APPRENTISSAGE PROFOND POUR LE TRAITEMENT AUTOMATIQUE DU LANGAGE : APPLICATION À LA CHIMIE

19 novembre 2020

Mémoire de Master 2

Rédigé sous la direction de M. Sylvain Desroziers

par

Amina Ghoul

Remerciements

Avant tout développement sur cette expérience professionnelle, il apparaît opportun de commencer ce rapport de stage par des remerciements, à ceux qui m'ont beaucoup appris au cours de ce stage, et même à ceux qui ont eu la gentillesse de faire de ce stage un moment très profitable.

Aussi, je remercie Monsieur Sylvain Desroziers, mon encadrant qui m'a formé et accompagné tout au long de ce stage avec beaucoup de patience et de pédagogie. Enfin, je remercie l'ensemble des employés de IFPEN pour les conseils qu'ils ont pu me prodiguer au cours de ce stage.

Table des matières

1. Introduction

Du 24 juin au 30 novembre 2020, j'ai effectué un stage au sein de IFP Energies nouvelles (IFPEN) situé à Rueil-Malmaison. Au cours de ce stage dans la direction *Sciences et Technologies du numérique*, j'ai pu m'intéresser à différentes méthodes de traitement du langage pour l'étude d'un corpus de brevets.

IFPEN est un établissement public français à caractère industriel et commercial ayant des missions de recherche et de formation dans les domaines de l'énergie et de l'environnement. Mon maître de stage étant Sylvain Desroziers, ingénieur calcul scientifique, j'ai pu apprendre dans d'excellentes conditions et j'ai bénéficié d'un soutien de qualité.

Le but du stage a été d'étudier l'apport de méthodes de traitement naturel du langage (ou NLP) pour aider à déterminer la brevetabilité d'une invention ou sa liberté d'exploitation. Pour ce faire, notre objectif a été de trouver des méthodes d'apprentissage profond performantes, pour les tâches de reconnaissance d'entités nommées et d'extraction de relation, afin de pouvoir extraire de l'information contenue dans des brevets appartenant au domaine de la catalyse hétérogène.

Dans ce rapport de stage, je commencerai par présenter l'entreprise IFPEN, le contexte du stage, puis les quatre parties suivantes.

Dans un premier temps, mon stage a consisté en une montée en compétence et une prise en main des outils de traitement du langage naturel (ou NLP) en appliquant différents modèles de Machine Learning et Deep Learning pour la classification de sentiments sur un jeu de données bien connu.

Dans un second temps, nous avons évalué différents modèles de Deep Learning pour une tâche de reconnaissance d'entités nommées sur, d'une part un jeu de données classique et fréquemment utilisé pour évaluer des modèles, et d'autre part des données de brevets annotées par les experts métiers IFPEN.

Dans une troisième partie, nous avons évalué dans la même idée des modèles pour une tâche d'extraction de relation sur un jeu de données classique, puis sur les données de brevets IFPEN.

Enfin, dans un dernier temps nous avons comparé les résultats obtenus pour ces tâches de reconnaissance d'entités nommées et d'extraction de relation sur le jeu de données brevets avec les résultats fournis par la plateforme Watson d'IBM.

2. Présentation de l'entreprise

IFPEN est un acteur majeur de la recherche et de la formation dans les domaines de l'énergie, du transport et de l'environnement. De la recherche à l'industrie, l'innovation technologique est au cœur de son action. IFPEN intervient dans quatre grands domaines :

- **mobilité durable** : motorisations hybrides et électriques, motorisations thermiques, véhicule connecté.
- **énergies renouvelables** : biocarburants, biogaz, chimie biosourcée, éolien offshore et énergies marines, géothermien, hydrogène, stockage d'énergie.
- **hydrocarbures responsables** : carburants, pétrochimie, traitement de gaz, bassin, réservoir, récupération améliorée (EOR), forage et production offshore.
- **climat et environnement** : captage, stockage et utilisation du CO_2, recyclage des plastiques, surveillance environnementale, analyse de cycle de vie, métaux critiques et terres rares.

Missions de IFPEN
Dans le cadre de la mission d'intérêt général confiée par les pouvoirs publics, IFPEN concentre ses efforts sur :

- l'**apport de solutions aux défis sociétaux de l'énergie et du climat**, en favorisant la transition vers une mobilité durable et l'émergence d'un mix énergétique plus diversifié ;
- la **création de richesse et d'emplois**, en soutenant l'activité économique française et européenne, et la compétitivité des filières industrielles associées.

Partie intégrante d'IFPEN, l'école d'ingénieurs IFP School prépare les générations futures à relever ces défis.

Un financement public/privé
IFPEN dispose d'un savoir-faire éprouvé sur l'ensemble de la chaîne de valeur allant de la recherche fondamentale jusqu'à l'innovation. Son financement est assuré à la fois par le budget de l'État et par des ressources propres, provenant de partenaires industriels. Ces dernières représentent plus de 50% du budget total d'IFPEN, une configuration quasi unique en France.

Organisation
La mise en œuvre des programmes de Recherche et Innovation (R&I) à IFPEN s'appuie sur une structure matricielle : les projets sont menés au sein d'équipes pluridisciplinaires constituées de

collaborateurs issus des directions de recherche et des directions fonctionnelles. Les centres de résultats sont eux responsables de l'élaboration et du suivi des programmes de R&I, ainsi que de leur valorisation industrielle.

Cette organisation apporte souplesse et efficacité en permettant d'utiliser les compétences en fonction des besoins. Elle offre également l'opportunité d'associer les talents, la pluridisciplinarité étant source d'innovation.

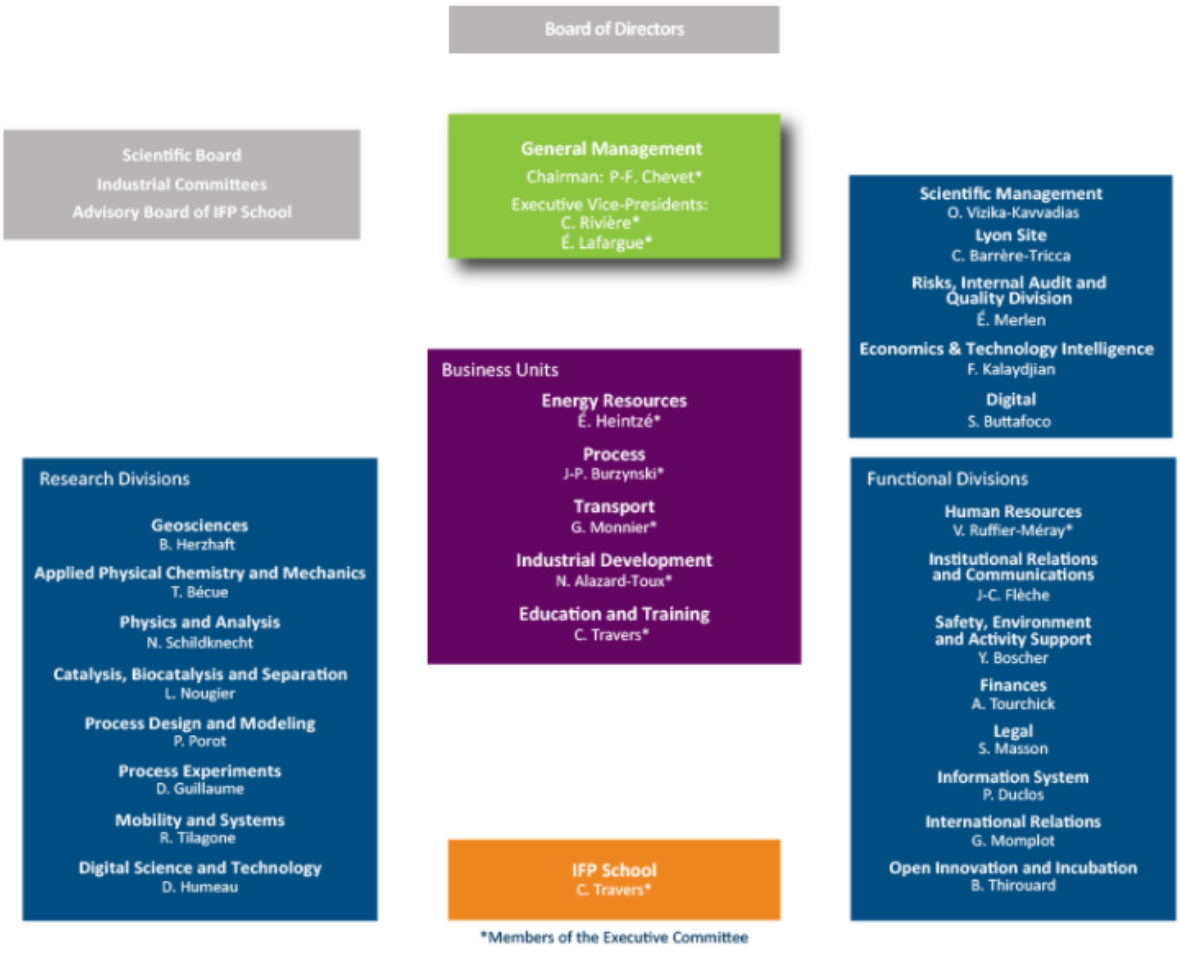


FIGURE 2.1: Organisation générale de IFPEN

Implantation régionale

IFPEN bénéficie d'une double implantation géographique :

- **Site de Rueil-Malmaison (dans la région Île-de-France)** : L'établissement de Rueil regroupe des moyens de conception, de modélisation, de simulation, d'expérimentation et de développement dans de nombreux domaines, tels que les motorisations thermiques, électriques et hybrides, la gestion du cycle de l'eau en contexte EOR, les logiciels de modélisation de bassin et de simulation de réservoir, a surveillance environnementale, l'analyse de cycle de vie.
  L'établissement abrite également IFP School, école d'application qui apporte à des étudiants et jeunes professionnels une formation complémentaire dans les domaines de l'énergie et de la mobilité durable.
- **Site de Lyon (dans la région Auvergne-Rhône-Alpes)** : Les travaux menés sur le site de Lyon permettent de valider la faisabilité industrielle des procédés et technologies développés par IFPEN, de s'assurer de leur fiabilité et de garantir leur transposition à l'échelle industrielle.
  L'établissement de Lyon regroupe des moyens de conception, de modélisation, d'expérimentation et de développement dans de nombreux domaines, parmi lesquels les procédés de production de carburants et de biocarburants, l'éolien offshore et les énergies marines, le stockage d'énergie, le captage du CO2.

IFPEN en chiffres (2019)

- 1 633 salariés (effectif total équivalent temps plein), dont 1 136 ingénieurs et techniciens en R&I.
- Près de 200 allocataires de recherche, postdoctorants et stagiaires.
- Plus de 50 métiers représentés.
- 283,3 M€ de budget, dont 236,2 M€ pour la R&I et un autofinancement à plus de 50%.
- 60% du budget consacré aux nouvelles technologies de l'énergie
- 185 premiers dépôts de brevets dont 94 dans les NTE.
- Plus de 600 publications scientifiques et communications à congrès.
- 71 projets de recherche collaboratifs en cours, dont 38 impliquant des partenaires étrangers.
- Plus de 500 élèves diplômés par IFP School.
- Plus de 30 entreprises créées par IFPEN depuis 1944.

Département d'accueil de Mathématiques appliquées
J'ai intégré le département de Mathématiques appliquées au sein de la direction *Sciences et Technologies du numérique* sur le site de Rueil-Malmaison. Ce département apporte ses compétences en informatique, traitement de l'information et mathématiques appliquées en complément des compétences des nombreux autres métiers de IFPEN, pour la mise en œuvre de projets internes et collaboratifs de R&I sur les aspects numériques au sens large. Plus spécifiquement,

les contributions du département se situent essentiellement dans l'optimisation et la maîtrise de systèmes technologiques complexes, dans les performances numériques et informatiques de codes scientifiques, et dans l'exploitation, à l'aide d'outils numériques, de grands volumes de données résultant d'expérimentations ou de simulations. Le département de Mathématiques appliquées regroupe des compétences dans les domaines suivants : schémas numériques, couplage de modèles, solveurs parallèles, optimisation, statistiques.

3. Contexte du stage

La protection de la propriété industrielle est un enjeu capital pour les acteurs de l'industrie, de la recherche et l'innovation comme IFPEN et ses filiales. De manière générale, notre système de propriété industrielle s'articule autour de la mise en place de monopoles d'exploitation des inventions protégés par la loi au moyen de brevets. Intégrant des composantes à la fois techniques et juridiques, un brevet se doit de définir légalement le périmètre d'une invention en intégrant une description précise et entière de celle-ci. Pour son titulaire, le brevet représente donc le « droit d'interdire » à autrui la reproduction de son invention sans son autorisation. Ne pas respecter le périmètre défini par un brevet peut ainsi mener à de lourdes compensations par les autorités. Les enjeux financiers portant sur cet aspect sont donc extrêmement clairs pour toute activité industrielle. De manière complémentaire, la liberté d'exploitation, que l'on peut définir ici comme le « droit de faire » est un enjeu tout aussi important en matière d'innovation. Savoir ce qui est déjà couvert ou non par des brevets est un avantage concurrentiel important accélérant la mise sur le marché d'innovations tout en assurant une sécurité légale quant à leur exploitation.

De manière générale, la somme des connaissances accumulées dans l'ensemble des documents disponibles est colossale mais reste en pratique sous-exploitée. Dans le cas particulier des brevets, la taille du corpus ne permet pas à un humain d'exploiter l'information contenue sans assistance d'outils spécifiques de fouille. D'autre part, les brevets sont des documents d'une nature particulière, à mi-chemin entre la note technique (pertinent pour un métier donné) et le document juridique (pertinent pour la propriété industrielle). Il est également à noter que le corpus s'enrichit constamment, impliquant une mise à jour fréquente de l'information et de sa représentation.

Pour répondre à la question de brevetabilité d'une invention, le processus repose sur l'étude d'un corpus de brevets d'une taille de plusieurs millions de documents. Afin de cibler les brevets pertinents, plusieurs étapes de sélection sont généralement appliquées :

1. Une étude documentaire par mots clés permet de réduire drastiquement le nombre de brevets à analyser, le faisant passer de $O(10^7)$ à $O(10^3)$ brevets. Ce tri rapide et automatique doit être suffisamment large pour contenir tous les documents pertinents.
2. La lecture des $O(10^3)$ brevets restants permet d'identifier les $O(10^1)$ brevets les plus pertinents. Ce tri manuel implique des experts ayant la responsabilité critique de ne pas rater des documents pertinents.
3. L'étude des libertés d'exploitation réalisée par une analyse fine des documents pertinents.

L'objectif du stage est d'étudier, comprendre et mettre au point des outils basés sur l'Intelligence Artificielle pour accélérer les étapes 2. et 3. du processus Fig. 3.1 dans le but de venir en aide

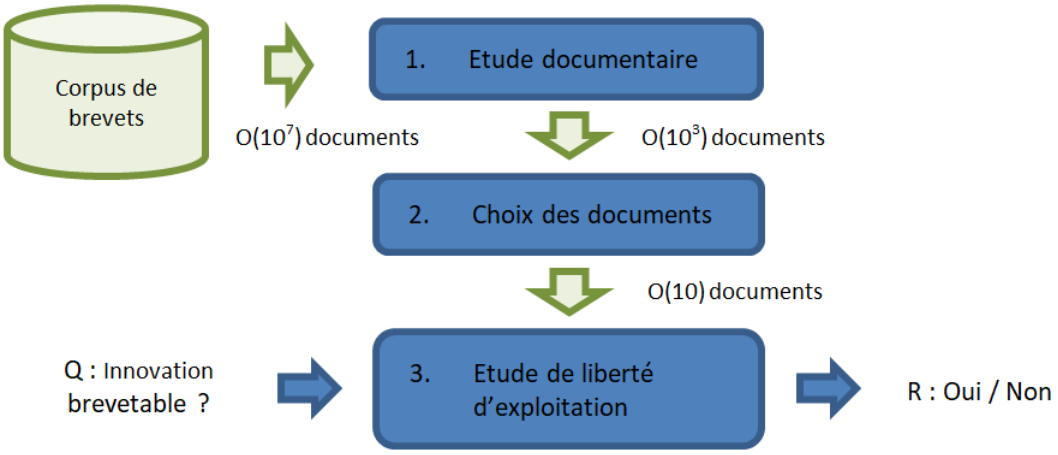


FIGURE 3.1: Processus métier de l'étude d'un corpus de brevets

aux experts brevets. Ce stage vise en particulier à étudier l'apport de méthodes de traitement automatique du langage (ou NLP, Natural Language Processing) pour aider à déterminer la brevetabilité d'une invention ou sa liberté d'exploitation.

Dans le domaine de la chimie, des travaux préliminaires conduits à IFPEN ont permis d'établir une ontologie métier pour la recherche et le développement de catalyseurs hétérogènes. Une version simplifiée est présentée Fig. 3.2. Nous définissons ici une ontologie comme une représentation de la connaissance sous la forme d'un graphe pour un domaine donné, par exemple ici la catalyse hétérogène. Ainsi, le *catalyseur* est un concept, i.e. un nœud du graphe, caractérisé par plusieurs *propriétés*, i.e. d'autres nœuds, qui sont décrites dans l'ontologie. Les différents concepts (ou nœuds) sont reliés entre eux au travers de relations nommées.

