Une approche de découverte des diagrammes de classes UML par l'apprentissage profond

par Yves Rigou

Mémoire présenté dans le cadre du programme de maitrise en informatique de l'UQAC offert par extension à l'UQAR en vue de l'obtention du grade de maître ès science (M. Sc.)

Québec, Canada

Composition du jury :

Hamid Mcheick, président du jury, Université du Québec à Chicoutimi

Ismaïl Khriss, directeur de recherche, Université du Québec à Rimouski

Hafedh Mili, examinateur externe, Université du Québec à Montréal

Dépôt initial le 18 novembre 2021 Dépôt final le 28 avril 2022

RÉSUMÉ

L'ingénierie logicielle a élaboré des outils afin de rationaliser le processus de développement d'une application informatique. Parmi ceux-ci, l'architecture dirigée par les modèles propose de passer des exigences écrites en langage naturel dans un cahier des charges au code de l'application en passant par deux modèles intermédiaires, un modèle indépendant de la plateforme puis un modèle dépendant de la plateforme. Puisque les modèles et le code sont écrits en langage semi-formel, il est possible de passer de l'un à l'autre de manière automatique, ce qui rend le processus automatisable sur les trois quarts de sa longueur. Reste la première étape qui produit un modèle indépendant de plateforme à partir d'un cahier des charges et qui n'est pas automatisable en l'état actuel du fait de la complexité du langage naturel. Nous faisons l'hypothèse que l'utilisation de techniques de l'apprentissage profond développées dans le cadre du traitement automatique des langues nous permette d'envisager l'automatisation de cette étape et de produire un diagramme de classes qui est un modèle indépendant de plateforme courant. Le traitement automatique des langues est constitué d'un ensemble de tâches dont certaines peuvent nous aider à valider notre hypothèse. Parmi celles-ci, nous pensons que la détection d'entités peut nous aider à identifier les concepts clés du diagramme de classes, que la classification de relations peut nous aider à identifier les liens entre les concepts et que la résolution de coréférences peut nous aider à rassembler l'ensemble des informations relatives à un concept. Nous proposons un modèle qui résout conjointement les trois tâches et dont les résultats doivent nous permettre d'élaborer le diagramme. Nous proposons également un jeu de données annoté pour résoudre ces tâches dans le contexte de la transformation de cahiers des charges en diagramme de classe, ce qui n'existait pas encore. Notre modèle composé d'une couche d'encodage de type BERT et trois couches de décodage de type réseau à propagation avant permet de produire un diagramme de classe simple certes imparfait, mais cohérent malgré un jeu de données encore petit, ce qui valide la démarche et présente une première série de résultats encourageante.

Mots-clés : Ingénierie logicielle, Cahier des charges, Diagramme de classes, Apprentissage profond, Réseaux de neurones, Modèles, Détection d'entités, Résolution de coréférences, Classification de relations.

ABSTRACT

Software engineering has developed tools to streamline the process of developing an IT application. Among these, the model-driven architecture proposes going from requirements written in natural language in a specification to the application code via two intermediate models, a platform-independent model and then a platform-specific model. Since the models and code are written in a semi-formal language, it is possible to switch between them automatically, making the process automatable over three-quarters of its length. The first step which creates a platform-independent model based on specifications cannot be automated in the current state because of the complexity of natural language. We hypothesize that the use of deep learning techniques developed within the framework of automatic language processing allows us to consider the automation of this step and to produce a class diagram. Automatic language processing consists of tasks, some of which can help us validate our hypothesis. We believe that entity detection can help us identify key concepts in the class diagram, that relation classification can help us identify links between concepts, and that coreference resolution can help us gather all the information relating to a concept. We propose a model that jointly resolves the three tasks, which should allow us to construct the diagram. We also provide an annotated dataset to solve these tasks in the context of transforming specifications into class diagrams. Our model, composed of a BERT-type encoding layer and three feed-forward-neural-network-type decoding layers makes it possible to produce a simple class diagram that is imperfect but consistent despite a still small data set, which validates the approach and presents a first series of encouraging results.

Keywords: Software engineering, Specifications, Class diagram, Deep learning, Neural network, Model, Entity extraction, Coreference resolution, Relation classification

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

REMERCIEMENTS

Je remercie le professeur Ismaïl Khriss pour l'opportunité qu'il m'a donnée de travailler sur ce projet et pour ses conseils, son encadrement et ses encouragements tout au long du processus.

Je remercie Dany Lamontagne pour l'aide précieuse qu'il m'a apportée à différentes étapes du projet, particulièrement pour les discussions relatives aux réseaux de neurones et la configuration du système final. Je remercie également Ayman Chafni pour son aide et sa contribution à l'annotation du jeu de données.

Finalement, je remercie les membres du jury pour leurs commentaires sur le document lui-même, pour les pistes de réflexion qu'ils ont engagées sur des sujets connexes et pour la générosité de leurs encouragements.

INTRODUCTION

La mise en œuvre d'une solution logicielle pour combler des besoins d'affaires est un processus complexe. Pour l'appréhender, l'ingénierie logicielle a proposé une variété de stratégies destinées à rationaliser le processus et ainsi gagner en efficacité, en maintenabilité et en pérennité des solutions. L'une des stratégies les plus communément adoptées est l'ingénierie dirigée par le modèle (Model Driven Engineering MDE). Avec cette approche, des exigences rédigées dans un cahier des charges sont représentées dans des modèles qui appréhendent différents aspects de la future solution. Ces modèles sont ensuite convertis en code par un ensemble de transformations automatisables ou semi-automatisables. La stratégie est donc basée sur deux axes centraux que sont l'utilisation de modèles et le recours aux transformations de manière à passer progressivement des exigences au code. Les modèles présentent l'avantage d'être un outil de représentation schématique pouvant servir 1° de langage commun entre différents acteurs qui ne travaillent pas dans le même contexte (typiquement les praticiens du domaine d'affaires et les développeurs informatiques), 2° d'élément structurant des différents composants du domaine et des liens qui existent entre eux et 3° de base architecturale de la conception de la solution (Evans 2003). Les transformations, quant à elles, présentent l'avantage de standardiser, rationaliser et possiblement automatiser une partie du processus (Chénard 2013).

Une approche pratique à la mise en œuvre de l'ingénierie dirigée par les modèles est l'architecture dirigée par les modèles (Model Driven Architecture MDA) proposée par l'Object Management Group (OMG – Kleppe et al. 2003). La MDA propose un ensemble de modèles pour passer du cahier des charges au code dont un modèle indépendant de plateforme (Plateforme Independent Model PIM) et un modèle dépendant de la plateforme (Plateforme Specific Model PSM). Schématiquement, le domaine d'affaires de la solution est conceptualisé dans un premier modèle comme, par exemple, le PIM qui ne tient compte d'aucune exigence d'implémentation de la solution. Ce modèle se concentre uniquement sur la représentation de la logique d'affaires ; à ce titre il peut servir d'interface entre praticiens et développeurs. Il subit ensuite un jeu de transformations pour produire un deuxième modèle comme, par exemple, le PSM qui est adapté à l'environnement informatique dans lequel la solution sera produite. Finalement celui-ci subit à son tour un ensemble de transformations destinées à produire tout ou une partie du code de la solution finale.

À partir du moment où un modèle tel que le PIM est établi, le processus n'implique que des langages formels ou semi-formels; il est donc automatisable. Reste la première étape qui est la seule à faire intervenir le langage naturel et qui, en l'état actuel, n'est pas automatisable parce qu'aucun système ne peut traiter le langage naturel de manière suffisamment fiable pour produire un modèle de façon autonome. Cette étape est donc encore du ressort de l'humain.

Des tentatives d'automatisation existent néanmoins. Le principe général est de construire un ensemble de règles morphosyntaxiques destiné à couvrir le plus de critères

possible permettant d'identifier les éléments modélisables à partir d'un cahier des charges. Ces règles prennent une forme générale du type « si … alors ». Par exemple « si le document comporte l'expression 'contient', 'est fait de' ou 'inclut', alors les éléments de part et d'autre sont liés par une relation d'agrégation ou de composition ». Cette approche a permis d'obtenir un certain niveau de modélisation, mais son inconvénient est d'appliquer un ensemble fini de règles à un ensemble de constructions du langage naturel qui est lui presque infini. Outre le caractère irréaliste d'espérer couvrir l'ensemble des formulations possibles pour présenter les concepts ou les relations qui existent entre ceux-ci, cela impose un travail très important en amont de l'étude pour colliger l'information pertinente sur la langue employée. Sans compter que le travail qui est fait pour une langue n'est pas transposable à une autre : les constructions syntaxiques dans deux langues telles que le français et l'anglais par exemple sont différentes, ce qui implique que les règles doivent être adaptées. Pour contrecarrer cette difficulté, nous devons disposer d'un outil qui découvre par lui-même les règles à appliquer pour construire un modèle formel à partir du langage naturel. Ceci résoudrait le problème du temps consacré à l'identification de ces règles et de leur exhaustivité.

L'apprentissage automatique est naturellement une avenue intéressante à explorer si l'on veut mettre en place un outil qui découvre par lui-même la diversité des règles qui sous-tendent la traduction d'exigences écrites en langage naturel en un modèle formel. Particulièrement l'apprentissage automatique appliqué au domaine du traitement du langage naturel est un outil qui est devenu extrêmement puissant pour résoudre un ensemble de tâches liées au langage. Des exemples marquants sont les progrès faits par

l'outil de traduction automatique proposé par Google ou la rédaction automatique d'articles de presse. La diversité des tâches résolues est grande : traduction, reconnaissance d'entités, génération de textes, résumé de texte, résolution de coréférences, réponse automatique aux questions, analyse de sentiment, classification de textes, classification de relations, recherche d'informations sont autant d'exemples, et ce uniquement pour le traitement du langage, sans compter l'analyse d'image, la composition de musique, etc. L'outil fait l'objet d'études intensives depuis le début des années 2000 et a donc produit un ensemble d'architectures intéressantes pour résoudre bon nombre de tâches.

L'objectif de l'étude est donc d'utiliser les outils de l'apprentissage automatique pour extraire le modèle du cahier des charges, et ainsi automatiser la dernière étape de l'approche MDA qui n'a pu l'être jusqu'à présent. Plus précisément, nous voulons utiliser les architectures de l'apprentissage automatique développées dans le cadre de tâches de traitement du langage naturel afin d'extraire les concepts présents et les liens qui les unissent. Nous faisons les hypothèses 1° que les outils développés pour la détection d'entités nous permettent de découvrir les concepts et 2° que les outils développés pour la résolution de coréférences et la classification de relation nous permettent de découvrir les liens qui existent entre ceux-ci de telle sorte qu'en colligeant l'ensemble de l'information, il soit possible de produire un modèle d'affaires sous forme de diagramme de classes.

L'utilisation de l'apprentissage automatique pour la production d'un modèle de domaine n'a encore jamais été réalisée. En ce sens il s'agit d'une nouvelle tâche dans le domaine de l'apprentissage automatique. Comme dans le cas de toute nouvelle tâche, il n'existe pas de référence, de modèle ou même de jeu de données pouvant servir de base. Le

mode de résolution de la tâche est inconnu. La contribution de cette étude est donc de proposer une voie de résolution de cette tâche impliquant 1° le développement d'un jeu de données composé de cahiers de charges annotés spécifiquement pour la résolution de cette tâche, 2° l'élaboration d'un modèle de résolution destiné à analyser la structure séquentielle des phrases qui constituent les cahiers des charges et à produire de manière conjointe un ensemble de résultats dans le domaine de l'identification de concepts, de résolution de coréférences et de classification de relations nécessaire à la résolution de la tâche et 3° une référence pour d'éventuelles futures études sur le sujet.

Le présent mémoire est organisé en quatre chapitres. Les deux premiers chapitres présentent les concepts qui sont en jeu aussi bien dans le domaine de l'apprentissage automatique (CHAPITRE 1) que dans le domaine du traitement automatique des langues (CHAPITRE 2). L'apprentissage automatique est un univers en soit composé de tâches de résolutions, de modèles, de métriques, de modes de résolutions, chaque concept ayant des exigences propres, des caractéristiques qui en font des bons candidats pour la résolution de notre tâche ou de moins bons. Il s'agit donc d'appréhender l'ensemble de ce bestiaire de manière à orienter le choix proposé dans l'étude. Le traitement automatique des langues est aussi un sujet vaste qui fait appel à plusieurs champs disciplinaires. Les contributions apportées au cours de l'histoire récente sont présentées de manière à présenter la panoplie d'outils disponibles quant au traitement de ce sujet et à justifier les outils qui ont été utilisés dans l'étude ainsi que la manière dont nous avons décidé de résoudre la tâche c'est-à-dire en faisant appel aux résultats obtenus dans trois autres tâches du traitement automatique des langues que sont l'identification de concepts, la résolution de coréférences et la

classification de relations. Une revue bibliographique de ces différentes tâches présente ensuite (CHAPITRE 3) la manière dont celles-ci ont été résolues et leur pertinence quant à leur utilisation dans le contexte de la production d'un modèle de domaine. Finalement, la dernière partie du mémoire (CHAPITRE 4) présente l'étude proprement dite c'est-à-dire une description détaillée de l'approche, le jeu de données que nous avons constitué pour résoudre la tâche, le modèle et les résultats que nous avons obtenus.

CHAPITRE 1

L'APPRENTISSAGE AUTOMATIQUE ET L'APPRENTISSAGE PROFOND

L'étude propose d'utiliser les outils de l'apprentissage automatique pour réaliser l'extraction du domaine d'affaires. Le premier chapitre est une présentation de différents concepts de l'apprentissage automatique d'une part et de l'une de ses composantes, l'apprentissage profond, d'autre part. Nous y abordons les notions générales relatives à ce type d'algorithme dans un premier temps avant de resserrer le point focal sur les notions de l'apprentissage profond qui nous seront utiles pour notre objectif qui est la découverte des concepts importants d'un cahier des charges et des relations qui existent entre eux. À ce titre, une partie importante du chapitre est consacrée aux réseaux de neurones, et plus spécifiquement aux réseaux de neurones récurrents qui sont une structure bien adaptée au traitement de données séquentielles comme les mots d'un document écrit en langage naturel.

1.1 L'APPRENTISSAGE AUTOMATIQUE

L'apprentissage automatique est la capacité d'un algorithme à apprendre des données qui lui sont présentées (Goodfellow et al. 2016). De manière plus précise, il s'agit de la capacité d'un algorithme développé pour résoudre une tâche précise à améliorer sa performance, mesurée par une ou plusieurs métriques, au fur et à mesure que des données

lui sont présentées. Il apparait dans cette définition les éléments importants de l'apprentissage automatique : la tâche pour laquelle un algorithme est développé, les métriques qui sont utilisées pour mesurer sa performance et les données qui servent à son entrainement et donc à l'amélioration (potentielle) de sa performance.

1.1.1 Les tâches résolues par l'apprentissage automatique

Il existe une diversité importante de tâches qui peuvent être résolues par l'apprentissage automatique, mais elles peuvent se classer en quelques grandes catégories dont on peut citer quelques exemples (Goodfellow et al. 2016) :

- La classification, qui consiste à séparer les données en deux (classification binaire) ou plusieurs catégories (classification en classes multiples). Il peut s'agir, par exemple, d'identifier la nature d'un mot dans une phrase, le sentiment exprimé par un document ou l'objet représenté par une image.

- La régression, qui consiste à prédire une valeur à partir de données existantes. On peut ainsi vouloir prédire la température qu'il fera le lendemain à partir de données climatiques des jours précédents ou le prix d'une action boursière en fonction du comportement de la bourse.

- La traduction, qui consiste à réécrire dans une langue un document écrit dans une autre langue, ce que font les systèmes de traduction automatique tels que Google Translate.

- La détection d'anomalies, qui consiste à identifier des données dont le patron diffère d'un ou de plusieurs patrons généraux. Ce type de tâche correspond notamment à la détection de comportements frauduleux sur un compte de crédit ou d'interférences sur un signal.

- La synthèse de document, qui consiste à produire un document reprenant les idées d'un document initial, mais dans une version abrégée. Il s'agit typiquement d'un service de résumé automatique.

Toutes les tâches impliquées dans le cadre de cette étude correspondent à des tâches de classification en classes multiples. La suite du document est donc axée essentiellement sur ce type de tâche.

1.1.2 Les métriques de performance

Le principe d'un travail de classification est de répartir des données dans deux ou plusieurs catégories. Par conséquent, une donnée est classée soit dans la bonne catégorie, celle à laquelle elle est censée appartenir, soit dans une mauvaise. Pour une catégorie donnée A, nous pouvons ainsi définir les concepts de vrais positifs qui correspondent aux données correctement classées dans A, les faux positifs qui correspondent aux données classées dans A alors qu'elles appartiennent à une autre, les faux négatifs qui correspondent aux données classées dans une autre catégorie alors qu'elles appartiennent à A et les vrais négatifs qui correspondent aux données correctement classées dans une autre catégorie. Pour illustrer ces concepts, prenons l'exemple illustré à la Figure 1 d'une tâche de

classification binaire dans laquelle les données doivent être classées dans une catégorie A ou une catégorie B. La tâche de classification a extrait pour la classe A, 15 vrais positifs, 5 faux positifs, 3 faux négatifs et 12 vrais négatifs et pour la classe B, 12 vrais positifs, 3 faux positifs, 5 faux négatifs et 15 vrais négatifs.

		Réelle	
		Catégorie A	Catégorie B
Prédite	Catégorie A	15	5
	Catégorie B	3	12

Figure 1 : Exemple de matrice de confusion produite par une tâche de classification binaire.

Ces concepts permettent de mettre en place plusieurs métriques :

- La précision est la métrique qui quantifie la valeur prédictive du modèle. Elle correspond au nombre de données classées dans une catégorie qui appartiennent effectivement à cette catégorie. Elle se calcule de la manière suivante :

$$Précision = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ positifs}$$

- Le rappel est la métrique qui quantifie la sensibilité du modèle. Elle correspond au nombre de données appartenant à une catégorie qui sont effectivement classées dans celle-ci. Elle se calcule de la manière suivante :

$$Rappel = \frac{Vrais\ positifs}{Vrais\ positifs + Faux\ négatifs}$$

- La mesure F est une mesure qui souhaite donner une idée générale de la performance d'un modèle quant à la réalisation d'une tâche. La précision et le rappel donnent une idée de la manière dont se comporte le modèle, mais il est difficile de réellement comparer deux modèles avec ces deux mesures. En effet, comment classer deux modèles dont l'un a une précision plus grande, mais un rappel plus petit que l'autre ? La mesure F, parce qu'elle fait une moyenne harmonique de la précision et du rappel, est destinée à être un outil de comparaison. Il existe plusieurs mesures F_β qui varient suivant la valeur du paramètre β et qui se calculent de la manière suivante :

$$F_\beta = \frac{(1 + \beta^2) \times (Précision \times Rappel)}{(\beta^2 \times Précision + Rappel)}$$

Parmi les valeurs possibles de β, la plus utilisée est 1, ce qui aboutit à la mesure F_1 :

$$F_1 = \frac{2 \times Précision \times Rappel}{Précision + Rappel}$$

$$F_1 = \frac{2 \times Vrais\ positifs}{2 \times Vrais\ positifs + Faux\ positifs + Faux\ négatifs}$$

Les métriques précédentes sont relatives à une classe. Il est néanmoins possible de calculer ces métriques sur l'ensemble des classes pour donner une idée globale de la performance du modèle. Dans ce cas, la précision et le rappel se calculent en effectuant la moyenne des mesures de précision et de rappel obtenues sur les classes :

$$Précision = \frac{\sum_{i=1}^{n} Précision_i}{n}$$

$$Rappel = \frac{\sum_{i=1}^{n} Rappel_i}{n}$$

Où i désigne l'identifiant de la classe.

Le calcul de la mesure F se fait de la même manière que dans le cas d'une classe, mais en prenant les valeurs de précision et de rappel globales.

1.1.3 Les données d'entrainement

Il existe deux grandes manières d'utiliser l'apprentissage automatique : soit on veut lui faire apprendre quelque chose, soit on veut lui faire découvrir quelque chose. Pour faire apprendre quelque chose à un modèle, il faut pouvoir lui montrer ce que l'on attend de lui pour lui faire apprendre de ses erreurs. Ceci implique d'avoir des données pour lesquelles on connait le résultat attendu, résultat que l'on peut confronter à la prédiction du modèle de manière à le corriger au fur et à mesure. Ces données pour lesquelles on a préalablement déterminé le résultat attendu sont dites « annotés » et l'utilisation de données annotées ou

pas marque la grande distinction entre deux approches de l'apprentissage automatique : le mode supervisé et le mode non supervisé.

1.1.3.1 Le mode supervisé

Le mode supervisé cherche à faire apprendre quelque chose au modèle. De manière générale, il s'agit de faire prédire au modèle le bon résultat y à partir d'une observation x. Par conséquent, de manière probabiliste, l'objectif est de déterminer la probabilité conditionnelle d'obtenir y sachant que l'on a observé x, soit la probabilité $P(y|x)$ (Goodfellow et al. 2016). Un exemple de tâche pouvant être réalisée en mode supervisé est la classification d'image dans laquelle une image x doit être classée dans la bonne catégorie y suivant ce qu'elle représente.

Le mode supervisé est donc caractérisé par l'utilisation de données annotées, c'est-à-dire des données pour lesquelles on a préalablement défini le résultat attendu. La comparaison entre ce résultat attendu et la prédiction générée par le modèle constitue la base de l'apprentissage : si le modèle génère le résultat attendu, il n'y a pas besoin d'en modifier les paramètres, sinon, des changements doivent être apportés et la différence constatée entre prédiction et résultat attendu oriente la manière dont les changements doivent être abordés.

L'inconvénient du mode supervisé est qu'il nécessite de disposer d'un jeu de données annotées. L'annotation d'un nombre conséquent d'échantillons est un travail laborieux qui doit être réalisé avant même le début de l'expérimentation et qui prend un temps certain.

Par conséquent, se lancer dans l'annotation est un investissement de temps important dans une expérimentation. Pour faciliter les démarches, un certain nombre de jeux de données ont été développés et rendus publics pour une diversité de tâche. Ceux-ci restent néanmoins spécifiques à une ou quelques tâches assez générales comme la catégorisation d'images, la reconnaissance d'entités nommées, la résolution de coréférences, et inutilisables dans le contexte de tâches plus spécifiques.

1.1.3.2 Le mode non supervisé

Le mode non supervisé cherche à faire découvrir quelque chose. De manière générale, on ne s'attend pas à faire le lien entre une observation x et un résultat y, mais à découvrir quelque chose sur les observations recueillies. Par conséquent, de manière probabiliste, l'objectif est de trouver les caractéristiques de la loi de distribution de x, $P(x)$ (Goodfellow et al. 2016). Un exemple de tâche pouvant être réalisée en mode non supervisé est le regroupement d'images dans laquelle on cherche à regrouper des images x suivant le motif de ce qu'elle représente, sans pour autant que nous ne sachions à l'avance quels sont ces motifs ou le type de regroupement attendu.

Le mode non supervisé n'utilise pas de données annotées puisqu'on ne s'intéresse pas au lien entre une observation et un résultat. Le travail en amont de l'utilisation du modèle est donc allégé puisqu'il se borne au recueil d'un certain nombre d'échantillons. C'est un avantage par rapport au mode supervisé, mais il ne permet pas d'effectuer le même type de tâche.

Le mode non supervisé n'est pas le mode envisagé dans le cadre de cette étude. Par conséquent, le reste du document se concentre essentiellement sur le mode supervisé.

1.2 LES APPROCHES DE L'APPRENTISSAGE AUTOMATIQUE

Cette section présente les principes régissant la résolution de tâche par quelques algorithmes de l'apprentissage automatique. L'objectif n'est pas de donner une liste exhaustive de tous les algorithmes, mais de présenter les approches de résolution et de donner quelques exemples à titre d'illustration.

1.2.1 Les modèles non probabilistes

1.2.1.1 Les arbres de décision

Les arbres de décision sont un algorithme simple de résolution qui permet de classer des données en suivant un chemin de décisions binaires dont le parcours dépend des réponses données aux différents embranchements. Les arbres de décisions tentent ainsi de faire correspondre chaque point de l'espace des entrées (X) à une région précise de l'espace des solutions (Y). Dans les arbres de décisions, les nœuds correspondent aux décisions tandis que les feuilles correspondent à une solution. Le parcours de l'arbre se fait de nœud en nœud jusqu'à l'arrivée sur une feuille qui marque la fin du processus.

L'apprentissage se fait en construisant dynamiquement l'arbre de décision qui permet de faire correspondre le plus adéquatement l'espace des entrées de l'ensemble

d'entrainement aux solutions attendues (Figure 2). Outre le fait de pouvoir associer la bonne sortie à la bonne entrée, la difficulté de l'apprentissage est de construire un arbre injectif, c'est-à-dire que chaque entrée ne peut être associée qu'à une seule sortie, de manière à lever les ambigüités.

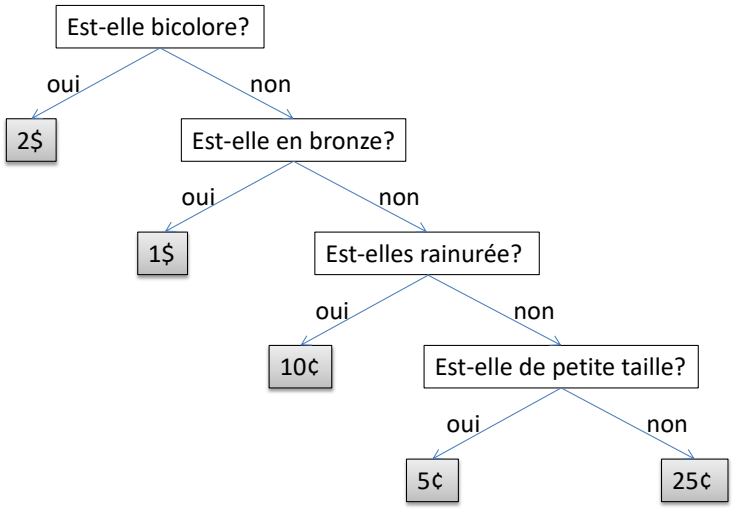


Figure 2. Illustration du principe de fonctionnement des arbres de décision sur la détermination de la valeur d'une pièce de monnaie.

1.2.1.2 Les machines à vecteurs de support

Les machines à vecteurs de support (Support Vector Machine SVM – Cortes & Vapnik 1995) sont une technique d'apprentissage automatique utilisée dans le cadre de problèmes binaires linéairement séparables, c'est-à-dire que les données peuvent être classées dans deux catégories et que ces deux catégories sont séparables par une ligne, un plan ou un hyperplan ($w \cdot x + b = 0$) suivant la dimension des données. Le modèle apprend sur un ensemble d'entrainement à discriminer les deux catégories de données et à trouver la séparation qui permet d'avoir la plus grande distance entre les deux catégories, ce

qui le rend plus robuste lors de l'analyse de nouvelles données. Pour classer une nouvelle donnée x, il suffit alors de déterminer le signe de $w \cdot x + b$ (Francoeur 2010) :

$$Classe(x) = signe(w \cdot x + b)$$

Or w peut être réécrit en fonction des données d'entrainements (x_1, x_2, …, x_l) et de leur valeur respective (y_1, y_2, …, y_l) :

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

L'entrainement du modèle sert à déterminer les coefficients α_i adéquats du problème. b est quant à lui déterminer à partir de certains points particuliers de l'ensemble d'entrainement. Finalement, le classement d'une nouvelle donnée revient à calculer :

$$Classe(x) = signe\left(\sum_{i=1}^{l} (\alpha_i y_i x_i \cdot x) + b\right)$$

Il est possible d'étendre l'utilisation des machines à vecteurs de support aux problèmes non linéairement séparables en considérant qu'il existe une transformation φ des données qui les rend linéairement séparables dans un autre espace, possiblement un espace de plus grande dimension (Francoeur 2010). Dans ce cas,

$$Classe(x) = signe\left(\sum_{i=1}^{l}\left(\alpha_i y_i \varphi(x_i) \cdot \varphi(x)\right) + b\right)$$

Le problème est tout d'abord de trouver la bonne fonction φ et ensuite de procéder aux calculs dans un espace qui peut être de grande dimension, ce qui est exigeant en termes de temps de calcul. Les fonctions noyaux ont été introduites pour contrecarrer ce problème. Elles sont définies de la manière suivante :

$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$

Donc,

$$Classe(x) = signe\left(\sum_{i=1}^{l}\left(\alpha_i y_i K(x_i, x)\right) + b\right)$$

La fonction de transformation φ n'apparait plus dans le calcul, ce qui permet d'éviter de la chercher et de faire des calculs sur de grandes dimensions, à condition de disposer de la fonction noyau. Or, les fonctions noyaux correspondent à toute fonction symétrique et semi-définie positive (Francoeur 2010). Par conséquent, il suffit de choisir une fonction respectant ces caractéristiques pour disposer d'une fonction noyau. Une fonction calculant le produit scalaire de deux données est un exemple de fonction noyau. Il est possible d'en obtenir d'autres, grâce à quelques opérations sur des fonctions noyaux déjà identifiées.

Les machines à vecteurs de support sont conçues pour discriminer des données binaires. En principe elles ne sont donc pas adaptées à une tâche de classification multi classes. Néanmoins, elles ont tout de même été employées pour ce type de travail. L'idée est d'utiliser non pas un classificateur pour la tâche, mais plusieurs, un pour chaque classe, chacun déterminant si une donnée appartient à la classe qu'il représente ou à la classe « autre » (Zhou et al. 2005). En croisant les informations entre les classificateurs et en identifiant celui dont la discrimination est la plus probante, il est possible de conclure que la donnée appartient à une classe précise.

1.2.2 Les modèles probabilistes

De manière probabiliste, le principe du mode supervisé est de trouver la loi de distribution $P(Y|X)$ qui modélise le mieux les résultats observés dans l'ensemble des échantillons, loi qui est déterminée par un ensemble de paramètres θ, ce que l'on peut noter $P(Y|X,\theta)$. Autrement dit, il s'agit de trouver θ tel que la probabilité $P(Y|X,\theta)$ explique le mieux l'obtention de l'ensemble des résultats Y à partir de l'ensemble des échantillons recueillis X. Pour identifier cet ensemble θ, nous avons recours à la fonction de vraisemblance :

$$L(\theta|X,Y) = P(Y|X,\theta) = \prod_{i=1}^{n} P(y_i|x_i,\theta)$$

Trouver l'ensemble θ optimal revient à trouver le maximum $\hat{\theta}$ de la fonction de vraisemblance $L(\theta|X)$:

$$\hat{\theta} = argmax_{\theta} L(\theta|X,Y)$$

$$\hat{\theta} = argmax_{\theta} \prod_{i=1}^{n} P(y_i|x_i,\theta)$$

Puisque la fonction logarithme est strictement croissante, nous pouvons prendre le logarithme de la fonction de vraisemblance sans en changer le maximum, ce qui simplifie la résolution :

$$\hat{\theta} = argmax_{\theta} \log L(\theta|X,Y)$$

Chercher le maximum du logarithme de la fonction de vraisemblance revient à chercher le minimum de la fonction opposée :

$$\hat{\theta} = argmin_{\theta} - \log L(\theta|X,Y)$$

$$\hat{\theta} = argmin_{\theta} - \log \prod_{i=1}^{n} P(y_i|x_i,\theta)$$

$$\hat{\theta} = argmin_{\theta} \sum_{i=1}^{n} - \log P(y_i|x_i,\theta)$$

Nous pouvons introduire une fonction de coût $J(\theta)$ telle que :

$$J(\theta) = \frac{1}{n}\big(-logL(\theta|X,Y)\big) = \frac{1}{n}\sum_{i=1}^{n} - \log P(y_i|x_i,\theta)$$

Dans ce cas, trouver $\hat{\theta}$ revient à minimiser la fonction de coût $J(\theta)$. Ce minimum s'observe lorsque la dérivée de la fonction s'annule. Par conséquent, nous voulons trouver $\hat{\theta}$ tel que :

$$\nabla_\theta J(\hat{\theta}) = g(\hat{\theta}) = 0$$

Déterminer $\hat{\theta}$ directement en résolvant l'équation n'est pas possible. Par contre, nous pouvons le trouver ou l'estimer de manière incrémentale en utilisant la technique de descente du gradient. L'idée, illustrée à la Figure 3, est de parcourir la courbe représentant la fonction de coût par pas successifs jusqu'à rejoindre son minimum. Pour ce faire, l'ensemble d'entrainement est parcouru par groupe d'une taille m de quelques échantillons (batch). Pour chaque groupe, le gradient de la fonction de coût est estimé :

$$g_{ech}(\theta) = \nabla_\theta J_{ech}(\theta) = \frac{1}{m}\sum_{i=1}^{m} -\log L(\theta|x_i, y_i)$$

L'ensemble $\hat{\theta}$ est obtenu progressivement en améliorant θ à chaque étape grâce à l'opération :

$$\theta = \theta - \epsilon g_{ech}$$

Où ϵ est le taux d'apprentissage définit dans les hyper paramètres de l'expérimentation.

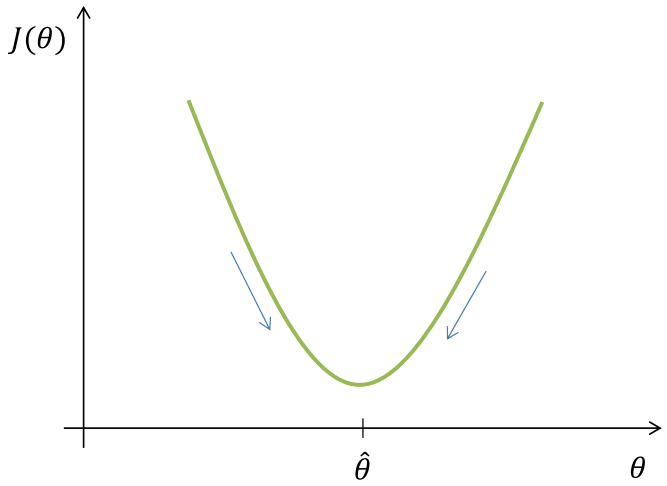


Figure 3. Illustration du principe de la descente de gradient.

1.2.2.1 La régression logistique

La régression logistique binaire est un outil de classification binaire au même titre que les machines à vecteurs de support. Elle estime la probabilité qu'une donnée appartienne à la classe c (par exemple 1 ou 0) en calculant le score $w \cdot x + b$ comme dans le cas des machines à vecteurs de support. Ce score correspond au logit c'est-à-dire au logarithme du rapport des probabilités des deux classes :

$$logit = ln\left(\frac{P(Y=1)}{P(Y=0)}\right) = \ln\left(\frac{p}{1-p}\right) = w \cdot x + b$$

D'où

$$P(Y=1) = p = \frac{e^{w \cdot x + b}}{1 + e^{w \cdot x + b}} = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

La régression logistique multi-classes est une extension de la régression logistique binaire pour le cas où le nombre de classes possibles est supérieur à deux. Dans ce cas, on peut recalculer une fonction logit pour les différentes classes en prenant l'une des classes (K) comme pivot :

$$logit(1) = ln\left(\frac{P(Y=1)}{P(Y=K)}\right) = w_1 \cdot x + b_1 \Rightarrow P(Y=1) = P(Y=K)e^{w_1 \cdot x + b_1}$$

$$logit(2) = ln\left(\frac{P(Y=2)}{P(Y=K)}\right) = w_2 \cdot x + b_2 \Rightarrow P(Y=2) = P(Y=K)e^{w_2 \cdot x + b_2}$$

$$logit(K-1) = ln\left(\frac{P(Y=K-1)}{P(Y=K)}\right) = w_{K-1} \cdot x + b_{K-1}$$

$$\Rightarrow P(Y=K-1) = P(Y=K)e^{w_{K-1} \cdot x + b_{K-1}}$$

D'où

$$P(Y=K) = 1 - \sum_{k=1}^{K-1} P(Y=k) = 1 - \sum_{k=1}^{K-1} P(Y=K)\, e^{w_k \cdot x + b_k}$$

$$\Rightarrow\ P(Y=K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k}}$$

Or,

$$logit(K) = ln\left(\frac{P(Y=K)}{P(Y=K)}\right) = w_K \cdot x + b_K$$

$$\Rightarrow 1 = e^{w_K \cdot x + b_K}$$

$$\Rightarrow 1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k} = e^{w_K \cdot x + b_K} + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k}$$

$$\Rightarrow 1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k} = \sum_{k=1}^{K} e^{w_k \cdot x + b_k} = Z(x)$$

On peut donc réécrire les probabilités associées à chaque classe :

$$P(Y=1) = \frac{e^{w_1 \cdot x + b_1}}{1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k}} = \frac{e^{w_1 \cdot x + b_1}}{\sum_{k=1}^{K} e^{w_k \cdot x + b_k}} = \frac{1}{Z(x)} e^{w_1 \cdot x + b_1}$$

$$P(Y=2) = \frac{e^{w_2 \cdot x + b_2}}{1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k}} = \frac{1}{Z(x)} e^{w_2 \cdot x + b_2}$$

$$P(Y=K-1) = \frac{e^{w_{K-1} \cdot x + b_{K-1}}}{1 + \sum_{k=1}^{K-1} e^{w_k \cdot x + b_k}} = \frac{1}{Z(x)} e^{w_{K-1} \cdot x + b_{K-1}}$$

Et de manière générale pour une classe c :

$$P(Y = c) = \frac{1}{Z(x)} e^{w_c \cdot x + b_c} \; avec \; Z(x) = \sum_{k=1}^{K} e^{w_k \cdot x + b_k}$$

On remarque que les paramètres w_c et b_c dépendent de la classe considérée. Pour généraliser la notation, on peut introduire des fonctions caractéristiques (Sutton & McCallum 2007) qui s'annulent pour toute classe différente de c et qui valent 1 dans le cas contraire. On a alors

$$P(Y = y) = P(y|x) = \frac{1}{Z(x)} e^{w \cdot f(y,x) + b} \; avec \; Z(x) = \sum_{k=1}^{K} e^{w \cdot f(k,x) + b}$$

Les paramètres w et b sont déterminées à partir de l'ensemble d'entrainement en appliquant le principe du maximum de vraisemblance. On peut simplifier l'écriture en considérant que b fait partie de la matrice w et que celle-ci est multipliée par le vecteur $(f(y,x),1)$:

$$P(y|x) = \frac{1}{Z(x)} e^{w \cdot f(y,x)} \; avec \; Z(x) = \sum_{k=1}^{K} e^{w \cdot f(k,x)}$$

1.2.2.2 Les champs aléatoires conditionnels

Les champs aléatoires conditionnels (Conditionnal Random Fields CRF – Lafferty et al. 2001) sont l'extension de la régression logistique pour les séquences linéaires (Figure 4) respectant la propriété de Markov, c'est-à-dire tels que la probabilité d'obtenir la classe y_t

au temps t ne dépend que de l'entrée x_t et de la classe y_{t-1} obtenue au temps $t-1$ (Lafferty et al. 2001) :

$$P(y_t|x_t, y_v, v \neq t) = P(y_t|x_t, y_v, v = t-1)$$

Dans ces conditions,

$$P(y|x) = \frac{1}{Z(x)} e^{\alpha \cdot f(y_t, x_t) + \beta \cdot g(y_t, y_{t-1})} \; avec \; Z(x) = \sum_{k=1}^{K} e^{\alpha \cdot f(k_t, x_t) + \beta \cdot g(k_t, k_{t-1})}$$

ou plus succinctement (Sutton & McCallum 2007),

$$P(y|x) = \frac{1}{Z(x)} e^{\theta \cdot h(y_t, y_{t-1}, x_t)} \; avec \; Z(x) = \sum_{k=1}^{K} e^{\theta \cdot h(k_t, k_{t-1}, x_t)}, \theta = (\alpha, \beta) \; et \; h = (f, g)$$

Encore une fois, les paramètres du modèle sont estimés avec les données de l'ensemble d'entrainement par la méthode du maximum de vraisemblance.

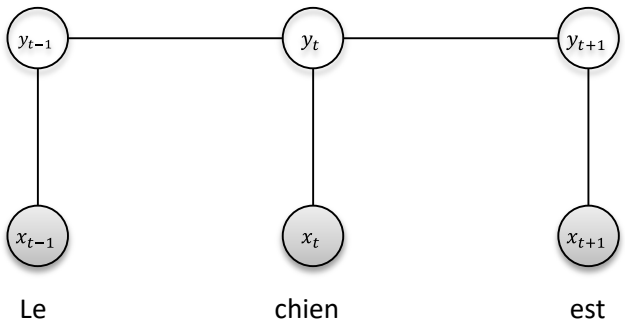


Figure 4. Illustration d'une séquence linéaire (d'après Sutton & McCallum 2007). Dans ce cas, la classe y_t attribuée au jeton x_t 'chien' dépend uniquement de la classe y_{t-1} attribuée au jeton précédent 'Le' dans la séquence et de x_t.

1.3 L'APPRENTISSAGE PROFOND

1.3.1 Les principes généraux

L'apprentissage automatique « traditionnel » décrit dans les sections précédentes se heurte à un obstacle lorsque la dimension des données d'entrée devient grande, ce qui est appelé le fléau de la dimension (Goodfellow et al. 2016). Si un échantillon est représenté par un scalaire dont la valeur ne peut être qu'un entier compris entre 0 et 9, il existe uniquement 10 échantillons différents possibles. Il est raisonnable d'envisager dans ce cas que l'on puisse récolter un échantillon de chaque sorte dans une campagne d'échantillonnage. Si un échantillon est représenté par un vecteur de 10 scalaires chacun pouvant être une valeur entière comprise entre 0 et 9, il existe 100 échantillons différents possibles. Un échantillonnage complet reste envisageable. Si un échantillon est représenté par une matrice (10×10) de scalaires chacun pouvant être une valeur entière comprise entre 0 et 9, il existe 1000 échantillons possibles. La possibilité de recueillir un échantillon de chaque sorte devient réduite. Or, les algorithmes d'apprentissage automatique fonctionnent

bien lorsque l'on dispose de toute la variabilité de données possibles, mais sont moins robustes lorsqu'on ne les entraine que sur un sous-échantillon de celles-ci. Prenons l'exemple des machines à vecteurs de support. L'hyper plan qui fait la distinction entre les deux classes de données est entièrement déterminé par les valeurs recueillies dans l'ensemble d'entrainement et les nouvelles données sont classées par rapport à celles-ci. La valeur du modèle repose donc sur la représentativité de ces données d'entrainement. Plus intrigant encore est le fait que seuls les points qui sont sur la marge, c'est-à-dire les points qui sont les plus proches de l'hyper plan, ont réellement une importance dans la détermination des coordonnées de ce plan. Il y a donc quelques échantillons qui ont une importance extrêmement grande dans l'efficacité du modèle. Manquer ces échantillons importants devient donc très préjudiciable. Le cas de l'arbre de décision est aussi très explicite : le modèle est entrainé à départager l'espace des solutions à partir des caractéristiques des exemples présents dans l'ensemble d'entrainement uniquement. Il n'y a donc aucun moyen de savoir comment le modèle va se comporter avec des données différentes dont l'assemblage de caractéristiques n'a jamais été rencontré.

L'apprentissage profond permet de développer des modèles plus robustes que l'apprentissage automatique dans le cas où il est difficile d'avoir accès à un ensemble d'entrainement qui couvrirait tout le champ des possibles. La conséquence de cette faculté de généralisation est que l'apprentissage profond a permis de passer de données constituées d'un ensemble relativement restreint de caractéristiques qui doivent être choisies avec soin à des données constituées de nombreuses caractéristiques simples que le modèle va apprendre à trier et à analyser. Avec l'apprentissage automatique, le choix des

caractéristiques est une étape d'ingénierie en soi. Il faut choisir les bonnes caractéristiques c'est-à-dire les caractéristiques qui discriminent le mieux les données, composer des caractéristiques nouvelles à partir de caractéristiques élémentaires pour renforcer leur significativité, évaluer les différentes compositions de caractéristiques pour trouver l'ensemble le plus pertinent. Ces phases préliminaires sont extrêmement couteuses en termes de temps de développement. En permettant de prendre en charge un ensemble de caractéristiques plus grand, l'apprentissage profond permet de rester au niveau de la caractérisation élémentaire et de s'affranchir de cette phase d'ingénierie.

1.3.2 Les réseaux de neurones

Les algorithmes d'apprentissage profond mettent en jeu une architecture modulaire de traitement constituée de couches de neurones successives. Le terme de neurones est employé parce que le fonctionnement de ces « cellules » est emprunté aux neurones que l'on retrouve naturellement chez les animaux. Ainsi, tout comme leur contrepartie naturelle, elles reçoivent un signal, en font un traitement qui dépend des caractéristiques de la cellule, ce qui peut déclencher l'émission d'une réponse non linéaire, réponse qui est transmise ensuite à une autre cellule située en aval dans le réseau et qui va répéter le processus de traitement jusqu'à ce que les cellules de la dernière couche produisent une réponse adaptée à un stimulus. On peut pousser la comparaison plus loin en observant que lorsqu'un animal reçoit un stimulus, ce que l'on peut comparer à la donnée d'entrée de notre modèle, la donnée est conduite au travers d'un ou de plusieurs neurones jusqu'à un centre de traitement central, le cerveau chez des animaux comme les mammifères, ce que l'on appelle la voie afférente, avant qu'une réponse ne soit déclenchée et propagée jusqu'aux

organes responsables de mettre en œuvre la réponse, ce que l'on appelle la voie efférente. Cette nomenclature constituée de deux sous-réseaux, afférent et efférent, est aussi une composante des réseaux de neurones de l'apprentissage profond que l'on peut décomposer en un sous-réseau d'encodage qui se charge de constituer une représentation des données d'entrée et un sous-réseau de décodage qui se charge d'interpréter cette représentation et de produire la solution.

Les réseaux de neurones les plus courants et les plus simples sont les réseaux à propagation avant (FeedForward Neural Network – FFNN). Ils sont constitués d'une ou de plusieurs couches de neurones, chacune constituée d'un ou de plusieurs neurones qui traitent en parallèle une partie de l'information d'entrée (Figure 5). Chaque neurone reçoit un signal d'entrée (typiquement un scalaire ou un vecteur), procède à un traitement linéaire dont le résultat passe au travers d'une fonction d'activation, généralement non linéaire, afin de produire un signal de sortie. Le traitement du signal se propage au travers des différentes couches jusqu'à l'obtention de la réponse.

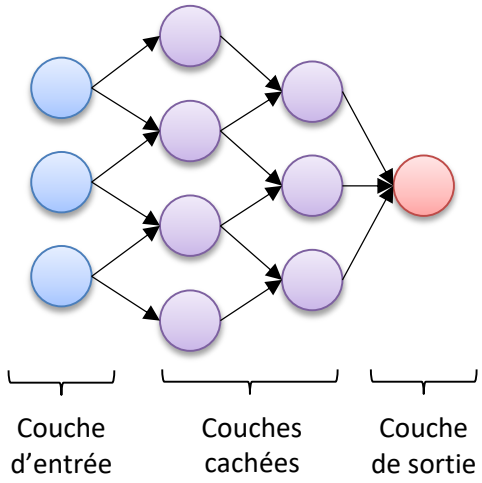


Figure 5. Illustration de l'architecture des réseaux de neurones

Parmi les FFNN, le plus connu est le perceptron qui est un cas particulier dans lequel tous les neurones de la couche c sont reliés à tous les neurones de la couche $c + 1$ (Figure 6).

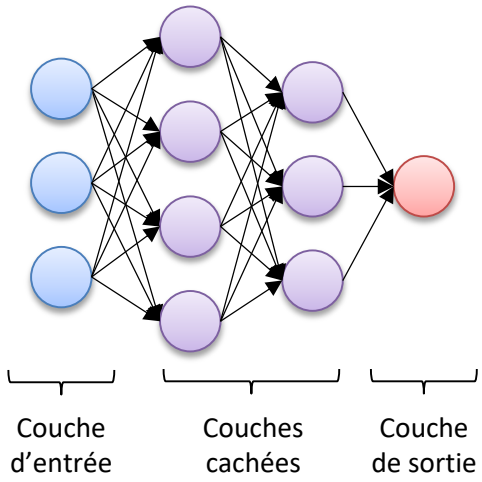


Figure 6. Illustration de l'architecture d'un perceptron multicouche.

Il existe d'autres architectures de neurones, adaptées à la résolution de tâches particulières, par exemple l'analyse de données séquentielles, que nous détaillons dans les sections suivantes.

