Étude comparative pour l'analyse de requêtes complexes dans le domaine du pneumatique

Abdenacer Keraghel**, Khalid Benabdeslem*
Bruno Canitia**

*Université Lyon 1, 43 Boulevard du 11 novembre 1918, Villeurbanne cedex 69622
khalid.benabdeslem@univ-lyon1.fr,
**Lizeo IT, 42 Quai Rambaud, 69002 Lyon
abdenacer.keraghel, bruno.canitia@lizeo-group.com
https://www.lizeo-group.com

Résumé. La recherche et l'extraction d'information dans une séquence textuelle (article scientifique, requête sur un moteur de recherche, post sur un forum de discussion) nécessitent un processus de reconnaissance d'entité nommée (REN). Cependant, les données disponibles pour effectuer ce processus varient selon leur nature et le domaine d'étude. Dans cet article, nous nous intéressons d'une part aux performances des systèmes de reconnaissance d'entité nommée, et d'autre part à leur complexité ainsi qu'à leur capacité à traiter des données de différentes origines. Une étude comparative entre plusieurs approches issues de l'état de l'art, appliquée à différents types de données (requêtes d'un moteur de recherche et posts de forums de discussion) liées au domaine du pneumatique, est proposée afin de sélectionner l'approche qui s'adapte le mieux à notre cas d'usage. Pour cela, nous nous appuierons sur les résultats de métriques d'évaluation des modèles d'apprentissage automatique telles que la précision, le rappel et la F-mesure.

1 Introduction

Une entité nommée est une unité linguistique référentielle, associée à des objets tels que : les noms de personnes (PER) comme « Barack Obama », les organisations (ORG) comme « Google », les lieux (LOC) comme « Paris », et autres (MISC) comme « Championship ». La reconnaissance d'entité nommée est une sous-tâche de l'activité d'extraction d'information qui consiste à identifier ces objets dans une séquence textuelle. Dans cet article, nous nous intéressons au domaine du pneumatique, dont les objets sont désignés par la marque ou le type de véhicule, la dimension ou la saison du pneu, etc.

Un système de reconnaissance d'entité nommée sans ambiguïté a pour but d'analyser (via une tokenisation [1] et un calcul de score de confiance) les données en entrée afin de détecter leurs classes associées. En l'espèce, un token [2] d'une séquence peut appartenir à plusieurs classes,

1. L'opération consiste à découper un texte en token, le plus souvent des mots.
2. C'est une unité lexicale.

d'où la nécessité d'une étape de désambiguïsation. Cette dernière a pour but de déterminer la bonne classe de ce token.

Après un examen de l'état de l'art, les principales approches de reconnaissance d'entité nommée existantes sont les suivantes (cf. fig. 1).

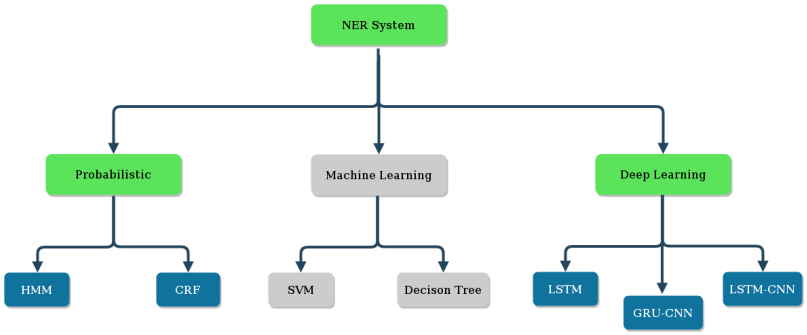


FIG. 1: Principales approches de reconnaissance d'entité nommée.

Deux familles ont ainsi retenu notre attention :

— La famille **Probabiliste** comme les chaînes de Markov cachées (**HMMs**) (e.g. Zhou et Su (2002); Zhao (2004)) et les **CRFs** (Conditional Random Fields) (e.g. McCallum et Li (2003)). Cette famille consiste à représenter le problème de reconnaissance sous forme d'un graphe non orienté, où les noeuds correspondent aux classes d'entités nommées et les arêtes aux probabilités de transition. Le but est de prédire la séquence de classes la plus probable à l'aide de l'ensemble des probabilités de transition. Cela correspond à la séquence de tokens observée (une requête sur un moteur de recherche dans notre cas) ;

— La famille du **Deep Learning** (e.g. Hochreiter et Schmidhuber (1997); Chiu et Nichols (2015); Hammerton (2003)). C'est une famille de réseaux de neurones de différents types : récurrents (LSTM, GRU) ou hybrides (hybridation des réseaux de neurones convolutifs (CNN) et récurrents). Elle consiste à apprendre des dépendances à long terme entre les différentes classes d'entités nommées possibles dans l'étape d'apprentissage. L'objectif de cette famille est de prédire la classe ultérieure d'entité nommée dans une séquence, à partir des classes précédentes.