L'intérêt de cette ontologie est de formaliser une description de la connaissance métier. Ainsi, en utilisant cette description, il est possible de mettre au point des outils permettant d'extraire la connaissance de corpus de textes, des brevets dans notre cas. L'approche envisagée ici est une approche supervisée reposant sur deux tâches :

- la reconnaissance d'entités nommées (ou NER, Named Entity Recognition) consistant à extraire les concepts des textes caractérisés par les nœuds du graphe de l'ontologie ;
- l'extraction de relations (ou RE, Relation Extraction) consistant à déterminer la relation entre entités caractérisée par le lien entre nœuds du graphe de l'ontologie.

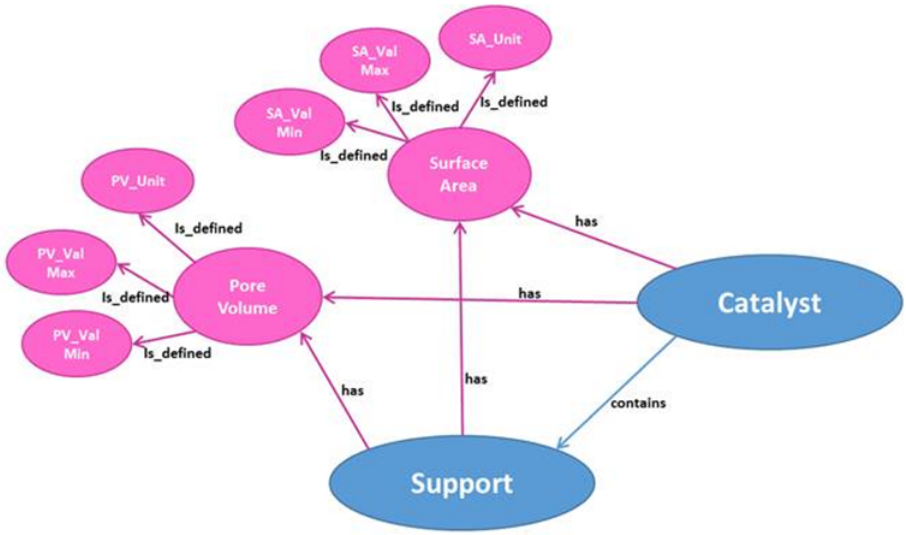


FIGURE 3.2: Ontologie simplifiée de la catalyse hétérogène

Des travaux à IFPEN ont permis de valider la pertinence de cette approche au travers d'une preuve de concept menée sur la plateforme bien connue d'Intelligence Artificielle Watson de IBM. Fort de cela, un ensemble de brevets (les revendications en particulier) ont été annotés (voir Fig. 3.3) par les experts métiers IFPEN afin de permettre la reconnaissance d'entités et leurs relations issues de l'ontologie de catalyse hétérogène construite.

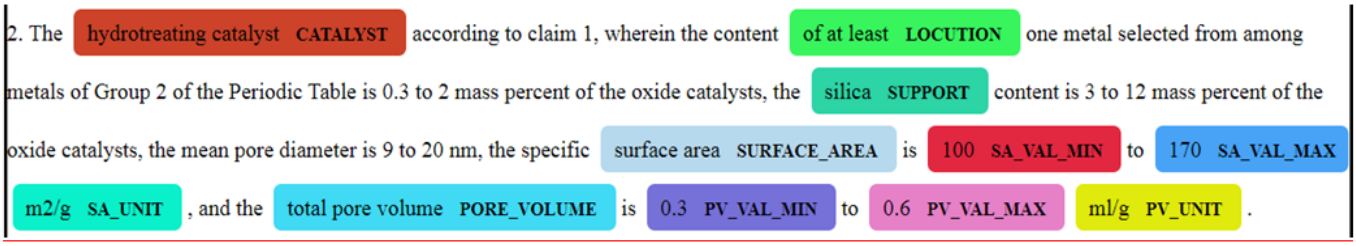


FIGURE 3.3: Exemple d'annotations d'une revendication de brevet

Toutefois, bien que extrêmement ergonomique, la plateforme Watson a les inconvénients d'être propriétaire, coûteuse et hébergée sur le cloud, rendant son utilisation difficile et contraignante pour les applications de IFPEN. Dès lors, une des ambitions de ce stage est de permettre à IFPEN d'acquérir les briques technologiques permettant de reproduire les résultats obtenus sur le corpus de brevets annotés en utilisant la plateforme Watson. Pour cela, l'idée est de se focaliser sur des méthodes d'apprentissage profond qui sont aujourd'hui l'état de l'art sur les applications de traitement du langage.

4. Étude préliminaire, la classification

Cette première partie consiste à faire un état de l'art des méthodes de NLP classiques sur un sujet de classification classique, et plus particulièrement, sur l'analyse de sentiments. Une analyse de sentiments a pour objectif de prédire si un commentaire a un sentiment positif ou négatif. Cette analyse est très utilisée par exemple en marketing pour permettre à une entreprise d'avoir un retour de ses clients sur ses produits ou services.

4.1 Le jeu de données IMDb

4.1.1 Présentation des données

Nous avons utilisé une base de données bien connue [23] provenant de IMDb Reviews. Cette base de données est composée de 50 000 commentaires de films labellisés « `0` » si le commentaire est négatif et « `1` » s'il est positif.

La base de données a été préparée au préalable et se compose de 25 000 commentaires dans l'échantillon d'entraînement et 25 000 commentaires dans l'échantillon de test. Par ailleurs, les données sont *équilibrées* car elles sont composées de 50% de commentaires positifs et 50% de commentaires négatifs (Voir Fig. 4.1).

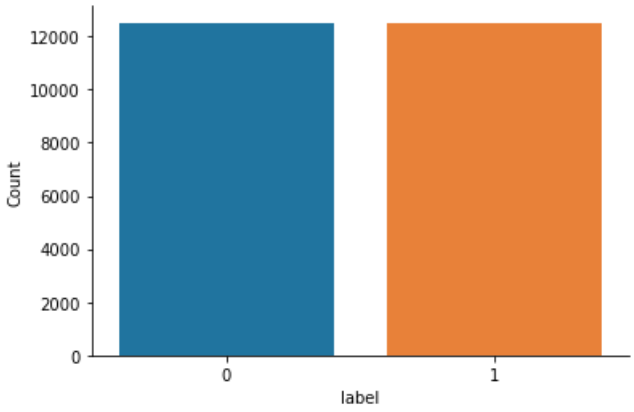


FIGURE 4.1: Distribution des labels dans le dataset IMDb

Ci-dessous, on donne un exemple de commentaire positif.

> This movie will always be a Broadway and Movie classic, as long as there are still people who sing, dance, and act.

Ainsi qu'un exemple de commentaire négatif.

> no comment - stupid movie, acting average or worse... screenplay - no sense at all... SKIP IT !

4.1.2 Pré-traitement des données

La première étape de notre analyse consiste à pré-traiter les données et plus précisément à :

- segmenter le texte en unités lexicales ou *tokens* ;
- supprimer la ponctuation, les caractères spéciaux, les nombres ;
- supprimer les mots de terminaison ou *stopwords* tels que « the », « this », etc. ;
- *lemmatiser*, c'est-à-dire, prendre la racine des mots. Par exemple, les mots « stepping », « stepped », « steps » seront transformés en « step » ;
- supprimer les mots liés aux films étant donné que ces mots ne permettent pas de définir le sentiment d'un commentaire.

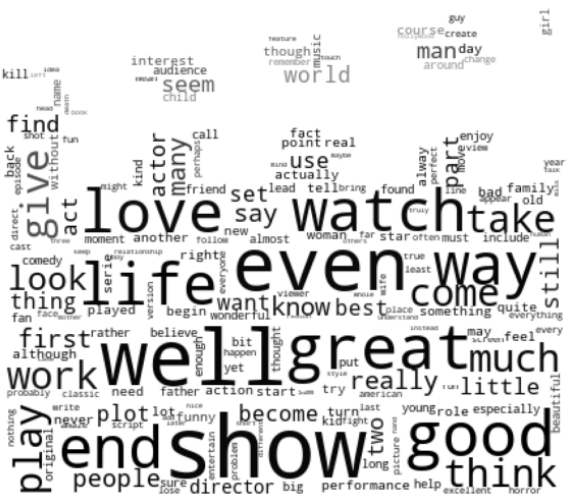
Ci-dessous, un commentaire original avant l'étape de pré-traitement :

> Swoozie Kurtz is excellent in a supporting role, but so what ?
< br />Equally annoying is this new IMDB rule of requiring ten lines for every review. When a movie is this worthless, it doesn't require ten lines of text to let other readers know that it is a waste of time and tape. Avoid this movie.

Voici le même commentaire après pré-traitement :

> swoozie kurtz excellent support role equally annoy new imdb rule require ten line every review worthless require ten line text let reader know waste tape avoid

On peut également choisir de représenter les mots les plus fréquents dans les commentaires positifs et négatifs. Les mots les plus fréquents sont de grande taille sur l'image.



(a) Commentaires positifs

(b) Commentaires négatifs

On remarque que les mots qui ressortent des commentaires positifs sont plutôt positifs et que ceux des commentaires négatifs sont négatifs ou neutres. Par exemple, « show », « watch »sont des mots sans connotations particulières qui sont fréquents à la fois dans les commentaires positifs et négatifs.

4.2 Méthodes de classification

Une fois nos données traitées, nous avons utilisé d'une part des méthodes de Machine Learning puis d'autre part, des méthodes de Deep Learning afin de réaliser notre analyse de sentiments.

4.2.1 Approches statistiques

Avant l'application d'un quelconque algorithme, nous devons préalablement convertir nos données textuelles en valeurs numériques. Pour ce faire, nous avons tout d'abord utilisé les représentations **Bag-of-Word** et **TF-IDF** qui se basent respectivement sur l'occurrence et sur la fréquence des mots dans un corpus.

Après avoir obtenu une représentation numérique de nos données, nous avons considéré deux algorithmes classique de Machine Learning : la **Régression Logistique** et le **Support Vector Machine** qui sont généralement utilisés pour l'analyse de sentiments.

4.2.1.1 Représentation numérique

Une représentation *Bag-of-Word* est une représentation dans laquelle chaque commentaire est représenté par un vecteur de la taille du vocabulaire. On utilise ainsi la matrice composée de l'ensemble de ces commentaires qui forment le corpus comme entrée de nos algorithmes. Ci-dessous est représenté la représentation *Bag-of-Word* pour un corpus constitué de trois commentaires.

TABLE 4.1: Représentation *Bag-of-Word* pour des exemples de commentaires

	this	movie	film	is	great	bad	very	good
this movie is great this is good	2	1	0	2	1	0	0	1
this movie is bad	1	1	0	1	0	1	0	0
this film is very good	1	0	1	1	0	0	1	1

Dans la méthode *TF-IDF* (Term Frequency - Inverse Document Frequency), on ne considère pas le poids d'un mot dans un commentaire comme sa fréquence d'apparition (partie *TF*) uniquement, mais on pondère cette fréquence par un indicateur si ce mot est commun ou rare dans tous les commentaires. En l'occurrence, il utilise comme indicateur l'« Inverse Document Frequency » (partie *IDF*) qui est l'inverse de la proportion de commentaire qui contient le terme à l'échelle logarithmique.

On peut définir les deux mesures suivantes pour un terme t :

$$\mathrm{tf}(t) = \frac{\text{Nombre d'occurrence du terme analysé}}{\text{Nombre de termes total}}$$

et

$$\mathrm{idf}(t) = \log\left(\frac{\text{Nombre total de commentaires}}{\text{Nombre de commentaires contenant le terme analysé}}\right).$$

Ainsi, le terme de pondération global peut s'écrire comme ceci :

$$\mathrm{tfidf}(t) = \mathrm{tf}(t) * \mathrm{idf}(t).$$

La pertinence lexicale se mesure donc avec l'indicateur *TF-IDF* grâce à une relation entre la rareté d'un mot au sein d'un ensemble de commentaires, mais également avec son occurrence dans un seul commentaire.

Ci-dessous sont représentés les indicateurs *TF-IDF* des trois commentaires utilisés comme exemple précédemment dans le tableau 4.2.

On remarque que les mots présents dans tous les commentaires tels que « this » ou « is », ne sont pas importants pour déterminer le sentiment d'un commentaire et leurs indicateurs *TF-IDF* valent zéro.

Les méthodes *Bag-of-Word* et *TF-IDF* permettent ainsi de représenter les données textuelles en valeurs numériques en appliquant une approche statistique sur les mots dans le corpus.

TABLE 4.2: Représentation *TF-IDF* pour des exemples de commentaires

	this	movie	film	is	great	bad	very	good
this movie is great this is good	0	0.03	0	0	0.07	0	0	0.03
this movie is bad	0	0.04	0	0	0	0.12	0	0
this film is very good	0	0	0.10	0	0	0	0.10	0.04

Néanmoins, l'un des principaux inconvénients de ces représentations est qu'elles considèrent les mots d'un commentaire comme étant indépendants les uns des autres et ne prennent donc pas en compte le contexte ou les éléments sémantiques. Par exemple, ces méthodes ne permettent pas de détecter l'ironie ou le sarcasme dans un commentaire.

4.2.1.2 Régression logistique

La régression logistique est un modèle de classification établissant une relation linéaire entre les observations et prédictions du modèle. Ici, on considère que les prédictions ont des valeurs possibles binaires (0 ou 1). Comme le modèle est linéaire, la fonction hypothèse peut s'écrire comme suit :

$$s(x^{(i)}) = w_0 + w_1\, x_1 + w_2\, x_2 + w_3\, x_3 + \cdots + w_n\, x_n$$

avec :

- $x^{(i)}$: une observation, cette variable est un vecteur $[\, x_1,\, x_2,\, \ldots,\, x_n \,]^T$ et dans notre cas, il s'agit du vecteur représentatif du mot i ;
- w_i : est un poids (ou paramètre) de la fonction hypothèse. On cherche à déterminer l'ensemble des w_i pour définir la fonction de prédiction ;
- Le paramètre w_0 est nommé le biais.

On note dans la suite w le vecteur des paramètres $\Big[\, w_0,\, w_1,\, ...,\, w_n \,]^T$.
Le but de la mise au point du modèle est de trouver les coefficients de w de sorte que :

- $s(x^{(i)}) > 0$ quand la classe vaut 1, ce qui correspond au sentiment positif ;
- $s(x^{(i)}) < 0$ quand la classe vaut 0, ce qui correspond au sentiment négatif.

La fonction s ainsi obtenue combine linéairement les différentes variables prédictives (i.e. les termes x_i). A cela, on appliquera ensuite la fonction sigmoid σ :

$$\sigma(x) = \frac{1}{1 + e^x}.$$

En effet, le résultat obtenu par la fonction sigmoid peut être interprété comme la probabilité qu'une observation $x^{(i)}$ soit d'un label 1. En appliquant cette fonction sigmoid sur notre fonction S, on obtient notre fonction hypothèse pour la régression logistique :

$$h(x^{(i)}) = \sigma(s(x^{(i)})) = \frac{1}{1 + e^{s(x^{(i)})}} = P(y = 1 | x^{(i)}; w)$$

On trouve le vecteur w en minimisant le maximum de vraisemblance l suivant :

$$l(w) = -\sum_{i=0}^{n} h(x^{(i)})^{y^{(i)}} \, (1 - h(x^{(i)}))^{(1-y^{(i)})}$$

où $y^{(i)}$ correspond au label du $i^{ème}$ commentaire.

4.2.1.3 Support Vector Machine (SVM)

Le SVM appartient à la catégorie des classificateurs linéaires binaires qui utilisent une séparation linéaire des données, et qui cherche à trouver un hyperplan séparant les deux catégories (positif et négatif) de notre problème. Pour trouver l'hyperplan qui convient le mieux, nous devons trouver un hyperplan de marge maximale. La marge d'un hyperplan séparateur se définit comme étant deux fois la distance de l'hyperplan au point du jeu de données qui en est le plus proche.
On rappelle que l'équation de notre hyperplan séparateur de marge maximale h est

$$h_w(x) = s(x) = \sum_k w_k \, x_k + w_0 = 0$$

où $w = \big[\, w_0,\, w_1,\, ...,\, w_n \,\big]^T$ forme le vecteur des paramètres qui satisfait le problème d'optimisation suivant :

$$\underset{w}{\operatorname{argmin}} \; \frac{||w||^2}{2}$$

avec la contrainte $y^{(i)} \, h_w(x^{(i)}) \geqslant 1$.
On peut ainsi trouver la solution optimale $\underline{w}$ et utiliser l'hyperplan séparateur comme fonction de décision :

$$f(x) = h_{\underline{w}}(x).$$

Ainsi, si le résultat de cette fonction est positive, le point est étiqueté positif, et inversement, s'il est négatif, le point est étiqueté négatif.
Dans le cas où les données ne sont pas linéairement séparables, il est possible d'utiliser une version améliorée basée sur une projection des données x dans un espace dit *de redescription* par l'application d'une transformation non linéaire Φ. Ainsi, plutôt que d'apprendre un modèle qui explique y comme une fonction linéaire des coordonnées de x, nous pouvons utiliser le SVM linéaire pour apprendre un modèle qui explique y comme une fonction linéaire des coordonnées de $\Phi(x)$. Nous créons ainsi un modèle non-linéaire dans l'espace initial.
Dès lors, l'équation de l'hyperplan est :

$$h_w(x) = \sum_k w_k \, \Phi(x) + w_0$$

avec la contrainte $y \, h_w(x) \geqslant 1$.
En pratique, la résolution performante d'une telle formulation nécessitera l'introduction d'une astuce basée sur une fonction noyau K pour remplacer le produit scalaire :

$$K(x^{(i)}, x^{(j)}) = \Phi(x^{(i)}) \, \Phi(x^{(j)}).$$

4.2.2 Approches par apprentissage profond

Nous décrivons maintenant des méthodes de Deep Learning utilisées pour notre analyse de sentiments.