1.3.3 La rétro propagation de l'erreur

Les algorithmes d'apprentissage profond sont des algorithmes probabilistes. À ce titre, ils peuvent être améliorés au cours d'une phase d'entrainement en utilisant la technique de descente de gradient qui modifie θ de la manière décrite précédemment :

$$\theta = \theta - \epsilon \nabla J$$

Cette formule est suffisante pour un réseau composé d'une seule couche de neurones. J est déterminée à partir de la fonction de coût et les paramètres θ de la couche de neurones sont modifiés en conséquence. Cependant, la plupart des réseaux sont constitués de plusieurs couches. Il faut donc pouvoir propager la modification du gradient dans les différentes couches du réseau qui contiennent des paramètres à ajuster.

Le principe est d'utiliser la règle de dérivation en chaine au niveau des différentes couches du réseau :

$$\frac{dz}{dx} = \frac{dz}{dy} \times \frac{dy}{dx}$$

Si l'on prend l'exemple simple d'un réseau constitué de deux couches de neurones avec la première couche qui prend le vecteur x en entrée et calcule la sortie y suivant la fonction $y = f(x)$, et la seconde couche qui prend la sortie y de la première couche comme entrée et calcule la sortie z suivant la fonction $g(y) = z$, alors $z = g\big(f(x)\big) = h(x)$ avec $h = g \circ f$. La dérivée de la fonction composée est alors :

$$\frac{d(g \circ f)}{dx} = \frac{dg}{df} \times \frac{df}{dx}$$

Pour un vecteur, ceci s'écrit :

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \times \frac{\partial y_j}{\partial x_i}$$

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \nabla_y z$$

Où $\left(\frac{\partial y}{\partial x}\right)$ est la matrice jacobienne de la fonction f.

Ce que l'on fait de manière concrète est de calculer le coût sur un sous-ensemble de l'ensemble d'entrainement puis on calcule les dérivées successives des fonctions impliquées dans les différentes couches en utilisant la règle de dérivation en chaine. Les poids des différentes couches sont alors modifiés suivant la méthode de descente du gradient impliquant la dérivée calculée au niveau de la couche considérée et un facteur d'apprentissage ε tel que décrit précédemment.

1.3.4 Les niveaux d'apprentissage

Au fur et à mesure de l'apprentissage, le modèle s'adapte aux données d'entrainement, ce qui améliore sa performance. Si l'ensemble d'entrainement est représentatif des données possibles, le modèle devient de plus en plus efficace à réaliser la tâche pour laquelle il est développé. Cette amélioration est généralement asymptotique c'est-à-dire que l'amélioration est rapide au départ puis de plus en plus lente jusqu'à se stabiliser. À partir du moment où la performance commence à se stabiliser, le modèle commence à se spécialiser sur les données de l'ensemble d'entrainement, à apprendre des relations qui peuvent ne se retrouver que dans ces données et ne pas être généralisables. Pour pouvoir distinguer ce moment où le modèle cesse de s'améliorer sur les données générales, nous avons recours à un ensemble de validation.

L'ensemble de validation est un sous-ensemble de l'ensemble d'entrainement qui ne sert pas à l'entrainement proprement dit, mais qui sert à détecter les deux phases du cycle d'apprentissage (Figure 7). Dans la première phase, le modèle améliore sa performance sur les données d'entrainement et sur les données de validation. Arrêter l'entrainement dans la phase 1 se solderait par un modèle sous-entraîné puisqu'il n'a pas encore atteint son niveau de performance optimal. Dans la seconde phase, le modèle améliore encore ses performances sur l'ensemble d'entrainement (quoi que généralement assez faiblement), mais plus sur les données de validation. Il a même tendance à augmenter l'erreur commise sur ces données. Ceci indique que le modèle commence à être surentrainé, c'est-à-dire qu'il apprend des relations qui sont spécifiques à l'ensemble d'entrainement, mais qui ne sont plus adaptées aux données générales. L'écart entre l'erreur commise sur l'ensemble

d'entrainement et l'ensemble de validation, que l'on appelle l'erreur de généralisation, augmente. L'objectif est donc de détecter le moment où la performance commence à se dégrader sur les données de validations de manière à arrêter l'entrainement du modèle au niveau optimal.

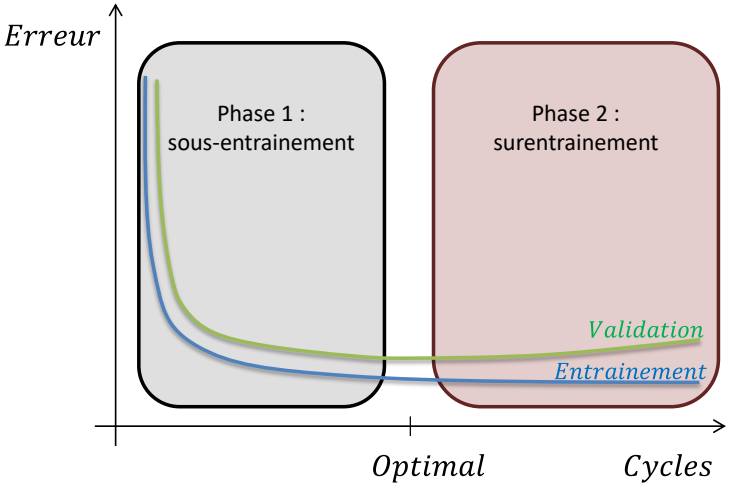


Figure 7. Illustration des phases d'apprentissage d'un réseau de neurones.

1.3.5 La régularisation

La détection de la capacité optimale d'un modèle est donc un point central de l'entrainement de ce modèle. Pour aider à atteindre ce point, différentes techniques ont été mises au point, ce que l'on appelle les techniques de régularisation. La régularisation est définie comme toute modification apportée à un algorithme d'apprentissage qui a pour vocation de diminuer l'erreur de généralisation, mais pas l'erreur d'entrainement (Goodfellow et al. 2016). En d'autres termes, ces techniques réduisent la performance du modèle sur l'ensemble d'entrainement au profit d'une meilleure capacité de généralisation. Il existe plusieurs techniques de régularisation, nous n'en donnons que quelques exemples.

L'arrêt anticipé est une technique simple qui utilise directement l'ensemble de validation. L'idée est de détecter le moment où l'erreur commise sur l'ensemble de validation commence à augmenter pour déterminer le point où la capacité optimale est atteinte. Concrètement, à chaque cycle d'entrainement, chaque époque (epoch), l'erreur sur l'ensemble de validation est calculée. Au moment où celle-ci commence à augmenter, l'entrainement est arrêté. Il est certain que l'erreur commise, que ce soit sur l'ensemble d'entrainement ou sur l'ensemble de validation, ne suit jamais une courbe parfaitement lisse comme illustrée dans la Figure 7. L'évolution est faite de diminutions et d'augmentations au fur et à mesure des cycles. Par conséquent, arrêter l'entrainement à la première hausse venue n'est pas forcément judicieux. Pour cela on introduit un tampon, c'est-à-dire un nombre de cycles durant lesquels on continue d'entrainer le modèle après une augmentation de l'erreur de validation. Si au cours de ce tampon, l'erreur recommence à diminuer, le tampon est réinitialisé et l'entrainement se poursuit. Si, par contre, l'erreur continue d'augmenter, alors ceci tend à montrer que l'on atteint la capacité optimale et l'entrainement s'arrête. Les paramètres du modèle tels qu'ils étaient lors de la dernière diminution de l'erreur de validation sont conservés.

L'augmentation de données est une autre technique de régularisation simple, mais qui n'est applicable que dans certains cas. Il s'agit d'augmenter « artificiellement » le nombre de données de l'ensemble d'entrainement, en créant de nouveaux échantillons à partir des données originales. La taille de l'ensemble d'entrainement est un facteur important de la performance d'un modèle : plus le modèle voit d'échantillons, meilleure est sa capacité de généralisation. La constitution d'un ensemble d'entrainement est cependant une tâche

longue et fastidieuse, notamment dans un contexte supervisé où chaque échantillon doit être annoté. Pour améliorer la qualité de l'ensemble d'entrainement, on a alors recours à la création de nouveaux échantillons à partir de ceux dont on dispose. La technique se prête bien au cas des tâches de classification d'images. Chaque image peut ainsi donner naissance, par quelques transformations (translations, symétries, déformations, rotations, changements de teintes) à une série de nouvelles images avec la même annotation que l'image originale.

L'injection de bruit peut s'utiliser à différents niveaux du modèle. L'injection de bruits au niveau des données d'entrées est une forme d'augmentation de données : de nouveaux échantillons sont ajoutés à l'ensemble d'entrainement en altérant un échantillon original. Dans ce cas l'objectif de l'injection de bruit est le même que dans le cas de l'augmentation de données, c'est-à-dire augmenter le nombre et la variabilité des données présentées au modèle afin d'augmenter sa robustesse.

L'injection de bruit au niveau des poids du modèle est une technique permettant d'explorer plus largement l'espace des solutions. La descente de gradient améliore les poids du modèle selon la pente de la fonction de coût dans l'environnement de la solution en cours d'évaluation. Si la fonction de coût devient constante aux alentours d'un point, le gradient devient nul et le modèle ne peut plus s'améliorer avec la technique de descente de gradient, même si un minimum (local ou global) se trouve à proximité. L'injection de bruit au niveau des poids permet d'apporter de petites modifications au modèle, ce qui offre la chance d'aller trouver ce minimum dans un contexte où le gradient est négligeable. Si le taux d'apprentissage n'est pas trop élevé, ce que l'on souhaite pour que le modèle

converge, la convergence aboutira au niveau du minimum (global ou local) le plus proche sans que l'on ait pu explorer d'autres régions de l'espace des solutions, possiblement plus prometteuses. L'injection de bruit au niveau des poids permet, à l'image d'un algorithme génétique[1], de créer des solutions différentes qui vont éventuellement converger vers le minimum global (Figure 8).

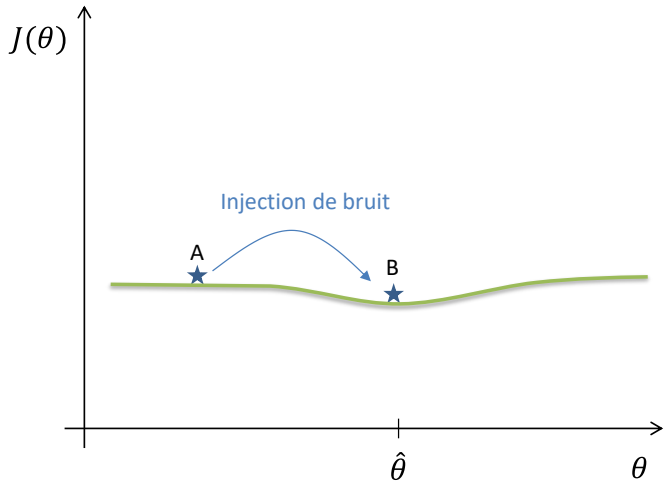


Figure 8. Illustration du principe de l'injection de bruit au niveau des poids. Le modèle situé en A dans l'espace des solutions ne peut plus s'améliorer avec la descente de gradient, mais se retrouve par injection de bruit au niveau de B qui est une solution plus efficace.

L'injection de bruit peut aussi se faire au niveau de la sortie. Dans un ensemble d'entrainement, les probabilités que certains échantillons soient mal annotés sont fortes. Or les conséquences de cette mauvaise annotation sont grandes pour la qualité de l'entrainement du modèle. L'idée est donc de ne pas considérer que l'annotation est bonne

[1] Algorithme appartenant à la famille des algorithmes évolutionnistes, dont le principe reprend celui de la sélection naturelle, destiné à produire une solution à un problème insoluble en un temps raisonnable. Un ensemble de solutions possibles est produit à priori et testé, puis les meilleures solutions sont sélectionnées pour participer à la production de la génération suivante de solutions potentielles. Le processus est répété jusqu'à ce qu'une ou quelques solutions soit suffisamment proches de l'objectif pour être acceptables.

à tous les coups, mais seulement avec une probabilité $1 - \varepsilon$. En pratique, cela permet d'améliorer la convergence des modèles.

L'ensachage (bagging) reprend l'idée d'explorer plus largement l'espace des solutions évoquée avec l'injection de bruit au niveau des poids. L'ensachage consiste, à l'image d'un algorithme génétique, de travailler avec une population de modèles plutôt qu'avec un seul modèle. Chaque modèle est généralement entrainé avec un sous-ensemble propre de l'ensemble d'entrainement et avec des paramètres initialisés différemment, ce qui leur permet de réagir différemment aux données. L'entrainement les conduit dans des régions différentes de l'espace des solutions, ce qui augmente les chances de découvrir le minimum global de la fonction de coût (Figure 9). L'inconvénient avec la technique d'ensachage est qu'elle est très coûteuse en termes de temps de calcul, surtout pour des modèles comportant de nombreux paramètres. L'entrainement d'un seul modèle peut déjà représenter une contrainte de temps, alors multiplier ce coût par x modèles peut constituer une barrière à l'emploi de la technique.

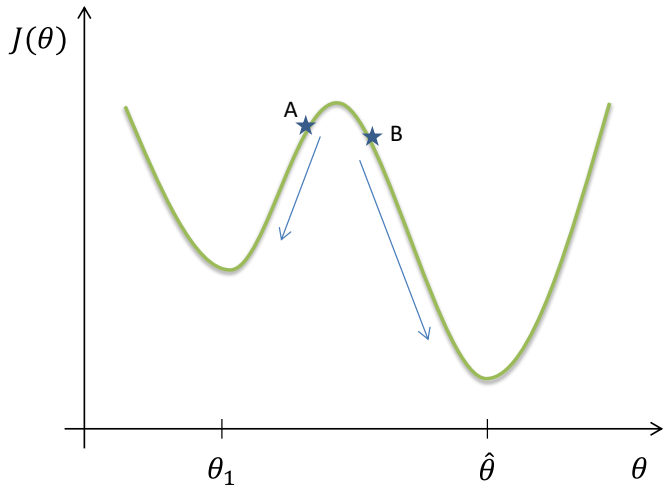


Figure 9. Illustration du principe de l'ensachage. Nous disposons de plusieurs modèles, dont A et B, qui vont converger vers des minima différents. Si nous n'avions que le modèle A, nous aurions convergé vers θ_1, mais puisque nous disposons aussi de B, nous pouvons converger vers l'ensemble optimal $\hat{\theta}$.

L'abandon (dropout) est une application de la technique d'ensachage, mais qui présente l'avantage de n'être que peu coûteuse en termes de temps de calcul. Elle est donc particulièrement importante pour les grands réseaux qui incluent plusieurs dizaines de milliers de paramètres. L'abandon consiste à abandonner aléatoirement certaines sections du réseau en cours d'entrainement, ce qui induit de nouvelles architectures à partir du réseau initial. Ceci revient à créer différents modèles comme l'ensachage et donc, encore une fois, à explorer différentes régions de l'espace des solutions.

La pénalité de norme est une expression mathématique $\Omega(\theta)$ ajoutée à la fonction de coût utilisée au cours de l'entrainement, proportionnelle au nombre de paramètres utilisés par le modèle. Ainsi plus le modèle a de paramètres non nuls, plus il est pénalisé, ce qui tend à favoriser les modèles plus simples moins spécifiques (Goodfellow et al. 2016) :

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\, \Omega(\theta)$$

Où α est un hyper paramètre, positif, qui indique l'importance que l'expérimentateur veut donner à la pénalité de norme.

Cette pénalité de norme peut prendre plusieurs formes :

- La régularisation L1 utilise la norme 1 des poids w du modèle :

$$\Omega(\theta) = \|w\|_1$$

- La régularisation L2 utilise la norme 2 des poids w du modèle :

$$\Omega(\theta) = \frac{1}{2}\|w\|_2^2$$

L'exemple de la régression polynomiale est un exemple qui illustre bien l'intérêt de la pénalité de norme dans l'entrainement des modèles (Figure 10). Si l'on utilise uniquement la fonction de coût, typiquement la moyenne des écarts au carré, plus le degré de la fonction polynomiale est grand, plus l'erreur commise est faible, jusqu'à arriver à des modèles très peu généralisables. L'introduction d'une pénalité de norme permet de favoriser les modèles qui utilisent moins de paramètres (les paramètres des polynômes de degrés élevés sont nuls), ce qui sélectionne les modèles suffisamment précis et généraux, même si ce ne sont pas les plus précis sur l'ensemble d'entrainement spécifiquement. Dans

le cas des réseaux de neurones, le principe est similaire : la pénalité de norme va favoriser les modèles dont les poids de certains neurones seront nuls de façon à disposer d'un modèle plus généralisable.

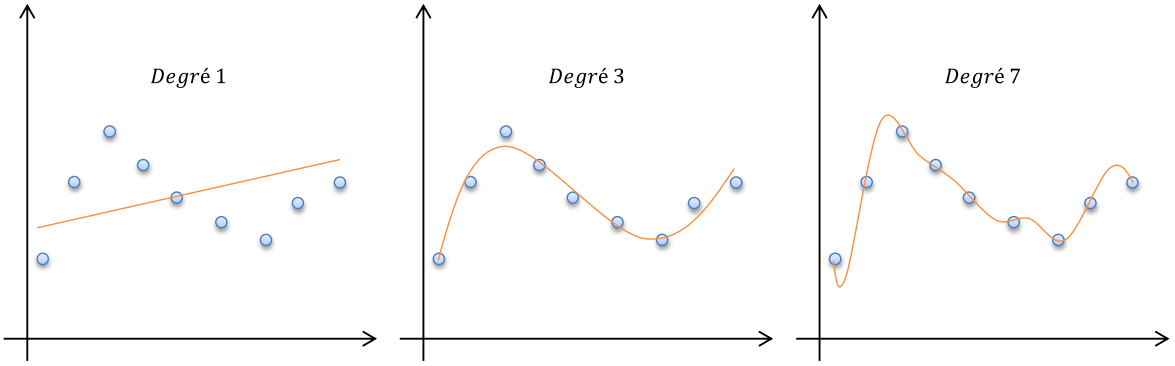


Figure 10. Illustration du principe de la pénalité de norme dans le cas d'une régression polynomiale.

1.3.6 L'analyse de données séquentielles

1.3.6.1 Les structures de décodages

a) *LES RESEAUX RECURRENTS*

Les réseaux de neurones récurrents (RNR ou RNN pour Recurrent Neural Network, parfois aussi noté RtNN pour les distinguer des réseaux de neurones récursifs RvNN pour Recursive Neural Network) permettent d'encoder une donnée d'entrée en y associant de l'information provenant des entrées la précédant et/ou la suivant dans la séquence. De manière générale, un vecteur d'entrée passe au travers d'une ou plusieurs couches cachées avant de produire un vecteur de sortie encodé. C'est au niveau de l'une ou de plusieurs de

ces couches cachées que se fait l'intégration de l'information provenant des entrées situées en amont et/ou en aval (Figure 11).

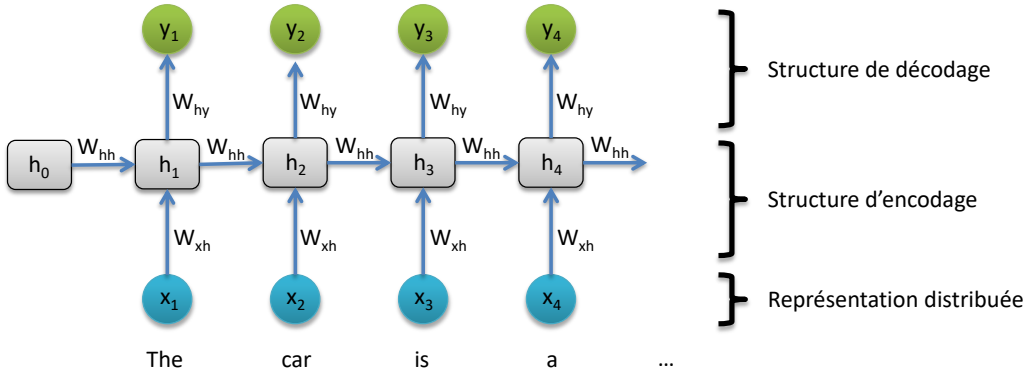


Figure 11. Illustration du principe d'un RNN au niveau duquel le vecteur caché h_t représentant l'entrée x_t tient compte à la fois de cette entrée x_t et du vecteur caché représentant la séquence en amont h_{t-1}.

Un RNN est composé de couches de cellules, généralement d'une à trois couches. Les cellules peuvent être de différentes sortes suivant la manière dont elles calculent le vecteur caché h_t.

Les cellules standards (Figure 12) calculent le vecteur caché $\mathrm{h_t}$ de la manière suivante :

$$\mathrm{h_t} = \tanh(\mathrm{W_{xh}x_t} + \mathrm{W_{hh}h_{t-1}})$$

Où W est un vecteur de poids incluant un paramètre de biais et x est le vecteur représentant un échantillon incluant un paramètre 1 pour tenir compte du biais.

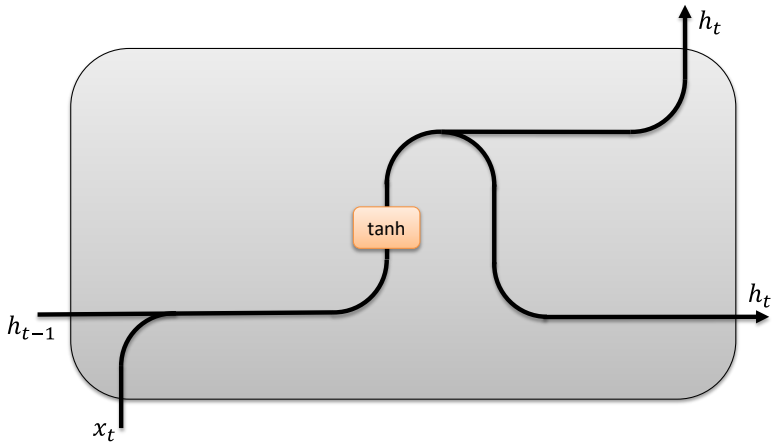


Figure 12. Illustration du fonctionnement d'une cellule standard d'un RNN.

Les cellules standards ne sont cependant viables que pour des pas de temps extrêmement courts. Au-delà d'un certain nombre de cellules, la rétro propagation de l'erreur lors de l'entrainement du réseau dans les nœuds en amont devient rapidement négligeable, ce qui limite la capacité d'apprentissage du réseau, c'est le problème de la disparition du gradient. Pour remédier à ce problème, deux autres types de cellules ont été développés : les cellules LSTM (Long-Short Term Memory – Hochreiter & Schmidhuber 1997) et les cellules GRU (Gated Recurrent Unit – Cho et al. 2014). Le calcul du vecteur caché se fait de manière plus complexe, mais permet de diffuser le gradient beaucoup plus efficacement dans le réseau. Dans le cas d'une cellule LSTM (Figure 13) le calcul du vecteur caché implique le calcul de quatre portes intermédiaires i_t, f_t, g_t, o_t et d'un état c_t de la manière suivante :

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1})$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1})$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1})$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1})$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

$$h_t = o_t * \tanh(c_t)$$

Où $*$ désigne le produit terme à terme ou produit matriciel de Hadamard.

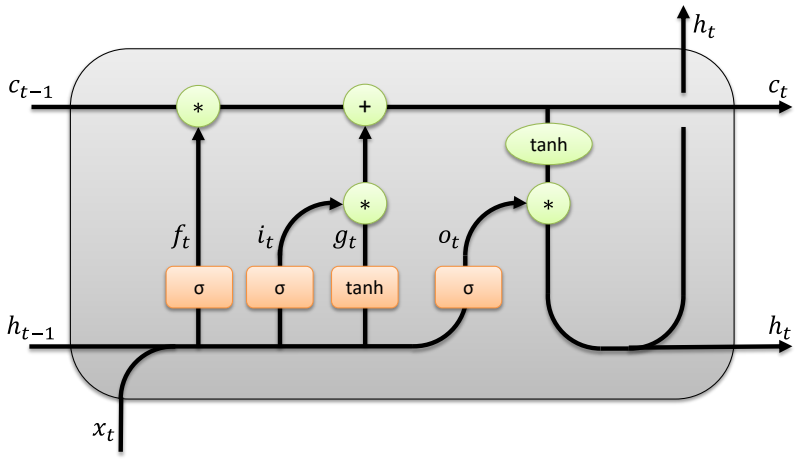


Figure 13. Illustration du fonctionnement d'une cellule LSTM d'un RNN.

Les cellules GRU (Figure 14) sont une version simplifiée et plus récente des cellules LSTM. Le calcul du vecteur caché n'implique le calcul que de trois portes intermédiaires r_t, z_t et n_t :

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1})$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1})$$

$$n_t = \tanh\big(W_{xn}x_t + r_t * (W_{hn}h_{t-1})\big)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * n_t$$

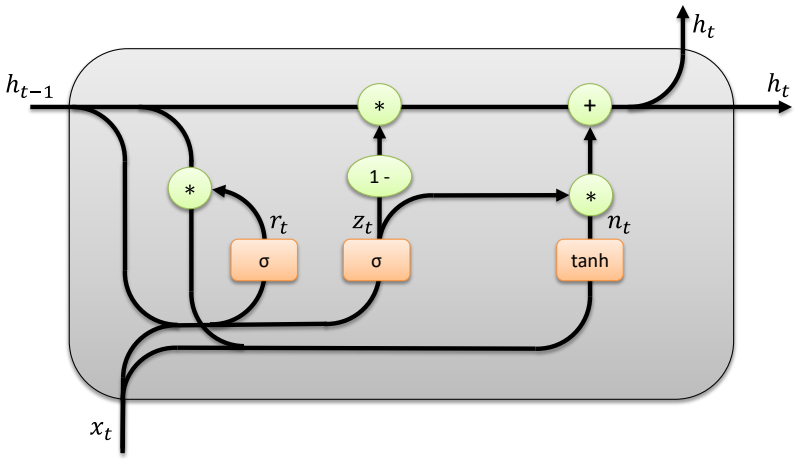


Figure 14. Illustration du fonctionnement d'une cellule GRU d'un RNN.

Les RNN illustrés dans les exemples précédents propagent le contexte de la séquence de gauche à droite, c'est-à-dire que le vecteur caché obtenu à partir du traitement de l'échantillon au temps $t-1$ est intégré dans le traitement de l'échantillon suivant au temps t, une architecture qui est qualifiée d'unidirectionnelle puisque le contexte ne se propage que dans un seul sens. Il est possible de conjuguer les deux sens de traitement en ajoutant une couche qui effectue le traitement dans le sens opposé, de droite à gauche. Pour le traitement de l'échantillon au temps t, nous obtenons ainsi deux vecteurs cachés, le vecteur $\overrightarrow{h_t}$ obtenu à partir de la couche gauche-droite et le vecteur $\overleftarrow{h_t}$ obtenu à partir de la couche

droite-gauche. Ces deux vecteurs sont ensuite groupés dans un vecteur caché final, généralement par concaténation. Dans ce cas, on parle d'un réseau bidirectionnel puisque le contexte est propagé dans les deux sens.

b) *LES RESEAUX RECURSIFS*

Les réseaux de neurones récursifs fonctionnent de manière similaire aux réseaux récurrents non plus sur une structure linéaire, mais sur une structure topologique en arbre. L'intérêt des réseaux récursifs est d'incorporer dans l'approche l'information soutenue par la structure interne de la séquence. Dans les réseaux récurrents, la séquence est parcourue de manière linéaire de gauche à droite et/ou de droite à gauche, ce qui ne tient pas compte de la structure interne de la séquence. Les réseaux récursifs au contraire utilisent cette structure pour organiser le parcours de transmission du contexte. Par exemple, dans le cas d'une phrase, les mots ne sont pas analysés dans l'ordre dans lequel ils sont lus, mais suivant l'arbre syntaxique de celle-ci. On ne parle plus de parcours droite-gauche et/ou gauche-droite, mais de parcours bas-haut et/ou haut-bas. Ceci suppose cependant de faire une analyse préliminaire de façon à découvrir une structuration interne des données (Li et al. 2017).

c) *LES RESEAUX CONVOLUTIFS*

Les réseaux de neurones convolutifs (RNC ou CNN pour Convolutional Neural Network) ne s'intéressent pas aux données de manière séquentielle, mais par zone. Les données sont regroupées en zones de taille n et passent au travers d'un certain nombre de

filtres organisés en couches successives de manière à produire une représentation encodée. Généralement les zones se chevauchent, ce qui permet à une donnée d'occuper différentes positions dans la zone (gauche, centre puis droite par exemple) et d'être considérée dans des contextes différents (Figure 15).

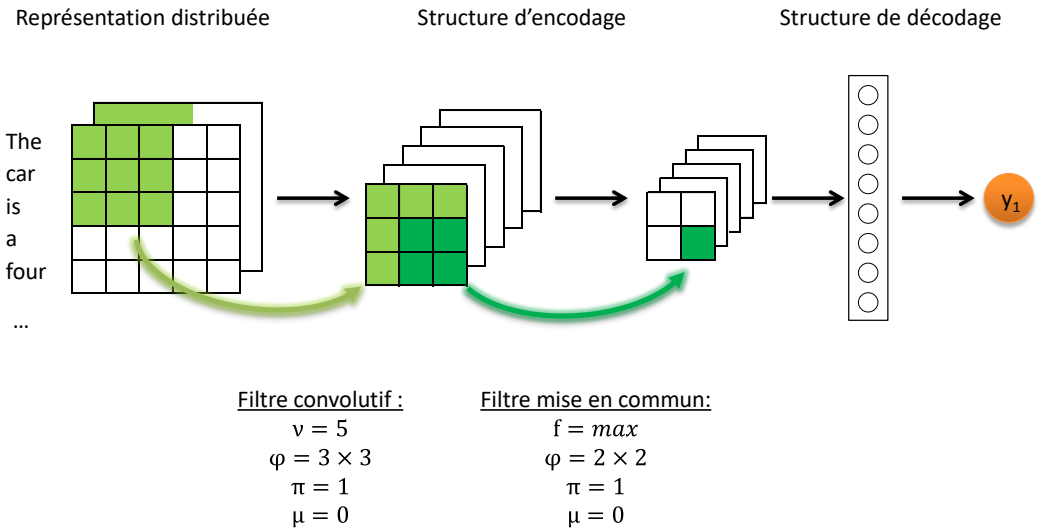


Figure 15. Illustration du fonctionnement d'un réseau convolutif dans le cas du traitement d'une phrase.

Les couches d'un réseau convolutif sont de deux sortes. Les couches de convolutions sont constituées de filtres qui transforment les données d'entrée. Il s'agit d'un ensemble de paramètres sous forme de tenseurs de poids qui sont appliqués par produit matriciel aux représentations des données et qui peuvent être modifiés au cours de l'entrainement de manière à adapter le réseau à la tâche à résoudre. Les couches de convolutions ont quatre hyper paramètres qu'il faut ajuster manuellement de manière à améliorer la performance : le nombre de filtres ν, la taille des filtres φ, le pas π et la marge μ. Le nombre de filtres détermine le volume du tenseur produit par la couche. La taille des filtres détermine la

surface couverte par le filtre sur la représentation entrante. Le pas détermine le chevauchement entre les différentes surfaces d'application d'un filtre. Finalement, la marge ajoute éventuellement une ou plusieurs séries de paramètres en bordure du tenseur d'entrée, ce qui permet d'ajuster la surface produite par l'application d'un filtre. Le premier hyper paramètre contrôle le volume du tenseur produit (typiquement pour un tenseur de trois dimensions, il contrôle la troisième dimension), tandis que les trois autres hyper paramètres en contrôle la surface (les deux premières dimensions).

Par exemple, si l'on prend les hyper paramètres suivants :

$$dimension\ des\ entrées : n_h^{t-1} \times n_l^{t-1} \times n_c^{t-1}$$

$$nombre\ de\ filtres :\ \nu$$

$$taille\ d'un filtre : \varphi$$

$$pas :\ \pi$$

$$marge :\ \mu$$

Alors,

$$dimension\ de\ la\ sortie : n_h^t \times n_l^t \times n_c^t$$

$$n_h^t = \left\lfloor \frac{n_h^{t-1} + 2\mu - \varphi}{\pi} + 1 \right\rfloor$$

$$n_l^t = \left\lfloor \frac{n_l^{t-1} + 2\mu - \varphi}{\pi} + 1 \right\rfloor$$

$$n_c^t = \nu$$

Les couches de mise en commun (pooling) fonctionnent de manière légèrement différente. Elles ne sont pas constituées de paramètres, mais appliquent une fonction, maximum ou moyenne en général, sur une zone particulière du tenseur résultant de la couche précédente. Ceci permet de sélectionner certaines caractéristiques ou d'harmoniser les valeurs à l'échelle de la région suivant la fonction choisie. Ces couches ont quatre hyper paramètres. La taille des filtres, le pas et la marge sont communs à la couche convolutive et fonctionnent de la même manière. Le quatrième hyper paramètre est constitué par la fonction d'activation choisie.

Par exemple, si l'on prend les hyper paramètres suivants :

$$dimension\ des\ entrée : n_h^{t-1} \times n_l^{t-1} \times n_c^{t-1}$$

$$fonction\ d'activation :\ f$$

$$taille\ d'unfiltre : \varphi$$

$$pas : \pi$$

$$marge : \mu$$

Alors,

$$dimension\ de\ la\ sortie : n_h^t \times n_l^t \times n_c^t$$

$$n_h^t = \left\lfloor \frac{n_h^{t-1} + 2\mu - \varphi}{\pi} + 1 \right\rfloor$$

$$n_l^t = \left\lfloor \frac{n_l^{t-1} + 2\mu - \varphi}{\pi} + 1 \right\rfloor$$

$$n_c^t = n_c^{t-1}$$

Réseaux de neurones récurrents, récursifs et convolutifs ne constituent pas des alternatives exclusives les unes des autres. Plusieurs études se servent d'une combinaison de ces réseaux afin de résoudre des tâches du traitement automatique des langues. Notamment, une des combinaisons couramment utilisées est l'association CNN-RNN dans laquelle un CNN est utilisé pour encoder les caractères d'un mot, encodage qui est ensuite concaténé aux représentations distribuées des mots pour produire une représentation étendue. Cette représentation entre dans un deuxième temps dans un RNN afin de résoudre la tâche (Chiu & Nichols 2016 par exemple).

1.3.6.2 Les structures de décodage

a) *LES RESEAUX DE NEURONES A PROPAGATION AVANT*

Les FFNN sont la structure de décodage la plus simple et une des plus couramment utilisées. Elle est composée généralement d'une seule couche de neurone et d'une fonction d'activation de type softmax pour les tâches de classification. La fonction softmax permet de prendre le score produit par le réseau pour toutes les sorties possibles, c'est-à-dire toutes les classes possibles, et de les replacer sur une échelle de 0 à 1. L'avantage de la fonction softmax est que la somme de tous les pointages vaut 1 et que la catégorie qui obtient le score le plus élevé produit par le réseau est celle qui a obtenu le pontage de softmax σ le plus élevé. Par conséquent, le pointage obtenu est aussi considéré comme la probabilité que la catégorie considérée soit effectivement la catégorie de l'échantillon. La fonction softmax est calculée de la manière suivante :

$$\sigma_i = \frac{e^{s_i}}{\sum_1^K e^{s_j}}$$

Où s_i désigne le score produit par le réseau pour la classe i.

b) *LES RESEAUX DE NEURONES RECURRENTS*

Les RNN peuvent jouer le rôle de couche de décodage de manière comparable à un FFNN, mais en intégrant une information contextuelle supplémentaire. La fonction

d'activation finale est généralement aussi une fonction softmax qui joue le même rôle que précédemment. Les cellules peuvent être des cellules standards, LSTM ou GRU.

c) *LE MECANISME D'ATTENTION*

Le mécanisme d'attention a été introduit par Bahdanau et al. (2014) dans le contexte de la traduction automatique. Le mécanisme vient en complément des structures d'encodage et de décodage pour spécifier l'importance (l'attention) relative qu'il faut accorder aux différents jetons d'entrée au moment du décodage de la séquence. Au moment de décoder un jeton particulier, un vecteur qui attribue un coefficient aux jetons de la séquence est produit. Ce vecteur est multiplié par les encodages de ces jetons de manière à produire un vecteur de contexte représentatif de l'influence de ces jetons sur le jeton à décoder :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Où c_i est le vecteur de contexte utilisé pour décoder le jeton i, α_{ij} est le coefficient d'importance ou d'attention accordée au jeton j de la séquence au moment du décodage du jeton i, h_j est l'encodage du jeton j et T_x est la longueur de la séquence.

Les coefficients d'attention α_{ij} sont des coefficients normalisés ; ils sont produits par une fonction de normalisation, en l'occurrence la fonction softmax à partir de scores relatifs à chaque couple de jetons :

$$\alpha_{ij} = sofmax(s_{ij}) = \frac{e^{s_{ij}}}{\sum_{k=1}^{T_x} e^{s_{ik}}}$$

Le score s_{ij} relatif à chaque couple ij est le vecteur de sortie d'un FFNN prenant en paramètre la concaténation de l'encodage du jeton j, et l'état du décodeur e_{j-1} au moment de décoder le jeton j :

$$s_{ij} = FFNN(e_{j-1}, h_j)$$

D'autres manières de calculer le score s_{ij} ont été proposées. Par exemple, Luong et al. (2015) propose d'utiliser plutôt le produit :

$$s_{ij} = e_{j-1}^T h_j$$

Vaswani et al. (2017) a généralisé le mécanisme d'attention en introduisant trois composantes (Figure 16) : la clé (key K), la valeur (value V) et la requête (query Q). Le vecteur d'attention est alors calculé avec la relation :

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{n}}\right)V$$

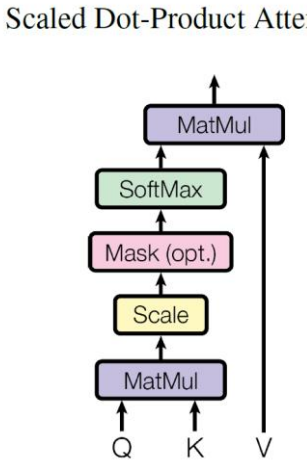


Figure 16. Illustration du mécanisme d'attention selon Vaswani et al. (2017).

Si l'on prend l'encodage du jeton h_j comme valeur V et comme clé K, l'état du décodeur e_{j-1} comme requête Q et que l'on prend 1 comme facteur d'échelle $\sqrt{n}$, la définition de l'attention de Vaswani et al. (2017) correspond à celle du contexte de Bahdanau et al. (2014) avec la variante de calcul apportée par Luong et al. (2015). De manière générale, ce modèle permet d'ouvrir le calcul et de prendre différents composants comme intrants. Par exemple, si l'encodage h_j est lui aussi pris comme valeur de requête Q, le modèle devient un mécanisme d'auto attention (self-attention) dans lequel la séquence traitée est prise à la fois comme entrée et comme base de comparaison, ce qui permet de déterminer l'importance relative des jetons d'une séquence et fournit un encodage pondéré de la séquence.

1.4 SYNTHESE

L’apprentissage automatique propose un ensemble de modèles adaptés au traitement de différents types de données pour résoudre différentes tâches. Parmi ces tâches se trouve l’analyse de données séquentielles. Le langage naturel est une structure de données séquentielle dans laquelle les mots, qu’ils soient écrits ou énoncés, constituent les jetons successifs. Il est donc possible de traiter le langage naturel avec l’apprentissage automatique et en particulier avec les modèles de l’apprentissage profond parmi lesquels les RNN semblent particulièrement bien adaptés.

CHAPITRE 2

LE TRAITEMENT AUTOMATIQUE DES LANGUES

L'objectif de l'étude est d'analyser des documents écrits en langage naturel. Ceci suppose de disposer de structures permettant de prendre en charges des données séquentielles comme nous l'avons vu au chapitre précédent, mais aussi de savoir comment aborder le langage naturel. Le langage naturel est une structure extrêmement complexe de par la variété des formes qu'il peut prendre. Exprimer une idée peut se faire de plusieurs façons différentes, et ce même dans le contexte d'un document écrit qui est pourtant une forme plus standardisée du langage que ne peut l'être le langage oral. Cette diversité est d'ailleurs la raison pour laquelle les machines éprouvent des difficultés avec les tâches relatives au traitement des langues. Il existe néanmoins des techniques et des outils pour aider à en appréhender cette complexité. Le deuxième chapitre est consacré à la présentation de ceux-ci.

2.1 LES CONCEPTS DU TRAITEMENT AUTOMATIQUE DES LANGUES

Le traitement automatique des langues est un ensemble de techniques de calcul employé pour analyser et représenter des textes naturels sur un ou plusieurs niveaux linguistiques de manière à réaliser un traitement du langage similaire à celui que ferait un humain dans le cadre d'une variété de tâches et d'applications (Liddy 2001).

2.1.1 Le modèle linguistique

Le modèle linguistique définit plusieurs niveaux au traitement automatique des langues (Liddy 2001) :

- Le niveau phonétique est l'interprétation des sons perçus. Ce niveau est l'étape préliminaire dans le cas du traitement d'un discours oral, mais ne s'applique pas dans le cadre d'un document écrit.

- Le niveau morphologique est la décomposition des mots. L'analyse morphologique a pour but de séparer les différents composants en ses différents morphèmes (composantes unitaires dotées d'un sens), typiquement le radical, les suffixes et les préfixes éventuels.

- Le niveau lexical est la signification des mots individuels. Plus précisément, l'analyse lexicale renseigne sur les caractéristiques intrinsèques des mots comme leur nature, leur genre, leur nombre.

- Le niveau syntaxique est la structuration de la phrase. Les mots ne sont plus traités individuellement, mais dans le contexte de la phrase dans laquelle ils apparaissent. Chaque mot occupe une fonction et l'ensemble confère une structure grammaticale à la phrase.

- Le niveau sémantique est la désambigüisation de la phrase. Les mots peuvent avoir différentes significations suivant le contexte dans lequel ils sont employés, l'analyse sémantique permet de déterminer le sens de la phrase en s'intéressant à l'interaction entre les sens des différents mots qui la composent.

- Le niveau discursif est la structuration du discours. Un discours est composé d'un ensemble de phrases et l'analyse discursive s'attache à établir des liens entre des composantes retrouvées dans différentes phrases du discours.

- Le niveau pragmatique est l'interprétation du discours. Il ne s'agit plus de déterminer les caractéristiques propres du discours, mais d'en élucider le contexte, les intentions.

Au fur et à mesure que l'on traverse les différents niveaux d'analyse linguistique, on remarque que le point focal s'élargit : on part du son dans l'analyse phonétique, pour passer à la composante du mot dans l'analyse morphologique, au mot dans l'analyse lexicale, à la phrase dans les analyses syntaxiques et sémantiques puis finalement au discours dans les analyses discursives et pragmatiques.

2.1.2 Les applications du traitement automatique des langues

Il existe une grande diversité de domaines d'application du traitement automatique des langues tels que l'extraction d'information, le résumé de document, la traduction,

l'association question-réponse, etc. (Liddy 2001). L'extraction d'informations est un domaine particulièrement vaste qui regroupe de nombreuses tâches comme l'étiquetage morphosyntaxique, l'extraction automatique de termes, la détection d'entités nommées, la résolution de coréférences, la classification de relations. Ces différentes tâches unitaires constituent des bacs à sable de la recherche fondamentale en apprentissage automatique, c'est-à-dire que beaucoup de recherches sont menées sur ces tâches prises individuellement et indépendamment les unes des autres de manière à faire avancer les connaissances sur les techniques de traitement automatique. Ces tâches peuvent avoir un intérêt pratique en elles-mêmes, par exemple l'extraction automatique de termes peut être utile pour constituer un glossaire. Néanmoins, l'extraction d'informations au niveau pratique et appliqué nécessite plutôt la résolution de plusieurs tâches de manière combinée. Extraire de l'information sur une compagnie à partir de pages Internet par exemple nécessite de détecter les occurrences de la compagnie dans les documents ce qui est en soi du ressort de deux tâches, la détection des entités nommées et la résolution de coréférences, et de caractériser les relations qui lie cette organisation avec d'autres organisations ou des individus qui peuvent en constituer le personnel, ce qui est du ressort de la classification de relations.

Les tâches individuelles correspondent à un angle d'analyse particulier d'un document écrit ou oral. Elles peuvent donc être associées à un niveau linguistique particulier. L'étiquetage morphosyntaxique, qui consiste à attribuer une étiquette aux mots de la phrase désignant par exemple leur nature, est un exemple d'une tâche d'extraction d'information appartenant au niveau syntaxique. L'analyse automatique approfondie d'un document écrit en langage naturel nécessite, comme le ferait un humain, de combiner le

traitement du document sur les différents niveaux linguistiques et donc au niveau de différentes tâches.

2.1.3 Les techniques de traitement

Les techniques de traitement sont les techniques utilisées pour résoudre les différentes tâches décrites précédemment. Ces techniques peuvent être regroupées en trois grands ensembles. Les approches symbolique et statistique sont les deux approches qui ont été utilisées en premier et sont donc considérées comme les approches traditionnelles du traitement automatique des langues (Liddy 2001). L'approche symbolique repose sur l'analyse des phénomènes linguistiques et leur traduction dans des algorithmes structurés d'extraction. Il peut par exemple s'agir d'un ensemble de règles permettant d'extraire les caractéristiques correspondant aux différents niveaux d'analyse linguistique précédemment décrits. L'approche statistique repose sur l'utilisation d'outils mathématiques de description de la composition de textes. De par sa nature, cette approche induit le recours à une quantité de données importantes. Avec le développement de l'apprentissage automatique et de l'apprentissage profond s'est développée une troisième voie, l'approche par apprentissage, qui a supplanté progressivement les deux autres approches du fait de la qualité des résultats qu'elle apporte.

2.1.3.1 L'approche symbolique

L'approche symbolique est caractérisée par l'utilisation de règles qui traduisent une connaissance à priori du langage et de son comportement en relation avec la résolution

d'une tâche. Ces règles sont composées en amont de la phase de traitement et traduisent l'idée que l'on se fait de la manière dont le langage doit se comporter pour exprimer une idée, une relation. Elles sont ensuite traduites en langage informatique pour composer un ensemble de filtres au travers duquel passe la séquence de mots d'un document. Les séquences qui correspondent à une règle sont ainsi filtrées au fur et à mesure, ce qui permet d'identifier les expressions d'intérêt et de résoudre la tâche. Pour illustrer le principe de fonctionnement de l'approche, prenons l'exemple de la tâche de reconnaissance des entités nommées dont voici quelques règles simples d'identification :

- Si un mot commence par une majuscule et qu'il n'est pas précédé d'un point, alors il constitue une entité nommée.

- Si un mot commençant par une majuscule est précédé du mot « Mr » ou du mot « Mme », alors il s'agit d'une entité nommée de type Personne.

- Si un mot ou un groupe de mots commençant par une majuscule précède le mot « Inc. », alors l'ensemble constitue une entité nommée de type Organisation.

Nous pouvons aussi prendre l'exemple de la tâche de classification de relations :

- Si deux noms communs sont séparés par la préposition « de », alors le premier nom est une partie de l'ensemble désigné par le second.

- Si une entité nommée de type Personne est impliquée dans une structure syntaxique quelconque avec une entité nommée de type Organisation, alors l'entité Personne est un(e) employé(e) de l'entité Organisation.

De nombreuses règles se basent sur la structure syntaxique des phrases d'un document. Ceci implique d'avoir recours à une phase de prétraitement qui fait l'analyse syntaxique et induit donc de construire un analyseur syntaxique, ce qui représente un coût de développement non négligeable, ou d'avoir recours à un outil externe, ce qui induit une dépendance à un outil tierce. Néanmoins, le travail réalisé par les analyseurs syntaxiques externes est fiable et ne constitue pas une faiblesse de l'approche.

Le principe de l'approche est simple et facile à mettre en œuvre avec une liste d'expressions régulières ou de conditions successives. Son inconvénient est d'être extrêmement général. Certaines règles ont une validité presque absolue, par exemple la règle impliquant les mots « Mr » et « Mme », mais ce cas est rare. La plupart du temps, les règles employées « fonctionnent » seulement dans la plupart des cas ou pire seulement dans certains cas, tout dépendant de la qualité de la règle. Ceci a pour effet de générer beaucoup de bruits lorsque la règle filtre de faux positifs ou beaucoup de silences lorsque la règle laisse passer des faux négatifs. D'autre part, la qualité de résolution de la tâche est extrêmement dépendante de la couverture offerte par l'ensemble de règles utilisé. Couvrir entièrement et proprement une tâche nécessite un grand nombre de règles très précises et une connaissance approfondie de la langue, ce qui implique un travail et donc un investissement de temps important en amont du traitement proprement dit.