La famille **Machine Learning** a été écartée car elle semblait globalement moins performante que les deux précédentes (Deep Learning et probabiliste). Les contraintes temporelles et celles imposées par le domaine applicatif (multi-support, hétérogénéité des univers d'entités nommées) ont rendu impossible toute expérimentation supplémentaire.

Dans cet article, nous présenterons dans un premier temps les familles d'approches que nous avons retenu afin de dresser une étude comparative. Nous présenterons par la suite nos jeux de données de test ainsi que notre protocole expérimental avant de faire une restitution des résultats.

2 Méthodes d'apprentissage pour la reconnaissance d'entité nommée

En se basant sur la complexité et la capacité des modèles à traiter des données de différentes origines, notre choix s'est porté sur les cinq approches suivantes (sans exhaustivité) :

2.1 Conditional Random Filed (CRF)

Également appelés "Champs Aléatoires Conditionnels" (CAC), les CRFs [3] sont des modèles graphiques statistiques non dirigés, dont un cas particulier est une chaîne linéaire qui correspond à un automate à états finis (HMM ou chaîne de Markov cachée), conditionnellement entraînée sur un dataset préalablement annoté par des experts. Ce genre de modèles convient bien à l'analyse séquentielle, et les CRFs en particulier se sont révélés utiles pour le marquage partiel de la parole (Lafferty et al. 2001) et la reconnaissance d'entité nommée pour les données de fil de presse (McCallum et Li (2003)). Ils ont également été appliqués à la tâche de reconnaissance des mentions de gènes et de protéines (McDonald et Pereira. 2004), avec des résultats préliminaires intéressants.

2.2 Long Short Term Memory (LSTM)

Ce modèle [3] est un cas particulier du *RNN* (réseau de neurones récurrent), la seule différence est qu'il contient des cellules mémoires lui permettant de garder l'information pour une durée théoriquement indéterminée. Il permet d'apprendre facilement des dépendances à long terme pour les tâches séquentielles d'étiquetage comme la reconnaissance vocale ou encore la reconnaissance d'entité nommée (Hammerton (2003)). Le but d'un LSTM est de créer une cellule mémoire pour chacun des mots de la séquence, cette dernière contenant le contexte du mot associé, qui permet facilement la reconnaissance de sa classe d'entité nommée.

2.3 GRU-CNN

Il s'agit d'une hybridation [3] de deux réseaux de neurones. Le premier est un RNN utilisant l'architecture GRU (Gated Recurrent Units) (Jiao et al. (2018)), qui vise à résoudre le problème de fuite du gradient (vanishing gradient) d'un RNN. GRU utilise ses portes d'oubli afin de décider quelles informations doivent être transmises à la sortie. Il est capable de conserver des informations d'un passé lointain, sans les perdre à travers le temps, ce qui est pertinent pour la prédiction des classes d'entités nommées.

Le deuxième (CNN) (Krizhevsky et al. (2012)) est un réseau de neurones à convolution (Convolutional Neural Network) dont le but est d'extraire une matrice de caractéristiques de taille fixe pour chaque entrée et de filtrer le bruit à l'aide d'un produit de convolution. L'objectif de cette hybridation est de pouvoir garder à la fois l'historique et les caractéristiques d'un mot donné, qui vont à leur tour nous permettre de repérer facilement les classes d'équivalences (mots qui appartiennent à la même classe).

3. Notre implémentation ne peut pas être diffusée pour des raisons de propriété intellectuelle.

Analyse de requêtes complexes.

2.4 LSTM-CNN

Il s'agit d'une hybridation [4] des deux réseaux de neurones CNN et LSTM précédemment définis. L'objectif est le même que celui du GRU-CNN, consistant à conserver sans perte des caractéristiques sur chaque mot de la séquence ainsi que son contexte (son historique du passé au futur). Cela permet d'identifier et de prédire facilement la classe d'entité nommée correspondante. Ce type de modèle convient bien à la détection d'anomalies dans les séries temporelles, et aussi à la reconnaissance d'entité nommée (Chiu et Nichols (2015)) pour les données de fil de presse ainsi que les séquences informelles [5].

2.5 BLSTM-CNN

Ce modèle est basé sur le LSTM-CNN avec la fonctionnalité bidirectionnelle [6] (Chiu et Nichols (2015)). Le but de cette fonctionnalité est de pouvoir conserver l'historique d'un mot donné dans les deux sens, du passé au futur et du futur au passé. Cela va renforcer la décision du modèle au niveau de la prédiction des classes d'équivalences d'entités nommées.