Dans la partie précédente, nous avions considéré que les mots d'un commentaire étaient indépendant entre eux d'un point de vue sémantique et nous n'avions pas pris en compte le contexte d'un commentaire pour déterminer son sentiment. La représentation numérique des mots par la technique de *word embedding* ainsi que certaines méthodes de Deep Learning permettent quant à elles de prendre en compte ce contexte.

4.2.2.1 Word Embedding

Dans la partie précédente, nous avons vu des manières de représenter les commentaires en valeurs numériques, en utilisant les méthodes de représentations *bag-of-words* et *TF-IDF* basées sur les statistiques d'occurrence des mots dans le corpus.

D'autres techniques existent pour obtenir une représentation des mots dans un espace avec une forme de similarité entre eux, dans lesquelles le sens des mots les rapproche dans cet espace, en terme de distances statistiques. Il s'agit de méthodes de plongement (ou *embedding*) dans un espace de dimension finie, typiquement autour de 50 à 300 dimensions.

Pour notre étude, nous avons utilisé la méthode *GloVe* [4]. GloVe est un algorithme d'apprentissage non supervisé qui permet d'obtenir des représentations vectorielles des mots. L'entraînement a été réalisé sur un corpus de texte de 6 milliards de mots, provenant de `Wikipédia`.

L'hypothèse principale de cette technique étant de prendre en compte le *contexte* dans lequel le mot a été trouvé, c'est-à-dire les mots avec lesquels il est souvent utilisé. On peut retrouver beaucoup de régularités linguistiques simplement en effectuant des translation linéaires dans cet espace de représentation, comme illustré ci-dessous.

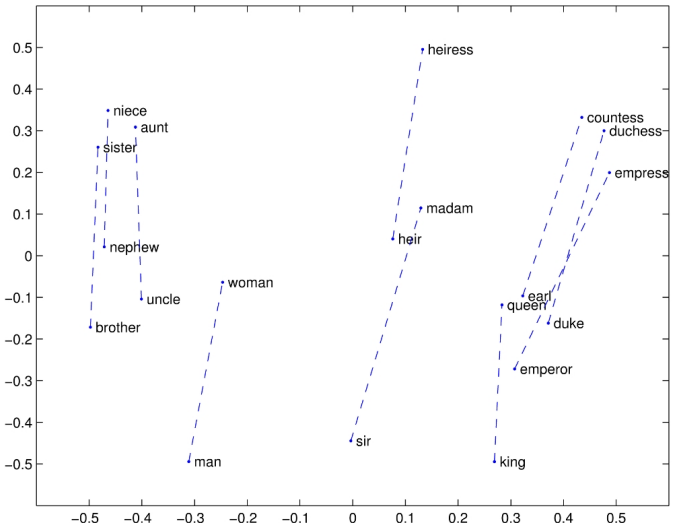


FIGURE 4.3: Word embedding GloVe

On observe dans Fig. 4.3 que les mots `sister`, `uncle`, `brother` appartiennent au lexique de la famille et sont proches dans l'espace vectoriel représenté. De plus, les différences de vecteur tels que `man` - `woman`, `king` - `queen` et encore `brother` - `sister` sont à peu près égales.

4.2.2.2 Réseaux de neurones récurrents

Le premier modèle de Deep Learning utilisé est le réseau de neurones récurrents (ou RNN, Recurrent Neural Network). Ce type de réseau de neurones permet de prendre en compte le contexte d'une phrase, et contrairement aux méthodes présentées jusqu'ici, considère que les mots d'un commentaires ne sont pas indépendants entre eux. C'est une architecture couramment employée pour le traitement du langage [6].

Un réseau de neurones récurrent est typiquement organisé en couches successives dont les cellules (ou neurones) sont connectés aux couches suivantes et contiennent des valeurs variant en fonction du temps. Concrètement, le modèle basique RNN prend une séquence de mots $X = \{x_1, \ldots, x_T\}$, un à la fois, et produit un état caché h_t, pour chaque mot. On utilise ce modèle de manière itérative en lui donnant le mot courant x_t ainsi que l'état caché du mot précédent, h_{t-1}, pour produire l'état caché suivant, h_t. On répète le processus pour tous les mots d'un commentaire. Dans ce modèle, l'architecture a une structure très simple, composé d'une seule couche d'un neurone intégrant une fonction d'activation de tanh. On peut alors écrire l'expression suivante :

$$h_t = \tanh(W * [h_{t-1}, x_t]),$$

où on note W la matrice de poids associée à la couche.

Une fois que l'on a obtenu l'état caché final h_T, c'est-à-dire après avoir fourni le dernier mot de la séquence x_T au modèle, on applique une couche f d'un réseau de neurones artificiel de

classification (que l'on nomme généralement FC, *fully connected layer*), pour finalement obtenir la prédiction du sentiment, $\hat{y} = f(h_T)$.

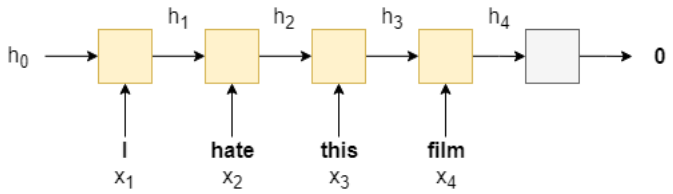


FIGURE 4.4: Application du RNN simple à un commentaire

Fig. 4.4 présente un exemple de traitement d'une phrase par le modèle RNN prédisant la valeur `0`, c'est-à-dire que le sentiment est négatif. Le modèle RNN est représenté en orange et la couche FC est en gris. Il convient de noter que c'est la même cellule (ou neurone) qui est utilisée à chaque mot, c'est-à-dire partageant les paramètres durant les itérations. L'état initial caché h_0, est initialisé à zéro.

Bien que très efficace pour modéliser les dépendances à court ou moyen terme, les modèles RNN restent toujours insuffisants pour modéliser des dépendances à long ou très long terme. En effet, dans la pratique, des problèmes d'*annulation de gradient* (ou *vanishing gradient*) et d'*explosion de gradient* (ou *exploding gradient*) pendant la rétro-propagation font qu'il est difficile d'apprendre suffisamment d'informations des étapes précédentes afin de capturer des dépendances à long terme. Le modèle de cellule LSTM (pour *Long Short-Term Memory*) a été proposé [25] pour répondre à ces difficultés. Ce modèle introduit des mécanismes régularisants au sein d'une cellule (voir Fig. 4.5) différente du modèle RNN se basant sur des filtres du flux d'information appelés *portes* (ou *gates*) et d'un état interne supplémentaire de cellule C_t appelé la *mémoire*. En particulier, les gradients peuvent rester inchangé durant le processus itératif évitant ainsi les problèmes d'annulation et explosion.

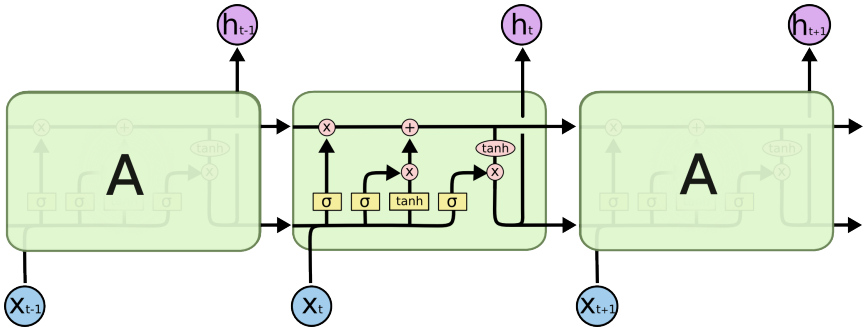


FIGURE 4.5: Architecture du LSTM

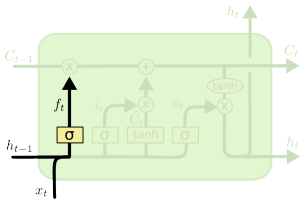
Comme pour le modèle RNN basique présenté précédemment, les états de la cellule parcourent toute une chaîne itérative. Durant le processus, la cellule LSTM a la capacité de supprimer

ou d'ajouter des informations à l'état courant de la cellule par un filtrage sélectif au moyen de *portes* (ou *gates*). Ces portes forment donc un moyen de laisser éventuellement passer des informations provenant des entrées. Le fonctionnement des portes est défini par des matrices de poids déterminées durant l'apprentissage.

La cellule LSTM définit 3 portes pour :

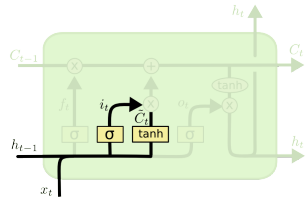
- Fig. 4.6 : Contrôle de l'oubli (**Forget gate**) de la valeur dans la cellule ;
- Fig. 4.7 : Contrôle de l'entrée (**Input gate**) d'une nouvelle valeur dans la cellule ;
- Fig. 4.8 : Contrôle de l'utilisation de la valeur dans la sortie (**Output gate**) de la cellule.

Notons que les fonctions d'activation des portes sont souvent des fonctions sigmoïdes ou `tanh`.



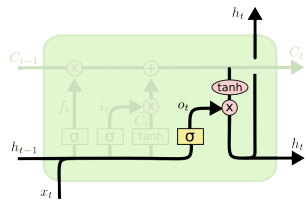
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

FIGURE 4.6: LSTM : porte *Forget*



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

FIGURE 4.7: LSTM : porte *Input*



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

FIGURE 4.8: LSTM : porte *Output*

Enfin, la mémoire C_t de la cellule est mise à jour (voir Fig. 4.9) en tenant de la mémoire de l'itération précédente et des valeurs transitant par les portes d'entrée et d'oubli.

Notons qu'il existe de nombreuses variantes au modèle LSTM telle que le modèle GRU [3].

Nous avons effectué diverses variations aux modèles présentés ci-dessus dont :

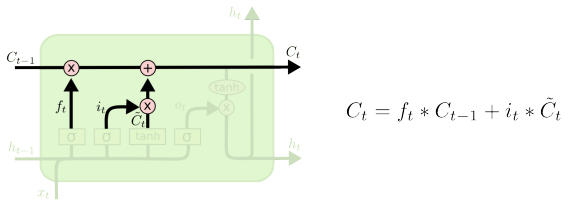
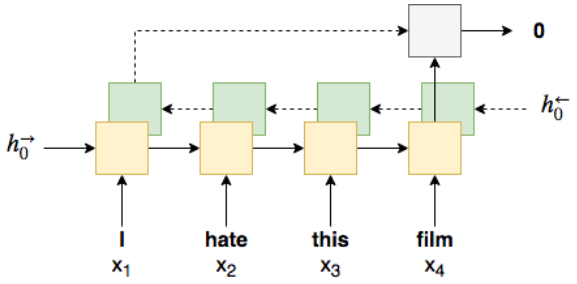
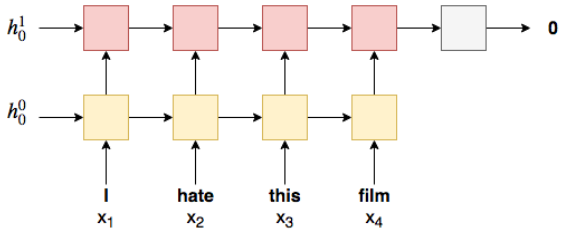


FIGURE 4.9: LSTM : mise à jour de l'état de la cellule

- Un modèle **bidirectionnel** [9] (ou BiLSTM, voir Fig. 4.10a) : en plus d'avoir un modèle RNN traitant les mots du commentaire du premier au dernier (RNN *forward*), on introduit un second modèle RNN traitant les mots du dernier au premier (RNN *backward*). Au pas de temps t, le RNN *forward* traite le mot x_t, et le RNN *backward* traite le mot x_{T-t+1}. La prédiction du sentiment est réalisée en utilisant une concaténation du dernier état caché du RNN *forward* (obtenu à partir du dernier mot de la phrase), $h_T^{\rightarrow}$, et le dernier état caché du RNN *backward* (obtenu à partir du premier mot de la phrase), $h_T^{\leftarrow}$ on le donne à une couche linéaire f, pour recevoir notre sentiment prédit, $\hat{y} = f(h_T^{\rightarrow}, h_T^{\leftarrow})$;
- Un modèle **multi-couches** [1] (voir Fig. 4.10b) : c'est une architecture de plusieurs modèles RNN superposés les uns aux autres et interconnectés par couche. La sortie d'une couche au pas de temps t sera l'entrée de la couche suivante. La prédiction est alors faite à partir de l'état caché final de la couche finale, c'est-à-dire la plus haute. Fig. (4.10b) décrit un modèle RNN unidirectionnel à deux couches.



(a) RNN bidirectionnel



(b) RNN multi-couches

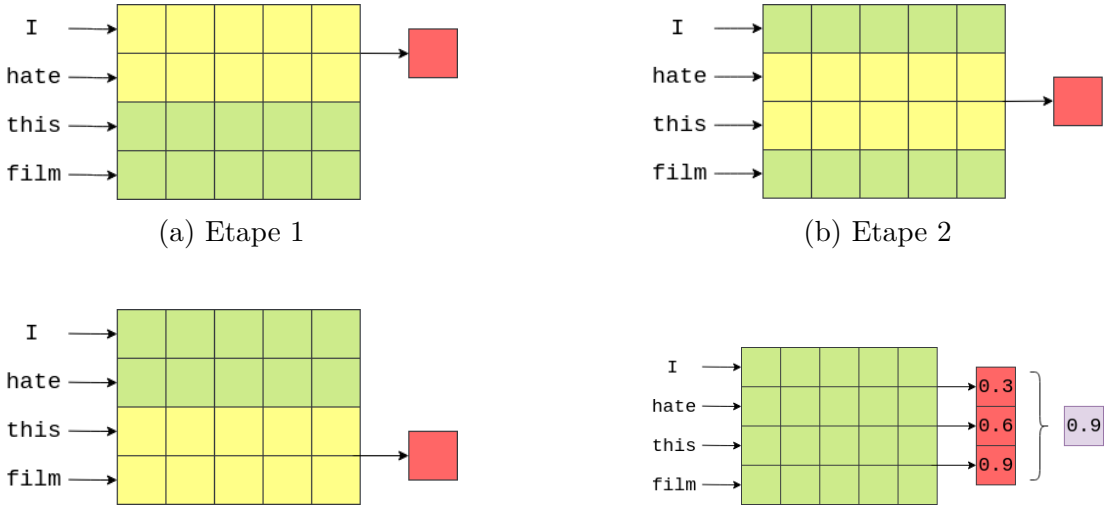
FIGURE 4.10: RNN bidirectionnel et multi-couches

4.2.2.3 Réseau de neurones convolutifs

Afin de s'affranchir du caractère récurrent de notre modèle, on peut utiliser les réseaux convolutifs (ou CNN, Convolutional Neural Network). Ce type de réseau de neurones est généralement utilisé pour analyser les images [2] et est constitué d'une ou plusieurs couches convolutives, suivies d'une ou plusieurs couches FC (*fully connected*). Ces couches convolutives utilisent des filtres qui sont convolués sur l'image et produisent une version traitée de l'image. Cette version

traitée de l'image peut alors être introduite dans une autre couche convolutive ou une couche FC. Chaque filtre a une forme, par exemple un filtre 3×3 couvre une zone de 3 pixels de large et 3 pixels de haut de l'image, et chaque élément du filtre a un poids qui lui est associé, ainsi le filtre 3×3 aurait 9 poids. Les couches convolutives agissent alors comme des extracteurs de caractéristiques, extrayant les parties de l'image les plus importantes.

