UNIVERSITÉ DU QUÉBEC À MONTRÉAL

EXTRACTION D'ENTITÉS NOMMÉES PAR APPRENTISSAGE PROFOND

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

YACINE AMIRAT

JUIN 2020

REMERCIEMENTS

Tout d'abord, j'adresse mes remerciements à mon directeur de recherche, Aziz Salah, pour son temps, son encadrement et sa patience tout au long de ma maitrise.

J'adresse également mes remerciements à mes proches et mes amis : Kaouther, Abdelhak, Djamel, Chakib, Derder, Amine, Ahmed, et en particulier à Sara qui m'a apporté son support moral tout au long de mes études, et aux membres du laboratoire CRIA : Tan, Ange, Bilel, Michael et à tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

Je remercie aussi la fondation de l'UQAM, la faculté des sciences et le gouvernement du Canada, de m'avoir octroyé la bourse d'excellence et les bourses d'exemption des frais de scolarité majorés qui m'ont beaucoup aidé.

Au final, je voudrais remercier du fond de mon coeur mes parents, mon frère et ma soeur qui sont ma source d'inspiration et qui ont tant donné pour que j'arrive là où j'en suis.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

ACRONYMES

ACE Automatic Content Extraction.

AM Apprentissage Machine.

BERT Bidirectional Encoder Representations from Transformers.

CNN Convolutional Neural Network.

CPM Context du plongement de Mots Pré-entraîné.

CRF Conditional Random Field.

ELMo Embeddings from Language Models.

FN Faux Négatif.

FP Faux Positif.

GloVe Global Vectors for Word Representation.

IA Intelligence Artificielle.

IREX Information Retrieval and Extraction Exercise.

MUC Message Understanding Conferences.

PM Plongement de Mots Pré-entraîné.

POS Part-Of-Speech.

REN Reconnaissance des Entités Nommées.

RMC Représentation de Mots à partir de Caractères.

RNR Réseau de Neurones Récurrents.

UNK Unknown.

VP Vrai Positif.

NOTATION

a	Un scalaire (entier ou réel)
$\boldsymbol{a}$	Un vecteur
$\boldsymbol{A}$	Une matrice
a	Une variable aléatoire
a_i	élément i du vecteur $\boldsymbol{a}$, avec index qui commence de 1
$A_{i,j}$	élément i, j de la matrice $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	La ligne i de la matrice $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	La colonne i de la matrice $\boldsymbol{A}$
a_i	élément i d'une variable aléatoire $\mathbf{a}$
$\boldsymbol{A}^\top$	Le transposé de la matrice $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Le produit de Hadamard de deux matrices $\boldsymbol{A}$ et $\boldsymbol{B}$
$P(\mathrm{a})$	Une distribution de probabilité d'une variable
$f(\boldsymbol{x}; \boldsymbol{\theta})$	Une fonction de $\boldsymbol{x}$ paramétré par $\boldsymbol{\theta}$.
$\log x$	Logarithme naturel de x

$\mathrm{x}^{<1:T>}$	Une séquence x de longueur T
$\mathrm{x}^{<t>}$	Un élément d'une séquence x
$\mathbb{R}$	L'ensemble des nombres réels
$\exp(x)$	La fonction exponentielle de x
$\sigma(x) = \frac{1}{1+e^{-x}}$	La fonction sigmoïde de x
$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	La tangente hyperbolique de x
$Relu(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{sinon} \end{cases}$	L'unité de rectification linéaire de x

RÉSUMÉ

L'extraction des entités nommées consiste à détecter dans un texte les noms propres et les classifier dans des catégories, ou sous catégories prédéfinies telles que : nom de personne, lieu, adresse, etc. La tâche de la reconnaissance des entités nommées est souvent utilisée comme une étape de prétraitement dans d'autres applications de traitement automatique du langage naturel telles que la traduction automatique, la recherche d'information, la création d' agent conversationnel ou autres. Récemment, l'apprentissage profond a été appliqué dans le domaine de la reconnaissance des entités nommées, montrant des résultats prometteurs.

Principalement, les modèles de reconnaissance des entités nommées, basés sur l'apprentissage profond avec les réseaux de neurones récurrents, reposent sur la concaténation de différents types de représentations de mots. Ces représentations doivent contenir des caractéristiques discriminantes qui permettent la détection des entités nommées. Cependant, l'indisponibilité ou le manque d'information dans certaines de ces représentations nuit à la performance du modèle. De plus, ces modèles souffrent parfois de la dégradation de performance due au problème de disparition de gradient.
Dans le cadre de ce mémoire, nous visons à améliorer la performance des modèles de reconnaissance des entités nommées basés sur les réseaux de neurones récurrents. Nous proposons dans une première tentative pour remédier au problème de l'indisponibilité de représentations de mots par une somme pondérée entre deux différents types de représentations de mots. Deuxièmement, afin de diminuer l'effet de dégradation de performance connu dans les réseaux de neurones profonds, nous avons appliqué, dans un deuxième modèle, une connexion par saut inspirée du modèle ResNET (He *et al.*, 2015). À la fin, nous proposons dans un troisième modèle, une technique pour l'extraction de caractéristiques de mots à partir de leurs caractères. Cette technique est une combinaison de deux autres techniques (CNN (LeCun *et al.*, 1989), LSTM (Hochreiter et Schmidhuber, 1997)) utilisées dans la littérature.

MOTS CLÉS : entités nommées, apprentissage profond, réseaux de neurones récurrents, extraction d'information, traitement automatique du langage naturel.

INTRODUCTION

Introduction générale

Durant les dernières années, le monde a connu un essor exponentiel en matière de science, d'avancée technologique, ainsi qu'une grande prolifération des données. Dans ce contexte, l'apprentissage profond s'est vu propulsé au-devant de la scène scientifique orné d'un succès colossal, occupant désormais un espace de plus en plus important en tant qu'outil de recherche dans la plupart des domaines scientifiques, y compris dans le domaine de l'extraction de l'information, plus particulièrement dans le cadre de la reconnaissance d'entités nommées considérée comme primordiale dans une variété d'autres applications de traitement de langage naturel à l'instar de la recherche d'informations (Meng *et al.*, 2019) ou la traduction automatique (Balabantaray, 2010) etc.

La reconnaissance des entités nommées a fait sa première apparition lors de la sixième conférence MUC-6 (Grishman, 1996). Depuis, maintes entreprises ont été menées par la communauté d'extraction d'information afin de définir cette tâche. Les premiers travaux ont restreint la définition de la reconnaissance des entités nommées à reconnaître les noms propres en général tels que ceux de personnes, de lieux ou d'organisations (S. Coates-Stephens et al. 1992), (C. Thielen et al. (1995) et (G. Petasis *et al.*, 2000).
Nadeau et Sekine (2007) considèrent que les entités nommées sont les mentions de désignateurs rigides appartenant à des types sémantiques prédéfinis tels que définis par (Kripke, 1982). Ces désignateurs rigides peuvent être, selon Kripke (1982), des noms de personnes, de lieux, d'espèces ou de substances biologiques, etc. La

tâche de reconnaissance des entités nommées consiste à extraire ces mentions à partir du texte.

Plusieurs autres conférences se sont intéressées à cette thématique et ont défini leurs propres entités à identifier selon l'objectif de la tâche. Nous pouvons notament citer CoNLL-2003 (Sang et De Meulder, 2003) : personnes, organisations, lieux, autres ; ACE (Doddington *et al.*, 2004) : personnes, organisations, lieux, installations, armes, véhicules et entités géopolitiques ; IREX (Demartini *et al.*, 2010) : personnes, organisations, lieux, artéfacts, dates, heures, monnaies et pourcentages. Nonobstant toutes ces définitions, nous pouvons scinder les entités nommées en deux catégories principales : entités de base (personnes, organisations, etc.) et entités spécifiques au domaine (protéine, gène, etc.).

Un intérêt grandissant s'est formé autour des systèmes de reconnaissance d'entités nommées, agissant en tant que précurseur à la grande évolution, qui s'est établie au fil des années. Diverses techniques ont été mises en place pour la reconnaissance des entités nommées. De plus, des approches traditionnelles qui ne requièrent pas des données annotées ont été utilisées (Sekine et Nobata, 2004). Elles nécessitent en revanche un travail manuel conséquent, comme c'est le cas des systèmes à base de règles qui reposent sur des règles faites à la main. L'apprentissage automatique, à son tour, a pris part dans cette avancée avec des approches supervisées classiques, basées sur l'ingénierie de caractéristiques [1] discriminantes (p. ex., dictionnaires, connaissances sur la capitalisation de mots). Les approches d'apprentissage machine classiques ont réussi à améliorer les résultats de l'état de l'art. L'ingénierie de caractéristiques nécessite toutefois une intervention ma-

1. L'ingénierie des caractéristiques consiste à utiliser les connaissances du domaine pour définir des caractéristiques des données brutes via des techniques d'exploration de données. Une caractéristique est un attribut partagé par toutes les données sur lesquelles la prédiction doit être effectuée.

nuelle. Cette manipulation peut s'avérer coûteuse et chronophage. C'est ainsi que l'apprentissage profond fait son apparition par l'automatisation de l'exploration et l'extraction de ces caractéristiques (p.ex., les sensibilités orthographiques) depuis les données brutes. Contrairement aux algorithmes d'apprentissage automatique classiques, l'apprentissage profond agit d'une manière directe, autonome et dans un processus sophistiqué de bout en bout.

Récemment, l'application de l'apprentissage profond à la tâche de reconnaissance des entités nommées a fait l'objet de plusieurs publications scientifiques. En effet, depuis l'article de Collobert *et al.* (2011a) qui ont proposé un système de reconnaissance d'entités nommées presque indépendant de l'ingénierie manuelle de caractéristiques, un nombre important de travaux ont suivi. Nous pouvons citer à titre d'exemple : Lample *et al.* (2016), Ma et Hovy (2016), Chiu et Nichols (2015) et Peters *et al.* (2018). Cela a animé notre intérêt face à cette problématique et nous a poussé à engendrer des questionnements concernant les facteurs et les enjeux qui lui sont associés.

Les premiers travaux qui ont fait usage de l'apprentissage profond (p. ex., Collobert *et al.* (2011a), Turian *et al.* (2010), Lin et Wu (2009)) ont été marqués par l'utilisation des techniques de plongement de mots (*word embedding*) pré-entraîné de manière non supervisée sur des grands corpus de texte, et ce, afin de surmonter le problème du manque de données annotées pour l'entraînement. Ces approches utilisaient le plongement de mots en complément des données externes et de l'ingénierie manuelle des caractéristiques. En revanche, d'autres travaux comme Ma et Hovy (2016), Lample *et al.* (2016) et Peters *et al.* (2017) ont employé les représentations distributionnelles de mots, qui sont les plongements de mots pré-entraîné, sans utiliser de données externes. De plus, des représentations de mots à partir des caractères (Ling *et al.*, 2015) ont aussi été utilisées afin de capturer la sensibilité orthographique des mots (p. ex., la capitalisation).

Afin d'extraire la représentation de mots à partir de caractères, Lample *et al.* (2016), Liu *et al.* (2017) et d'autres ont opté pour un réseau LSTM (Hochreiter et Schmidhuber, 1997). Ils ont réalisé de très bons résultats compétitifs et stables. D'autre part, Ma et Hovy (2016), Chiu et Nichols (2015) Peters *et al.* (2018) ont utilisé un réseau à convolution (CNN) (LeCun *et al.*, 1989) pour l'extraction des représentations de mots à partir de caractères comme une alternative aux LSTM en raison de l'efficacité d'entraînement des CNN qui permet le calcul parallèle.
Le mécanisme d'attention [2] a été adopté aussi dans les systèmes de reconnaissance des entités nommées. Devlin *et al.* (2018) ont proposé un modèle basé sur les transformeurs (Vaswani *et al.*, 2017). Il s'agit de permettre au modèle de se concentrer plus sur des parties de la séquence. Un autre type de mécanisme d'attention a été proposé par Rei *et al.* (2016) qui consiste à décider sur lesquelles des représentations (la représentation de mots à partir de caractères ou le plongement de mots pré-entraîné) l'attention doit être portée. Mais, au final, le résultat fut médiocre en comparaison avec les approches simples (Lample *et al.*, 2016; Ma et Hovy, 2016).

Problématique

L'un des problèmes majeurs que nous avons observé dans les modèles de reconnaissance des entités nommées basés sur l'apprentissage profond est la représentation des mots par une simple concaténation de la représentation de mots à partir de caractères et le plongement de mots pré-entraîné. Cette approche simple pose un problème lorsque le plongement de mots pré-entraîné est indisponible pour certains mots. Ces mots sont souvent représentés par une représentation générique partagé. Cela donne lieu à des représentations finales longues et biaisées. En outre,

2. C'est un mécanisme qui permet au modèle de se focaliser sur une partie des données en entrée en leur attribuant des poids plus fort que les autres (Bahdanau *et al.*, 2014)

le manque d'information dans le plongement des mots rares représente un inconvénient majeur de la représentation distributionnelle de ces mots.(Rei *et al.*, 2016) D'autre part, l'entraînement des modèles basés sur l'apprentissage profond n'est pas toujours évident. Théoriquement, la superposition de plusieurs couches de réseaux de neurones peut être favorable à l'amélioration des performances de l'apprentissage. Cependant, il a été observé, dans la pratique qu'après une certaine limite de couches superposées, la performance du réseau commence à se saturer et finit par se dégrader.(Bengio *et al.*, 1994)

Dans la littérature, il existe deux solutions proposées pour l'extraction d'une représentation de mots à partir de caractères : la première est basée sur un réseau LSTM (p. ex., Lample *et al.* (2016)) et la deuxième est basée sur un réseau CNN (p. ex., Ma et Hovy (2016)). Chaque technique a montré ses avantages par rapport à l'autre. Néanmoins, aucune combinaison de ces deux techniques n'a été proposée auparavant dans la littérature.

Objectifs et contribution

Dans le cadre de notre travail, nous nous sommes intéressés aux modèles de reconnaissance des entités nommées basés sur les réseaux de neurones récurrents tels que ceux de Lample *et al.* (2016) et Ma et Hovy (2016). L'objectif principal de notre étude est de proposer une amélioration de ce type de modèle. Dans ce cadre, la contribution de notre travail peut être résumée dans les éléments suivants :

— Dans un premier modèle, nous allons essayer de remédier au problème de l'indisponibilité de représentations de mots pré-entraîné. Pour ce faire, nous allons adapter un mécanisme de somme pondérée qui offre au modèle la capacité de choisir la combinaison la plus avantageuse de deux types de représentations de mots (la représentation de mots à partir de caractères et le plongement de mots pré-entraîné) pour former une représentation finale des mots au lieu d'une simple concaténation, telle qu'utilisée dans les modèles

de Lample *et al.* (2016) et (Ma et Hovy, 2016).

— Le deuxième modèle proposé s'inspire de la technique de connexion par saut (He *et al.*, 2015) avec un mécanisme de somme pondérée en l'appliquant sur le modèle de Lample *et al.* (2016). Nous espérons par cela diminuer le problème de saturation des réseaux de neurones profond et améliorer la performance de ce modèle de base.

— Dans une tentative d'améliorer la représentation de mots à partir de caractères, nous avons essayé, dans notre troisième modèle, de combiner les représentations de mots à partir de caractères à base de CNN (Ma et Hovy, 2016) et celle à base de LSTM (Lample *et al.*, 2016).

Structure du mémoire

Le chapitre I introduit les différentes techniques de l'apprentissage profond qui sont utilisées dans les systèmes de reconnaissances des entités nommées afin de se mettre dans le contexte.
Le chapitre II résume l'état de l'art des systèmes de reconnaissance des entités nommées. Nous allons parcourir la liste des travaux les plus pertinents dans la littérature, en abordant brièvement dans un premier temps les approches classiques, puis nous allons étudier en profondeur les rouages d'un système de reconnaissance d'entités nommées basé sur l'apprentissage profond selon trois axes :

– Les représentations distributionnelles de mots (représentation de mots à partir de caractères, plongement de mots pré-entraîné)

– L'encodage du contexte.

– Le décodage ou la classification des entités.

Nous mettrons également en lumière quelques approches récentes basées sur l'apprentissage par transfert de connaissances.
Le chapitre III sera consacré à la présentation des modèles que nous proposons

pour la reconnaissance des entités nommées. Ces modèles sont basés sur l'apprentissage profond. Nous allons présenter en premier lieu deux modèles de base afin d'éclaircir notre contribution par rapport à ces deux modèles. Par la suite, nous détaillons les architectures de nos modèles proposés.

Le chapitre IV présente l'entraînement et l'évaluation des performances des modèles que nous proposons. Dans un premier temps, nous mènerons une série d'expérimentations afin choisir les hyperparamètres de nos modèles. Par la suite, nous allons rapporter une synthèse des résultats obtenus suite à l'évaluation de nos modèles sur le corpus CoNLL-2003 (Sang et De Meulder, 2003) et nous discuterons la pertinence de ces résultats.

Pour finir, nous conclurons avec une synthèse du travail accompli ainsi qu'une esquisse des directions futures.

CHAPITRE I

CONCEPTS PRÉLIMINAIRES

Afin de se familiariser et faciliter la compréhension des concepts fondamentaux utilisés lors de l'élaboration de notre proposition, il sera primordial d'aborder brièvement et de mettre en avant les notions préliminaires nécessaires associées à notre problématique, et de ce fait, faciliter la lisibilité du document aux lecteurs néophytes.

Nous commençons ce chapitre par définir ce qu'est un système de reconnaissance des entités nommées (REN), ses domaines d'application et ses méthodes d'évaluation. Nous enchaînerons par une introduction aux fondements des réseaux de neurones artificiels avant de présenter quelques architectures des réseaux de neurones profonds récurrents et le modèle probabiliste CRF (Lafferty *et al.*, 2001). Nous présenterons aussi, leurs fonctionnements et comment en tirer parti pour notre objectif. Enfin, nous aborderons les différentes méthodes utilisées pour la représentation numérique des données textuelles afin qu'elles puissent être traitées pour l'apprentissage automatique.

1.1 La reconnaissance des entités nommées (REN)

La reconnaissance des entités nommées (REN) est une sous-tâche de l'extraction de l'information. Elle a pour but de distinguer un item parmi d'autres dans une séquence de données textuelles. Il s'agit d'identifier et de localiser clairement ces items dans des données non structurées ainsi que de les classifier dans des catégories d'entités sémantiques prédéfinies. Ces items, peuvent être des noms de personnes, d'organisations, de gènes, de protéines, etc.
Formellement, un système de reconnaissance des entités nommées prend en entrée une séquence de mots qui représente une phrase et renvoie en sortie la liste d'entités détectées dans la séquence.(voir figure 1.1)

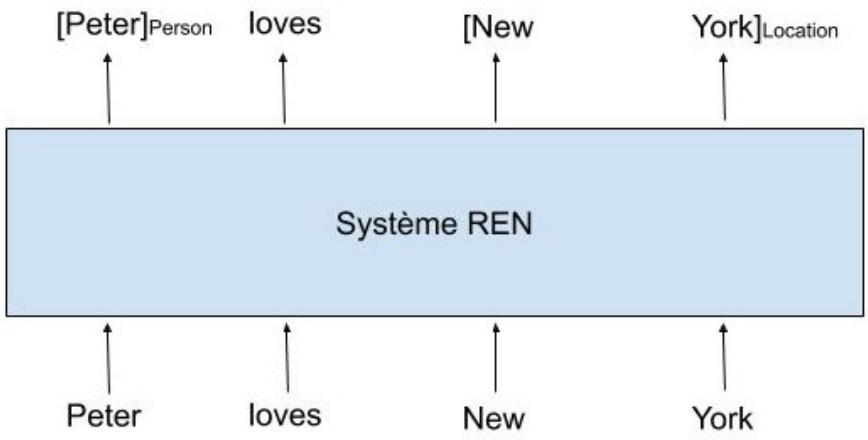


Figure 1.1: Modèle d'un système REN

Depuis sa première apparition lors de la conférence MUC-6 (Grishman, 1996), la tâche du REN avait pour but de d'extraire certaines entités nommées comme les noms de personnes, les organisations, les noms des lieux ainsi que d'autres champs comme les pourcentages, le temps et les devises. Elle a fait l'objet de plusieurs autres campagnes d'évaluation, notamment CoNLL-2003 (Sang et De Meulder, 2003), ACE (Doddington *et al.*, 2004). Par conséquent, les types d'entités nom-

mées n'ont cessé d'augmenter et d'être de plus en plus élaborés (Ling et Weld, 2012), (Lal *et al.*, 2017), (Corro *et al.*, 2015).
La reconnaissance des entités nommées est souvent employée comme une sous-tâche de prétraitement dans une variété d'applications en aval telles que la traduction automatique, la recherche d'informations, etc.

1.1.1 La REN comme tâche d'étiquetage de séquence

La tâche de la reconnaissance des entités nommées est réduite à un problème d'étiquetage de séquences. Cela est réalisé en adoptant un schéma de marquage. En effet, afin d'identifier les limites et marquer le début et la fin des segments des entités dans une phrase donnée, plusieurs schémas sont utilisés dans la littérature à l'instar du schéma BIO (Begin, Inside, Outside). Le BIO marque le début d'un segment d'une entité avec une balise B- (p. exp., Mobile B-ORG) et tous les éléments de même type qui suivent dans le segment avec une balise I- (p. exp., : World I-ORG, Congress I-ORG). Les mots qui n'appartiennent pas au segment sont marqués avec la balise O. La figure 1.2 est un exemple tiré de l'ensemble d'entraînement CoNLL-2003 (Sang et De Meulder, 2003) marquées avec ce schéma.