2.1.3.2 L’approche statistique

L’approche statistique utilise un ensemble de métriques statistiques pour identifier des mots, des expressions d’importance ou des relations particulières. Dans la grande majorité des cas, ces métriques sont des métriques de distribution, absolue ou relative, des mots ou des expressions dans les documents ou les corpus d’entrainement. Une métrique courante est la fréquence d’un mot (Kageura & Umino 1996, Pazienza et al. 2005). Si l’on prend une tâche comme l’extraction automatique de termes en vue de la constitution d’un glossaire d’affaires, la fréquence absolue des mots ou des expressions est un exemple de métrique simple, mais utile puisqu’elle dresse la liste des termes fréquemment utilisés et donc susceptibles d’être des termes d’intérêt. Cependant, cette métrique présente aussi l’inconvénient d’être peu discriminante : quel que soit le document, les prépositions, les articles et autres mots de liaisons sont les mots les plus fréquents et donc ceux qui sont extraits prioritairement. Ceci engendre beaucoup de bruit et une faible précision (mais un rappel fort). Pour aller plus loin, il est possible d’utiliser la fréquence relative des termes, c’est-à-dire la différence de fréquence entre des documents. La fréquence relative est, du point de vue du bruit, une donnée plus précise : un mot ou une expression qui a une fréquence plus importante dans un document particulier que dans le reste du corpus est susceptible d’être caractéristique du domaine d’affaires de ce document. Elle permet de discriminer les prépositions et les déterminants, dont la fréquence relative est supposée constante, des mots ou expressions d’intérêts qui sont répétés à plusieurs reprises dans certains documents au moins pour les définir et en donner les caractéristiques.

L'approche statistique et l'approche symbolique ne sont pas incompatibles et il est donc logique qu'elles aient été utilisées en complément l'une de l'autre pour améliorer la performance de résolution des tâches (Pazienza et al. 2005). Dans ce cas, les deux approches sont utilisées en série : l'approche symbolique est utilisée en premier pour faire une première sélection des termes puis l'approche statistique est utilisée pour discriminer les candidats potentiels. Si l'on reprend l'exemple décrit dans le paragraphe précédent, la mise en place de filtres syntaxiques adéquats de premier niveau permet d'éviter de considérer les prépositions ou les déterminants, ce qui améliore la précision des outils statistiques utilisés au second niveau.

2.1.3.3 L'approche par apprentissage

L'approche par apprentissage consiste à utiliser les méthodes de l'apprentissage automatique et de l'apprentissage profond pour résoudre les tâches du traitement automatique des langues. Cette approche a connu un essor important avec le développement des modèles d'apprentissage profond et particulièrement des réseaux de neurones.

La majorité des tâches du traitement automatique des langues sont des tâches prédictives et non pas exploratoires, c'est-à-dire qu'on souhaite que le modèle produise un résultat attendu. Par conséquent, l'apprentissage supervisé est le mode privilégié. D'autre part, les réseaux de neurones offrent un certain nombre de structures particulièrement bien adaptées au traitement des langues dont les données d'entrées (les lettres ou les mots suivant le niveau d'analyse) sont séquentielles. Ainsi les couches convolutives, récursives

et surtout récurrentes permettent de tenir compte de son contexte séquentiel à la fois en amont et en aval, c'est-à-dire qu'elles permettent de tenir compte, au moment du traitement d'une donnée, des autres données situées avant et après elle. Chaque donnée est donc considérée en fonction de ses caractéristiques propres, absolues, mais aussi en fonction de ses caractéristiques relatives (la manière dont elle est utilisée) dans le contexte du document en cours d'analyse. Dans les approches par apprentissage, les caractéristiques absolues sont traduites dans une représentation utilisée pour désigner la donnée d'entrée dans le modèle. Les caractéristiques relatives sont traduites dans une deuxième représentation cachée qui est composée en tenant compte de la représentation absolue de l'entrée en cours de traitement et de la représentation relative issue du traitement des données antérieures dans la séquence. Ainsi, les représentations relatives tiennent compte de l'ensemble des mots qui a déjà été analysé et constitue une représentation du contexte.

Les modèles de l'apprentissage automatique et de l'apprentissage profond sont des modèles mathématiques. L'information véhiculée par les lettres ou les mots d'un document est donc encodée dans une représentation vectorielle composée de valeurs réelles. L'encodage le plus simple est l'encodage « point chaud » (« one hot »). Dans cette représentation, chaque mot est encodé par un vecteur de dimension V où V est la taille du vocabulaire employé, c'est-à-dire le nombre de mots distincts qui composent les documents ou un dictionnaire de référence. Ce vecteur n'est composé que de 0 sauf à l'indice correspondant à la position du mot dans le dictionnaire (Figure 17). Cette représentation ne fournit que très peu d'information sur le mot et a été depuis remplacée par des représentations plus informatives.

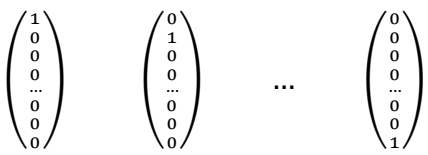


Figure 17. Illustration de la représentation « point chaud ».

L'ajout d'informations sur les mots s'est fait dans un premier temps dans le cadre du document ou du corpus en cours d'analyse. Pour ce faire deux techniques existent. La première reprend les méthodes des approches symboliques et statistiques en composant un vecteur dont les valeurs sont les résultats obtenus par le mot à encoder sur un ensemble de tests issus de ces deux approches (Figure 18). La seconde utilise la technique de plongement, c'est-à-dire que chaque mot est associé à une représentation composée de valeurs réelles qui est une partie intégrante du modèle et qui est donc modifiée au cours de l'apprentissage (par rétro propagation par exemple) pour participer à l'amélioration du modèle.

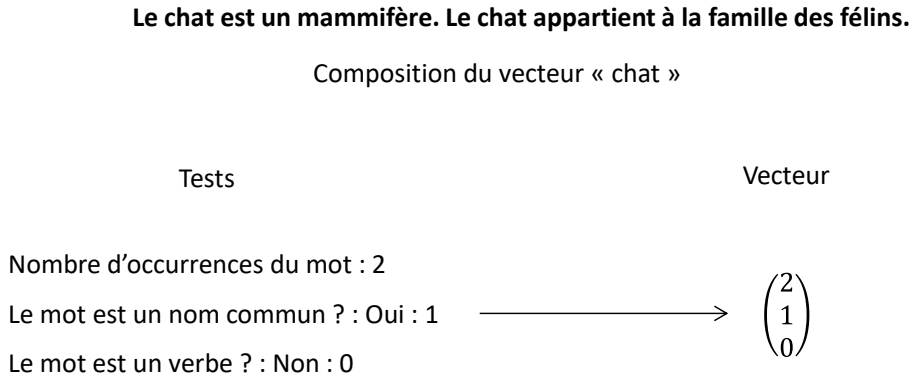


Figure 18. Illustration du principe de composition d'une représentation vectorielle à partir des approches symboliques et statistiques.

L'ajout d'informations sur les mots s'est fait dans un deuxième temps dans le cadre de corpus de documents beaucoup plus vastes, en dehors du cadre de réalisation d'une tâche spécifique du traitement automatique des langues et destiné à fournir une connaissance générale sur le fonctionnement du langage utilisable par la suite dans ces tâches de traitement. Cette connaissance intrinsèque du langage est représentée par les modèles de langues et participe à une nouvelle démarche de l'apprentissage automatique, l'apprentissage par transfert. Avec son développement, la résolution des tâches du traitement automatique des langues a pu s'extraire de sa dépendance à une analyse préliminaire longue et fastidieuse du langage nécessaire à l'utilisation des approches symboliques et statistiques et dans une certaine mesure à la représentativité des ensembles d'entrainement.

2.2 LES MODELES DE LANGUES

Plusieurs directions ont été explorées afin de modéliser le comportement d'une langue et ainsi fournir une connaissance de base sur celle-ci exploitable pour les tâches de traitement des langues, mais tous les modèles de langues sont des modèles probabilistes.

2.2.1 Les n-grammes

Un n-gramme est une séquence de n jetons qui sont soit des mots soit des lettres suivant le sujet d'étude. Le modèle n-gramme cherche à déterminer la probabilité d'apparition d'une séquence de jetons $(w_1, w_2, \dots, w_{l-1}, w_l)$. De manière formelle et en appliquant la chaine de probabilités, celle-ci s'écrit :

$$P(w_1, w_2, \dots, w_{l-1}, w_l) = P(w_1) \times P(w_2|w_1) \times \dots \times P(w_l|w_1, \dots, w_{l-1})$$

L'estimation d'une probabilité d'apparition $P(w_l|w_1, \dots, w_{l-1})$ est délicate du fait du faible nombre d'occurrences potentielles d'une telle séquence dans le langage. Pour contrecarrer cette difficulté, le modèle n-gramme utilise l'hypothèse simplificatrice apportée par le principe des chaines de Markov. Les chaines de Markov sont une approximation qui stipule que l'état d'un système au temps ou en position t est déterminé par les quelques états qui le précèdent. S'il est déterminé par l'état précédent, on parle de chaines de niveau 1, s'il est déterminé par les $n-1$ états qui le précèdent, on parle de chaines de niveau $n-1$. De ce fait, les n jetons successifs d'une séquence constituent une unité d'évaluation et la probabilité d'apparition d'un jeton à une position quelconque d'une séquence est uniquement déterminée par les $n-1$ jetons qui le précèdent :

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P\left(w_i\middle|w_{i-(n-1)}, w_{i-(n-2)}, \dots, w_{i-1}\right)$$

Si $n = 1$ (modèle unigramme) la probabilité d'apparition d'un jeton ne dépend d'aucun antécédent et correspond dans ce cas à sa probabilité d'apparition dans le langage :

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P(w_i)$$

$$P(w_1, w_2, \dots, w_{l-1}, w_l) = \prod_{k=1}^{l} P(w_k)$$

Si $n = 2$ (modèle bigramme), la probabilité d'apparition d'un jeton dépend de sa probabilité d'apparition dans le langage dans le contexte où il suit le jeton qui est apparu dans la position précédente :

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P(w_i|w_{i-1})$$

$$P(w_1, w_2, \dots, w_{l-1}, w_l) = P(w_1) \times \prod_{k=2}^{l} P(w_k|w_{k-1})$$

Le modèle ne s'étend généralement pas au-delà de $n = 3$ (modèle trigramme). L'entrainement se fait en dénombrant le nombre de cas d'apparition de chaque combinaison de jetons sur de grands ensembles d'entrainement et en compilant les probabilités qui en découlent. Dans un modèle bigramme, si l'on note $N(w_{i-1}w_i)$ le nombre d'occurrences du bigramme $w_{i-1}w_i$ dans le corpus et $\sum_w N(w_{i-1}w)$ le nombre d'occurrences du jeton w_{i-1}, peu importe le jeton suivant, alors

$$P(w_i|w_{i-1}) = \frac{N(w_{i-1}w_i)}{\sum_w N(w_{i-1}w)}$$

Le modèle n-grammes est utile dans le cadre de tâche de détection de séquences comme la reconnaissance vocale, ou de génération de séquences comme la traduction. Les inconvénients du modèle sont 1° sa faible portée de contextualisation, c'est-à-dire que le contexte d'utilisation d'un jeton n'est prise en compte que sur le pas de temps des quelques jetons qui composent le n-gramme avec lui, et 2° sa dépendance à l'ensemble

d'entrainement et qui implique que si un n-gramme n'est pas observé, sa probabilité d'apparition devient nulle.

2.2.2 Les plongements

Les plongements (embeddings en anglais) sont un modèle d'encodage du contexte d'utilisation d'un mot dans une langue développé à partir de réseaux de neurones. Concrètement il s'agit d'une signature du mot sous forme de représentation vectorielle composée de nombres réels. Les valeurs n'ont pas de signification particulière propre, mais puisqu'elles encodent le contexte d'utilisation du mot, deux mots qui ont l'habitude d'être employés dans des contextes similaires se retrouvent assez proches dans l'espace vectoriel des plongements et des associations sémantiques particulières entre mots se retrouvent à avoir des comportements similaires (Figure 19).

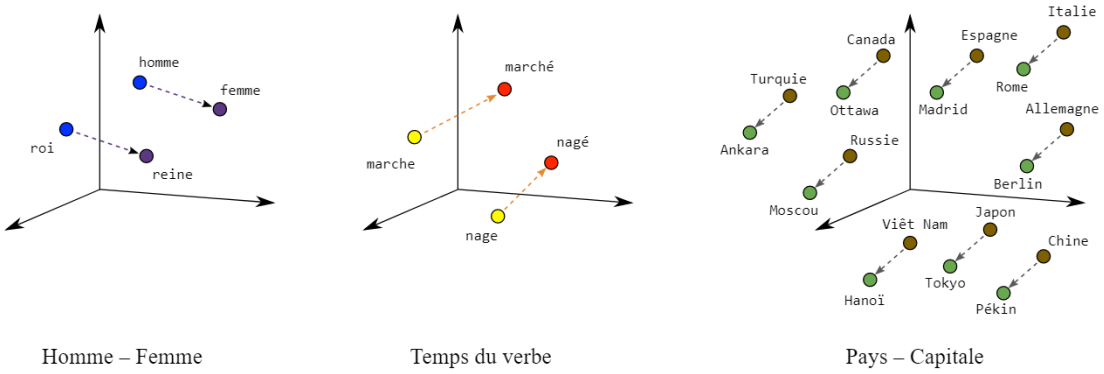


Figure 19. Illustration du comportement des représentations des mots dans un sous-espace réduit des plongements (source : https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space).

Les techniques utilisées pour obtenir ces plongements sont diverses : FFNN, RNN, approche supervisée ou non supervisée. Il existe d'ailleurs plusieurs bibliothèques de tels

plongements, mais les deux plus utilisées sont word2vec (Mikolov et al. 2013) et GloVe (Pennington et al. 2014). Ces bibliothèques ont plusieurs caractéristiques importantes qui les rendent très utiles pour le traitement automatique des langues. Premièrement, elles couvrent l'ensemble du vocabulaire commun d'une langue, ce qui permet de disposer d'un encodage pour la très grande majorité des mots trouvés dans un document. Les quelques exceptions sont en général composées de noms propres, de termes techniques ou de mots étrangers. Celles-ci peuvent tout de même être gérées par les bibliothèques grâce à l'existence d'un vecteur correspondant au mot inconnu (<UNK> pour Unknown) et qui fournit un encodage pour ces exceptions. Deuxièmement, elles sont entrainées sur des corpus de document extrêmement vastes de plusieurs milliers de documents, ce qui leur permet de tenir compte d'une grande panoplie d'utilisation d'un mot. Il existe en général plusieurs encodages pour une même bibliothèque variant sur la taille de l'ensemble d'entrainement utilisé et la dimension des représentations. Les encodages de plus grande dimension et obtenus sur les corpus les plus grands fournissent des représentations plus efficaces, puisqu'ils tiennent compte d'une plus grande variabilité. Cependant, des représentations de plus grandes dimensions sont aussi plus exigeantes en termes de capacité de calcul. Il s'agit donc de faire un choix entre des représentations plus complètes, mais plus lourdes et des données moins complètes, mais plus légères. À titre d'exemple la bibliothèque principale de GloVe a été entrainée sur un corpus de six milliards de jetons, elle encode 400000 mots du vocabulaire anglais et fournit des représentations de dimensions de 50, 100, 200 et 300 valeurs. Il existe d'autres bibliothèques GloVe entrainées sur des corpus encore plus grands de 42 milliards et 840 milliards de jetons (https://nlp.stanford.edu/projects/glove/).

Les plongements représentent une solution ingénieuse pour encoder de manière détournée de l'information générale sur le mot et d'introduire cette information sous forme vectorielle pour une utilisation dans le cadre de modèles mathématiques d'apprentissage automatique et en particulier dans le cadre des réseaux de neurones. L'inconvénient du modèle est que l'encodage est constant pour un mot donné, quel que soit son contexte d'utilisation ou même sa signification dans le document en cours d'analyse. Par exemple, le mot « identité » dans les phrases « L'élève a dû montrer sa carte d'identité. » et « L'élève a dû apprendre les identités remarquables. », n'a pas le même sens. Pourtant la bibliothèque ne contient qu'un vecteur d'encodage pour le mot (et même si elle en contenait plusieurs, comment pourrait-elle choisir lequel appliquer ?). Bien que cet encodage tienne compte de nombreux exemples d'utilisation du mot, il ne peut pas être parfaitement adapté à ces deux exemples.

2.2.3 Le modèle BERT

Le modèle BERT (Bidirectional Encoder Representations from Encoder) va plus loin que les plongements décrits précédemment, en ne fournissant pas simplement un dictionnaire de représentations vectorielles, mais directement une couche de réseau préentrainé (elle-même composée de plusieurs sous-couches) qui doit produire dynamiquement les plongements des mots, adaptés au contexte du document en cours d'analyse.

Le modèle présente plusieurs similitudes avec les plongements. Premièrement, il permet de traduire sous forme vectorielle un ensemble d'informations linguistiques de

manière à ce que le mot soit intégré dans la chaine de traitement de réseaux de neurones. Deuxièmement, il est entrainé sur des corpus de documents extrêmement grands (800 millions de mots et 2,5 milliards de mots – Devlin et al. 2019), ce qui assure une bonne représentativité des contextes d'utilisation des mots. Troisièmement, le modèle existe en plusieurs versions variant sur le nombre de paramètres utilisés. Comme dans le cas des plongements, plus les dimensions sont grandes, plus le modèle est précis, mais plus il est exigeant en temps de calcul (Devlin et al. 2019). Quatrièmement, le modèle entrainé est rendu disponible dans des bibliothèques directement utilisables comme point de départ pour la résolution de tâches du traitement automatique des langues (https://huggingface.co/transformers).

Le modèle reprend l'architecture d'encodeur-décodeur utilisée dans les approches non supervisées ou dans les tâches de génération de séquences (par exemple la tâche de traduction qui, à partir d'une séquence d'entrée, doit produire une séquence de sortie). Les réseaux de type encodeurs-décodeurs fonctionnent en deux temps : la partie encodeur produit un encodage de la séquence d'entrée sous forme de représentation vectorielle, encodage qui est utilisé ensuite par la partie décodeur pour générer la séquence de sortie. Chacune des deux parties est composée de réseaux de neurones composés de modules récursifs, récurrents ou convolutifs pour le traitement des données séquentielles, auxquels sont greffés des modules auxiliaires, des modules d'attention, pour la synchronisation des données. Un module d'attention peut aussi être utilisé à la jonction entre la partie encodeur et la partie décodeur. Le modèle BERT est particulier puisqu'il repose sur une architecture encodeur-décodeur composée exclusivement de modules d'attention appelée

« Transformateur » (Vaswani et al. 2017). Il est entrainé de manière non supervisée sur deux tâches du traitement automatique des langues que sont la prédiction d'un mot dans une séquence et la détection de la contigüité de deux séquences (Devlin et al. 2019).

Le module de décodage est un module adapté à la résolution d'une tâche particulière. Il n'est utile au modèle BERT que dans le contexte de son entrainement. Ce n'est donc que la partie encodeur qui est utilisée dans le contexte de l'apprentissage par transfert et qui est rendue disponible dans les bibliothèques publiques. BERT constitue la couche d'encodage d'un réseau, située en amont d'une couche de décodage adaptée à la tâche que l'on souhaite réaliser et qui varie d'une étude à l'autre (voir les couches de décodages possibles au CHAPITRE 1).

2.2.4 Le modèle GPT

Le modèle GPT (Generative Pre-trained Transformer) s'inscrit dans la lignée évolutive des modèles de langues. Tout comme ses prédécesseurs, GPT est entrainé de manière non supervisée sur un grand nombre de documents de façon à produire un modèle général de langue par-dessus lequel des tâches plus spécifiques du traitement automatique des langues sont conduites. L'architecture d'encodage reprend elle aussi la solution des « transformateurs » apportée par Vaswani et al. (2017) en une structure multicouche de modules d'attention (Figure 20). Le modèle a été décliné en trois versions successives, GPT-1 (Radford et al. 2018), GPT-2 (Radford et al. 2019) et GPT-3 (Brown et al. 2020) qui améliorent progressivement les performances du modèle en augmentant à la fois la taille

des ensembles d'entrainement et surtout la complexité du modèle (117 millions de paramètres pour GPT-1, 1,5 milliard pour GPT-2 et 175 milliards pour GPT-3).

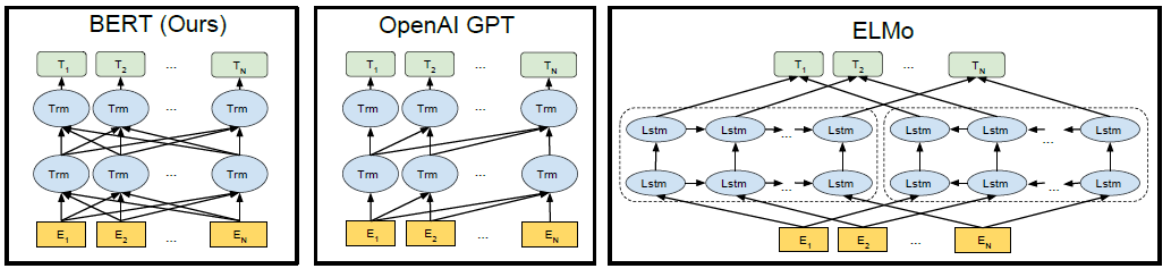


Figure 20. Illustration des architectures des modèles BERT et GPT[2] selon Devlin et al. 2019.

Tout comme les autres modèles de langues, GPT a été rendu disponible dans une bibliothèque permettant d'obtenir l'encodage des mots à partir d'un texte (https://huggingface.co/transformers).

2.3 SYNTHESE

Le langage naturel est un environnement extrêmement complexe. Les tentatives pour organiser, quantifier, structurer sa composition sont nombreuses, que ce soit par des méthodes linguistiques ou statistiques. Cette connaissance peut être utilisée directement pour résoudre certaines tâches du traitement automatique des langues, mais l'émergence récente de techniques d'analyse par apprentissage et des modèles de langues qui en ont

[2] ELMo (Peters et al. 2018) est un autre modèle de langue, tout comme UlmFit (Howard & Ruder 2018). ELMo et UlmFit sont restés quelques semaines/mois des modèles intéressants pour les tâches du traitement des langues avant d'être remplacés définitivement par BERT et GPT plus performants.

résulté permet d'envisager l'utilisation de représentations plus complexes du langage pour résoudre ces tâches.

CHAPITRE 3

L'ÉTAT DE L'ART

La découverte d'un modèle à partir d'un cahier des charges écrit en langage naturel est une tâche qui a déjà été tentée par des techniques empiriques de traitement du langage. Ces techniques reposent sur la mise au point d'un certain nombre de règles extraites de la connaissance du langage naturel. Nous présentons dans ce chapitre les techniques utilisées à ce jour et les résultats obtenus pour déterminer si l'objectif est réalisé ou si on peut faire mieux. La découverte de modèles est aussi une tâche qui, à notre connaissance, n'a encore jamais été traitée par l'apprentissage profond. Il n'existe pas encore de standard quant à sa réalisation, la voie reste à paver. Nous proposons une technique qui mélange les résultats de plusieurs tâches existantes de l'apprentissage profond : la détection d'entités pour découvrir les concepts importants du cahier des charges, la résolution de coréférences pour faire le lien entre toutes les mentions d'un même concept et la classification de relations pour déterminer le lien qui existe entre ces concepts. Ce chapitre est également consacré à présenter les travaux qui ont déjà été réalisés dans ces domaines et les différents modèles envisagés en vue de la construction de notre propre modèle.

3.1 L'EXTRACTION DE MODELES

3.1.1 Les principes de la tâche

L'extraction automatique de modèles est une tâche du traitement automatique des langues qui appartient au niveau discursif. Elle consiste à découvrir de manière automatique le modèle qui sous-tend un cahier des charges rédigé en langage naturel. La nature du modèle reste à la discrétion des besoins de chacun, il peut s'agir d'un modèle indépendant de plateforme ou d'un modèle dépendant de plateforme. L'avantage du modèle indépendant de plateforme est qu'il ne préfigure aucun détail de la future implémentation et qu'il reste donc une traduction directe du cahier des charges. De plus la production d'un modèle dépendant de plateforme à partir d'un modèle indépendant de plateforme, de même que la génération de code à partir d'un modèle dépendant de plateforme sont des étapes automatisables. La production automatique du modèle indépendant de plateforme est donc la dernière étape avant l'automatisation complète du cycle de production d'une application à partir d'un cahier des charges. C'est aussi la plus complexe puisque c'est la seule qui fait intervenir le langage naturel.

Les études qui se sont intéressées à l'extraction automatique de modèles ont abordé le problème d'un point de vue linguistique en utilisant une approche symbolique. Le processus de résolution de la tâche utilise avant tout les caractéristiques du langage pour extraire les différents concepts, même si des outils issus de l'apprentissage automatique sont employés, notamment des analyseurs syntaxiques. La difficulté avec cette approche est d'appréhender la complexité du langage naturel et la possibilité d'exprimer de plusieurs

façons des idées similaires. Pour limiter cette difficulté, plusieurs études ont fait le choix de limiter la variabilité offerte par le langage naturel en imposant certains formalismes.

La première limitation est le recours à un modèle de cas d'utilisation pour exprimer les exigences du système plutôt qu'à un cahier des charges (Liu 2003, Yue et al. 2013, Thakur & Gupta 2016). Ces modèles comportent une description du comportement du système écrit en langage naturel, c'est pourquoi ils sont inclus dans la tâche d'extraction automatique de modèles, mais ils constituent une simplification du travail en comparaison d'autres études qui prennent comme point de départ un cahier des charges dans lequel les concepts sont décrits en texte libre (Harmain & Gaizauskas 2000, Popescu et al. 2008, Joshi & Deshpande 2012, Arora et al. 2016). L'avantage du modèle de cas d'utilisation est d'isoler certains concepts comme les acteurs du système. De plus, la description de la tâche de manière isolée induit implicitement un style plus simple et plus direct que ne peut le faire un texte libre. La syntaxe est donc éventuellement plus concise et plus répétitive, ce qui facilite l'utilisation des techniques de l'approche symbolique. Le passage par un modèle de cas d'utilisation est cependant contraignant dans un cas réel parce qu'il suppose que la personne qui rédige les exigences du système (le client) maitrise un outil de développement. En outre, il ne correspond pas au principe de fonctionnement classique de l'industrie qui est de rédiger un cahier des charges en langage naturel.

La seconde limitation est l'imposition de règles syntaxiques simplifiées. La simplification de la syntaxe poursuit le même objectif que la première limitation : simplifier et uniformiser le langage de manière à rendre les outils linguistiques plus efficaces. Cette simplification peut prendre plusieurs formes : l'emploi de syntaxes simples de type sujet,

verbe, complément (Popescu et al. 2008, Thakur & Gupta 2016), l'utilisation systématique de la voie active (Popescu et al. 2008), l'utilisation de propositions subordonnées simples (Popescu et al. 2008), la restriction de l'emploi de pronoms (Thakur & Gupta 2016), etc. Certaines études font le pari de n'imposer aucune restriction au langage utilisé (Harmain & Gaizauskas 2000, Mich & Garigliano 2002, Ilieva & Ormandjeva 2006, Mohd & Rodina 2010, Bhagat et al. 2012, Arora et al. 2016), mais il arrive que toutes les phrases ne puissent être traitées correctement soit parce que leur syntaxe est trop complexe (Arora et al. 2016), soit parce qu'elles sont trop longues (Mich & Garigliano 2002). On doit alors se poser la question de la couverture du modèle généré par rapport aux exigences formulées.

Certaines études font l'emploi de modules complémentaires permettant de clarifier soit les termes du domaine d'affaires soit les liens unissant des concepts variés (Yue et al. 2011). La première catégorie est composée essentiellement des glossaires (Liu 2003). La seconde est composée de modèles qui résument des connaissances générales préexistantes sur le domaine d'affaires (Harmain & Gaizauskas 2000, Mich & Garigliano 2002, Mohd & Rodina 2010, Joshi & Deshpande 2012). Éventuellement ces modèles peuvent être complétés par les informations extraites de l'étude courante de manière à ce qu'ils deviennent plus pertinents lors d'études ultérieures. Là encore, l'objectif de tels modules est de faciliter l'étape de traitement automatique ou semi-automatique en apportant une couche d'informations supplémentaires. Les études plus anciennes sont davantage concernées par l'utilisation de ces modules tandis que les études plus récentes essaient d'extraire l'information de manière plus naïve et plus reproductible d'un domaine d'affaires

à un autre en s'attachant uniquement à l'information contenue dans le document (Bhagat et al. 2012, Thakur & Gupta 2016, Arora et al. 2016).

Le modèle final généré est le plus souvent un diagramme de classes (Mohd & Rodina 2010, Bhagat et al. 2012, Yue et al. 2013, Joshi & Deshpande 2012, Thakur & Gupta 2016, Arora et al. 2016). Certaines études produisent une combinaison de documents, par exemple un modèle de domaine et un diagramme d'activités (Ilieva & Ormandjieva 2006) ou un diagramme de robustesse, un diagramme de communication et un diagramme de classes (Liu 2003).

3.1.2 La résolution de la tâche

Un document rédigé en langage naturel est difficile à aborder pour une machine. Pour en faciliter le traitement, un ensemble d'opérations de prétraitement est réalisé. Ces opérations décomposent d'abord le document en unités de traitement, le mot généralement (analyse morphologique/lexicale), elles analysent ensuite le contexte dans lequel est placé le mot, c'est–à-dire la phrase (analyse syntaxique, analyse sémantique) et, finalement, elles replacent celui-ci dans son environnement large, le document (analyse de discours). Ces opérations sont réalisées de manière chronologique, le résultat d'une étape servant de point de départ pour l'étape suivante, si bien que le texte écrit en langage naturel disparait progressivement au profit de représentations plus schématiques et standardisées. En ce sens, elles participent à la formalisation progressive des exigences permettant de réduire la variabilité du langage naturel et de la transposer en un ensemble de structures limitées.

Le Tableau 1 présente les différentes démarches utilisées dans les études citées dans ce document.

Tableau 1. Présentation des différentes approches citées dans ce document.

Étude	Support	Modèle	Limitations	Production
Harmain & Gaizauskas 2000	Cahier des charges	Analyse lexicale - syntaxiques - sémantique - discours + règles	Modèles pré existants	Diagramme de classe
Mich & Garigliano 2002	Cahier des charges	Analyse lexicale - syntaxiques - sémantique + règles	Modèles pré existants	Diagramme de classe
Liu 2003	Modèle de cas d'utilisation	Analyse lexicale - syntaxiques + règles	Syntaxe simplifiée - Glossaire	Diagrammes multiples
Ilieva & Ormandjeva 2006	Cahier des charges	Analyse lexicale - syntaxiques - sémantique - discours + règles	Aucune	Diagrammes multiples
Popescu et al. 2008	Cahier des charges	Analyse lexicale - syntaxiques + règles	Syntaxe simplifiée	Diagramme de classe
Mohd & Rodina 2010	Cahier des charges	Analyse lexicale + règles	Modèles pré existants	Diagramme de classe
Bhagat et al. 2012	Cahier des charges	Analyse lexicale - syntaxiques - sémantique + règles	Aucune	Diagramme de classe
Joshi & Deshpande 2012	Cahier des charges	Analyse lexicale - syntaxiques + règles	Modèles pré existants	Diagramme de classe
Yue et al. 2013	Modèle de cas d'utilisation	Analyse lexicale - syntaxiques - sémantique + règles	Syntaxe simplifiée	Diagramme de classe
Arora et al. 2016	Cahier des charges	Analyse lexicale - syntaxiques - sémantique + règles	Aucune	Diagramme de classe
Thakur & Gupta 2016	Modèle de cas d'utilisation	Analyse lexicale - syntaxiques - sémantique + règles	Syntaxe simplifiée	Diagramme de classe

3.1.2.1 Les phases de prétraitement

L'analyse lexicale est le point d'entrée du traitement. On la retrouve donc dans toute analyse du langage naturel. Son objectif est de séparer les phrases les unes des autres pour les traiter individuellement comme des unités d'analyse séparées, individualiser les différents mots du document, en identifier la forme de base s'ils sont conjugués, contractés ou au pluriel et finalement d'en donner la nature (nom, verbe, adjectif, adverbe, etc.). Le document est ainsi décomposé en blocs d'analyses (les phrases) et chaque bloc est décomposé en briques unitaires (les mots) qui constituent les éléments de base des traitements ultérieurs.

L’analyse syntaxique est aussi répandue que l’analyse lexicale. Son objectif est d’attribuer une fonction aux mots (sujet, verbe, complément d’objet direct, complément d’objet indirect, attribut, épithète, déterminant, etc.) de façon à dégager leur organisation les uns par rapport aux autres. Les mots sont ensuite regroupés en groupes fonctionnels s’ils forment une entité cohérente comme un groupe nominal composé par exemple d’un nom, de son adjectif épithète et de son article déterminant. La réalisation de ces deux étapes est confiée à des modules spécialisés. Il en existe plusieurs, par exemple le Stanford Parser (https://nlp.stanford.edu/software/srparser.html). Les résultats des analyses lexicales et syntaxiques peuvent être schématisés dans des structures particulières (Figure 21) : un arbre syntaxique (Harmain & Gaizauskas 2000, Liu 2003, Bhagat et al. 2012, Arora et al. 2016) et/ou une liste de dépendances (Popescu et al. 2008, Thakur & Gupta 2016, Arora et al. 2016).

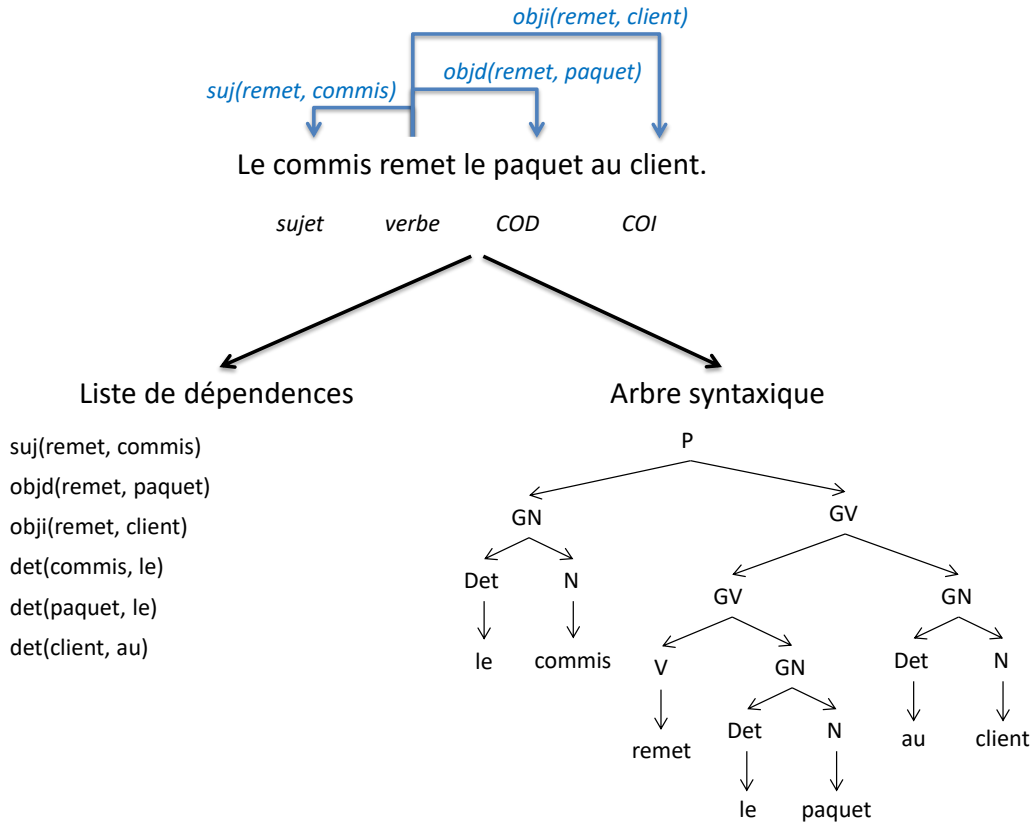


Figure 21. Illustration de l'aboutissement des analyses lexicales et sémantiques (inspiré de Thakur & Gupta 2016).

L'analyse sémantique est, lorsqu'elle a lieu, la troisième étape de la phase de prétraitement. Elle prend comme point de départ les résultats de l'analyse syntaxique menée au niveau des mots individuels ou des groupes de mots pour constituer l'architecture fonctionnelle de la phrase. La phrase est décomposée en un ensemble de relations de dépendances liant les groupes fonctionnels. La fonction des ensembles n'est plus envisagée au niveau des mots individuels, mais au niveau des groupes cohérents qui composent la phase. On se retrouve à parler de groupes sujet, de groupes complément d'objet par exemple, organisés autour d'un groupe verbal qui est le centre du prédicat liant ces éléments. Il s'agit véritablement de savoir qui fait quoi et comment. À l'inverse des étapes

précédentes, cette étape ne fait pas appel à un module externe, mais constitue un apport original des différentes études. L'extraction des dépendances entre groupes fonctionnels est réalisée par la mise en œuvre d'un ensemble de règles et d'heuristiques simples (Harmain & Gaizauskas 2000) ou plus complexes (Ilieva & Ormandjieva 2006, Thakur & Gupta 2016, Arora et al. 2016), Ces règles et heuristiques sont appliquées sur les unités extraites de l'analyse syntaxique et sont propres à chaque étude. Les résultats sont formalisés de différentes façons suivant les études. Cela peut être une représentation tabulaire des relations entre les composants fonctionnels (sujet, prédicat, objet) (Ilieva & Ormandjieva 2006), une représentation graphique schématique des relations entre les mots ou les groupes de mots (Ilieva & Ormandjieva 2006) ou des listes de relations (Harmain & Gaizauskas 2000, Arora et al. 2016) ou de structures de phrases (Thakur & Gupta 2016).

L'analyse de discours est l'étape de prétraitement la plus avancée. L'échelle d'analyse change de manière à synthétiser l'information disponible sur les concepts au niveau de l'ensemble du texte (Harmain & Gaizauskas 2000, Ilieva & Ormandjieva 2006). Ceci suppose de pouvoir faire le lien entre les différentes mentions d'un même concept, ce qui est l'objet d'une tâche spécifique du traitement automatique des langues qui est la résolution de coréférences telle que décrite dans les sections suivantes. Il s'agit d'une tâche difficile à résoudre notamment sans employer des méthodes de l'apprentissage profond et elle est donc peu souvent entreprise. Malgré tout, certaines études se sont attardées à rassembler l'information disponible tout au long du texte, ne serait-ce qu'en rapprochant les mots ou les expressions identiques qui apparaissent à différents moments et à produire des réseaux sémantiques globaux (Morgan et al. 1995, Ilieva & Ormandjieva 2006) ou des

modèles de discours (Harmain & Gaizauskas 2000) qui sont une représentation graphique schématique de l'ensemble des relations existantes entre les différents concepts (Figure 22).

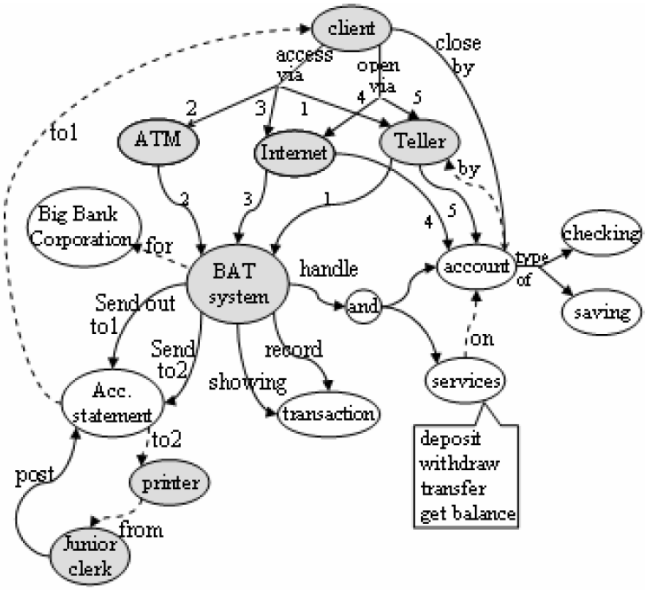


Figure 22. Illustration d'un réseau sémantique selon Ilieva & Ormandjieva (2006).

3.1.2.2 La production du modèle

L'approche de traitement privilégié est l'approche symbolique : un ensemble de règles prédéfinies est appliqué sur les résultats issus des phases de prétraitement et dicte la manière dont les structures linguistiques sont traduites en concepts et en relations. Les règles sont élaborées à priori; elles sont donc le reflet de ce que les auteurs considèrent comme la manière générale d'exprimer une relation entre des concepts. Dans le cas d'un diagramme de classes, qui est le PIM le plus courant, il s'agit de déterminer quelles expressions doivent devenir des classes, des attributs, des opérations, quelles relations doivent devenir des associations simples, des généralisations, des compositions ou des

agrégations. La précision des règles est l'élément clé qui permet de produire un PIM complet.

Dans les cheminements les plus courts, les règles sont appliquées sur les résultats de l'analyse lexicale (Mohd & Rodina 2010) ou de l'analyse syntaxique (Liu 2003, Popescu et al. 2008, Joshi & Deshpande 2012). Il s'agit de règles assez directes basées sur des structures simples comme sujet-verbe-complément, si-alors, génitif, etc. Dans les cheminements intermédiaires, les règles sont appliquées sur les résultats de l'analyse sémantique (Bhagat et al. 2012, Yue et al. 2013, Thakur & Gupta 2016, Arora et al. 2016). Ces modèles présentent un niveau d'extraction d'information plus complet (Figure 23). Les règles qui s'appliquent prennent en compte des structures de phrases plus complexes. Généralement, dans ces deux premiers cheminements, une étape de validation finale doit avoir lieu pour raffiner le modèle et supprimer les éventuelles redondances. Ce raffinement est effectué à la main (Harmain & Gaizauskas 2000, Popescu et al. 2008) ou de manière automatique avec un autre jeu de règle (Liu 2003). Finalement dans les cheminements les plus longs, les règles permettent essentiellement de raffiner et de transformer le modèle produit par l'analyse de discours en un modèle de domaine (Ilieva et Ormandjieva 2006).

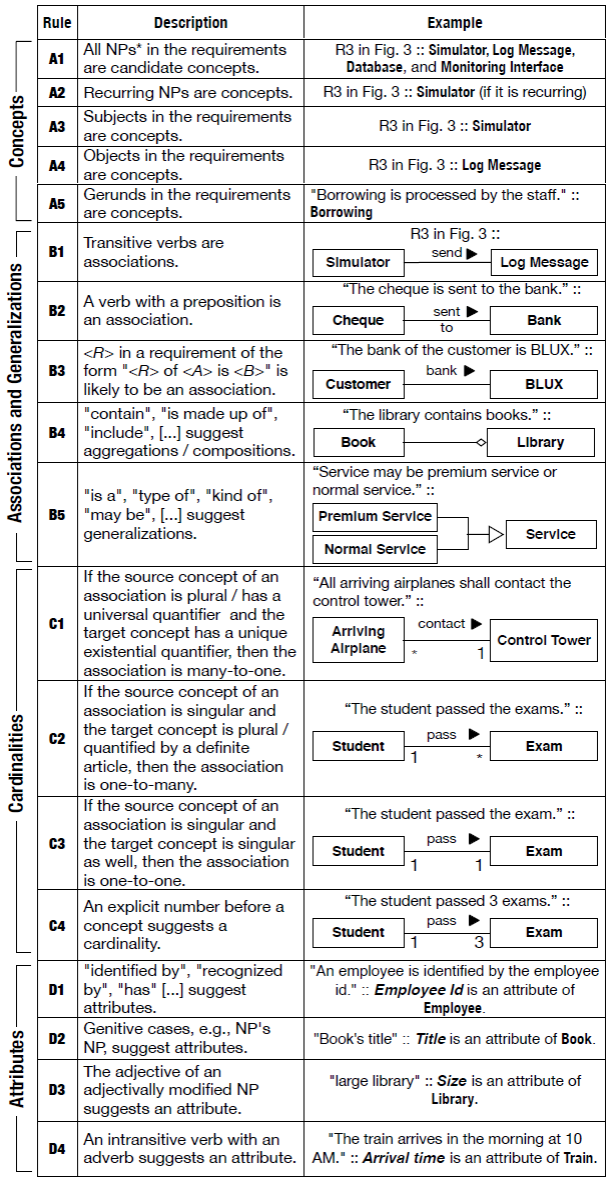


Figure 23. Illustration de règles utilisées pour l'extraction de modèles (d'après Arora et al. 2016).

3.2 LES TACHES PERTINENTES DU TRAITEMENT AUTOMATIQUE DES LANGUES

L'objectif de l'étude est de résoudre la tâche d'extraction de modèles en utilisant les outils de l'apprentissage profond. Si l'on décompose la tâche en activités unitaires, il s'agit d'extraire toutes les occurrences des différents concepts d'un document (les futurs classes,

attributs, opérations, multiplicités) et les relations qui existent entre ces concepts (lesquels sont des classes, lesquels sont des attributs et de quelle classe, quelles classes sont associées, quelles classes sont des agrégats de quelles classes, etc.). Ces activités unitaires recoupent des tâches existantes du traitement automatique des langues. Cette section décrit ces tâches.

3.2.1 L'extraction automatique de termes

3.2.1.1 Les principes de la tâche

L'extraction automatique de termes (Automatic Terminology Extraction ATE) et la reconnaissance d'entités nommées (Entity Detection and Tracking EDT) décrites dans la section suivante sont deux tâches dont les objectifs sont similaires : extraire/détecter des entités. Cependant, la différence de nature des entités entre les deux tâches impose des traitements différents. L'extraction automatique de termes a pour objet des noms communs qui ont une importance relative suivant le domaine d'affaires considéré, importance qui dépend de deux facteurs : le document dans lequel il apparait, c'est-à-dire l'information que ce document transmet, et l'angle avec lequel le document est envisagé, c'est-à-dire l'information que le lecteur veut en extraire. Par exemple, le mot « clé » peut représenter une notion pertinente dans un document traitant de mobilier et sans importance dans un document traitant d'informatique. Mais si le lecteur du document d'informatique s'intéresse aux modes de chiffrement de l'information, alors le mot « clé » peut retrouver une importance certaine. La détection d'entités nommées, quant à elle, a pour objet des noms propres qui ont une signification absolue dans le monde dans lequel nous vivons. Québec,

Jacques Cartier ou le château Frontenac ont une signification précise et ont le statut d'entités nommées, quel que soit le document dans lequel elles apparaissent. Bien que notre étude soit plus proche conceptuellement de l'extraction automatique de termes, la reconnaissance d'entités nommées a développé des techniques de traitement qui s'avèrent utiles, quel que soit l'objet d'études. C'est pour cela qu'elle est également présentée.

La notion de terme manque d'une définition précise pour la caractériser sans ambigüité (Pazienza et al. 2005). Vu et al. (2008) définissent un terme comme « un mot ou une expression qui véhicule un sens spécial », Kageura & Umino (1996) comme « une unité linguistique caractéristique d'un domaine d'affaires ». Même si cette deuxième définition précise ce qu'est un terme dans l'absolu, elle laisse aussi entrevoir une certaine subjectivité, car établir qu'un mot ou une expression est caractéristique d'un domaine d'affaires dépend en partie de la perception de chaque personne. Malgré cette difficulté, il est possible d'en donner des caractéristiques fonctionnelles en introduisant deux propriétés : la cohérence (unithood) et la représentativité (termhood – Kageura et Umino 1996). La cohérence est une caractéristique linguistique. Elle relève de la structure syntagmatique d'un terme et correspond à sa capacité à former une expression structurée et significative. Un mot constitue forcément un ensemble cohérent, la notion s'intéresse donc uniquement à la structuration d'une expression composée de plusieurs mots. L'expression « satellite géostationnaire » par exemple forme un ensemble plus cohérent que l'expression « module en orbite stationnaire autour de la Terre ». La représentativité est quant à elle une caractéristique fonctionnelle et correspond à la capacité d'un mot ou d'une expression à être caractéristique du contenu d'un document ou d'un domaine d'affaires (Kageura &

Umino 1996). L'expression « satellite géostationnaire » est possiblement un terme plus représentatif d'un cahier des charges d'une agence spatiale que ne peut l'être le mot « végétal ».