3 Sources de données

Afin de réaliser notre étude comparative, nous avons utilisé les deux jeux de données suivants :

3.1 CONLL-2003

Il s'agit d'un jeu de données public [7], contenant des articles d'actualité, découpées en phrases sous la forme de posts. Il est utilisé pour étalonner un modèle d'apprentissage automatique sur la tâche de reconnaissance d'entité nommée. Il se compose de huit fichiers couvrant les deux langues : allemand et anglais. Dans notre cas, nous n'avons utilisé que l'anglais. Pour chaque langue, nous disposons de 3 fichiers principaux : apprentissage, test, et validation. Les deux tableaux suivants illustrent la description et la volumétrie (Sang et De Meulder (2003)) de chacun d'entre eux.

Fichiers	Nb d'articles d'actualité	Nb de phrases	Nb de tokens
Apprentissage	946	14 987	203 621
Test	231	3 684	46 435
Validation	216	3466	51 362

TAB. 1: Description des différents fichiers du dataset CONLL'03 (anglais).

4. Code adapté de : https ://github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs.
5. Séquences textuelles contenant des abréviations comme les tweets, les blogs, les posts de forums de discussion, etc.
6. Code adapté de : https ://github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs.
7. https ://www.clips.uantwerpen.be/conll2003/ner.tgz.

Fichiers / ENs	LOC	MISC	ORG	PER
Apprentissage	7140	3438	6321	6600
Test	1668	702	1661	1617
Validation	1837	922	1341	1842

TAB. 2: Nombre d'entités nommées par fichier.

3.2 Requêtes provenant du domaine de pneumatique

Le domaine du pneumatique est un domaine particulier, aux entités nommées spécifiques. Ces dernières sont classées par domaine, le tableau ci-dessous illustre ces différents domaines ainsi que leurs classes d'ENs associées.

Domaine	Classe Entité Nommée	Exemple
Vehicle	Vehicle-Maker	Volkswagen
	Vehicle-Segment	Golf
	Vehicle-Cartype	Golf 4
	Vehicle-Motor	Hdi 1.6
Pattern	Pattern-Brand	Pirelli
	Pattern-Product	Sp 22
	Pattern-Name	AMH2
	Patter-Season	Winter
Dimension	Dimension-Complete	255/62 R15 96h
	Dimension-GeoBox	255/62 R15
	Dimension-Diameter	R15
Dealer	Dealer-Name	Norauto
Localisation	Localisation-City	Lyon

TAB. 3: L'ensemble des domaines et des classes d'entités nommées du domaine du pneumatique.

L'un des problèmes majeurs que nous avons rencontré lors de la réalisation de notre étude comparative est l'insuffisance de la quantité de données étiquetées disponibles provenant du pneumatique. Nous disposions en interne d'un jeu de données restreint contenant quelques centaines de requêtes (491) en français, émises par des utilisateurs d'un comparateur en ligne de prix de pneumatiques, annotées par les experts du domaine. Ce volume de données reste insuffisant pour une tâche d'apprentissage et de validation des modèles de l'état de l'art. A titre palliatif, nous avons décidé de mettre en place un processus de génération de données à base d'apprentissage automatique sur les données existantes. Pour ce faire, plusieurs approches ont été envisagées telles que :

— **La loi jointe** : c'est une loi de probabilités à plusieurs variables. Le but est d'estimer les paramètres des transitions entre chaque EN et les autres. Par exemple, la probabilité d'avoir un Vehicle-Maker après une Localisation-City est de 0.02. Cette approche est adaptée, mais semble très sensible à la pauvreté du jeu de données d'apprentissage, ce qui est notre cas ;

— **Les réseaux de neurones récurrents** (Sutskever et al. (2011)) : ce sont des modèles basés sur les caractères (Character Level Model), permettant de prédire le prochain caractère dans une séquence. Ces modèles ne sont pas adaptés à notre cas, car nous allons avoir trop de bruit au niveau des séquences générées (séquences incorrectes) ce qui implique une mauvaise annotation ;

Analyse de requêtes complexes.

— **Les réseaux bayésiens** (Young et al. (2009)) : ce sont des modèles graphiques probabilistes, basés sur un ensemble de probabilités conditionnelles (paramètres) estimées à partir d'une étape d'apprentissage sur un jeu de données. Ils sont très similaires à la loi jointe, sauf que ces derniers peuvent injecter des connaissances expertes lors de l'étape d'apprentissage. Ces modèles sont très adaptés à notre cas, notamment en raison du fait que nous disposons de quelques connaissances métier ainsi que des relations entre les différentes classes d'entités nommées *via* un dictionnaire contenant l'univers du pneumatique disponible en interne. Pour cela nous allons étudier ces derniers en détail.