1.1.2 Méthodes d'évaluation

Afin d'évaluer un système de reconnaissance des entités nommées, ses résultats sont souvent comparés à une base de référence qu'on appelle «*ground truth*» qui a été annotée par des humains et considérée comme le résultat correct. Pratiquement, on peut distinguer deux catégories : une évaluation à correspondance exacte et une évaluation relaxée. Dans les deux classes d'évaluation, nous cherchons à déterminer si le segment de l'entité et son type concordent avec la base

```
I   O
can O
feel    O
the O
Mobile  B-ORG
World   I-ORG
Congress    I-ORG
vibe    O
on  O
Twitter B-ORG
See O
you O
guys    O
in  O
Barcelona   B-LOC
next    O
week    O
.   O
```

Figure 1.2: Exemple de donnée CoNLL-2003 (Sang et De Meulder, 2003)

de référence.

La campagne d'évaluation de MUC (Grishman, 1996) propose une méthode d'évaluation plus relaxée ; cette dernière repose sur un système de score où chaque entité avec le type correct est créditée, même si les limites de la fenêtre ne sont pas exactes. Les entités sont aussi créditées si le fenêtrage est bon, malgré l'inexactitude du type.
ACE (Doddington *et al.*, 2004) offre aussi une méthode relaxée, mais plutôt complexe et non intuitive due à la complexité du calcul des paramètres, ce qui rend l'analyse d'erreurs difficile.

La campagne d'évaluation de CoNLL-2003 (Sang et De Meulder, 2003) adopte la correspondance exacte où la limite de la fenêtre et le type de l'entité doivent correspondre exactement à la base de référence. Pour cela, nous identifions pour

chaque entité :

- VP_{entite} : Les vrais positifs sont les entités annotées par le système avec la vraie annotation.
- Les faux positifs FP_{entite} : sont les entités avec fausse annotation.
- Les faux négatifs FN_{entite} : sont les vraies entités que le système n'a pas détectées.

Sur cette base, les métriques suivantes sont calculées :

- la **Précision** qui représente la fraction d'entités correctement extraites :

$$Precision_{entite} = \frac{VP_{entite}}{VP_{entite} + FP_{entite}}$$

- le **Rappel** qui permet de mesurer si le système est capable de détecter toutes les entités d'un type donné dans le corpus :

$$Rappel_{entite} = \frac{VP_{entite}}{VP_{entite} + FN_{entite}}$$

La précision et le rappel globaux du système sont obtenus en calculant la moyenne des précisions, rappels par entité.
À la fin, la **mesure-F** qui est la moyenne harmonique des précisions et rappels globaux est calculée comme suit :

$$F_1 = 2 * \frac{Precision_{globale} * Rappel_{globale}}{Precision_{globale} + Rappel_{globale}}$$

Dans la littérature, la méthode d'évaluation de CoNLL-2003 (Sang et De Meulder, 2003) est souvent utilisée pour évaluer les systèmes REN grâce à sa simplicité. Dans notre évaluation, nous utiliserons cette méthode afin de comparer nos résultats aux résultats existants dans la littérature.

1.2 L'apprentissage machine

L'apprentissage machine (AM) est un domaine de recherche de l'intelligence artificielle (IA) qui se fonde sur des approches mathématiques et statistiques afin de construire un modèle qui permet aux systèmes d'apprendre à partir d'exemples appelés les données d'entraînement, et ce dans le but de résoudre une tâche (p.ex., classifier les images de chats et de chiens). Ce modèle doit être capable de généraliser sur de nouvelles données non vues durant l'entraînement.

L'apprentissage supervisé est un type d'apprentissage machine qui permet d'entraîner un modèle à partir d'un jeu de données préalablement étiqueté par un expert. Ce jeu de données est généralement divisé en trois parties de tailles différentes.

— Ensemble d'entraînement : représente la plus grande partie des données. Cet ensemble sert à régler les paramètres du modèle par optimisation.

— Ensemble de validation : il permet de choisir les hyper-paramètres qui ne peuvent pas être réglés par optimisation tels que le taux d'apprentissage (Goodfellow *et al.*, 2016).

— Ensemble de test : il permet de mesurer les performances du système.

1.3 Réseau de neurones artificiels

Les réseaux de neurones artificiels (RNA) appartiennent aux modèles d'apprentissage machine. Il s'agit d'un modèle mathématique qui permet de modéliser une fonction à l'aide d'un ensemble de paramètres qui sont ajustés par entraînement. L'unité de base d'un RNA est le neurone (figure 1.3). Ce dernier est caractérisé

par ses poids et sa fonction d'activation non linéaire [1] selon l'équation suivante :

$$y = f(\sum w_i x_i + b)$$

Où :

- les x_i sont des scalaires représentant les entrées.
- les poids w_i et le biais b sont les paramètres réglés par entraînement.
- f est la fonction d'activation.
- y est la sortie du neurone

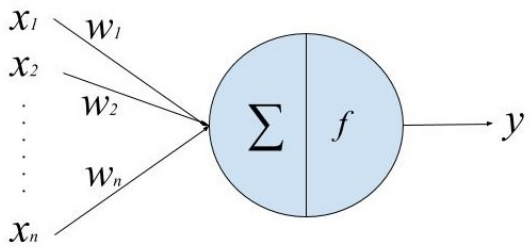


Figure 1.3: Modèle d'un neurone artificiel

Le rôle de la fonction d'activation vise à donner un comportement non linéaire au neurone (Goodfellow *et al.*, 2016). Plusieurs fonctions d'activation ont été utilisées comme la sigmoïde (eq., 1.1a), la tangente hyperbolique (eq., 1.1b) et l'unité de rectification linéaire ReLU (eq., 1.1c)

1. https ://fr.wikipedia.org/wiki/Fonction_d%27activation

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1.1a}$$

$$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{1.1b}$$

$$relu(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{1.1c}$$

Les connexions dans un réseau de neurones multicouches (perceptron multicouche) sont organisées en forme de couches interconnectées (les neurones au sein de la même couche ne sont pas reliés). Ainsi, les données se propagent dans le réseau dans un seul sens vers l'avant (propagation vers l'avant) depuis la couche d'entrée, en passant par les couches cachées jusqu'à la couche de sortie tel que cela est illustré dans figure 1.4.

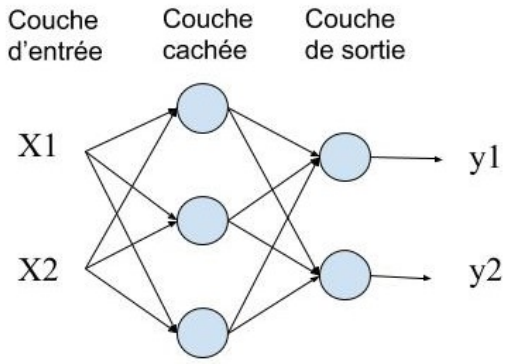


Figure 1.4: Exemple d'un modèle de réseau de neurones multicouches

L'ajustement des paramètres d'un réseau de neurones artificiels se fait par optimisation de la fonction objectif du RNA sur le jeu d'entraînement. La fonction objectif d'un réseau de neurones artificiels est définie de sorte à aligner son optimisation avec l'apprentissage de l'ensemble d'entraînement. La fonction objectif est souvent appelée la fonction d'erreur ou la fonction de perte.

Dans un problème de régression une fonction de perte telle que la norme L^2 peut être utilisée et elle est définie comme suit :

$$L(y, \hat{y}) = (y - \hat{y})^2 \tag{1.2}$$

où y (resp. $\hat{y}$) est la valeur vraie valeur (resp. valeur prédite).

D'autre part, dans un problème de classification binaire (deux classes), la fonction d'entropie croisée est souvent utilisée comme fonction de perte. Cette fonction est définie par l'équation suivante :

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) log(1 - \hat{y}) \tag{1.3}$$

où $y \in \{0, 1\}$ représente la classe de l'entrée et $\hat{y}$ est la sortie du réseau. La décision sera la classe 1 si $\hat{y} \geq \frac{1}{2}$, sinon c'est la classe 0.
L'apprentissage consiste à minimiser la moyenne de la fonction de perte sur l'ensemble d'entraînement.

1.3.1 Réseau de neurones récurrents (RNR)

Les réseaux de neurones récurrents répondent à la limitation des réseaux de neurones de type perceptron multicouche. Ces derniers n'acceptent que des entrées de taille fixe. En effet, l'entrée d'un perceptron multicouches est un vecteur de taille fixe, et ceci pour tous les éléments d'un jeu de données. En revanche, dans un problème de traitement de séquences tel que la reconnaissance des entités nommées, les phrases en entrée sont de longueurs variables.

Un RNR (Figure 1.5) est un réseau de neurones spécialisé dans le traitement de séquences de tailles variables grâce au principe du partage des paramètres entre les différents pas de traitement de la séquence en entrée. Étant donné une séquence en entrée $\mathbf{x}^{<1:T>}$, le RNR produit une sortie $\mathbf{h}^{<1:T>}$ telle que définie à l'équation

1.4. La fonction f représente la sortie du module A paramétré avec θ.

$$\boldsymbol{h}^{<t>} = f(\boldsymbol{x}^{<t>}, \boldsymbol{h}^{<t-1>}, \theta) \tag{1.4}$$

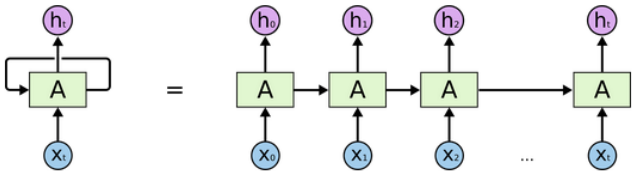


Figure 1.5: Réseau de neurones récurrents RNR (Olah Christopher, 2015)

Théoriquement, les réseaux de neurones récurrents visent à se rappeler de la dépendance à long terme au besoin. En pratique, les RNR ne paraissent pas pouvoir apprendre une telle longue dépendance. Cette problématique a été abordée par Hochreiter (1991) et Bengio *et al.* (1994) qui ont étudié les raisons fondamentales des faiblesses de RNR. En effet, ils souffrent du problème de la dissipation du gradient suite à la multiplication des valeurs décimales du gradient d'erreur à chaque pas du modèle. La dissipation du gradient rend l'entraînement du modèle difficile lorsque la séquence est très longue.

De tous les modèles des RNN, les GRU «*Gated Recurrent Unit*» (Cho *et al.*, 2014) et les LSTM «*Long short-term memory*» (Hochreiter et Schmidhuber, 1997) se distinguent par leur capacité à maintenir un flux de gradient permettant l'entraînement des modèles.

1.3.2 Réseau de neurones récurrents avec mémoire à court long terme (LSTM)

Introduit par Hochreiter et Schmidhuber (1997), les LSTM sont des RNR, qui permettent de résoudre le problème de dépendance à long terme. Ils ont été largement utilisés dans la littérature.

L'idée principale derrière les LSTM se résume dans la cellule mémoire «*The memory cell*» ($\boldsymbol{c}_t$) (Figure 1.6), qui est la clé de leurs succès. Elle permet de retenir l'information pertinente pour la tâche à travers les pas (le temps) lors du traitement des éléments de la séquence.

La cellule mémoire dans un LSTM est maintenu grâce au principe de portes «*gating*». Un module LSTM est composé de trois portes : $\boldsymbol{f}_t$ est la porte d'oubli, $\boldsymbol{i}_t$ est la porte d'entrée alors que $\boldsymbol{o}_t$ est celle de sortie (Figure 1.6).

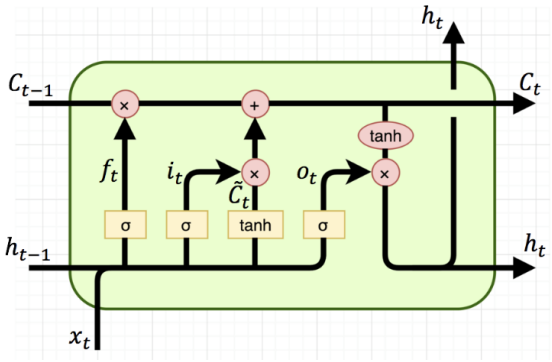


Figure 1.6: Une cellule LSTM (Olah Christopher, 2015)

Formellement, un module LSTM est définie comme suit :

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i)$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c)$$

$$\boldsymbol{c}_t = (\boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t)$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)$$

Où :

— $\boldsymbol{W}_f, \boldsymbol{U}_f, \boldsymbol{W}_i, \boldsymbol{U}_i, \boldsymbol{W}_o, \boldsymbol{U}_o, \boldsymbol{W}_c, \boldsymbol{U}_c$ sont les matrices de poids à régler durant

l'entraînement.

— $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_o$, $\boldsymbol{b}_c$ sont les vecteurs de biais à régler durant l'entraînement.

— $\boldsymbol{x}_t$ est le vecteur en entrée de l'unité LSTM au pas t.

— σ et tanh sont la fonction sigmoïde et la tangente hyperbolique respectivement.

— $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ et $\boldsymbol{o}_t$ vecteurs représentent les portes d'oubli, d'entrée et de sortie respectivement.

— $\boldsymbol{h}_t$ est le vecteur de sortie (état caché) issu du module LSTM au pas t.

— L'opération $\odot$ est le produit vectoriel de Hadamard [2].

1.3.3 Réseau neuronal convolutif (CNN)

Célèbre dans le domaine de l'imagerie et la vision artificielle, les CNN (LeCun *et al.*, 1989) ou réseaux de neurones à convolution sont une classe de l'apprentissage profond. Leur invention a engendré beaucoup de progrès dans plusieurs applications telles que la classification d'images, la reconnaissance faciale, etc. Récemment, l'utilisation des CNN s'est étendue au domaine du traitement du langage naturel (TALN) ; par conséquent, des résultats intéressants ont été observés concernant ce domaine (Ma et Hovy, 2016).

L'application des CNN à une séquence de texte est similaire à son application sur une image. Les éléments d'une séquence sont représentés sous la forme d'une matrice où chaque ligne représente le plongement d'un mot comme Glove (Pennington *et al.*, 2014) ou Word2vec (Mikolov *et al.*, 2013b).

La figure 1.7 est un exemple d'application des CNN avec des convolutions à une

2. https ://fr.wikipedia.org/wiki/Produit_matriciel_de_Hadamard#cite_note-1

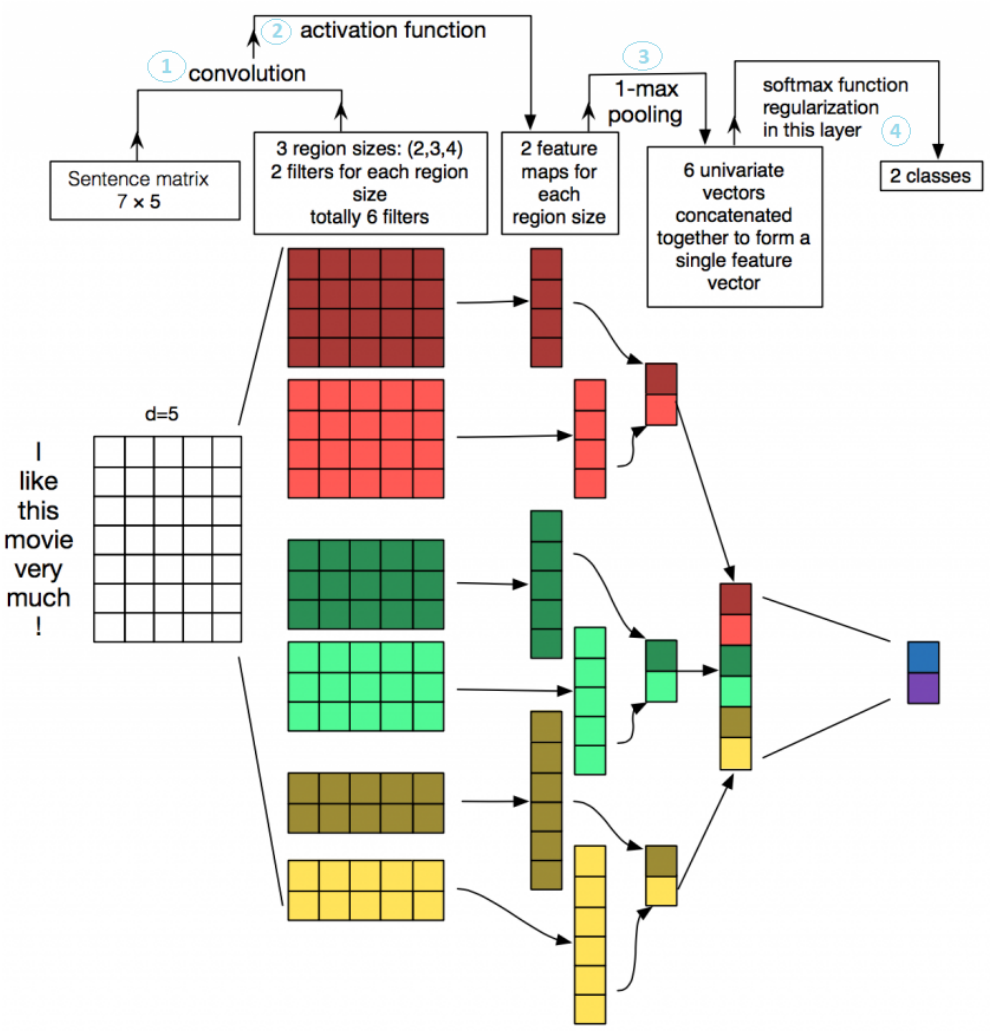


Figure 1.7: CNN pour NLP (wildml, 2015)

dimension [3] (1D) sur une matrice de mots $\boldsymbol{X}$ de taille $T \times d$ où d est la dimension des vecteurs de mots et T est le nombre de mots dans la séquence. Une couche de convolution comporte plusieurs filtres qui peuvent être de différentes hauteurs f "*region size*" et de longueurs égales à d, est appliquée à la matrice pour obtenir des vecteurs $\boldsymbol{a}^{(k)}$ qu'on appelle les «*feature maps*» où k est le numéro du filtre.

3. Une convolution où le filtre se déplace dans une seule direction

L'application d'un filre $\boldsymbol{K} \in \mathbf{R}^{f \times d}$ sur une entrée $\boldsymbol{X} \in \mathbf{R}^{T \times d}$ donne lieu à une carte de caractéristiques (*feature map*) $\boldsymbol{a}$ définie par :

$$a_i = g(\sum_{n=1}^{f} \sum_{j=1}^{d} \boldsymbol{X}[i+n-1][j] \times \boldsymbol{K}[n][j]) \tag{1.5}$$

Où :

— $\boldsymbol{a} \in \mathbf{R}^{T-f+1}$.

— g est une fonction d'activation (p. ex., Relu).

Nous appliquons sur chaque «*feature map*» obtenu précédemment un sous-échantillonnage maximal global tel que défini dans l'équation suivante :

$$\max_{1 \leq i \leq T-f+1} (\boldsymbol{a}[i]) \tag{1.6}$$

Les valeurs obtenues seront concaténées pour former une représentation finale qui va alimenter une couche de sortie ayant un softmax (Figure 1.7).

1.4 Champs aléatoires conditionnels (CRF)

L'une des techniques utilisées dans l'étiquetage de séquences est les CRF ou champs aléatoires conditionnels. Proposé par Lafferty *et al.* (2001), leur utilisation ne cesse de croître. Ceci est dû à leur efficacité dans l'étiquetage des données séquentielles (langage naturel, séquences biologiques, vision par ordinateur) lorsqu'ils sont utilisés comme un classificateur au dessus d'une couche de LSTM, cela a été observé dans la majorité des travaux (Lample *et al.*, 2016), (Ma et Hovy, 2016), etc.

Les CRF représentent un modèle graphique probabiliste non orienté, qui appartient à la famille des modèles discriminatifs. Ces modèles ont pour but de modéliser la distribution de la probabilité conditionnelle $P(\mathbf{y}^{<1:T>}|\mathbf{x}^{<1:T>})$ en essayant d'apprendre la dépendance d'une séquence de labels à estimer $\mathbf{y}^{<1:T>}$ étiquetant

une séquence de variables observées $\mathbf{x}^{<1:T>}$; autrement dit, trouver la probabilité d'une séquence de prédictions $\mathbf{y}^{<1:T>} = (\mathrm{y}^{<1>}, \mathrm{y}^{<2>}, \ldots, \mathrm{y}^{<T>})$ étant donnée une séquence $\mathbf{x}^{<1:T>} = (\mathrm{x}^{<1>}, \mathrm{x}^{<2>}, \ldots, \mathrm{x}^{<T>})$ donnée en entrée, où T est la taille de séquence. De ce fait, les CRF s'avèrent très performants dans la classification de séquences par rapport aux approches génératives telles que les HMM (Modèle de Markov caché (Baum et Petrie, 1966)) qui essayent de représenter la probabilité jointe $P(\mathbf{x}^{<1:T>}, \mathbf{y}^{<1:T>})$. La formulation du cadre théorique des CRF sera présentée au chapitre II.

1.5 Représentation de mots

La représentation de mots est une méthode souvent utilisée dans le domaine du TALN. Elle permet de représenter les mots d'un vocabulaire par un vecteur. Cette représentation peut être soit de type canonique, soit une représentation distributionnelle obtenue par l'apprentissage automatique à partir d'un corpus de texte. Cette dernière permet que les mots apparaissant dans des contextes similaires soient représentés par des vecteurs denses et relativement peu distants dans l'espace vectoriel.