Les réseaux de neurones convolutifs ont été introduit plus récemment pour le traitement du langage [8]. En effet, en convertissant les mots d'un commentaire en leur représentation numérique (en utilisant typiquement une méthode de *word embedding*), nous pouvons représenter la séquence de texte sous une forme matricielle $n_w \times N$ où n_w est le nombre de mots du commentaire et N la dimension de l'espace choisi pour représenter numériquement les données. De la même manière qu'un filtre 3×3 peut être appliqué sur une zone d'une image, un filtre $2 \times N$ peut être appliqué sur 2 mots dans la séquence (voir Fig. 4.11). Dans ce cas, il y a donc intuitivement une analogie possible avec le traitement de la séquence par bi-gramme.



(a) Etape 1 (b) Etape 2

(c) Etape 3 (d) Etape 4 - Pooling

FIGURE 4.11: Étapes d'un réseau de neurones convolutifs pour un commentaire

De manière générale, on peut utiliser un filtre de taille $n \times N$ où n est un entier correspondant au nombre de mots séquentiels à considérer pour traiter de n-grammes. Dans Fig. 4.11a, un filtre de dimension 2×5 qui couvre deux mots à la fois (c'est-à-dire des bi-grammes) est représenté en jaune. Chaque élément du filtre aura un poids associé. La sortie de ce filtre (représentée en rouge) sera un seul nombre réel qui est la somme pondérée de tous les éléments couverts par le filtre (Voir Fig. 4.11a). Le filtre est appliqué le long de la phrase pour couvrir les bi-grammes suivants (Voir Fig. 4.11b et 4.11c) et la sortie finale est obtenue (Voir Fig. 4.11d). Chacun des filtres $2 \times N$ recherchera ainsi l'occurrence de différents bi-grammes.

Dans notre modèle, nous avons également utilisé des filtres de tailles 3, 4 et 5 afin de rechercher l'apparition de différents tri-grammes, 4-grammes et 5-grammes qui sont pertinents pour

analyser le sentiment des critiques de films.

La dernière étape de notre modèle (Voir Fig. 4.11d) consiste à prendre la valeur maximale des différentes sorties en utilisant une couche de *max pooling* pour enfin appliquer une couche FC pour obtenir la prédiction du sentiment.

4.2.2.4 Mécanisme d'attention

Les réseaux convolutifs permettent donc de comprendre le contexte d'une séquence et ce, sans posséder de caractère récurrent. En revanche, ce type de réseau est très positionnel, c'est-à-dire que son principe dépend beaucoup de la position des mots dans la phrase et de leur position les uns par rapport aux autres pour créer un contexte, plus que de leur similarité de contexte sémantique.

Les mécanismes d'attention [11], et plus particulièrement les mécanismes d'*auto-attention*, permettent de déterminer l'importance des mots de la séquence d'entrée pour la classification en considérant la similarité entre ces mots. Ces mécanismes ont l'avantage de pouvoir faire des liens entre des éléments très distants d'une séquence. Étant donné que tous les mots d'un commentaire ne contribuent pas de manière égale à la représentation du commentaire, le mécanisme d'auto-attention est utilisé pour extraire les mots importants en leur donnant un poids plus élevé.

Nous avons implémenté un type de modèle d'attention associé à un modèle *BiLSTM* [21]. Fig. 4.12 illustre l'architecture de ce modèle. En particulier, celui-ci est constitué d'une couche *embedding GloVe*, d'une couche *BiLSTM*, puis d'une couche *Attention* et enfin de deux couches *dense* (ou fully connected).

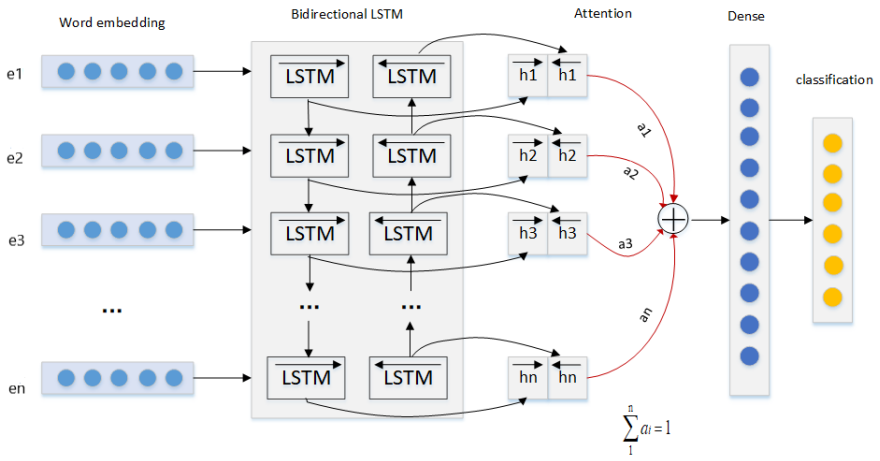


FIGURE 4.12: Architecture du modèle BiLSTM-Attention

Dans ce modèle, l'attention consiste à prendre en compte un vecteur de contexte spécifique qui est une somme pondérée de tous les vecteurs d'état émis par la partie *BiLSTM*. On note $\overrightarrow{h_i}$ le

vecteur d'état associé au mot i pour le modèle *LSTM forward* et $\overleftarrow{h_i}$ l'équivalent pour le modèle *backward*. Ainsi, l'attention est calculée de la manière suivante :

$$m_i = \tanh(W * [\overrightarrow{h_i}, \overleftarrow{h_i}] + b)$$

$$a_i = \frac{m_i}{\sum_k^n m_k}$$

où i représente le $i$ème mot du commentaire, W est la matrice des poids et b est un vecteur de biais.

Finalement, le résultat de la couche d'attention est introduit dans deux couches denses pour la prédiction du sentiment associé au commentaire.

4.2.2.5 Modèle Transformer

Le modèle le plus connu utilisant les mécanismes d'*attention* est le modèle *Transformer* [13] introduit par `Google` en 2017. Ce modèle comporte une architecture assez complexe utilisant certaines méthodes décrites précédemment comme les *word embedding* (partie 4.2.2.1), les mécanismes d'*attention* (partie 4.2.2.4) et également les *réseaux récurrents* (partie 5.3.1). L'architecture de ce modèle est illustrée dans la figure suivante.

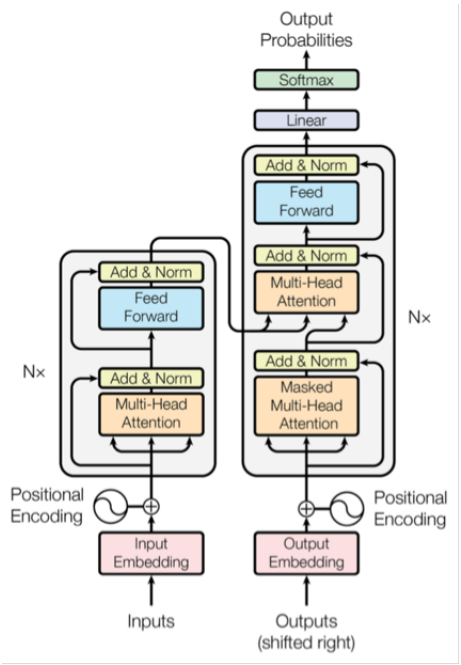


FIGURE 4.13: Architecture du modèle *Transformer*

On peut remarquer dans Fig. 4.13 que ce modèle est composé d'un encodeur, partie gauche de l'architecture, connecté à un décodeur, partie droite de l'architecture. Cette architecture est

initialement définie pour les tâches de traduction de séquences. Afin que le modèle *Transformer* puisse être adapté à la classification, nous avons réalisé certaines modifications à l'architecture initiale. En particulier, seul l'encoder est utilisé, puis on applique une couche FC pour obtenir le label associé à la séquence.

Chaque encodeur se compose de deux sous-couches :

- une couche d'*auto-attention multi-têtes*
- suivie d'une couche résiduelle notée *Feed Forward Network* (ou FFN).

Nous donnons dans la suite les grandes lignes du modèle.

Embeddings De manière usuelle, chaque mot est d'abord représenté numériquement par un *word embedding*. Associé à cet *embedding*, un *positional encoding* est ajouté, ce qui détermine une façon d'encoder la place de chaque élément dans la séquence. Comme la longueur des phrases n'est pas prédéterminée, des fonctions sinusoïdales sont utilisées (explicitées ci-dessous) donnant de petites valeurs entre 0 et 1, pour modifier légèrement les représentations numériques (ou *embeddings*) de chaque mot.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

où *pos* est la position du mot dans la phrase et i la dimension dans le vecteur *embedding* pour ce mot.

Ensuite, le modèle transformer est généralement composé de six encodeurs empilés (N_x dans Fig. 4.13). Chaque encodeur prenant en entrée la sortie de l'encodeur précédent (sauf le premier qui prend en entrée les *embeddings*). Ces blocs ne partagent par ailleurs pas les mêmes matrices de poids.

Attention multi-têtes Le mécanisme d'attention est différent de celui utilisé précédemment, bien plus raffiné et complexe. Il est de type *clé-valeur*. Décrit comme la somme de ses *embeddings* sémantiques et positionnels, chaque mot est décomposé en trois abstractions :

- Q : Une requête (*query*)
- K : Une clé (*key*)
- V : Une valeur (*value*)

Chacune de ces abstractions est un vecteur qui est obtenu lors du processus d'apprentissage grâce à une matrice de poids. Chaque matrice est distincte, une pour la requête W_q, une pour la clé W_k, une pour la valeur W_v. Le principe de ce mécanisme d'attention est illustré ci-dessous.

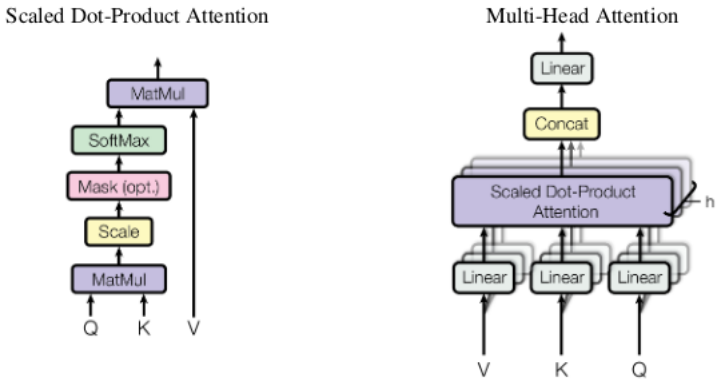


FIGURE 4.14: Attention

Chaque tête de l'attention possède ses propres matrices $W_{q,i}, W_{k,i}, W_{v,i}$ pour la tête i. L'attention pour une tête est ensuite calculée avec la formule suivante :

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V$$

où d_k est la dimension pour chaque tête. La sortie de chaque tête est concaténée puis le tout multiplié par une matrice W_0 :

$$\text{MultiHead}(Q, V, K) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_0$$

où

$$\text{head}_i = \text{Attention}(QW_{q,i}, KW_{k,i}, VW_{v,i})$$

Couche Feed Forward Network En plus des sous-couches d'attention, chaque couche dans l'encoder contient une couche *fully connected* ou feed forward network qui est appliquée à chaque position. Il s'agit d'un réseau de neurones standard à deux couches. Ainsi, cela consiste en deux transformations linéaires avec une couche d'activation $ReLU$ [10] entre elles :

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

où W_1 et W_2 sont les matrices de poids et b_1 et b_2 les vecteurs de biais.

Couche résiduelle Chaque sous-couche possède en sortie une couche `Add & Norm` qui ajoute les sorties de la couche et du raccourci à une connexion dite *résiduelle*, c'est-à-dire qui connecte directement les valeurs d'entrée de la couche à la sortie de la couche. L'ensemble est finalement normalisé puis une dernière couche *sigmoid* est appliqué afin de retourner la prédiction.

4.2.2.6 Modèle BERT

Le modèle *BERT* (*Bidirectional Encoder Representations from Transformers*) [20] est un raffinement du modèle *Transfomer* publié en 2018. Il s'agit d'un modèle de représentation du langage qui se compose d'une suite *Transformer* de $N_x = 12$ pour la version dite *Base* ou $N_x = 24$ pour la version *Large*. C'est un modèle extrèmement performant qui a initié de nombreux travaux amenant à un saut de performance des modèles d'apprentissage profond pour le langage.

Le modèle *BERT* s'inscrit par ailleurs dans le paradigme de l'apprentissage par transfert (ou en anglais *tranfer learning*) qui suppose de procéder en deux temps : d'abord un *pré-entraînement* de l'algorithme est effectué en apprenant à représenter le texte à l'aide une tâche non-supervisée pour ensuite être raffiné sur une tâche précise de manière supervisée.

L'approche non-supervisée est constituée de 2 étapes. D'une part, une partie des mots des séquences d'entrée est masquée et le modèle apprend à les retrouver. Ceci permet d'acquérir une connaissance générale et bi-directionnelle du langage tout en pouvant utilisé n'importe quel corpus de textes. D'autre part, le modèle apprend à reconnaître si deux phrases sont consécutives ou non.

La force du modèle *BERT* est la capacité de généralisation à des tâches spécifiques une fois le modèle pré-entraîné, par exemple pour la traduction, la réponse à des questions, etc. Dans notre cas, l'algorithme est spécialisé pour la classification sur le corpus de commentaires. Cette étape est dite de *fine-tuning*. En particulier, nous réutilisons le pré-entrainement du modèle sur un corpus multilingue extrêmement volumineux dont l'apprentissage non-supervisé est hors du cadre de ce stage.

4.3 Évaluation des modèles

Dans cette section, les différents modèles présentés précédemment sont comparés. Les implémentations ont été effectuées en `python` sur un serveur spécifiquement mis en place pour les activités en science des données à IFPEN. Le framework d'apprentissage `PyTorch` a été utilisé dans un environnement `Jupyter` notebooks.

4.3.1 Entraînement des modèles

Concernant l'entraînement des modèles de machine learning présentés dans la partie 4.2.1, nous avons trouvé les meilleurs hyperparmètres du modèle qui maximisent l'accuracy. Puis nous avons entraîné le modèle avec ces hyperparamètres en utilisant la librairie `sklearn`.

L'entraînement des modèles d'apprentissage profond s'est réalisé en plusieurs étapes.

La première étape a consisté à traiter les données en utilisant la librairie `TorchText` afin d'obtenir des données segmentées en *tokens*. Ici, un token correspond à un mot. Cette librairie

permet également d'obtenir le vocabulaire provenant de l'échantillon d'entraînement ainsi que de représenter les données textuelles en données numériques à l'aide de la méthode *GloVe*.

Ensuite, les données ont été assemblées en *batch* ou lot de taille donnée. Afin que tous les commentaires soient de la même longueur l, les commentaires de taille supérieure à l sont tronqués, et pour les commentaires de taille inférieure à l, des zéros sont ajoutés. L'étape d'ajout de zéros est appelée *padding*.

De plus, nous avons utilisé un optimiseur *Adam* [16] qui est un algorithme permettant de mettre à jour les poids du réseau neuronal afin de réduire les pertes. Les pertes sont calculées à l'aide d'une fonction *entropie croisée* [29]. Nous avons entraîné les modèles en un certain nombre d'*epoch* jusqu'à convergence de la courbe représentant les pertes calculées.

4.3.2 Métrique

Afin de pouvoir déterminer quels algorithmes prédisent le mieux le caractère positif ou négatif des commentaires, nous utilisons ici l'*accuracy* comme métrique discriminante. En effet, nos classes sont équilibrées dans l'échantillon d'entraînement.

L'*accuracy* se calcule comme suit :

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

où on a noté :

- TP : vrais positifs, c'est-à-dire le nombre de commentaires labellisés «`1`» et correctement classifiés ;
- TN : vrais négatifs, c'est-à-dire le nombre de commentaires labellisés «`0`» et correctement classifiés ;
- FP : faux positifs, c'est-à-dire le nombre de commentaires labellisés «`0`» et classifiés «`1`» ;
- FN : faux négatifs, c'est-à-dire le nombre de commentaires labellisés «`1`» et classifiés «`0`».

4.3.3 Résultats

Le tableau 4.3 présente les résultats des différents modèles considérés.

TABLE 4.3: Résultats pour la classification de sentiment

Modèle	**Section**	**Accuracy**
SVM+BoW	4.2.1.3	0.87
RL+BoW	4.2.1.2	0.87
SVM+TF-IDF	4.2.1.3	0.88
RL+TF-IDF	4.2.1.2	0.88
BiLSTM	5.3.1	0.89
CNN	4.2.2.3	0.85
BiLSTM+Attention	4.2.2.4	0.83
Transformers	4.2.2.5	0.73
BERT	4.2.2.6	**0.92**

Globalement, les résultats sont très bons pour tous les modèles. Les modèles statistiques classiques présentés en partie 4.2.1 donnent de très bons résultats et ce n'est pas une surprise. Les modèles d'apprentissage profond présentés en partie 4.2.2 sont également bons avec des résultats en retrait pour certains et très bons pour d'autres. Nous verrons toutefois la performance de ces modèles sur des tâches plus complexes dans les sections suivantes.

On remarque que le meilleur modèle est le modèle *BERT*. Néanmoins, son résultat est assez proche des méthodes classiques de Machine Learning alors que le temps d'entraînement de *BERT* est beaucoup plus grand que celui des autres méthodes. Par ailleurs, le modèle *Transformers* donne le moins bon résultat, ce qui est assez surprenant, étant donné que ce modèle est récent ayant architecture assez complexe donnant généralement de très bons résultats sur des corpus volumineux. La taille limitée du jeu de données est certainement un point qui peut expliquer ces résultats.