Description d'un réseau bayésien. Un réseau bayésien est un modèle graphique acyclique (de type génératif) d'une distribution de probabilités communes à un ensemble de variables. Le réseau bayésien est composé de deux composantes : une structure graphique et un ensemble de probabilités conditionnelles qui représentent les paramètres du modèle. La structure graphique est un ensemble de noeuds, où chacun représente une variable discrète ou continue dans le jeu de données d'apprentissage. Ces derniers sont liés entre eux par des arcs orientés qui représentent les dépendances. Si un arc entre deux noeuds est présent, un lien de type noeud père et noeud fils est établi ainsi qu'une association d'une distribution de probabilités conditionnelles pour chaque noeud.

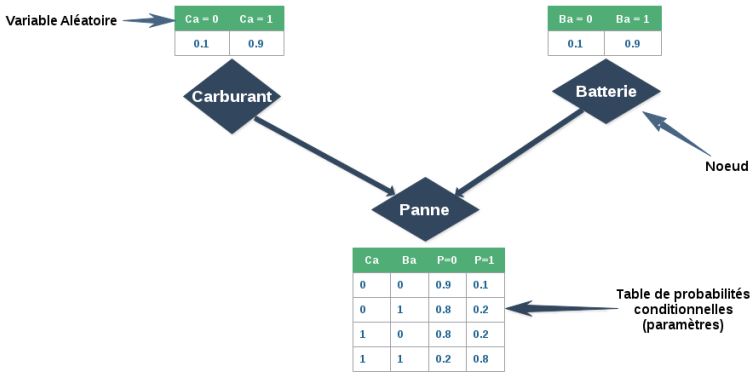


FIG. 2: Exemple d'un réseau bayésien avec les tables de paramètres.

Les rectangles bleus sur la figure 2 représentent les noeuds du graphe (variables), chaque noeud possède sa propre table de paramètres (distribution de probabilités conditionnelles) calculée à partir de lui sachant ses parents. Ex : les paramètres du noeud Panne sont calculés à partir de sa distribution de probabilités sachant les noeuds Carburant, Batterie, ce qui donne la formule suivante : $P(Panne = 1|Ba = 1, Ca = 1)$ = 0.8, etc. À l'aide de cette formule, nous pouvons régénérer la table de paramètres du noeud Panne par le fait de remplacer le couple (Carburant, Batterie) par les valeurs associées.

Dans notre cas, les variables sont les classes d'entités nommées telles que Vehicle-Maker, Pattern-Brand, etc. Les tables de paramètres sont les probabilités d'existence ou non (1=existe, 0=n'existe pas) d'une entité nommée sachant les autres (celles en relation directe avec elle) dans une séquence de texte (requête ou post).

Génération de données avec un réseau bayésien. A titre de rappel, les réseaux bayésiens sont constitués d'un ensemble de variables aléatoires $V = \{V_1, V_2, ..., V_n\}$, et d'un ensemble de paramètres $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ calculés à l'étape d'apprentissage. Le nombre de paramètres m = $\sum_{i=1}^{|V|} 2^{nb_i}$, où nb_i est le nombre de parents du noeud v_i. Ces deux ensembles définissent une distribution de probabilités (Forbes (2016)) P sur V qui se factorise comme suit :

$$P(V) = \prod_{v_i \in V} P(v_i | parents(v_i)). \tag{1}$$

La génération de données par un réseau bayésien se fait donc par imputation multiple (Young et al. (2009)), i.e pour chaque observation (sample), nous faisons l'imputation de chaque variable à l'aide des autres (processus d'inférence), en commençant par celle qui a le moins de parents. Ex : pour l'exemple illustré par la figure 2, nous avons 3 variables (Panne, Carburant, Batterie). Pour générer une observation (ligne dans un dataset), nous commençons par imputer les variables Carburant et Batterie (moins de parents dans le graphe). En prenant par exemple la variable Carburant, nous avons 90% pour 1 et 10% pour 0, n'importe quel processus de simulation de variables aléatoires selon une distribution de probabilité peut le faire. En supposant que nous voulons obtenir 1, pour calculer la valeur de Batterie, nous devons calculer par inférence la probabilité $P(Ba|Ca = 1)$ à l'aide de la loi jointe donnée par la formule (1), et ainsi de suite pour Panne. En répétant ce processus plusieurs fois, avec à chaque fois des valeurs différentes issues d'un processus de simulation, nous remplissons un dataset dont les lignes contiennent des 0 et des 1. À l'issue de ce processus, nous traitons chaque ligne de ce dataset, en remplaçant chaque variable contenant 1 (existe) par une vraie valeur tirée aléatoirement d'un dictionnaire contenant l'univers du pneumatique (toutes les marques de voitures, dimensions de pneus, etc). Enfin, un ajout de bruit et une permutation bien étudiée ainsi qu'une concaténation des colonnes est appliquée sur chaque ligne. La figure ci-dessous illustre le processus d'association des vraies valeurs sur un exemple donné. L'une des séquences possibles après une concaténation des colonnes et un ajout de bruit est la suivante : « **je cherche un pneu Michelin pour ma Renault clio** ».