1.5.1 Représentation canonique (*One hot encoding*)

La représentation canonique consiste à représenter chaque mot du vocabulaire d'un corpus par un vecteur composé de zéros (0), sauf la position réservée au mot qui est mise à un (1). Par exemple, pour le vocabulaire [Peter, took, this, plane, from, Paris, to, Montreal, this, morning] de taille égale à 9, la représentation canonique du mot «Paris» est [0,0,0,0,1,0,0,0,0].
La sémantique des mots n'est pas prise en considération dans l'encodage cano-

nique, où chaque mot est indépendant de l'autre. La similarité cosinus [4] entre les mots «Paris = [0,0,0,0,1,0,0,0,0]» et «Montréal = [0,0,0,0,0,0,1,0,0]» est égale à zéro, alors qu'il est évident que les deux mots sont dépendants sémantiquement où les deux sont des villes.

Dans le but de remédier à ces problèmes, plusieurs techniques d'encodage ont été proposées par la suite, à l'instar de «Word2vec» (Mikolov *et al.*, 2013b), «Glove» (Pennington *et al.*, 2014), «FastText» (Bojanowski *et al.*, 2017). Ces derniers prennent en considération la cooccurrence des mots dans les phrases.

1.5.2 Word2Vec

word2vec (Mikolov *et al.*, 2013b) est un réseau neuronal à deux couches qui est entraîné d'une manière supervisée dans le but de produire une représentation distributionnelle des mots. Il prend comme entrée un grand corpus [5] de mots et produit un espace vectoriel de faible dimension (entre 50 et 300). Les vecteurs de mots sont positionnés dans l'espace vectoriel de telle sorte que les mots qui partagent des contextes communs dans le corpus sont situés les uns à côté des autres dans l'espace vectoriel.

Il existe deux types d'architectures utilisées par «*word2vec*» :

— le modèle de **sacs de mots continus (CBOW)** vise à apprendre un plongement en entrainant un réseau qui prédit le mot à partir de son contexte. Le contexte peut être formé par exemple de cinq mots à droite et de cinq mots à gauche du mot à prédire.

— le modèle ***skip-gram*** a une architecture symétrique visant à apprendre un

4. La similarité cosinus entre deux vecteurs est définie par le cosinus de l'angle qu'ils forment.

5. Le corpus «*Google News*» contient environ 6 milliards de mots

plongement en entrainant un réseau qui prédit les mots du contexte étant donné un mot en entrée.

Selon Mikolov *et al.* (2013a), l'architecture «*CBOW*» est plus rapide à entraîner, mais le modèle «*skip-gram*» donne généralement de meilleurs résultats.

1.5.3 GloVe (Global Vectors for Word Representation)

Étant donné que les méthodes word2vec ne prennent pas en considération les statistiques de la co-occurrence des mots, «Glove» (Pennington *et al.*, 2014) a fait son apparition peu après et propose d'utiliser la matrice de co-occurrence des mots, où chaque élément de la matrice représente le nombre de fois, dans le corpus, où le mot d'une colonne apparaît à proximité du mot d'une ligne de la matrice. Glove cherche à apprendre des vecteurs représentatifs de mots qui reflètent leurs cooccurrences, en utilisant les statistiques globales de la matrice.

1.5.4 FastText : plongement des n-grammes

Proposée par Bojanowski *et al.* (2017), «FastText» est une technique de plongement de mots, dérivée du modèle word2vec, sauf que dans cette méthode, au lieu d'apprendre directement les représentations de mots, on utilise plutôt les n-grammes. Par exemple pour le mot «Canada» et n=3, alors on obtient <Ca, Can, ana, nad, ada, da>, où "<" et ">" représentent le début et la fin du mot. Un modèle word2vec est entraîné dans le but d'obtenir un plongement des n-grammes. Par la suite, chaque mot sera représenté par la somme des plongements de ses n-grammes.

Les trois modèles de plongement de mots présentés dans cette section sont les modèles les plus utilisés dans la littérature. Chacun d'eux possède ses propres avantages. «Glove» nous permet de tirer avantage des statistiques des nombres

d'occurrences des mots, alors que «FastText» permet d'améliorer «Word2vec» en utilisant le plongement des sous-mots. De facto, les mots rares sont mieux représentés (Bojanowski *et al.*, 2017). En pratique le choix du plongement à utiliser fait partie des hyper-paramètres du modèle.

1.6 Conclusion

La reconnaissance des entités nommées est une tâche essentielle pour plusieurs autres applications en aval. Un tel système doit être en mesure d'identifier correctement les étiquettes des entités nommées. Plusieurs algorithmes d'apprentissage automatique, y compris l'apprentissage profond, peuvent être appliqués pour surmonter ce problème d'étiquetage de séquence.
Dans le chapitre suivant, nous présentons l'application des méthodes d'apprentissage automatique sur la reconnaissance des entités nommées. Un état de l'art de l'utilisation des méthodes d'apprentissage profond sera également détaillé.

CHAPITRE II

ÉTAT DE L'ART

La tâche de détection des entités nommées similairement aux autres tâches TALN suscite un intérêt croissant pour beaucoup de chercheurs dans ce domaine, dû à son importance cruciale. En effet, durant les deux dernières décennies, plusieurs recherches ont été réalisées afin d'améliorer les résultats de cette tâche et faire face à ses enjeux.

Le progrès des travaux sur l'extraction d'entités nommées durant les deux dernières décennies peut être divisé en trois phases essentielles. Dans un premier temps, nous notons l'utilisation des approches d'apprentissage machine classiques, notamment les modèles probabilistes. La deuxième période est marquée par l'utilisation des réseaux de neurones profonds. Leur grand succès a poussé les chercheurs dans ce domaine à s'orienter dans cette direction. Enfin, la communauté des chercheurs s'est récemment intéressée à l'apprentissage par le transfert de connaissances, ce qui a provoqué un changement de paradigme remarquable en atteignant l'état de l'art dans plusieurs tâches de traitement automatique du langage naturel.

Dans ce chapitre nous allons essayer de se pencher sur les approches basées sur l'apprentissage profond (AP) ainsi que présenter les techniques utilisées dans la littérature. Les résultats distingués obtenus dans les travaux d'extraction d'entités nommées seront dûment synthétisés à la fin de ce chapitre.

2.1 REN à l'âge du pré-apprentissage profond

Avant l'arrivée de l'apprentissage profond, l'utilisation des algorithmes d'apprentissage automatique classiques était la pierre angulaire du développement des systèmes REN à l'instar des modèles à base de règles (Sekine et Nobata, 2004), CRF (Lafferty *et al.*, 2001), etc. Étant donné que dans le cadre de notre travail, nous nous sommes limités qu'aux systèmes de reconnaissance des entités nommées basés sur l'apprentissage profond, nous allons donc présenter brièvement certaines de ces approches dans cette section.

Certains travaux tels que (Nadeau et Sekine, 2007) et (Yadav et Bethard, 2018) ont essayé de recueillir et de classifier les approches basées sur l'apprentissage automatique qui traitent la tâche de la reconnaissance des entités nommées. Nous pouvons distinguer trois courants principaux : les approches à base de règles, l'apprentissage supervisé et l'apprentissage non supervisé.

Les approches à base de règles sont conçues manuellement. Les règles se basent sur des lexiques et des connaissances spécifiques au domaine (Sekine et Nobata, 2004). Par conséquent, ces systèmes dépendent fortement de l'expertise dans le domaine et rendent impossible leur portabilité entre domaines.

Les approches d'apprentissage non supervisées telles que le partitionnement de données (*clustering*) ont été utilisées dans la littérature pour la classification des entités nommées. En effet, les entités du même type ont tendance à apparaître dans le même groupe (Liu et Zhou, 2013). D'autres approches ont été proposées, lesquelles reposent sur les ressources lexicales à l'instar de Alfonseca et Manandhar (2002) qui utilisent le «*WordNet*» (Miller, 1995) pour classifier les entités sous le «*SynSet*» [1] le plus similaire.

---

1. Liste de mots qui apparaît souvent simultanément dans un corpus

Par ailleurs, les approches d'apprentissage supervisées ont été largement utilisées dans ce domaine. Ce type d'approche repose sur l'ingénierie de caractéristiques discriminantes qui permet d'identifier les entités nommées à l'instar de la morphologie de mots (préfixe, suffixe et la capitalisation), les dictionnaires, les ontologies, les statistiques sur le corpus (Salton et McGill, 1986), etc. Toutefois, l'ingénierie des caractéristiques est une tâche coûteuse, chronophage et nécessite de l'expertise dans le domaine.

L'une des approches notables utilisée dans cette catégorie est le CRF (Lafferty *et al.*, 2001) qui a permis de réaliser de bons résultats (McCallum et Li, 2003) grâce à sa capacité de prendre en considération le voisinage lors de l'étiquetage d'un mot. Toutefois, cette approche nécessite une ingénierie manuelle des caractéristiques.

2.2 REN à l'ère de l'apprentissage profond

Comme nous l'avons déjà vu dans le chapitre précédent, l'apprentissage profond est un sous-domaine de l'apprentissage machine. Un réseau de neurones profond est constitué de plusieurs couches de neurones permettant d'apprendre des caractéristiques «*features*» discriminantes depuis les données avec différents niveaux d'abstraction (LeCun *et al.*, 2015).

La clé du succès de l'apprentissage profond est sa capacité d'exploration et d'extraction automatique des caractéristiques utiles et latentes. Cela est obtenu depuis les vecteurs représentatifs des données brutes d'une manière directe et selon un processus de bout en bout. De ce fait, l'apprentissage profond nous épargne le processus de l'ingénierie manuelle des caractéristiques «*features engineering*» [2]

2. Par exemple, définir manuellement que les mots commençant par une majuscule est une caractéristique qui permet de prédire si ce mot est une entité nommée ou pas.

qui nécessite du temps et de l'expertise dans le domaine. De plus, l'apprentissage profond permet d'apprendre des motifs complexes depuis les représentations de données. Ces avantages ont permis à cette technique d'être un choix compétitif.

Les modèles de reconnaissance des entités nommées basés sur l'apprentissage profond se constituent en général de trois couches : la couche de représentation de données, la couche d'encodage de contextes et la couche de décodage des entités nommées par étiquetage des mots (figure 2.1). Dans ce qui suit, nous allons présenter les différentes techniques utilisées dans chaque couche.

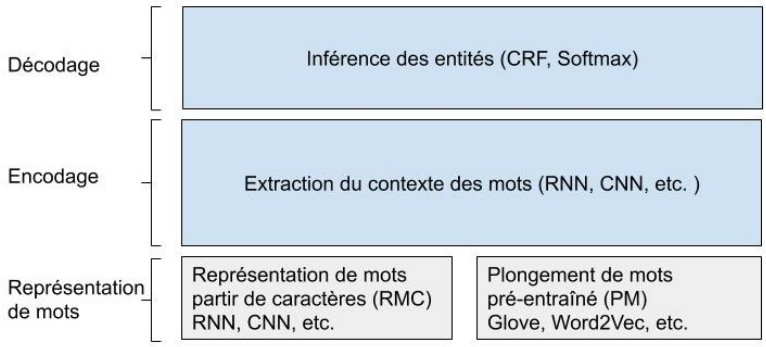


Figure 2.1: Architecture d'un système REN basé sur l'AP

2.2.1 La couche de la représentation distributionnelles de mots

Afin de permettre aux modèles de traiter les données en entrée, ces dernières devront être en format compréhensible par la machine ; pour cela, il existe plusieurs techniques de représentation dans la littérature. Nous pouvons distinguer deux types de représentations de mots utilisés : les représentations par plongement de mots pré-entraîné (PM) comme «GloVe», «Word2vec», etc. et les représentations de mots à partir de leurs caractères (RMC)) en utilisant des architectures neuronales (p. ex., CNN, LSTM) directement sur des caractères.
Les représentations de mots permettent d'encapsuler des caractéristiques latentes

des données brutes ainsi que de capturer automatiquement certaines propriétés syntaxiques et sémantiques des mots.

Plongement de mots pré-entraîné.

L'avantage clé du plongement de mots pré-entraîné est sa représentation du mot par des vecteurs denses de faibles dimensions. Ils sont le résultat d'entraînement de modèles sur des corpus de textes non annotés. Nous pouvons lister certains plongements de mots utilisés dans la littérature à l'instar de SENNA [3] (Collobert *et al.*, 2011b), Google Word2vec [4] (Mikolov *et al.*, 2013b), Stanford Glove [5] (Pennington *et al.*, 2014) et Facebook fastText [6] (Bojanowski *et al.*, 2017), etc..

Plusieurs travaux comme ceux de Collobert *et al.* (2011b), Turian *et al.* (2010) et Huang *et al.* (2015) ont utilisé les plongements de mots pré-entraîné dans leurs modèles pour enrichir les données en entrée. D'autre part, (Lample *et al.*, 2016), (Ma et Hovy, 2016)et (Peters *et al.*, 2017) ont tiré parti des avantages de cette méthode en éliminant les données externes telles que les dictionnaires. De plus, Lample *et al.* (2016) ont mis en évidence l'importance des plongements de mots pré-entraînés et leur rôle dans l'amélioration des résultats. Lample *et al.* (2016) ont testé leur modèle avec et sans un plongement de mots de type «*skip-n-gram*» (Mikolov *et al.*, 2013b) qui est une variante de Word2vec dont la dimension des vecteurs de plongement est de 100. Ils ont observé une dégradation considérable du résultat lorsque le plongement de mots pré-entraîné n'est pas utilisé.

---

3. https ://ronan.collobert.com/senna/

4. https ://code.google.com/archive/p/word2vec/

5. http ://nlp.stanford.edu/projects/glove/

6. https ://fasttext.cc/docs/en/english-vectors.html

Ma et Hovy (2016) ont expérimenté leur modèle avec trois plongements de mots pré-entraîné de différentes dimensions : Glove (100), SENNA (50) et Word2vec (300). Le résultat avec Glove était légèrement meilleur que celui de SENNA, tandis que le Word2vec et l'initialisation aléatoire étaient médiocres.

De même, Yang *et al.* (2018) ont réimplémenté les modèles de Lample *et al.* (2016) et Ma et Hovy (2016). Ils ont rapporté les mêmes résultats : Glove (100) était meilleur que SENNA (50).

Reimers et Gurevych (2017) ont étudié une variété de plongement de mots pré-entraîné (Word2vec, Glove 100, Glove 200, Glove 300, FastText, Levy BoW (Levy et Goldberg, 2014), Komn (Komninos et Manandhar, 2016)). Glove 300 s'est montré très performant dans la tâche de la reconnaissance des entités nommées. De plus, ils ont affirmé que le choix du plongement de mots pré-entraîné a un impact très important sur les résultats du modèle.

Peters *et al.* (2017) ont utilisé le plongement de mots pré-entraîné SENNA (50) dans leur modèle TagLM.

Représentation du mot à partir de ses caractères.

La deuxième méthode souvent utilisée dans la littérature est la représentation du mot à partir de ses caractères (RMC) (Ling *et al.*, 2015).
En effet, dans la tâche de la reconnaissance des entités nommées, les caractéristiques orthographiques (majuscule, préfixe, suffixe, etc.) des mots sont très importantes et peuvent nous indiquer à quoi ressemble le mot à étiqueter. Par exemple, si le mot commence par une majuscule, il est probable qu'il s'agisse d'une entité nommée. De ce fait, cette méthode nous permet d'explorer la morphologie des mots et de capturer les sensibilités orthographiques. En outre, la représentation du mot basée sur ses caractères est très utile pour les mots hors du vocabulaire.

En effet, certains mots rares qui n'apparaissent pas dans le corpus d'entraînement du plongement de mots ne possèdent pas un vecteur représentatif. Ces mots sont souvent représentés par un vecteur générique. De ce fait, la représentation des mots à partir de caractères devrait permettre de mieux représenter les mots hors vocabulaire.

Afin d'obtenir une représentation de mots à partir de caractères (RMC), on distingue deux types d'architectures utilisés dans la littérature : les réseaux à convolution (RMC-CNN) et les réseaux récurrents (RMC-RNR).

Ma et Hovy (2016) ont opté pour l'architecture à convolution pour l'extraction des représentation de mots à partir de caractères via une couche de convolution suivie d'une couche de sous-échantillonnage, et ce afin d'obtenir un vecteur représentatif pour chaque mot. Ensuite, ils ont combiné ces représentations avec un plongement de mots pré-entraîné pour former une représentation finale. Ce modèle sera présenté plus en détail dans le chapitre suivant.
De même, Chiu et Nichols (2015) ont utilisé une représentation de mots à partir de caractères basée sur les CNN en combinaison avec un plongement de mots pré-entraîné ainsi que des connaissances externes conçues manuellement (dictionnaires).
Peters *et al.* (2017) et Peters *et al.* (2018) ont utilisé, dans leurs deux modèles TagLM et ELMo, un réseau de neurones à convolution pour la représentation de mots à partir de leurs caractères.

La représentation de mots à partir de caractères basés sur les réseaux de neuronnes récurrents utilise les LSTM ou les GRU. Lample *et al.* (2016) ont employé un réseau LSTM bidirectionnel (Bi-LSTM) pour obtenir une représentation du mot à partir de ses caractères. Ils ont observé que les réseaux récurrents sont biaisés en faveur des dernières entrées traitées. Par conséquent, lorsqu'on effectue une

passe de la gauche vers la droite sur le mot, le vecteur représentatif résultant va contenir plus d'information sur le suffixe du mot en question. De ce fait, il serait intéressant de faire une deuxième passe sur le mot par un autre réseau LSTM de la droite vers la gauche, obtenant par cela une meilleure représentation du préfixe. Les deux représentations seront concaténées pour former la représentation finale du mot. Ce modèle sera présenté plus en détail dans le prochain chapitre.

Yang *et al.* (2016) ont utilisé un GRU bidirectionnel (Bi-GRU) au lieu d'un Bi-LSTM pour apprendre des informations morphologiques à partir de la séquence de caractères d'un mot. Le vecteur issu de cette étape est concaténé à un plongement de mots pré-entraîné. De plus, leur modèle est multilingue et peut être employé dans n'importe quelle tâche d'étiquetage de séquences.

Jansson et Liu (2017) ont concaténé la représentation de mots à partir de caractères basés sur LSTM, le plongement de mots pré-entraîné et l'étiquetage morpho-syntaxique POS pour former la représentation des mots en entrée du modèle. De plus, ils ont utilisé le modèle LDA (Allocation de Dirichlet Latente) de Hoffman *et al.* (2010) afin d'assigner un thème pour chaque mot. Le modèle permet d'identifier les entités nommées dans un texte bruyant tel que les tweets.

Combinaison des deux représentations.

L'approche la plus classique pour former une représentation finale pour le mot est de concaténer tout simplement le plongement de mots pré-entraîné et la représentation de mots à partir de caractères. Rei *et al.* (2016) ont essayé de combiner les deux représentations avec un mécanisme d'attention pour pousser le modèle à choisir la meilleure représentation entre les deux. Étant donné une représentation de mot à partir de ses caractères $\boldsymbol{x}$ et un plongement de mot pré-entraîné $\boldsymbol{m}$, la représentation finale $\tilde{\boldsymbol{x}}$ est le résultat de l'addition pondérée des deux représentations en utilisant un réseau à deux couches pour calculer les poids, comme illustré dans les deux formules suivantes :

$$\boldsymbol{z} = \sigma(\boldsymbol{W}_z^{(3)} \tanh(\boldsymbol{W}_z^{(2)}\boldsymbol{x} + \boldsymbol{W}_z^{(1)}\boldsymbol{m})) \qquad \tilde{\boldsymbol{x}} = \boldsymbol{z}.\boldsymbol{x} + (1 - \boldsymbol{z})\boldsymbol{m}$$

où :

— σ et tanh sont : la fonction sigmoïde et la tangente hyperbolique respectivement.

— $\boldsymbol{W}_z^{(1)}, \boldsymbol{W}_z^{(2)}, \boldsymbol{W}_z^{(3)}$ sont des matrices composées des paramètres à apprendre durant l'entraînement.

— Le vecteur $\boldsymbol{z}$ a la même dimension que $\boldsymbol{x}$ et $\boldsymbol{m}$, agissant comme le poids entre les deux vecteurs.

2.2.2 La couche d'encodage du contexte

Après avoir représenté chaque mot dans la séquence par une représentation vectorielle finale, l'encodage du contexte vient comme deuxième étape. En effet, un système de reconnaissance des entités nommées basé sur l'apprentissage profond essaie d'extraire le contexte autour de chaque mot en considérant la séquence entière. Ainsi, le contexte [7] dans lequel un mot apparaît pourra nous indiquer le type de l'étiquette correspondante. Par exemple, si le mot au pas courant est "Langdon" et en sachant que le mot précédent est "Mr.", il est très probable qu'il s'agisse d'une entité nommée de type personne.

De même que la couche précédente (la représentation distributionnelle de mots), des architectures telles que CNN et RNR sont largement utilisées dans la littérature. Pour l'encodage du contexte, Collobert *et al.* (2011b) étaient les premiers à employer un réseau CNN pour l'encodage des contextes des mots. Leur approche consiste à appliquer une couche de convolution sur toute la phrase (représentée

7. Représenté par phrase

par un ensemble de plongement de mots) afin d'extraire des caractéristiques de niveau supérieur. Une couche de sous-échantillonnage maximum est appliquée après la couche de convolution pour pousser le modèle à capturer les caractéristiques les plus importantes. De même, Yao *et al.* (2015) ont adapté cette méthode pour extraire des entités nommées dans le domaine biomédical.

D'autre part, les LSTM et les GRU sont un choix typique pour l'encodage du contexte, dû à leurs bonnes performances et à leur capacité à conserver le contexte à long terme grâce aux cellules-mémoires. Ces réseaux récurrents se sont montrés très efficaces surtout lorsqu'ils sont déployés d'une manière bidirectionnelle afin d'obtenir le contexte à gauche et à droite du mot. De ce fait, l'encodage du contexte de chaque mot aura une vision globale de la phrase.