L'extraction automatique de termes est une tâche dont les objets d'études dépendent beaucoup du contexte (domaine d'affaires et perception du document). Il est donc difficile de développer un modèle universel et un jeu de données générique qui puissent permettre l'entrainement d'outils spécifiques. Si l'on ajoute à cela la diversité des langues impliquées, les études ont tendance à se focaliser sur des jeux de données spécifiques et qui par conséquent ne sont pas adoptés par d'autres études. Un jeu de données a néanmoins été développé pour la tâche et le domaine d'affaires spécifique de l'informatique, le ACL RD-TEC (Zadeh & Handschuh 2014), grâce auquel il est possible d'avoir une base de comparaison entre les études. Un autre jeu de données, GENIA (Ohta et al. 2002), a aussi été utilisé dans le cadre de l'extraction automatique de termes (Yuan et al. 2017) même s'il est aussi utilisé pour la tâche de reconnaissance des entités nommées, ce qui soulève la dualité des termes biologiques et médicaux qui sont à la fois des entités nommées et des termes du domaine d'affaires.

De manière générale, la tâche consiste à statuer sur le fait qu'un candidat est un terme ou pas. Il s'agit donc d'attribuer la classe « terme » ou « non-terme » aux candidats. En ce sens, il s'agit d'une tâche de classification binaire dans laquelle les candidats sont soit des termes qui reçoivent l'étiquette « terme » (vrais positifs), soit des termes qui reçoivent l'étiquette « non-terme » (faux négatifs), soit des non-termes qui reçoivent l'étiquette « terme » (faux positifs), soit des non-termes qui reçoivent l'étiquette « non-terme » (vrais

négatifs). Les métriques utilisées pour démontrer la performance d'un modèle sont donc naturellement la précision, le rappel et la mesure F_1 (voir CHAPITRE 1).

3.2.1.2 La résolution de la tâche

Les approches symboliques s'intéressent essentiellement à la cohérence des termes (Kageura & Umino 1996, Pazienza et al. 2005) et sont constituées de la succession de deux étapes : une étape d'analyse syntaxique automatique qui décompose la phrase en unités lexicales et attribue à chacune une étiquette identifiant sa nature (nom, verbe, adjectif, article, etc.) puis une étape d'application de règles préétablies (Pazienza et al. 2005). L'apport principal des études est constitué par les règles employées et leur composition afin de résoudre la tâche. La technique la plus simple est de disposer d'un corpus de termes de référence listant les mots d'importance et d'extraire du document les termes qui ont une correspondance (Zadeh & Handschuh 2014). Ceci suppose de disposer en amont de tel corpus de référence, ce qui est justement l'objectif de la tâche. Par conséquent l'approche ne présente un intérêt que lorsque l'on dispose d'un corpus de référence général que l'on souhaite raffiner pour extraire les termes d'importance dans un ensemble de documents précis. De manière plus générale, les techniques symboliques s'intéressent à tous les ensembles de mots de longueur variable et cherchent à y reconnaitre une caractéristique soit pour les sélectionner soit pour les éliminer. On voit apparaitre deux stratégies : les stratégies qui partent d'une liste vide et qui sélectionnent au fur et à mesure de l'analyse du document des candidats et les stratégies qui partent d'une liste complète comprenant tous les ensembles de mots possibles du document d'une longueur de n mots au plus et qui les éliminent au fur et à mesure qu'une règle n'est pas respectée. Ces règles sont soit 1° des

règles syntaxiques précisément identifiées comme désirées pour sélectionner une expression ou non désirées pour exclure des candidats et qui se présentent sous la forme de grammaires ou d'expressions régulières, soit 2° des listes de mots marqueurs pour sélectionner une expression ou de mots rejets pour exclure des candidats (Kageura & Umino 1996, Pazienza et al. 2005, Zadeh & Handschuh 2014). Par exemple, des règles syntaxiques telles que l'enchainement « Nom-Nom », « Nom-Adjectif » en français ou « Adjectif-Nom » en anglais sont des exemples de structures souhaitées. À l'inverse, les expressions composées d'articles indéfinis ou de pronoms, qui sont des types de mots jugés trop vagues pour faire partie d'un terme d'intérêt, sont exclues (Simon 2018). Des structures parallèles telles que les structures « Nom1-Nom2 » et « Nom2 of Nom1 » en anglais qui désignent souvent la même chose sont aussi utilisées pour reconnaitre de nouvelles expressions d'intérêt à partir d'expressions déjà identifiées.

Les approches statistiques s'intéressent aussi bien à la cohérence qu'à la représentativité des termes (Kageura & Umino 1996, Pazienza et al. 2005, Vu et al. 2008). La cohérence des termes se mesure en analysant la cooccurrence des termes dans un document ou un corpus. L'idée est que les mots qui apparaissent plus souvent associés que ce que ne peut le laisser présager une distribution indépendante des termes sont susceptibles de former une expression cohérente. La représentativité des termes se mesure en regardant la fréquence d'apparition absolue ou relative des termes dans un document ou un corpus. La fréquence absolue, par définition, identifie les termes fréquents dans un document qu'il est donc logique de considérer comme important. La fréquence relative identifie les termes qui ont une fréquence plus importante dans un document donné que dans un corpus

d’autres documents. Ces termes ne sont pas forcément les plus utilisés, mais leur importance relative suggère qu’ils ont une plus grande importance dans le document considéré et qu’ils peuvent donc correspondre à des termes clés du domaine d’affaires.

Les approches symboliques et statistiques ont été utilisées de manière complémentaire pour former une approche hybride (Pazienza et al. 2005, Cram & Daille 2016). L’approche hybride est en fait une cascade des deux techniques. Les techniques symboliques font une première sélection suivant la structure des termes; elles réduisent le champ des possibles en excluant tous les constituants du document qui ne peuvent pas être des termes. Les techniques statistiques fournissent dans un deuxième temps des indices pour sélectionner de manière définitive les termes les plus cohérents et les plus représentatifs parmi les candidats obtenus lors de la première étape.

Les techniques heuristiques décrites précédemment se sont attardées à développer un ensemble d’indicateurs symboliques et statistiques permettant de statuer sur l’importance de termes. Les approches par apprentissage vont dans un premier temps utiliser ces indices et les rassembler dans une représentation vectorielle des mots afin qu’elle soit analysée par des modèles d’apprentissage. Dès lors, la tâche devient une tâche de classification où chaque terme potentiel se voit attribuer une des deux classes « terme » ou « non-terme ». Les indices utilisés sont identiques ou dérivés de ceux déjà cités : booléen indiquant si le mot est un nom, un adjectif, un verbe, un nom propre, un nom commun; booléen indiquant si le mot fait partie d’une phrase à la voix active; nombre de noms communs que la racine du mot a engendré dans le document, nombre d’adjectifs qu’elle a engendré, nombre de verbes qu’elle a engendré; fréquence du mot dans le document; fréquence moyenne du mot

dans le corpus; nombre de documents qui contiennent le mot dans le corpus; rapport entre la fréquence du mot dans le document et sa fréquence dans le corpus (Conrado et al. 2013, Yuan et al. 2017). La tâche étant une tâche de classification binaire, elle permet d'utiliser à la fois des modèles de discrimination binaires et les modèles multi-classes avec deux classes. On retrouve ainsi les arbres de décision et les forêts d'arbres décisionnels, les modèles bayésiens naïfs, les machines à vecteurs de support et les modèles de régression logistique (Conrado et al. 2013, Yuan et al. 2017).

L'extraction automatique de termes est une tâche du traitement automatique des langues, au même titre que les autres tâches décrites dans les sections suivantes. À ce titre, elle doit analyser la structure d'un document décomposé en phrases qui sont elles-mêmes constituées d'une séquence de jetons. Le développement des réseaux de neurones et particulièrement des architectures d'encodage de structures séquentielles a ouvert de nouvelles voies d'analyse pour ces tâches. L'extraction automatique de termes a donc vu émerger l'utilisation de structures d'encodage de types récurrentes et convolutives (Wang et al. 2016a). Puisque la tâche est une tâche de classification, les couches de décodage de type softmax ou régression logistiques sont utilisées pour déterminer la catégorie la plus probable d'un mot ou d'une expression (Wang et al. 2016a).

La résolution de la tâche a ensuite bénéficié de l'avènement des plongements. L'inconvénient de la représentation sous forme de vecteurs de caractéristiques est la nécessité de faire toutes les démarches préparatoires à la constitution des vecteurs d'entrée de l'apprentissage automatique. Comme dans les approches traditionnelles, elles nécessitent une analyse approfondie de la structure du langage et des particularités linguistiques du

domaine d'affaires auquel on s'intéresse pour déterminer les caractéristiques utiles à inclure dans la préparation des entrées (Conrado et al. 2013); en ce sens l'approche devient dépendante du domaine d'affaires dans lequel elle est utilisée (Amjadian 2019). Elle oblige également à tester l'utilité des caractéristiques, la manière de les encoder numériquement. Toute cette démarche d'ingénierie nécessite une certaine connaissance du langage et des outils statistiques, mais est surtout très exigeante en termes de temps de préparation. Les plongements sont l'évolution de la représentation sous forme de vecteurs de caractéristiques destinée à éviter le recours à la phase d'ingénierie préparatoire de l'approche précédente (Wang et al. 2016a, Amjadian 2019).

Le Tableau 2 résume les différentes approches concernant l'extraction automatique de termes.

Tableau 2. Résumé des approches concernant l'extraction automatique de termes. Dans la section Modèles, les parenthèses servent à délimiter les différentes couches du modèle, typiquement la structure utilisée pour représenter les jetons, la couche d'encodage et la couche de décodage.

Étude	Modèle	Jeu de données	Score F1
Vu et al. 2008	Règles linguistiques + règles statistiques	s.o.	s.o.
Conrado et al. 2013	Règles linguistiques + règles statistiques + classificateurs automatiques	s.o.	s.o.
Cram & Daille 2016	Règles linguistiques + règles statistiques	s.o.	s.o.
Wang et al. 2016a	(Plongements) + (RNN/CNN) + (FFNN)	ACL RD-TEC GENIA	69,2 70,7
Yuan et al. 2017	Règles linguistiques + règles statistiques + classificateurs automatiques	ACL RD-TEC GENIA	86 82
Simon 2018	Règles linguistiques	s.o.	s.o.

3.2.1.3 Les limites de la tâche

Un point important de la tâche est le caractère relatif de l'importance d'un terme dans un document. À l'inverse des entités nommées qui ont un aspect universel, il est difficile de statuer, même de manière conceptuelle, sur ce qu'est un terme d'intérêt (Zadeh & Hansdschuh 2014). De plus, chaque type de document, chaque domaine d'affaires possède ses propres particularités linguistiques. On ne fait pas la reconnaissance des termes biologiques exactement comme on pourrait le faire avec des termes informatiques. Ces deux aspects sont les raisons pour lesquelles il est difficile de trouver des ensembles d'entrainement utilisés à grande échelle pour la tâche d'extraction automatique des termes (Zadeh & Handschuh 2014). Il existe des ensembles d'entrainement dans le domaine biologique (GENIA) ou le domaine de la linguistique informatique (ACL RD-TEC), mais beaucoup d'études utilisent leurs propres données, ce qui complexifie la comparaison entre les différents outils utilisés.

Un autre point particulier à la tâche est qu'elle est réalisée à l'échelle d'un corpus de documents. L'objectif est de sélectionner les termes d'intérêts à l'échelle de plusieurs documents, donc des termes qui se répètent un certain nombre de fois, sur lesquelles on est capable de calculer des statistiques ou des patrons de distribution. C'est d'ailleurs l'objectif de l'approche statistique et la base de la détermination de la notion de représentativité. L'objectif dans la présente étude est différent. Si on cherche bien à reconnaitre des termes, on cherche à pouvoir le faire à l'échelle d'un document ou de quelques documents parmi un ensemble plus grand de documents. Il est possible qu'un terme ne soit présent qu'une fois dans un document, qu'il ne concerne absolument pas le domaine d'affaires traité par le

document dans lequel il apparait, mais qu'il soit tout de même important pour la modélisation. Le mot « nom » par exemple est un terme très général qui ne fait certainement pas partie du domaine d'affaires de la construction d'avion, mais il y a de bonnes chances que ce soit un attribut d'une classe de notre modèle final modélisant une application de construction d'avion. Par conséquent, nous ne pouvons pas nous baser sur une approche statistique distributionnelle pour extraire les termes.

Enfin, l'extraction automatique de termes s'intéresse uniquement aux termes. Il n'y a pas d'autres objectifs que d'extraire de manière statique ces termes. Par conséquent, les unités lexicales ciblées sont les noms et les expressions nominales, mais pas les pronoms qui ne constituent pas une information du domaine d'affaires. Dans notre étude, l'extraction de termes n'est que la première phase d'une séquence de tâches impliquant de pouvoir faire le lien entre des termes. Par conséquent, nous devons pouvoir identifier toutes les mentions d'un terme, qu'elle soit nominale ou pronominale.

3.2.2 La reconnaissance d'entités nommées

3.2.2.1 Les principes de la tâche

Une tâche « d'entités nommées » est apparue pour la première fois lors de la conférence MUC-6 (Message Understanding Conference) tenue en 1995 (Grishman & Sundheim 1995). Les conférences MUC sont une série de sept conférences qui se sont tenues entre 1987 et 1997 et qui ouvraient à chaque édition un concours portant sur la résolution d'une ou de plusieurs tâches du domaine de l'extraction d'informations. Les cinq

premières éditions se sont concentrées sur une tâche d'extraction d'information générale portant sur des évènements relatés dans des documents écrits, avant que la sixième ne définisse quatre tâches plus précises dont une tâche « d'entités nommées » et une tâche « de coréférences » (Grishman & Sundheim 1995). Les entités nommées sont alors définies comme les expressions désignant une personne, une organisation, un lieu géographique ainsi que les expressions désignant une date, une heure, une quantité d'argent ou un pourcentage.

La tâche relative aux entités nommées a évolué avec le programme ACE (Automatic Content Extraction). Le programme ACE est un programme de recherche centré sur le développement de technologies destinées à extraire l'information contenue dans différentes sources multimédias (Doddington et al. 2004). Il s'est déroulé entre 1999 et 2008 prenant ainsi la relève des conférences MUC. Avec le programme ACE, la tâche devient la détection des entités nommées (Entity Detection and Tracking EDT), ce qui induit des changements. Tout d'abord, les catégories de classification changent : les entités sont dorénavant les expressions qui désignent une personne, une organisation, une entité géopolitique, un lieu, un bâtiment, un véhicule ou une arme. Ensuite, l'objectif de la tâche s'élargit puisque ACE ne se contente plus de vouloir extraire uniquement les mentions qui désignent de manière directe et explicite une entité nommée (typiquement les noms propres), mais veut extraire toute l'information relative à une entité (Doddington et al. 2004). Cela implique 1° d'identifier et de catégoriser les mentions directes relatives à une entité nommée, 2° d'identifier les mentions indirectes relatives à une entité nommée comme un pronom ou une expression synonyme et 3° de faire le lien entre toutes les

mentions et leur entité parente ce qui suppose d'inclure un travail de résolution de coréférences. Toutefois, le travail de résolution de coréférences est progressivement devenu une tâche à part entière et la détection d'entités nommées s'est spécialisée sur le premier et éventuellement le deuxième aspect de la tâche telle que l'avaient envisagée Doddington et al. (2004).

La détection d'entités nommées dispose de nombreux ensembles d'entrainements (Li et al. 2020). Certains de ces ensembles ont été mis en place à l'occasion de concours, mais ont continué à être utilisés après, ce qui permet de disposer de nombreuses études sur un même ensemble d'entrainement et de pouvoir comparer facilement les résultats obtenus. Cela a aussi contribué à renforcer la popularité de la tâche. Parmi ces ensembles d'entrainement, on trouve notamment :

- MUC-6 (Grishman & Sundheim 1996) et MUC-7 (Chinchor & Marsh 1998) développés entre 1995 et 1997 qui classent les entités en sept catégories,

- ACEs (Doddington et al. 2004) développés entre 2000 et 2008 qui classent les entités en sept catégories,

- GENIA (Ohta et al. 2002) développé à partir de 2001 qui classe les entités en 36 catégories et a la particularité d'être spécifique au domaine biologique et médical,

- CoNLL-2003 (Tjong Kim Sang & De Meulder 2003) développé en 2003 qui classe les entités en quatre catégories,

- OntoNotes (Hovy et al. 2006) développé entre 2006 et 2012 qui classe les entités en 18 catégories.

Suite aux approches traditionnelles basées sur l'application de règles linguistiques et statistiques, la détection d'entités nommées est fondamentalement devenue une tâche de classification et d'étiquetage dans laquelle chaque mot se voit attribuer une étiquette suivant qu'il appartient à une entité nommée ou au reste du document. Le système d'étiquetage qui s'est le plus popularisé pour l'annotation des mots est le système à trois étiquettes BIO (Beginning, Inside, Outside) qui a évolué ensuite en un système à cinq étiquettes BILOU (Beginning, Inside, Last, Outside, Unit) plus efficace (Ratinov & Roth 2009). Les mots qui ne font pas partie d'une entité nommée doivent se voir attribuer l'étiquette O. Les mots qui font partie d'une entité doivent se voir attribuer l'une des autres étiquettes suivant sa place dans l'entité nommée : si c'est le premier mot, il doit avoir l'étiquette 'B'; si c'est le dernier mot, il doit avoir l'étiquette 'L', si c'est un mot entre le premier et le dernier, il doit avoir l'étiquette 'I'; si c'est le seul mot de l'entité, il doit avoir l'étiquette 'U'. Pour répondre à la tâche, il faut aussi préciser la nature de l'entité nommée. Pour cette raison, un suffixe est accolé à l'étiquette précisant la nature de l'entité dans laquelle le mot apparait, B-ORG par exemple signifie que le mot est le premier mot d'une entité de type organisation (Figure 24). Suivant le nombre de classes que l'ensemble d'entrainement se propose de reconnaitre, le nombre d'étiquettes peut devenir assez grand.

Il est par exemple de 73 sur OntoNotes analysé avec un système BILOU, mais uniquement de 9 sur CoNLL-2003 analysé avec un système BIO.

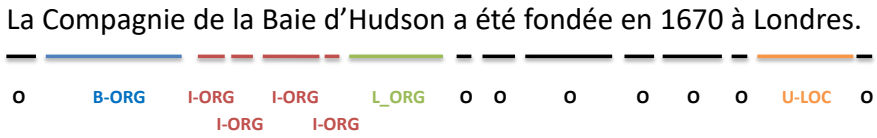


Figure 24. Illustration de l'étiquetage d'une phrase avec le système BILOU.

La délimitation des entités nommées est néanmoins l'objet de débats dans les ensembles d'entrainement : où commence une entité nommée, où finit-elle et quid des entités imbriquées ? Dans l'exemple précédent, l'expression « Compagnie de la Baie d'Hudson » peut être étiquetée de plusieurs manières (Figure 25).

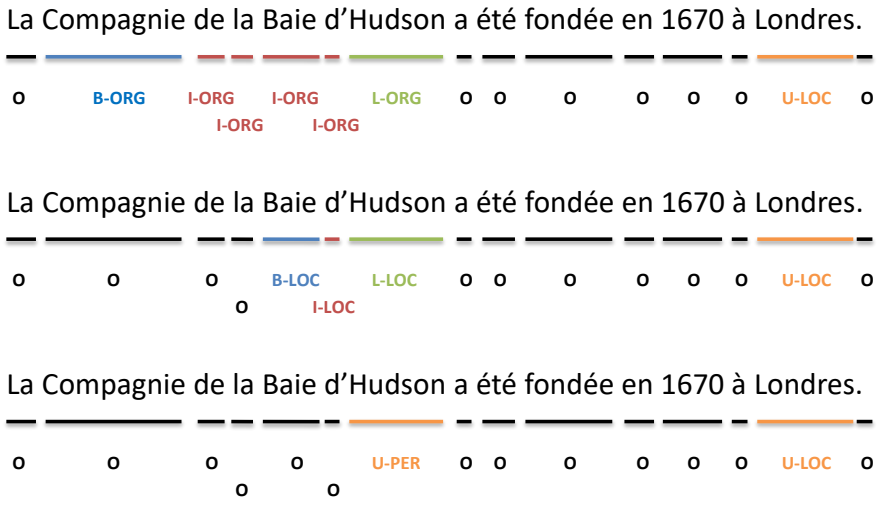


Figure 25. Illustration de l'étiquetage d'entités imbriquées avec le système BILOU.

L'ambigüité induite notamment par les entités imbriquées rend la tâche de reconnaissance des entités nommées parfois subjective. Certaines études ont voulu se pencher sur le problème en essayant d'extraire toutes les entités, imbriquées ou non

(Katiyar & Cardie 2018), remettant en question du même coup l'annotation des ensembles de données standardisés.

Puisque la tâche est une tâche de classification, les métriques utilisées pour quantifier la qualité d'un modèle sont les métriques classiques, précision, rappel et F_1 (voir CHAPITRE 1).

3.2.2.2 La résolution de la tâche

a) *L'APPROCHE SYMBOLIQUE*

Les premières tentatives de résolution de la tâche ont impliqué l'utilisation de règles ou d'index de références. Ces règles ou ces dictionnaires servent de table de correspondance entre une structure et une étiquette : si le mot ou l'expression correspond à la règle x ou à la référence y alors l'étiquette qui lui est attribuée est l'étiquette prédéfinie associée à cette règle ou à cette référence. Étant donné que les objets d'étude de la détection des entités nommées sont des concepts universellement reconnus, il est logique que l'utilisation d'index de références soit répandue, que ce soit seul (Nadeau et al. 2006, Pomares-Quimbaya et al. 2016), en complément de l'utilisation de règles (Kim & Woodland 2000, Sekine & Nobata 2004) ou même comme une caractéristique dans des approches d'apprentissage automatique (Ratinov & Roth 2009, Chiu & Nichols 2016, Shao et al. 2016, Yang et al. 2016). Les index de références sont des dictionnaires déjà existants (Kim & Woodland 2000, Ratinov & Roth 2009, Chiu & Nichols 2016, Pomares-Quimbaya et al. 2016, Shao et al. 2016) ou composés lors de l'étude, à la main, sur des articles de

journaux ou des pages Web (Sekine & Nobata 2004) ou par des approches non supervisées sur des ensembles de pages Web (Nadeau et al. 2006). Les dictionnaires ont l'inconvénient d'apporter une solution peu robuste à la détection d'entités nommées puisqu'elle dépend grandement de l'inventaire d'entités présentes. Le taux de correspondance entre les entités du document et les entités de la référence reste aléatoire, notamment lorsque l'ensemble de données ne fait pas partie des ensembles standards. De plus, son utilisation conjointe avec d'autres approches, notamment les approches d'apprentissage automatique, est un élément de faible impact sur la résolution de la tâche (Shao et al. 2016, Yang et al. 2016).

Les règles utilisées sont essentiellement des règles morphosyntaxiques. Un patron dans l'enchainement des mots ou dans la structure du mot constitue l'élément déclencheur pour l'attribution d'une étiquette. Il peut s'agir par exemple de l'abréviation « Mr » qui induit la présence d'un nom de personne au niveau du mot suivant (Collins & Singer 1999, Sekine & Nobata 2004), de la présence du mot « Incorporated » qui indique le nom d'une compagnie (Collins & Singer 1999) ou encore de l'existence d'une majuscule dans le mot indiquant la présence d'une entité nommée dont il reste à définir la nature (Kim & Woodland 2000). Le jeu de règles utilisées peut être dynamique c'est-à-dire qu'il peut évoluer au cours de l'étude pour conserver les règles qui améliorent les résultats et supprimer les règles qui les altèrent ou qui n'ont pas d'impact (Collins & Singer 1999, Kim & Woodland 2000).

b) *L'APPROCHE PAR APPRENTISSAGE*

Contrairement à la tâche d'extraction automatique de termes qui est une tâche de classification binaire, la détection d'entités nommées est une tâche de classification multi classes. Or la plupart des approches d'apprentissage automatique sont adaptées à des problèmes binaires. Pour pouvoir être utilisées dans le cadre de la détection d'entités nommées, ces approches doivent donc être adaptées. Le principe est alors de décomposer la tâche multi-classes en un ensemble de classificateurs binaires, un pour chaque classe possible et de forger la décision sur le classificateur donnant le résultat le plus probant. Par exemple, si le problème doit aboutir à la classification d'un mot selon trois classes A, B ou C, on construit trois classificateurs binaires, un qui détermine si le mot appartient à la classe A ou pas, un qui détermine si le mot appartient à la classe B ou pas et enfin un qui détermine si le mot appartient à la classe C ou pas. Si les différents classificateurs donnent un résultat cohérent, c'est-à-dire que le mot est identifié comme appartenant à l'une des classes et pas à toutes les autres, le problème est réglé. L'inconvénient survient lorsque plusieurs classes sont identifiées ou aucune. Il faut alors se baser sur la force des résultats obtenus au niveau des différents classificateurs. Dans le cas de la détection des entités nommées, les approches utilisées comprennent notamment les arbres de décision, les machines à vecteurs de support et les champs aléatoires conditionnels (McDonald & Pereira 2005)

La seconde technique utilisée est de déterminer le modèle, représenté par un ensemble de paramètres θ, qui calcule la probabilité la plus adaptée d'attribuer une étiquette y à un mot de la séquence X, $P_\theta(y|X)$. Pour ce faire, il s'agit de trouver le

maximum de la fonction de vraisemblance ($L(\theta)$) sur la séquence de mots (maximum de vraisemblance) :

$$\hat{\theta} = argmax(L(\theta))$$

$$avec\ L(\theta) = \prod P_\theta(y_i|X_i)$$

Puisque la fonction logarithme est strictement croissante, ceci revient à maximiser la log-vraisemblance

$$logL(\theta) = l(\theta) = \sum logP_\theta(y_i|X_i)$$

Le maximum peut se trouver en déterminant $\hat{\theta}$ où la dérivée s'annule :

$$\frac{\partial l(\theta)}{\partial \theta} = 0$$

McDonald & Pereira (2005) utilisent les champs conditionnels aléatoires (CRF) pour déterminer la combinaison linéaire de règles la plus adaptée aux observations. Ratinov & Roth (2009) utilisent plutôt un réseau de neurones de type Perceptron, pour élaborer leur modèle. Chiu & Nichols (2016) intègrent l'encodage de leur jeu de règles dans un vecteur plus complexe constitué par la concaténation d'informations multiples dont la présence dans un index de référence, une représentation distribuée du mot (plongement) et une représentation distribuée basée sur les caractères obtenus grâce à un réseau de neurones

convolutif. Le modèle choisi pour traiter ce vecteur est un réseau de neurones composé d'une couche d'encodage bidirectionnelle récurrente constitué de cellules LSTM (RNN bi-LSTM) et d'une couche de décodage linéaire avec une fonction d'activation softmax.

La composition des vecteurs utilisés comme représentations des mots dans l'approche par apprentissage a tout d'abord utilisé les règles morphosyntaxiques de l'approche symbolique. Les règles sont encodées numériquement et chaque mot est représenté par un vecteur de valeurs α_i où chaque α_i représente la note obtenue par le mot pour la règle i. Cependant, comme nous l'avons précisé, le recours à la composition de règles morphosyntaxiques impose un travail d'analyse en amont fastidieux. L'utilisation des règles est donc progressivement mise de côté, au profit des représentations distribuées. Les plongements et les représentations distribuées basées sur les caractères obtenues par un encodage issu d'un CNN (Chiu & Nichols 2016, Shao et al. 2016) ou d'un RNN (Lample et al. 2016) sur la séquence de lettres qui compose les mots sont souvent employés de concert et leur association donne les meilleurs résultats. Cependant l'influence de la représentation basée sur les caractères est assez faible en comparaison de l'importance de l'utilisation de plongements, notamment de plongements pré-entrainés (Lample et al. 2016, Shao et al. 2016, Yang et al. 2016, Yu et al. 2020).

L'essor de l'utilisation des représentations distribuées a aussi été encouragé par la démocratisation des architectures neuronales récurrentes et bidirectionnelles qui présente l'avantage d'encoder le contexte entourant le mot. Ainsi le réseau tient compte à la fois de caractéristiques intrinsèques du mot grâce aux plongements et de caractéristiques contextuelles grâce à l'architecture du réseau. L'utilisation de RNN bidirectionnels

composés de cellules LSTM (Chiu & Nichols 2016, Huang et al. 2015, Lample et al. 2016, Shao et al. 2016, Katiyar & Cardie 2018) ou GRU (Yang et al. 2016, Dinarelli & Grobol 2018) s'est ainsi révélé plus pertinent que les FFNN, les CNN ou les RNN unidirectionnels (Huang et al. 2015, Shao et al. 2016).

La reconnaissance d'entités nommées est donc fondamentalement devenue une tâche d'étiquetage servie par un RNN bidirectionnel (essentiellement Bi-LSTM) dans laquelle les paramètres de réseau sont redécouverts par chaque étude ou chaque utilisation de l'architecture. L'arrivée de BERT (Devlin et al. 2018) a modifié cet état de fait en apportant une structure d'encodage pré-entrainée et capable de tenir compte du contexte de la phrase. Ce pré entrainement apporte un éclairage encore plus large sur l'utilisation d'un mot, non pas uniquement à l'échelle de la phrase en cours d'analyse, mais au niveau de la langue utilisée, ce qui a permis d'améliorer encore les performances obtenues sur la tâche (Devlin et al. 2018, Luo et al. 2019, Strakova et al. 2019, Yu et al. 2020, Wang & Lu 2020).

Les structures de décodage, qui sont l'étape finale de traitement de l'apprentissage profond, sont de deux sortes : les structures prédictives directes et les structures prédictives contextuelles. Les premières utilisent uniquement le vecteur issu de la couche d'encodage pour produire l'étiquette ou la catégorie associée au mot analysé et sont essentiellement composées de FFNN, par exemple les classificateurs bi-affines (Yu et al. 2020), éventuellement terminé par une fonction d'activation softmax (Huang et al. 2015, Devlin et al. 2018). Les secondes utilisent à la fois le vecteur issu de la couche d'encodage et les étiquettes déjà prédites des mots qui ont précédé dans la phase d'analyse et sont composés des couches CRF (Huang et al. 2015, Lample et al. 2016, Shao et al. 2016, Yang et al.

2016, Luo et al. 2019), les plus répandues, ou d'un RNN (Dinarelli & Grobol 2018) ou linéaire (Katiyar & Cardie 2018) s'appliquant sur la concaténation du vecteur issu de la couche d'encodage et du plongement de l'étiquette du mot précédent.

Le Tableau 3 résume les différentes approches concernant la reconnaissance d'entités nommées présentées dans ce document.

Tableau 3. Résumé des approches concernant la reconnaissance d'entités nommées. Dans la section Modèles, les parenthèses servent à délimiter les différentes couches du modèle, typiquement la structure utilisée pour représenter les jetons, la couche d'encodage et la couche de décodage.

Étude	Modèle	Jeu de données	Score F1
Collins & Singer 1999	Règles	s.o.	s.o.
Kim & Woodland 2000	Index de références + règles	s.o.	s.o.
Sekine & Nobata 2004	Index de références + règles	s.o.	s.o.
McDonald & Pereira 2005	Index de référence + classificateurs binaires	MEDLINE	82,4
Nadeau et al. 2006	Index de références + règles	MUC-7	73,0
Ratinov & Roth 2009	(Index de références + caractéristiques) + (FFNN)	CoNLL-2003	90,8
Chiu & Nichols 2015	(Index de références + caractéristiques + CNN + plongements) + (Bi-RNN) + (FFNN)	CoNLL-2003 OntoNotes	91,6 86,3
Huang et al. 2015	(Index de références + caractéristiques + plongements) + (Bi-RNN) + (CRF)	CoNLL-2003	90,1
Lample et al. 2016	(RNN + plongements) + (Bi-RNN) + (CRF)	CoNLL-2003	90,9
Shao et al. 2016	(Index de références + CNN + plongements) + (Bi-RNN) + (CRF)	CoNLL-2003	88,5
Yang et al. 2016	(Index de références + RNN + plongements) + (Bi-RNN) + (CRF)	CoNLL-2003	90,9
Dinarelli & Grobol 2018	(Plongements) + (Bi-RNN) + (RNN)	MEDIA ATIS	87,5 95,7
Katiyar & Cardie 2018	(Plongements) + (Bi-RNN) + (FFNN)	ACE04 ACE05 GENIA	72,7 70,5 73,6
Devlin et al. 2018	(BERT) + (FFNN)	CoNLL-2003	92,8
Luo et al. 2019	(CNN + Bi-RNN + BERT) + (Bi-RNN) + (CRF)	OntoNotes CoNLL-2003	90,3 93,4
Strakova et al. 2019	(BERT) + (Bi-RNN) + (RNN)	ACE04 ACE05 GENIA	84,4 84,3 78,3
Yu et al. 2020	(CNN + BERT) + (Bi-RNN) + (FFNN)	ACE04 ACE05 GENIA OntoNotes CoNLL-2003	86,7 85,4 80,5 91,3 93,5
Wang & Lu 2020	(RNN + BERT) + (4D-RNN + Attention) + (RNN)	ACE04 ACE05 CoNLL04	88,6 89,5 90,1

3.2.3 La résolution de coréférences

3.2.3.1 Les principes de la tâche

La résolution de coréférences est l'identification des mentions d'un document qui font référence à la même entité du monde réel (Ng 2017). La notion d'entité du monde réel peut être source de confusion. Étant donné que la résolution de coréférences est une tâche liée à la reconnaissance d'entités nommées depuis les conférences MUC (Grishman & Sundheim 1995) et que les jeux de données utilisés sont souvent les mêmes (MUC, ACE, OntoNotes), les entités du monde réel qui sont analysées sont les entités nommées. Sukthanker et al. (2018) fait la distinction entre résolution de coréférences et résolution d'anaphores, la première impliquant la référence à une entité du monde réel, donc à une information de connaissance générale non contenue dans le document, la seconde étant un phénomène purement linguistique qui fait le lien entre deux mots ou expressions du document sans chercher à les relier à une entité extérieure. La résolution d'anaphores sous-entend également une relation unidirectionnelle entre les deux mots ou expressions considérés. De par sa définition, le mot anaphore signifie « qui fait référence à un terme passé » (Sukthanker et al. 2018, Stylianou & Vlahavas 2019). L'anaphore est donc le second terme d'un couple qui implique également un antécédent et qui apporte l'éclairage sur l'entité représentée par l'anaphore. Il arrive cependant que l'entité principale ne soit désignée qu'après une de ses références. Dans ce cas, on ne parle pas d'anaphore et d'antécédent, mais de cataphore et de postcédent (Figure 26).

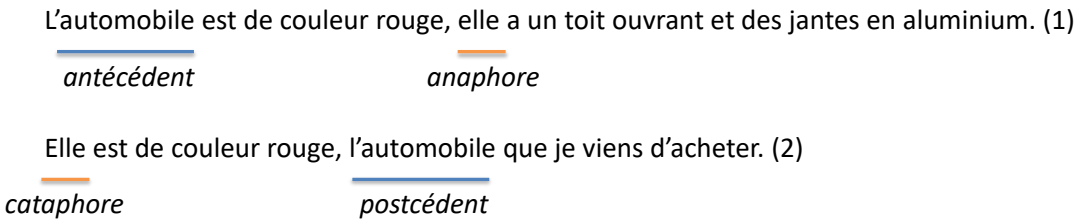


Figure 26. Illustration de structures anaphorique (1) et cataphorique (2).

La résolution de coréférences n'implique pas de sens de relation entre les deux mentions et peut donc désigner les deux types de structures grammaticales. Dans le reste du document, nous parlerons uniquement de résolution de coréférences qui semble un terme plus général et plus approprié à l'objectif que l'on souhaite atteindre et qui est de pouvoir rassembler toutes les mentions qui désignent la même entité, que la référence soit située avant ou après. Nous employons également le terme de coréférence, même si les entités dont il est question peuvent ne pas correspondre à des entités précises du monde réel : une automobile ne fait pas référence à un objet unique. C'est un concept qui peut désigner un objet réel ou l'idée que nous nous faisons d'un objet réel. Néanmoins, si l'on veut pouvoir retrouver toutes ses caractéristiques, nous devons rassembler toutes les mentions qui lui font référence.

La résolution de coréférences est une tâche de regroupement (Clark & Manning 2016a). Il s'agit de grouper dans un même ensemble les mentions qui désignent la même entité. Décrit de manière générale, le processus de résolution de la tâche consiste à parcourir le document du début à la fin et à chaque mention d'intérêt de déterminer si la mention peut être liée à une mention ou un ensemble déjà rencontré ou si c'est la première occurrence de l'entité. La résolution de coréférences implique donc de répondre à deux

sous questions : 1° est-ce que la mention est anaphorique, c'est-à-dire est-ce qu'elle fait référence à une entité mentionnée précédemment dans le document et 2° quelle est la référence (Wiseman et al. 2015, Ng 2017). Bien sûr, répondre non à la première question simplifie grandement la seconde. Cependant, répondre à la première question n'est pas plus facile que de répondre à la seconde question et puisque la seconde doit de toute façon être adressée, la plupart des études s'attaquent aux deux questions simultanément (Ng 2017).

Les approches par association de mentions (mention-pair) analysent le potentiel d'une paire de mentions à représenter la même entité et donc à faire partie du même ensemble (Ng 2017). Les paires de mentions sont analysées de manière indépendante les unes des autres. Si une mention ne peut être liée avec une mention la précédant, c'est qu'elle n'est pas anaphorique et constitue donc la première mention d'un nouvel ensemble. Cette approche ouvre la porte à des ambigüités dans les ensembles construits : puisque les paires de mentions sont analysées indépendamment, il est possible qu'une mention soit liée à plusieurs autres mentions qui n'appartiennent pas aux mêmes ensembles. On se retrouve avec des ensembles non disjoints qu'il faut séparer dans un deuxième temps (Ng 2017). Des algorithmes sont donc construits pour spécifiquement construire les grappes disjointes après l'analyse des paires de mentions (Ng 2017).

Les approches par association entité-mention (entity-mention) reposent sur un principe similaire aux approches par paire de mentions, mais en analysant le potentiel d'association d'une mention non pas avec les mentions la précédant, mais avec les grappes déjà en cours de constitution. Le premier avantage est que cela réduit le nombre d'associations potentielles à analyser. Le deuxième avantage est qu'il est possible d'extraire

une variété de caractéristiques plus grande en travaillant au niveau de la grappe de mentions qu'en travaillant au niveau d'une mention et donc de trouver des corrélations plus évidentes avec la mention en cours de traitement (Ng 2017). Par contre, cela n'empêche pas les ambigüités dans les grappes construites puisqu'encore une fois les potentiels d'association sont analysés de manière indépendante et une mention peut donc se retrouver associée à plusieurs grappes. Là encore cela oblige à désambigüiser les grappes dans un deuxième temps.

Les approches par classement de mentions (mention-ranking) ont été mises au point pour contrer les ambigüités d'association liées aux associations multiples des deux approches précédentes. Les associations potentielles d'une mention avec les mentions précédentes ne sont cette fois plus analysées de manière indépendante, mais sont évaluées de manière simultanée et compétitive, c'est-à-dire qu'une mention est associée à la mention précédente avec laquelle elle a la plus forte affinité (Sukthanker et al. 2018). De ce fait, une mention ne peut plus être associée à plusieurs mentions. Par contre il y a encore des sources d'ambigüités puisqu'une mention c peut se retrouver associée à une mention b, que la mention b peut être associée à la mention a et que par transitivité c puisse se retrouver associée à a alors que ça ne devrait pas être le cas (Sytlianou & Vlahavas 2019).

Les approches par classement de grappes (cluster-ranking) sont l'aboutissement de la tentative de désambigüisation des regroupements de mentions. Les mentions sont évaluées en fonction des grappes en cours de formation comme dans l'approche par association entité-mention et cette évaluation se fait de manière simultanée et compétitive comme dans l'approche par classement de mentions de sorte qu'une mention ne peut être associée qu'à

une grappe en cours de formation éliminant de ce fait les ambigüités (Ng 2017, Sukthanker et al. 2018). Cela ne garantit pas que l'association est correcte, cela garantit uniquement qu'il n'y a pas d'incohérence dans les groupements qui sont constitués.

Toutes les approches décrites précédemment cherchent à regrouper les mentions de manière itérative en mettant l'accent sur la relation entre la mention i et les mentions ou les grappes déjà identifiées. D'autres approches, les approches de structure latentes, construisent une structure représentant les liens potentiels entre les mentions, par exemple une structure en arbre, au fur et à mesure de la lecture du document. Le point focal n'est plus la vraisemblance de la relation entre la mention i et un antécédent, mais la vraisemblance de la structure latente obtenue (Björkelund & Kuhn 2014).

Suite à la conférence MUC-6 qui a identifié la tâche de résolution de coréférences comme une tâche associée à la reconnaissance d'entités nommées (Grishman & Sundheim 1995), plusieurs jeux de données utilisés pour la reconnaissance d'entités nommées ont également été annotés pour servir à la résolution de coréférences. Pour cette raison, les entités ciblées par ces jeux de données pour la résolution de coréférences sont naturellement des entités nommées. MUC-6 et MUC-7 ont été les premiers utilisés et à permettre une comparaison entre les études bien que les jeux de données soient d'une taille limitée (Sukthanker et al. 2018, Stylianou & Vlahavas 2019). ACE, développé par la suite, a quant à lui l'inconvénient de ne pas avoir eu d'ensemble de tests clairement identifié et donc de ne permettre que des comparaisons relatives entre les études qui l'utilisent (Sukthanker et al. 2018, Stylianou & Vlahavas 2019). Le jeu de données le plus utilisé reste donc OntoNotes5.

La tâche de résolution de coréférences étant une tâche de groupement, les indices de précision, de rappel ne peuvent pas être utilisés dans leur définition standard pour évaluer la performance d'un modèle. Ces indices sont pertinents pour une tâche dont le résultat est binaire, Vrai ou Faux, Positif ou Négatif. Dans ces conditions, il est possible de calculer un nombre de réponses vraiment positives, faussement négatives, etc. Dans le cas où le résultat consiste en des regroupements ou des associations entre mentions, les notions de positif ou négatif doivent être considérées au regard de l'appartenance d'une mention à un groupe ou à un autre dans le résultat obtenu. Prenons l'exemple décrit dans la Figure 27. Le regroupement véridique est composé des deux groupes (A, B, C) et (D, E. F), mais le modèle a prédit les groupes (A, B) et (C, D, E, F). Il est évident que la mention C n'est pas bien compartimentée, mais il ne fait pas de sens de parler de faux négatif. Il faut donc envisager de redéfinir les indices de précision et de rappel pour avoir une base de comparaison de la performance des modèles. Si ces deux indices sont redéfinis, l'indice F_1 est quant à lui toujours calculé de la même façon.

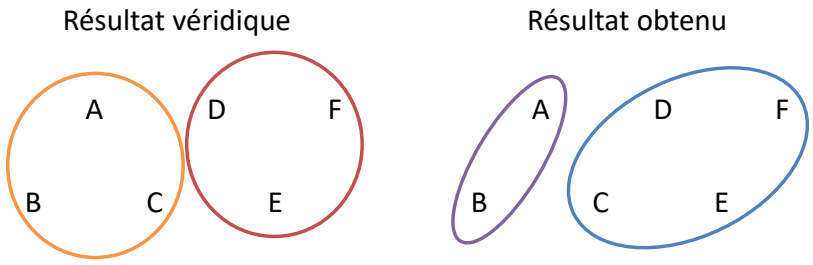


Figure 27. Illustration d'un regroupement d'une tâche de résolution de coréférence.