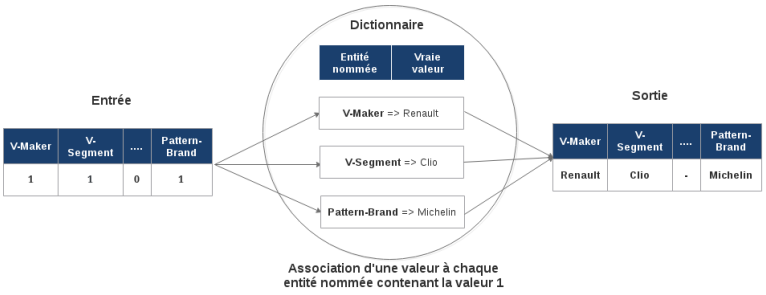


FIG. 3: Processus d'association des vraies valeurs à l'aide d'un dictionnaire contenant l'univers du pneumatique.

Ce processus de génération nécessite un modèle pré-entraîné (structure + paramètres) sur

un jeu de données. Pour ce faire nous avons utilisé le framework *PyAGrum* [8] (disponible en *Python*) pour entraîner plusieurs modèles (sur les 491 requêtes), en utilisant des algorithmes supportés par ce dernier, afin de sélectionner le meilleur en se basant sur des métriques d'évaluation (précision, rappel, F-score). Il n'existe pas de méthodes exploitables pour évaluer un dataset généré par un réseau bayésien, nous avons donc calculé pour chaque algorithme, des distances (*Hellinger*, etc) et divergences (*KL divergence*) entre la structure du modèle entraîné sur les données réelles et celle du modèle entraîné sur les données générées. Enfin, nous avons pris le modèle pour lequel la structure reste inchangée. Le tableau ci-dessous montre les algorithmes utilisés dans la comparaison ainsi que les scores obtenus, avec en gras les meilleurs.

	klPQ	errorPQ	klQP	errorQP	hellinger	bhattacharya	jensen-shannon	rappel	précision	f-score	dist2opt
K2	**0.001486**	0.000000	**0.001263**	0.000000	**0.021086**	**0.000222**	**0.000309**	**0.986301**	**1.000000**	**0.993103**	**0.013699**
GHC	0.004978	0.000000	0.005196	0.000000	0.040392	0.000816	0.001123	0.853333	0.984615	0.914286	0.147471
3off2	0.036641	0.000000	0.042804	0.000000	0.103878	0.005410	0.007088	0.870130	0.943662	0.905405	0.141563

TAB. 4: Scores de comparaison entre les 3 algorithmes « *K2, GHC, 3off2* » [9].

En s'appuyant sur les résultats de comparaison tels que ; Hellinger, rappel, précision, F-score, entre les modèles entraînés sur données réelles et données générées par chaque algorithme illustrés par la Table 4, le meilleur algorithme est K2 (basé sur la maximisation de score). Ce dernier est très robuste et performant pour la tâche de génération de données. Nous allons donc l'utiliser dans la suite de notre étude afin de générer des séquences de type requêtes.

Étude des données générées par K2. Dans cette partie nous présentons une comparaison de la fréquence d'apparition (illustrée par les diagrammes en barre à droite de la figure 4) de chaque classe d'entité nommée, ainsi que de la taille des séquences (illustrée par les 2 courbes à gauche de la figure 4) entre les données réelles et celles générées par K2.

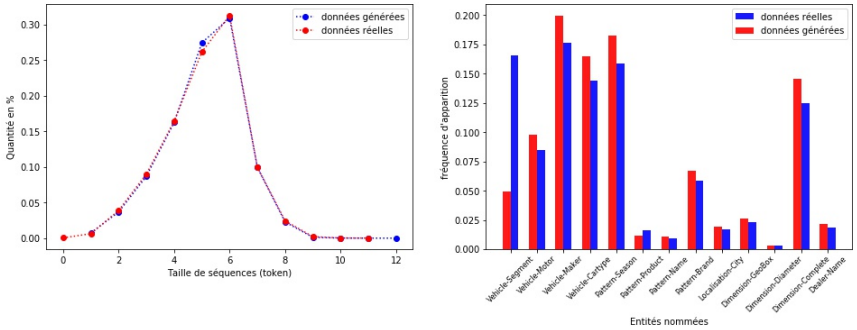


FIG. 4: Comparaison de la fréquence d'apparition des ENs et la distribution de la longueur des séquences, entre les données générées (1 000 000 requêtes) et réelles (491 requêtes).