Huang *et al.* (2015) étaient les premiers à s'intéresser à l'encodage du contexte par Bi-LSTM lorsqu'ils ont mis en place un modèle pour l'étiquetage de séquences pour les tâches de reconnaissance des entités nommées, de l'étiquetage morpho-syntaxique et de l'analyse syntaxique de surface en utilisant un réseau Bi-LSTM. Depuis, l'utilisation de Bi-LSTM pour l'encodage du contexte est en vogue (Lample *et al.*, 2016), (Ma et Hovy, 2016), (Chiu et Nichols, 2015), etc.
Yang *et al.* (2016) ont préféré utiliser un Bi-GRU pour l'encodage de contexte et leur résultat était compétitif. De même, (Peters *et al.*, 2017), dans leur modèle TagLM, ont utilisé un premier GRU bidirectionnel pour l'encodage du contexte à partir d'un plongement de mots pré-entrainé concaténés à une représentation de mots à partir de caractères. L'encodage est combiné par la suite avec un modèle de langue pour alimenter un deuxième GRU bidirectionnel.

2.2.3 La couche de décodage des entités

Avant de procéder au décodage des entités, il va falloir projeter les vecteurs de contextes des mots obtenus lors de la phase précédente en utilisant une couche de neurones entièrement connectés. Le but est d'obtenir un vecteur de scores $\boldsymbol{s} \in \mathbb{R}^n$ où n est le nombre d'étiquettes distinctes dans le corpus. Étant donné un mot $\mathrm{x}^{<t>}$ au pas t de la phrase en traitement, $\boldsymbol{s}^{<t>}$ représente son vecteur de scores. Chaque $s_i^{<t>}$ est le score estimant la confiance accordée à étiqueter $\mathrm{x}^{<t>}$ avec l'étiquette i. Dans la tâche de reconnaissance des entités nommées, le système est entraîné dans le but de prédire la probabilité conditionnelle $P(\mathrm{y}^{<1:T>}|\mathrm{x}^{<1:T>})$ où $\mathrm{y}^{<1:T>}$ représente une séquence d'étiquettes de la phrase $\mathrm{x}^{<1:T>}$, T étant la taille de la phrase.

Dans la littérature, on distingue deux modèles pour estimer $P(\mathrm{y}^{<1:T>}|\mathrm{x}^{<1:T>})$; le premier est basé sur la fonction exponentielle normalisée (softmax) et le deuxième utilise un CRF linéaire.

Décodage avec la fonction exponentielle normalisée (softmax).

Ce type de décodage consiste à prédire l'étiquetage de la phrase en utilisant, dans la dernière couche du système de reconnaissance des entités nommées, une couche de softmax qui produit à la sortie une séquence de prédiction $\hat{\mathbf{y}}^{<1:T>}$. Chaque élément $\hat{\mathbf{y}}^{<t>}$ de cette séquence est une distribution de probabilité du vecteur de score $\mathbf{s}^{<t>}$ du mot au pas t qui est calculée via une softmax (eq. 2.1a). Chaque élément $\hat{\mathrm{y}}_i^{<t>}$ représente la probabilité que l'étiquette i soit attribuée au mot du pas t étant donné la séquence $\mathrm{x}^{<1:T>}$ donnée en entrée (eq. 2.1b). À la fin, l'étiquette avec la plus grande probabilité est assignée au mot correspondant (eq. 2.1c). Ainsi, $\hat{i}^{<1:T>}$ représente l'étiquetage prédit pour $\mathrm{x}^{<1:T>}$.

$$\hat{\mathbf{y}}^{<t>} = softmax(\boldsymbol{s}^{<t>}) \tag{2.1a}$$

$$\hat{\mathrm{y}}_i^{<t>} = P(y^{<t>} = i|\mathrm{x}^{<1:T>}) \tag{2.1b}$$

$$\hat{i}^{<t>} = \underset{1\leq i\leq w}{\arg\max}\, P(\mathrm{y}^{<t>} = i|\mathrm{x}^{<1:T>}) \tag{2.1c}$$

Dans cette approche, l'étiquetage de chaque mot est supposé être indépendant des autres mots. Cette hypothèse est trop forte et non réaliste pour une tâche comme la reconnaissance des entités nommées, car il y a une forte dépendance de chaque étiquette avec son voisinage.

Décodage avec CRF.

Les CRF représentent un choix intéressant pour le décodage. Ce modèle probabiliste a été utilisé abondamment dans la littérature et s'est montré très efficace pour cette tâche. Les CRF se démarquent par leur capacité à considérer les étiquettes du voisinage et de raisonner conjointement sur les décisions de marquage pour chaque mot. Lample *et al.* (2016), Ma et Hovy (2016) ainsi que d'autres ont démontré que l'utilisation du CRF permet d'améliorer les résultats de leurs modèles sur la tâche de la reconnaissance des entités nommées.

Dans ce type de décodage, le système de reconnaissance des entités nommées est entraîné à maximiser la probabilité $P(\mathrm{y}^{<1:T>}|\mathrm{x}^{<1:T>})$ où la séquence $\mathrm{y}^{<1:T>}$ est instanciée avec l'étiquetage de la phrase $\mathrm{x}^{<1:T>}$ selon l'ensemble d'entraînement (*ground truth*).

La théorie du CRF linéaire fait l'hypothèse que la probabilité d'un étiquetage s'exprime selon l'expression :

$$P(\mathrm{y}^{<1:T>}|\mathrm{x}^{<1:T>}) = \frac{1}{Z(\mathrm{x}^{<1:T>})} \exp(\sum_{i=1}^{T} s_{y^{<t>}}^{<t>}) + \sum_{i=1}^{T-1} A_{y^{<t>},y^{<t+1>}}) \tag{2.2}$$

où :

– $Z(\mathrm{x}^{<1:T>})$ est la constante de normalisation définie comme suit :

$$Z(\mathrm{x}^{<1:T>}) = \sum_{\bar{y}^{<1>}} \cdots \sum_{\bar{y}^{<T>}} \exp(\sum_{t=1}^{T} s_{y^{<t>}}^{<t>}) + \sum_{i=1}^{T-1} A_{y^{<t>},y^{<t+1>}}) \tag{2.3}$$

– $s_{y^{<t>}}^{<t>}$ représente le score de l'étiquette $y^{<t>}$ au pas t dans la séquence.

– $\boldsymbol{A}$ est une matrice $n \times n$ où chaque élément $A_{i,j}$ est un score qui représente le degré de confiance que l'étiquette i soit suivie par l'étiquette j.

Calculer la fonction Z peut s'avérer impossible en terme de complexité algorithmique, où on peut observer qu'elle est de l'ordre $O(n^T)$ où n est le nombre de lables et T est la taille de la séquence. Ce qui s'avère coûteux en terme de calcul. Afin de calculer la fonction de partition Z dans un temps plausible, l'implémentation utilise un algorithme dynamique (Bellman, 1957).

À la fin, le meilleur étiquetage $\hat{\mathrm{y}}^{<1:T>}$ est calculé à l'aide de l'algorithme de viterbi (Forney, 1973).

$$\hat{i}^{<1:T>} = \arg\max_{\hat{i}^{<1:T>}} P(\mathrm{y}^{<1:T>} = i^{<1:T>}|\mathrm{x}^{<1:T>}) \tag{2.4}$$

Où $1 \leq i^{<t>} \leq n$ et $1 \leq t \leq T$.

2.3 REN et l'apprentissage par transfert (*Transfer learning*)

La technique d'apprentissage par transfert des connaissances consiste à entraîner un modèle d'apprentissage profond sur des corpus gigantesques d'une manière non

supervisée. Ensuite, le modèle résultant est utilisé soit comme un plongement de mots avec leurs contextes (Peters *et al.*, 2018) soit avec une couche d'adaptation à ajuster pour la tâche cible (Devlin *et al.*, 2018). La tâche cible profite ainsi des connaissances acquises auparavant. Récemment, cette technique prend de plus en plus d'ampleur au sein de la communauté d'extraction d'information en TALN. Nous présentons dans cette section les modèles les plus pertinents.

2.3.1 Plongement à partir d'un modèle de langue (ELMo)

L'un des inconvénients majeurs de plongement de mots pré-entraîné est que chaque mot est représenté par le même vecteur quelque soit son contexte (mots polysémiques). Par exemple : le mot «vers» peut signifier «un vers» en poésie, une direction comme dans «vers le sud» ou le pluriel de l'animal «vers».

Peters *et al.* (2017) et Peters *et al.* (2018) ont pris en considération cette problématique par la capture de la signification des mots selon leurs contextes. Au lieu d'utiliser un vecteur de mot unique, ils ont proposé un modèle de langue ELMo qui prend en considération toute la phrase où le mot apparaît. Ce modèle a été pré-entraîné sur la base d'une quantité massive de données d'une manière non supervisée avec comme objectif de prédire le prochain mot dans la séquence. Un autre réseau bidirectionnel est utilisé afin de prédire le mot précédent aussi. Les deux sens permettent d'apprendre une représentation du mot dans son contexte. Le modèle est incorporé par la suite comme un composant dans des tâches de TALN en aval (la reconnaissance des entités nommées, l'étiquetage morpho-syntaxique, etc ....).

L'architecture du modèle se compose d'une couche de plongement de mots et de L couches de Bi-LSTM empilées (voir Figure 2.2a). Les deux contextes à droite $\overleftarrow{\boldsymbol{h}}_{k,j}^{LM}$ et à gauche $\overrightarrow{\boldsymbol{h}}_{k,j}^{LM}$ d'un mot k à la couche j sont concaténés pour former le contexte du mot k à la couche j, $\boldsymbol{h}_{k,j}^{LM} = [\overrightarrow{\boldsymbol{h}}_{k,j}^{LM}, \overleftarrow{\boldsymbol{h}}_{k,j}^{LM}]$. Une somme pondérée est

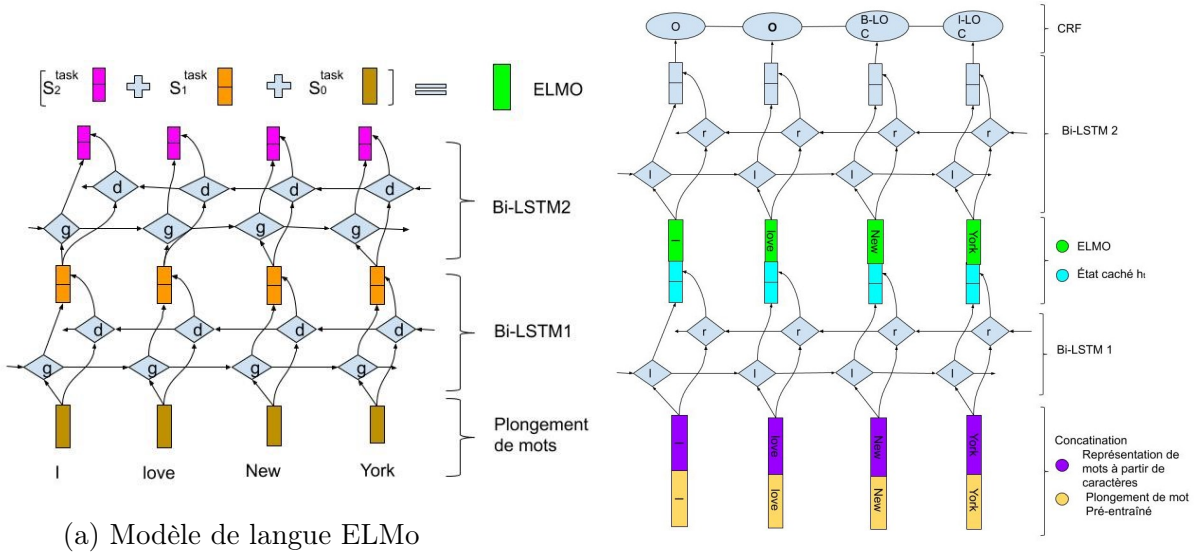
effectuée entre les représentations des L couches Bi-LSTM pour former le vecteur final ELMo (vecteur en vert dans la figure 2.2a) pour le mot k, comme illustré dans la formule qui suit :

$$\mathbf{ELMo}_k^{LM} = \gamma^{task} \sum_{j=0}^{L} (s_j^{task} \boldsymbol{h}_{k,j}^{LM})$$

où :

- L est le nombre de couches de Bi-LSTM.
- s_j^{task} sont des poids normalisés avec la fonction exponentielle normalisée (softmax) avec $0 \leq j \leq L$.
- le paramètre scalaire γ^{task} permet au modèle de mettre à l'échelle le vecteur ELMo.
- k représente l'indice du mot dans la séquence.
- $1 \leq j \leq L$ correspond à l'indice de la couche du Bi-LSTM et j=0 pour le plongement.
- Les poids s_j^{task} sont appris lors de l'entraînement de la tâche finale qui est dans notre cas la reconnaissance des entités nommées.

Peters *et al.* (2018) ont utilisé pour l'extraction des entités nommées la même architecture que Lample *et al.* (2016) en ajoutant une deuxième couche Bi-LSTM (figure 2.2b). La concaténation de plongement de mots (vecteur en jaune) et les représentations de mots à partir de caractères basés sur CNN (vecteurs en violet) est introduite à la première couche Bi-LSTM. Les états cachés issus de cette première couche (vecteurs en cyan) sont concaténés avec le vecteur ELMo (vecteur en vert) du mot correspondant obtenu depuis le modèle de langue pré-entraîné (Voir figure 2.2a). Cette concaténation est passée par la suite vers une deuxième couche Bi-LSTM du modèle de la tâche et le résultat de cette étape est envoyé au CRF pour classification de la séquence (Peters *et al.*, 2018).



(a) Modèle de langue ELMo

(b) Modèle REN principale

Figure 2.2: Plongement à partir d'un modèle de langue (ELMo)

2.3.2 Représentations de l'encodeur bidirectionnel à partir de transformateurs (BERT)

Les modèles de langues (p. ex., ELMo (Peters *et al.*, 2018)) ne sont pas typiquement bidirectionnels en tant que tels, car ils utilisent une concaténation superficielle du contexte gauche et droit après les avoir entraînés indépendamment l'un de l'autre. Contrairement à ces derniers, BERT (Représentations de l'encodeur bidirectionnel à partir de transformateurs) (Devlin *et al.*, 2018) offre une nouvelle méthode d'entraînement appelée le «modèle de langue masqué» qui permet de prendre en considération les préfixes et les suffixes en même temps. En cachant aléatoirement 15% des mots en entrée, cette technique a pour objectif de prédire un mot caché en se basant sur le contexte autour. De cette façon, le modèle est forcé à apprendre toutes les informations de la phrase pour prédire les mots qui manquent. Le modèle BERT est ainsi tout à fait bidirectionnel. D'autre part, une deuxième technique est utilisée par BERT qui est appelée la «prédiction de la

prochaine phrase». Elle consiste à entraîner le modèle pour prédire la prochaine phrase. Cette technique est fort utile pour des tâches comme Question/Réponse ou la classification de paires de phrases.
BERT fait usage d'une nouvelle architecture appelée Transformeur (Vaswani *et al.*, 2017). C'est un modèle basé sur le mécanisme d'attention seulement et sans réseaux de neurones récurrents.
Le transfert de connaissances dans BERT se compose de deux étapes fondamentales :

— Une phase de pré-entraînement est effectuée d'une manière non supervisée sur des données massives non étiquetées (BooksCorpus) (900 millions de mots) (Zhu *et al.*, 2015) et (Wikipedia Anglais) (2,500 millions de mots).

— Une phase de peaufinage «*fine tuning*» est effectuée selon la tâche souhaitée. Cela est fait en ajoutant une couche de classification au dessus et en initialisant le modèle avec les paramètres du modèle pré-entraîné. Seules les paramètres de la couche de classification sont mis à jour par entraînement sur les données étiquetées.

2.3.3 Plongement de chaînes contextuelles (CSE)

Proposé par Akbik *et al.* (2018), le modèle CSE a comme particularité d'opérer sur les caractères comme entités atomiques. Cela permet de traiter le texte comme une séquence de caractères. Il a ainsi l'avantage de mieux manipuler les mots rares, typos, et aussi, de mieux capturer les prefixes/suffixes au sein des mots.
Le modèle de langue CSE a été entraîné sur un large corpus (Chelba *et al.*, 2014) qui contient un milliard de mots. La séquence de caractères est passée à un LSTM bidirectionnel qui va être entraîné à prédire le prochain caractère dans la séquence. En effet, ce modèle de langue essaie de capturer la distribution de probabilité des caractères étant donnés les caractères précédents et les caractères à venir. Le

modèle pré-entraîné est utilisé par la suite dans des tâches ultérieures en adaptant ses paramètres par peaufinage.

Comme illustré dans la figure 2.3, la représentation de mots à partir de caractères est composé d'un Bi-LSTM qui parcourt tous les caractères de la phrase. Chaque état caché issu du LSTM droit (resp. gauche) capture les information depuis le début (resp. depuis la fin) de la séquence jusqu'au caractère en question. Les états cachés du premier et du dernier caractère de chaque mot issus depuis le Bi-LSTM seront concaténés afin de former une représentation de contexte du mot à partir des caractères. Ainsi, le modèle capture les informations sémantico-syntaxiques du mot lui-même y compris son contexte environnant. La représentation obtenue est combinée avec un plongement de mot pré-entraîné tel que Glove pour former une représentation finale du mot. Cette représentation peut être utilisée par la suite pour alimenter un modèle d'encodage-décodage de séquence tel que la reconnaissance des entités nommées.

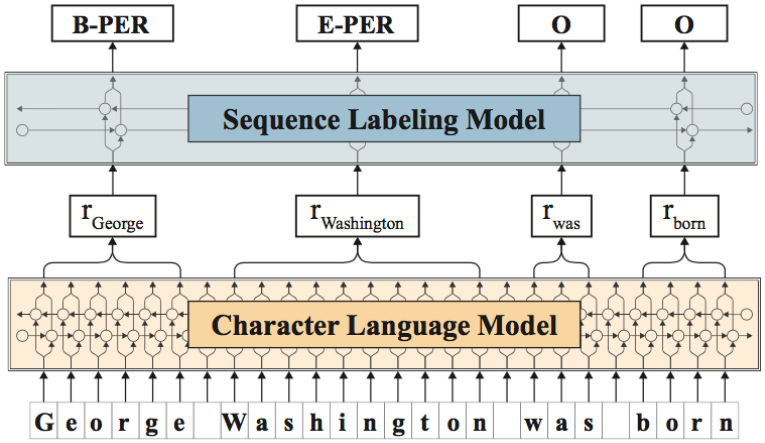


Figure 2.3: Modèle de plongement de chaînes contextuelles (CSE) (Akbik *et al.*, 2018)

Tableau 2.1: Tableau récapitulatif des résultats des systèmes REN basés sur l'apprentissage profond.

Référence	Représentation de mots à partir de caractères	Plongement de mots pré-entrainé	Données externes ou manuelles	Encodeur du contexte	Décodeur	Mesure-F%
Collobert *et al.* (2011b)	N/A	SENNA	POS	CNN	CRF	89.86
Huang *et al.* (2015)	N/A	SENNA	N-grammes Dictionnaires Orthographe	LSTM	CRF	90.10
Chiu et Nichols (2015)	CNN	SENNA	Orthographe Lexique	LSTM	CRF	91.62 ±0.33
Rei *et al.* (2016)	LSTM	Word2Vec	N/A	LSTM	CRF	84.09
Yang *et al.* (2016)	GRU	SENNA	Dictionnaires	GRU	CRF	90.2
Lample *et al.* (2016)	LSTM	Word2vec	N/A	LSTM	CRF	90.94
Ma et Hovy (2016)	CNN	Glove	N/A	LSTM	CRF	91.21
Strubell *et al.* (2017)	N/A	SENNA	N/A	ID-CNN	CRF	90.65 ±0.15
Yang *et al.* (2018)	CNN	Glove	N/A	CNN	CRF	90.26 ±0.19
Peters *et al.* (2017)(TagLM)	GRU	SENNA	Modèle de langue	GRU	CRF	91.93 ±0.19
Peters *et al.* (2018)(ELMo)	CNN	N/A	Modèle de langue	LSTM	CRF	92.22 ±0.10
Devlin *et al.* (2018)(BERT)	N/A	N/A	N/A	Transformeur	Softmax	92.8
Akbik *et al.* (2018)(CSE)	LSTM (Modèle de langue)	Glove	N/A	LSTM	CRF	**93.10 ±0.12**

2.4 Récapitulatif des résultats REN

Dans le tableau 2.1, nous avons regroupé les résultats des travaux que nous avons décrits dans ce chapitre. Il s'agit principalement des travaux qui ont utilisé l'apprentissage profond et qui ont traité le jeu de données CoNLL-2003 (Sang et De Meulder, 2003). La présentation sous forme de tableau permet facilement de comparer la performance des systèmes incluant une brève description des technique utilisées.

Nous pouvons constater, à partir du tableau 2.1, que les modèles basés sur l'apprentissage par transfert tels que CSE, BERT et ELMo sont meilleurs. Cela est

dû à leurs pré-entraînements sur des données massives et des machines dotées de capacités de calcul supérieures.

Plusieurs travaux à l'instar de Lample *et al.* (2016), Ma et Hovy (2016) et autres ont affirmé que l'utilisation d'une représentation de mots à partir de caractères a amélioré la performance de leurs modèles.

L'importance de l'utilisation d'un plongement de mots pré-entraîné a été observée par la plupart des travaux (Lample *et al.*, 2016), (Yang *et al.*, 2018), etc. De plus, le type et la taille du plongement de mots peut améliorer la performance du modèle (Reimers et Gurevych, 2017).

L'utilisation du plongement de mots et la représentation de mots à partir de caractères conjointement semblent être une bonne stratégie. Nous pouvons constater cela sur la plupart des travaux (Yang *et al.*, 2018), (Peters *et al.*, 2017), etc.