Le premier indice développé pour estimer la performance d'un modèle pour la tâche est l'indice MUC développé dans le cadre des conférences du même nom (Vilain et al. 1995). Si l'on désigne par V l'ensemble des groupes dans le résultat véridique et v l'un de

ces groupes, O l'ensemble des groupes dans le résultat obtenu et o l'un de ces groupes, et si l'on désigne par $partition(x, Y)$ l'ensemble des groupes dans Y qui contient les éléments du groupe x, alors :

$$Précision = \frac{\sum_{o \in O}(|o| - |partition(o, V)|)}{\sum_{o \in O}(|o| - 1)}$$

$$Rappel = \frac{\sum_{v \in V}(|v| - |partition(v, O)|)}{\sum_{v \in V}(|v| - 1)}$$

Par exemple, pour le cas décrit à la Figure 27, les indices sont calculés de la manière suivante :

- Précision : le résultat obtenu est composé des deux groupes (A, B) et (C, D, E, F) :
    - Pour le groupe (A, B), le groupe (A, B, C) du résultat véridique contient les éléments de (A, B) donc $partition((A, B), V)$ n'est composée que d'un élément, le groupe (A, B, C) :

$$\frac{|o| - |partition(o, V)|}{|o| - 1} = \frac{2 - 1}{2 - 1}$$

    - Pour le groupe (C, D, E, F), les groupes (A, B, C) et (D, E, F) du résultat véridique contiennent les éléments de (C, D, E, F) donc

$partition((C, D, E, F), V)$ est composée de deux éléments, les groupes (A, B, C) et (D, E, F) :

$$\frac{|o| - |partition(o, V)|}{|o| - 1} = \frac{4 - 2}{4 - 1}$$

Donc,

$$Précision = \frac{\sum_{o \in O}(|o| - |partition(o, V)|)}{\sum_{o \in O}(|o| - 1)} = \frac{(2 - 1) + (4 - 2)}{(2 - 1) + (4 - 1)} = \frac{1 + 2}{1 + 3} = \frac{3}{4} = 75\%$$

- Rappel : Le résultat véridique est composé de deux groupes (A, B, C) et (D, E, F) :

    - Pour le groupe (A, B, C), les groupes (A, B) et (C, D, E, F) du résultat obtenu contiennent les éléments de (A, B, C), donc $partition((A, B, C), O)$ est composée de deux éléments, les groupes (A, B) et (C, D, E, F) :

$$\frac{|v| - |partition(v, O)|}{|v| - 1} = \frac{3 - 2}{3 - 1}$$

    - Pour le groupe (D, E, F), le groupe (C, D, E, F) du résultat obtenu contient les éléments de (D, E, F), donc $partition((D, E, F), O)$ n'est composée que d'un élément, le groupe (C, D, E, F) :

$$\frac{|v| - |partition(v, O)|}{|v| - 1} = \frac{3 - 1}{3 - 1}$$

Donc,

$$Rappel = \frac{\sum_{v \in V}(|v| - |partition(v, O)|)}{\sum_{v \in V}(|v| - 1)} = \frac{(3-2)+(3-1)}{(3-1)+(3-1)} = \frac{1+2}{2+2} = \frac{3}{4} = 75\%$$

Donc,

$$F1 = \frac{2 \times \frac{3}{4} \times \frac{3}{4}}{\frac{3}{4} + \frac{3}{4}} = \frac{\frac{9}{8}}{\frac{3}{2}} = \frac{6}{7} \cong 85{,}7\%$$

Le problème de l'indice MUC est qu'il n'est pas conçu pour tenir compte des singletons, c'est-à-dire les mentions qui ne sont représentées qu'une seule fois dans un document (Bagga & Baldwin 1998). Pour compenser cette situation, l'indice B^3 a été développé (Bagga & Baldwin 1998). Si l'on désigne par vo_m les éléments correctement groupés dans le groupe contenant l'entité m du résultat obtenu, v_m les éléments du groupe contenant l'entité m dans le résultat véridique, o_m les éléments du groupe contenant l'entité m dans le résultat obtenu et w_m le poids accordé à l'entité m que l'on peut prendre identique pour toutes les entités égal à $\frac{1}{|v_m|}$ alors,

$$Précision = \sum_m w_m \frac{|vo_m|}{|o_m|}$$

$$Rappel = \sum_m w_m \frac{|vo_m|}{|v_m|}$$

Par exemple, pour le cas décrit à la Figure 27, les indices sont calculés de la manière suivante :

- Précision : le résultat obtenu est composé des groupes (A, B) et (C, D, E, F)

    - Les mentions A et B font partie d'un groupe de deux éléments dans le résultat obtenu qui fait écho au groupe (A, B, C) dans le résultat véridique. Pour ces deux mentions, le groupe obtenu contient donc deux mentions (A et B) correctement groupées au regard du groupe (A, B, C) du résultat véridique et le groupe obtenu contient deux mentions :

$$\frac{|vo_m|}{|o_m|} = \frac{2}{2}$$

    - La mention C fait partie d'un groupe de quatre éléments (C, D, E, F) dans le résultat obtenu, mais dont C est la seule mention qui fait écho au groupe (A, B, C) du résultat véridique. Pour cette mention, le groupe obtenu contient donc une seule mention correctement groupée au regard du groupe (A, B, C) du résultat véridique et le groupe obtenu contient quatre mentions :

$$\frac{|vo_m|}{|o_m|} = \frac{1}{4}$$

    - Les mentions D, E et F font parties d'un groupe de quatre éléments (C, D, E, F) dans le résultat obtenu qui fait écho au groupe (D, E, F) dans le résultat

véridique. Pour ces trois mentions, le groupe obtenu contient donc trois mentions (D, E et F) correctement groupées au regard du groupe (D, E, F) du résultat véridique et le groupe obtenu contient quatre mentions :

$$\frac{|vo_m|}{|o_m|} = \frac{3}{4}$$

Donc,

$$Précision = \sum_m w_m \frac{|vo_m|}{|o_m|} = \frac{1}{6} \times \left(\frac{2}{2} + \frac{2}{2} + \frac{1}{4} + \frac{3}{4} + \frac{3}{4} + \frac{3}{4}\right) = \frac{1}{6} \times \frac{9}{2} = \frac{3}{4} = 75\%$$

- Rappel : le résultat obtenu est composé des groupes (A, B) et (C, D, E, F) :

    - Les mentions A et B font partie d'un groupe de deux éléments dans le résultat obtenu qui fait écho au groupe (A, B, C) dans le résultat véridique. Pour ces deux mentions, le groupe obtenu contient donc deux mentions (A et B) correctement groupées au regard du groupe (A, B, C) du résultat véridique qui contient trois éléments :

$$\frac{|vo_m|}{|v_m|} = \frac{2}{3}$$

    - La mention C fait partie d'un groupe de quatre éléments (C, D, E, F) dans le résultat obtenu, mais dont C est la seule mention qui fait écho au groupe (A,

B, C) du résultat véridique. Pour cette mention, le groupe obtenu contient donc une seule mention correctement groupée au regard du groupe (A, B, C) du résultat véridique qui contient trois éléments :

$$\frac{|vo_m|}{|v_m|} = \frac{1}{3}$$

- Les mentions D, E et F font parties d'un groupe de quatre éléments (C, D, E, F) dans le résultat obtenu qui fait écho au groupe (D, E, F) dans le résultat véridique. Pour ces trois mentions, le groupe obtenu contient donc trois mentions (D, E et F) correctement groupées au regard du groupe (D, E, F) du résultat véridique qui contient trois éléments :

$$\frac{|vo_m|}{|o_m|} = \frac{3}{3}$$

Donc,

$$Rappel = \sum_m w_m \frac{|vo_m|}{|ov_m|} = \frac{1}{6} \times \left(\frac{2}{3} + \frac{2}{3} + \frac{1}{3} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3}\right) = \frac{1}{6} \times \frac{14}{3} = \frac{7}{9} \cong 77{,}8\%$$

Donc,

$$F1 = \frac{2 \times \frac{3}{4} \times \frac{7}{9}}{\frac{3}{4} + \frac{7}{9}} = \frac{\frac{7}{6}}{\frac{55}{36}} = \frac{42}{55} \cong 76{,}4\%$$

L'inconvénient de l'indice B^3 est que les entités sont considérées plusieurs fois (Luo 2005). Pour compenser cet inconvénient, l'indice CEAF (Constrained Entity Alightment F-measure) a donc été développé (Luo 2005). L'indice CEAF est en fait un ensemble de quatre indices qui correspondent à quatre manières de calculer la ressemblance entre deux groupes o et v :

$$\varphi_1(o,v) = \begin{cases} 1 \; si \; o = v \\ 0 \; sinon \end{cases}$$

$$\varphi_2(o,v) = \begin{cases} 1 \; si \; o \cap v \neq \emptyset \\ 0 \; sinon \end{cases}$$

$$\varphi_3(o,v) = |o \cap v|$$

$$\varphi_4(o,v) = \frac{2 \times |o \cap v|}{|o| + |v|}$$

Pour chaque manière de calculer la ressemblance, on peut calculer la précision et le rappel en considérant o comme un groupe du résultat obtenu, v comme un groupe du résultat véridique et $v(o)$ le groupe du résultat véridique qui correspond le mieux au groupe o du résultat obtenu (un groupe véridique ne peut être mis en correspondance qu'avec un groupe obtenu) :

$$Précision = \frac{\sum_{o \in O} \varphi_i(o, v(o))}{\sum_{o \in O} \varphi_i(o,o)}$$

$$Rappel = \frac{\sum_{o \in O} \varphi_i(o, v(o))}{\sum_{o \in O} \varphi_i(v, v)}$$

En pratique, seuls φ_3 et φ_4 sont utilisés et si on calcule les indices pour l'exemple de la Figure 27 :

- Précision : le groupe (A, B, C) du résultat véridique et le groupe (A, B) du résultat obtenu peuvent être mis en correspondance, tout comme le groupe (D, E, F) du résultat véridique et le groupe (C, D, E, F) du résultat obtenu.

    - Le groupe (A, B, C) du résultat véridique et le groupe (A, B) du résultat obtenu ont deux mentions en commun.

$$\varphi_3\big(o, v(o)\big) = |o \cap v(o)| = 2$$

$$\varphi_3(o, o) = |o \cap o| = 2$$

$$\varphi_4\big(o, v(o)\big) = \frac{2 \times |o \cap v(o)|}{|o| + |v(o)|} = \frac{2 \times 2}{2 + 3} = \frac{4}{5}$$

$$\varphi_4(o, o) = \frac{2 \times |o \cap o|}{|o| + |o|} = \frac{2 \times 2}{2 + 2} = 1$$

    - Le groupe (D, E, F) du résultat véridique et le groupe (C, D, E, F) du résultat obtenu ont trois mentions en commun.

$$\varphi_3\big(o, v(o)\big) = \ |o \cap v(o)| = 3$$

$$\varphi_3(o, o) = \ |o \cap o| = 4$$

$$\varphi_4\big(o, v(o)\big) = \frac{2 \times |o \cap v(o)|}{|o| + \ |v(o)|} = \frac{2 \times 3}{4 + 3} = \frac{6}{7}$$

$$\varphi_4(o, o) = \frac{2 \times |o \cap o|}{|o| + |o|} = \frac{2 \times 4}{4 + 4} = 1$$

Donc,

$$Précision_{\varphi_3} = \frac{\sum_{o \in O} \varphi_3(o, v(o))}{\sum_{o \in O} \varphi_3(o, o)} = \frac{2 + 3}{2 + 4} = \frac{5}{6} \cong 83{,}3\%$$

$$Précision_{\varphi_4} = \frac{\sum_{o \in O} \varphi_4(o, v(o))}{\sum_{o \in O} \varphi_4(o, o)} = \frac{\frac{4}{5} + \frac{6}{7}}{1 + 1} = \frac{26}{35} \cong 74{,}3\%$$

- Rappel :
    - Pour le groupe (A, B, C) du résultat véridique et le groupe (A, B) du résultat obtenu :

$$\varphi_3\big(o, v(o)\big) = \ |o \cap v(o)| = 2$$

$$\varphi_3(v, v) = |v \cap v| = 3$$

$$\varphi_4\big(o, v(o)\big) = \frac{2 \times |o \cap v(o)|}{|o| + |v(o)|} = \frac{2 \times 2}{2 + 3} = \frac{4}{5}$$

$$\varphi_4(v, v) = \frac{2 \times |v \cap v|}{|v| + |v|} = \frac{2 \times 3}{3 + 3} = 1$$

- Pour le groupe (D, E, F) du résultat véridique et le groupe (C, D, E, F) du résultat obtenu :

$$\varphi_3\big(o, v(o)\big) = |o \cap v(o)| = 3$$

$$\varphi_3(v, v) = |v \cap v| = 3$$

$$\varphi_4\big(o, v(o)\big) = \frac{2 \times |o \cap v(o)|}{|o| + |v(o)|} = \frac{2 \times 3}{4 + 3} = \frac{6}{7}$$

$$\varphi_4(v, v) = \frac{2 \times |v \cap v|}{|v| + |v|} = \frac{2 \times 3}{3 + 3} = 1$$

Donc,

$$Rappel_{\varphi_3} = \frac{\sum_{o \in O} \varphi_3(o, v(o))}{\sum_{o \in O} \varphi_3(v, v)} = \frac{2 + 3}{3 + 3} = \frac{5}{6} \cong 83{,}3\%$$

$$Rappel_{\varphi_4} = \frac{\sum_{o \in O} \varphi_4(o, v(o))}{\sum_{o \in O} \varphi_4(v, v)} = \frac{\frac{4}{5} + \frac{6}{7}}{1+1} = \frac{26}{35} \cong 74{,}3\%$$

Donc,

$$F1_{\varphi_3} = CEAF_{\varphi_3} = \frac{2 \times \frac{5}{6} \times \frac{5}{6}}{\frac{5}{6} + \frac{5}{6}} = \frac{\frac{25}{18}}{\frac{5}{3}} = \frac{5}{6} \cong 83{,}3\%$$

$$F1_{\varphi_4} = CEAF_{\varphi_4} = \frac{2 \times \frac{26}{35} \times \frac{26}{35}}{\frac{26}{35} + \frac{26}{35}} = \frac{\frac{1352}{1225}}{\frac{52}{35}} = \frac{26}{35} \cong 74{,}3\%$$

Finalement l'indice CoNLL est un autre indice qui est calculé en faisant une synthèse des résultats obtenus par les trois premiers indices (Pradhan et al. 2012).

$$CoNLL = \frac{MUC_{F1} + B^3_{F1} + CEAF_{F1}}{3}$$

Par exemple, pour le cas décrit à la Figure 27 et en prenant l'indice $CEAF_{\varphi_4}$, l'indice est calculé de la manière suivante :

$$CoNLL = \frac{\frac{6}{7} + \frac{42}{55} + \frac{26}{35}}{3} = \frac{\frac{182}{77}}{3} = 78{,}8\%$$

3.2.3.2 La résolution de la tâche

a) *L'APPROCHE SYMBOLIQUE*

Les approches à bases de règles ont été les premières approches mises en œuvre pour résoudre la tâche. Ces approches reposent sur une connaissance à priori du langage et des propriétés syntaxiques, sémantiques et discursives qui permettent d'établir une relation de coréférence entre deux mentions, deux grappes ou une mention et une grappe ou au contraire de statuer qu'une mention est la première référence à une entité rencontrée dans un texte. La manière dont sont utilisées ces règles varie d'une étude à l'autre : elles peuvent être utilisées comme des filtres qui permettent de prendre une décision directement sur la relation de coréférence entre deux structures (Hobbs 1978) ou servir à caractériser les structures avant qu'un modèle ne vienne comparer les caractéristiques et statuer sur la relation de coréférence (Lappin & Leass 1994).

L'approche de Lappin & Leass (1994) utilise notamment un système de poids accordé aux différentes caractéristiques destiné à calculer un score représentatif de la relation entre une mention (un pronom en l'occurrence) et ses différents antécédents potentiels. L'antécédent dont le score est le plus élevé est choisi comme antécédent. Cette démarche préfigure les démarches par classement plus récentes reposant sur l'apprentissage automatique et l'apprentissage profond. Elle présente néanmoins les inconvénients de ne considérer que les relations pronom-antécédent et d'être difficile à calibrer puisque les poids accordés aux différentes caractéristiques sont obtenus par essais successifs.

La démarche la plus aboutit des approches à bases de règles est l'emploi des règles sous forme de tamis (Raghunathan et al. 2010). Les règles sont appliquées sous forme de filtres successifs qui constituent le tamis. Les filtres les plus hauts du tamis (qui sont donc rencontrés en premier) sont composés des règles qui permettent de décider facilement qu'il existe une relation de coréférence entre deux structures (mention ou grappe – par exemple l'identité de deux mentions ou la présence de structures syntaxiques précises) tandis que les filtres les plus bas du tamis sont composés de règles plus aléatoires qui doivent attraper les associations qui n'ont pu être filtrées plus tôt (par exemple la présence d'un mot en commun entre deux mentions). Si l'association passe au travers des filtres sans être retenue c'est que les deux structures ne sont pas anaphoriques. C'est le principe « easy-first » dans lequel les décisions faciles sont prises en premier tandis que les décisions plus difficile sont reportées lorsque de l'information supplémentaire issue du passage au travers des premiers filtres est possiblement devenue disponible.

b) *L'APPROCHE PAR APPRENTISSAGE*

Les approches par apprentissage automatique ont repris le principe des approches à base de règles en exploitant les caractéristiques linguistiques (syntaxiques, sémantiques et discursives) des mentions ou des grappes pour rendre une décision sur la relation entre deux structures. Chaque structure ou chaque association se voit attribuer une représentation (généralement vectorielle) composée du résultat obtenu par celle-ci sur un certain nombre de caractéristiques préétablies et ce sont ces représentations qui passent au travers du modèle pour déterminer s'il existe une relation de coréférence entre les parties impliquées.

Les premières études exploitant l'apprentissage automatique ont utilisé une approche par association (paires de mentions ou entité-mention), c'est-à-dire qu'elles évaluent le potentiel d'association d'une mention avec les mentions précédentes ou les ensembles latents de manière indépendante. La tâche se résume ainsi à développer un classificateur binaire qui détermine si oui ou non une mention et un antécédent sont susceptibles d'être en relation, avant qu'un deuxième modèle ne vienne reconstituer les grappes désignant la même entité. La technique utilisée est une méthode d'apprentissage automatique supervisé de type arbre de décision (Soon et al. 2001, Ng & Cardie 2002, Yang et al. 2004). Les caractéristiques linguistiques employées sont par exemple la distance entre les deux mentions, la nature pronominale de l'antécédent potentiel, la nature pronominale de l'anaphore potentielle, la correspondance dans les chaines de caractères des deux mentions, la présence d'un article défini dans l'anaphore potentielle, la présence d'un adjectif démonstratif dans l'anaphore potentielle, la concordance de nombre entre les deux mentions, la nature (nom propre) des deux mentions, etc. (Soon et al. 2001). Soon et al. (2001) exploitent ainsi un jeu de 12 règles linguistiques appliqué aux mentions, jeu qui est par la suite étendu à 53 par Ng & Cardie (2002), tandis que Yang et al. (2004) utilisent un jeu de 18 caractéristiques appliqué aux mentions auquel s'ajoute un jeu supplémentaire de six caractéristiques appliqué sur la mention et sur la grappe en cours d'évaluation. Le modèle attribue un score de vraisemblance à chaque association. Dès qu'un score dépasse 0,5, les deux parties sont considérées appartenir à la même relation de coréférence. Pour procéder aux regroupements, deux stratégies sont adoptées. La première consiste à évaluer les antécédents potentiels dans l'ordre en partant de l'antécédent le plus proche et en remontant dans le document, et à prendre le premier antécédent qui dépasse le seuil comme

référent, principe du modèle de groupement « closest-first » (Soon et al. 2001). La seconde consiste à évaluer tous les antécédents potentiels puis de choisir, parmi ceux qui ont une vraisemblance supérieure au seuil, celui qui a le score le plus élevé comme référent, principe du modèle de regroupement « best-first » (Ng & Cardie 2002) qui préfigure les approches par classement.

Au début des années 2010, des études ont adapté les techniques d'apprentissage automatique à l'approche « easy-first » en tamis développée par Raghunathan et al. (2010). Pour Stoyanov & Eisner (2012), chaque association potentielle est représentée par une représentation de caractéristiques linguistiques et un modèle de type Perceptron simple vient pondérer l'influence des différentes caractéristiques dans les décisions prises. À chaque étape, la décision la plus simple est prise. Cette décision peut être soit de combiner deux structures (mention ou grappe), soit d'arrêter les regroupements. Le processus est répété jusqu'à ce que la décision d'arrêter soit prise. Les poids de chaque caractéristique sont mis à jour après chaque action si le nouvel ensemble de structures obtenu obtient un résultat inférieur en termes de métriques MUC et B^3. Ratinov & Roth (2012) utilise également un modèle de type Perceptron, mais pour calculer un score de vraisemblance d'association dans des contextes syntaxiques différents, un contexte étant un filtre du tamis. La particularité du modèle repose sur son utilisation de ressources externes pour assembler les caractéristiques des mentions ou des grappes.

L'approche par classement a également été envisagée par les méthodes d'apprentissage automatiques. Les approches par classement (de mentions ou de grappes) évaluent la meilleure association d'une mention avec une mention précédente ou une

grappe latente. La meilleure association se définit par le meilleur score dans un modèle qui calcule un score pour chaque association potentielle. Dans ce type d'approche, la mention ou la grappe à laquelle est rattachée la mention en cours d'analyse est donc définie par :

$$\hat{y} = argmax_{y \in Y(x)} s(x, y)$$

où Y(x) est l'ensemble des mentions qui précèdent la mention x ou des grappes latentes.

La variabilité entre les études repose sur la manière dont est calculé le score d'une association $s(x, y)$. Les approches par classement doivent également statuer sur le caractère anaphorique d'une mention. Il ne s'agit pas de définir par défaut qu'une mention n'est pas anaphorique si elle n'a pas d'antécédent potentiel comme dans les approches par association, il s'agit de déterminer que le score associé au caractère non anaphorique de la mention est le plus élevé. Le problème est que le calcul du score d'une association prend en compte des caractéristiques ou des distributions relatives aux deux parties impliquées. Dans le cas d'une association non anaphorique, il n'y a pas de deuxième partie impliquée, ce que l'on peut noter ε. Par conséquent le score est défini différemment suivant que l'on évalue l'association avec une mention précédente ou une grappe latente, ou avec ε (Rahman & Ng 2009, Wiseman et al. 2015, Clark & Manning 2016a, Wiseman et al. 2016, Lee et al. 2017) :

$$s(x, y) \triangleq \begin{cases} s_1(x, y) & si\ y \neq \varepsilon \\ s_2(x) & si\ y = \varepsilon \end{cases}$$

Rahman & Ng (2009) utilisent une architecture d'apprentissage automatique supervisé de type classificateur SVM pour déterminer le score d'une association mention-grappe latente. Ils utilisent des représentations composées de caractéristiques linguistiques, mais cotées différemment suivant que la caractéristique est partagée par la mention en cours d'analyse et la totalité des mentions constituant la grappe envisagée, la majeure partie des mentions constituant la grappe envisagée, une petite partie des mentions constituant la grappe envisagée ou aucune des mentions constituant la grappe envisagée.

Les représentations utilisées pour les mentions, dans le cadre de l'apprentissage profond, sont d'abord composées de caractéristiques linguistiques uniquement (Wiseman et al. 2015, Wiseman et al. 2016) puis d'une concaténation de caractéristiques linguistiques et de plongements (Clark & Manning 2016a) et finalement de plongements essentiellement auxquels s'ajoutent quelques caractéristiques, notamment la distance entre les mentions (Lee et al. 2017, Lee et al. 2018). Les représentations utilisées pour les grappes sont plus complexes, car elles doivent tenir compte de la variabilité des mentions qui les composent. Wiseman et al. (2016) proposent d'incorporer à leur modèle un RNN composé de cellules LSTM pour encoder une représentation de la grappe à partir de la séquence de mentions qui la compose. L'objectif est toujours de trouver le score maximal qui unit une mention et les antécédents potentiels, mais en ajoutant un membre supplémentaire qui tient compte des grappes latentes :

$$\hat{y} = argmax_{y \in Y(x)} \big(f(x_n, y) + g(x_n, y, z_{1:n-1}) \big)$$

où $f(x, y)$ est une fonction du même type que $s(x, y)$ et modélisée par un FFNN :

$$f(x_n, y) \triangleq \begin{cases} f_1(x_n, y) \; si \; y \neq \varepsilon \\ f_2(x_n) \quad si \; y = \varepsilon \end{cases}$$

et $g(x_n, y, z_{1:n-1})$ est une fonction « globale » calculée en tenant compte des vecteurs cachés issus du RNN qui sert à encoder les grappes latentes, également modélisée par un FFNN :

$$g(x_n, y, z_{1:n-1}) \triangleq \begin{cases} g_1(x_n, h_{<n}^{z_y}) \quad si \; y \neq \varepsilon \\ g_2(x_n, \sum_m h_{<n}^m) \; si \; y = \varepsilon \end{cases}$$

où $h_{<n}^m$ est le vecteur caché de la grappe contenant la mention m construit avec les mentions composant la grappe avant d'évaluer x_n.

Clark & Manning (2016b) utilisent une approche différente pour faire le groupement des grappes. Dans cette approche, chaque mention est considérée comme une grappe constituée d'un élément et le modèle cherche à regrouper les grappes qui désignent la même entité. Pour cela, le modèle est en fait une succession de plusieurs sous-modèles qui 1° encode une paire mention-antécédent grâce à un FFNN et 2° encode une paire de grappes grâce à une couche de pooling sur les encodages de toutes les combinaisons de paires de mentions qui composent les deux grappes.

Le calcul du score proprement dit fait intervenir différentes architectures. La couche d'encodage peut être composée de FFNN (Wiseman et al. 2015, Wiseman et al. 2016,

Clark & Manning 2016a), de CNN (Wu & Ma 2017) ou encore de RNN bi-LSTM (Lee et al. 2017). Étant donné l'objectif des approches par classement, la couche de décodage est toujours constituée d'une couche à propagation avant dont la fonction d'activation est une fonction softmax. Les études cherchent dans la plupart des cas à encoder une relation mention-antécédent en s'intéressant aux représentations des deux parties considérées prises indépendamment de leur contexte immédiat. Il existe des caractéristiques linguistiques qui permettent de résumer une certaine information entourant les mentions ou les grappes, mais l'évaluation au travers de l'encodage et du décodage isolent les deux parties dans une bulle sur laquelle le modèle est utilisé. Lee et al. (2017) utilise une approche différente en considérant les mentions potentielles avec leur contexte immédiat grâce au recours aux RNN bi-LSTM et obtiennent de très bons résultats (Lee et al. 2017, Lee et al. 2018) sur la tâche malgré le fait qu'ils n'utilisent aucune caractéristique linguistique à part la distance entre les mentions et l'auteur de la phrase (Figure 28).

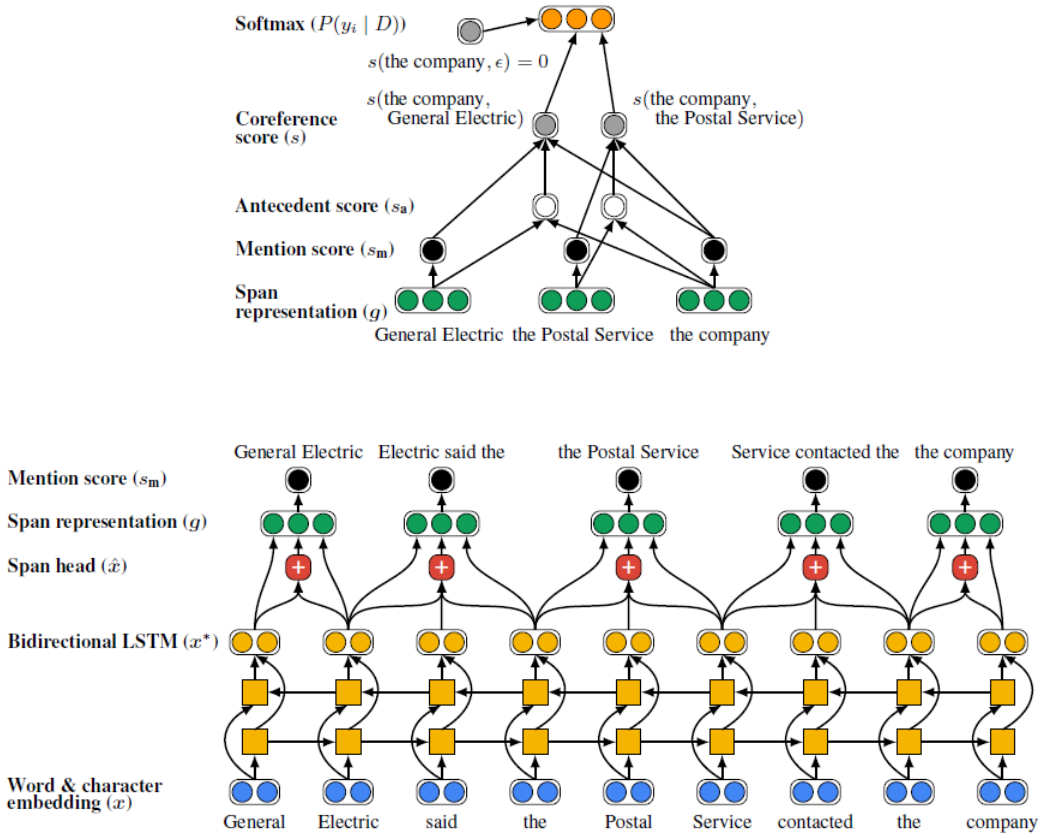


Figure 28. Illustration de l'architecture utilisée par Lee et al. (2017).

L'avènement des architectures d'encodage pré entrainées de type BERT a amélioré encore les résultats obtenus sur la tâche. L'architecture BERT (Joshi et al. 2019) ou un dérivé de celle-ci (Joshi et al. 2020) est utilisée en remplacement de la structure récurrente de cellules LSTM utilisée par Lee et al. (2017, 2018) et a permis d'améliorer de plus de six points le score F_1 obtenu sur la tâche de résolution de coréférences (Joshi et al. 2020).

Le Tableau 4 résume les différentes approches concernant la résolution de coréférences présentées dans ce document.

Tableau 4. Résumé des approches concernant la résolution de coréférences. Dans la section Modèles, les parenthèses servent à délimiter les différentes couches du modèle, typiquement la structure utilisée pour représenter les jetons, la couche d'encodage et la couche de décodage. Les scores F_1 marqués d'une étoile sont les scores obtenus avec l'indice B^3.

Étude	Modèle	Jeu de données	Score F1
Hobbs 1978	Règles	s.o.	s.o.
Lapin & Leass 1994	Règles	s.o.	s.o.
Soon et al. 2001	Caractéristiques + classificateurs binaires	MUC-6 MUC-7	62,6 60,4
Ng & Cardie 2002	Caractéristiques + classificateurs binaires	MUC-6 MUC-7	69,1 63,4
Yang et al. 2004	Caractéristiques + classificateurs binaires	MEDLINE	81,7
Rahman & Ng 2009	Caractéristiques + classificateurs SVM	ACE05	64,0*
Raghunathan et al. 2010	Règles	MUC-6	73,2*
Stoyanov & Eisner 2012	(Caractéristiques) + (FFNN)	MUC-6 ACE04	77,5* 81,8*
Ratinov & Roth 2012	(Caractéristiques) + (FFNN)	ACE04	83,0*
Wiseman et al. 2015	(Caractéristiques) + (FFNN) + (FFNN)	OntoNotes	60,5*
Clark & Manning 2016	(Plongements + caractéristiques + distance) + (FFNN) + (FFNN)	OntoNotes	62,9*
Wiseman et al. 2016	(Caractéristiques) + (RNN) + (FFNN)	OntoNotes	61,5*
Lee et al. 2017	(Plongements + distance + attention) + (Bi-RNN) + (FFNN)	OntoNotes	66,6*
Wu & Ma 2017	(Plongements + caractéristiques) + (CNN + FFNN) + (FFNN)	OntoNotes	75,3*
Lee et al. 2018	(Plongements + distance + attention + ELMo) + (Bi-RNN) + (FFNN)	OntoNotes	70,8*
Joshi et al. 2019	(BERT) + (FFNN)	OntoNotes	75,3*
Joshi et al. 2020	(BERT) + (FFNN)	OntoNotes	78,1*

3.2.4 La classification de relations

3.2.4.1 Les principes de la tâche

La classification de relations a été envisagée dès les conférences MUC (Template Relation – Chinchor & Marsh 1998) mais précisément définie lors du programme ACE (Relation Detection and Characterization, RDC – Doddington et al. 2004). La classification de relations est la tâche qui doit déterminer si une relation, un lien sémantique existe entre deux mentions et le cas échéant d'en caractériser la nature. La tâche a donc deux sous-tâches que sont la détection de relations d'une part et la classification de relations

proprement dite d'autre part. Ces deux sous-tâches sont traitées soit de manière hiérarchique, c'est-à-dire que la tâche de détection de relation est menée en premier pour identifier les échantillons au niveau desquels une relation existe, avant que la tâche de classification ne vienne caractériser la relation proprement dite, soit de manière parallèle, c'est-à-dire que la détection de relation est intégrée dans la tâche de classification en ajoutant une catégorie nulle aux catégories de relations pour désigner l'absence de relation (Sun et al. 2011).

Le programme ACE définit cinq catégories et 24 sous-catégories pour préciser la nature de la relation (Tableau 5 – Doddington et al. 2004). Le nombre et le type de ces catégories ont évolué par la suite dans les différents ensembles de données, mais en gardant les mêmes principes généraux.

Tableau 5. Liste des catégories et sous-catégories de la tâche de classification de relations définies par le programme ACE.

Catégorie	Sous-catégorie
Rôle (rôle que peut jouer une personne dans une organisation)	Direction
	État-major
	Membre
	Propriétaire
	Fondateur
	Client
	Affilié/Partenaire
	Citoyen de
	Autre
Composant (relation partitive)	Sous-type
	Partie de
	Autre
Localisation (localisation absolue)	Localisé à
	Basé à
	Résidence
Près (localisation relative)	Près
Social	Parent
	Fratrie
	Époux/Épouse
	Grand-parent
	Autre membre de la famille
	Autre connaissance
	Associé
	Autre relation professionnelle

La tâche ayant été définie dans le cadre du programme ACE dont les entités nommées constituaient le point de focalisation, les relations y sont envisagées du point de vue des entités nommées. Pour cette raison, elles envisagent les types de relations entre des personnes, des organisations et des lieux. Ces types de relations n'ont pas d'intérêt pour le cas qui nous concerne puisque nous nous intéressons à des entités généralement inanimées et dont nous voulons extraire des relations liées à l'élaboration d'un diagramme de classe

c'est-à-dire des relations classe-classe, classe-attribut, agrégat-agrégé, etc. Si les notions d'entités et de relations diffèrent, la manière de résoudre la tâche reprend par contre les mêmes principes de base : identifier les mentions qui représentent les entités et trouver le lien qui peut les unir dans un document.

Les ensembles de données annotés utilisés dans les approches supervisées sont moins nombreux que dans le cas de la reconnaissance des entités nommées. Il en existe deux qui ont servis de support à la très grande majorité des études sur le sujet. Le premier est ACE (Doddington et al. 2004) avec ses différentes versions (ACE, ACE 2003, ACE 2004, ACE 2005 et ACE 2007) dont ACE 2005 est la plus couramment utilisée. Le second est SemEval-2010 Task-8 (Hendrickx et al. 2010).

La classification de relations telle que décrite jusqu'à maintenant envisage de catégoriser les relations en fonction d'un certain nombre de catégories préalablement définies. Elle peut aussi être envisagée comme une tâche ouverte dont l'objectif est d'extraire et de caractériser des relations à partir du document directement et non pas en les classant parmi un ensemble de catégories préétablies (Konstantinova 2014), on parle alors davantage d'extractions de relations. L'extraction d'informations ouverte (Open Information Extraction) n'est pas traitée ici, car nous nous concentrons sur la classification de relations qui correspond à la tâche que nous voulons réaliser dans cette étude. Puisque la tâche est une tâche de classification, la performance des modèles est décrite à l'aide des trois métriques classiques que sont la précision, le rappel et la mesure F_1 (voir CHAPITRE 1).

3.2.4.2 La résolution de la tâche

a) *L'APPROCHE SYMBOLIQUE*

La définition de la tâche de classification de relations est relativement récente. Les études qui ont utilisé une approche basée sur les règles sont en nombre restreint et sont liées à la résolution de la tâche de relations (Template Relations) de la conférence MUC-7. Cette tâche est résolue de manière optionnelle suite à la reconnaissance d'entités nommées et la résolution de coréférences (Fukumoto et al. 1998) et éventuellement en tant que sous-tâche de l'analyse d'évènements (Template Events – Garigliano et al. 1998, Humphreys et al. 1998). Les règles sont des structures syntaxiques simples qui lorsqu'elles sont reconnues associent automatiquement deux mentions à une relation. Par exemple, si la structure implique une mention de type personne et une mention de type organisation, la relation est de type « employée de », ou si elle implique une mention de type organisation et une mention de type localisation, la relation est de type « localisé à ».

b) *L'APPROCHE PAR APPRENTISSAGE*

Les approches d'apprentissage automatique se sont basées sur deux stratégies pour déterminer la relation entre deux mentions : utiliser un vecteur de caractéristiques colligées sur les mentions et la structure lexicale, syntaxique et sémantique qui les lie, ou utiliser une fonction noyau (voir section 2.3.2) pour représenter de manière implicite la similitude de structure entre deux échantillons intégrant chacun deux mentions dont on veut déterminer la nature de la relation. Dans le premier cas, on veut classer un vecteur de caractéristiques;

dans le second on veut classer une structure. Dans les deux cas, un outil de discrimination est ensuite utilisé pour déterminer s'il existe une relation entre les deux mentions et éventuellement la classe à laquelle appartient la relation.

Les approches par vecteurs de caractéristiques reprennent les principes exposés dans les chapitres précédents. Un ensemble de caractéristiques lexicales, syntaxiques et sémantiques, accompagnées éventuellement de ressources externes comme une liste de pays (Zhou et al. 2005) collectées sur les deux mentions dont on veut déterminer la nature de la relation et leur environnement est rassemblé dans un vecteur. Celui-ci est ensuite traité par un outil de discrimination, que ce soit un modèle à entropie maximale (Kambhatla 2004), un modèle de classificateurs binaires multi-classes (Sun et al. 2011) ou une machine à vecteurs de support (SVM – Zelenko et al. 2003, Culotta & Sorensen 2004, Bunescu & Mooney 2005, Zhou et al. 2005). Puisque les classificateurs binaires et les machines à vecteurs de support sont des modèles de discrimination binaires, il est nécessaire d'adapter le modèle pour le conformer à la tâche de classification de relations qui est une tâche de classification multi-classes. Pour ce faire, le modèle est en fait composé d'un ensemble de machines à vecteurs de support, une pour l'existence d'une relation et une pour chaque classe potentielle, et les vecteurs sont analysés par chacune des machines. La classe (ou la décision dans le cas de l'absence de relation) concernée par la discrimination la plus probante est attribuée à l'échantillon (Zhou et al. 2005). Les caractéristiques choisies sont relativement similaires entre les études et impliquent la liste des mots des deux mentions, la liste des mots avant la première mention, entre les deux mentions, après la seconde mention, un indice de recoupement ente les deux mentions, les types de mentions

impliquées (Personne, organisation, lieu, etc.), le niveau de mention (nom, pronom ou expression synonyme) ainsi que plusieurs indices décrivant la structure de dépendance syntaxique unissant les deux mentions (Zhou et al. 2005).

Les approches par noyaux ont été développées pour limiter le temps passé en recherche de caractéristiques significatives permettant de discriminer les différents types de relations. Elles utilisent plusieurs modes de schématisation de la structure linguistique pour établir une fonction de similitude (la fonction noyau) entre les structures qui représentent un type de relation particulier et les structures à évaluer. Collins & Duffy (2001) sont les premiers à avoir montré l'intérêt des fonctions noyaux pour les différentes tâches du traitement des langages naturels. Leur fonction noyau est basée sur la représentation en arbre syntaxique des phrases (Figure 29a) et calcule un score de similitude entre deux structures en déterminant le nombre de sous arbres syntaxiques communs. L'approche a été reprise spécifiquement pour la résolution de la tâche de classification de relations, mais avec des fonctions noyaux légèrement différentes et en ajoutant un certain nombre d'attributs complémentaires aux nœuds de la représentation schématique pour en faire des approches hybrides entre noyaux et vecteurs de caractéristiques. Ces attributs complémentaires sont par exemple la nature du mot ou le type d'entité. La représentation schématique utilisée peut être des arbres syntaxiques superficiels (Zelenko et al. 2003), c'est-à-dire des arbres qui identifient les groupes de mots qui organisent la phrase, mais sans les décomposer en leurs composants élémentaires (Figure 29b), des arbres de dépendances augmentés (Culotta & Sorensen, 2004), c'est-à-dire des arbres qui identifient la structure hiérarchique qui unit les composants élémentaires de la phase, mais sans en

identifier la nature ou la fonction (Figure 29c) ou une simple lecture linéaire des mots de la phrase décomposée en trois volets : 1° les mots avant la première mention et entre les deux mentions, 2° les mots entre les deux mentions et 3° les mots entre les deux mentions et après la seconde mention (Bunescu & Mooney 2005). Chaque échantillon est constitué de l'arbre qui schématise la phrase incluant les deux mentions dont on veut déterminer la relation. Le score de similitude permet de classer les échantillons par rapport aux échantillons de référence en utilisant un classificateur binaire.

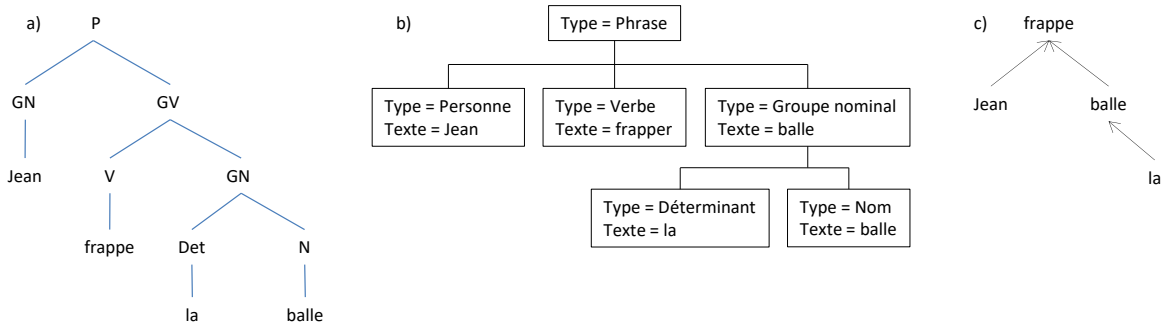


Figure 29. Illustration de différents modes de schématisation d'une phrase. a) arbre syntaxique, b) arbre syntaxique superficiel (inspiré de Zelenko et al. (2003)) et c) arbre de dépendance (inspiré de Culotta & Sorensen (2004)).

La tâche d'extraction de relation est une tâche de classification dont la seule entrée est une séquence de mots, au même titre que les autres tâches décrites dans les chapitres précédents. Les architectures d'apprentissage profond utilisées sont donc similaires et sont composées de modèles qui sont capables de tenir compte de l'aspect séquentiel de l'entrée. À la différence cependant de la détection des entités nommées, une classification n'est pas produite pour chaque jeton d'entrée, mais pour un groupe. Si les techniques d'apprentissage automatique étaient par nature destinées à traiter l'extraction de relations de manière binaire, l'apprentissage profond, quant à lui, permet un traitement multi-classes direct grâce

à l'utilisation d'une couche de décodage composée d'un FFNN avec une fonction d'activation de type softmax. Celle-ci permet de représenter la confiance relative que l'échantillon puisse appartenir aux différentes classes possibles et la classe attribuée à l'échantillon est la classe qui obtient le score le plus élevé :

$$\hat{y} = argmax_{c \in C} s_c(x)$$

où x est l'échantillon (généralement une phrase dans les ensembles d'entrainement utilisés), c est une classe et $s_c(x)$ est le score obtenu par la phrase pour la classe c.

Les modèles d'apprentissage profond utilisent tous une des deux architectures disponibles pour prendre en compte l'aspect séquentiel de l'entrée : les CNN ou les RNN. Les principales différences entre les études résident dans 1° la nature de l'information présentée au modèle et 2° la manière d'intégrer cette architecture de traitement séquentiel dans le modèle. Comme dans d'autres tâches de traitement du langage naturel, l'information utilisée en entrée s'est progressivement épurée. Les modèles d'apprentissage automatique reposaient sur des représentations basées sur des caractéristiques des mots (explicitement ou implicitement via les fonctions noyaux), il est donc logique que les modèles d'apprentissage profond qui leur ont succédé partent de ce type de représentations. Toutefois, contrairement à d'autres tâches, les modèles d'apprentissage profond d'extraction de relations ont rapidement utilisé les plongements des mots, possiblement sous l'influence de ce qui se faisait déjà dans les autres tâches. Les premières études utilisaient donc un assemblage, par concaténation (Zeng et al. 2014, Xu et al. 2015, Zhang et al. 2015) ou par produit dyadique (Gormley et al. 2015), composé de plongements des mots et de caractéristiques lexicales, syntaxiques et sémantiques. Ces caractéristiques sont

notamment les plongements des mots environnant les mentions d'intérêts (Zeng et al. 2014) ou encore les hyperonymes des deux mentions d'intérêts tels que référencés dans la base de connaissance WordNet (Zeng et al. 2014, Xu et al. 2015, Zhang et al. 2015). Les caractéristiques sont cependant progressivement éliminées des informations d'entrée, à l'exception de la caractéristique de position relative des mots de la phrase par rapport aux deux mentions d'intérêts qui est jugée primordiale et utilisée dans la plupart des études d'extraction de relations. L'encodage de cette information peut se faire sous différentes formes. Il peut se faire par la concaténation d'un vecteur encodant la position relative des mots des deux mentions, sorte de plongement de position (Zeng et al. 2014, dos Santos et al. 2015), dont les valeurs peuvent être ou non modifiées au cours de l'entrainement (tout comme les plongements de mots), ou il peut se faire par l'ajout de balise de reconnaissance des mentions d'intérêt de part et d'autre de celles-ci dans la phrase (<e1>, </e1> et <e2>, </e2>), balises qui sont encodées par leur propre plongement au même titre que les autres mots de la phrase au moment de leur entrée dans le modèle (Zhang & Wang 2015, Vu et al. 2016, Zhou et al. 2016).

Le modèle de traitement séquentiel (typiquement le CNN ou le RNN) n'a tout d'abord constitué qu'une partie de l'information utilisée pour réaliser la tâche. Le résultat obtenu par un CNN (Zeng et al. 2014) ou par un RNN bi-LSTM (Zhang et al. 2015) constitue alors un vecteur de caractéristiques de niveau phrastique qui est concaténé à un vecteur de caractéristiques de niveau nominal pour être traité par un FFNN et une fonction d'activation softmax. Le modèle de traitement séquentiel phrastique devient par la suite la couche pivot du traitement de la tâche alimenté de manière directe ou par l'intermédiaire

d'une couche d'attention par des vecteurs de plongements. CNN (dos Santos et al. 2015, Nguyen & Grishman 2015, Vu et al. 2016, Wang et al. 2016b), RNN unidirectionnels (Xu et al. 2015) ou bidirectionnels (Zhang & Wang 2015, Miwa & Bansal 2016, Vu et al. 2016, Zhou et al. 2016, Lee et al. 2019) ont été utilisés. Outre les différences dans la nature des réseaux, les études varient aussi sur la séquence d'analyse utilisée. Le choix le plus courant est d'utiliser la séquence de mots de la phrase directement et un traitement gauche-droite (pour les RNN unidirectionnels) et droite-gauche (pour les RNN bidirectionnels). L'autre choix qui a été fait est d'utiliser l'arbre de dépendance liant les deux mentions d'intérêts et un traitement bas-haut (pour les RNN unidirectionnels – Xu et al. (2015)) et haut-bas (pour les RNN bidirectionnels – Miwa & Bansal (2016)). Les études plus récentes ont commencé à intégrer une couche d'attention dans le modèle soit à la sortie du CNN (Wang et al. 2016b) ou du RNN (Zhou et al. 2016), soit de part et d'autre, en entrée et en sortie (Lee et al. 2019). Cette couche d'attention permet de sélectionner certaines caractéristiques importantes soit sur les mots qui constituent la phrase lorsqu'elle est employée en amont de la couche de traitement soit sur la séquence encodée de vecteurs cachés lorsqu'elle est employée en aval.

Comme dans le cas des deux tâches présentées plus tôt, l'état de l'art de la résolution de la tâche de classification de relations a été établi par les développements récents des structures d'encodage pré-entrainés BERT (Huang et al. 2019, Wu & He 2019, Wang & Lu 2020). La classification de relations est envisagée comme une tâche de classification dont le résultat dépend du contexte évoqué par la phrase, ce qui est similaire aux approches adoptées dans les deux autres tâches et à d'autres tâches du traitement automatique des

langues. Il est donc logique que BERT remplace ici aussi les structures d'encodage séquentielles classiques de l'apprentissage profond. BERT a ainsi permis d'améliorer les résultats de quelques points de score F_1 en quelques mois et constitue désormais le nouveau standard d'encodage séquentiel du traitement automatique des langues.

Le Tableau 6 résume les différentes approches concernant la résolution de coréférences présentées dans ce document.

Tableau 6. Résumé des approches concernant la classification de relations. Dans la section Modèles, les parenthèses servent à délimiter les différentes couches du modèle, typiquement la structure utilisée pour représenter les jetons, la couche d'encodage et la couche de décodage.

Étude	Modèle	Jeu de données	Score F1
Fukumoto et al. 1998	Règles	MUC-7	39,1
Garigliano et al. 1998	Règles + base de connaissance	s.o.	s.o.
Humphreys et al. 1998	Règles	s.o.	s.o.
Zelenko et al. 2003	Fonctions noyaux + classificateurs SVM	s.o.	s.o.
Culotta & Sorensen 2004	Fonctions noyaux + classificateurs SVM	ACE	45,8
Kambhatla 2004	Caractéristiques + classificateurs entropie maximale	ACE	52,8
Bunescu & Mooney 2005	Fonctions noyaux + classificateurs SVM	ACE	47,7
Zhou et al. 2005	Caractéristiques + classificateurs SVM	ACE	55,5
Sun et al. 2011	Caractéristiques + classificateurs binaires	ACE04	71,5
Zeng et al. 2014	(Caractéristiques + plongements) + (CNN) + (FFNN)	SemEval-2010 Task 8	82,7
dos Santos et al. 2015	(Position + plongements) + (CNN) + (FFNN)	SemEval-2010 Task 8	84,1
Gormley et al. 2015	(Caractéristiques + plongements) + (FFNN) + (FFNN)	ACE05 SemEval-2010 Task 8	63,5 83,4
Nguyen & Grishman 2015	(Position + plongements) + (CNN) + (FFNN)	ACE05 SemEval-2010 Task 8	61,3 82,8
Xu et al. 2015	(Caractéristiques + plongements) + (RNN) + (FFNN)	SemEval-2010 Task 8	83,7
Zhang et al. 2015	(Caractéristiques + plongements) + (Bi-RNN) + (FFNN)	SemEval-2010 Task 8	84,3
Zhang & Wang 2015	(Position + plongements) + (Bi-RNN + CNN) + (FFNN)	SemEval-2010 Task 8	79,6
Miwa & Bansal 2016	(Caractéristiques + plongements) + (Bi-RNN) + (FFNN)	ACE04 ACE05 SemEval-2010 Task 8	48,4 55,6 85,6
Vu et al. 2016	(Position + plongements) + (CNN) + (FFNN) (Position + plongements) + (Bi-RNN) + (FFNN)	SemEval-2010 Task 8	84,2 83,4
Wang et al. 2016b	(Position + plongements) + (CNN + attention) + (FFNN)	SemEval-2010 Task 8	88,0
Zhou et al. 2016	(Position + plongements) + (Bi-RNN + attention) + (FFNN)	SemEval-2010 Task 8	84,0
Huang et al. 2019	(BERT) + (FFNN)	s.o.	s.o.
Lee et al. 2019	(Position + plongements + attention) + (Bi-RNN + attention) + (FFNN)	SemEval-2010 Task 8	85,2
Wu & He 2019	(BERT) + (CNN) + (FFNN)	SemEval-2010 Task 8	89,2
Wang & Lu 2020	(RNN + BERT) + (4D-RNN + Attention) + (RNN)	ACE04 ACE05	63,3 67,6

3.3 SYNTHESE

L’extraction de modèles de domaine n’est pas une discipline nouvelle. Elle a déjà été tentée en ayant recours à des prétraitements et des jeux de filtres morphosyntaxiques de façon à extraire d’un cahier des charges les concepts et les relations qui les unissent. La grande difficulté de l’approche tient dans l’exhaustivité du jeu de règles employé. Pour être efficace, l’approche nécessite un jeu de règles qui couvre toutes les constructions possibles permettant d’exprimer un concept ou un lien entre concepts dans un langage donné, ce qui est virtuellement impossible et très demandant en temps. Mais, nous pouvons envisager de résoudre le problème en ayant recours à l’apprentissage automatique pour s’affranchir de la découverte manuelle de ces règles, notamment en combinant les résultats issus de trois tâches du traitement automatique des langues que sont la reconnaissance d’entités, la résolution de coréférences et la classification de relations.