Nous remarquons que la fréquence d'apparition de l'EN Vehicle-Segment est différente dans le jeu de données généré par rapport au réel, ce qui est dû à l'injection des connaissances

8. Un wrapper Python, qui fournit une interface de haut niveau à la partie d'aGrUM, permettant de créer, modéliser, apprendre, utiliser, calculer avec et intégrer des réseaux bayésiens et autres modèles graphiques.
9. Trois algorithmes d'apprentissage de structure d'un réseau bayésien disponibles en PyAgrum.

expertes (l'EN Vehicle-Cartype et Vehicle-Segment ne doivent pas apparaître ensemble dans une requête) avant l'étape d'apprentissage des paramètres du réseau bayésien. Cette injection est réalisée par le fait d'interdire les dépendances (arcs) entre les noeuds qui représentent ces 2 ENs. En revanche la fréquence d'apparition des autres ENs et la distribution de la taille des séquences sont parfaitement conservées, ce qui motive le choix de l'algorithme K2 pour la tâche de génération de données.

4 Étude comparative

4.1 Données utilisées dans l'évaluation des modèles

Dans cette partie, nous présentons la volumétrie ainsi que la description des jeux de données, utilisés dans l'évaluation des modèles.

Source	**Données provenant du domaine de pneumatique**		**CoNLL03**	
Type / Taille séquences	**Requêtes générées / [1 à 23] tokens**		**Posts/ [1 à 124] tokens**	
Nb séquences / Langue	24 000 / français		6703 / anglais	
ENs et fréquence d'apparition	Dealer-Name	1.9%	LOC	16.89%
	Dimension-Complete	13%		
	Dimension-GeoBox	2.4%		
	Dimension-Diameter	0.3%	MISC	7.51%
	Localisation-City	1.7%		
	Pattern-Brand	6.1%		
	Pattern-Product	1%	ORG	16.4%
	Pattern-Name	1.9%		
	Pattern-Season	16.4%		
	Vehicle-Maker	18.3%	PER	13.68
	Vehicle-Segment	4.6%		
	Vehicle-Cartype	15%		
	Vehicle-Motor	8.7%	O	45.52%
	O	9.7%		

TAB. 5: Description et volumétrie des deux jeux de données principaux utilisés dans l'évaluation des modèles.

Enfin, pour mener à bien des tests statistiques, nous avons créé huit autres jeux de données en découpant en deux le CONLL'03 et en six les requêtes. Nous obtiendrons ainsi les jeux de données : Requêtes A, Requêtes B, Requêtes C, Requêtes D, requêtes E, Requêtes F, CONLL'03 A et CONLL'03 B.

4.2 Méthodologie

Les cinq modèles issus de l'état de l'art ont d'abord effectué leur tâche de reconnaissance et de désambiguïsation d'entité nommée sur les différents jeux de données. Ils ont été ensuite évalués par des métriques d'évaluation comme la précision, le rappel et la F-mesure. La précision mesure le nombre d'ENs bien classées rapporté au nombre d'ENs total. Le rappel va mesurer le nombre d'ENs pertinentes retrouvées au regard du nombre d'ENs pertinentes total. La F-mesure est la moyenne harmonique de ces deux métriques. L'intervalle de confiance est donné par le score de *Wilson* (Wilson score interval), avec 95% de confiance.

4.3 Tests statistiques

Pour l'étude statistique nous avons utilisé les tests suivants (Demšar (2006)) :

Analyse de requêtes complexes.

1. **test de Friedman** : le but est de réfuter l'hypothèse suivante : «Tous les algorithmes ont des performances équivalentes». Un degré de confiance de 0.95 sera utilisé ;
2. **test de Nemenyi** : une fois le test de Friedman validé, nous pouvons utiliser celui-ci pour montrer qu'un algorithme ou un groupe d'algorithmes est significativement plus performant qu'un autre. Le degré de confiance utilisé sera de 0.1 à cause du faible nombre de jeux de données pour mener cette étude.

4.4 Résultats

Dans cette partie, nous présentons les résultats de notre étude comparative et statistique. Pour pouvoir comparer des approches de type apprentissage automatique et profond, la variété des jeux de données d'évaluation est très importante. Pour cela, nous avons utilisé les 10 jeux de données sus-mentionnés (*cf.* section 4.1) pour réaliser notre expérimentation.

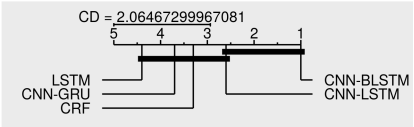
Le tableau ci-dessous illustre les résultats atteints par les cinq modèles, avec en gras ceux du meilleur.