Ajouter les données externes telles que les dictionnaires, l'étiquetage morpho-syntaxique (POS), etc., en plus de la représentation de mots à partir de caractères et le plongement de mots pré-entraîné, permet d'améliorer le résultat du modèle ; cela peut être observé avec le résultat de Chiu et Nichols (2015). Cependant, l'élaboration de ces données externes est chronophage, nécessite beaucoup de travail manuel et implique une dépendance du modèle au domaine.

Le modèle de Rei *et al.* (2016) qui utilise un mécanisme d'attention pour combiner la représentation de mots à partir de caractères et le plongement de mots pré-entraîné n'a pas donné de bons résultats.

2.5 Conclusion

Nous avons présenté, dans ce chapitre, un aperçu sur les approches proposées dans la littérature pour la tâche de reconnaissance des entités nommées au moyen de

méthodes d'apprentissage machine. Notre état de l'art a été établi dans le but de suivre le progrès de ces travaux au fil du temps. Les approches classiques basées sur l'ingénierie manuelle des caractéristiques peuvent donner des résultats acceptables. Cependant, ces approches sont coûteuses et nécessitent de l'expertise dans le domaine de la tâche. Par la suite, nous avons décrit les différentes architectures utilisées dans l'élaboration d'un modèle de reconnaissance des entités nommées basé sur l'apprentissage profond et l'apprentissage par transfert de connaissances. Nous proposons, dans le prochain chapitre trois modèles basés sur l'apprentissage profond pour la reconnaissance des entités nommées comme la contribution du mémoire. Nos modèles représentent des améliorations de deux modèles de base que nous avons retenus afin d'évaluer la valeur ajoutée de nos modèles.

CHAPITRE III

APPROCHES PROPOSÉES

Dans ce chapitre, nous proposons trois modèles pour la reconnaissance des entités nommées. Nos modèles sont basés sur l'apprentissage profond et n'utilisent ni des données externes ni l'ingénierie manuelle des caractéristiques. Bien que nos modèles soient indépendants du corpus d'évaluation, le jeu de données CoNLL-2003 (Sang et De Meulder, 2003) sera utilisé au chapitre suivant pour les évaluer.

Ce chapitre sera donc dédié à la présentation des modèles de base ainsi que nos modèles en suivant la même démarche utilisée pour le cadre des systèmes de reconnaissance des entités nommées basé sur l'apprentissage profond vue dans le chapitre précédent. Dans cette optique, nous visons alors à décrire les rouages qui rentrent dans la construction des différentes couches qui constituent nos modèles et notre intuition.

Tout d'abord, nous allons présenter en premier lieu deux modèles de base : Base1 de Lample *et al.* (2016) et Base2 de Ma et Hovy (2016) afin d'éclaircir notre contribution par rapport à ces deux derniers. Par la suite, nous allons décrire les architectures de nos trois modèles proposés. Notre premier modèle (Modèle1) est basé sur une somme pondérée entre la représentation de mots à partir de caractères (RMC) et la représentation du contexte d'un plongement de mots pré-entraîné (CPM). Le deuxième modèle (Modèle2) s'inspire de la technique de connexion

par saut (He *et al.*, 2015) pour la mise à jour du contexte, tandis que le dernier modèle (Modèle3) représente une combinaison de deux techniques CNN et LSTM pour l'extraction d'une représentation de mots à partir de caractères.

3.1 Modèles de base

Dans le but d'évaluer les performances des modèles que nous allons proposer dans ce chapitre, nous présentons deux modèles de base qui serviront comme modèles de comparaison. À cette fin, nous avons prévu d'implémenter les architectures de Lample *et al.* (2016) (base1) et Ma et Hovy (2016) (base2) que nous avons déjà vues dans le chapitre de l'état de l'art. Ces deux approches sont considérées comme les modèles d'apprentissage profond les plus fondamentaux de l'état de l'art, surtout en ce qui concerne les tâches d'étiquetage de séquences à l'instar de la reconnaissance des entités nommées, l'étiquetage morpho-syntaxique et l'analyse syntaxique de surface basés sur les réseau de neurones récurrents.

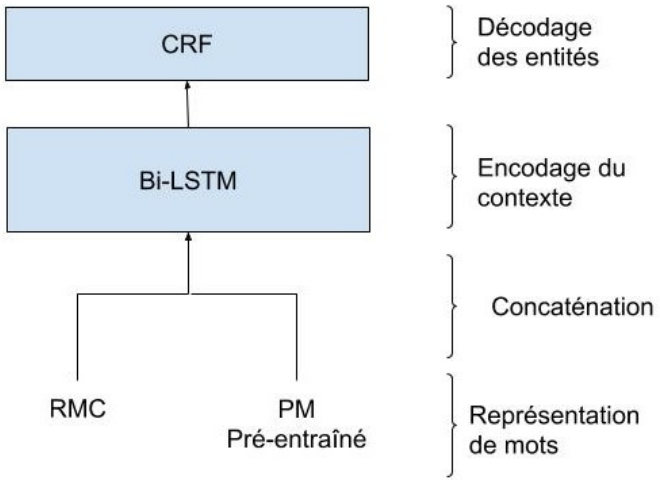
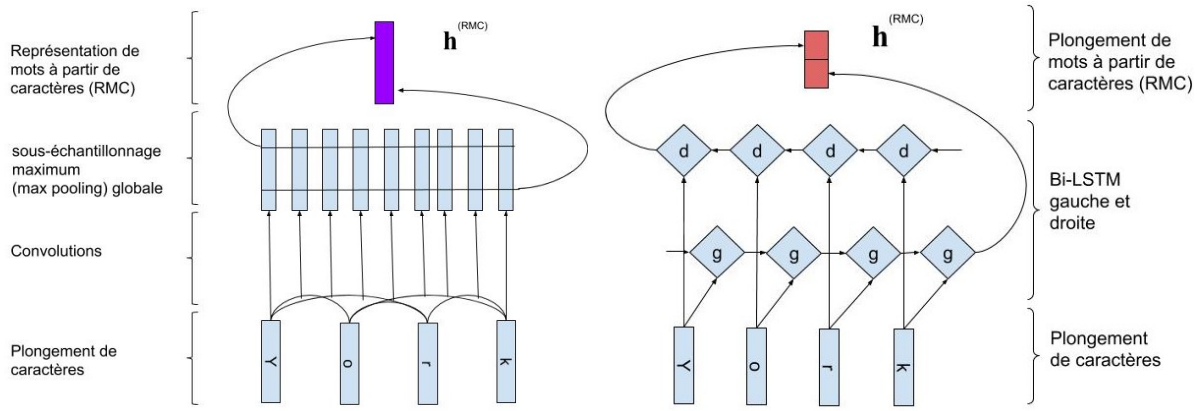


Figure 3.1: Architecture modèles de base

La figure 3.1 présente une architecture générique pour les deux modèles. En effet, les architectures des modèles Base1 et Base2 sont identiques à l'exception de la



(a) CNN (Ma et Hovy, 2016) (b) LSTM (Lample *et al.*, 2016)

Figure 3.2: Représentation de mots à partir de caractères

technique utilisée au niveau de la représentation de mots à partir de caractères (RMC). Le modèle de Lample *et al.* (2016) Base1 utilise un réseau Bi-LSTM pour la représentation de mots à partir de caractères tel qu'illustré dans la figure 3.2b.

D'autre part, Ma et Hovy (2016) font usage d'un réseau CNN pour extraire une représentation des mots à base de caractères. En effet, une couche de convolutions suivie d'une couche de sous-échantillonnage maximum global, qui se retrouvent dans la figure 3.2a, sont utilisées.

Dans les deux modèles de base, la représentation de mots à partir de caractères est concaténée par la suite avec un plongement de mots pré-entraîné [1] pour obtenir la représentation finale des mots. Un Bi-LSTM reçoit la représentation finale des mots pour l'encodage du contexte qui va alimenter un CRF pour le décodage des entités.

Dans le cadre de notre projet, nous nous sommes inspirés de ces deux modèles

1. Dans notre implémentation, nous avons utilisé le PM pré-entraîné Glove à ce niveau.

de base et avons donc expérimenté plusieurs architectures, parmi lesquelles nous avons choisi trois modèles.

Notre première intuition a été d'essayer d'apporter une solution au problème des mots rares qui sont hors vocabulaire [2]. Pour cela, nous avons proposé un modèle (Modèle 1) qui combine deux types de représentations de mots via une somme pondérée au lieu d'une simple concaténation. De plus, nous avons ajouté une couche Bi-LSTM dans le but d'extraire une représentation contextuelle à partir d'un plongement de mots pré-entraîné.

Dans le deuxième modèle (Modèle 2), nous avons essayé d'appliquer le concept de connexion par saut (He *et al.*, 2015) sur le modèle de Lample *et al.* (2016), et ce, en combinant l'encodage du contexte avec la représentation de mots à partir de caractères via une somme pondérée.

Dans une tentative d'améliorer la représentation de mots à partir de caractères (RMC), nous avons essayé, dans notre troisième modèle (Modèle 3), de combiner les représentations de mots à partir de caractères à base de CNN (Ma et Hovy, 2016) et celle à base de LSTM (Lample *et al.*, 2016).

Dans ce qui suit, nous présenterons en détail les trois modèles proposés. Notre objectif étant de voir si les modifications apportées aux modèles de base pourraient donner de meilleurs résultats.

3.2 Modèle 1 : Une somme pondérée pour la représentation de mots

Notre premier modèle se constitue de trois couches principales (Voir figure 3.3), la première étant la **représentation de mots**. Cette dernière est obtenue par une

2. Les mots hors vocabulaire sont les mots qui n'ont pas de plongement de mots pré-entraîné

somme pondérée entre la représentation de mots à partir de caractères (RMC) et le contexte extrait à partir d'un plongement de mots pré-entraîné (CPM). La deuxième couche représente un réseau Bi-LSTM pour l'**encodage du contexte** de mots. Nous avons employé un CRF linéaire comme dernière couche pour le **décodage des entités**. Les différentes couches sont détaillées dans ce qui suit.

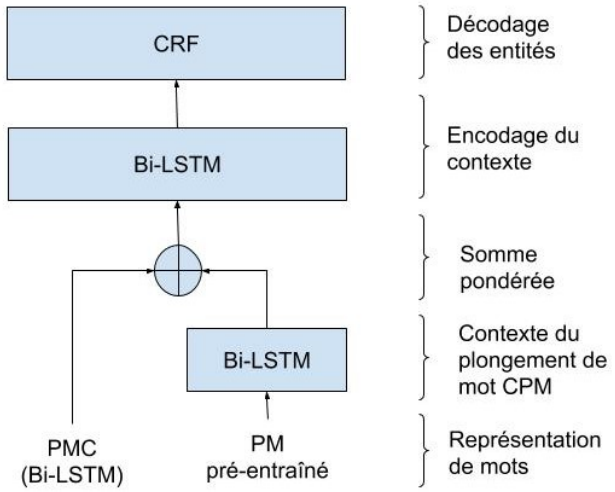


Figure 3.3: Modèle 1

3.2.1 Représentation de mots

– **Représentation de mots à partir de caractères.** Afin de capturer la sensibilité orthographique des mots, nous allons extraire des caractéristiques morphologiques à partir de leurs caractères. Ce type de représentation s'est montré efficace dans la littérature, notamment pour représenter les mots rares qui n'ont pas un plongement de mots pré-entrainé. De plus, il nous permet d'éviter l'usage de l'ingénierie manuelle des caractéristiques.

Pour parvenir à cette fin, nous avons eu recours à un réseau Bi-LSTM qui va parcourir les caractères d'un mot (p.ex., York) de manière bidirectionnelle. Nous alimentons chaque caractère d'un mot donné (dans notre exemple :

Y, o, r, k) à notre réseau en utilisant un plongement de caractère aléatoire initial. Ce plongement initial est un paramètre entraînable. À la fin, chaque mot sera représenté par la concaténation des deux derniers états cachés du réseau Bi-LSTM (figure 3.2b). La représentation du mot $\boldsymbol{h}^{(RMC)}$ est un vecteur (le vecteur en rouge) qui a une taille deux fois plus grande que la taille de l'état caché du LSTM.

– **Représentation du contexte d'un mot.** Comme nous l'avons déjà vu dans la littérature, le plongement de mots pré-entraîné (PM) a fait preuve d'efficacité, car il peut nous indiquer la sémantique des mots. En outre, ces représentations sont denses, de faibles dimensions et conçues automatiquement par apprentissage non supervisée depuis de larges corpus. L'utilisation du plongement de mots pré-entraîné permet de surmonter le problème de données d'entraînements limitées en nombre.

  Le plongement de mots pré-entraîné incorpore des caractéristiques individuelles du mot. Afin de représenter chaque mot en tenant compte de son contexte dans notre corpus, nous avons employé un réseau Bi-LSTM qui prend en entrée la séquence composée des plongements de mots de la phrase en entrée tel qu'illustré dans la figure 3.4. Dans notre exemple (Peter loves New York), Chaque mot de la phrase sera représenté par son plongement de mot. Ces vecteurs seront fournis par la suite à un Bi-LSTM pour l'extraction des contextes des mots. Le contexte du mot à la position i sera représenté par la concaténation des deux états cachés à la position i $\boldsymbol{h}_i^{(CPM)}$ (les vecteurs en vert).

– **La somme pondérée.** L'inconvénient majeur lors de l'utilisation d'un plongement de mots pré-entraîné est parfois l'indisponibilité du plongement de certains mots non observés auparavant dans le corpus sur lequel il a été entraîné. Par conséquent, on se retrouve souvent à représenter ces derniers

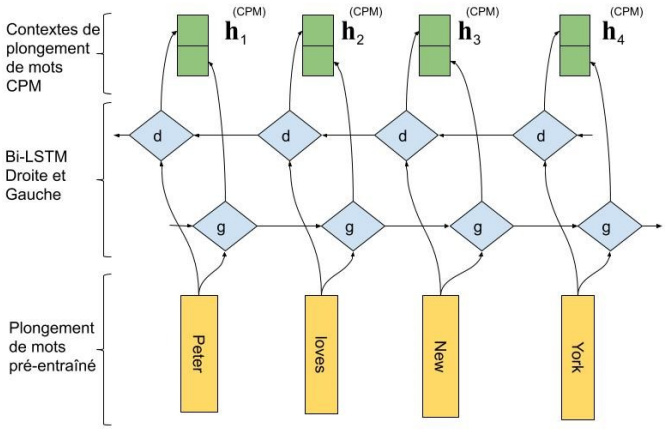


Figure 3.4: L'extraction du contexte à partir du plongement de mots pré-entraîné

par un vecteur générique partagé appelé UNK (*unknown*) [3]. Également, le manque d'information dans le plongement des mots rares représente un inconvénient majeur de la représentation distributionnelle.

D'autre part, la capacité de la représentation de mots à partir de caractères (RMC) permet d'extraire des informations morphologiques qui pourraient être utiles afin de pallier à ce manque d'informations sur les mots rares. Par exemple, le nom rare «Abdulsamed» n'a pas été observé auparavant ; par contre, le plongement de ce mot à partir de ses caractères à déjà appris le préfixe «Abdul» avec un autre nom plus connu, ce qui permet de déduire qu'il s'agit d'un nom propre.

L'approche souvent utilisée dans la littérature est la concaténation de ces deux représentations RMC et PM pré-entraînées qui donnent lieu à des vecteurs longs et biaisés par le vecteur générique UNK. Par conséquent, nous avons mis en place une somme pondérée qui offre au modèle la capacité de choisir la combinaison la plus avantageuse des deux représentations.

3. À l'exception de *FastText*, car il fournit un plongement de mots à partir des n-grammes

Nous estimons que cela va améliorer la qualité d'informations discriminantes contenues dans les vecteurs représentatifs de mots et remédier au problème des mots rares dans les plongements de mots pré-entraînés.
Formellement, ce mécanisme consiste à effectuer une somme pondérée entre le contexte du plongement de mots (CPM) et la représentation de mots à partir de caractères (RMC). Nous présentons cette formulation comme suit pour le mot à la position i :

$$\boldsymbol{h}_i^{(SP)} = \beta(\sigma(\alpha)\tilde{\boldsymbol{h}}_i^{(CPM)} + (1 - \sigma(\alpha))\tilde{\boldsymbol{h}}_i^{(RMC)}) \tag{3.1}$$

où :

— $\tilde{\boldsymbol{h}}_i^{(CPM)}$ (resp. $\tilde{\boldsymbol{h}}_i^{(RMC)}$) sont des vecteurs normalisés obtenus en soustrayant la moyenne $\mu^{(CPM)}$ (resp. $\mu^{(RMC)}$) et en divisant par la variance $(\sigma^{(CPM)})^2$ (resp. $(\sigma^{(RMC)})^2$) du mini-batch.

$$\tilde{\boldsymbol{h}}_i^{(CPM)} = \frac{\boldsymbol{h}_i^{(CPM)} - \mu^{(CPM)}}{\sqrt{(\sigma^{(CPM)})^2 + \epsilon}} \quad \text{et} \quad \tilde{\boldsymbol{h}}_i^{(CPM)} = \frac{\boldsymbol{h}_i^{(RMC)} - \mu^{(RMC)}}{\sqrt{(\sigma^{(RMC)})^2 + \epsilon}}$$

Nous ajoutons ϵ (égale à 1E-12) qui est un nombre infiniment petit pour éviter la division par zero dans le cas où la variance est nulle.
Pendant l'entraînement nous allons garder une estimation moyenne des moyennes (resp. variances) des différentes mini-batchs afin de l'utiliser lors du test.

— α : est un poids normalisé avec sigmoid σ et considéré comme paramètre.

— β : est un paramètre scalaire qui permet au modèle de mettre à l'échelle le vecteur en entier.

3.2.2 Encodage du contexte

Les vecteurs représentatifs des mots $(\boldsymbol{h}_1^{(SP)}, \boldsymbol{h}_2^{(SP)}, \ldots, \boldsymbol{h}_T^{(SP)})$ obtenus à la sortie de la somme pondérée sont fournis en entrée à un autre réseau Bi-LSTM afin d'extraire le contexte droit et gauche correspondant à chaque mot. Les deux contextes sont concaténés pour former les contextes finaux de mots $(\boldsymbol{h}_1^{(CF)}, \ldots, \boldsymbol{h}_T^{(CF)})$.(voir figure 3.5)

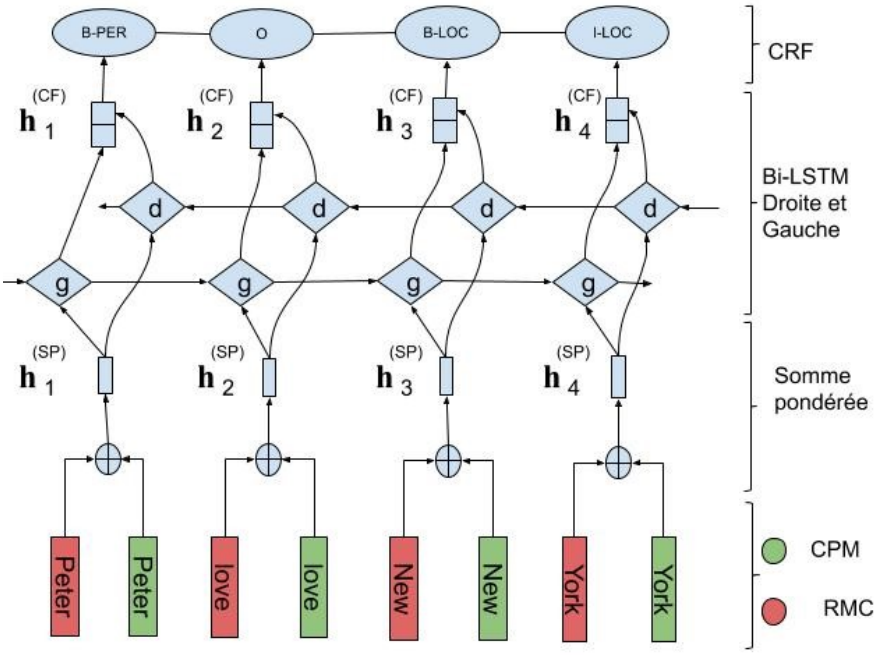


Figure 3.5: Architecture principale du Modèle 1

3.2.3 Décodage des entités

Nous avons effectué le décodage des entités à l'aide d'un réseau CRF linéaire tel que présenté dans le chapitre précédent.

3.3 Modèle 2 : une somme pondérée avec une connexion par saut

Théoriquement, la superposition de plusieurs couches de réseau de neurones peut être favorable à l'amélioration des performances de l'apprentissage. Cependant, il a été observé dans la pratique qu'après une certaine limite de couches superposées, la performance du réseau commence à se saturer et finit par se dégrader. En effet, à mesure que le réseau s'approfondit, le réseau souffre de plus en plus du problème de la disparition du gradient (Bengio *et al.*, 1994).

He *et al.* (2015) dans leur modèle ResNet, ont proposé une nouvelle technique de connexion par saut «*skip connection*» qui permet au besoin d'ignorer certaines couches dans un réseau profond, et ce dans le but d'achever au moins la performance d'un réseau peu profond sinon mieux.