5. Reconnaissance d'entités nommées

Dans cette partie, nous réalisons une tâche de reconnaissance d'entités nommées (ou NER, Named Entity Recognition) sur d'une part le jeu de données CoNLL-2003 fréquemment utilisé pour évaluer des modèles sur cette tâche, et d'autre part sur des données de brevets annotées par les experts métiers IFPEN.

La reconnaissance d'entités nommées est une sous-tâche d'extraction d'information qui consiste à reconnaître des entités nommées (Named Entities) dans un corpus et de leur attribuer une étiquette. C'est donc en quelque sorte une tâche de classification mot à mot.

5.1 Le jeu de données CoNLL-2003

Nous avons utilisé la base de données bien connue CoNLL-2003 [18]. Cette base de données est composée de phrases annotées. Les entités sont :

- *persons* (`PER`) désignant des personnes,
- *organizations* (`ORG`) désignant des organisations et entreprises,
- *locations* (`LOC`) désignant des lieux,
- *miscellaneous names* (`MISC`) désignant des noms divers,

et les *tokens* (ou mots) qui ne sont pas parmi ces entités sont annotés `O`.

Le schéma de marquage `BIO` [22] est utilisé dans ce jeu de données. Ainsi, les entités sont préfixées par «*B-*» et «*I-*». En particulier, le préfixe «`B-`» permet de renseigner qu'il s'agit du premier mot ou du seul mot de l'expression associée à une entité. Le préfixe «`I-`» quant à lui correspond aux mots à l'intérieur ou à la fin d'une expression caractérisant une entité. Par exemple, l'expression *United Arab Emirates* qui est une entité de catégorie `LOC` sera annotée comme suit :

- *United* : `B-LOC`,
- *Arab* : `I-LOC`,
- *Emirates* : `I-LOC`.

Le jeu de données est composé de 14986 phrases dans l'échantillon d'entraînement et 3683 phrases dans l'échantillon test. Chaque ligne des jeux de données consiste en un mot, l'étiquetage morpho-syntaxique (ou *Pos-Tagging*) du mot et finalement la catégorie d'entité associée au mot. Enfin, les différentes phrases sont séparées par une ligne vide. (Voir Fig. 5.1a).

Nous considérons dans ce travail uniquement les mots et leurs entités. Les données d'entraînement sont par ailleurs *déséquilibrées* comme on peut le voir dans Fig. 5.1b, 83.4% des mots ont pour entité `O` alors que 0.6% des mots ont pour entité `I-MISC`. Ce déséquilibre est bien entendu une contrainte naturelle de la tache NER car il y a finalement peu d'entités par phrase.

```
-DOCSTART- -X- -X- O

SOCCER NN B-NP O
- : O O
JAPAN NNP B-NP B-LOC
GET VB B-VP O
LUCKY NNP B-NP O
WIN NNP I-NP O
, , O O
CHINA NNP B-NP B-PER
IN IN B-PP O
SURPRISE DT B-NP O
DEFEAT NN I-NP O
. . O O

Nadim NNP B-NP B-PER
Ladki NNP I-NP I-PER

AL-AIN NNP B-NP B-LOC
, , O O
United NNP B-NP B-LOC
Arab NNP I-NP I-LOC
Emirates NNPS I-NP I-LOC
1996-12-06 CD I-NP O
```

(a) Jeu de données

Tag	Count	Percentage
O	170522	83.4%
B-LOC	7140	3.5%
B-PER	6600	3.2%
B-ORG	6319	3.1%
I-PER	4528	2.2%
I-ORG	3704	1.8%
B-MISC	3438	1.7%
I-LOC	1157	0.6%
I-MISC	1155	0.6%

(b) Répartition

FIGURE 5.1: Données CoNLL-2003

5.2 Le jeu de données brevets IFPEN

Nous avons également utilisé un jeu de données composé de revendications de brevets dans le milieu de la catalyse annotées par les équipes IFPEN pour la reconnaissance d'entités nommées. Ici, les entités présentes dans notre corpus de brevets sont les suivantes :

- `CATALYST` : concept de catalyseur,
- `SUPPORT` : support associé à un catalyseur,
- `PORE_VOLUME` : propriété de volume poreux relative à un catalyseur ou un support,
- `SURFACE_AREA` : propriété de surface relative à un catalyseur ou un support,
- `PV_UNIT` : unité du volume poreux,
- `PV_VAL_MAX` : valeur maximale du volume poreux,

- `PV_VAL_MIN` : valeur minimale du volume poreux,
- `SA_UNIT` : unité de surface,
- `SA_VAL_MAX` : valeur maximale de surface,
- `SA_VAL_MIN` : valeur minimale de surface,
- `LOCUTION` : élément permettant d'identifier les valeurs.

Ces entités ont été identifiées pour permettre une première évaluation des méthodes de traitement automatique du langage afin d'extraire des valeurs numériques associées à des concepts métiers issus de la catalyse (Voir Fig. 3.2).

Ajout de préfixes aux entités Nous avons réaliser une étape de pré-traitement des données consistant à intégrer le schéma de marquage `BIO` aux entités de manière similaire aux données CoNLL-2003. En effet, cet ajout de préfixe permet de considérer les expressions annotées composées de plusieurs mots, comme par exemple, «*of at least*» ou «*total pore volume*» qui sont des entités de catégorie respective `LOCUTION` et `PORE_VOLUME`. De plus, les mots qui ne sont pas annotésc sont classiquement dans la catégorie `O`.

Segmentation des phrases Une étape importante de pré-traitement des données consiste à utiliser une méthode de *tokenisation* ou segmentation afin de transformer nos données sous forme de texte en une série de *tokens* individuels. Nous considérons qu'un token correspond à un mot. Dans la suite, on appelle *tokenizer* l'outil permettant de segmenter une phrase, c'est-à-dire créer les tokens. Par exemple, considérons l'exemple de revendication suivante :

> 5. The process of claim 1 wherein said terminal catalyst composition comprises 0 to 50 wt percent as much alkali metal or alkaline earth metal as said initial catalyst composition .

Si nous utilisons un *tokenizer* classique qui considère les espaces comme des séparateur, nous obtenons :

> ['5.', 'The', 'process', 'of', 'claim', '1', 'wherein', 'said', 'terminal', 'catalyst', 'composition', 'comprises', '0', 'to', '50', 'wt', 'percent', 'as', 'much', 'alkali', 'metal', 'or', 'alkaline', 'earth', 'metal', 'as', 'said', 'initial', 'catalyst', 'composition', '.']

Le tokenizer provenant de `SpaCy` qui est une bibliothèque `Python` de traitement automatique des langues, nous donne la segmentation suivante :

['5', '.', 'The', 'process', 'of', 'claim', '1', 'wherein', 'said', 'terminal', 'catalyst', 'composition', 'comprises', '0', 'to', '50', 'wt', 'percent', 'as', 'much', 'alkali', 'metal', 'or', 'alkaline', 'earth', 'metal', 'as', 'said', 'initial', 'catalyst', 'composition', '.']

On observe que par exemple '5.' est segmenté en '5' et '.' par le tokenizer `SpaCy`, contrairement au premier tokenizer. Enfin, le modèle *BERT* nécessite l'utilisation d'un *tokenizer* spécifique :

['5', '.', 'the', 'process', 'of', 'claim', '1', 'wherein', 'said', 'terminal', 'catalyst', 'composition', 'comprises', '0', 'to', '50', 'w', '##t', 'percent', 'as', 'much', 'al', '##kali', 'metal', 'or', 'al', '##kali', '##ne', 'earth', 'metal', 'as', 'said', 'initial', 'catalyst', 'composition', '.']

On remarque la présence de mots comportant les symboles «##». Il s'agit des mots qui ne sont pas présents dans le vocabulaire du tokenizer BERT. Par exemple, 'alkali' n'est pas dans ce vocabulaire donc la tokenisation de BERT segmente ce mot en sous-mots qu'il connaît : 'al' et '##kali'. Les symboles ## signifient que le token doit être attaché au token précédent sans espaces.

Dans la suite, nous avons utilisé le premier tokenizer sauf lors de l'application du modèle BERT, nous avons utilisé le tokenizer BERT.

Le jeu de données de revendications IFPEN est composé de 684 phrases (ou revendications) dans l'échantillon d'entraînement et de 194 phrases dans l'échantillon test. Dans l'échantillon d'entraînement, la distribution des entités est représentée dans Fig. 5.2b et on observe que 92% des mots du jeu de données sont des entités `O`. Les données ne sont donc pas *équilibrées*.

and	O
catalyst	B-Catalyst
particles,	O
said	O
catalyst	B-Catalyst
comprising	O
molybdenum	O
and	O
carbon	O
particles	O
comprising	O
pore	B-Pore_volume
volume	I-Pore_volume
of	B-Locution
at	I-Locution
least	I-Locution
0.2	B-PV_val_min
cc/g,	B-PV_unit
and	O
a	O

(a) Extrait du jeu de données

Tag	Count	Percentage
O	34174	92.0%
I-Locution	722	1.9%
B-Locution	515	1.4%
B-Catalyst	510	1.4%
B-Support	318	0.9%
I-Pore_volume	193	0.5%
B-Pore_volume	139	0.4%
I-Support	105	0.3%
I-Surface_area	90	0.2%
B-Surface_area	83	0.2%
B-PV_val_min	63	0.2%
B-SA_val_min	63	0.2%
B-PV_val_max	50	0.1%
B-SA_val_max	49	0.1%
I-Catalyst	32	0.1%
B-PV_unit	25	0.1%
B-SA_unit	23	0.1%
I-PV_unit	5	0.0%
I-SA_unit	4	0.0%

(b) Pourcentages et nombre d'entités dans le jeu de données d'entraînement

FIGURE 5.2: Données brevets

Il est également à noter la faible représentation des entités `SA_UNIT` et `PV_UNIT` qui risquent d'être mal détectées.

5.3 Méthodes utilisées

On présente dans cette section les différents modèles d'apprentissage profond utilisés pour la tâche de reconnaissance d'entités nommées.

5.3.1 Modèle BiLSTM

Nous avons tout d'abord implémenté un modèle *LSTM multi-couche et bidirectionnel* (ou BiLSTM) pour faire la reconnaissance d'entités nommés. L'architecture du modèle est similaire à celle de la partie précédente . Néanmoins, il s'agit ici de classifier chaque mot d'une revendication et non pas de classifier la revendication globalement (Voir Fig. 5.3).

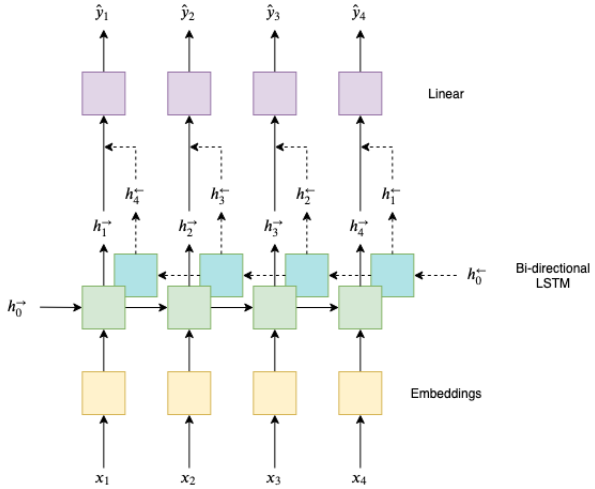


FIGURE 5.3: Modèle BiLSTM pour NER

Le modèle prend une séquence de tokens $X = \{x_1, x_2, \ldots, x_T\}$, les convertit numériquement à travers une couche d'*embedding* notée e pour obtenir les vecteurs de représentation

$$e(X) = \{e(x_1), e(x_2), ..., e(x_T)\}.$$

Ces vecteurs sont ensuite opérés de manière itérative (un vecteur par itération) par les modèles LSTM *forward* et *backward*. Le modèle LSTM *forward* traite la séquence de gauche à droite, tandis que le modèle LSTM *backward* traite la séquence de droite à gauche, c'est-à-dire que la première entrée du modèle *forward* est x_1 tandis que la première entrée du modèle *backward* est x_T. Ces modèles prennent également en compte les états cachés h et cellule c de l'itération précédente.

$$h_t^{\rightarrow} = \text{LSTM}^{\rightarrow}(e(x_t^{\rightarrow}), h_{t-1}^{\rightarrow}, c_{t-1}^{\rightarrow})$$
$$h_t^{\leftarrow} = \text{LSTM}^{\leftarrow}(e(x_t^{\leftarrow}), h_{t-1}^{\leftarrow}, c_{t-1}^{\leftarrow})$$

Une fois que toute la séquence a été traitée, les états masqués et cellules sont ensuite transmis à la couche suivante du LSTM. Les états masqués h_0 et de cellules c_0 initiaux pour chaque direction et couche sont initialisés à un tenseur égal à zéro.

Les états cachés *forward* et *backward* de la couche finale du modèle sont ensuite concaténés

$$H = \{h_1, h_2, ...h_T\}$$

où $h_i = [h_i^{\rightarrow}, h_{T+i-1}^{\leftarrow}]$ pour tout $i \in \{1, \ldots, T\}$.

Finalement, chaque vecteur h_i passe à travers une couche FC qui est utilisé pour prédire la catégorie $\hat{y}_i$ d'entité s'appliquant au token telle que $\hat{y}_i = f(h_i)$.

5.3.2 Modèle CRF

Afin d'apporter une meilleure robustesse, il est possible d'associer au modèle précédent une méthode dite de *champs aléatoires conditionnels* (ou *Conditional Random Fields*, CRF). On présente ici les grandes lignes de cette méthode.

La méthode CRF est généralement utilisée pour la reconnaissance d'entités nommées dans le but de prédire des séquences valides. Par exemple, il ne devrait pas être possible d'avoir d'entité `B-LOCUTION` suivie directement par une autre `B-LOCUTION`. Les modèles CRF visent à réduire ces incohérences et sont des modèles graphiques probabilistes fréquemment utilisés pour la reconnaissance d'entités nommées. Ils prennent en entrée un ensemble de structure d'éléments x et fournissent en sortie un étiquetage structuré d'éléments y de cet ensemble. La probabilité conditionnelle d'un ensemble d'étiquettes y selon une entrée x est modélisée. Dans le cas *séquentiel*, c'est-à-dire l'étiquetage d'observations x_i par des labels y_i, la distribution de probabilité d'une séquence d'annotations y selon une séquence observable x est donnée par :

$$p(y|x) = \frac{1}{Z(x)} \prod_t \exp\left[\sum_{k=1}^{k_1} \sum_{i=1}^{n} \lambda_k f_k(y_i, x) + \sum_{k=1}^{k_2} \sum_{i=1}^{n} \mu_k g_k(y_{i-1}, y_i, x) \right]$$

où

- $Z(x)$ est un facteur de normalisation,
- f et g sont repsectivement les fonctions caractéristiques locales et globales (i.e. fonctions *features*). Les fonctions f caractérisent les relations locales entre le label courant en position i et les observations. Les fonctions g caractérisent les transitions entre chaque paires de labels i et $i1$ et la séquence d'observations.
- les valeurs k_1, k_2 et n sont respectivement le nombre de fonctions *features* f, le nombre de fonctions *features* g, et la taille de la séquence de labels à prédire.

Les fonctions f et g sont généralement des fonctions binaires vérifiant une certaine combinaison de labels et d'attributs décrivant les observations et appliquées à chaque position de la séquence. Ces fonctions sont pondérées par les coefficients λ_k et μ_k qui estiment l'importance de l'information apportée pour déterminer la classe. Ce sont les paramètres associés aux fonctions f et g. L'apprentissage des modèles CRF consiste à estimer le vecteur de paramètres

$$\theta = \{\lambda_1, \ldots, \lambda_{k_1}, \mu_1, \ldots, \mu_{k_2}\}$$

à partir de données d'entraînement, c'est-à-dire N séquences étiquetées $(x(i), y(i))_{i=1}^{i=N}$.

Après cette étape d'apprentissage, l'application des modèles CRF à de nouvelles données consiste à trouver la séquence de labels la plus probable étant donnée une séquence d'observations non-vue.

Ainsi, nous avons construit un modèle composé :

1. une couche de *word embedding*,

2. une couche *BiLSTM*,
3. une couche *FC*,
4. une couche *CRF*

afin de prédire les entités.

5.3.3 Modèle BiLSTM avec Attention et CRF

Le modèle d'attention présenté dans la partie 4.2.2.4 peut être ajouté au modèle composé du BiLSTM et du CRF présenté dans la section précédente. Le type d'attention utilisé est celui de type *clé-valeur* décrit dans la partie 4.2.2.5.

De manière classique, un conversion par une couche d'*embedding* est utilisée puis un modèle BiLSTM est appliqué. Les vecteurs de sortie de ce BiLSTM passent ensuite par une couche d'auto-attention muti-têtes. Dans cette couche d'attention muti-têtes, le score d'attention est calculé en appliquant la formule suivante :

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

où d_k est la dimension pour chaque tête et où les valeurs de Q, K et V sont toutes égales à la sortie du modèle BiLSTM.