CHAPITRE 4

L'ÉTUDE

Cette étude est une proposition de résolution de la tâche d'extraction d'un modèle à partir d'un cahier des charges par les outils de l'apprentissage profond. Il existe plusieurs manières d'envisager la résolution de cette tâche. Notre proposition est adaptée à l'utilisation de techniques déjà existantes dont les résultats s'améliorent d'année en année. À titre de proposition, la démarche envisagée est un résultat de l'étude au même titre que les résultats en eux-mêmes. Le quatrième chapitre est consacré à la description de cette démarche. Il présente le modèle que nous avons utilisé, le jeu de données que nous avons élaboré, les résultats que nous avons obtenus et les conclusions que nous pouvons en tirer.

4.1 LA DESCRIPTION DE L'APPROCHE

L'approche développée dans cette étude consiste à regrouper la résolution de différentes tâches de l'apprentissage profond de manière à réaliser un objectif de grande échelle qui est la découverte du modèle indépendant de plateforme correspondant aux exigences décrites dans un cahier des charges. La découverte du PIM peut se décomposer schématiquement en une étape de découverte des notions importantes qui constituent le modèle et une étape d'identification des rapports qui existent entre ces notions. En termes d'apprentissage automatique, on retrouve les prémisses des trois tâches décrites dans les chapitres précédents : la détection d'entités pour découvrir les notions, la classification de

relations pour identifier les rapports et la résolution de coréférences pour faire le lien entre les différentes occurrences d'une entité dans un document. Le produit obtenu par la résolution de ces trois tâches doit fournir une représentation schématique du document, constituée de grappes de mentions qui forment les notions du document et d'un ensemble de relations entre ces grappes. Cette représentation est suffisante pour décrire un modèle de base du système.

Nous proposons de résoudre ces trois tâches de manière conjointe et non pas de manière indépendante, en construisant un modèle global entrainé sur les trois tâches. Le recours à des modèles de résolution conjoint à montrer qu'il était possible de bénéficier de l'entrainement d'un modèle sur différentes tâches de manière à améliorer l'efficacité du modèle quant à la résolution de chacune des tâches (Durett & Klein 2014, Lee et al. 2017). Notre modèle est donc constitué d'une couche d'encodage commune et trois couches de décodages distinctes et spécifiques à chaque tâche. Le principe du modèle reprend celui développé par Lee et al. (2017) mais adapté à la résolution des trois tâches que nous avons identifiées.

4.1.1 La tâche de détection d'entités

La tâche de détection d'entités est une tâche d'étiquetage similaire aux tâches de reconnaissance d'entités nommées (NER) décrites dans le CHAPITRE 3. Les approches de NER ont généralement fait appel à un ensemble d'étiquettes BIO ou BILOU pour catégoriser les différents mots d'un texte auxquelles est ajouté un descripteur désignant la nature de la mention (organisation, personne, date, etc.). Les étiquettes BIO et surtout

BILOU ont l'avantage de bien identifier le début et la fin des entités, notamment dans le cas d'entités imbriquées. Par contre, elles n'identifient pas le mot principal du groupe nominal (head) qui doit être détecté en utilisant une autre méthode (un module d'attention par exemple). De plus, la nature des mentions ne présente pas d'intérêt dans le cadre de notre objectif. Pour ces deux raisons, nous avons décidé d'annoter nos cahiers des charges en utilisant un autre système d'étiquetage HRO (Head, Relative, Out) sans spécification supplémentaire de manière à augmenter la représentativité de chaque étiquette dans nos documents et à identifier le mot principal d'un groupe nominal sans avoir recours à un modèle complémentaire. Chaque mot i se voit donc attribuer une étiquette x_i appartenant à l'ensemble $\mathrm{X} = \{H, R, O\}$.

4.1.2 La tâche de résolution de coréférences

La tâche de résolution de coréférences est abordée suivant l'approche de classement de grappes (cluster-ranking). Dans ce contexte, la tâche peut être envisagée comme une tâche de classification dans laquelle chaque mention se voit attribuer une grappe antécédente parmi l'ensemble des grappes déjà constituées depuis le début de l'analyse du document, incluant l'antécédent nul ε. Si chaque grappe potentielle est représentée par son numéro d'apparition dans le document, la tâche revient à assigner à la mention i la classe représentant la bonne grappe antécédente dans l'ensemble $\Psi_i = \{\varepsilon, 1, \dots, l\}$. À l'inverse des autres tâches, la taille de l'ensemble des classes potentielles augmente avec chaque nouvelle grappe découverte au fur et à mesure de l'analyse du document. La première mention doit ainsi se voir attribuer la bonne classe dans l'ensemble $\Psi_1 = \{\varepsilon\}$, ce qui est immédiat. Puisque cette première mention se voit attribuer la classe ε, elle devient le

premier élément d'une nouvelle grappe. De ce fait, l'ensemble des classes potentielles pour la deuxième mention est maintenant $\Psi_2 = \{\varepsilon, 1\}$. À chaque fois qu'une mention se voit attribuer la classe ε, l'ensemble des classes potentielles s'agrandit d'un élément pour les mentions subséquentes. Cette caractéristique contraint de traiter chaque mention de manière individuelle et non pas en lot. Par contre, le lien de coréférence n'est envisagé dans le cadre de cette étude qu'entre les mentions de la phrase en cours de traitement et les grappes découvertes dans le traitement des phrases précédentes. Les liens de coréférences éventuels entre mentions de la même phrase sont traités par la tâche de classification de relations. Il s'agit d'une relation uniquement « inter-phrase », ce qui implique que toutes les mentions d'une phrase sont évaluées par rapport au même ensemble de grappes potentielles antécédentes.

4.1.3 La tâche de classification de relations

La tâche de classification de relations est, quant à elle, une tâche classique de classification dans laquelle chaque paire de mentions se voit attribuer une classe désignant la relation la plus susceptible de caractériser le lien qui les unit. La liste des relations possibles est composée des éléments « aucune », « identité », « classe-classe », « classe-attribut », « attribut-classe », ce qui correspond à l'ensemble $\Omega = \{\varepsilon, 1, 2, 3, 4\}$ en attribuant un numéro à chaque relation. Ces relations sont les relations élémentaires qui permettent de constituer un diagramme de classes cohérent. Il existe d'autres notions importantes pour l'élaboration d'un diagramme de classes complets, telles que l'héritage, l'agrégation, les opérations, les multiplicités, mais nous nous sommes concentrés sur les quatre relations citées du fait du trop faible nombre d'exemples des autres notions dans

notre jeu de données et pour des raisons de simplification (voir la section de discussion). Les quelques exemples de relations d'héritage et d'agrégation ont été inclus dans la relation générale « classe-classe », tandis que les notions d'opérations et de multiplicités sont simplement ignorées.

À l'inverse de la résolution de coréférences, la classification de relations n'est envisagée qu'à l'échelle d'une phrase. Il s'agit ici d'une relation uniquement « intra-phrase » qui ne concerne que les mentions de la phrase en cours d'analyse. Cette idée part du principe que si un lien doit être exprimé entre deux entités, il le sera au sein d'une phrase du document impliquant deux expressions qui désignent les entités concernées pour des raisons de clarté du document. Elle permet aussi de limiter le nombre de paires de relations à envisager : si l'unité d'analyse contient n mentions, le nombre de paires à traiter est de C_n^2. Si un document contient quatre phrases, chacune constituée de trois mentions et que l'échelle d'analyse est placée au niveau du document, le nombre de paires à analyser sera de $C_{12}^2 = 66$, alors que si elle est placée au niveau de la phrase, ce nombre sera de $4 \times C_3^2 = 12$.

4.1.4 La description du modèle

Le modèle proposé (Figure 30 et ANNEXE I pour le code) est composé d'une couche d'encodage commune à la résolution des trois tâches et de trois couches de décodages distinctes, une pour chaque tâche à résoudre.

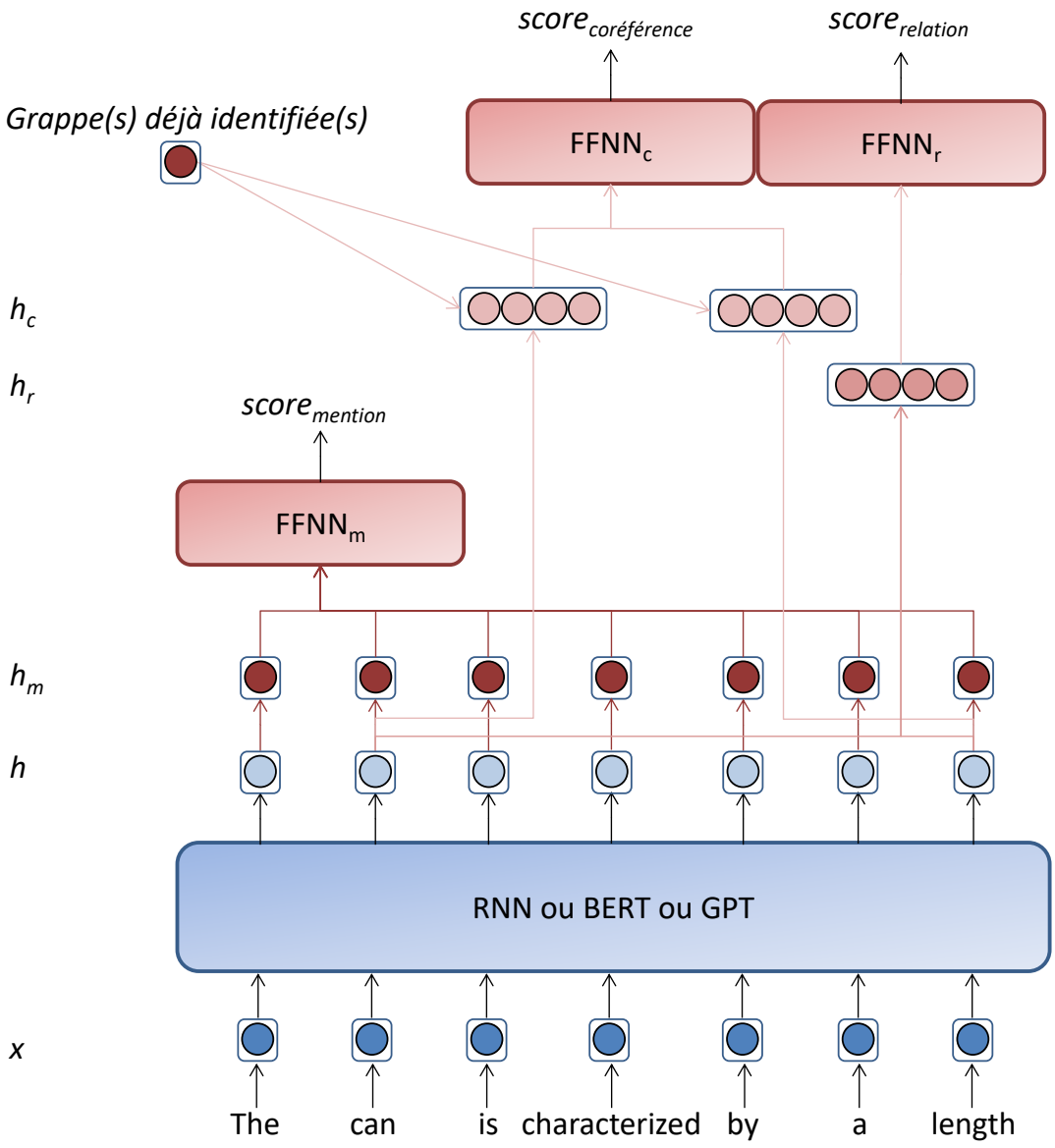


Figure 30. Illustration du modèle.

La couche d'encodage du modèle est destinée à produire une représentation vectorielle de chaque jeton (mots ou signes de ponctuations) d'une phrase et du contexte dans lequel il apparait, c'est-à-dire des jetons qui l'entourent dans la phrase. Le vecteur produit par cette couche d'encodage est ensuite utilisé comme base d'analyse dans les trois couches de décodage.

De manière plus précise, chaque phrase du document entre dans la structure d'encodage sous forme d'une séquence de jetons (x). Il en ressort une séquence de vecteurs d'encodage, un pour chaque jeton (h). Les vecteurs d'encodage produits subissent alors un ensemble de transformation et de décodage qui dépend de la tâche et du résultat obtenu au niveau de la tâche de détection d'entités.

Nous avons comparé trois types de couches d'encodage différentes pour cette étude. Le premier type, qui constitue la référence, est un RNN bidirectionnel constitué de cellules LSTM (RNN bi-LSTM) appliqué sur les plongements individuels des jetons. Les plongements utilisés sont les plongements GloVe (Pennington et al. 2014) issus de l'entrainement sur une base de données de 6 milliards de mots (Glove.6B). Chaque jeton se voit attribuer le plongement qui le représente issu de GloVe et c'est ce plongement qui entre dans le RNN bi-LSTM. Le vecteur encodé produit par cette couche est la concaténation des deux vecteurs cachés issus des deux sens de parcours du réseau bidirectionnel :

$$h = [\vec{h}, \overleftarrow{h}]$$

Le deuxième type est un réseau BERT. Chaque jeton est encodé par l'analyseur lexical puis traité par un modèle BERT afin de produire un vecteur encodé. Le modèle BERT choisi est le modèle « base-uncased » produisant un vecteur de 768 dimensions issu de la bibliothèque https://huggingface.co/transformers.

Le troisième type est un réseau GPT. Comme dans le cas du modèle BERT, un analyseur lexical encode les jetons avant que ceux-ci n'entrent dans le modèle GPT. Le vecteur produit est également un vecteur de 768 dimensions. Nous avons utilisé le modèle GPT-2 dans le cadre de cette étude, car il est rendu disponible publiquement (https://huggingface.co/transformers) contrairement au modèle GPT-3.

Le vecteur h de tous les jetons entre ensuite, sans transformation (h_m), dans la structure de décodage spécifique à la détection d'entités. Il se voit attribuer un $score_{mention}$ qui détermine si le jeton correspondant est un composant d'une mention intéressante (classe 'H', 'R') ou s'il s'agit d'un jeton de liaison inutile pour la constitution de notre modèle (classe 'O'). S'il s'agit d'un jeton inutile, le traitement s'arrête ; sinon son vecteur h va entrer dans le processus de décodage relatif aux deux autres tâches. La décision quant à l'utilité d'un jeton se base sur la classe déterminée par la tâche de détection d'entités. Par conséquent, le reste du processus dépend de la validité de ce premier résultat et les erreurs commises à ce moment-là ont des répercussions sur les autres tâches, notamment si un jeton supposé inutile se voit attribuer une étiquette 'H' ou 'R' ou à l'inverse si un jeton utile se voit attribuer une étiquette 'O'. Pour ne pas altérer l'entrainement du modèle, la décision durant la phase d'entrainement ne repose pas sur le résultat réel de la première tâche, mais sur les résultats attendus, ce qui permet de ne pas propager l'erreur commise au niveau de la tâche de détection d'entités dans l'entrainement des deux autres tâches. Au cours de la phase de prédiction, par contre, ce sont réellement les résultats obtenus lors de la première tâche qui dictent l'utilité d'un jeton et toute erreur commise à ce moment a des répercussions sur le reste du traitement. Les jetons utiles qui

constituent une mention sont regroupés et vont participer au décodage des deux autres tâches. Ces deux tâches ne portent pas sur la classification d'un jeton, mais sur la classification d'une paire de mentions composées d'un ou de plusieurs jetons. Il s'agit donc de trouver une manière d'encoder les mentions à partir des vecteurs encodés h des jetons qui la constituent. Nous avons comparé quatre techniques d'encodage des mentions.

La première technique, la plus simple et la référence, consiste à prendre le vecteur encodé du jeton central (classe 'H') comme encodage de l'ensemble de la mention. Ceci constitue effectivement une simplification de l'information dans le cas d'une mention de plusieurs jetons.

La deuxième technique consiste à concaténer l'encodage de tous les jetons qui composent la mention. Ceci peut créer un problème de dimension puisque toutes les mentions ne sont pas composées du même nombre de jetons. Pour remédier à cet état, les vecteurs d'encodage sont complétés avec des 0 jusqu'à la taille maximale d'un vecteur d'encodage (padding) dont la dimension est égale à :

$$dim = n_{max} \times dim_h$$

où n_{max} est la taille maximale d'une mention en nombre de jetons et dim_h est la dimension de l'encodage d'un jeton.

La troisième technique consiste à produire un vecteur d'encodage à partir d'une moyenne pondérée des encodages des jetons de la mention. Le jeton central est considéré

comme le plus important, son coefficient est donc plus grand que celui des autres jetons. Typiquement, tous les jetons secondaires se voient attribuer un coefficient de 1 tandis que le jeton central a un coefficient égal au nombre de jetons qui composent la mention. Les coefficients sont ensuite normalisés grâce à une fonction softmax. Les vecteurs d'encodage des jetons de la mention sont multipliés par leurs coefficients respectifs et additionnés termes à termes de façon à produire l'encodage de la mention. Ce paramétrage permet de tenir compte de la prépondérance de l'information apportée par le jeton central, tout en tenant compte de l'information complémentaire fournie par les autres jetons.

La quatrième technique reprend le principe de la troisième, mais en pondérant suivant des coefficients déterminés par un module d'auto attention (self-attention). Ce module d'attention reprend le module utilisé par Lee et al. (2017) et permet de déterminer les coefficients à affecter à chaque encodage de manière dynamique :

$$\alpha_t = FFNN_\alpha(h_t)$$

$$coef_{mention,t} = \frac{e^{\alpha_t}}{\sum_{k=0}^{n} e^{\alpha_k}}$$

$$h_{mention} = \sum_{t=0}^{n} coef_{mention,t} \cdot h_t$$

Le réseau $FFNN_\alpha$ fait partie de la structure du modèle qui est entrainée au fur et à mesure de l'entrainement.

Une fois l'encodage des mentions réalisé, reste à construire le vecteur qui va entrer dans la couche de décodage des deux tâches (h_c pour la résolution de coréférences et h_r pour la classification de relations). Étant donné que pour ces deux tâches, la classification porte sur une paire de mentions, le vecteur à décoder doit être une représentation du couple de mentions. Il est donc constituer de la concaténation de l'encodage de la première mention, de l'encodage de la seconde mention, de la multiplication terme à terme des deux encodages et d'un plongement de distance qui sépare les deux mentions. Le plongement de distance encode la distance qui sépare le dernier mot de la première mention du premier mot de la seconde. L'implication d'un paramètre de distance entre les expressions a déjà prouvé son intérêt dans les tâches de résolution de coréférences (Lee et al. 2017) et de classification de relations (Zeng et al. 2014, Nguyen & Grishman 2015, Lee et al. 2019). Intuitivement, il représente l'idée que deux expressions qui sont proches dans le document sont plus susceptibles de partager un lien sémantique que deux expressions plus distantes. Ce plongement est modifié par l'algorithme de rétro propagation de l'erreur au cours du processus d'entrainement. Dans le cas de la résolution de coréférences, chaque mention d'une phrase est comparée à une mention des grappes déjà formées et à l'antécédent nul. La phase de décodage produit un vecteur $score_{coréférence}$ dont la valeur la plus élevée indique l'antécédent le plus probable. Pour les grappes déjà formées, on prend toujours le vecteur encodé de la première mention rencontrée dans le document comme représentation (la représentation jugée la plus explicite de la grappe parce qu'étant la première, c'est elle qui doit le mieux indiquer la nature du concept), mais en lui conférant l'indice de position de la dernière mention de la grappe rencontrée. Dans le cas de la classification de relations, chaque mention d'une phrase est comparée à une mention la précédant dans la même

phrase. La phase de décodage produit un vecteur $score_{relation}$ dont la valeur la plus élevée indique la classe de relation la plus probable

Les trois tâches constituent des tâches de classification. Par conséquent, les différentes couches de décodage possèdent une structure similaire de FFNN constitué de deux couches linéaires avec une fonction d'activation Relu et d'une couche linéaire finale dotée d'une fonction d'activation softmax. La fonction softmax normalise le score obtenu pour chaque catégorie :

$$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = P(y_i)$$

L'objectif pour chaque tâche est de maximiser la fonction de vraisemblance (likelihood) de la tâche.

$$L = \prod_i P(y_i)$$

Ceci revient à minimiser la fonction de log-vraisemblance négative (negative log-likelihood) qui constitue la fonction objectif (loss function).

$$NLL = -\log \prod_i P(y_i) = -\sum_i \log P(y_i) = -\sum_i \log \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Dans le cas de la tâche de reconnaissance d'entités, c'est directement le vecteur encodé d'un mot qui entre dans la couche de décodage :

$$score_{mention}(i) = FFNN_m(h_{m_i}) = FFNN_m(h_i)$$

Dans le cas des deux autres tâches, le vecteur qui entre dans la couche de décodage concatène l'information de deux mentions :

$$score_{coreference}(ij) = FFNN_c\left(h_{c_{ij}}\right) = FFNN_c([h_i, h_j, h_i * h_j, d_{ij}])$$

$$score_{relation}(ik) = FFNN_r(h_{r_{ik}}) = FFNN_r([h_i, h_k, h_i * h_k, d_{ik}])$$

4.2 LE JEU DE DONNEES

4.2.1 La composition du jeu de données

Le jeu de données est composé de 20 documents comportant de 178 à 1 537 jetons chacun pour un total de 10 604 jetons. Il est divisé en trois ensembles, un ensemble d'entrainement comprenant 14 documents (8 258 jetons) soit 70% de l'ensemble des documents, un ensemble de validation comprenant trois documents (1 572 jetons) soit 15% de l'ensemble des documents et un ensemble de test comprenant également trois documents (784 jetons). L'annotation du document a été réalisée par une personne, ce qui permet d'assurer une certaine homogénéité du travail.

4.2.2 L’annotation du jeu de données

Chaque document est composé d’un ensemble de phrases, chacune constituée de jetons. Chaque jeton est annoté de manière individuelle pour les trois tâches proposées. L’annotation de tous les documents respecte un standard propre à notre jeu de données et qui permet au modèle d’identifier la classe attribuée au jeton pour chacune des tâches (Figure 31). Ce standard est composé des règles suivantes :

- Règle 1 : chaque jeton occupe une ligne du document.

- Règle 2 : les phrases sont délimitées par des lignes vides.

- Règle 3 : les annotations de chaque jeton pour les trois tâches sont situées sur la même ligne que celui-ci, séparées du jeton et entre elles par un espace.

- Règle 4 : l’annotation des tâches de résolution de coréférences et de classification de relations n’est réalisée que si le jeton est le mot principal d’une mention, c’est-à-dire que son annotation pour la tâche de détection de mentions correspond à la lettre ‘H’ ; c’est ce jeton qui est le cœur de la mention et qui la représente dans l’annotation des deux autres tâches.

- Règle 5 : l’annotation pour la tâche de détection des mentions est constituée d’une lettre unique appartenant à l’ensemble des classes possibles pour cette tâche.

- Règle 6 : l'annotation pour la tâche de résolution de coréférences est constituée d'un nombre unique appartenant à l'ensemble des antécédents possibles.

- Règles 7 : l'annotation pour la tâche de classification de relations est constituée d'une série de nombres séparés par des tirets, indiquant la classe de relations qui existe entre le jeton et les mentions qui le précèdent dans la phrase. Chaque nombre doit appartenir à l'ensemble des classes possibles pour cette tâche. Les relations sont envisagées dans l'ordre chronologique d'apparition des mentions dans la phrase : le premier nombre désigne la classe de relation entre la présente mention et la première mention de la phrase, le deuxième nombre désigne la relation entre la présente mention et la deuxième mention de la phrase et ainsi de suite.

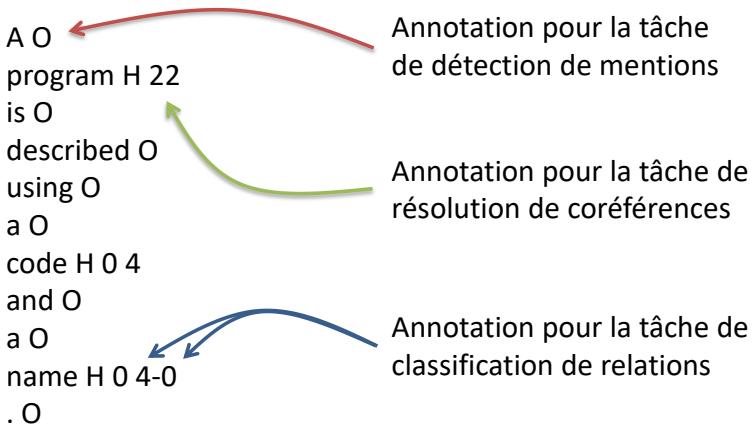


Figure 31. Illustration des principes d'annotation du jeu de données.

Dans l'exemple présenté dans la Figure 31, la phrase contient trois mentions de concepts importants du futur modèle, soit les mots 'program', 'code' et 'name'. Ceux-ci se voient donc attribuer une étiquette 'H' pour la tâche de détection de mentions (il n'y a pas d'étiquette 'R' puisque les trois mentions ne sont constituées que d'un seul mot), tandis que les autres mots se voient attribuer l'étiquette 'O' indiquant qu'ils ne sont pas à considérer dans l'élaboration du diagramme. L'étiquette 'H' des trois mentions précédentes est suivie d'un nombre indiquant si la mention considérée fait référence à un concept déjà rencontré dans le document. Le nombre 22 associé à la mention 'program' indique que cette mention n'est pas la première du concept 'program' et qu'elle fait référence au 22^{e} élément de la liste des concepts déjà rencontrés depuis le début du document. Les deux autres mentions, elles, sont les premières mentions de leurs concepts respectifs. Enfin, la liste de nombre qui suit les mentions 'code' et 'program' indique la classe de relations qui les unit aux mentions les précédant dans la phrase. Les chiffres 4, après 'code' et 'name', indique qu'il existe une relation de type « attribut-classe » entre ces deux mentions et la mention 'program', tandis que le chiffre 0 après 'name' indique qu'il n'y a pas de relation entre cette mention et la mention 'code'

L'exercice d'annotation peut paraitre au premier abord très mécanique, presque automatisable, mais le langage naturel est complexe et, par conséquent, ce qui est une difficulté pour le traitement par une machine devient aussi une difficulté pour l'annotation, et ce, pour la même raison : il est difficile de formaliser des idées qui sont écrites en utilisant des structures de langage complexes.

La première difficulté de l'exercice est d'annoter individuellement les jetons de manière à refléter l'organisation du modèle prévue par le document. Lorsque l'on réalise un diagramme de classes, le cerveau humain est capable de mettre en perspective certaines informations grâce à ce qu'il a déjà appris précédemment dans le document et grâce à l'expérience. Cela va au-delà du simple respect de la cohérence de l'annotation qui est purement logique. La réalisation d'un modèle fait appel à une certaine part d'intuition et d'automatisme qui viennent avec la connaissance des systèmes informatiques et avec la connaissance du fonctionnement des systèmes réels. Par exemple, si l'on parle d'un système de gestion du transport scolaire, on peut imaginer sans avoir lu une ligne du document que le modèle va impliquer des bus scolaires, des élèves, des horaires, des trajets, des établissements scolaires. La structure grossière du modèle est déjà en place sans avoir commencé l'exercice. Il n'est pas possible de transposer cette intuition dans l'exercice d'annotation qui oblige à suivre le cheminement du document et à espérer que tous les liens seront explicitement décrits. Mais, parfois, le document n'explicite pas certains liens parce que l'idée est suffisamment claire pour que le cerveau comprenne le lien sans avoir à le transposer à l'écrit. Ce sont ces relations qui sont difficiles à annoter et à détecter par un traitement machine. Dans un document qui décrirait un concept 'ordinateur' dans une phrase et qui parlerait du concept de 'micro-processeur' dans une autre, sans indiquer qu'il s'agit du 'micro-processeur' de 'l'ordinateur', l'humain comprend qu'il s'agit du 'micro-processeur' de cet 'ordinateur' et qu'il existe donc un lien entre les deux concepts (Figure 32).

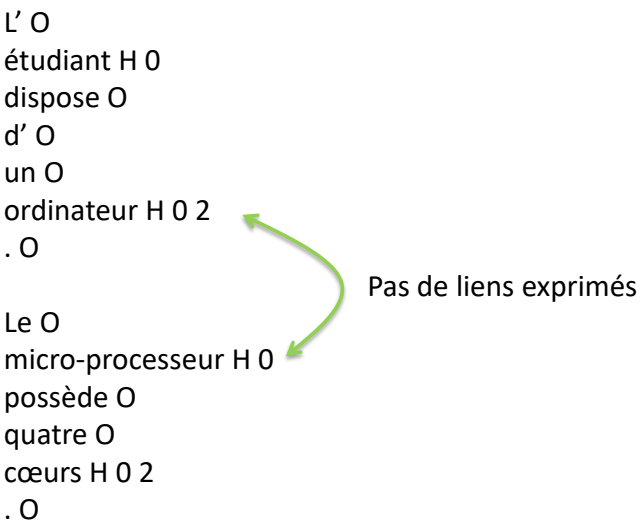


Figure 32. Illustration de la difficulté d’annoter certaines structures (1).

La deuxième difficulté est de maintenir la cohérence de l’annotation tout au long du document. Il arrive souvent qu’un concept puisse être considéré comme un attribut à la lecture d’une partie du document puis doive être considéré comme une classe à la lecture d’une autre partie. La question est de savoir s’il faut l’annoter comme une classe dans les deux cas, ce qui serait logique au vu de l’ensemble du document, ou comme un attribut d’abord puis comme une classe ce qui serait logique au vu des structures syntaxiques, mais qui induit une ambiguïté dans les connaissances finales extraites du document. Par exemple, si l’on décrit un concept ‘client’ et qu’on dit qu’il a une ‘adresse’, cette ‘adresse’ doit être considérée comme un attribut du concept ‘client’. Mais si l’on ajoute que cette ‘adresse’ est composée d’un ‘numéro civique’ et d’un ‘nom de rue’ alors ‘adresse’ se retrouve à avoir deux attributs, ce qui implique que le concept ‘adresse’ soit une classe (Figure 33). La seule information dont dispose le modèle pour discriminer la nature des concepts et leurs relations sont les structures syntaxiques du document. Si l’on souhaite que l’entrainement permette d’améliorer les capacités de détection du modèle, il faut être

cohérent avec les structures syntaxiques, ce qui implique de considérer un concept tour à tour comme un attribut puis comme une classe, ce que nous avons préconisé. Pour compenser les ambigüités nées de ce choix, nous avons établi la règle suivante : si un concept peut être considéré comme un attribut ou comme une classe, nous gardons la catégorie de plus haut niveau hiérarchique, en l'occurrence la classe.

Le O	Le O
client H 0	client H 0
a O	a O
une O	une O
adresse H 0 2	adresse H 0 4
composée O	composée O
d' O	d' O
un O	un O
numéro H 0 0-4	numéro H 0 0-4
civique R	civique R
et O	et O
d' O	d' O
un O	un O
nom H 0 0-4-0	nom H 0 0-4-0
de R	de R
rue R	rue R
. O	. O

Figure 33. Illustration de la difficulté d'annoter certaines structures (2).

La troisième difficulté est d'annoter les concepts qui sont désignés par les mêmes mots. Il arrive couramment que des attributs se retrouvent dans plusieurs classes. Par exemple l'attribut 'nom' est un concept très commun. On peut le retrouver dans deux classes 'animal' et 'bâtiment'. D'un point de vue sémantique et orthographique, il s'agit du même concept alors doit-on considérer qu'il s'agit du même concept qui est référencé deux fois, une fois lorsqu'on l'attribue au concept 'animal' et une fois lorsqu'on l'attribue au concept 'bâtiment' ou comme deux concepts différents ? Il est difficile d'imaginer qu'un

même attribut appartienne à deux classes différentes. Pour cette raison, nous avons considéré qu'il s'agissait de deux concepts (Figure 34). Néanmoins, au cours de l'entrainement sur la tâche de résolution de coréférences, on se retrouve à analyser la relation qui existe entre deux mentions identiques. Il est difficile de penser que le modèle puisse à tous les coups les ranger dans deux grappes différentes.

Un O animal H 0 possède O un O nom H 0 4 et O un O numéro H 0 4-0 de R puce R . O	Un O animal H 0 possède O un O nom H 0 4 et O un O numéro H 0 4-0 de R puce R . O
Un O bâtiment H 0 possède O un O nom H 2 4 et O une O adresse H 0 4-0 . O	Un O bâtiment H 0 possède O un O nom H 0 4 et O une O adresse H 0 4-0 . O

Figure 34. Illustration de la difficulté d'annoter certaines structures (3).

La quatrième difficulté est d'annoter les relations de généralisation entre classes. Comme nous l'avons mentionné, pour des raisons de simplification, nous avons décidé de ne pas extraire ce type de relation et de considérer qu'il s'agit de relations d'associations entre classes. Néanmoins, cela pose la question de savoir à qui affecter les attributs de ces classes. Il est bien évident que si le document parle d'un concept de 'véhicule' et indique

également que d'une part le concept de 'voiture' à un 'numéro de série' et un 'type de volant' et que d'autre part le concept de 'bateau' a un 'numéro de série' et un 'type de gouvernail', l'esprit humain comprend que le concept de 'véhicule' est la classe parente des classes 'voiture et 'bateau' et qu'elle va se voir confier l'attribut 'numéro de série' tandis que les deux autres classes auront le deuxième attribut qui leur est propre. En principe, l'attribut 'numéro de série' doit donc être attribué à la classe 'véhicule'. Comment annoter ce fait sans altérer la structure syntaxique de la phrase qui indique au contraire que les attributs devraient être rattachés aux deux concepts 'voiture' et 'bateau' ? Nous avons pris le parti de ne pas dénaturer la structure syntaxique des phrases de manière à régulariser la manière dont notre modèle apprend à reconnaitre la relation entre une classe et un attribut. Ceci à l'inconvénient de potentiellement altérer le diagramme de classes en répétant l'attribut 'numéro de série' dans les classes 'voiture' et 'bateau' et de ne pas attribuer d'attribut à la classe 'véhicule' (Figure 35). Puisque notre ensemble d'entrainement est d'une taille modeste, nous avons privilégié la redondance des structures syntaxiques plutôt que leur diversité.

Un O	Un O
véhicule H 0	véhicule H 0
est O	est O
une O	une O
voiture H 0 2	voiture H 0 2
avec O	avec O
un O	un O
numéro H 0 4-0	numéro H 0 0-4
de R	de R
série R	série R
et O	et O
un O	un O
type H 0 0-4-0	type H 0 0-4-0
de R	de R
volant R	volant R
ou O	ou O
un O	un O
bateau H 0 2-0-0-0	bateau H 0 2-0-0-0
avec O	avec O
un O	un O
numéro H 0 4-0-0-0-0	numéro H 0 0-0-0-0-4
de R	de R
série R	série R
et O	et O
un O	un O
type H 0 0-0-0-0-4-0	type H 0 0-0-0-0-4-0
de R	de R
gouvernail R	gouvernail R
. O	. O

Figure 35 Illustration de la difficulté d'annoter certaines structures (4).

La cinquième difficulté est d'annoter certaines structures syntaxiques particulières qui sont évidentes à comprendre pour l'esprit humain, mais qui sont délicates à formaliser. Les structures qui éludent certains mots pour des raisons de redondance en sont des exemples. Une expression telle que « l'heure d'entrée et de sortie de l'étudiant » est difficile à annoter parce que le mot 'heure' se rapporte à la fois au concept 'heure d'entrée' et au concept 'heure de sortie' tout en étant mentionné qu'une seule fois. Il existe plusieurs

manières d'envisager le problème : soit en dénaturant les concepts, soit en dénaturant l'idée qu'une mention ait nécessairement un mot d'entête et en élaborant une règle plus complexe pour recréer une mention complète à partir d'une annotation fragmentaire (Figure 36). Dans le premier cas, les concepts identifiés seront 'heure d'entrée' et 'sortie' ce qui est approximatif. Dans le second cas, les concepts seront bien 'heure d'entrée' et 'heure de sortie', mais la deuxième mention n'a pas de mots d'entête propre. Il faut donc ajouter une règle qui précise d'aller chercher son mot d'entête en amont ou en aval. Encore une fois, cela pose des difficultés pour savoir où aller chercher. Nous avons choisi la seconde solution, mais ceci a pour conséquence d'introduire dans l'entrainement du modèle des structures d'annotation différentes qui vont nécessairement altérer la performance du modèle.

l' O	l' O
heure H	heure H
d' R	d' R
entrée R	entrée R
et O	et O
de O	de R
sortie H	sortie R
de O	de O
l' O	l' O
étudiant H	étudiant H

Figure 36. Illustration de la difficulté d'annoter certaines structures (5).

4.3 LA VALIDATION DE L'APPROCHE

4.3.1 Les résultats

Chaque modèle a été entrainé avec différentes combinaisons d'hyper paramètres. Les taux d'apprentissage (valeurs testées de 0,1 – 0,01 – 0,005 et 0,001) et de dropout (valeurs testées de 0 – 0,1 – 0,2 et 0,5) ont fait l'objet de la première série de tests sur les différents modèles. Les valeurs les plus performantes ont été établies à 0,001 pour le taux d'apprentissage et à 0,2 pour le taux de dropout et ont été fixées pour la seconde série de tests qui portent sur des hyper paramètres plus spécifiques aux différents modèles. Pour les modèles de type RNN, les variations portent sur la taille du vecteur caché de la couche d'encodage (valeurs testées de 50 et 100), la taille du vecteur caché de la couche de décodage (valeurs testées de 100 et 150) et la taille du vecteur d'encodage de la distance entre les mentions utilisé dans le cadre de la résolution de coréférences et de classification de relations (valeurs testées de 20 et 50). Pour les modèles de type BERT et GPT, les variations portent sur le nombre de couches du modèle gelées (c'est-à-dire le nombre de couches dont les paramètres sont laissés intacts par rapport à la valeur initiale qu'ils ont suite à leur pré entrainement – valeurs testées de 5, 10, 12, 13 et toutes pour BERT et de 12, 13, 14, 15 et toutes pour GPT), la taille du vecteur caché de la couche de décodage (valeurs testées de 100 et 150) et la taille du vecteur d'encodage de la distance entre les mentions (valeurs testées de 20 et 50). Les résultats pour chaque combinaison de paramètres sont présentés dans l'ANNEXE II dans le cas du modèle RNN, dans l'ANNEXE III dans le cas du modèle BERT et dans l'ANNEXE IV dans le cas du modèle GPT.

Pour chaque itération, le niveau d'entrainement optimal du modèle est déterminé par la méthode de l'arrêt précoce (early stopping). L'idée est d'entrainer le modèle jusqu'au moment où il entre en phase de surentrainement, c'est-à-dire le moment où l'erreur commise sur l'ensemble d'entrainement continue de diminuer, mais où l'erreur commise sur l'ensemble de validation commence à augmenter. Avant ce point, le modèle n'est pas assez entrainé et sa performance peut encore s'améliorer ; au-delà de ce point, le modèle est trop adapté aux données d'entrainement et ses performances sur des données nouvelles diminuent (voir CHAPITRE 1). Pour connaitre le moment où la performance du modèle sur l'ensemble d'entrainement se détériore, nous avons choisi de poursuivre l'entrainement pendant trois époques après un minimum pour vérifier qu'un nouveau minimum n'était pas atteint. Si aucune amélioration n'est constatée sur trois époques, l'entrainement s'arrête et le modèle conservé pour le test est le modèle correspondant à la meilleure époque. Puisque nous réalisons trois tâches de manière conjointe, nous avons construit un indicateur hybride pour évaluer la performance des modèles. Cet indicateur est calculé à partir du facteur de perte calculé sur la réalisation des trois tâches. Comme la perte est plus importante pour la résolution de coréférences que pour la classification de relations (d'un facteur quatre selon une approximation sur des résultats destinés à construire l'indicateur) et la détection de mentions (d'un facteur sept), nous avons ajouté un facteur d'échelle. Ainsi, la perte associée aux deux dernières tâches a été multipliée par un coefficient de manière à ce que les facteurs de perte sur les trois tâches participent de manière égale au calcul :

$$Indicateur = 7 \times Perte_{mention} + Perte_{coreference} + 4 \times Perte_{relation}$$

L'évolution des facteurs de perte sur les trois tâches est illustrée avec la Figure 37.

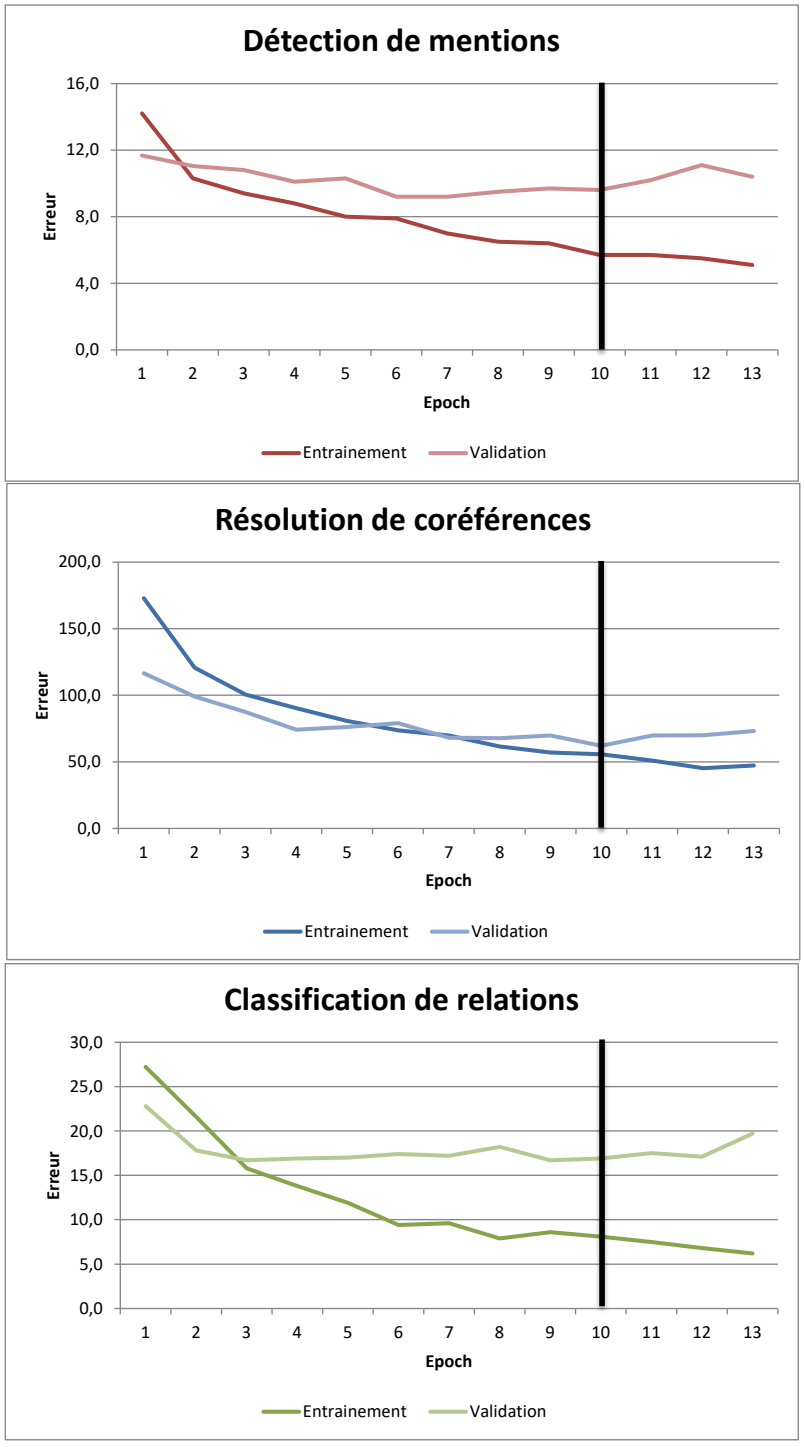


Figure 37. Illustration de l'évolution du facteur de perte calculé pour les trois tâches sur les ensembles d'entrainement et de validation. Le trait noir indique l'époque où le modèle optimal a été obtenu.

Les meilleurs résultats obtenus pour chaque combinaison de chaque modèle sont résumés dans le Tableau 7.

Tableau 7. Résultats obtenus sur les trois tâches par les différents sous-modèles. Les résultats de la résolution de coréférence correspondent à l'indice B^3.

	Détection de mentions			Résolution de coréférences			Classification de relations		
	Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
RNN + Mot central (Baseline)	86,0	86,2	86,0	45,6	81,2	58,4	72,9	67,8	69,5
RNN + Rembourrage	86,5	84,8	85,5	48,3	81,6	60,6	69,1	64,2	65,9
RNN + Moyenne pondérée	88,0	86,0	86,7	45,8	82,9	59,0	72,0	68,7	69,1
RNN + Auto-attention	87,1	86,0	86,4	46,6	78,8	58,6	69,9	63,9	65,5
BERT + Mot central	90,7	89,5	89,9	44,4	76,0	56,0	73,4	68,4	69,8
BERT + Rembourrage	90,4	87,2	88,3	47,2	78,8	59,0	70,3	68,1	68,7
BERT + Moyenne pondérée	**90,7**	**88,6**	**89,3**	**45,6**	**68,9**	**54,9**	**75,9**	**72,6**	**73,6**
BERT + Auto-attention	90,7	88,5	89,2	45,3	69,0	54,7	73,3	69,9	70,9
GPT + Mot central	74,9	77,7	76,0	49,6	100,0	66,3	67,8	65,7	65,1
GPT + Rembourrage	73,2	78,6	75,7	49,8	100,0	66,5	66,2	68,1	65,0
GPT + Moyenne pondérée	71,4	77,3	72,0	49,4	100,0	66,2	68,3	66,0	66,0
GPT + Auto-attention	72,9	77,6	75,2	49,8	100,0	66,5	67,8	68,4	61,6

Le sous-modèle qui obtient les meilleurs résultats combinés sur les trois tâches est la combinaison d'une couche d'encodage de type BERT et de l'utilisation d'une moyenne pondérée pour l'encodage des mentions. De manière plus générale, les modèles BERT obtiennent tous des résultats cumulés supérieurs ou égaux aux deux autres modèles, les modèles RNN étant les deuxièmes plus performants et les modèles GPT les troisièmes. Ce constat cache cependant une autre réalité : les différents modèles ont des performances hétérogènes sur les trois tâches. Les modèles BERT réussissent particulièrement bien la détection de mentions et la classification de relations qui sont deux tâches de classification pures, mais sont moins performants que les modèles RNN et les modèles GPT sur la tâche de résolutions de coréférences. Les modèles RNN sont quant à eux plus équilibrés puisqu'ils réussissent bien les trois tâches. Nous constatons également que les tâches de résolution de coréférences et de classification de relations influencent les modèles de

manière opposée : lorsqu'un modèle montre de bons résultats sur la première tâche, il en montre de moins bons sur la seconde et ceci se vérifie à la fois entre les modèles et entre les sous-modèles, et de manière plus générale et subjective d'une itération à l'autre.