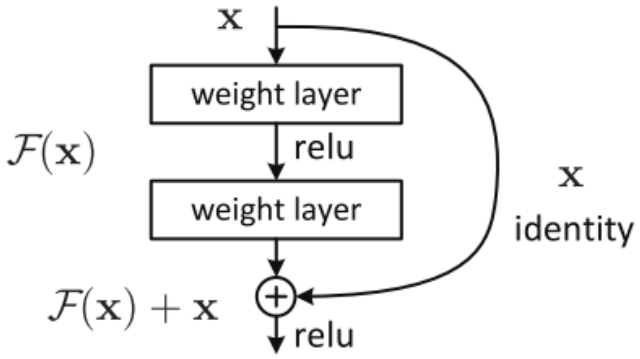


Figure 3.6: Un bloque de connexion par saut (He *et al.*, 2015)

La figure 3.6 illustre le concept de connexion par saut. La formulation $\mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$ représente un raccourci de connexion dans un réseau de neurones à propagation directe (*feed-forward neural network*) où les sorties de $\boldsymbol{x}$ sont ajoutées à une couche plus profonde. Cette opération est appelée le mappage d'identité (*identity mapping*). Par conséquent, ce mappage permet au modèle de retrouver son identité $\boldsymbol{x}$ si le mappage $\mathcal{F}(\boldsymbol{x})$ est égal à zéro. Cela signifie que si le modèle ne s'améliore

pas, il ne perdra pas sa performance, car apprendre à mettre $\mathcal{F}(\boldsymbol{x})$ à zéro est une solution facile à atteindre par le réseau.

Grâce à cette technique, on peut propager les gradients d'erreurs plus loin vers les couches initiales tout en permettant à ces couches d'apprendre aussi rapidement que les couches finales.

Dans notre modèle (voir figure 3.7), nous nous sommes inspirés de la technique de connexion par saut en l'appliquant sur le modèle de Lample *et al.* (2016). Nous espérons par cela que dans le pire cas notre modèle obtiendra un résultat similaire au modèle de Lample *et al.* (2016) (voir figure 3.1).

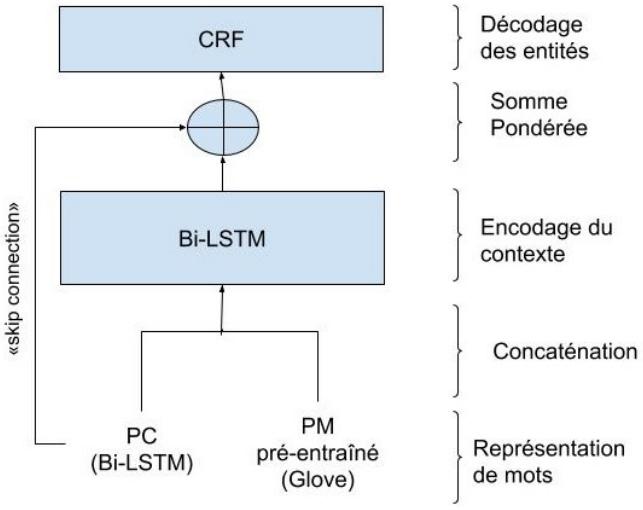


Figure 3.7: Modèle 2

3.3.1 Représentation de mots

Afin de former une représentation finale des mots à l'entrée de l'étape d'encodage, nous avons cette fois concaténé tout simplement le plongement de mots pré-entraîné avec la représentation de mots à partir de caractères (RMC).

3.3.2 Encodage et décodage du contexte

À ce niveau, nous avons appliqué la technique de connexion par saut. Ainsi, nous avons effectué une combinaison linéaire de l'encodage du contexte de mot $\boldsymbol{h}^{(CF)}$ avec la représentation de mots à partir de caractères qui correspond. Nous espérons par cela renforcer et réalimenter notre encodage du contexte avec les caractéristiques discriminantes extraites au niveau de caractères afin de former un nouveau contexte par somme pondérée $\boldsymbol{h}_i^{(SP)}$, tel qu'illustré dans la figure 3.8. Nous formulons ce mécanisme comme suit :

$$\boldsymbol{h}_i^{(SP)} = \beta(\sigma(\alpha)\tilde{\boldsymbol{h}}_i^{(RMC)} + (1 - \sigma(\alpha))\tilde{\boldsymbol{h}}_i^{(CF)}) \tag{3.2}$$

— $\tilde{\boldsymbol{h}}_i^{(RMC)}$ et $\tilde{\boldsymbol{h}}_i^{(CF)}$ sont normalisés par la même procédure que l'équation 3.1

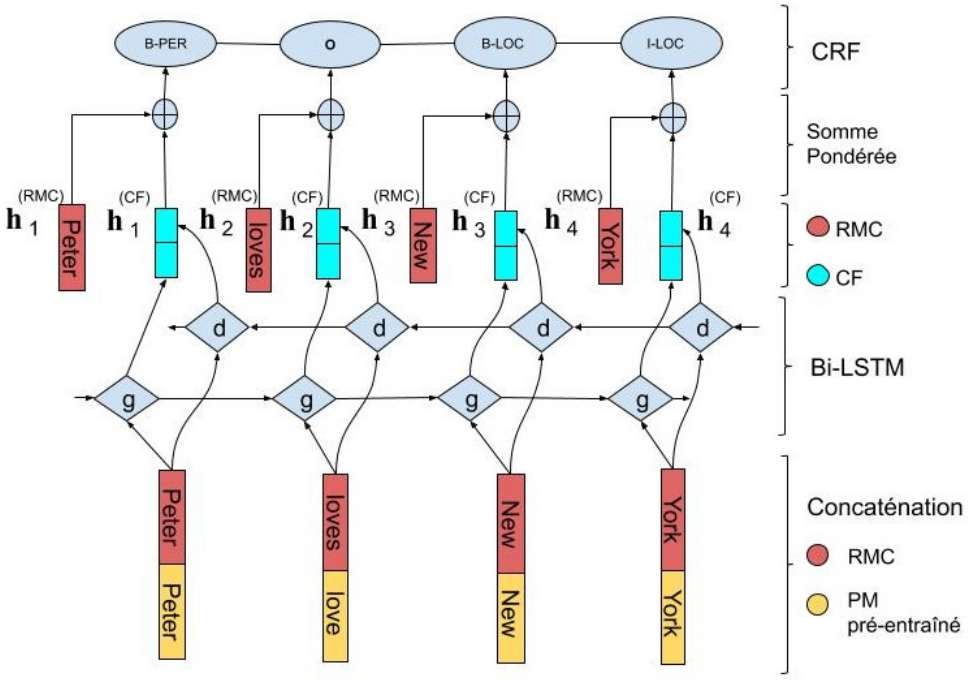


Figure 3.8: Architecture principale du modèle 2

3.3.3 Décodage des entités

Nous avons envisagé à ce niveau d'utiliser les CRF pour l'inférence des entités de la même manière que le modèle 1.

3.4 Modèle 3 : Une combinaison CNN-LSTM pour la représentation de mots à partir de caractères

La figure 3.9 présente un aperçu de l'architecture globale de notre troisième modèle. Le principal changement dans ce modèle par rapport à un modèle de base est la combinaison d'un CNN et d'un Bi-LSTM en série pour l'extraction d'une représentation de mots à partir de caractères (RMC).

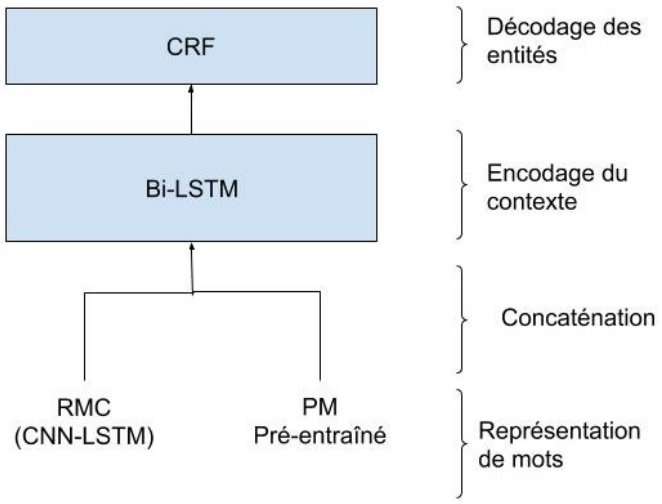


Figure 3.9: Modèle 3

3.4.1 Représentation de mots.

Comme nous l'avons vu dans les modèles de base, il existe deux solutions proposées dans la littérature pour l'extraction d'une représentation de mots à partir

de caractères : la première est basée sur un réseau LSTM (p. ex., Lample *et al.* (2016)) et la deuxième est basée sur un réseau CNN (p. ex., Ma et Hovy (2016)). Chaque technique a montré ses avantages par rapport à l'autre. De ce fait, nous avons supposé que la combinaison entre ces deux techniques pourrait apporter une amélioration dans l'extraction des caractéristiques.

Notre technique se scinde en deux sous-couches. La première consiste à effectuer une couche de convolution 1D avec un réseau CNN. Cette couche prend en entrée une représentation initiale aléatoire pour chaque caractère du jeu de caractères. Cela va nous permettre d'encoder les caractères en vecteurs. Les cartes des caractéristiques issues des convolutions sont converties en une séquence de vecteurs qui vont alimenter un réseau Bi-LSTM. Nous considérons seulement les états cachés finaux du Bi-LSTM pour former une représentation de mots à partir de caractères.

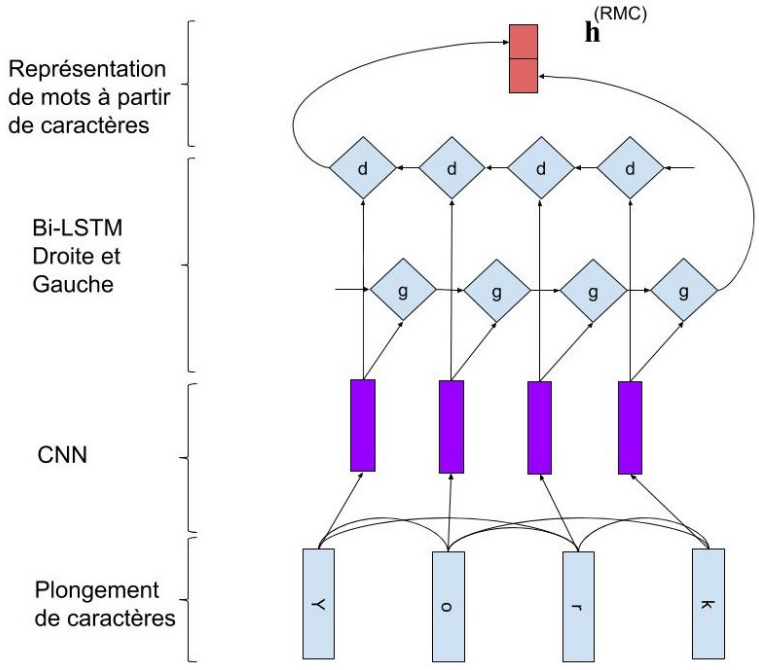


Figure 3.10: CNN-LSTM pour le plongement de mots à partir de caractères

Par la suite, la concaténation de plongement de mots pré-entraîné avec la re-

présentation de mots à partir de caractères forment la représentation finale des mots.

3.4.2 Encodage et décodage du contexte.

Le restant du modèle demeure inchangé tel qu'il a été décrit dans le modèle 1 (figure 3.11), la représentation finale des mots alimente un Bi-LSTM pour l'encodage du contexte, puis nous utilisons un CRF pour le décodage des entités.

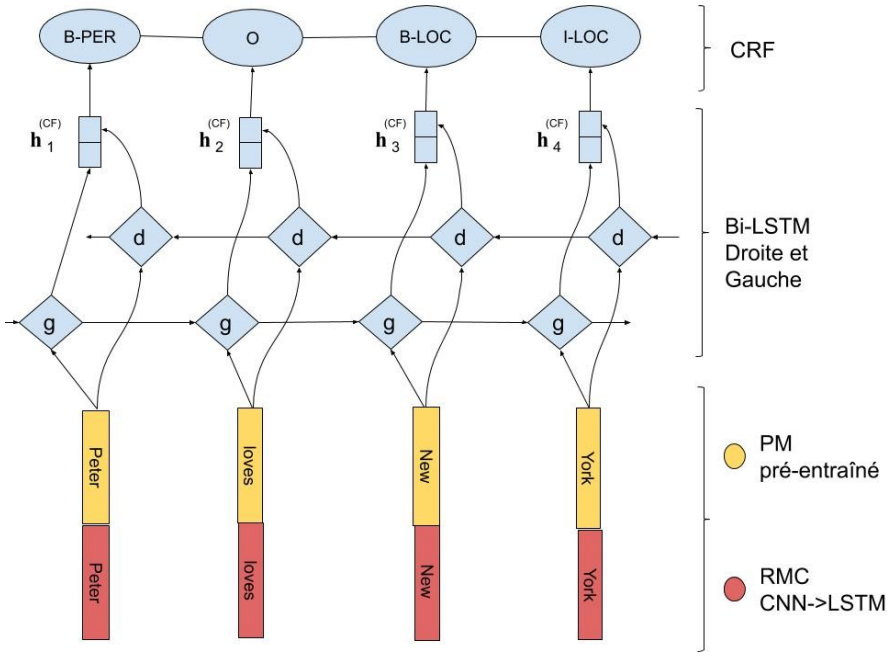


Figure 3.11: Architecture principale du Modèle 3

3.5 Conclusion

Dans ce chapitre, nous avons présenté la première partie de notre méthodologie. Dans un premier, temps nous avons présenté deux modèles de référence afin de quantifier la valeur ajoutée de notre contribution. Par la suite, nous avons mis en

évidence l'architecture des trois modèles proposés ainsi que leur fonctionnement et notre intuition derrière. De plus, nous avons présenté chacune des couches qui constituent nos modèles en donnant les techniques utilisées.

Nos modèles basés sur l'apprentissage profond permettent la reconnaissance des entités nommées. Les modèles ne font intervenir ni des ressources externes ni l'ingénierie manuelle des caractéristiques.

Le chapitre suivant détaillera les différentes expériences menées pour valider ce travail. Nous commençons par une série d'expérimentations dans le but de déterminer les hyperparamètres adéquats pour l'entraînement des modèles proposés. Par la suite, nous présentons et discutons les différents résultats obtenus sur le jeu de données CoNLL-2003 (Sang et De Meulder, 2003).

CHAPITRE IV

EXPÉRIMENTATIONS ET RÉSULTATS

Après avoir présenté les modèles proposés [1] dans le chapitre précédent, nous présentons, dans ce chapitre, une étude empirique de performance. Nous commençons en premier lieu par la définition de notre cadre expérimental en décrivant les différentes configurations utilisées pour l'évaluation de nos modèles. Dans un second temps, nous mettons en évidence l'impact du choix des hyperparamètres sur la performance de nos modèles. Ainsi, nous espérons apporter des précisions sur la méthode de choix des hyperparamètres d'un système basé sur l'apprentissage profond. Par la suite, nous allons rapporter une synthèse des résultats obtenus sur le jeu de données CoNLL-2003 (Sang et De Meulder, 2003) ainsi que ceux d'autres travaux. Finalement, nous allons discuter la pertinence de ces résultats.

1. Les différents modèles mis en œuvre sont disponibles sur notre github https ://github.com/uqam-project/REN.

4.1 Configurations

4.1.1 Données

Afin de nous comparer aux autres approches dans la littérature, nous avons utilisé le jeu de données de la compagne d'évaluation CoNLL-2003 (Sang et De Meulder, 2003) pour la tâche de reconnaissance des entités nommées pour la langue anglaise. Ce jeu de données annoté manuellement a été extrait à partir du corpus «*Reuters*» qui est une collection d'articles. Chaque mot a été étiqueté selon le schéma de marquage BIO avec une des neuf étiquettes : B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, B-MISC, I-MISC et O pour les noms de personnes, organisations, lieux «*locations*», divers «*miscellaneous*» et hors entité respectivement.
Le jeu de données est divisé en trois fichiers :

— Entraînement (Train) : représente la plus grande partie des données. Cet ensemble sert à régler les paramètres du modèle par optimisation.

— Développement (Dev) : il permet de choisir les hyper-paramètres qui ne peuvent pas être réglés par optimisation tels que le taux d'apprentissage et autres (Goodfellow *et al.*, 2016).

— Test (Test) : il permet de mesurer les performances du système.

De plus, un script «*conlleval*» est offert afin de calculer les métriques d'évaluation des performances.
Les statistiques du jeu de données sont illustrées dans le tableau 4.1

Tableau 4.1: Statistiques du jeu de données CoNLL2003

Données	Train	Dev	Test	Label
#Article	946	216	231	9
#Phrase	14,987	3,466	3,684	
#Jeton	203,621	51,362	46,435	
#Entité	23,499	5,942	5,658	

Entité	Train	Dev	Test
LOC	7140	1837	1668
MISC	3438	922	702
ORG	6321	1341	1661
PER	6600	1842	1617

4.1.2 Environnement

Nous avons effectué l'entraînement des modèles sur la machine virtuelle de Google Colab [2]. Cette machine dispose d'une carte graphique Nvidia Tesla k80, CPU Intel Xeon 2.3 Mhz et 12.6 Go de RAM, ce qui permet un temps d'exécution très optimal d'environ 18 minutes par modèle. Les modèles ont été implémentés en utilisant la bibliothèque Tensorflow [3].

4.1.3 Évaluation

Afin d'évaluer les performances de nos modèles, nous avons considéré les métriques précision, rappel et la mesure-F telles qu'expliquées dans le chapitre 1. Le script «*conlleval*» fourni par CoNLL-2003 (Sang et De Meulder, 2003) est utilisé pour cette fin. Les résultats obtenus sur l'ensemble de données test seront rapportés.
Dans la littérature, certains travaux ont rapporté seulement la meilleure performance (mesure-F) obtenue sur une série d'exécutions. Cependant, cette mesure n'est pas informative puisque les réseaux de neurones sont de nature non déter-

2. https ://colab.research.google.com

3. https ://www.tensorflow.org

ministe. De ce fait, nous avons décidé de calculer la moyenne et l'écart type de la mesure-F sur cinq exécutions pour chaque expérience.

4.1.4 Régularisation

L'arrêt précoce *«early stopping»* est une forme de régularisation utilisée pour éviter le sur-apprentissage des modèles. Cette technique sert à arrêter le processus d'entraînement lorsque la performance du modèle commence à se dégrader sur l'ensemble de développement par rapport à l'ensemble d'entraînement. L'idée principale est qu'à chaque fois que le taux d'erreur sur l'ensemble de validation s'améliore, nous enregistrons une copie des paramètres du modèle. Lorsque l'algorithme d'apprentissage se termine, nous restaurons ces paramètres, plutôt que les derniers paramètres obtenus. L'algorithme se termine lorsqu'aucun paramètre ne s'est amélioré par rapport au meilleur taux d'erreur sur l'ensemble de validation enregistrée pour un nombre d'itérations prédéfini. Par cela, nous empêchons le modèle de se spécifier beaucoup sur les données d'entraînement. Il est à noter que nous avons appliqué cette technique après le choix des hyperparamètres.

4.1.5 Hyperparamètres

Trouver la configuration optimale d'un modèle basé sur l'apprentissage profond peut s'avérer un point critique et sensible et qui peut influencer sa performance. De ce fait, nous nous sommes penchés sur cette question et nous avons mis en évidence les principaux facteurs à régler pour un meilleur rendement. La section suivante présente les expériences que nous avons menées pour le choix des différents hyperparamètres.

4.2 Évaluation des hyperparamètres

Le choix des hyperparamètres est une étape délicate dans l'entraînement d'un modèle d'apprentissage automatique. La recherche de la meilleure configuration n'est pas évidente. De ce fait, il existe différentes méthodes pour la sélection de ces hyperparamètres et parmi elles, nous trouvons la méthode de *baby-sitting*[4] du modèle. Le *baby-sitting* d'un modèle consiste à l'évaluer en changeant les variables manuellement au fur et à mesure. Cette stratégie est aussi appelé «*Pandas*». Une autre stratégie appelée «*Caviar*» consiste à entraîner plusieurs modèles en parallèle en évaluant différents hyperparamètres. «*Grid search*» permet de choisir les combinaisons d'hyperparamètres qui vont alimenter la stratégie «*caviar*». Toutefois, la recherche des combinaisons possibles dans un espace large est très coûteux en matière de temps et de calcul. Le «*Random search*» s'avère plus efficace et moins coûteux, où des combinaisons aléatoires des hyperparamètres sont utilisées. L'avantage principal du «*Random search*» est qu'il n'y a pas de cycles expérimentaux gaspillés lorsque deux valeurs d'un hyperparamètre (étant donné les autres hyperparamètres) donnent le même résultat dans deux exécutions différentes. Dans le cas de «*Grid search*», les autres hyperparamètres auraient les mêmes valeurs pour ces deux exécutions, alors qu'avec «*Random search*», ils auraient généralement des valeurs différentes. Par conséquent, si le changement entre ces deux valeurs ne fait pas beaucoup de différence en termes d'erreur dans l'ensemble de validation, le «*Grid search*» répétera inutilement deux expériences équivalentes, alors que le «*Random search*» donnera toujours deux explorations indépendantes des autres hyperparamètres.(Goodfellow *et al.*, 2016)

---

4. https ://www.coursera.org/lecture/deep-neural-network/hyperparameters-tuning-in-practice-pandas-vs-caviar-DHNcc

Le choix manuel des hyperparamètres tel que le *baby-sitting* nécessite de comprendre ce que font les hyperparamètres et comment les modèles d'apprentissage automatique parviennent à une bonne généralisation, tandis que les algorithmes de sélection automatique des hyperparamètres comme le «*Random search*» réduisent considérablement la nécessité de comprendre ces idées, mais ils sont souvent beaucoup plus coûteux en calcul.(Goodfellow *et al.*, 2016)

Afin de sélectionner la meilleure configuration pour nos modèles proposés, nous avons opté pour la stratégie de baby-sitting en réglant un paramètre à la fois selon notre expérience. L'ordre dont lequel nous allons régler nos hyperparamètres est important aussi. Selon Goodfellow *et al.* (2016) et Greff *et al.* (2017), le taux d'apprentissage est peut-être l'hyperparamètre le plus important à régler, car il contrôle la capacité effective du modèle lorsque la bonne valeur est choisie. Le réglage des hyperparamètres autres que le taux d'apprentissage nécessite de surveiller à la fois le taux d'erreur sur l'ensemble d'entraînement et de test afin de diagnostiquer si le modèle est en sur-apprentissage ou sous-apprentissage.
Ainsi, le nombre des unités cachées est le deuxième plus important hyperparamètre à régler selon Greff *et al.* (2017). En effet, augmenter la taille du réseau permet d'améliorer la capacité de représentation du modèle (Goodfellow *et al.*, 2016) et ainsi d'éviter le sous-apprentissage du modèle.
D'autre part, le choix d'un bon algorithme d'optimisation qui converge plus rapidement est important. La vitesse de la convergence de l'algorithme d'optimisation dépend aussi du choix de la taille de la mini-batch [5] [6]. Ce dernier a un impact sur la fréquence de réglage des paramètres du modèle. L'utilisation des petites

---

5. https //www.coursera.org/lecture/deep-neural-network/mini-batch-gradient-descent-qcogH

6. C'est la division de l'ensemble d'entraînement en petits ensembles qui sont utilisés pour calculer l'erreur du modèle et mettre à jour les paramètres du modèle

mini-batchs permet à l'algorithme d'optimisation de régler les paramètres plus fréquemment, plutôt que de les régler une fois en utilisant tout l'ensemble d'entraînement à chaque itération de l'algorithme d'optimisation.
Le taux de régularisation *dropout* (Srivastava *et al.*, 2014) doit être évalué à son tour. L'objectif est de réduire le taux d'erreur sur l'ensemble de validation tout en gardant le taux d'erreur de l'ensemble d'entraînement faible.
À la fin, le choix du plongement de mots pré-entraîné est très crucial pour notre tâche puisqu'il représente l'une des deux représentations de mots utilisées par nos modèles.
L'évaluation des hyperparamètres a été effectuée sur l'ensemble de développement et la moyenne/écart-type de la mesure-F a été utilisée comme métrique.