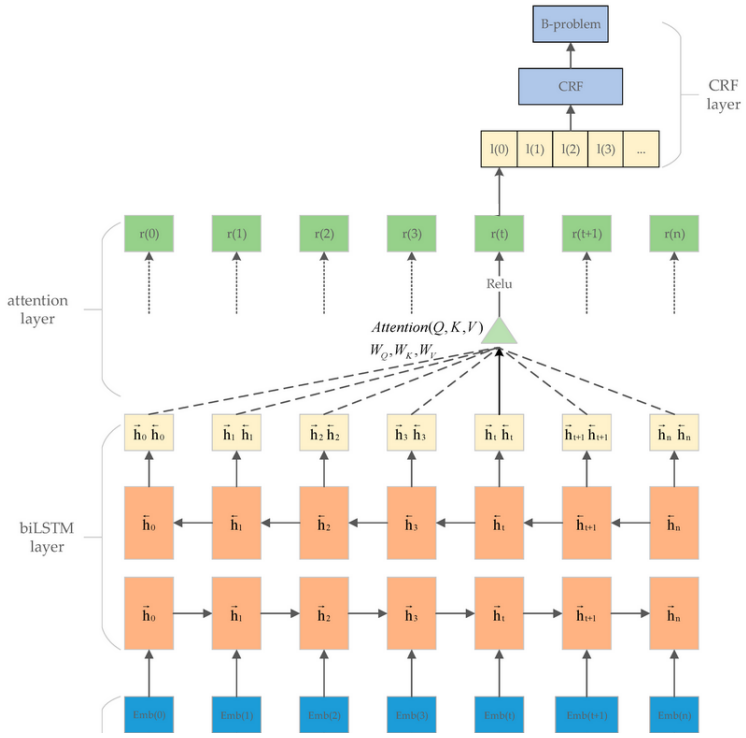


FIGURE 5.4: Modèle BiLSTM, Attention et CRF

Enfin, les matrices d'attention obtenues sont concaténées puis une couche FC est utilisée pour prédire les labels. En dernier lieu, une couche CRF est utilisée pour décoder les étiquettes séquentielles.

5.3.4 Modèle Transformer

Nous avons ensuite remplacé le modèle BiLSTM utilisé dans le modèle précédent par un modèle *Transformer*. Comme dans le cas de la classification, seuls les encodeurs sont utiles pour encoder l'information de chaque mot et déterminer son entité.

Le modèle contient :

- une couche *word embedding*,
- une couche *transformer* composée :
  - d'une couche de *positional encoder* qui est additionnée au *word embedding*,
  - d'une couche *encodeur* décrit dans la partie 4.2.2.5 et constituée d'une couche d'*attention* à quatre têtes,
- une couche *FC* avec une fonction d'activation GELU [14] et une autre couche *FC* qui transforme la dimension de la sortie en nombre correcte de tags possible pour la couche *CRF*,
- une couche *CRF*.

5.3.5 Modèle BERT

Nous avons ensuite utilisé un modèle *BERT* pré-entraîné puis *fine-tuné* pour une tâche de reconnaissance d'entités nommées (Voir Fig. 5.5).

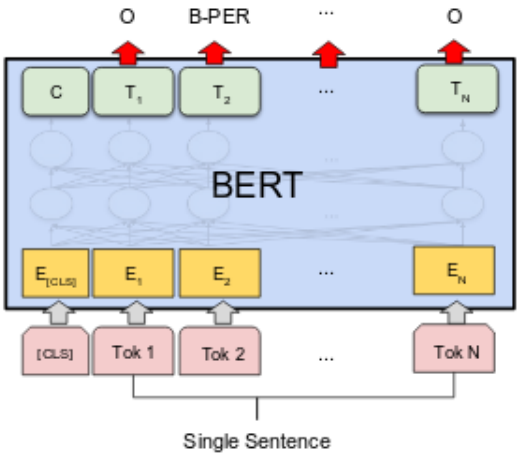


FIGURE 5.5: Modèle BERT pour la reconnaissance d'entités nommées

Nous avons appliqué le tokenizer BERT pour pouvoir utiliser ce modèle. Étant donné que ce tokenizer produit des sous-mots, il a fallu réaliser une étape supplémentaire d'ajout de tags. Par exemple, si le mot 'alkali' a pour entité `B-SUPPORT`. Les sous-mots 'al' et '##kali' produit par le tokenizer BERT ont tous deux pour entité `B-SUPPORT`.

Le modèle BERT a été utilisé d'une part en utilisant une librairie implémentée par *Hugging Face* [5]. Nous avons donc simplement appliqué une tâche de reconnaissance d'entités nommées en utilisant nos données avec un modèle BERT pré-entraîné mis à disposition par *Hugging Face*. D'autre part, en utilisant les librairies proposées par `PyTorch`, nous avons également implémenté le modèle composé d'une succession d'encodeur et finalement d'une couche *FC* pour prédire les entités.

5.4 Évaluation des modèles

Dans cette section, les différents modèles présentés précédemment sont comparés. Le cadre d'implémentation est le même que celui présenté dans la partie 4.3. Des notebooks ont été mis à disposition pour chaque apprentissage.

5.4.1 Métriques

Le jeu de données étant *déséquilibré*, on utilise les métriques suivantes pour pouvoir comparer les modèles entre eux :

- Rappel (ou *Recall*) :
$$rappel = \frac{TP}{TP + FN}$$
,
- Précision :
$$precision = \frac{TP}{TP + FP}$$
,
- F1 score :
$$F1\ Score = 2 * \frac{precision * recall}{precision + recall}$$

où les valeurs TP, TN, FP et FN sont définis dans la partie 4.3.2.

5.4.2 Résultats

On présente ici les résultats obtenus sur les deux jeux de données.

5.4.2.1 Les données CoNLL-2003

Le tableau 5.1 présente les résultats des différents modèles considérés.

Table 5.1: Résultats pour la reconnaissance d'entités nommés (en moyenne)

Modèle	Section	F1 Score
BiLSTM	5.3.1	0.69
BiLSTM+CRF	5.3.2	0.62
BiLSTM+Attention+CRF	5.3.3	0.64
Transformers	5.3.4	0.64
BERT	5.3.5	**0.93**

Les résultats pour chaque entité sont présentés dans les tableaux 5.2, 5.3, 5.4, 5.5 et 5.6.

Table 5.2: Modèle BiLSTM par entité

Entité	F1 Score	Précision	Rappel
O	0.97	0.96	0.98
LOC	0.83	0.88	0.79
PER	0.77	0.97	0.0.63
ORG	0.66	0.6	0.73
MISC	0.7	0.77	0.64

Table 5.3: Modèle BiLSTM+CRF par entité

Entité	F1 Score	Précision	Rappel
O	0.95	0.91	0.99
LOC	0.60	0.78	0.49
PER	0.59	0.74	0.49
ORG	0.56	0.76	0.45
MISC	0.42	0.68	0.31

Table 5.4: Modèle BiLSTM+Attention+CRF par entité

Entité	F1 Score	Précision	Rappel
O	0.95	0.91	1
LOC	0.75	0.88	0.65
PER	0.42	0.68	0.31
ORG	0.53	0.91	0.38
MISC	0.58	0.69	0.5

Table 5.5: Modèle Transformer par entité

Entité	F1 Score	Précision	Rappel
O	0.95	0.91	1
LOC	0.78	0.82	0.75
PER	0.25	0.92	0.15
ORG	0.58	0.91	0.43
MISC	0.67	0.81	0.58

Table 5.6: Modèle BERT par entité

Entité	F1 Score	Précision	Rappel
O	1	1	1
LOC	0.94	0.93	0.94
PER	0.99	0.98	1
ORG	0.92	0.93	0.92
MISC	0.87	0.86	0.87

Les résultats obtenus, sauf pour le modèle BERT, sont bien inférieurs à ceux attendus. En effet dans la littérature [7], les modèles présentés dans cette partie donne de très bons résultats sur le jeu de données CoNLL-2003. Cette différence peut s'expliquer par un problème d'apprentissage, il faudrait alors faire une étude plus poussée pour trouver les meilleurs hyperparamètres pour chacun de ces modèles. On remarque par ailleurs que le meilleur modèle est le modèle *BERT* et son résultat est bien supérieur au autres modèles. Finalement, on observe que les résultats du modèle *BiLSTM* est supérieur aux modèles *BiLSTM+CRF* et *BiLSTM+Attention+CRF*, ce qui est très surprenant étant donné que les ajouts des couches *Attention* et *CRF* sont censés améliorer le modèle *BiLSTM* de base. De même, un meilleur choix des hyperparamètres des modèles peut très certainement améliorer ces résultats et nous ramener à l'état de l'art.

5.4.2.2 Les données brevets

Le tableau 5.7 présente les résultats des différents modèles considérés.

TABLE 5.7: Résultats pour la reconnaissance d'entités nommés (en moyenne)

Modèle	Section	F1 Score
BiLSTM	5.3.1	0.77
BiLSTM+CRF	5.3.2	0.76
BiLSTM+Attention+CRF	5.3.3	0.79
Transformers	5.3.4	0.73
BERT	5.3.5	**0.98**

Les résultats pour chaque entité sont présentés dans les tables 5.8, 5.9, 5.10 et 5.12.

TABLE 5.8: Modèle BiLSTM par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.87	0.88	0.85
LOCUTION	0.90	0.83	0.97
PV_UNIT	0.67	1	0.50
PV_VAL_MAX	0.50	1	0.33
PV_VAL_MIN	1	1	1
PORE_VOLUME	0.86	0.79	0.93
SA_UNIT	0.29	1	0.17
SA_VAL_MAX	0.73	0.67	0.80
SA_VAL_MIN	0.70	0.89	0.57
SUPPORT	0.72	0.79	0.67
SURFACE_AREA	0.95	0.90	1

Table 5.9: Modèle BiLSTM+CRF par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.81	0.72	0.93
LOCUTION	0.99	0.99	0.99
PV_UNIT	0.5	1	0.33
PV_VAL_MAX	0.77	0.83	0.71
PV_VAL_MIN	0.89	0.89	0.89
PORE_VOLUME	0.71	0.73	0.69
SA_UNIT	0	0	0
SA_VAL_MAX	0.5	1	0.33
SA_VAL_MIN	0.95	0.9	1
SUPPORT	0.81	0.8	0.83
SURFACE_AREA	0.65	0.85	0.52

Table 5.10: Modèle BiLSTM+Attention+CRF par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.88	0.92	0.85
LOCUTION	0.90	0.85	0.96
PV_UNIT	0.67	1	0.5
PV_VAL_MAX	0.5	1	0.33
PV_VAL_MIN	0.94	1	0.89
PORE_VOLUME	0.93	0.96	0.9
SA_UNIT	0.45	0.5	0.42
SA_VAL_MAX	0.89	1	0.8
SA_VAL_MIN	0.78	1	0.64
SUPPORT	0.76	0.83	0.71
SURFACE_AREA	0.96	0.93	1

Table 5.11: Modèle Transformer par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.86	0.87	0.86
LOCUTION	0.87	0.91	0.83
PV_UNIT	1	1	1
PV_VAL_MAX	0.44	0.67	0.33
PV_VAL_MIN	0.67	0.67	0.67
PORE_VOLUME	0.9	0.87	0.93
SA_UNIT	0.4	1	0.25
SA_VAL_MAX	0.6	0.6	0.6
SA_VAL_MIN	0.56	0.64	0.5
SUPPORT	0.7	0.74	0.67
SURFACE_AREA	0.9	0.85	0.95

Table 5.12: Modèle BERT par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.86	0.87	0.84
LOCUTION	0.83	0.91	0.76
PV_UNIT	0	0	0
PV_VAL_MAX	0.84	0.82	0.87
PV_VAL_MIN	0.84	0.72	1
PORE_VOLUME	0.85	0.74	1
SA_UNIT	0.88	0.79	1
SA_VAL_MAX	0.99	1	0.99
SA_VAL_MIN	0.92	0.84	1
SUPPORT	0.88	0.82	0.95
SURFACE_AREA	0.47	0.81	0.33

Les résultats sont globalement bons et sont meilleurs que ceux obtenus pour le jeu de données CoNLL-2003. Le meilleur modèle est le modèle *BERT* qui donne un très bon F1 Score (0.98). Le modèle *Transformers*, quant-à-lui est le modèle le moins performant. Ce résultat peut s'expliquer par son architecture complexe et par le fait que, contrairement au modèle *BERT*, il est entraîné seulement avec nos données. Ainsi, on note l'importance du pré-entraînement qui explique la grande différence de performance entre le modèle *BERT* et les autres modèles.

6. Extraction de relation

L'*extraction de relation* (ou RE) consiste à extraire des relations sémantiques à partir de texte. Ces relations se produisent entre deux entités. Ceci implique donc d'avoir détecter les entités au préalable, typiquement en utilisant un algorithme de NER qui pourrait être un de ceux présentés dans la section précédente.

6.1 Le jeu de données SemEval-2010 Task 8

Le jeu de données SemEval-2010 Task 8 [19] est un jeu de données fréquemment utilisé pour évaluer les algorithmes d'extraction de relations. Toutefois, nous ne présenterons pas en détail ce jeu de données. Les relations présentes dans ce jeu de données sont les paires d'entités suivantes :

- `Cause - Effect`,
- `Instrument - Agency`,
- `Product - Producer`,
- `Content - Container`,
- `Entity - Origin`,
- `Entity - Destination`,
- `Component - Whole`,
- `Member - Collection`,
- `Message - Topic`,
- `Other`.

Le format des données est illustré par les exemples suivants :

```
15 "They saw that the <e1>equipment</e1> was put inside rollout
    <e2>drawers</e2>, which looked aesthetically more pleasing and tidy."
Content-Container(e1,e2)
Comment: the drawer contains the equipment, typical example of
         Content-Container; no movement
```

```
20 ""Any time," he told her before turning to the <e1>boy</e1> who was in
   the <e2>desk</e2> next to him."
Other
Comment: the desk does not contain the boy.
```

La première ligne contient la phrase entre guillemets précédée d'un identifiant numérique. Chaque phrase est annotée de trois informations :

- Deux mentions d'entité dans la phrase sont marquées entre les balises `<e1></e1>` et `<e2></e2>`.
- Si l'une des relations sémantiques est valable entre les entités `e1` et `e2`, la phrase est étiquetée avec le nom de cette relation et l'ordre dans lequel les arguments de la relation sont remplis par `e1` et `e2`. Par exemple, la relation `Cause-Effect(e1, e2)` signifie que `e1` est de type «`Cause`» et `e2` est de type «`Effect`». A l'inverse, `Cause-Effect(e2, e1)` signifie que `e2` est de type «`Cause`» et `e1` est de type «`Effect`». Si aucune des relations n'est correcte, la phrase est étiquetée «`Other`». Au total, donc, 19 relations sont possibles.
- Un commentaire peut être fourni pour expliquer pourquoi les annotateurs choisissent une relation donnée. Les commentaires sont destinés aux lecteurs humains et doivent être ignorés par les systèmes automatiques participant à la tâche.

Le jeu de données est composée de 8000 phrases dans l'échantillon d'entraînement et 2717 phrases dans l'échantillon de test. La distribution des relations dans l'échantillon d'entraînement est présentée dans Fig. 6.1.

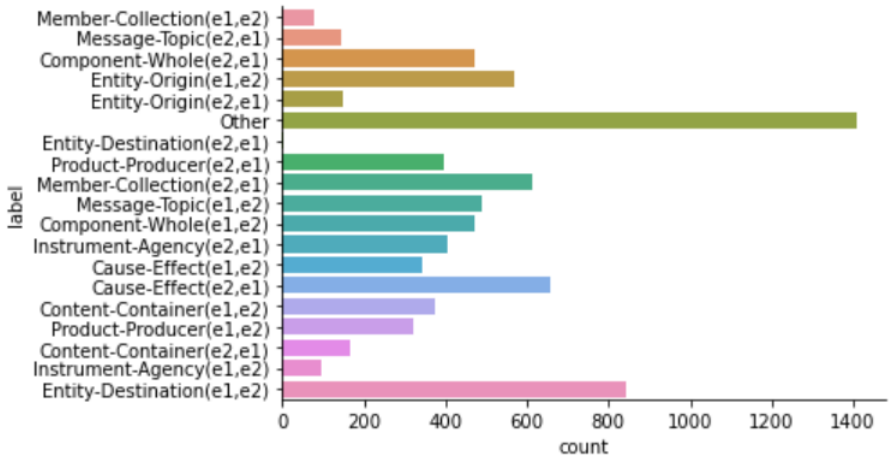


FIGURE 6.1: Distribution des relations dans l'échantillon d'entraînement

C'est un jeu de données correctement équilibré.

6.2 Les données brevets IFPEN

Les relations présentes dans le corpus de brevets sont issues de l'ontologie mise au point par les experts IFPEN (Voir Fig. 3.2) :

- `Contains`,
- `Has`,
- `Is_associated`
- `Is_defined`.