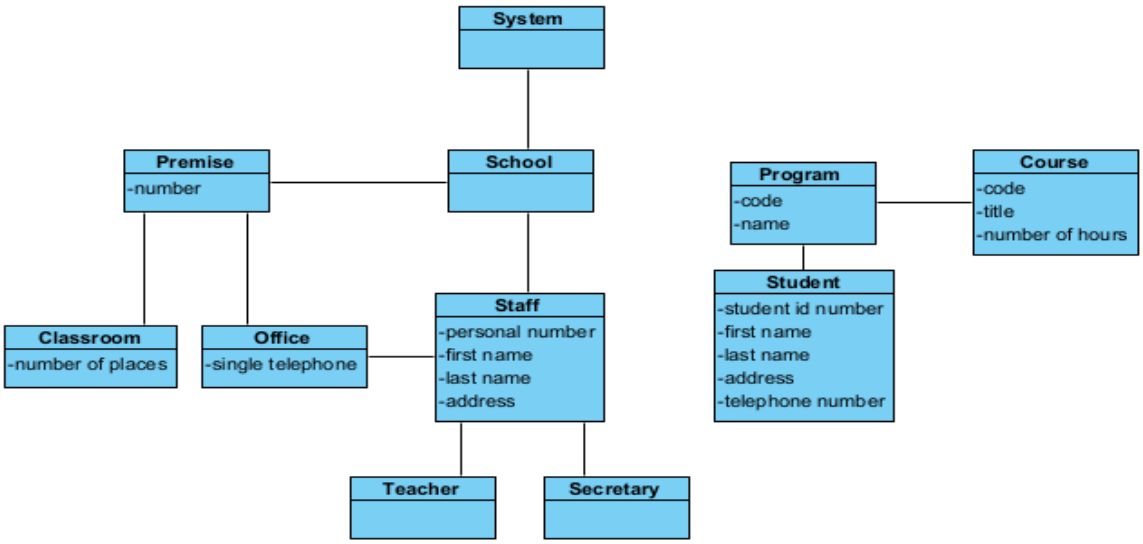
La conclusion quant à la meilleure technique d'encodage des mentions est plus difficile à dresser parce que les résultats ne sont pas constants entre les trois modèles. La moyenne pondérée a permis d'obtenir les meilleurs scores pour les modèles RNN et BERT, mais pas pour les modèles GPT ou c'est le mot central qui a le mieux performé. Par contre le recours au module d'auto attention pour l'encodage des mentions a obtenus les moins bons scores pour les trois modèles ce qui suggère une moindre efficacité générale.

En se fiant sur ces chiffres, il est néanmoins difficile de déterminer lequel des modèles est le plus efficace pour produire un diagramme de classes : est-ce-que ce sont les modèles BERT qui présentent les meilleurs résultats cumulés, mais le plus bas score F_1 pour la résolution de coréférence, les modèles RNN qui présentent les résultats les plus équilibrés ou les modèles GPT qui présentent les meilleurs scores de résolution de coréférences ? Nous avons construit les diagrammes de classes obtenus avec les meilleurs résultats des trois modèles pour tenter de répondre à cette question. Cette expérimentation a été conduite sur un des cahiers des charges de test identique pour les trois modèles, dont l'énoncé est le suivant :

> We want to build a system to computerize part of the management of a school. The staff of this school is composed of teachers and secretaries. Each member of staff is identified by a personnel number and is described by a last name, a first name and an address. The school is made up of premises (identified by a number) which are either offices (in this case, they are provided with a single telephone), or classrooms (which include a certain number of places). An office is occupied by a

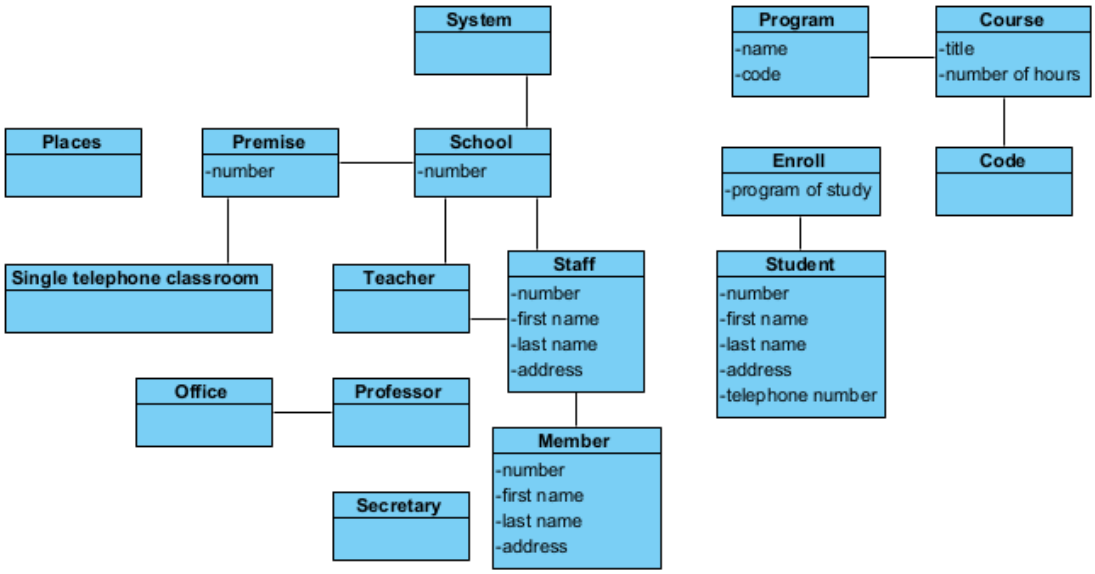
> professor and / or several secretaries. A student has a student ID number. He is described by a last name, a first name, an address and a telephone number. A student enrolls in only one program of study. A program is described using a code and a name. A study program consists of a set of courses (described by code, title, and number of hours).

Les modèles entrainés ont été utilisés pour prédire les grappes de mentions et leurs relations entre elles et ainsi constituer la structure du diagramme de classes. À titre de comparaison, le diagramme théorique tel que constitué par l'annotateur est également présenté (Figure 38).



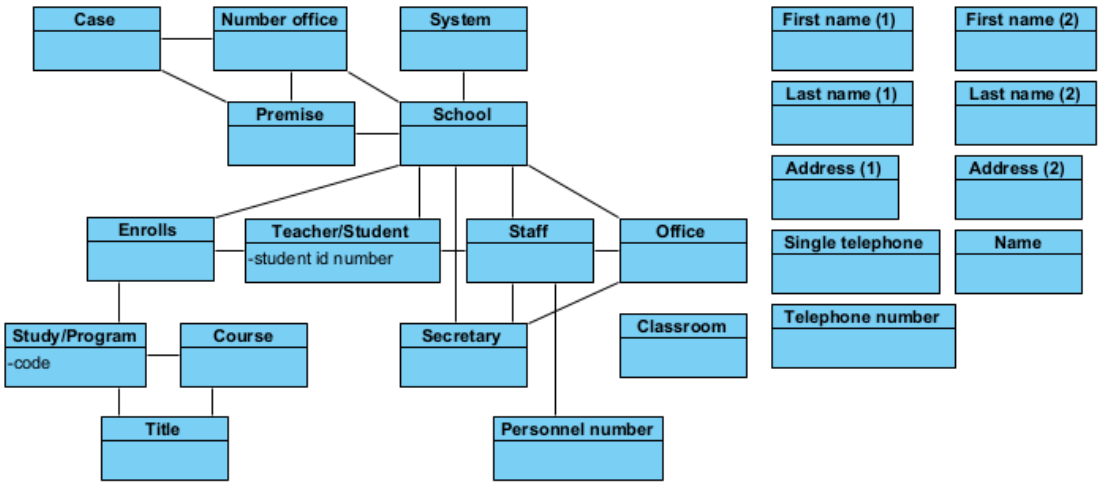
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0. system																												
1. school	2																											
2. staff		2																										
3. teacher			2																									
4. secretary			2																									
5. personnel number			4																									
6. last name			4																									
7. first name			4																									
8. address			4																									
9. premise		2																										
10. number										4																		
11. office				2	2					2																		
12. single telephone												4																
13. classroom										2																		
14. number of places														4														
15. student																												
16. student id number																4												
17. last name																4												
18. first name																4												
19. address																4												
20. telephone number																4												
21. program																2												
22. code																						4						
23. name																						4						
24. course																						2						
25. code																									4			
26. title																									4			
27. number of hours																									4			
Grappes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Figure 38. Diagramme de classes théorique du cahier des charges de test et table de relations entre les grappes de concepts.



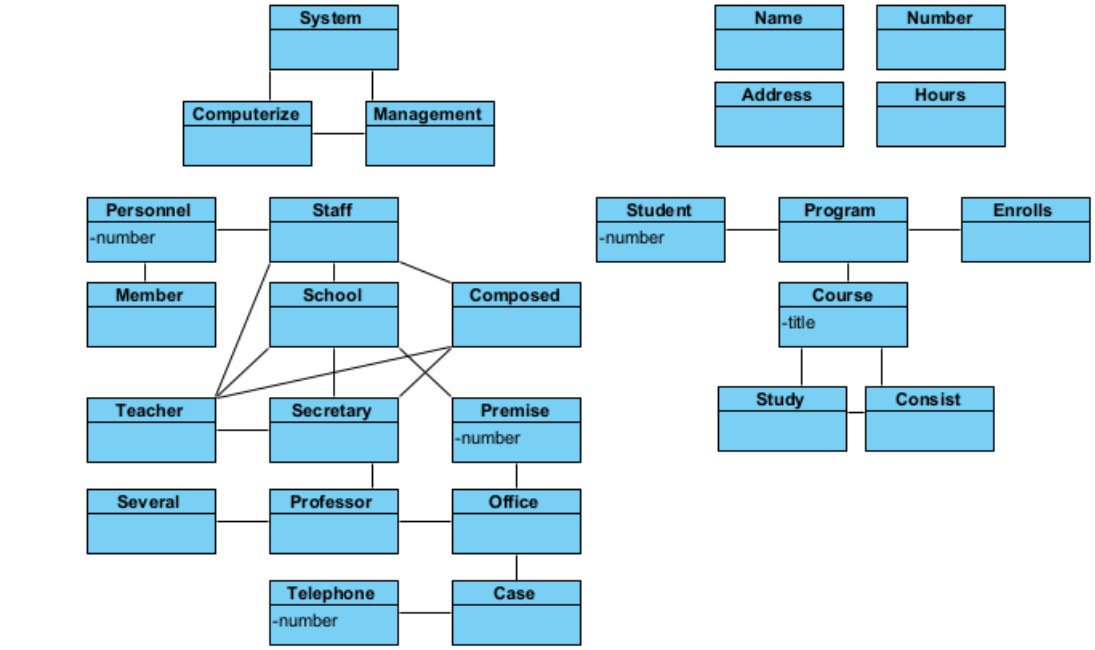
0. system																						
1. school - management school	2																					
2. staff		2																				
3. teacher		2	2																			
4. member of			2																			
5. number - personnel number - student id number		4	4		4																	
6. first name - last name			4		4																	
7. address			4		4																	
8. premise		2				4																
9. office																						
10. single telephone classroom									2													
11. of places																						
12. professor										2												
13. secretary																						
14. student						4	4	4														
15. telephone number														4								
16. enrolls														4								
17. program of study																4						
18. program - study program																						
19. name																		4				
20. course																		2				
21. code																		4		2		
22. title																				4		
23. number of hours																				4		
Grappes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figure 39. Diagramme de classes obtenu avec le modèle BERT sur le cahier des charges de test et table de relations entre les grappes de concepts.



0. system																								
1. school - management school - program	2																							
2. staff		2																						
3. teacher - professor - student		2	2																					
4. secretary																								
5. personnel number - student id number			2	4																				
6. last name																								
7. first name																								
8. address																								
9. premise		2																						
10. number office		2								2														
11. case										2	2													
12. single telephone																								
13. classroom																								
14. office			2																					
15. secretary															2									
16. last name																								
17. first name																								
18. address																								
19. telephone number																								
20. enrolls		2	2																	2				
21. study - program - consists																					4			
22. code																								
23. name																					2			
24. course																					2			2
25. title																								
Grappes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figure 40. Diagramme de classes obtenu avec le modèle RNN sur le cahier des charges de test et table de relations entre les grappes de concepts.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
0. system																																												
1. computerize	2																																											
2. management	2	2																																										
3. staff																																											
4. school				2																																							
5. composed				2																																							
6. teacher				2	2	2																																					
7. secretary					2	2	2																																					
8. member																																											
9. staff																																											
10. personnel									2	2																																	
11. number											4																																	
12. name																																											
13. address																																											
14. school																																											
15. premise															2																													
16. number																4																												
17. office																2																												
18. case																		2																										
19. telephone																			2																									
20. number																																											
21. office																																											
22. professor																						2																						
23. several																							2																				
24. secretary																							2	4																			
25. student																																											
26. student																										2																		
27. number																										4																		
28. name																																											
29. address																																											
30. telephone																																											
31. number																														4													
32. student																																											
33. enrolls																																	4										
34. program																																	4	4									
35. program																																											
36. name																																											
37. study																																											
38. program																																											
39. consist																																											
40. course																																											
41. title																																											
42. number																																											
43. hour																																											
Grappes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43

Figure 41. Diagramme de classes obtenu avec le modèle GPT sur le cahier des charges de test et table de relations entre les grappes de concepts.

Bien que la différence de performance au niveau de la résolution des trois tâches ne soit pas très grande, les conséquences au moment de la production du diagramme des classes apparaissent évidentes.

Le modèle BERT permet de produire un diagramme des classes proches de l'objectif théorique (Figure 39) : les concepts sont bien identifiés et les relations entre ces concepts sont cohérentes. Le modèle BERT montrait quelques signes de faiblesses lors de la résolution de coréférences et celle-ci se traduit dans des regroupements entre mentions moins précis que dans le modèle théorique. La séparation en plusieurs concepts des attributs tels que 'address', 'first name' et 'last name' suivant qu'ils doivent être attachés à différentes classes est en particulier mal gérée par BERT qui a regroupé toutes les mentions dans un seul concept, allant même jusqu'à regrouper les concepts de 'first name' et de 'last name' dans la même grappe. Quelques erreurs ponctuent la résolution des trois tâches telles que le mauvais découpage de la mention 'single telephone classroom' pour la détection de mentions, l'absence de regroupement entre les mentions 'teacher' et 'professor' pour la résolution de coréférences ou l'absence de lien entre 'secretary' et 'staff' ou 'office' et l'erreur commise entre 'code' et 'course' pour la classification de relations, mais d'un point de vue global, le diagramme s'approche du modèle théorique.

Avec le modèle RNN, les confusions deviennent plus grandes (Figure 40). La moins bonne performance au niveau des tâches de détection de mentions et de classification de relations devient gênante pour la lecture du diagramme : des concepts apparaissent sans raison et la détection de la relation classe-attribut est déficiente, ce qui a pour conséquence l'apparition de concepts orphelins sans lien avec d'autres concepts. La bonne tenue du

modèle dans la tâche de résolution de coréférences se matérialise néanmoins au niveau de la bonne gestion des concepts ‘address’, ‘first name’ et ‘last name’ qui sont séparés en plusieurs concepts contrairement à ce qu’avait produit BERT. Malheureusement, l’absence de relations entre ces concepts et les concepts parents ne permet pas de tirer un avantage concret de ce bon résultat au niveau de la production du diagramme de classe.

Finalement, avec le modèle GPT le diagramme de classe est désorganisé (Figure 41). Même si globalement, les concepts d’intérêts sont bien identifiés, il existe beaucoup de bruits composés de concepts sans intérêt et de redondances entre les concepts. Les bons résultats obtenus sur la tâche de résolution de coréférences sont le reflet d’un score de rappel très élevé, mais d’une précision relativement faible en comparaison et ceci a pour conséquence une déficience dans la prise de décision au moment de faire les regroupements entre mentions et de constituer des grappes. Le modèle GPT ne produit que des grappes constituées d’une seule mention, ce qui conduit à la multiplication des éléments constitutifs du diagramme des classes. Certes, on retrouve la distinction entre les attributs homonymes lorsqu’ils sont destinés à être attribués à des classes différentes, mais les regroupements entre classes homonymes sont inexistants. Nous obtenons ainsi deux grappes constituées de la mention ‘program’ consécutives par exemple. Même si les deux autres modèles montrent des résultats plus faibles pour cette tâche, le meilleur équilibre entre précision et rappel obtenu avec ceux-ci implique que des décisions de regroupement (bonnes ou mauvaises) ont été prises et le résultat produit est plus intéressant pour la constitution du diagramme de classes.

4.3.2 Les conclusions

L'utilisation des outils de l'apprentissage profond nous a permis de produire un diagramme de classes sommaire, mais cohérent, en particulier grâce au recours à des structures d'analyse récurrentes efficaces pour le traitement de structures de données séquentielles. Ce résultat est obtenu avec un jeu de données relativement mince de 20 documents ; en ce sens il est particulièrement encourageant quant à la possibilité de développer des modèles plus étoffés avec un jeu de données plus conséquent.

La décision de produire un diagramme simple montrant les relations principales entre concepts d'un diagramme de classes a été prise parce que le jeu de données que nous avons utilisé ne contenait pas assez d'échantillons de relations plus complexes telles que la généralisation, l'agrégation ou la composition. Les documents ont été annotés une première fois en tenant compte de ce type de relations, ce qui signifie que le nombre de classes considérées pour la classification de relations était plus grand initialement, 11 classes au total composées des cinq classes effectivement utilisées et des classes « parent-enfant », « enfant-parent », « agrégat-agrégé », « agrégé-agrégat », « composant-composé », « composé-composant » supplémentaires. Les six dernières classes manquant de représentativité dans les documents utilisés, nous avons décidé d'inclure ces relations dans la relation « classe-classe », ce qui permet à la fois de moins disperser le modèle et d'augmenter la représentativité de cette dernière.

Les résultats obtenus dans chacune des trois tâches prises de manière individuelle sont difficilement comparables à des résultats précédemment obtenus sur les tâches

individuelles ou sur des résolutions conjointes de tâches pour plusieurs raisons. La première raison est que la tâche de détection de mentions telles que nous l'avons abordée est une nouvelle tâche par rapport aux tâches existantes de reconnaissance d'entités nommées ou d'extraction automatique de termes. Nous avons utilisé les techniques de la reconnaissance d'entités nommées de classification des jetons, mais en l'adaptant à notre contexte, c'est-à-dire en l'utilisant sur des noms communs au lieu de noms propres et en modifiant les classes utilisées pour correspondre à notre objectif. Il s'agit en fait d'une nouvelle tâche hybride entre la reconnaissance d'entités nommées dont elle reprend les architectures de modèles et la classification des jetons en catégories et de l'extraction automatique de termes dont elle reprend l'objectif qui est d'extraire des mots ou des expressions d'intérêt. La deuxième raison, qui découle de la première, est que nous avons utilisé un jeu de données et un formalisme nouveaux. Nous n'avons pas pour objectif de produire un nouveau modèle destiné à la résolution des tâches de reconnaissance d'entités nommées, de résolution de coréférences ou de classification de relations, ces trois tâches ne sont envisagées que dans la mesure où les résultats de celles-ci nous permettent de construire un diagramme. En ce sens, nous avons décidé de produire notre propre jeu de données et de l'annoter en conséquence de l'objectif final et non pas des objectifs intermédiaires de résolutions des tâches individuelles, malgré les inconvénients que cela représente en termes de temps passé à produire le jeu de données en lui-même. De ce fait, 1° notre jeu de données n'a pas été utilisé auparavant dans des études antérieures et 2° nos classes ne ressemblent pas à celles utilisées par les jeux de données courants. Nous pourrions toujours observer que la performance obtenue sur la détection de mentions (F_1de 89,3 sur notre meilleur modèle conjoint) n'est que de quelques points inférieurs aux meilleurs résultats

actuels de la tâche de reconnaissance d'entités nommées obtenus sur un jeu de données standard comme OntoNotes 5 (F_1de 92,07 – Li et al. 2020) ou CoNLL-2003 (F_1de 94,6 – Wang et al. 2021), mais les données sont différentes et le nombre de classes que nous avons utilisé est extrêmement réduit par rapport au nombre de classes utilisé avec ces ensembles de données. L'architecture de notre modèle est également plus simple que les architectures utilisées dans ces modèles, mais est destinée à produire des résultats conjoints sur plusieurs tâches. Ces remarques sur la tâche de détection des entités sont les mêmes pour l'autre tâche de classification pure de notre étude, la classification de relations. Cette dernière diffère également dans le fait que les jeux de données standards utilisés sont plus épurés que le jeu de données que nous avons utilisé : le nombre de relations présent dans une phrase et qu'il s'agit de catégoriser est limité et généralement unique dans un jeu de données comme SemEval-2010 Task-8 (Hendrickx et al. 2010). De ce point de vue, ces études se font en milieu « contrôlé », contrairement à cette étude qui est en milieu « naturel » puisque chaque phrase peut comporter plusieurs dizaines de relations à catégoriser. Ceci a pour conséquence que les mêmes structures syntaxiques d'une phrase sont impliquées dans la classification de nombreuses relations dans notre cas et non pas une seule, ce qui complexifie la manière dont l'entrainement influe sur le modèle. Nous pouvons observer que le résultat que nous avons obtenu sur la tâche en elle-même (F_1de 73,6 sur notre meilleur modèle conjoint) est passablement inférieur aux meilleurs résultats obtenus sur un jeu de données standard comme SemEval-2010 Task-8 (F_1de 86,3 – Cai et al. 2016), mais, encore une fois, les études sont très différentes. La tâche de résolution de coréférences est pour sa part plus proche des études courantes publiées sur le sujet en ce sens que le principe de résolution ne diffère pas ou peu (la classification consiste à

retrouver le bon antécédent, il n'y a pas de classes envisagées à priori par les auteurs, seul le jeu de données est différent). Pour cette tâche également nos résultats (F_1de 54,9 sur notre meilleur modèle conjoint) sont très inférieurs aux résultats obtenus sur un jeu de données standard comme OntoNotes 5 (F_1de 79,6 – Joshi et al. 2019) pourtant obtenus avec un modèle de type BERT, SpanBERT, mais pré entrainé de manière différente et qui suggère que des progrès peuvent être réalisés pour cette tâche particulièrement. Malgré cela, les résultats obtenus permettent de construire un diagramme des classes simple, mais cohérent. Le modèle BERT en particulier qui montre les meilleurs résultats pour les deux tâches de classifications pures est particulièrement encourageant. Il suggère également que malgré un résultat relativement faible obtenu sur la tâche de résolution de coréférences, le modèle obtenu n'expose pas de redondances ou d'orphelins en quantité importante. Ceci suggère que la résolution de coréférences est possiblement un peu moins importante pour la cohérence du diagramme de classes final ou qu'elle peut être compensée au moment de la constitution du diagramme. La détection de mentions et la classification de relations sont en effet les deux tâches qui permettent de construire l'architecture du modèle en fournissant les concepts clés d'une part et les liens entre ces concepts. Qu'un concept important soit présent une ou plusieurs fois est relativement moins important quant à la cohérence du modèle et peut toujours être compensé au moment de la constitution de celui-ci.

L'objectif de cette étude est de valider la possibilité de produire un diagramme de classes grâce à un modèle utilisant les techniques de l'apprentissage automatique et, donc, de résoudre toutes les tâches nécessaires à la réalisation de cet objectif. Il s'agit donc de mettre en place un modèle qui tente de résoudre trois tâches conjointement et non pas une

tâche particulière. Même s'il a déjà été montré que le recours à des modèles conjoints pouvait apporter une amélioration de la performance sur certaines tâches (Durett & Klein 2014, Lee et al. 2017), la résolution conjointe de trois tâches reste une contrainte, notamment dans le cas des tâches de résolution de coréférences et de classification de relations qui semblent influencer l'entrainement du modèle dans des directions contraires.

Cette étude propose une démarche d'extraction d'un modèle simple à partir d'un cahier des charges. La simplicité du modèle est en partie liée à la petite quantité de documents utilisée dans le jeu de données, insuffisante pour extraire certaines relations plus complexes comme la généralisation, l'agrégation ou la composition. Néanmoins, maintenant que la démarche expérimentale est validée, d'autres études peuvent compléter l'ensemble de données de manière à étendre le nombre de relations extraites et peaufiner le modèle extrait. Il reste à également à extraire des informations que nous avons choisi de ne pas extraire dans le cadre de cette étude comme les opérations des classes et les multiplicités. Ce choix a été fait pour simplifier la démarche, non pas du fait de la faible quantité de données disponibles, mais parce que ces notions sont plus difficiles à envisager d'un point de vue de leur résolution automatique. Les relations entre classes et entre les classes et leurs attributs peuvent se déduire de la structure syntaxique de la phrase assez aisément d'un point de vue théorique. Dans la mesure où les liens sont décrits assez explicitement, il est possible de comprendre la relation qui unit deux concepts uniquement avec les indices donnés par le document. Les notions de multiplicités et d'opérations sont plus subtiles. Le cas où un cahier des charges décrit explicitement la multiplicité qui doit être associée à une classe est assez rare et souvent les expressions sont même trompeuses.

Dire qu'un « propriétaire possède une maison » peut faussement indiquer qu'il faut associer une multiplicité de un à la classe « maison ». Le facteur humain joue un rôle important pour comprendre qu'un « propriétaire » peut avoir plusieurs « maisons ». Les indices fournis par le document ne sont pas nécessairement suffisant pour évaluer les multiplicités. Il en va de même pour les opérations. Comprendre qu'une classe devra réaliser telle ou telle opération repose bien souvent sur des indices indirects fournis par le document. Or appréhender ce genre d'indices est difficile à concevoir dans le cadre d'un apprentissage automatique basé sur de l'inférence statistique[3]. Les travaux futurs pourront donc se pencher sur la question de l'extraction de ce genre de notions pour l'élaboration d'un modèle complexe.

4.4 SYNTHESE

L'étude que nous avons menée vise à valider la possibilité de produire un diagramme de classes à partir d'un cahier des charges écrit en langage naturel, en utilisant les outils de l'apprentissage profond. Pour cela, nous avons créé un jeu de données en annotant à la main un ensemble de cahiers de charges de manière à pouvoir entrainer un modèle à résoudre trois tâches intermédiaires, la détection de mentions, la résolution de coréférences et la classification de relations, dont les résultats servent à élaborer le diagramme. Nous avons comparé les performances obtenues par trois grands modèles, chacun subdivisé en quatre sous-modèles, sur les différentes tâches avant de produire le diagramme de classes correspondant à ces résultats sur un cahier des charges de test. Notre meilleur modèle, un modèle de type BERT avec un encodage des mentions sous forme de moyenne pondérée, a

[3] Au niveau de l'analyse d'un cahier des charges, le plus important est de découvrir les fonctionnalités du système. L'attribution des responsabilités aux classes peut se faire en fin d'analyse ou début de conception.

permis de produire un diagramme, certes simple, mais cohérent, très encourageant quant à la possibilité de produire de manière automatique un diagramme de classe à partir d'un cahier des charges.

CONCLUSION GÉNÉRALE

La réalisation d'une application correspond à la volonté d'élaborer une solution informatique à un problème. Concrètement, au niveau informatique, il s'agit de transformer le problème traduit en exigences dans un cahier des charges écrit en langage naturel en code informatique écrit en langage formel (ou semi-formel). L'ingénierie logicielle a développé un ensemble de processus pour effectuer efficacement cette transformation impliquant l'élaboration de modèles intermédiaires avant la production du code final. L'étape cruciale de ces processus est le moment inévitable où le langage naturel doit être transposé en langage formel (ou semi-formel) puisqu'une fois cette étape franchie, le reste du processus est automatisable. Dans une démarche comme l'architecture dirigée par les modèles (MDA), cette conversion est la première étape du processus qui consiste à traduire les exigences en un premier modèle comme le modèle indépendant de la plateforme (PIM). Par la suite ce PIM, peut être transformé en un modèle dépendant de la plateforme (PSM) puis en code par des transformations automatiques. Le processus est donc automatisable à partir du moment où un modèle tel que le PIM est produit, ce qui fait que l'étape de traduction des exigences en PIM est la seule étape qui empêche l'automatisation complète du processus.

Il n'est toutefois pas anodin que cette étape n'ait pas encore été automatisée : traduire automatiquement un texte écrit en langage naturel en un modèle en langage formel ou semi-

formel est très complexe de par la nature du langage naturel. Exprimer une idée peut se faire de tellement de manières différentes qu'il est difficile d'envisager de les englober toutes dans un ensemble de règles conçues à priori pour faire la traduction. Le travail a déjà été tenté dans plusieurs études. Le succès est mitigé, mais surtout la démarche impose un travail très important en amont pour comprendre les structures du langage naturel et composer un ensemble de règles suffisamment intéressant et complet pour appréhender assez de schémas syntaxiques et sémantiques pour permettre l'élaboration d'un modèle relativement cohérent. Sans compter que le travail est à répéter pour toutes les langues parce qu'évidemment les structures changent d'une langue à l'autre. Au regard de ces difficultés, il apparait que la solution devrait s'adapter à la langue et aux structures intéressantes sans que l'humain n'ait à les découvrir en amont. Il faudrait donc une solution qui « apprenne » à traduire des structures syntaxiques et sémantiques en concepts et en relations.

L'intelligence artificielle, et plus spécifiquement l'apprentissage profond représente un ensemble de techniques et de modèles capables d'apprendre à reconnaitre des structures de manière à produire un résultat. De manière générale, un modèle est confronté à un jeu de données annoté avec les résultats attendus duquel il apprend à produire une réponse attendue. Une fois que le modèle est entrainé, il est prêt à être utilisé sur des cas réels non annotés et à produire une solution ou un diagnostic. Il reste donc à définir comment utiliser l'apprentissage profond pour produire un PIM à partir d'un cahier des charges.

La solution que nous proposons dans cette étude est de décomposer la tâche générale en trois tâches unitaires résolues de manière conjointe. Ces tâches reprennent des tâches

déjà connues de l'apprentissage profond : la détection de mentions, la résolution de coréférences et la classification de relations. La tâche de détection de mentions a le mandat d'identifier les mentions d'intérêt du document, c'est-à-dire les mentions qui désignent les futurs concepts du diagramme de classes. La tâche de résolution de coréférences doit ensuite faire le lien entre les différentes mentions qui désignent un même concept afin d'éliminer la redondance et de mettre en commun l'ensemble des informations relatives à un concept. Enfin, la classification de relations identifie les relations qui existent entre les mentions et donc entre les concepts. Lorsque ces trois tâches sont résolues, nous disposons d'un ensemble de grappes qui désignent les concepts du diagramme de classes que nous voulons produire, chacune disposant d'un ensemble de relations avec d'autres grappes qui permettent de structurer les concepts dans le diagramme. Comme pour tout travail relatif à l'apprentissage automatique supervisé, nous devons avoir un jeu de données annoté pour entrainer notre modèle, ce qui n'existe pas dans le domaine public. Nous en proposons un que nous avons constitué à partir de cahiers des charges simples rédigés en anglais et que nous avons annotés suivant un formalisme que nous avons élaborés pour les besoins de l'étude, mais qui est disponible pour d'autres études ultérieures. Nous proposons également un ensemble de modèles pour la réalisation de l'objectif. Tous ces modèles sont constitués suivant une architecture générale similaire composée d'une structure d'encodage adaptée au traitement de données séquentielles et commune à la résolution des trois tâches et de trois structures de décodage adaptées à la résolution de chacune des trois tâches décrites précédemment. Le premier ensemble de modèles utilise un RNN bidirectionnel composé de cellules LSTM (RNN Bi-LSTM) comme structure d'encodage, le deuxième ensemble un réseau de type BERT et le troisième ensemble un réseau de type GPT-2. Les résultats

obtenus avec l'ensemble de modèles BERT et dans une moindre mesure avec l'ensemble de modèles RNN Bi-LSTM sont suffisamment intéressants pour permettre la constitution d'un diagramme de classes simple et cohérent.

Le jeu de données que nous rendons disponible avec cette étude est d'une taille encore modeste (20 documents comprenant un peu plus de 10 000 jetons). Malgré cela, les résultats obtenus sont encourageants, ce qui permet d'envisager qu'en augmentant la taille de l'ensemble de données, les résultats deviennent encore plus précis.

La solution envisagée qui consiste à décomposer le travail en trois tâches unitaires est la façon la plus simple de résoudre le problème posé et certainement la plus adaptée avec un jeu de données qui reste présentement assez succinct. L'autre façon d'aborder le problème est d'envisager une solution plus intégrée qui produirait le PIM de manière directe à partir du cahier des charges. Le même type de travail se fait déjà dans des tâches complexes telles que la traduction automatique (par exemple Vaswani et al. 2017) ou l'annotation d'image (par exemple Cornia et al. 2020) dans lesquelles une représentation initiale est traduite directement en une représentation finale. Nous pouvons envisager que le cahier des charges soit traduit directement en langage XML représentant la structure du PIM. Après tout, il n'y a pas tant de différences entre passer du français à l'anglais et passer du français au XML. Le problème qui se poserait cependant serait la cohérence d'ensemble du PIM. La tâche de traduction permettrait d'exprimer les idées exprimées dans une phrase en un fragment de diagramme, mais il faudrait encore trouver un moyen de recoller toutes les idées concernant un même concept à travers un document complet, tâche qui est du ressort de la résolution de coréférences dans notre étude. Si cela peut être fait,

l’amélioration récente des outils de traduction suggère un potentiel intéressant de l’approche.

RÉFÉRENCES BIBLIOGRAPHIQUES

Amjadian, E. (2019). Representation Learning for Information Extraction [Thèse de doctorat]. Carleton University.

Arora, C., Sabetzadeh, M., Briand, L. et Zimmer, F. (2016). Extracting domain models from natural-language requirements: approach and industrial evaluation. *MODELS'16 Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 250-260.

Bagga, A. et Baldwin, B. (1998). Algorithms for Scoring Coreference Chains. *The first international conference on language resources and evaluation workshop on linguistics coreference, 1*, 563-566.

Bahdanau, D., Cho, K. et Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473v7*.

Bhagat, S., Kapadni, P., Kapadnis, N., Patil, D. et Mamta, B. (2012). Class Diagram Extraction Using NLP. *1st International Conference on Recent Trends in Engineering & Technology, Mar-2012 Special Issue of International Journal of electronics, Communication & Soft Computing Science & Engineering*, 125-128.

Bjorkelund, A. et Kuhn, J. (2014). Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 47-57.

Brown, T. D., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. et Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv:2005.14165v4*.

Bunescu, R. C. et Mooney, R. J. (2005). Subsequence Kernels for Relation Extraction. *Proceedings of the 18th International Conference on Neural Information Processing Systems*, 171-178.

Cai, R., Zhang, X. et Wang, H. (2016). Bidirectional Recurrent Convolutional Neural Network for Relation Classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 756-765.

Chénard, G. (2013). Vers une nouvelle approche de la modernisation des systèmes légataires à travers la migration vers un environnement dirigé par les modèles [Thèse de doctorat]. Université du Québec à Montréal.

Chinchor, N. et Marsh, E. (1998). Appendix D: MUC-7 Information Extraction Task Definition (version 5.1). *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, https://aclanthology.org/M98-1027.

Chiu, J. P. C. et Nichols, E. (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics, 4*, 357-370.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. et Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1724-1734.

Clark, K. et Manning, C. D. (2016a). Improving Coreference Resolution by Learning Entity-Level Distributed Representations. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 1*, 643-653.

Clark, K. et Manning, C. D. (2016b). Deep Reinforcement Learning for Mention-Ranking Coreference Models. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2256-2262.

Collins, M. et Singer, Y. (1999). Unsupervised Models for Named Entity Classification. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 100-110.

Collins, M. et Duffy, N. (2001). Convolution Kernels for Natural Language. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 625-632.

Conrado, M., Salguiro Pardo, T. A. et Oliveira Rezende, S. (2013). A Machine Learning Approach to Automatic Term Extraction using a Rich Feature Set. *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2013 Student Research Workshop*, 16-23.

Cornia, M., Stefanini, M., Baraldi, L., Cucchiara, R. (2020). Meshed-Memory Transformer for Image Captioning. *arXiv:1912.08226v2*.

Cortes, C. et Vapnik, V. (1995). Support-vector Networks. *Machine Learning, 20*, 273-297.

Cram, D. et Daille, B. (2016). TermSuite: Terminology Extraction with Term Variant Detection. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – System Demonstrations*, 13-18.

Culotta, A. et Sorensen, J. (2004). Dependency Tree Kernels for Relation Extraction. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 423-429.

Devlin. J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1*, 4171–4186.

Dinarelli, M. et Grobol, L. (2018). Modélisation d'un contexte global d'étiquettes pour l'étiquetage de séquences dans les réseaux neuronaux récurrents. *Journée commune AFIA-ATALA sur le Traitement Automatique des Langues et l'Intelligence Artificielle pendant la onzième édition de la plate-forme Intelligence Artificielle (PFIA 2018), Jul 2018, Nancy, France. hal-02002111*.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S. et Weischedel, R. (2004). The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, 837-840.

Dos Santos, C., Xiang, B. et Zhu, B. (2015). Classifying Relations by Ranking with Convolutional Neural Networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 1*, 626-634.

Durett, G. et Klein, D. (2014). A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics, 2*, 477-490.

Evans, E. (2003). Domain-Driven Design : Tackling Complexity in the Heart of Software. Addison-Wesley Longman Publishing Co., Inc.

Francoeur, D. (2010). Machines à vecteurs de support – Une introduction. *Cahier Mathématiques de l'Université de Sherbrooke, 1*, 7-25.

Fukumoto, J., Masui, F., Shimohata, M. et Sasaki, M. (1998). Oki Electric Industry : Description of the Oki System as Used for MUC-7. *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, https://aclanthology.org/M98-1004.

Garigliano, R., Urbanowicz, A. et Nettleton, D. J. (1998). University of Durham : Description of the LOLITA System as Used in MUC-7. *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, https://aclanthology.org/M98-1005.

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

Gormley, M. R., Yu, M. et Dredze, M. (2015). Improved Relation Extraction with Feature-Rich Compositional Embedding Models. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1774-1784.

Grishman, R. et Sundheim, B. (1995). Design of the MUC-6 Evaluation. *MUC6 '95: Proceedings of the 6th conference on Message understanding*, 1-11.

Grishman, R. et Sundheim, B. (1996). Message Understanding Conference- 6: A Brief History. *COLING-96, Proceedings of the International Conference on Computational Linguistics, 1*, 466-471.

Harmain, H. M. et Gaizauskas, R. (2000). CM-Builder: an automated NL-based CASE tool. *Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering*, 45-53.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., O Seaghdha, D., Pado, S., Pennacchiotti, M., Romano, L. et Szpakowicz, S. (2010). SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. *Proceedings of the 5th International Workshop on Semantic Evaluation*, 33-38.

Hobbs, J. R. (1978). Resolving pronoun references. *Lingua, 44,* 311-338.

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

Howard, J. et Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 1*, 328-339.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R. (2006). OntoNotes: The 90% Solution. *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 57-60.

Huang, Z., Xu, W. et Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv:1508.01991v1*.

Huang, W., Cheng, X., Wang, T. et Chu, W. (2019). BERT-Based Multi-Head Selection for Joint Entity-Relation Extraction. *Natural Language Processing and Chinese Computing*, 713-723.

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H. et Wilks, Y. (1998). University of Sheffield: Description of the LaSIE-II System as Used for MUC-7. *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, https://aclanthology.org/M98-1007.

Ilieva, M. G. et Ormandjeva, O. (2006). Models Derived from Automatically Analyzed Textual User Requirements. *Fourth International Conference on Software Engineering Research, Management and Applications*, 13-21.

Joshi, S. D. et Deshpande, D. (2012). Textual Requirement Analysis for UML Diagram Extraction by using NLP. *International Journal of Computer Applications, 50*, 42-46.

Joshi, M., Levy, O., Weld, D. S. et Zettlemoyer, L. (2019). BERT for Coreference Resolution: Baselines and Analysis. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 5803-5808.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L. et Levy, O. (2020). SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics, 8*, 64-77.

Kageura, K. et Umino, B. (1996). Methods of automatic term recognition – A review. *Terminology*, *3*, 259-289.

Kambhatla, N. (2004). Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, 22-25.

Katiyar, A. et Cardie, C. (2018). Nested Named Entity Recognition Revisited. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1*, 861-871.

Kim, J.-H. et Woodland, P. C. (2000). A rule-based named entity recognition system for speech input. *Proceedings of the International Conference on Spoken Language Processing, 1*, 528-531.

Klepp, A. G., Warner, J. et Bast, W. (2003). MDA Explained : The Model Driven Architecture : Practice and Promis. Addison-Wesley Longman Publishing Co., Inc.

Konstantinova, N. (2014). Review of Relation Extraction Methods: What Is New Out There?. *Communications in Computer and Information Science, 436*, 15-28.

Lafferty, J., McCallum, A. et Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the 18th International Conference on Machine Learning*, 282-289.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C. (2016). Neural Architectures for Named Entity Recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 260-270.

Lappin, S. et Leass, H. J. (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics, 20*, 535-561.

Lee, K., He, L., Lewis, M. et Zettlemoyer, L. (2017). End-to-end Neural Coreference Resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 188-197.

Lee, K., He, L. et Zettlemoyer, L. (2018). Higher-order Coreference Resolution with Coarse-to-fine Inference. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2*, 687-692.

Lee, J., Seo, S. et Choi, Y. S. (2019). Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing. *Symmetry*, *11*, 785-796.

Li, F., Zhang, M., Fu, G. et Ji, D. (2017). A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinformatics, 18*, 198, https://doi.org/10.1186/s12859-017-1609-9.

Li, J., Sun, A., Han, J. et Li, C. (2020). A Survey on Deep Learning for Named Entity Recognition. *arXiv:1812.09449v3*.

Liddy, E. D. (2001). Natural Language Processing. Dans M. A. Drake (ed.) *Encyclopedia of Library and Information Science, 2e edition*. Marcel Decker, Inc.

Liu, D. (2003). Automating Transition from Use Cases to Class Model [Mémoire de maitrise]. University of Calgary.

Luo, X. (2005). On Coreference Resolution Performance Metrics. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 25-32.

Luo, Y., Xiao, F. et Zhao, H. (2019). Hierarchical Contextualized Representation for Named Entity Recognition. *arXiv:1911.02257v2*.

Luong, M.-T., Pham, H et Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412-1421.

McDonald, R. et Pereira, F. (2005). Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, *6*, S6.

Mich, L. et Garigliano, R. (2002). NL-OOPS: A requirements analysis tool based on natural language processing. Dans A. Zanasi, C. A. Brebbia, N. F. F. Ebecken et P. Melli (eds.) *Data Mining III* (p. 321-330). WIT Press.

Mikolov, T., Chen, K., Corrado, G. et Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781v3*.

Miwa, M. et Bansal, M. (2016). End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 1*, 1105-1116.

Mohd, I. et Rodina, A. (2010). Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Technique. *2010 Second International Conference on Computer Research and Development*, 200-204.

Morgan, R., Garigliano, R., Callaghan, P., Poria, S., Smith, M. et Cooper, C. (1995). University of Durham: description of the LOLITA System as Used in MUC-6. *MUC6 '95: Proceedings of the 6th conference on Message understanding*, 71-85.

Nadeau, D., Turney, P. D. et Matwin, S. (2006). Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. *Proceedings of the 19th international conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*, 266-277.

Ng, V. et Cardie, C. (2002). Improving Machine Learning Approaches to Coreference Resolution. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 104-111.

Ng, V. (2017). Machine Learning for Entity Coreference Resolution: A Retrospective Look at Two Decades of Research. *Proceedings of the AAAI Conference on Artificial Intelligence, 31*, 4877-4884.

Nguyen, T. H. et Grishman, R. (2015). Relation Extraction: Perspective from Convolutional Neural Networks. *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 39-48.

Ohta, T., Tadeisi, Y. et Kim, J.-D. (2002). The GENIA corpus: an annotated research abstract corpus in molecular biology domain. *Proceedings of the second international conference on Human Language Technology Research*, 82-86.

Pazienza, M. T., Pennacchiotti, M. et Zanzotto, F. M. (2005). Terminology Extraction: An Analysis of Linguistic and Statistical Approaches. Dans S. Sirmakessis (ed.) *Knowledge Mining. Studies in Fuzziness and Soft Computing, 185* (p. 255-279). Springer.

Pennington, J., Socher, R. et Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532-1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. et Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, *1*, 2227-2237.

Pomares-Quimbaya, A., Sierra Munera, A., Gonzalez Rivera, R. A., Daza Rodriguez, J. C., Munoz Velandia, O. M., Garcia Pena, A. A. et Labbe, C. (2016). Named Entity Recognition Over Electronic Health Records Through a Combined Dictionary-based Approach. *Procedia Computer Science*, *100*, 55-61.

Popescu, D., Rugaber, S., Medvidovic, N. et Berry, D. M. (2008). Reducing Ambiguities in Requirements Specifications via Automatically Created Object-Oriented Models. Dans B. Paech et C. Martell (eds.) *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*. *Monterey Workshop 2007. Lecture Notes in Computer Science, 5320* (p. 103–124). Springer.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O. et Zhang, Y. (2012). CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, 1-40.

Radford, A., Narasimhan, K., Salimans, T. et Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D. et Manning, C. (2010). A Multi-Pass Sieve for Coreference Resolution. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 492-501.

Rahman, A. et Ng, V. (2009). Supervised Models for Coreference Resolution. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 968-977.

Ratinov, L. et Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, 147-155.

Ratinov, L. et Roth, D. (2012). Learning-based Multi-Sieve Co-reference Resolution with Knowledge. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1234-1244.

Sekine, S. et Nobata, C. (2004). Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, 1977-1980.

Shao, Y., Hardmeier, C. et Nivre, J. (2016). Multilingual Named Entity Recognition using Hybrid Neural Networks. *The Sixth Swedish Language Technology Conference*.

Simon, N.I. (2018). Automatic Term Extraction in Technical Domain Using Part-of-Speech and Common-Word Features [Mémoire de maitrise]. Dalhousie University.

Soon, W. M., Nh, H. T. et Chung Yong Lim, D. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics, 27*, 521-544.

Stoyanov, V. et Eisner, J. (2012). Easy-first Coreference Resolution. *24th International Conference on Computational Linguistics - Proceedings of Computational Linguistics 2012*, 2519-2534.

Strakova, J., Straka, M. et Hajic, J. (2019). Neural Architectures for Nested NER through Linearization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5326-5331.

Stylianou, N. et Vlahavas, I. (2019). A Neural Entity Coreference Resolution Review. *arXiv:1910.09329v2*.

Sukthander, R., Poria, S., Cambria, E. et Thirunavukarasu, R. (2018). Anaphora and Coreference Resolution: A Review. *arXiv:1805.11824v1*.

Sun, A., Grishman, R. et Sekine, S. (2011). Semi-supervised Relation Extraction with Large-scale Word Clustering. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 521-529.

Sutton, C. et McCallum, A. (2007). An Introduction to Conditional Random Fields in Relational Learning. Dans L. Getoor et B. Taskar (eds.) *Introduction to Statistical Relational Learning* (p 93-128). The MIT Press.

Thakur, J. S. et Gupta, A. (2016). AnModeler: A tool for generating domain models from textual specifications. *2016 31st IEEE/ACM International Conference on Automated Software Engineering*, 828-833.

Tjong Kim Sang, E. F. et De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*, 142–147.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreil, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762v5*.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D. et Hirschman, L. (1995). A model-theoretic coreference scoring scheme. *MUC6 '95: Proceedings of the 6th conference on Message understanding*, 45-52.

Vu, T., Aw, A. T. et Zhang, M. (2008). Term Extraction Through Unithood And Termhood Unification. *Proceedings of International Joint Conference on Natural Language Processing*, 631-636.

Vu, N. T., Adel, H., Gupta, P. et Schutze, H. (2016). Combining Recurrent and Convolutional Neural Networks for Relation Classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 534-539.

Wang, R., Liu, W. et McDonald, C. (2016a). Featureless Domain-Specific Term Extraction with Minimal Labelled Data. *Proceedings of Australasian Language Technology Association Workshop 2016*, 103-112.

Wang, J., Cao, Z., de Melo, G. et Liu, Z. (2016b). Relation Classification via Multi-Level Attention CNNs. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 1*, 1298-1307.

Wang, J. et Lu, W. (2020). Two are Better than One: Joint Entity and Relation Extraction with Table-Sequence Encoders. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 1706-1721.

Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K. (2021). Automated Concatenation of Embeddings for Structured Prediction. *arXiv:2010.05006v4*.

Wiseman, S., Rush, A. M., Shieber, S. M. et Weston, J. (2015). Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 1*, 1416-1426.

Wiseman, S., Rush, A. M. et Shieber, S. M. (2016). Learning Global Features for Coreference Resolution. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 994-1004.

Wu, J.-L. et Ma, W.-Y. (2017). A Deep Learning Framework for Coreference Resolution Based on Convolutional Neural Network. *2017 IEEE 11th International Conference on Semantic Computing*, 61-64.

Wu, S. et He, Y. (2019). Enriching Pre-trained Language Model with Entity Information for Relation Classification. *arXiv:1905.08284v1*.

Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H. et Jin, Z. (2015). Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1785-1794.

Yang, X., Su, J., Zhou, G. et Tan, C. L. (2004). An np-cluster based approach to coreference resolution Paths. *Proceedings of the 20th international conference on Computational Linguistics*, 226-232.

Yang, Z., Salakhutdinov, R. et Cohen, W. (2016). Multi-Task Cross-Lingual Sequence Tagging from Scratch. *arXiv:1603.06270v2*.

Yu, J., Bohnet, B. et Poesio, M. (2020). Named Entity Recognition as Dependency Parsing. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6470-6476.

Yuan, Y., Gao, J. et Zhang, Y. (2017). Supervised learning for robust term extraction. *2017 International Conference on Asian Language Processing*, 302-305.

Yue, T., Briand, L. C. et Labiche, Y. (2011). A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering, 16*, 75-99.

Yue, T., Briand, L. C. et Labiche, Y. (2013). Automatically Deriving a UML Analysis Model from a Use Case Model. Rapport technique 2010-15, Simula Reseach Laboratory.

Zadeh, B. Q. et Handschuh, S. (2014). The ACL RD-TEC: A Dataset for Benchmarking Terminology Extraction and Classification in Computational Linguistics. *Proceedings of the 4th International Workshop on Computational Terminology*, 52-63.

Zelenko, D., Aone, C. et Richardella, A. (2003). Kernel Methods for Relation Extraction. *Journal of Machine Learning Research, 3*, 1083-1106.

Zeng, D., Liu, K., Lai, S., Zhou, G. et Zhao, J. (2014). Relation Classification via Convolutional Deep Neural Network. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2335-2344.

Zhang, D. et Wang, D. (2015). Relation Classification via Recurrent Neural Network. *arXiv:1508.01006v2*.

Zhang, S., Zheng, D., Hu, X. et Yang, M. (2015). Bidirectional Long Short-Term Memory Networks for Relation Classification. *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, 73-78.

Zhou, G., Su, J., Zhang, J. et Zhang, M. (2005). Exploring Various Knowledge in Relation Extraction. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 427-434.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B, Hao, H. et Xu, B. (2016). Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2*, 207-212.

ANNEXE I CODE DU PROGRAMME

```
import datetime
import os
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import bcubed
from transformers import BertTokenizer, BertModel, GPT2Tokenizer, GPT2Model
from math import sqrt
# from sklearn_crfsuite.metrics import flat_classification_report
from sklearn.metrics import classification_report


def cudaize(x):
    if torch.cuda.is_available():
        x = x.cuda()
    return x


def prepare_index():
    mentions_index = {"O": 0, "H": 1, "R": 2}
    embeddings_index = {}

    fichier = open("data/glove.6B/glove.6B.50d.txt", 'r', encoding="UTF-8")
    for ligne in fichier.readlines():
        rangee = ligne.strip().split(' ')
        embeddings_index[rangee[0]] = [float(i) for i in rangee[1:]]
    fichier.close()

    return embeddings_index, mentions_index


def prepare_document(entry, embeddings_index, mentions_index):
    document = Document()
    sentence = Sentence()
    iterator = 0
```

```
    file = open(entry, 'r', encoding="UTF-8")
    for line in file.readlines():
        row = line.strip().split(' ')
        if len(row) < 2:
            document.sentences.append(sentence)
            sentence = Sentence()
        else:
            wording = row[0].lower()
            position = iterator
            embedding = torch.cuda.FloatTensor([embeddings_index.get(row[0].lower(), embeddings_inde
x.get("unk"))])
            sentence.words.append(Word(wording, embedding, position))
            sentence.mention_goldens = torch.cat((sentence.mention_goldens, torch.cuda.LongTensor([m
entions_index.get(row[1])])), dim=0)
            if len(row) > 2:
                sentence.coreference_goldens = torch.cat((sentence.coreference_goldens, torch.cuda.L
ongTensor([[int(row[2])]])), dim=0)
            if len(row) > 3:
                row_relations = row[3].split('-')
                for row_relation in row_relations:
                    sentence.relation_goldens = torch.cat((sentence.relation_goldens, torch.cuda.Lon
gTensor([int(row_relation)])), dim=0)
            iterator += 1

    document.sentences.append(sentence)

    return document


def prepare_corpus(prefixe, entries, embeddings_index, mentions_index):
    corpus = Corpus()

    for entry in entries:
        document = prepare_document(prefixe + entry, embeddings_index, mentions_index)
        corpus.documents.append(document)

    return corpus


def prepare_corpora(embeddings_index, mentions_index):
    train_entries = os.listdir('data/train')
    val_entries = os.listdir('data/val')
    test_entries = os.listdir('data/test')
```

```
    train_corpus = prepare_corpus('data/train/', train_entries, embeddings_index, mentions_index)
    val_corpus = prepare_corpus('data/val/', val_entries, embeddings_index, mentions_index)
    test_corpus = prepare_corpus('data/test/', test_entries, embeddings_index, mentions_index)

    return train_corpus, val_corpus, test_corpus


class Corpus:
    def __init__(self):
        self.documents = []


class Document:
    def __init__(self):
        self.sentences = []
        self.clusters = []


class Sentence:
    def __init__(self):
        self.words = []
        self.mentions = []
        self.mention_goldens = torch.cuda.LongTensor()
        self.coreference_goldens = torch.cuda.LongTensor()
        self.relation_goldens = torch.cuda.LongTensor()


class Word:
    def __init__(self, wording, embedding, position):
        self.wording = wording
        self.embedding = embedding
        self.position = position
        self.encoding = torch.cuda.FloatTensor()


class Mention:
    def __init__(self, word, head=-1):
        self.words = [word]
        self.cluster = 0
        self.head = head
        self.attentionLayer = torch.cuda.FloatTensor()


class Cluster:
    def __init__(self, mention):
```

```
        self.mentions = [mention]
        self.relations = {2: [], 3: [], 4: []} # relations par type avec les autres clusters {2: [],
 3: [], 4: [], ...}


class DistanceLayer(nn.Module):
    # Distance encoding
    def __init__(self, distance_dim, dropout):
        super().__init__()
        self.refs = [1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
        self.distance_dim = distance_dim
        self.layer = nn.Sequential(
            nn.Embedding(len(self.refs)+1, self.distance_dim),
            nn.Dropout(p=dropout)
        )

    def forward(self, distance):
        return self.layer(self._lookup(distance))

    def _lookup(self, distance):
        for i in range(len(self.refs)-1, -1, -1):
            if distance >= self.refs[i]:
                return torch.cuda.LongTensor([i])


class EncodingRNNLayer(nn.Module):
    # BiLSTM
    def __init__(self, embedding_dim, hidden_dim, dropout):
        super().__init__()
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, bidirectional=True)
        self.dropout = nn.Dropout(dropout)

    def forward(self, tensor):
        output, _ = self.lstm(tensor.view(len(tensor), 1, -1))
        return self.dropout(output).view(output.size(0), output.size(2))


class EncodingBERTLayer(nn.Module):
    # BERT
    def __init__(self, dropout):
        super().__init__()
        self.bert_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
        self.bert = cudaize(BertModel.from_pretrained('bert-base-uncased'))
        self.dropout = nn.Dropout(dropout)
```

```
    def forward(self, sentence):
        word_sentence = [word.wording for word in sentence]
        inputs = self.bert_tokenizer.encode(word_sentence)
        outputs = self.bert(input_ids=cudaize(torch.tensor(inputs).view(-
1, len(inputs))), attention_mask=cudaize(torch.ones(1, len(inputs), dtype=torch.long)), token_type_i
ds=cudaize(torch.zeros(1, len(inputs), dtype=torch.long)))
        return self.dropout(outputs.last_hidden_state[0][1:-1])


class EncodingGPTLayer(nn.Module):
    # GPT
    def __init__(self, dropout):
        super().__init__()
        self.gpt_tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
        self.gpt = cudaize(GPT2Model.from_pretrained('gpt2'))
        self.dropout = nn.Dropout(dropout)

    def forward(self, sentence):
        word_sentence = [word.wording for word in sentence]
        inputs = self.gpt_tokenizer.encode(word_sentence)
        outputs = self.gpt(input_ids=cudaize(torch.tensor(inputs).view(-
1, len(inputs))), attention_mask=cudaize(torch.ones(1, len(inputs), dtype=torch.long)), token_type_i
ds=cudaize(torch.zeros(1, len(inputs), dtype=torch.long)))
        return self.dropout(outputs.last_hidden_state[0])


class ScoreLayer(nn.Module):
    # Classification
    def __init__(self, input_dim, hidden_dim, target_dim, dropout):
        super().__init__()
        self.layer = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, target_dim)
        )

    def forward(self, tensor):
        return self.layer(tensor)


class ScoreTLayer(nn.Module):
```

```
    # Classification coreference
    def __init__(self, input_dim, hidden_dim, target_dim, dropout):
        super().__init__()
        self.layer = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, target_dim)
        )

    def forward(self, tensor):
        output = self.layer(tensor)
        return torch.transpose(output, 0, 1)


class Model(nn.Module):
    def __init__(self, method, technique, embedding_dim, encoding_hidden_dim, decoding_hidden_dim, m
ention_dim, relation_dim, distance_dim, dropout):
        super().__init__()
        self.method = method
        self.technique = technique
        self.embedding_dim = embedding_dim
        self.encoding_hidden_dim = encoding_hidden_dim
        self.decoding_hidden_dim = decoding_hidden_dim
        self.mention_dim = mention_dim
        self.relation_dim = relation_dim
        self.distance_dim = distance_dim
        self.padding_dim_coef = 1
        if self.technique == 'padding':
            self.padding_dim_coef = 7   # Max size padding

        if self.method == 'RNN':
            self.encodingLayer = EncodingRNNLayer(self.embedding_dim, self.encoding_hidden_dim, drop
out)
            self.encoding_hidden_dim = 2*encoding_hidden_dim    # Bilateral
        elif self.method == 'BERT':
            self.encodingLayer = EncodingBERTLayer(dropout)
        else:
            self.encodingLayer = EncodingGPTLayer(dropout)
        self.mentionScoreLayer = ScoreLayer(self.encoding_hidden_dim, self.decoding_hidden_dim, self
.mention_dim, dropout)
```

```
        self.attentionLayer = ScoreLayer(self.encoding_hidden_dim, self.decoding_hidden_dim, 1, drop
out)
        self.coreferenceDistanceLayer = DistanceLayer(self.distance_dim, dropout)
        self.coreferenceScoreTLayer = ScoreTLayer(self.encoding_hidden_dim*3*self.padding_dim_coef +
 self.distance_dim, self.decoding_hidden_dim, 1, dropout)
        self.relationDistanceLayer = DistanceLayer(self.distance_dim, dropout)
        self.relationScoreLayer = ScoreLayer(self.encoding_hidden_dim*3*self.padding_dim_coef + self
.distance_dim, self.decoding_hidden_dim, self.relation_dim, dropout)

    def forward(self, sentence, clusters, mention_loss_function, coreference_loss_function, relation
_loss_function):
        inputs = sentence.words
        if self.method == 'RNN':
            sentence_embeddings = torch.cuda.FloatTensor()
            for word in sentence.words:
                sentence_embeddings = torch.cat((sentence_embeddings, word.embedding), dim=0)

            inputs = sentence_embeddings

        encodings = self.encodingLayer(inputs)
        for i in range(encodings.size(0)):
            sentence.words[i].encoding = encodings[i].view(1, encodings.size(1))

        # Mentions - Minibatch = sentence
        mention_scores = self.mentionScoreLayer(encodings.view(len(sentence.words), -1))
        mention_scores_max_index = torch.max(mention_scores, dim=1).indices
        mention_loss = mention_loss_function(mention_scores, sentence.mention_goldens)
        if self.training:
            mention_loss.backward(retain_graph=True)
        self._extract_mentions(sentence)

        # Coreferences - Minibatch = 1
        coreference_total_scrores_max_index = torch.cuda.LongTensor()
        coreference_total_loss = 0
        for i in range(len(sentence.mentions)):
            coreference_input = torch.zeros((1, 3*self.encoding_hidden_dim*self.padding_dim_coef + s
elf.distance_dim), device=torch.device('cuda'))  # epsilon
            mention_attention = sentence.mentions[i].attention
            mention_position = sentence.mentions[i].words[0].position
            for cluster in clusters:
                cluster_attention = cluster.mentions[0].attention
                cluster_last_position = cluster.mentions[-1].words[0].position
                distance = self.coreferenceDistanceLayer(mention_position - cluster_last_position)
                new_input = torch.cat((mention_attention, cluster_attention.detach(), torch.mul(ment
ion_attention, cluster_attention.detach()), distance), dim=1)
```

```
                coreference_input = torch.cat((coreference_input, new_input), dim=0)

            coreference_scores = self.coreferenceScoreTLayer(coreference_input)
            coreference_scores_max_index = torch.max(coreference_scores, dim=1).indices
            coreference_total_scrores_max_index = torch.cat((coreference_total_scrores_max_index, co
reference_scores_max_index.view(1, 1)), dim=1)
            coreference_loss = coreference_loss_function(coreference_scores, sentence.coreference_go
ldens[i])
            coreference_total_loss += coreference_loss
            if self.training:
                coreference_loss.backward(retain_graph=True)
            self._extract_clusters(sentence, clusters, i)

        # Relations - Minibatch = sentence
        relation_input = torch.cuda.FloatTensor()
        relation_scores_max_index = torch.cuda.LongTensor()
        relation_loss = 0
        for i in range(len(sentence.mentions)):
            mention_attention = sentence.mentions[i].attention
            mention_position = sentence.mentions[i].words[0].position
            for j in range(i):
                previous_mention_attention = sentence.mentions[j].attention
                previous_mention_position = sentence.mentions[j].words[0].position
                distance = self.relationDistanceLayer(mention_position - previous_mention_position)
                new_input = torch.cat((mention_attention, previous_mention_attention, torch.mul(ment
ion_attention, previous_mention_attention), distance), dim=1)
                relation_input = torch.cat((relation_input, new_input), dim=0)

        if relation_input.size()[0] > 0:
            relation_scores = self.relationScoreLayer(relation_input)
            relation_scores_max_index = torch.max(relation_scores, dim=1).indices
            relation_loss = relation_loss_function(relation_scores, sentence.relation_goldens)
            if self.training:
                relation_loss.backward(retain_graph=True)
            self._extract_relations(sentence, clusters)

        return mention_loss, coreference_total_loss, relation_loss, mention_scores_max_index, corefe
rence_total_scrores_max_index, relation_scores_max_index

    def _extract_mentions(self, sentence):
        mentions = []
        for i in range(len(sentence.words)):
            # O tag
            if (sentence.mention_goldens[i] == 0):
                if (all(mention.head > -1 for mention in mentions)):
```

```
                    self._extract_attention(mentions)
                    sentence.mentions.extend(mentions)
                    mentions.clear()
            # H tag
            elif (sentence.mention_goldens[i] == 1):
                if (len(mentions) == 0):
                    mention = Mention(sentence.words[i], 0)
                    mentions.append(mention)
                else:
                    if (all(mention.head == -1 for mention in mentions)):
                        for mention in mentions:
                            mention.words.append(sentence.words[i])
                            mention.head = len(mention.words) - 1
                    else:
                        self._extract_attention(mentions)
                        sentence.mentions.extend(mentions)
                        mentions.clear()
                        mention = Mention(sentence.words[i], 0)
                        mentions.append(mention)

            # R tag
            elif (sentence.mention_goldens[i] == 2):
                if (len(mentions) == 0):
                    mention = Mention(sentence.words[i])
                    mentions.append(mention)
                else:
                    inserted = False
                    for mention in mentions:
                        if (mention.words[-1].position == sentence.words[i].position - 1):
                            mention.words.append(sentence.words[i])
                            inserted = True
                            break
                    if (not inserted):
                        mention = Mention(sentence.words[i])
                        mentions.append(mention)

    def _extract_clusters(self, sentence, clusters, index):
        if (sentence.coreference_goldens[index] == 0):
            cluster = Cluster(sentence.mentions[index])
            clusters.append(cluster)
            sentence.mentions[index].cluster = len(clusters) - 1
        else:
            clusters[sentence.coreference_goldens[index] -
1].mentions.append(sentence.mentions[index])
            sentence.mentions[index].cluster = sentence.coreference_goldens[index] - 1
```

```
    def _extract_attention(self, mentions):
        for mention in mentions:
            if self.technique == 'head':
                mention.attention = mention.words[mention.head].encoding
            elif self.technique == 'padding':
                att = torch.cuda.FloatTensor()
                max = 7*self.encoding_hidden_dim
                for word in mention.words:
                    att = torch.cat((att, word.encoding), dim=1)
                mention.attention = F.pad(att, (0, max-att.size(1)), mode='constant', value=0)
            elif self.technique == 'weightedaverage':
                encodings = mention.words[mention.head].encoding
                weights = torch.cuda.FloatTensor([[len(mention.words)]])
                for i in range(len(mention.words)):
                    if i != mention.head:
                        encodings = torch.cat((encodings, mention.words[i].encoding), dim=0)
                        weights = torch.cat((weights, torch.cuda.FloatTensor([[1]])), dim=0)

                weights = F.softmax(weights, dim=0)
                mention.attention = torch.sum(torch.mul(encodings, weights), dim=0).view(1, -1)
            else :
                encodings = torch.cuda.FloatTensor()
                for word in mention.words:
                    encodings = torch.cat((encodings, word.encoding), dim=0)
                alphas = self.attention(encodings)
                weights = F.softmax(alphas, dim=0)
                mention.attention = torch.sum(torch.mul(encodings, weights), dim=0).view(1, -1)

    def _extract_relations(self, sentence, clusters):
        reference = (-1 + sqrt(1 + 8 * len(sentence.relation_goldens))) / 2
        iterator = 0
        for i in range(int(reference) + 1):
            for j in range(i):
                if (sentence.relation_goldens[iterator] > 1):
                    clusters[sentence.mentions[i].cluster].relations[sentence.relation_goldens[itera
tor].item()].append(sentence.mentions[j].cluster)
                iterator += 1


class Training():
    def __init__(self, model, train_corpus, val_corpus, test_corpus, freezed_layers, lr):
        self.train_corpus = train_corpus
        self.val_corpus = val_corpus
        self.test_corpus = test_corpus
```

```
        self.freezed_layers = freezed_layers
        self.early_stopping = False
        self.epoch_counter_no_improve = 0
        self.min_loss = 10000
        self.model = cudaize(model)
        self.optimizer = optim.Adam(self.model.parameters(), lr=lr)

    def freeze_model(self):
        if self.model.method == 'BERT':
            counter = 0
            bert_embeddings_layers = [self.model.encodingLayer.bert.embeddings]
            bert_encoder_layers = self.model.encodingLayer.bert.encoder.layer
            bert_pooler_layers = [self.model.encodingLayer.bert.pooler]

            for layer in bert_embeddings_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False

            for layer in bert_encoder_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False

            for layer in bert_pooler_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False
        elif self.model.method == 'GPT':
            counter = 0
            gpt_wte_layers = [self.model.encodingLayer.gpt.wte]
            gpt_wpe_layers = [self.model.encodingLayer.gpt.wpe]
            gpt_drop_layers = [self.model.encodingLayer.gpt.drop]
            gpt_h_layers = self.model.encodingLayer.gpt.h
            gpt_lnf_layers = [self.model.encodingLayer.gpt.ln_f]

            for layer in gpt_wte_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False
```

```
            for layer in gpt_wpe_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False

            for layer in gpt_drop_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False

            for layer in gpt_h_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False

            for layer in gpt_lnf_layers:
                if counter < self.freezed_layers:
                    counter += 1
                    for param in layer.parameters():
                        param.requires_grad = False
        else :
            return

    def train(self, num_epochs):
        self.freeze_model()

        for epoch in range(1, num_epochs + 1):
            print('EPOCH %d' % epoch)
            self.model.zero_grad()
            self.train_epoch()
            self.evaluate_epoch(self.val_corpus, epoch)

            if self.early_stopping == True:
                break

        print('----- TEST -----\n')
        self.evaluate(self.test_corpus)
        print('\n')

    def train_epoch(self):
        self.model.train()
```

```
        epoch_mention_loss = torch.Tensor()
        epoch_coreference_loss = torch.Tensor()
        epoch_relation_loss = torch.Tensor()

        for document in self.train_corpus.documents:
            document_mention_loss, document_coreference_loss, document_relation_loss = self.train_do
cument(document)

            epoch_mention_loss = torch.cat((epoch_mention_loss, torch.Tensor([document_mention_loss]
)), dim=0)
            epoch_coreference_loss = torch.cat((epoch_coreference_loss, torch.Tensor([document_coref
erence_loss])), dim=0)
            epoch_relation_loss = torch.cat((epoch_relation_loss, torch.Tensor([document_relation_lo
ss])), dim=0)

        print('Training   -- Loss Mention: %f | Loss Coreference: %f | Loss Relation: %f' \
            % (torch.mean(epoch_mention_loss, dim=0), torch.mean(epoch_coreference_loss, dim=0), tor
ch.mean(epoch_relation_loss, dim=0)))

    def train_document(self, document):
        mention_loss_function = nn.CrossEntropyLoss(weight=torch.cuda.FloatTensor([1.0, 2.0, 4.0]))
        coreference_loss_function = nn.CrossEntropyLoss()
        relation_loss_function = nn.CrossEntropyLoss(weight=torch.cuda.FloatTensor([1.0, 4.0, 2.0, 4
.0, 2.0]))
        document_mention_loss = 0
        document_coreference_loss = 0
        document_relation_loss = 0

        for i in range(len(document.sentences)):
            self.optimizer.zero_grad()

            mention_loss, coreference_loss, relation_loss, _, _, _ = self.model(document.sentences[i
], document.clusters, mention_loss_function, coreference_loss_function, relation_loss_function)

            document_mention_loss += mention_loss.item()
            if coreference_loss != 0:
                document_coreference_loss += coreference_loss.item()
            if relation_loss != 0:
                document_relation_loss += relation_loss.item()

            self.optimizer.step()

            document.sentences[i].mentions.clear()

        document.clusters.clear()
```

```
        return document_mention_loss, document_coreference_loss, document_relation_loss

    def evaluate_epoch(self, corpus, epoch):
        self.model.eval()

        corpus_mention_loss = torch.Tensor()
        corpus_coreference_loss = torch.Tensor()
        corpus_relation_loss = torch.Tensor()
        corpus_mention_predictions = []
        corpus_coreference_predictions = []
        corpus_relation_predictions = []
        corpus_mention_goldens = []
        corpus_coreference_goldens = []
        corpus_relation_goldens = []

        for document in corpus.documents:
            document_mention_loss, document_coreference_loss, document_relation_loss, document_menti
on_predictions, document_coreference_predictions, document_relation_predictions, document_mention_go
ldens, document_coreference_goldens, document_relation_goldens = self.evaluate_document(document)

            corpus_mention_loss = torch.cat((corpus_mention_loss, torch.Tensor([document_mention_los
s])), dim=0)
            corpus_coreference_loss = torch.cat((corpus_coreference_loss, torch.Tensor([document_cor
eference_loss])), dim=0)
            corpus_relation_loss = torch.cat((corpus_relation_loss, torch.Tensor([document_relation_
loss])), dim=0)

            corpus_mention_predictions.extend(document_mention_predictions)
            corpus_coreference_predictions.extend(document_coreference_predictions)
            corpus_relation_predictions.extend(document_relation_predictions)

            corpus_mention_goldens.extend(document_mention_goldens)
            corpus_coreference_goldens.extend(document_coreference_goldens)
            corpus_relation_goldens.extend(document_relation_goldens)

        print('Evaluation -- Loss Mention: %f | Loss Coreference: %f | Loss Relation: %f' \
            % (torch.mean(corpus_mention_loss, dim=0), torch.mean(corpus_coreference_loss, dim=0), t
orch.mean(corpus_relation_loss, dim=0)))

        agregated_loss = torch.mean(corpus_mention_loss, dim=0)*7 + torch.mean(corpus_coreference_lo
ss, dim=0) + torch.mean(corpus_relation_loss, dim=0)*4
        if agregated_loss < self.min_loss:
            self.save_model('model.pth')
            self.epoch_counter_no_improve = 0
```

```
            self.min_loss = agregated_loss
        else:
            self.epoch_counter_no_improve += 1

        if (self.epoch_counter_no_improve > 4) or (epoch > 7 and self.epoch_counter_no_improve > 2):
            self.early_stopping = True

    def evaluate_document(self, document):
        mention_loss_function = nn.CrossEntropyLoss(weight=torch.cuda.FloatTensor([1.0, 2.0, 4.0]))
        coreference_loss_function = nn.CrossEntropyLoss()
        relation_loss_function = nn.CrossEntropyLoss(weight=torch.cuda.FloatTensor([1.0, 4.0, 2.0, 4
.0, 2.0]))
        document_mention_loss = 0
        document_coreference_loss = 0
        document_relation_loss = 0
        document_mention_predictions = []
        document_coreference_predictions = []
        document_relation_predictions = []
        document_mention_goldens = []
        document_coreference_goldens = []
        document_relation_goldens = []

        with torch.no_grad():
            for i in range(len(document.sentences)):
                mention_loss, coreference_loss, relation_loss, mention_scores_max_index, coreference
_scrores_max_index, relation_scores_max_index = self.model(document.sentences[i], document.clusters,
 mention_loss_function, coreference_loss_function, relation_loss_function)

                document_mention_loss += mention_loss.item()
                document_mention_predictions.extend(mention_scores_max_index.tolist())
                document_mention_goldens.extend(document.sentences[i].mention_goldens.tolist())

                if coreference_loss != 0:
                    document_coreference_loss += coreference_loss.item()
                    document_coreference_predictions.extend(coreference_scrores_max_index.tolist())
                    document_coreference_goldens.extend(document.sentences[i].coreference_goldens.to
list())
                if relation_loss != 0:
                    document_relation_loss += relation_loss.item()
                    document_relation_predictions.extend(relation_scores_max_index.tolist())
                    document_relation_goldens.extend(document.sentences[i].relation_goldens.tolist()
)

                document.sentences[i].mentions.clear()
```

```
            document.clusters.clear()

        return document_mention_loss, document_coreference_loss, document_relation_loss, document_me
ntion_predictions, document_coreference_predictions, document_relation_predictions, document_mention
_goldens, document_coreference_goldens, document_relation_goldens

    def evaluate(self, corpus):
        self.load_model('model.pth')
        self.model.eval()

        corpus_mention_loss = torch.Tensor()
        corpus_coreference_loss = torch.Tensor()
        corpus_relation_loss = torch.Tensor()
        corpus_mention_predictions = []
        corpus_coreference_predictions = []
        corpus_relation_predictions = []
        corpus_mention_goldens = []
        corpus_coreference_goldens = []
        corpus_relation_goldens = []

        for document in corpus.documents:
            document_mention_loss, document_coreference_loss, document_relation_loss, document_menti
on_predictions, document_coreference_predictions, document_relation_predictions, document_mention_go
ldens, document_coreference_goldens, document_relation_goldens = self.evaluate_document(document)

            corpus_mention_loss = torch.cat((corpus_mention_loss, torch.Tensor([document_mention_los
s])), dim=0)
            corpus_coreference_loss = torch.cat((corpus_coreference_loss, torch.Tensor([document_cor
eference_loss])), dim=0)
            corpus_relation_loss = torch.cat((corpus_relation_loss, torch.Tensor([document_relation_
loss])), dim=0)

            corpus_mention_predictions.extend(document_mention_predictions)
            corpus_coreference_predictions.extend(document_coreference_predictions)
            corpus_relation_predictions.extend(document_relation_predictions)

            corpus_mention_goldens.extend(document_mention_goldens)
            corpus_coreference_goldens.extend(document_coreference_goldens)
            corpus_relation_goldens.extend(document_relation_goldens)

        print('--Mentions statistics--\n')
        mention_report = classification_report(y_pred=corpus_mention_predictions, y_true=corpus_ment
ion_goldens, digits=3, zero_division=0)
        print(mention_report)
```

```
        print('--Coreferences statistics--\n')
        prediction_clusters, golden_clusters = self._extract_clusters(corpus_coreference_predictions
, corpus_coreference_goldens)
        precision = bcubed.precision(prediction_clusters, golden_clusters)
        recall = bcubed.recall(prediction_clusters, golden_clusters)
        fscore = bcubed.fscore(precision, recall)
        print('Precision: %f' % precision)
        print('Recall: %f' % recall)
        print('F1: %f \n' % fscore)

        print('--Relations statistics--\n')
        relation_report = classification_report(y_pred=corpus_relation_predictions, y_true=corpus_re
lation_goldens, digits=3, zero_division=0)
        print(relation_report)

    def _extract_clusters(self, predictions, goldens):
        """ Extract clusters from raw corefernce scoring data """
        prediction_clusters = self._extract_clusters_from_list(predictions)
        golden_clusters = self._extract_clusters_from_list(goldens)

        return prediction_clusters, golden_clusters

    def _extract_clusters_from_list(self, list):
        """ Extract clusters from a list """
        clusters = {}
        iterator = 1
        for i in range(len(list)):
            if (list[i] == 0):
                clusters[i + 1] = set([iterator])
                iterator += 1
            else:
                clusters[i + 1] = set(list[i])

        return clusters

    def save_model(self, savepath):
        torch.save(self.model.state_dict(), savepath)

    def load_model(self, loadpath):
        state = torch.load(loadpath)
        self.model.load_state_dict(state)
        self.model = cudaize(self.model)

# 'RNN', 'BERT' ou 'GPT'
METHOD = 'RNN'
```

```
# 'head', 'padding', 'weightedaverage' ou 'autoattention'
TECHNIQUE = 'head'

EMBEDDING_DIM = 50
# 100 pour RNN, 768 pour BERT ou GPT
ENCODING_HIDDEN_DIM = 100
MENTION_DIM = 3
RELATION_DIM = 5

FREEZED_LAYERS = 12
DECODING_HIDDEN_DIM = 150
DISTANCE_DIM = 50
DROPOUT = 0.2
LEARNING_RATE = 0.001

EPOCHS = 30

embeddings_index, mentions_index = prepare_index()
train_corpus, val_corpus, test_corpus = prepare_corpora(embeddings_index, mentions_index)

model = Model(METHOD, TECHNIQUE, EMBEDDING_DIM, ENCODING_HIDDEN_DIM, DECODING_HIDDEN_DIM, MENTION_DI
M, RELATION_DIM, DISTANCE_DIM, DROPOUT)
training = Training(model, train_corpus, val_corpus, test_corpus, FREEZED_LAYERS, lr=LEARNING_RATE)
training.train(EPOCHS)
```

ANNEXE II RÉSULTATS OBTENUS AVEC LES DIFFÉRENTES COMBINAISONS DE PARAMÈTRES POUR LE MODÈLE RNN

Technique d'encodage des mentions 'Jeton central'

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,8	76,6	74,9	45,2	77,2	57,0	62,3	66,9	61,1

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,4	86.2	86,7	45,7	77,1	57,4	72,3	67,2	68,8

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,7	79,1	78,1	44,3	68,4	53,8	61,5	61,7	59,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,0	**86,2**	**86,0**	**45,6**	**81,2**	**58,4**	**72,9**	**67,8**	**69,5**

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,2	81,8	81,4	44,2	73,6	55,2	64,2	68,4	62,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,3	85,3	86,0	46,7	82,2	59,6	70,1	59,3	62,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,3	80,9	78,0	44,0	77,7	56,2	55,7	30,4	32,4

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,6	86,6	87,2	46,8	78,1	58,6	70,0	61,4	63,8

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,9	82,9	81,9	45,8	90,4	60,8	59,8	42,5	44,5

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,7	87,0	87,2	46,0	81,9	58,9	71,4	65,1	66,8

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,8	81,6	80,2	43,1	67,9	52,8	61,7	37,0	41,3

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,0	85,2	85,8	45,2	77,4	57,0	71,4	63,6	66,0

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,6	79,6	77,1	45,5	83,1	58,8	57,1	40,7	43,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
7,6	86,9	87,0	42,6	81,4	55,9	72,3	67,5	68,0

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,3	79,6	76,6	43,6	71,1	54,1	58,1	64,2	58,0

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,4	86,4	86,7	45,1	80,3	57,8	68,4	62,0	64,0

Technique d'encodage des mentions 'Concaténation

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,8	77,2	76,6	45,7	89,8	60,6	62,5	66,9	58,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,5	**84,8**	**85,5**	**48,3**	**81,6**	**60,6**	**69,1**	**64,2**	**65,9**

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
84,6	80,0	79,0	46,2	77,2	57,8	63,2	42,5	45,8

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,0	83,9	85,0	45,0	81,7	58,1	71,5	63,9	66,3

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,0	77,5	76,1	47,4	88,4	61,7	56,9	65,4	57,3

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,1	85,4	86,2	47,1	83,1	60,1	70,2	59,3	62,5

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,5	76,1	75,6	43,6	69,8	53,7	62,5	55,1	54,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,2	83,7	84,6	48,2	73,8	58,3	72,4	66,6	68,0

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,4	80,9	79,0	45,5	78,8	57,6	58,1	36,4	38,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,4	85,0	85,8	44,1	82,9	57,6	71,3	62,7	64,9

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,2	79,7	76,9	45,9	78,1	57,9	65,8	29,5	30,2

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,7	84,1	85,3	47,4	81,2	59,9	69,3	64,2	65,4

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,5	80,3	78,0	47,5	89,6	62,1	51,4	43,7	44,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,8	85,7	86,4	46,0	82,3	59,0	73,1	64,2	66,2

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
7,3	80,6	77,9	47,3	91,9	62,5	62,6	40,4	43,7

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,0	87,0	87,4	46,6	77,8	58,3	68,3	64,2	65,6

Technique d'encodage des mentions 'Moyenne pondérée'

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,0	77,6	76,8	45,1	83,9	58,6	69,3	43,1	45,8

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,0	84,7	85,2	45,0	81,7	58,0	74.5	63,9	66,5

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,9	79,5	78,3	43,4	76,0	55,3	67,6	69,3	58,3

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
85,7	84,4	84,9	45,1	77,8	57,1	73,8	65,4	67,6

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,6	79,0	77,0	43,4	67,3	52,8	54,6	55,1	53,7

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
84,6	82,5	83,4	44,3	85,2	58,3	71,0	66,0	67,5

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,1	80,4	80,0	44,7	79,1	57,2	65,9	65,7	59,3

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
89,1	86,1	87,0	44,4	71,3	54,7	71,7	60,2	63,1

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,3	76,8	76,7	44,7	86,5	59,0	56,0	45,5	46,3

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,4	85,1	85,6	45,4	76,7	57,0	73,1	68,7	69,9

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,2	80,2	78,1	45,2	82,8	58,5	54,0	56,6	53,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,0	**86,0**	**86,7**	**45,8**	**82,9**	**59,0**	**72,0**	**68,7**	**69,1**

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,6	78,8	77,3	44,7	75,7	56,2	66,7	64,2	60,3

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,1	85,5	86,1	45,9	78,2	57,9	71,7	65,1	67,0

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,5	82,3	81,8	45,3	74,9	56,5	65,3	65,7	58,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,0	85,8	86,3	44,4	74,1	55,5	70,2	65,1	66,8

Technique d'encodage des mentions 'Auto-attention'

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,2	78,7	76,6	44,6	84,2	58,3	59,6	62,7	60,2

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,0	84,8	85,3	43,7	78,1	56,1	70,5	63,0	65,1

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,3	79,1	77,3	46,5	83,7	59,8	62,4	33,1	34,6

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,6	86,9	87,1	44,6	72,9	55,3	70,6	65,1	66,9

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,8	75,1	73,1	43,5	73,6	54,7	55,2	46,7	47,7

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,5	**84,8**	**85,3**	**45,2**	**75,5**	**56,5**	**71,3**	**67,2**	**68,7**

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,8	78,7	77,5	46,8	96,0	62,9	61,6	38,9	41,0

ENCODING_HIDDEN_DIM: 50, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
86,3	85,8	86,1	44,2	78,1	56,5	70,8	61,7	64,3

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,4	80,0	79,6	44,6	79,2	57,1	69,2	39,8	41,6

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,0	86,0	86,6	44,1	83,7	57,8	69,4	63,9	65,8

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,3	80,4	79,1	43,5	79,7	56,3	56,1	67,8	56,8

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,0	85,2	86,0	45,9	76,3	57,3	69,8	65,4	66,8

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,8	76,9	76,2	45,5	79,1	57,8	62,2	62,3	58,3

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
88,2	86,6	87,1	45,8	78,3	57,8	68,3	60,8	62,5

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.010000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
71,8	76,0	72,7	51,4	93,6	66,4	52,6	58,7	54,7

ENCODING_HIDDEN_DIM: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
87,1	**86,0**	**86,4**	**46,6**	**78,8**	**58,6**	**69,9**	**63,9**	**65,5**

ANNEXE III RÉSULTATS OBTENUS AVEC LES DIFFÉRENTES COMBINAISONS DE PARAMÈTRES POUR LE MODÈLE BERT

Technique d'encodage des mentions 'Jeton central'

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	59,6	55,7	54,8

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	54,8	55,7	53,6

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,5	80,1	78,9	49,8	100,0	66,5	59,0	52,7	54,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,6	83,3	81,7	49,1	100,0	65,9	67,6	53,0	55,0

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
80,0	81,4	80,0	49,0	97,7	65,3	67,4	54,8	57,6

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,4	82,5	81,1	48,0	100,0	64,8	65,3	57,8	59,2

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,2	90,1	90,4	45,1	69,1	54,6	69,6	68,1	68,5

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	**89,5**	**89,9**	**44,4**	**76,0**	**56,0**	**73,4**	**68,4**	**69,8**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,6	87,4	88,4	46,9	76,9	58,3	70,0	66,3	66,8

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,5	90,4	90,8	44,1	74,9	55,5	71,4	64,8	66,6

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,8	88,4	89,1	44,9	69.1	54,5	72,9	67,2	68,8

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,6	88,3	89,0	45,2	75,2	56,5	71,1	67,8	68,9

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
92,1	90,6	91,0	45,3	75,8	56,7	73,3	61,4	63,8

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,1	89,7	90,1	44,6	68,7	54,1	73,9	67,2	68,9

Technique d'encodage des mentions 'Concaténation

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,0	76,1	76,2	49,8	100,0	66,5	66,2	61,4	61,8

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,4	79,5	76,8	49,8	100,0	66,5	59,0	59,3	57,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,0	81,8	80,0	46,2	96,0	62,3	63,2	50,3	52,9

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,9	82,8	80,7	49,4	100,0	66,2	65,7	57,5	58,4

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,4	89,2	89,8	44,3	80,6	57,2	71,8	64,8	66,7

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,1	88,1	88,8	43,7	74,1	55,0	72,4	69,0	70,1

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,4	**87,2**	**88,3**	**47,2**	**78,8**	**59,0**	**70,3**	**68,1**	**68,7**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
89,9	88,1	88,7	47,1	84,0	60,4	71,0	64,5	66,1

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,3	89,5	90,1	45,3	83,7	58,8	72,2	63,9	65,9

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,1	89,0	89,6	47,3	81,2	59,8	65,8	63,6	64,2

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	87,9	88,8	46,5	78,8	58,5	70,6	67,5	68,6

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,0	89,5	90,0	45,1	86,8	59,4	69,7	62,7	64,5

Technique d'encodage des mentions 'Moyenne pondérée'

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	50,8	67,5	57,6

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	63,5	55,1	55,8

FREEZED_LAYERS: 5, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	50,9	69,4	56,7

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	50,8	67,5	57,6

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	53,5	69,0	57,7

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
54,5	73,9	62,7	49,8	100,0	66,5	61,3	56,6	56,2

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
80,6	81,0	79,8	49,8	100,0	66,5	64,5	54,5	55,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,9	79,8	78,6	49,1	100,0	65,9	68,2	58,1	60,6

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,0	83,0	81,4	49,5	96,0	65,3	71,4	51,8	53,1

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,5	85,8	83,5	51,1	97,7	67,1	66,0	55,4	57,7

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,6	88,6	89,3	44,8	72,7	55,4	71,9	67,8	69,2

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,6	89,3	90,0	47,4	76,1	58,4	71,6	61,4	63,8

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,9	89,5	90,0	45,4	75,3	56,7	70,5	65,4	66,7

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	**88,6**	**89,3**	**45,6**	**68,9**	**54,9**	**75,9**	**72,6**	**73,6**

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,7	89,5	90,2	46,0	72,7	56,3	73,9	65,4	67,3

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,2	87,1	88,1	45,7	75,3	56,9	72,9	66,9	68,6

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,0	97,4	88,2	45,3	72,7	55,8	73,8	67,8	69,2

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,1	87,8	88,8	46,0	73,8	56,7	71,7	64,2	66,0

Technique d'encodage des mentions 'Auto-attention'

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,0	83,9	82,0	49,8	100,0	66,5	57,7	53,0	53,8

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,4	82,1	80,1	48,7	100,0	65,5	69,7	60,8	63,3

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,0	79,0	78,0	48,3	96,4	64,4	61,7	53,3	55,1

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,4	77,3	76,0	49,3	100,0	66,1	74,7	57,5	59,7

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	88,4	89,1	44,1	77,1	56,1	69,4	64,5	65,9

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,5	89,9	90,3	44,9	72,7	55,5	71,9	64,5	66,3

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,8	90,6	90,9	46,8	76,1	58,0	69,8	63,0	64,8

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,3	88,5	89,4	44,8	76,4	56,5	71,9	66,9	68,3

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	89,7	90,0	44,3	71,8	54,8	70,4	65,4	66,7

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,7	**88,5**	**89,2**	**45,3**	**69,0**	**54,7**	**73,3**	**69,9**	**70,9**

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
90,5	88,3	89,0	45,6	73,3	56,3	71,2	66,6	67,9

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
91,2	89,4	90,0	45,5	79,5	57,9	72,0	63,3	65,3

ANNEXE IV RÉSULTATS OBTENUS AVEC LES DIFFÉRENTES COMBINAISONS DE PARAMÈTRES POUR LE MODÈLE GPT

Technique d'encodage des mentions 'Jeton central'

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,8	75,0	74,1	49,8	100,0	66,5	55,3	63,9	58,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
71,5	77,3	74,1	48,1	96,1	64,1	70,1	58,1	60,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
70,9	77,2	72,1	49,8	100,0	66,5	54,3	69,6	60,8

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,9	**77,7**	**76,0**	**49,6**	**100,0**	**66,3**	**67,8**	**65,7**	**65,1**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,4	76,0	75,2	51,5	97,7	67,5	67,4	56,3	58,4

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,2	77,4	72,0	47,2	94,7	63,0	66,0	63,3	64,1

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,1	78,6	78,1	47,8	69,0	63,8	73,1	59,0	61,8

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,0	76,3	75,5	49,5	100,0	66,2	69,5	59,3	60,4

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
71,6	74,6	73,0	49,7	100,0	66,4	64,9	63,3	59,8

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,1	76,5	74,2	48,7	100,0	65,5	62,6	60,8	59,2

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,6	74,5	73,9	49,8	100,0	66,5	55,1	64,5	59,0

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
79,2	75,3	75,1	49,4	100,0	66,2	66,5	61,4	61,2

FREEZED_LAYERS: 15, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,1	78,8	75,6	49,8	100,0	66,5	45,8	55,1	50,0

FREEZED_LAYERS: 15, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,5	78,3	77,0	49,8	100,0	66,5	50,3	17,5	13,8

FREEZED_LAYERS: 15, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,3	77,6	75,0	49,8	100,0	66,5	45,5	67,5	54,4

FREEZED_LAYERS: 15, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,3	75,1	72,8	49,8	100,0	66,5	46,2	63,6	53,2

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,3	78,4	75,7	49,8	100,0	66,5	47,1	63,6	53,9

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
78,2	73,3	73,4	49,8	100,0	66,5	51,7	69,0	58,1

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,2	77,0	74,6	49,8	100,0	66,5	54,2	22,6	22,0

FREEZED_LAYERS: 100, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
77,4	78,3	77,7	49,8	100,0	66,5	45,5	66,9	54,2

Technique d’encodage des mentions ‘Concaténation

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,2	77,6	74,7	49,8	100,0	66,5	63,2	60,5	58,9

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,8	74,1	73,7	49,6	100,0	66,3	63,5	61,7	61,3

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,0	75,5	74,0	49,8	95,5	65,5	67,1	63,9	63,1

FREEZED_LAYERS: 10, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,1	76,8	74,4	48,6	96,6	64,7	66,5	61,1	62,2

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,3	77,0	75,1	49,8	100,0	66,5	50,4	64,5	56,5

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,1	76,7	74,3	49,8	100,0	66,5	60.6	64,8	61,3

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,6	75,6	73,9	49,8	100,0	66,5	61,9	64,5	60,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,1	77,4	74,6	49,8	100,0	66,5	62,0	65,1	61,8

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,6	77,8	76,3	49,8	100,0	66,5	61,6	64,2	61,6

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
70,4	76,5	73,1	49,8	100,0	66,5	64,4	61,1	59,9

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,2	**78,6**	**75,7**	**49,8**	**100,0**	**66,5**	**66,2**	**68,1**	**65,0**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,4	76,9	75,0	49,8	100,0	66,5	62,6	59,9	58,8

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
65,7	71,7	68,4	49,8	100,0	66,5	71,4	63,9	63,9

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
81,9	76,1	75,4	49,8	100,0	66,5	60,4	59,0	57,9

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,9	74,9	74,3	49,8	100,0	66,4	69,0	64,8	62,9

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,9	77,0	74,9	46,7	97,2	63,1	68,8	66,6	66,8

Technique d'encodage des mentions 'Moyenne pondérée'

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,3	76,1	74,8	50,5	97,7	66,6	66,5	55,4	57,7

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
70,9	77,3	73,0	48,8	96,0	64,7	73,2	64,2	66,2

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,8	77,8	76,1	49,3	100,0	66,0	68,7	54,8	57,1

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
71,4	**77,3**	**72,0**	**49,4**	**100,0**	**66,2**	**68,3**	**66,0**	**66,0**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,3	78,3	74,9	47,9	65,7	63,8	66,0	59,3	60,6

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,9	77,6	75,6	49,8	100,0	66,5	64,3	60,5	59,0

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,2	74,1	73,5	47,6	96,1	63,7	66,5	59,9	61,9

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,7	78,2	72,9	49,8	100,0	66,5	68,7	66,3	61,7

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,7	76,7	75,6	49,8	100,0	66,5	54,5	65,1	59,0

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,7	76,0	74,8	49,8	100,0	66,5	57,4	64,2	59,6

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,5	73,1	72,2	49,8	100,0	66,5	62,5	61,1	60,5

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,4	77,2	75,8	49,8	100,0	66,5	68,4	60,2	57,6

Technique d'encodage des mentions 'Auto-attention'

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,6	77,8	75,6	49,8	100,0	66,5	53,2	67,2	59,4

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,9	**73,9**	**74,6**	**49,3**	**100,0**	**66,1**	**72,5**	**59,9**	**62,6**

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,3	78,4	76,6	49,8	100,0	66,5	55,1	66,6	60,1

FREEZED_LAYERS: 12, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,5	70,3	71,7	51,5	97,7	67,4	66,4	60,8	61,4

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
73,6	76,8	75,0	49,8	95,7	65,5	64,7	62,0	60,5

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,9	**77,6**	**75,2**	**49,8**	**100,0**	**66,5**	**67,8**	**68,4**	**61,6**

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
72,4	76,1	74,2	49,8	100,0	66,5	56,9	61,7	58,0

FREEZED_LAYERS: 13, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,7	**77,6**	**76,4**	**49,8**	**100,0**	**66,5**	**57,2**	**67,5**	**60,4**

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
76,7	76,5	75,6	49,8	100,0	66,5	54,3	61,1	56,6

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 100, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
80,3	72,7	73,0	49,8	100,0	66,5	66,4	61,4	62,7

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 20, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
74,7	**77,0**	**75,5**	**49,5**	**100,0**	**66,3**	**60,6**	**65,1**	**61,5**

FREEZED_LAYERS: 14, DECODING_HIDDEN_DIM: 150, DISTANCE_DIM: 50, DROPOUT: 0.200000, LEARNING_RATE: 0.001000

Détection de mentions			Résolution de coréférences			Classification de relations		
Précision	Rappel	F1	Précision	Rappel	F1	Précision	Rappel	F1
75,1	73,5	74,2	48,0	100,0	64,9	71,3	61,7	64,1