4.2.1 Le choix du taux d'apprentissage

Le taux d'apprentissage représente l'hyperparamètre le plus important à mettre au point dans un modèle basé sur l'apprentissage profond. Cet hyperparamètre est un scalaire strictement positif. Il permet de contrôler la vitesse avec laquelle le modèle va apprendre. En fait, lors de la rétropropagation de l'erreur afin de mettre à jour les poids du modèle, le taux d'apprentissage permet de contrôler cette mise à jour. Ainsi, nous voulons évaluer ce paramètre sur nos modèles afin de choisir une valeur qui favorise la convergence de nos modèles. Pour cela, nous avons choisi trois valeurs différentes : large, moyenne et petite {0.01, 0.001, 0.0001}. Étant donné que les résultats étaient similaires pour les trois modèles, nous avons rapporté seulement le résultat d'un seul modèle (Modèle 2). Les autres hyperparamètres sont indiqués dans le tableau 4.6.

Le graphique 4.1 illustre les trois courbes de la fonction de perte avec les différents taux d'apprentissage. Nous avons observé qu'une large valeur de ce paramètre

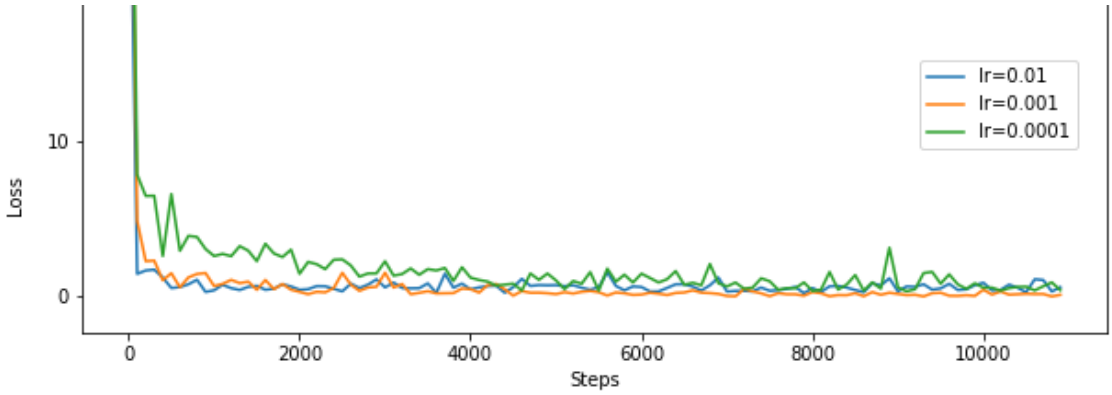


Figure 4.1: Taux d'apprentissage

(courbe en bleu) a convergé plus rapidement au début dans notre cas, tandis qu'un taux très petit (en vert) converge beaucoup plus lentement. Au final, un taux d'apprentissage moyen (lr=0.001 courbe orange) nous a donné le meilleur rendement global (Tableau 4.2). En effet, théoriquement un taux large va mettre à jour les poids avec une valeur plus large et pourra manquer la solution optimale. D'autre part, un taux très petit peut ne pas converger ou rester bloqué dans un minimum local. Mais cela dépend aussi de la taille de la mini-batch. Une petite mini-batch pourrait nécessiter un faible taux d'apprentissage pour maintenir la stabilité en raison de la forte variance dans l'estimation du gradient (Goodfellow *et al.*, 2016). De ce fait, nous concluons que le choix de cette valeur est empirique selon l'architecture et les données utilisées, une valeur moyenne ni très grande ni petite pourrait être un bon choix. Dans notre cas, nous avons choisi de fixer cette valeur à 10^{-2}.

Tableau 4.2: Choix du taux d'apprentissage

	0.01	0.001	0.0001
Modèle 2	92.43 ±0.19	**94.61 ±0.07**	92.22 ±0.10

4.2.2 Le choix de nombre d'unités cachées LSTM

Afin de diminuer le taux d'erreur sur l'ensemble d'entraînement, augmenter la taille du réseau pourrait être une solution qui permet d'améliorer la capacité de représentation des modèles (Goodfellow *et al.*, 2016). De ce fait, nous avons augmenté le nombre des unités cachées des LSTM pour permettre à nos modèles de bien s'adapter à l'ensemble d'entraînement. Le nombre d'unités cachées des LSTM pour chaque modèle sont représentés dans le tableau 4.6.

4.2.3 Le choix de la taille du mini-batch

Un autre hyperparamètre que nous avons choisi d'évaluer est la taille du mini-batch. Le choix de ce hyperparamètre est important dans les modèles d'apprentissage profond, car il impacte directement la vitesse de convergence de l'optimisation de la fonction objectif. De plus, il permet d'obtenir une meilleure stabilité d'entraînement et une meilleure performance de généralisation (Masters et Luschi, 2018). De plus, diviser l'ensemble d'entraînement en petits ensembles permet de ne pas avoir toutes les données d'entraînement en mémoire à la fois et régler les paramètres sans attendre le traitement de tout l'ensemble d'entraînement. Dans la littérature, le choix de ce hyperparamètre est souvent entre 1 à quelques centaines (Goodfellow *et al.*, 2016). Masters et Luschi (2018) ont confirmé que l'utilisation des mini-batchs de petites tailles donne de meilleurs résultats. De ce fait, nous avons évalué nos modèles avec différentes mini-batchs et nous avons rapporté les résultats obtenus sur l'ensemble de développement pour les valeurs (1, 8, 32, 64, 128). Les autres hyperparamètres sont présentés dans le tableau 4.6

Le tableau 4.3 illustre les résultats obtenus sur l'ensemble de développement. Nous avons observé que les résultats des mini-batch 8, 32 et 64 étaient proches et ont donné la meilleure performance pour les trois modèles, tandis que les mini-batchs

Tableau 4.3: Choix de la taille du batch

Batch	1	8	32	64	128
Modèle 1	90.9±0.33	94.54±0.09	**94.55±0.07**	94.51±0.09	93.09±0.17
Modèle 2	90.88±0.40	**94.61 ±0.07**	94.32±0.11	94.21±0.09	93.37±0.15
Modèle 3	90.21±0.29	94.22±0.13	**94.24±0.07**	93.93±0.15	93.17±0.21

1 et 128 étaient moins bons. En outre, l'entraînement avec une mini-batch égal à 1 était très lent puisque nous entraînons les modèles avec un exemple à la fois. De ce fait, nous avons fixé la taille de mini-batch à 32 pour les modèles 1 et 3, et une mini-batch de taille 8 pour le modèle 2.

4.2.4 Le choix de l'algorithme d'optimisation

Nous avons également évalué certains algorithmes d'optimisation afin d'étudier leur impact sur l'entraînement de nos modèles : Momentum (Sutskever *et al.*, 2013), RMSProp (Hinton *et al.*, 2012), Adam (Kingma et Ba, 2014), ProximalSGD - une variante de SGD (J. Kiefer et J. Wolfowitz, 1952) et ProximalAdagrad - une variante de Adagrad (Duchi JDUCHI et Singer, 2011). Les autres hyperparamètres restent inchangés comme indiqué dans le tableau 4.6.

Le graphique sur la figure 4.2 montre les différents algorithmes évalués sur le modèle 1. Nous avons observé la meilleure performance avec Adam, suivie par RMSProp avec une légère différence. De plus, ces deux algorithmes ont pu converger plus rapidement que les autres algorithmes. De ce fait, nous avons opté pour l'utilisation d'Adam comme algorithme d'optimisation.

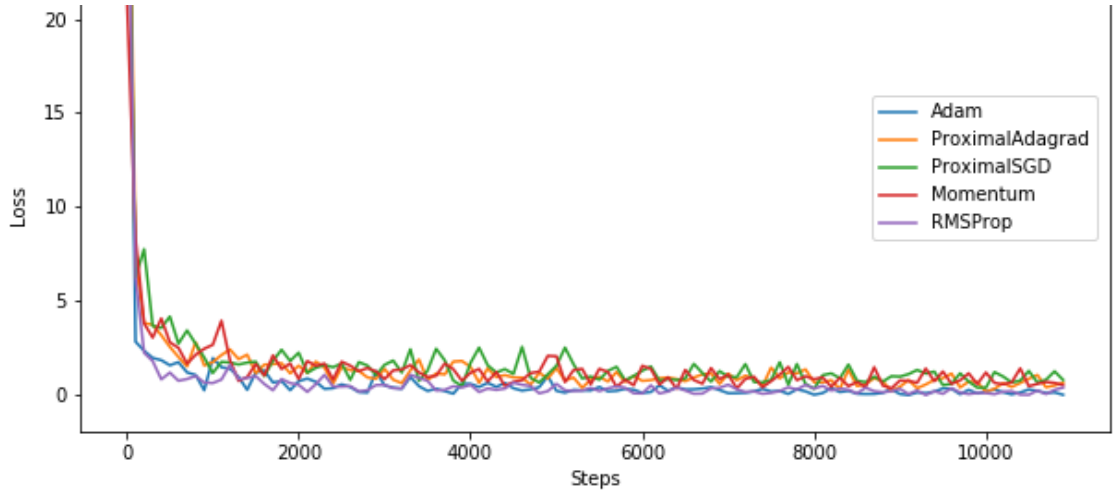


Figure 4.2: Les algorithmes d'optimisation

4.2.5 Le choix du taux de dropout

La méthode de régularisation «*dropout*» est une étape importante dans l'élaboration d'un modèle d'apprentissage profond afin d'éviter le sur-apprentissage. Cette méthode consiste à désactiver aléatoirement certains neurones (Srivastava *et al.*, 2014). Cependant, le choix du taux de dropout est à déterminer empiriquement. Pour cela, nous avons expérimenté différents taux de dropout {0.1, 0.25, 0.5, 0.75, 0.9} sur nos modèles. Nous avons appliqué un dropout naïf (Zaremba *et al.*, 2014) sur les entrées et les sorties de chaque LSTM dans les modèles. Les autres hyper-paramètres restent inchangés comme indiqué dans le tableau 4.6. La performance des modèles sur l'ensemble de développement sont représentés dans le tableau 4.4.

Comme nous pouvons l'observer dans le tableau 4.4, les meilleurs résultats ont été obtenus lors de la désactivation de 50% du réseau. Les valeurs inférieures ont causé un léger sur-apprentissage des modèles, où la mesure-F a diminué sur l'ensemble de développement en dépit du score élevé sur l'ensemble de l'entraînement (Figure 4.3). Peu désactiver les unités des modèles augmente leur capacité à mémoriser des

Tableau 4.4: Choix du taux de dropout

Dropout	0.1	0.25	0.5	0.75	0.9
Modèle 1	93.57±0.7	94.34±0.10	**94.55±0.07**	92.96±0.16	83.38±0.17
Modèle 2	93.62 ±0.18	94.00 ±0.25	**94.61 ±0.07**	93.32±0.20	86.94±0.12
Modèle 3	93.76±0.20	94.09±0.15	**94.24±0.07**	93.44±0.14	85.95±0.10

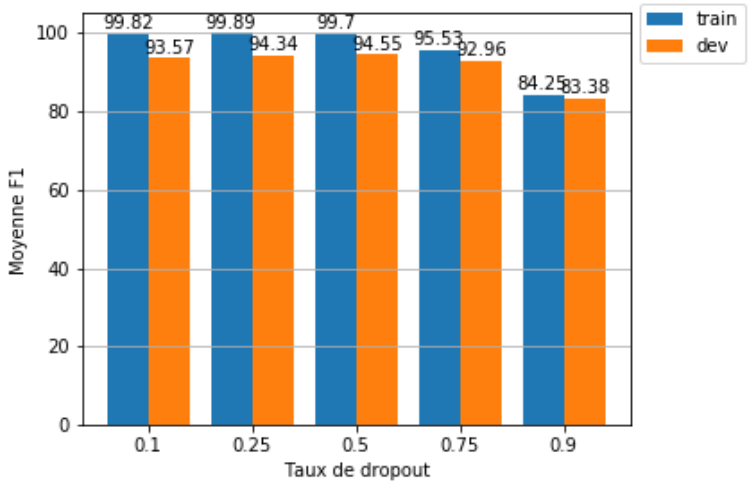


Figure 4.3: L'impact du choix de dropout sur l'ensemble d'entraînement et test (Modèle 1)

propriétés de l'ensemble d'entraînement qui ne les servent pas bien sur l'ensemble de test et causant par le fait même un sur-apprentissage. En revanche, les valeurs supérieures ont empêché les modèles de bien apprendre les données d'entraînement causant un sous-apprentissage, ce qui a mené à des résultats médiocres sur l'ensemble de l'entraînement et du développement, tel qu'illustré dans la figure 4.3. Selon Goodfellow *et al.* (2016), souvent les meilleures performances proviennent d'un modèle large bien régularisé. Toutefois, désactiver les unités moins souvent leur donne plus d'occasions de coopérer les unes avec les autres pour s'adapter à l'ensemble de l'entraînement et vice versa (Goodfellow *et al.*, 2016). De ce fait,

Tableau 4.5: L'impact du choix du plongement de mots sur nos modèles

	Glove 100	Glove 300	Word2vec 100	FastText 300
Modèle 1	89.79±0.28	**94.55±0.07**	89.82±0.33	94.18±0.17
Modèle 2	90.12±0.17	**94.61 ±0.07**	89.86±0.17	93.64±0.18
Modèle 3	87.95±0.35	**94.24±0.07**	87.52±0.21	93.57±0.16

nous avons décidé de maintenir un dropout de 50% pour nos modèles.

4.2.6 Le choix du plongement de mots

L'utilisation du plongement de mots pré-entraîné dans le domaine TALN est devenue essentielle, notamment en raison de sa capacité à généraliser et à représenter la sémantique des mots. De ce fait, nous nous sommes posés la question : le choix du plongement de mots pré-entraîné est-il vraiment nécessaire ? Quel serait son impact ?

Afin de répondre à ces questions, nous avons mis au point une série d'expériences avec différents plongements de mots pré-entraînés de différentes tailles : Word2vec (taille = 100) (Mikolov *et al.*, 2013b), Glove (taille = 100, 300) (Pennington *et al.*, 2014) et FastText (taille = 300) (Bojanowski *et al.*, 2017). Les autres hyperparamètres sont fixés comme présentés dans le tableau 4.6.

Le tableau 4.5 montre les performances obtenues avec nos trois modèles en utilisant les différents plongements de mots évalués. Nous pouvons observer que l'impact du plongement de mot est évident. Le Glove 300 a donné la meilleure performance sur les trois modèles, suivi par FastText avec une différence de l'ordre de 1%. Cela peut être dû à la taille du vocabulaire retrouvé dans les deux plongements. En revanche, les plongements avec une plus petite dimension (Word2vec, Glove 100)

ont obtenu des résultats moins bons ($\simeq -5\%$).

La performance de FastText est meilleure que celle de Word2vec, ce qui correspond à nos attentes, puisque la taille du plongement FastText est plus grande que celle de word2vec, ce qui lui permet de contenir plus d'informations discriminantes.

Nous avons également observé que les modèles 1 et 2 sont un peu plus robustes au changement de plongement de mots par rapport au troisième modèle ($\simeq 2\%$). Cela est dû à notre somme pondérée qui permet au modèle de considérer la représentation de mots à partir de caractères.

Nous avons conclu que le choix du plongement de mots pré-entraîné est crucial pour notre tâche. Il est primordial de considérer la nature des données sur lesquelles le plongement a été entrainé ainsi que l'algorithme utilisé. De ce fait, nous avons choisi Glove 300 comme plongement de mots pré-entraîné pour nos modèles.

4.2.7 Récapulatif des hyperparamètres

Le tableau 4.6 présente les différents hyperparamètres utilisés lors de nos expériences.

4.3 Résultats

4.3.1 Résultats généraux

Nous avons récapitulé les résultats de nos expériences dans le tableau 4.7. Ces résultats ont été obtenus sur le jeu de données CoNLL-2003 pour la langue anglaise dédiée à la tâche de reconnaissance des entités nommées. À titre de comparaison, nous avons rapporté les résultats des autres approches qui existent dans la littérature ainsi que nos deux modèles de base. Comme nous l'avons déjà mentionné,

Tableau 4.6: Tableau récapitulatif des hyperparamètres : $LSTM^{(RMC)}$ LSTM pour la représentation de mots à partir de caractères, $LSTM^{(CPM)}$ LSTM pour le contexte du plongement de mots pré-entraîné, $LSTM^{(CF)}$ LSTM pour l'encodage du contexte finale.

Paramètre	Base1	Base2	Modèle1	Modèle2	Modèle3
Taille du plongement de caractères	100	100	100	100	100
Taille état caché $LSTM^{(RMC)}$	150	NA	150	200	100
Taille état caché $LSTM^{(CPM)}$	NA	NA	150	NA	NA
Plongement de mots (Glove)	300	300	300	300	300
Taille état caché $LSTM^{(CF)}$	200	250	200	200	200
L'optimiseur	Adam	Adam	Adam	Adam	Adam
Taille mini-batch	32	32	32	8	32
Taux d'apprentissage	0.001	0.001	0.001	0.001	0.001
Dropout	0.5	0.5	0.5	0.5	0.5
Epoch	25	25	25	25	25
CNN Nombre de kernel	NA	50	NA	NA	50
CNN taille de la fenêtre	NA	3	NA	NA	3

Tableau 4.7: Résultats REN obtenus sur l'ensemble test de jeu de données CoNLL-2003 pour la langue anglaise. † : utilise des données externes, ‡ : utilise un modèle de langue pré-entraîné à base de caractères, ∗ : utilise un modèle d'attention, - : indique la non présence du résultat.

Modèle	Max F_1%	Moyenne ±std F_1%
Huang *et al.* (2015)	90.10	-
Lample *et al.* (2016)	90.94	-
Ma et Hovy (2016)	91.21	-
Rei *et al.* (2016)∗	84.09	-
Chiu et Nichols (2015)†	91.61	-
Peters *et al.* (2018) ELMO‡	-	92.22±0.10
Devlin *et al.* (2018) BERT‡∗	**92.8**	-
Akbik *et al.* (2018) CSE‡	-	**93.09 ±0.12**
Modèle Base1 (Lample et al.)	91.27	91.04±0.18
Modèle Base2 (Ma et al.)	91.23	91.10±0.10
Modèle 1	91.41	91.22±0.10
Modèle 2	**91.57**	**91.39±0.11**
Modèle 3	91.31	91.20±0.10

nous avons rapporté le maximum, la moyenne et l'écart type de la mesure-F de cinq exécutions sur l'ensemble des tests.

Selon les résultats obtenus, nous observons en premier lieu que la réimplémentation des modèles de Lample *et al.* (2016) et Ma et Hovy (2016) (Modèles de base 1 et 2) sont meilleurs que les modèles originaux, cela est dû probablement à l'utilisation d'un plongement de mots plus large et meilleur. De plus, on constate que le modèle de Ma et Hovy (2016) est légèrement meilleur que celui de Lample *et al.* (2016).

Cela pourrait signifier que les CNN sont plus efficaces par rapport aux LSTM en ce qui concerne la représentation de mots à partir de caractères. Ce constat confirme l'observation de Yang *et al.* (2018).

Dans un second temps, nous remarquons que nos trois modèles ont un résultat plus stable, où le modèle 2 nous a donné le meilleur rendement global en matière de moyenne/écart type et performance maximale de la mesure-F par rapport aux modèles 1 et 3.

La performance du modèle 1 basé sur la somme pondérée pour la combinaison des deux types de représentations est de loin meilleure que celle du modèle proposé par Rei *et al.* (2016) (+7.48%). Cela peut s'expliquer par notre utilisation d'une couche supplémentaire pour l'extraction de contextes de plongements de mots pré-entraîné et notre utilisation de Glove 300 au lieu de Word2vec 100 utilisé par Rei *et al.* (2016).