Le jeu de données est composé de 1150 phrases dans l'échantillon d'entraînement et 149 phrases dans l'échantillon test. Chaque phrase (ou revendication) contient deux entités annotées et une relation entre ces entités (Voir Fig. 6.2).

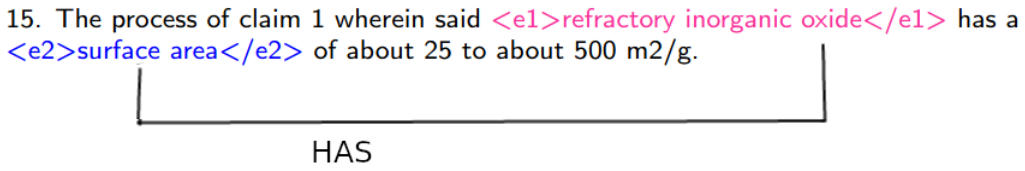


FIGURE 6.2: Revendication contenant le relation «`Has`» entre `e1` et `e2`

Pré-traitement des données Dans l'étape de pré-traitement des données, nous avons ajouté une relation «`Other`» qui relie deux entités annotées mais qui n'ont pas de relation entre elles définie. En effet, deux entités dans une même phrase ne sont pas nécessairement reliées par une relation définie dans l'ontologie. Cette relation a été introduite de sorte que la distribution des relations dans le jeu de données soit équilibrée au mieux.

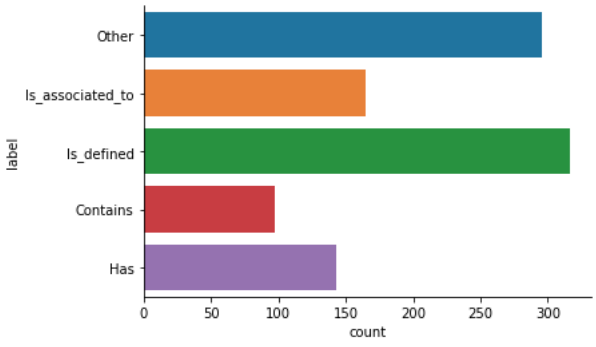


FIGURE 6.3: Distribution des relations dans l'échantillon d'entraînement

On observe dans Fig. 6.3 que la distribution des relations dans l'échantillon d'entraînement est relativement *équilibrée*.

6.3 Méthodes utilisées

On présente dans cette section les différents modèles d'apprentissage profond utilisés pour la tâche d'extraction de relation.

6.3.1 Modèle CNN

La première méthode utilisée pour réaliser l'extraction de relation est l'application de réseaux de neurones convolutifs (CNN) [27] dans une approche similaire à celle présentée dans la partie 4.2.2.3.

L'architecture du modèle est décrite sur la figure 6.8. Ce modèle s'appuie sur une représentation numérique des mots par *word embedding* comme dans tous les modèles précédents. Dans le cas de l'extraction de relation, on considère également une représentation des *positions* des mots par rapport à la première entité ainsi que par rapport à la deuxième entité. Ces représentations sont nommées les *position embeddings* et joue significativement le même rôle qu'un *word embedding*. En effet, dans la tâche d'extraction de relation, les mots proches des entités cibles sont généralement plus informatifs pour déterminer la relation entre les entités. Cela peut ainsi aider le modèle CNN à suivre la proximité de chaque mot à la première entité ou la seconde entité. Cette proximité est définie comme la combinaison des distances relatives du mot courant à la tête ou à l'entité de queue.

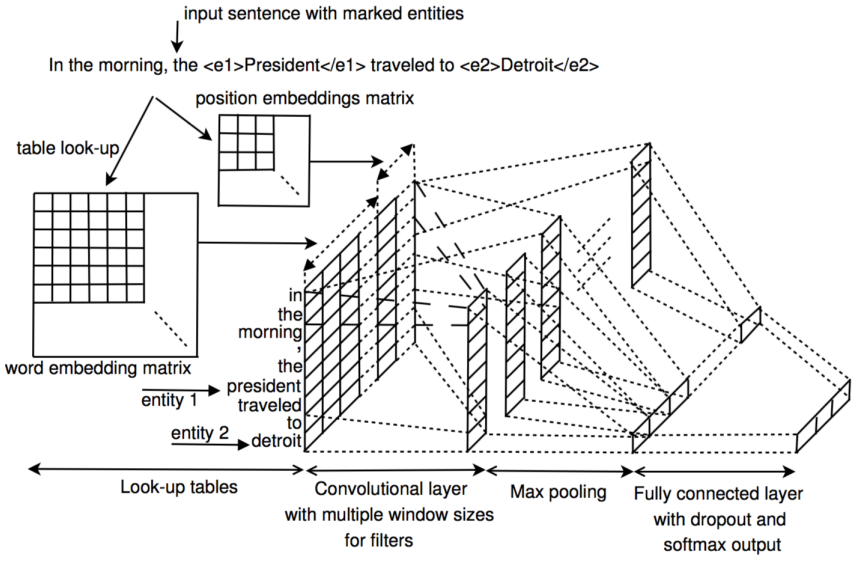


FIGURE 6.4: Réseaux de neurones convolutifs pour l'extraction de relations

Pour illustrer le principe de proximité, considérons la phrase suivante :

The <e1> game </e1> was sealed in the original <e2> packing </e2> unopened and untouched.

Dans cette phrase, la distance relative entre le mot «sealed» et l'entité «game» est de 2 et la distance relative entre le mot «sealed» et l'entité «packing» est -4.

Les représentations *word embedding* et *position embeddings* sont ensuite concaténées. Puis, des réseaux convolutifs composés de différents filtres sont appliqués à ces *embeddings* de la même manière que dans la partie 4.2.2.3 proposant ainsi la même analogie au n-grammes. Enfin, on applique une couche *max pooling* et une dernière couche *FC* pour prédire la relation.

6.3.2 Modèle CNN avec Attention

Nous ajoutons ensuite un mécanisme d'attention au modèle précédent [28].

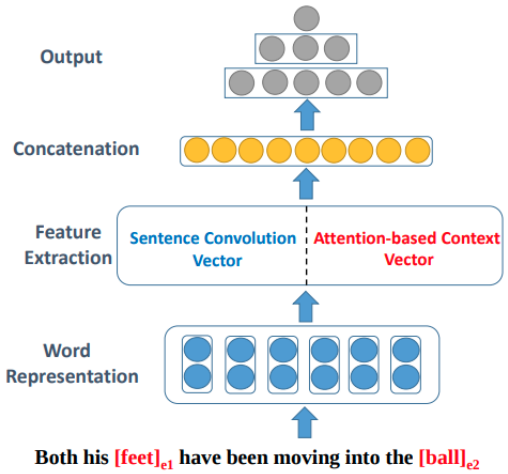
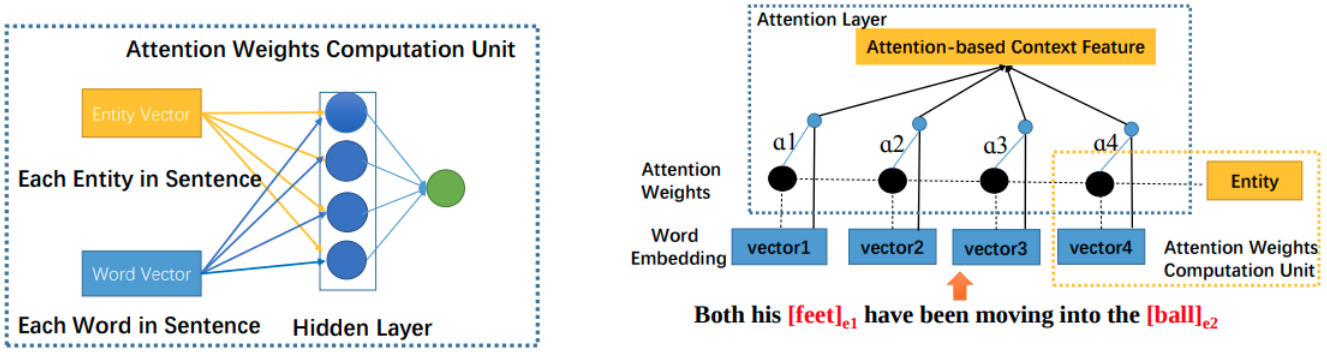


FIGURE 6.5: Réseaux de neurones convolutifs et attention pour l'extraction de relation

Comme pour le modèle précédent, le modèle illustré dans Fig. 6.9 est constitué des mêmes représentations *word embedding* et *position embeddings* ainsi que des couches de réseaux convolutifs et de *max pooling*. Néanmoins, le modèle présenté ici utilise en plus un mécanisme d'attention. Les deux couches *CNN* et *attention* sont alors concaténées et une couche *FC* est finalement appliquée pour prédire la relation. Ainsi, la couche d'attention joue un rôle d'extraction de caractéristiques au même titre que les couches convolutives.

Le modèle d'attention utilisé est présenté maintenant. Ce modèle d'attention est appliqué à un type d'usage assez différent de celui vu précédemment dans la partie 4.2.2.4. Ici, l'attention

porte sur les mots d'une phrase et des deux entités. Il est donc nécessaire que le modèle ait la capacité de déterminer quelles parties de la phrase sont les plus influentes par rapport aux deux entités concernées.



(a) Attention weights computation unit.

(b) Attention layer network.

FIGURE 6.6: Mécanisme d'attention

Afin de calculer le poids de chaque mot dans la phrase, chaque mot de la phrase et chaque entité alimentent un réseau neuronal. La structure du réseau de calcul du poids d'attention est illustré à la figure 6.6. On considère ici que chaque phrase contient T mots. Dans la suite, $w_{i,t}$ pour $t \in [1, T]$ est la représentation numérique des mots de la phrase i et $e_{i,j}$ pour $j \in [1, 2]$ la représentation de l'entité j. Les représentations de l'entité $e_{i,j}$ et du mot $w_{i,t}$ sont concaténées pour obtenir une nouvelle représentation du mot t :

$$h_{i,t}^{j} = [w_{i,t}, e_{i,j}].$$

On note $u_{i,t}^{j}$ le degré de pertinence du mot t par rapport à l'entité j de la phrase i qui se calcule de la manière suivante :

$$u_{i,t}^{j} = W_a[\tanh(W_{we}h_{i,t}^{j} + b_{we})] + b_a$$

où W_a, W_{we}, b_a et b_{we} sont respectivement des poids et biais appris par le modèle.
Une fonction *softmax* est finalement utilisée :

$$\alpha_{i,t}^{j} = \frac{e^{u_{i,t}^{j}}}{\sum_t e^{u_{i,t}^{j}}}.$$

L'architecture de la couche d'attention proposée est illustrée dans Fig. 6.6 (b). Après cela, le vecteur de contexte de phrase $s_{i,j}$ sur l'entité j est calculée comme une somme pondérée du mot dans la phrase i :

$$s_{i,j} = \sum_t \alpha_{i,t}^{j} w_{i,t}$$

Ce vecteur de contexte peut être interprété comme une représentation renseignant sur le mot informatif parmi les mots de la phrase. Le poids de l'attention du réseau est initialisé aléatoirement et conjointement appris pendant le processus de formation.

Au final, les sorties des trois réseaux présentés comprenant le résultat du réseau de convolution et des vecteurs de contexte de phrase des deux entités sont obtenues. Ces vecteurs sont ensuite concaténés en un vecteur de *features* de longueur fixe. Le vecteur obtenu est ensuite introduit dans une couche *FC*. Pour la tâche de classification, les sorties sont les probabilités de différentes classes qui sont calculées par une fonction *softmax* en sortie de la couche *FC*.

6.3.3 Modèle BiLSTM avec Attention

Dans cette partie, un modèle BiLSTM associé à un mécanisme d'attention pour la tâche d'extraction de relations est utilisé [24].

Le modèle est composé des couches suivantes :

- couche de *word embedding* pour la représentation des mots en un vecteur numérique de dimension fixe ;
- couche BiLSTM comme présentée 5.3.1 ;
- couche d'attention afin produire un vecteur de poids et de fusionner les caractéristiques des mot de chaque pas de itération en un vecteur de caractéristiques pour la phrase (en multipliant le vecteur de poids) ;
- Couche finale où le vecteur d'entités au niveau de la phrase est finalement utilisé pour la classification des relations.

Dans ce modèle, le mécanisme d'attention est le suivant. Soit H une matrice constituée de vecteurs de sortie $[h_1, h_2, ..., h_T]$ que la couche LSTM a produite où T est la longueur de la phrase. La représentation r de la phrase est formée par une somme pondérée de ces vecteurs de sortie :

$$M = \tanh(H)$$
$$\alpha = \text{softmax}(w^T M)$$
$$r = H\alpha^T$$

où w est un vecteur de paramètres entraîné. La représentation finale utilisée pour la classification est $h^* = \tanh(r)$. Une fonction *softmax* est utilisée pour trouver les probabilités des différentes relations.

6.3.4 Modèle BERT

En dernier lieu, une variante du modèle BERT est utilisée pour l'extraction d'extraction [26]. Pour rappel, le modèle BERT a donné de très bons résultats sur les tâches précédentes, il est donc naturel d'évaluer son potentiel de généralisation à la tâche d'extraction de relation.

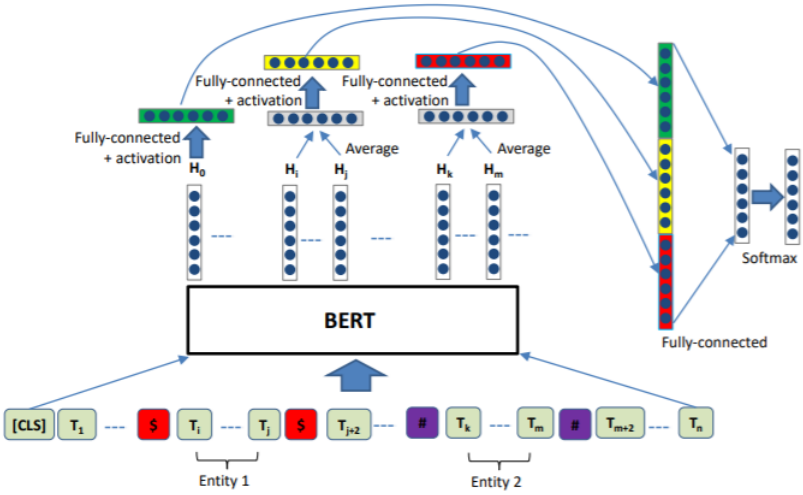


FIGURE 6.7: BERT

Un modèle BERT pré-entraîné est utilisé. Pour une phrase s contenant deux entités e_1 et e_2, le modèle doit capturer l'emplacement des deux entités dans l'ensemble des mots. Pour cela, une convention particulière est appliquée. Un caractère spécial «$» est placé au début et à la fin de la première entité et de manière similaire un caractère spécial «#» autour de la deuxième entité. Au début de chaque phrase, la chaîne « [CLS] » est également ajoutée. Par exemple, après l'insertion des caractères spéciaux, on peut avoir :

> [CLS] The $kitchen$ is the last renovated part of the #house#.

où les entités sont donc "kitchen" et "house".

Étant donné une phrase s avec les entités e_1 et e_2, supposons que la sortie d'état caché final fourni par le modèle BERT soit H. On suppose que les vecteurs H_i à H_j sont les derniers vecteurs d'état cachés de BERT pour l'entité e_1, et H_k à H_m sont les derniers vecteurs d'état cachés de BERT pour l'entité e_2. Nous appliquons l'opération moyenne pour obtenir une représentation vectorielle pour chacune des deux entités cibles. Puis après application d'une fonction d'activation *tanh*, nous ajoutons une couche FC à chacun des deux vecteurs. Les sorties pour ce traitement des entités e_1 et e_2 sont respectivement des vecteurs H_1' et H_2'. Ce processus peut être formalisé comme suit :

$$H_1' = W_1 \left[\tanh \left(\frac{1}{j-i+1} \sum_{t=i}^{j} H_t \right) \right] + b_1$$

$$H_2' = W_2 \left[\tanh \left(\frac{1}{m-k+1} \sum_{t=k}^{m} H_t \right) \right] + b_2$$

avec $W_1 = W_2$, $b_1 = b_2$. Le vecteur d'état caché final du début de phrase (i.e. « `[CLS]` ») est obtenu par :

$$H'_0 = W_0(\tanh(H_0)) + b_0.$$

Les vecteurs b_0, b_1, b_2, b_3 sont des vecteurs de biais. Les matrices de poids W_0, W_1, W_2 ont les mêmes dimensions. Les sorties H'_0, H'_1, H'_2 sont finalement concaténées puis une couche *softmax* est appliquée pour obtenir la prédiction.

6.4 Évaluation des modèles

Pour pouvoir comparer nos modèles nous avons utilisé les mêmes métriques que dans la partie précédente.

6.4.1 Les données SemEval2010 Task 8

Le tableau 6.1 présente les résultats des différents modèles considérés.