Jeux de données	Métriques	CRF	LSTM	CNN-GRU	CNN-LSTM	CNN-BLSTM
Requêtes	**précision**	0.8072 +/- 0.005	0.7200 +/- 0.005	0.8193 +/- 0.005	0.8371 +/- 0.004	**0.9651 +/- 0.002**
	Rappel	0.8059 +/- 0.005	0.6421 +/- 0.006	0.7716 +/- 0.005	0.7903 +/- 0.005	**0.9622 +/- 0.002**
	F-mesure	0.8065 +/- 0.005	0.6788 +/- 0.006	0.7947 +/- 0.005	0.8132 +/- 0.005	**0.9636 +/- 0.002**
Requêtes A	**précision**	0.8310 +/- 0.012	0.7183 +/- 0.014	0.8202 +/- 0.012	0.8378 +/- 0.011	**0.9649 +/- 0.006**
	Rappel	0.8307 +/- 0.012	0.6412 +/- 0.015	0.7726 +/- 0.013	0.7912 +/- 0.012	**0.9620 +/- 0.006**
	F-mesure	0.8309 +/- 0.012	0.6776 +/- 0.014	0.7957 +/- 0.012	0.8138 +/- 0.012	**0.9634 +/- 0.006**
Requêtes B	**précision**	0.8033 +/- 0.012	0.7189 +/- 0.014	0.8186 +/- 0.012	0.8359 +/- 0.011	**0.9647 +/- 0.006**
	Rappel	0.8017 +/- 0.012	0.6407 +/- 0.015	0.7702 +/- 0.013	0.7888 +/- 0.013	**0.9622 +/- 0.006**
	F-mesure	0.8025 +/- 0.012	0.6776 +/- 0.014	0.7937 +/- 0.013	0.8117 +/- 0.012	**0.9634 +/- 0.006**
Requêtes C	**précision**	0.8027 +/- 0.012	0.7210 +/- 0.014	0.8208 +/- 0.012	0.8384 +/- 0.011	**0.9658 +/- 0.006**
	Rappel	0.8010 +/- 0.012	0.6427 +/- 0.015	0.7732 +/- 0.013	0.7915 +/- 0.013	**0.9627 +/- 0.006**
	F-mesure	0.8018 +/- 0.012	0.6796 +/- 0.014	0.7963 +/- 0.012	0.8143 +/- 0.012	**0.9643 +/- 0.006**
Requêtes D	**précision**	0.8010 +/- 0.012	0.7209 +/- 0.014	0.8191 +/- 0.012	0.8379 +/- 0.011	**0.9652 +/- 0.006**
	Rappel	0.7995 +/- 0.012	0.6424 +/- 0.015	0.7705 +/- 0.013	0.7901 +/- 0.013	**0.9622 +/- 0.006**
	F-mesure	0.8003 +/- 0.012	0.6794 +/- 0.014	0.7941 +/- 0.013	0.8133 +/- 0.012	**0.9637 +/- 0.006**
Requêtes E	**précision**	0.8030 +/- 0.012	0.7207 +/- 0.014	0.8183 +/- 0.012	0.8375 +/- 0.011	**0.9658 +/- 0.006**
	Rappel	0.8011 +/- 0.012	0.6425 +/- 0.015	0.7704 +/- 0.013	0.7894 +/- 0.013	**0.9627 +/- 0.006**
	F-mesure	0.8021 +/- 0.012	0.6794 +/- 0.014	0.7937 +/- 0.013	0.8128 +/- 0.012	**0.9643 +/- 0.006**
Requêtes F	**précision**	0.8034 +/- 0.012	0.7204 +/- 0.014	0.8187 +/- 0.012	0.8370 +/- 0.011	**0.9651 +/- 0.006**
	Rappel	0.8016 +/- 0.012	0.6430 +/- 0.015	0.7727 +/- 0.013	0.7908 +/- 0.013	**0.9622 +/- 0.006**
	F-mesure	0.8025 +/- 0.012	0.6795 +/- 0.014	0.7950 +/- 0.013	0.8133 +/- 0.012	**0.9637 +/- 0.006**
CONLL'03	**précision**	0.8465 +/- 0.009	0.7929 +/- 0.010	0.7528 +/- 0.010	0.7759 +/- 0.010	**0.8626 +/- 0.008**
	Rappel	0.8065 +/- 0.009	0.7517 +/- 0.010	0.7351 +/- 0.011	0.7666 +/- 0.010	**0.8627 +/- 0.008**
	F-mesure	0.8261 +/- 0.009	0.7718 +/- 0.010	0.7438 +/- 0.010	0.7712 +/- 0.010	**0.8627 +/- 0.008**
CONLL'03 A	**précision**	0.8457 +/- 0.012	0.7932 +/- 0.014	0.7521 +/- 0.015	0.7806 +/- 0.014	**0.8648 +/- 0.012**
	Rappel	0.8067 +/- 0.013	0.7548 +/- 0.015	0.7367 +/- 0.015	0.7762 +/- 0.014	**0.8704 +/- 0.011**
	F-mesure	0.8258 +/- 0.013	0.7735 +/- 0.014	0.7443 +/- 0.015	0.7784 +/- 0.014	**0.8676 +/- 0.011**
CONLL'03 B	**précision**	0.8473 +/- 0.012	0.7926 +/- 0.014	0.7534 +/- 0.015	0.7712 +/- 0.014	**0.8604 +/- 0.012**
	Rappel	0.8064 +/- 0.013	0.7486 +/- 0.015	0.7336 +/- 0.015	0.7572 +/- 0.015	**0.8552 +/- 0.012**
	F-mesure	0.8263 +/- 0.013	0.7700 +/- 0.014	0.7434 +/- 0.015	0.7641 +/- 0.014	**0.8578 +/- 0.012**

TAB. 6: Résultats de différentes approches étudiées sur les six jeux de données.