La combinaison entre LSTM et CNN que nous avons proposée dans le modèle 3 a surpassé leurs utilisations séparément dans les modèles de base 1 et 2.

Bien que Chiu et Nichols (2015) ont fait intervenir des données externes et de l'ingénierie manuelle des caractéristiques, la performance de nos modèles est très proche de la leur. Par contre, leur modèle se dégrade lorsque les lexiques ne sont pas utilisés (mesure-F 90.91%).

Malgré les résultats notables obtenus, des modèles plus sophistiqués basés sur le transfert de connaissances telles que BERT, ELMo et CSE qui sont entraînés sur des corpus gigantesques avec une grande capacité de calcul se montrent supérieurs par rapport aux approches classiques telle que les nôtres.

4.3.2 Analyse d'erreurs

Le tableau 4.8 résume les performances (mesure-F %) de nos modèles par type d'entité. Manifestement, le modèle 2 s'est montré performant sur tous les types, en particulier sur les entités «LOC» et «MISC». D'autre part, les modèles 1 et 3 ont été meilleurs pour la détection des types «ORG» et «PER» respectivement. Pratiquement tous les modèles ont fait moins d'erreurs pour la détection des entités «LOC» et «PER», tandis que les entités «MISC» et «ORG» étaient plus difficiles à cerner et souvent confondues par les modèles.

Tableau 4.8: Résultats par entité (mesure-F%)

	Base1	Base2	Modèle1	Modèle2	Modèle3
LOC	92.70	92.97	92.91	**93.58**	92.75
MISC	81.16	80.09	80.87	**81.55**	80.66
ORG	88.79	89.09	**89.54**	89.23	89.42
PER	95.82	96.26	96.39	96.24	**96.48**

Dans le tableau 4.9, nous exhibons quelques exemples de détection des entités nommées par nos modèles.

Dans le premier exemple, le mot «Tychy» a été détecté par tous les modèles en tant qu'entité, mais, la plupart des modèles ont confondu entre la ville polonaise «Tychy» et l'organisation «Tychy» à l'exception du modèle 2 qui a pu prendre en considération le contexte autour pour classifier ce mot en tant qu'organisation (B-ORG).

Le modèle 1 a souvent réussi à classifier les mots OOV (*Out-Of-Vocabulary*) grâce à notre somme pondérée qui permet au modèle de favoriser la représentation de mots à partir de caractères dans ce cas-là. Par exemple, le mot «mid-Mississippi»

a été bien classifié par le modèle 1 malgré que le mot soit un mot OOBV (*Out-Of-Both-Vocabulary*) [7].

Dans le troisième exemple, l'organisation «*Sale Grammar School*» a été détectée par le modèle «base 1» (Lample *et al.*, 2016), basé sur un LSTM mais avec le mauvais fenêtrage. En revanche, le modèle «Base2» (Ma et Hovy, 2016) basé sur un CNN a réussi à détecter le bon fenêtrage mais avec le mauvais type d'entité. Le modèle 3 qui est une combinaison des deux architectures a été capable de bien extraire l'entité avec le bon fenêtrage.

Certaines erreurs d'étiquetage dans l'ensemble des tests ont été détectées par les modèles. Par exemple, le mot «*AUSTRALIAN*» dans le tableau 4.9 devrait être étiqueté avec l'entité «MISC» comme toutes les nationalités dans l'ensemble d'entraînement, alors que dans l'ensemble de test il a été annoté avec l'étiquette «O» ; cela a induit les modèles en erreur, car les modèles ont été entraînés pour étiqueter les nationalités avec «MISC». Par exemple, dans la phrase «Thais hunt for Australian jail breaker .» tirée depuis l'ensemble de l'entraînement, le mot «Australian» a été étiqueté avec l'étiquette «B-MISC».

Le lieu «*Santiago Bernabeu stadium*» a été détecté par tous les modèles, mais pas avec le bon fenêtrage à cause du mot «*stadium*» qui devrait commencer par une majuscule puisque cette caractéristique est déterminante pour identifier une entité. Nous avons réévalué cette phrase en corrigeant cette erreur et les modèles ont réussi cette fois à étiqueter correctement le mot avec l'étiquette «I-LOC».

Certaines erreurs peuvent être attribuées aux erreurs d'annotation ou à l'ambi-

---

7. Un mot OOBV est un mot qui ne possède pas un plongement de mot pré-entraîné OOEV (*Out-of-Embedding-Vocabulary*) et, de plus, il ne figure pas dans l'ensemble de l'entraînement OOTV «*Out-Of-Training-Vocabulary*».

guïté de la langue.

Tableau 4.9: Exemples de données Test

	Vérité	Base1	Base2	Modèle1	Modèle2	Modèle3
Earlier this year South African Breweries Ltd (SAB) bought strategic stakes in the unlisted Lech and **Tychy** brewers , which together hold more than 20 percent of the market , and Australia 's Brewpole BV has a controlling stake in Poland 's largest brewery , Elbrewery Company Ltd. (EB) .	**B-ORG**	B-LOC	B-LOC	B-LOC	**B-ORG**	B-LOC
- Five barges , 30-day open , **mid-Mississippi** (McGregor and south) bid at 160 percent , offered at 170 percent , no comparisons .	**B-MISC**	O	O	**B-MISC**	O	O
The 16-year-old who attended **Sale Grammar School** in the northern England city of Manchester died less than a day after becoming ill .	**B-ORG** **I-ORG** **I-ORG**	B-ORG I-ORG O	B-LOC I-LOC I-LOC	B-MISC I-MISC O	B-LOC I-LOC O	**B-ORG** **I-ORG** **I-ORG**
BOXING - SCHULZ DEFEATS RIBALTA IN **IBF** HEAVYWEIGHT FIGHT .	**B-ORG**	O	B-MISC	**B-ORG**	**B-ORG**	**B-ORG**
CRICKET - LARA SUFFERS MORE **AUSTRALIAN** TOUR MISERY .	**O**	B-MISC	B-MISC	B-MISC	B-MISC	B-MISC
Spanish police will breathalyse fans at the gates of the **Santiago Bernabeu stadium** and ban drunk supporters from Saturday 's big Real Madrid-Barcelona game , the Madrid daily El Mundo said on Friday .	**B-LOC** **I-LOC** **I-LOC**	**B-LOC** **I-LOC** O	**B-LOC** **I-LOC** O	**B-LOC** **I-LOC** O	**B-LOC** **I-LOC** O	**B-LOC** **I-LOC** O

4.4 Conclusion

Dans ce chapitre, nous avons évalué les performances de nos modèles d'extraction des entités nommées. Les résultats que nous avons obtenus révèlent que nos modèles sont stables et permettent une détection automatique dans un processus de bout en bout sans données externes ni intervention humaine, ce qui nous permet de supposer que les modèles pourraient être capables de fonctionner parfaitement dans différents domaines ; cela peut être confirmé en effectuant plus de tests sur d'autres jeux de données. De plus, nous avons confirmé que le choix des hyperparamètres est crucial et qu'il a un impact direct sur les performances d'un système

de reconnaissance d'entités nommées basé sur l'apprentissage profond.

CONCLUSION

La détection des entités nommées est une sous-tâche cruciale dans les applications de traitement automatique du langage naturel modernes. Nombreuses sont les tentatives menées afin d'apporter des réponses à cette problématique. Récemment encore, l'emploi de l'apprentissage profond dans cette thématique est devenue indéniable. Par delà son efficacité, il a permis à la communauté de recherche d'atteindre facilement des résultats probants.

Dans le cadre de notre travail, nous avons effectué une étude sur les différentes approches utilisées dans ce domaine, notamment celles basées sur l'apprentissage profond principalement les réseaux de neurones récurrents. Nous avons également proposé trois différents modèles qui permettent la reconnaissance des entités nommées. Nos deux premiers modèles sont basés sur un mécanisme de somme pondérée, tandis que le troisième fait intervenir une combinaison de deux techniques différentes pour la représentation de mots à partir de caractères. Nos trois modèles sont autonomes, de bout en bout et ne font intervenir ni des ressources externes ni l'ingénierie manuelle des caractéristiques.

À des fins de comparaison, nous avons mené une étude empirique sur le corpus CoNLL-2003 (Sang et De Meulder, 2003) dans le but d'évaluer la performance de nos modèles et se comparer aux résultats de l'état de l'art. Afin de mettre en évidence la valeur ajoutée de notre contribution, nous avons retenu deux modèles de base. Nous avons aussi investigué l'impact du choix des différents hyperparamètres sur ces modèles. Au cours de cette étude, nous avons observé que certains hyperparamètres impactent négativement la performance de nos modèles plus que

d'autres, tels que le plongement de mots pré-entraîné. Néanmoins, nous avons évalué ces hyperparamètres dans une seule direction en changeant un paramètre à la fois en commençant du plus important et ainsi de suite. Il est possible qu'une recherche aléatoire donne de meilleurs résultats. Notre meilleur résultat a été obtenu via le modèle 2 avec une moyenne/écart type de la mesure-F de $91.39\% \pm 0.11$. La réimplémentation des modèles de Lample *et al.* (2016) et Ma et Hovy (2016) a donné $91.04\% \pm 0.18$ et $91.10\% \pm 0.10$ respectivement, alors que les modèles originaux de Lample *et al.* (2016) et Ma et Hovy (2016) ont donné une mesure-F maximale de 90.90 et 91.21 respectivement.

En matière de perspective, il serait intéressant d'explorer différentes pistes dans la suite de ce travail. La première est d'étendre le champs d'action de nos modèles afin de détecter des entités à granularité plus fine dans des domaines spécifiques différents telle que la bio-informatique. De plus, nous pouvons adapter nos modèles pour inclure d'autres tâches, à l'instar de l'étiquetage morpho-syntaxique (POS)(Sang et De Meulder, 2003), l'analyse syntaxique de surface (Tjong *et al.*, 2000), Coréférence (Hobbs, 1979), puisque ces derniers se réduisent à une tâche de marquage de séquence comme la nôtre. D'autre part, utiliser des ressources externes telles que les dictionnaires pourrait améliorer la performance des modèles. Une autre perspective très intéressante qui est désormais possible dans ce domaine est le transfert de connaissances. Cette technique est en vogue récemment (BERT (Devlin *et al.*, 2018), ELMO (Peters *et al.*, 2018), CSE (Akbik *et al.*, 2018)) et nous laisserait penser qu'il y aurait encore énormément de pistes à explorer.

Finalement, nous souhaitons qu'à travers ce modeste travail, nous avons pu recueillir les principales approches utilisées dans la reconnaissance des entités nommées en particulier celles basées sur l'apprentissage profond avec les réseau de neurones récurrents. En outre, nous espérons que les modèles proposés inspirent d'autres recherches.

RÉFÉRENCES

Akbik, A., Blythe, D. et Vollgraf, R. (2018). *Contextual String Embeddings for Sequence Labeling*. Rapport technique

Alfonseca, E. et Manandhar, S. (2002). *An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery*. Rapport technique.

Bahdanau, D., Cho, K. et Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. Récupéré de `http://arxiv.org/abs/1409.0473`

Balabantaray, R. C. (2010). *Name Entity Recognition in Machine Translation*. Rapport technique 3.

Baum, L. E. et Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Ann. Math. Statist.*, *37*(6), 1554–1563. `http://dx.doi.org/10.1214/aoms/1177699147`. Récupéré de `https://doi.org/10.1214/aoms/1177699147`

Bellman, R. (1957). *Dynamic Programming*. Numéro 0-486-42809-5. Mineola, New-York : Dover Publication.

Bengio, Y., Simard, P. et Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, *5*(2).

Bojanowski, P., Grave, E., Joulin, A. et Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. http://dx.doi.org/10.1162/tacl{_}a{_}00051

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P. et Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. Dans *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2635–2639. International Speech and Communication Association.

Chiu, J. P. C. et Nichols, E. (2015). Named Entity Recognition with Bidirectional LSTM-CNNs.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. et Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. Rapport technique.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. et Kuksa, P. (2011a). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. et Kuksa, P. (2011b). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537.

Corro, L. D., Abujabal, A., Gemulla, R. et Weikum, G. (2015). *FINET Context-Aware Fine-Grained Named Entity Typing*. Rapport technique.

Demartini, G., Iofciu, T. et De Vries, A. P. (2010). Overview of the INEX 2009 Entity Ranking Track. Dans *Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for the Evaluation of XML*

Retrieval, INEX'09, 254–264., Berlin, Heidelberg. Springer-Verlag. Récupéré de http://dl.acm.org/citation.cfm?id=1881065.1881096

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2018). BERT Pre-training of Deep Bidirectional Transformers for Language Understanding. Récupéré de http://arxiv.org/abs/1810.04805

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S. et Weischedel, R. (2004). The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. Dans *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA). Récupéré de http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf

Duchi JDUCHI, J. et Singer, Y. (2011). *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization * Elad Hazan*. Rapport technique.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268–278. http://dx.doi.org/10.1109/PROC.1973.9030

G. Petasis, A. Cucchiarelli, P. Velardi, G. Paliouras, V. Karkaletsis et C. D. Spyropoulos (2000). Automatic adaptation of proper noun dictionaries through cooperation of machine learning and prob- abilistic methods. *SIGIR*, p. pp. 128–135.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep Learning*. The MIT Press.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R. et Schmidhuber, J. (2017). LSTM A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. http://dx.doi.org/10.1109/TNNLS.2016.2582924

Grishman, R. (1996). *Message Understanding Conference-6 : A Brief History*. Rapport technique.

He, K., Zhang, X., Ren, S. et Sun, J. (2015). Deep Residual Learning for Image Recognition. Récupéré de `http://arxiv.org/abs/1512.03385`

Hinton, G., Srivastava, N. et Swersky, K. (2012). *Neural Networks for Machine Learning Lecture 6a Overview of mini--batch gradient descent*. Rapport technique.

Hobbs, J. R. (1979). Coherence and Coreference*. *Cognitive Science*, *3*(1), 67–90. `http://dx.doi.org/10.1207/s15516709cog0301{\_}4`. Récupéré de `https://doi.org/10.1207/s15516709cog0301_4`

Hochreiter, J. (1991). *DIPLOMARBEIT IM FACH INFORMATIK Untersuchungen zu dynamischen neuronalen Netzen*. Rapport technique.

Hochreiter, S. et Schmidhuber, U. J. (1997). LONG SHORT-TERM MEMORY. *Neural Computation*, *9*(8), 1735–1780. Récupéré de `http://www7.informatik.tu-muenchen.de/~hochreithttp://www7.informatik.tu-muenchen.de/~hochreit`

Hoffman, M. D., Blei, D. M. et Bach, F. (2010). *Online Learning for Latent Dirichlet Allocation*. Rapport technique.

Huang, Z., Xu, W. et Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging.

Jansson, P. et Liu, S. (2017). *Distributed Representation, LDA Topic Modelling and Deep Learning for Emerging Named Entity Recognition from Social Media*. Rapport technique

Kingma, D. P. et Ba, J. L. (2014). ADAM A METHOD FOR STOCHASTIC OPTIMIZATION.

Komninos, A. et Manandhar, S. (2016). *Dependency Based Embeddings for Sentence Classification Tasks*. Rapport technique.

Kripke, S. (1982). *Naming and Necessity*. Rapport technique, Boston : Harvard University Press.

Lafferty, J., Mccallum, A., Pereira, F. C. N. et Pereira, F. (2001). Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. 282–289. Récupéré de `http://repository.upenn.edu/cis_papershttp://repository.upenn.edu/cis_papershttp://repository.upenn.edu/cis_papers`

Lal, A., Tomer, A. et Chowdary, C. R. (2017). SANE. Dans *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, 227–230., New York, New York, USA. ACM Press. `http://dx.doi.org/10.1145/3041021.3054724`. Récupéré de `http://dl.acm.org/citation.cfm?doid=3041021.3054724`

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. et Dyer, C. (2016). Neural Architectures for Named Entity Recognition. `http://dx.doi.org/10.18653/v1/N16-1030`

LeCun, Y., Bengio, Y. et Hinton, G. (2015). *Deep learning* (vol.521, no. 7553 éd.).

LeCun, Y., Bernhard, B., John, S. D., Donnie, H., Richard, E. H., Wayne, H. et Lawrence, D. J. (1989). Backpropagation applied to handwrit- ten zip code recognition. *Neural computation*.

Levy, O. et Goldberg, Y. (2014). *Dependency-Based Word Embeddings*. Rapport technique.

Lin, D. et Wu, X. (2009). *Phrase Clustering for Discriminative Learning*. Rapport technique.

Ling, W., Luís, T., Marujo, L., Fernandez, R., Amir, A. S., Dyer, C., Black, A. W. et Trancoso, I. (2015). *Finding Function in Form : Compositional Character Models for Open Vocabulary Word Representation*. Rapport technique.

Ling, X. et Weld, D. S. (2012). *Fine-Grained Entity Recognition*. Rapport technique

Liu, L., Shang, J., Xu, F. F., Ren, X., Gui, H., Peng, J. et Han, J. (2017). Empower Sequence Labeling with Task-Aware Neural Language Model. Récupéré de http://arxiv.org/abs/1709.04109

Liu, X. et Zhou, M. (2013). Two-stage NER for tweets with clustering. *Information Processing & Management*, *49*(1), 264–273. http://dx.doi.org/https://doi.org/10.1016/j.ipm.2012.05.006. Récupéré de http://www.sciencedirect.com/science/article/pii/S0306457312000714

Ma, X. et Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. http://dx.doi.org/10.18653/v1/P16-1101

Masters, D. et Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. Récupéré de http://arxiv.org/abs/1804.07612

McCallum, A. et Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. Dans *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* -. http://dx.doi.org/10.3115/1119176.1119206

Meng, Y., Li, X., Sun, Z. et Li, J. (2019). Query-Based Named Entity Recognition. Récupéré de http://arxiv.org/abs/1908.09138

Mikolov, T., Chen, K., Corrado, G. et Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. Récupéré de https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. et Dean, J. (2013b). Distributed representations ofwords and phrases and their compositionality. Dans *Advances in Neural Information Processing Systems*. Neural information processing systems foundation.

Miller, G. A. (1995). WordNet A Lexical Database for English. *Commun. ACM*, *38*(11), 39–41. http://dx.doi.org/10.1145/219717.219748. Récupéré de https://doi.org/10.1145/219717.219748

Nadeau, D. et Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), pp. 3–26.

Olah Christopher (2015). Understanding LSTM Networks. Récupéré de https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pennington, J., Socher, R. et Manning, C. D. (2014). Global vectors for word representation (Glove). Dans *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1532–1543. Association for Computational Linguistics (ACL).

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. et Zettlemoyer, L. (2018). Deep Contextualized Word Representations. 2227–2237. Association for Computational Linguistics (ACL). http://dx.doi.org/10.18653/v1/n18-1202

Peters, M. E., Ammar, W., Bhagavatula, C. et Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. Dans *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings*

of the Conference (Long Papers), volume 1, 1756–1765. Association for Computational Linguistics (ACL). http://dx.doi.org/10.18653/v1/P17-1161

Rei, M., Crichton, G. K. O. et Pyysalo, S. (2016). *Attending to Characters in Neural Sequence Labeling Models*. Rapport technique.

Reimers, N. et Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. Récupéré de http://arxiv.org/abs/1707.06799

Salton, G. et McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. USA : McGraw-Hill, Inc.

Sang, E. F. T. K. et De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task : Language-Independent Named Entity Recognition. http://dx.doi.org/10.3115/1119176.1119195

Sekine, S. et Nobata, C. (2004). *Definition, dictionaries and tagger for Extended Named Entity Hierarchy*. Rapport technique

Srivastava, N., Hinton, G., Krizhevsky, A. et Salakhutdinov, R. (2014). *Dropout A Simple Way to Prevent Neural Networks from Overfitting*. Rapport technique.

Strubell, E., Verga, P., Belanger, D. et Mccallum, A. (2017). *Fast and Accurate Entity Recognition with Iterated Dilated Convolutions*. Rapport technique

Sutskever, I., Martens, J., Dahl, G. et Hinton, G. (2013). *On the importance of initialization and momentum in deep learning*. Rapport technique.

Tjong, E. F., Sang, K. et Buchholz, S. (2000). *Introduction to the CoNLL-2000 Shared Task : Chunking*. Rapport technique

Turian, J., Ratinov, L. et Bengio, Y. (2010). *Word representations : A simple and general method for semi-supervised learning*. Rapport technique

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017). Attention is all you need. Dans *Advances in Neural Information Processing Systems*, volume 2017-December, 5999–6009. Neural information processing systems foundation.

wildml (2015). wildml. Récupéré de `http://www.wildml.com/wp-content/uploads/2015/11/`

Yadav, V. et Bethard, S. (2018). *A Survey on Recent Advances in Named Entity Recognition from Deep Learning models*. Rapport technique

Yang, J., Liang, S. et Zhang, Y. (2018). *Design Challenges and Misconceptions in Neural Sequence Labeling*. Rapport technique

Yang, Z., Salakhutdinov, R. et Cohen, W. (2016). Multi-Task Cross-Lingual Sequence Tagging from Scratch. Récupéré de `http://arxiv.org/abs/1603.06270`

Yao, L., Liu, H., Liu, Y., Li, X. et Anwar, M. W. (2015). Biomedical Named Entity Recognition based on Deep Neutral Network. *International Journal of Hybrid Information Technology*, *8*(8), 279–288. `http://dx.doi.org/10.14257/ijhit.2015.8.8.29`

Zaremba, W., Sutskever, I. et Vinyals, O. (2014). Recurrent Neural Network Regularization. Récupéré de `http://arxiv.org/abs/1409.2329`

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. et Fidler, S. (2015). Aligning Books and Movies : Towards Story-like Visual Explanations by Watching Movies and Reading Books. Récupéré de `http://arxiv.org/abs/1506.06724`