TABLE 6.1: Résultats pour l'extraction de relations (en moyenne)

Modèle	**Section**	**F1 Score**
CNN	6.3.1	0.68
CNN+Attention	6.3.2	0.69
BiLSTM+Attention	6.3.3	0.76
BERT	6.3.4	**0.84**

Les résultats pour chaque relation sont présentés dans Fig. 6.8, Fig. 6.9, Fig. 6.10 et Fig. 5.12.

	precision	recall	f1-score
Cause-Effect(e1,e2)	0.91	0.84	0.87
Cause-Effect(e2,e1)	0.85	0.88	0.87
Component-Whole(e1,e2)	0.56	0.81	0.66
Component-Whole(e2,e1)	0.59	0.55	0.57
Content-Container(e1,e2)	0.68	0.92	0.78
Content-Container(e2,e1)	0.70	0.77	0.73
Entity-Destination(e1,e2)	0.74	0.92	0.82
Entity-Destination(e2,e1)	0.00	0.00	0.00
Entity-Origin(e1,e2)	0.72	0.89	0.79
Entity-Origin(e2,e1)	0.70	0.85	0.77
Instrument-Agency(e1,e2)	0.50	0.27	0.35
Instrument-Agency(e2,e1)	0.59	0.65	0.62
Member-Collection(e1,e2)	0.56	0.28	0.38
Member-Collection(e2,e1)	0.76	0.89	0.82
Message-Topic(e1,e2)	0.68	0.87	0.76
Message-Topic(e2,e1)	0.36	0.82	0.50
Other	0.53	0.15	0.24
Product-Producer(e1,e2)	0.68	0.69	0.68
Product-Producer(e2,e1)	0.82	0.37	0.51

FIGURE 6.8: CNN

	precision	recall	f1-score
Cause-Effect(e1,e2)	0.84	0.87	0.86
Cause-Effect(e2,e1)	0.88	0.84	0.86
Component-Whole(e1,e2)	0.70	0.69	0.70
Component-Whole(e2,e1)	0.63	0.54	0.58
Content-Container(e1,e2)	0.76	0.82	0.79
Content-Container(e2,e1)	0.72	0.67	0.69
Entity-Destination(e1,e2)	0.81	0.87	0.84
Entity-Destination(e2,e1)	0.00	0.00	0.00
Entity-Origin(e1,e2)	0.76	0.84	0.80
Entity-Origin(e2,e1)	0.84	0.79	0.81
Instrument-Agency(e1,e2)	0.44	0.36	0.40
Instrument-Agency(e2,e1)	0.65	0.53	0.58
Member-Collection(e1,e2)	0.59	0.41	0.48
Member-Collection(e2,e1)	0.70	0.83	0.76
Message-Topic(e1,e2)	0.70	0.71	0.71
Message-Topic(e2,e1)	0.71	0.59	0.65
Other	0.38	0.41	0.40
Product-Producer(e1,e2)	0.70	0.59	0.64
Product-Producer(e2,e1)	0.59	0.46	0.52

FIGURE 6.9: CNN+Attention

	precision	recall	f1-score
Cause-Effect(e1,e2)	0.90	0.90	0.90
Cause-Effect(e2,e1)	0.89	0.90	0.90
Component-Whole(e1,e2)	0.82	0.78	0.80
Component-Whole(e2,e1)	0.75	0.71	0.73
Content-Container(e1,e2)	0.74	0.91	0.82
Content-Container(e2,e1)	0.88	0.74	0.81
Entity-Destination(e1,e2)	0.87	0.87	0.87
Entity-Destination(e2,e1)	0.00	0.00	0.00
Entity-Origin(e1,e2)	0.78	0.89	0.83
Entity-Origin(e2,e1)	0.93	0.79	0.85
Instrument-Agency(e1,e2)	1.00	0.05	0.09
Instrument-Agency(e2,e1)	0.71	0.71	0.71
Member-Collection(e1,e2)	0.00	0.00	0.00
Member-Collection(e2,e1)	0.74	0.93	0.82
Message-Topic(e1,e2)	0.79	0.86	0.82
Message-Topic(e2,e1)	0.78	0.55	0.64
Other	0.51	0.48	0.49
Product-Producer(e1,e2)	0.79	0.77	0.78
Product-Producer(e2,e1)	0.71	0.65	0.68

FIGURE 6.10: BiLSTM+Attention

	precision	recall	f1-score
Cause-Effect(e1,e2)	0.90	0.94	0.92
Cause-Effect(e2,e1)	0.91	0.94	0.93
Component-Whole(e1,e2)	0.68	0.77	0.72
Component-Whole(e2,e1)	0.83	0.79	0.81
Content-Container(e1,e2)	0.85	0.89	0.87
Content-Container(e2,e1)	0.86	0.85	0.86
Entity-Destination(e1,e2)	0.87	0.92	0.89
Entity-Destination(e2,e1)	0.86	0.82	0.84
Entity-Origin(e1,e2)	0.89	0.88	0.89
Entity-Origin(e2,e1)	0.82	0.89	0.86
Instrument-Agency(e1,e2)	0.93	0.93	0.93
Instrument-Agency(e2,e1)	0.00	0.00	0.00
Member-Collection(e1,e2)	0.87	0.85	0.86
Member-Collection(e2,e1)	0.83	0.85	0.84
Message-Topic(e1,e2)	0.74	0.78	0.76
Message-Topic(e2,e1)	0.86	0.92	0.89
Other	0.87	0.93	0.90
Product-Producer(e1,e2)	0.86	0.94	0.90
Product-Producer(e2,e1)	0.00	0.00	0.00

FIGURE 6.11: BERT

On observe que le meilleur modèle est une fois de plus le modèle *BERT*. De plus, les résultats obtenus sont inférieurs à ceux trouvés dans la littérature [27] [28] [24], comme dans la partie précédente, un meilleur réglage des hyperparamètres permettrait d'améliorer ces résultats.

6.4.2 Les données brevets

Le tableau 6.2 présente les résultats des différents modèles considérés.

Table 6.2: Résultats pour l'extraction de relation (en moyenne)

Modèle	Section	F1 Score
CNN	6.3.1	0.77
CNN+Attention	6.3.2	0.76
BiLSTM+Attention	6.3.3	0.79
BERT	6.3.4	**0.83**

Les résultats pour chaque relation sont présentés dans les tableaux 6.3, 6.4, 6.5 et 6.6.

Table 6.3: Modèle CNN par relation

Relation	F1 Score	Précision	Rappel
`Contains`	0.63	0.59	0.68
`Has`	0.3	0.23	0.41
`Is_Associated_To`	0.62	0.53	0.75
`Is_Defined`	0.91	0.89	0.92

Table 6.4: Modèle CNN+Attention par relation

Relation	F1 Score	Précision	Rappel
`Contains`	0.92	0.85	1
`Has`	0.73	0.62	0.88
`Is_Associated_To`	0.89	0.86	0.92
`Is_defined`	0.95	0.92	0.98

Table 6.5: Modèle BiLSTM+Attention par relation

Relation	F1 Score	Précision	Rappel
`Contains`	0.9	0.85	0.97
`Has`	0.85	0.74	1
`Is_Associated_To`	0.86	0.8	0.92
`Is_Defined`	0.69	0.95	0.55

Table 6.6: Modèle BERT par relation

Relation	F1 Score	Précision	Rappel
`Contains`	0.75	0.6	1
`Has`	1	1	1
`Is_Associated_To`	1	1	1
`Is_Defined`	0.81	0.9	0.75

Les résultats obtenus sont globalement bons et ont tous un F1 score d'environ 0.8. On remarque que le meilleur modèle est le modèle *BERT*. Néanmoins, le modèle *BiLSTM + Attention* fournit un F1 score assez proche de celui de *BERT* alors qu'il est beaucoup plus simple et moins long à entraîner que *BERT*.

7. Comparaison avec Watson

Dans cette partie, les résultats obtenus par les meilleurs modèles décrits dans les sections précédentes pour les tâches de reconnaissance d'entités nommées et d'extraction de relation sont comparés aux résultats obtenus par la plateforme Watson d'IBM. En effet, des travaux menés à IFPEN autour de l'évaluation de cette plateforme ont été menés en amont de ce stage.

7.1 Reconnaissance d'entités nommées

Pour la tâche de reconnaissance d'entités nommées, le meilleur modèle est le modèle *BERT*. Les tableaux 7.1 et 7.2 représentent les résultats par entités fournis respectivement par le modèle *BERT* et Watson.

Table 7.1: Modèle BERT par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.86	0.87	0.84
LOCUTION	0.83	0.91	0.76
PV_UNIT	0	0	0
PV_VAL_MAX	0.84	0.82	0.87
PV_VAL_MIN	0.84	0.72	1
PORE_VOLUME	0.85	0.74	1
SA_UNIT	0.88	0.79	1
SA_VAL_MAX	0.99	1	0.99
SA_VAL_MIN	0.92	0.84	1
SUPPORT	0.88	0.82	0.95
SURFACE_AREA	0.47	0.81	0.33

Table 7.2: Résultats de Watson par entité

Entité	F1 Score	Précision	Rappel
CATALYST	0.64	0.65	0.63
LOCUTION	0.84	0.85	0.84
PV_UNIT	0.84	0.9	0.79
PV_VAL_MAX	0.95	0.95	0.95
PV_VAL_MIN	0.88	0.9	0.87
PORE_VOLUME	0.96	0.97	0.95
SA_UNIT	0.79	0.85	0.74
SA_VAL_MAX	0.95	0.92	0.98
SA_VAL_MIN	0.92	1	0.84
SUPPORT	0.61	0.71	0.53
SURFACE_AREA	0.84	0.85	0.82

On remarque que pour certaines entités le modèles *BERT* donne de meilleurs résultats que celui de Watson (par exemple, `CATALYST`, `SUPPORT`). Par contre, pour les entités `PV_UNIT`, `SURFACE_AREA` les résultats fournis par Watson sont bien meilleurs que ceux de *BERT*.

7.2 Extraction de relation

Pour la tâche de reconnaissance d'entités nommées, le meilleur modèle est le modèle *BERT*. Les tableaux 7.3 et 7.4 représentent les résultats par entités fournis respectivement par le modèle *BERT* et Watson.

Table 7.3: Modèle BERT par relation

Relation	F1 Score	Précision	Rappel
Contains	0.75	0.6	1
Has	1	1	1
Is_Associated_To	1	1	1
Is_Defined	0.81	0.9	0.75

Table 7.4: Résultats de Watson par relation

Relation	F1 Score	Précision	Rappel
Contains	0.43	0.43	0.43
Has	0.32	0.37	0.29
Is_Associated_To	0.72	0.69	0.74
Is_Defined	0.77	0.79	0.76

Le modèle *BERT* donne de très bons scores pour toutes les relations et qui sont tous bien supérieurs aux résultats fournis par Watson.

7.3 En résumé

Globalement, le résultat à retenir est que les modèles utilisés dans ce stage sont du même ordre de précision voire meilleurs que ceux de la plateforme Watson dont on ignore le type. En particulier, ce résultat était une attente du stage.

8. Conclusion : bilan et perspectives

En conclusion, lors de ce stage, j'ai pu m'intéresser à différentes méthodes de traitement automatique du langage pour l'étude de différentes tâches en utilisant divers jeux de données. Ce stage a été très enrichissant et a été l'opportunité pour moi de développer mes connaissances en traitement du langage, en Deep Learning ainsi qu'en programmation.

L'objectif du stage a été de trouver des méthodes d'apprentissage profond performantes permettant d'extraire de l'information contenue dans des brevets appartenant au domaine de la catalyse hétérogène. Cette extraction d'information se fait actuellement à IFPEN au travers de la plateforme Watson de IBM. Une des ambitions de ce stage a ainsi été de permettre à IFPEN d'obtenir des modèles permettant de reproduire les résultats obtenus sur le corpus de brevets annotés en utilisant Watson. Un des principaux défis de ce stage a justement été de reproduire les résultats de Watson étant donné que son algorithme n'est pas connue.

Le stage a ainsi tout d'abord consisté à faire un état de l'art des différents algorithmes d'apprentissage profond et de Machine Learning pour la classification. Nous avons remarqué que les modèles statistiques classiques sont tout aussi performant que le modèle BERT.

Concernant la tâche de reconnaissance d'entités nommées, les résultats de différents modèles d'apprentissage profond appliqués au jeu de données CoNLL-2003 sont inférieurs à ceux trouvés dans la littérature. Ces modèles ont néanmoins donné de très bons résultats sur le jeu de données de brevets annotés. Le meilleur de ces modèles est le modèle BERT. En comparant les scores par entité de ce modèle avec les résultats de Watson, nous avons remarqué que le modèle BERT était plus performant pour prédire certaines entités et moins performant pour en prédire d'autres.

Finalement, pour l'extraction de relation, sur le jeu de données SemEval-2010 Task 8, bien connu pour cette tâche, les résultats obtenus sont, tout comme pour le jeu de données CoNLL-2003, inférieurs à ceux trouvés dans l'état de l'art. Concernant le jeu de données de brevets, le meilleur modèle BERT fournit des résultats bien supérieurs aux résultats de Watson. Néanmoins d'autre modèles plus simples et moins longs à entraîner comme les modèles CNN+Attention et BiLSTM+Attention donnent également des résultats supérieurs à ceux de Watson.

Lors de ce stage un des objectifs a également été d'évaluer les performances des modèles les plus récents comme les Transformers et BERT comparées aux performances de modèles plus classiques. On a ainsi remarqué que le modèle BERT est le modèle le plus performant pour les trois tâches décrites plus haut. Par contre, le modèle Transformers nous donne des résultats assez faible, ce qui nous montre l'importance du pré-entraînement de BERT.

Une perspective envisagée et suite possible à donner à ces travaux serait le réglage des différents hyperparamètres des modèles utilisés pour la reconnaissance d'entité nommées et l'extraction de relation sur les données CoNLL-2003 et SemEval-2010 Task 8 pour retrouver l'état de l'art.

Bibliographie

[1] 2 - updated sentiment analysis.

[2] The complete beginner's guide to deep learning : Convolutional neural networks and image classification.

[3] Comprendre le fonctionnement d'un lstm et d'un gru en schémas.

[4] Glove : Global vectors for word representation. https ://nlp.stanford.edu/projects/glove/.

[5] Hugging face. https ://huggingface.co/.

[6] Les réseaux de neurones récurrents : des rnn simples aux lstm. https ://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/.

[7] Named entity recognition on conll 2003 (english).

[8] Understanding convolutional neural networks for nlp.

[9] Understanding neural networks : Bi-lstm explained in detail. https ://valiancesolutions.com/bi-lstm-explained/.

[10] Abien Fred M. Agarap. Deep learning using rectified linear units (relu). February 2019.

[11] Paolo Torroni Andrea Galassi, Marco Lippi. Attention in natural language processing. February 2019.

[12] Piotr Bojanowski Tomas Mikolov Armand Joulin, Edouard Grave. Bag of tricks for efficient text classification. August 2016.

[13] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Łukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. December 2017.

[14] Kevin Gimpel Dan Hendrycks. Gaussian error linear units (gelus). November 2018.

[15] Paul Denoyes. Bert : Le transformer model qui s'entraîne et qui représente. `https://lesdieuxducode.com/blog/2019/4/bert--le-transformer-model-qui-sentraine-et-qui-represente`.

[16] Jimmy Lei Ba Diederik P. Kingma. Adam : A method for stochastic optimization. January 2017.

[17] KyungHyun Cho Yoshua Bengio Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. May 2016.

[18] Fien De Meulder Erik F. Tjong Kim Sang. Introduction to the conll-2003 shared task :language-independent named entity recognition. June 2020.

[19] Zornitsa Kozareva Preslav Nakov Diarmuid O Seaghdha Sebastian Pado Marco Pennacchiotti Lorenza Romano Stan Szpakowicz Iris Hendrickx, Su Nam Kim. Semeval-2010 task 8 : Multi-way classification of semantic relations between pairs of nominals. July 2010.

[20] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert : Pre-training of deep bidirectional transformers for language understanding. October 2018.

[21] Hao Wu JQimin Zhou, Zhengxin Zhang. Nlp at iest 2018 : Bilstm-attention and lstm-attention via soft voting in emotion classification. October 2018.

[22] Mitchell P. Marcus Lance A. Ramshaw. Text chunking using transformation-based learning. 1995.

[23] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[24] Jun Tian Zhenyu Qi Bingchen Li Hongwei Hao Bo Xu Peng Zhou, Wei Shi. Attention-based bidirectional long short-term memory networks for relation classification. August 2016.

[25] Jurgen Schmidhuber Sepp Hochreiter. Long short-term memory. December 1997.

[26] Yifan He Shanchan Wu. Enriching pre-trained language model with entity information for relation classification. May 2019.

[27] Ralph Grishman Thien Huu Nguyen. Relation extraction : Perspective from convolutional neural networks. June 2015.

[28] Xuanjing Huang Yatian Shen. Attention-based convolutional neural network for semantic relation extraction. December 2016.

[29] Mert R. Sabuncu Zhilu Zhang. Generalized cross entropy loss for training deep neural networks with noisy labels. May 2018.