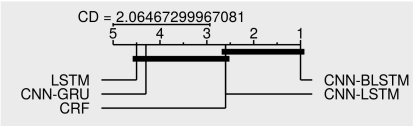
Nous constatons à l'aide des résultats illustrés par le tableau 6, que le rendement en termes de reconnaissance d'entité nommée de la famille Deep Learning est meilleure que celui de la famille probabiliste. Ainsi, nous remarquons que la fonctionnalité bidirectionnelle du modèle LSTM a un impact remarquable sur tous les types de données (requêtes ou posts CONLL'03).

Une fois le test de *Friedman* validé pour les différents modèles, nous avons procédé au test de *Nemenyi* avec pour résultat les figures suivantes.



(a) Nemenyi critical distance - Précision

(b) Nemenyi critical distance - Rappel



(c) Nemenyi critical distance - F-mesure

FIG. 5: Résultats des tests de Nemenyi.

Les résultats des tests de Nemenyi montrent que le modèle CNN-BLSTM est significativement plus performant que tous les autres, ce qui affirme les résultats du tableau 6.

5 Conclusion et perspectives

Dans cet article, nous avons établi un état de l'art sur la **reconnaissance d'entité nommée (REN)**, après avoir examiné les données de pneumatiques disponibles en interne. Nous avons ensuite présenté cinq modèles appartenant à deux familles d'approches *Probabiliste et Deep Learning*. Ces recherches nous ont permis de récupérer voire d'implémenter les architectures de modèles sélectionnés et de les adapter à notre cas d'étude. Faisant face à un volume de données insuffisant, nous avons mis en place un processus de génération de données afin de réaliser une étude comparative entre ces modèles. L'ensemble des développements réalisés a fait l'objet d'expérimentations à chaque étape afin de s'assurer que la qualité des résultats obtenus soit suffisante pour poursuivre. Notre étude comparative (*cf.* table 6 et figure 5) sur des données réelles (CONLL'03) et des données générées (requêtes générées par un réseau bayésien), a montré que la vision *Deep Learning* est significativement meilleure que la vision probabiliste. En perspective, nous supposons que la bonne performance de notre réimplémentation du CNN-BLSTM pourrait être encore meilleure en rajoutant un mécanisme d'évolution dans le temps tel que le *self learning*.

Références

Chiu, J. P. C. et E. Nichols (2015). Named Entity Recognition with Bidirectional LSTM-CNNs. *arXiv :1511.08308 [cs]*.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research 7*(Jan), 1–30.

Forbes, F. (2016). Modelling structured data with Probabilistic Graphical Models. *EAS Publications Series 77*, 195–219.

Analyse de requêtes complexes.

Hammerton, J. (2003). Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, Stroudsburg, PA, USA, pp. 172–175. Association for Computational Linguistics.

Hochreiter, S. et J. Schmidhuber (1997). Long Short-Term Memory. *Neural Comput. 9*(8), 1735–1780.

Jiao, Z., S. Sun, et K. Sun (2018). Chinese lexical analysis with deep bi-gru-crf network.

Krizhevsky, A., I. Sutskever, et G. E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105. Curran Associates Inc.

McCallum, A. et W. Li (2003). Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pp. 188–191.

Sang, E. F. et F. De Meulder (2003). Introduction to the conll-2003 shared task : Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sutskever, I., J. Martens, et G. E. Hinton (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024.

Young, J., P. Graham, et R. Penny (2009). Using bayesian networks to create synthetic data. *Journal of Official Statistics 25*(4), 549.

Zhao, S. (2004). Named Entity Recognition in Biomedical Texts Using an HMM Model. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, JNLPBA '04, pp. 84–87. Association for Computational Linguistics.

Zhou, G. et J. Su (2002). Named Entity Recognition Using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 473–480. Association for Computational Linguistics.

Summary

Searching and extracting information in a text sequence (scientific article, search engine query, discussion forum post) requires a named entity recognition (NER) process. However, the data available to carry out this process vary according to their nature and field of study. In this article, we focus on the performance of named entity recognition systems, on the one hand, and on their complexity and ability to process data from different origins, on the other. A comparative study between several state-of-the-art approaches, applied to different types of data (search engine queries and discussion forum posts) related to the tyre sector, is proposed in order to select the approach that best suits our use case. To do this, we will rely on the results of evaluation metrics of automatic learning models such as precision, recall, F-score.