliser le projet. Nous avons traité le corpus GENIA de sorte à construire un vocabulaire des entités choisies et d'extraire les descripteurs intéressants pour la tâche de catégorisation. Nous avons également implélementé les modèles probabilistes comme Naive Bayes, Hidden Markov Model et Conditional Random Field et analyser voire comparer ses performances.

L'une des difficultés rencontrées dans le projet vient des erreurs de l'annotation du corpus. Il existe quelques erreurs dans l'annotation comme les entités sont mals écrites, les termes sont mals classées, plusieurs termes ont classées dans plus d"une classe d'entités, etc. Nous avons aussi un problème avec HMM qui nous nécessite de l'implémenter à la main sans utiliser les packages de R. ce qui passe beaucoup de temps à exécter. Le manque de connaissance dans le domaine biomédicale est également une difficulté. Il nous ralentie dans le choix des composants du système (choix des features, choix du modèle, etc.)

Pour améliorer, nous pensons à optimiser HMM car il prend trop de temps à exécuter et tester la performance HMM pour comparer avec les autres modèles. Nous pourrons également tester les modèles avec les POS spécifiques au domain de biomédicale ou tester avec les autres features qui portent plus d'informations intéressantes.

Glossaire

Les notations principales et les acronymes sont rassemblées ci-dessous.

CRF	Conditional Random Field
EM	Expectation-Maximization (Espérance-Maximisation)
HMM	Hidden Markov Model
LBFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
NB	Naive Bayes
POS	Part-Of-Speech
SGD	Stochastic gradient descent
SVM	Support Vector Machine
SVT	Special Verb Trigger
WFP	Word Formation Pattern
XML	Extensible Markup Language

Annexe

Liste des packages R utilisés :
`dplyr`
`quanteda`
`stringr`
`udpipe`
`xml2`
`crfsuite`
`e1071`
`data.table`