Apprentissage par transfert pour l'extraction de relations pharmacogénomiques à partir de textes

Walid Hafiane

Université de Lorraine — Laboratoire LORIA

Rapport de stage

présentée pour obtenir le titre de Master 2 en
Informatique spécialité AVR – Apprentissage, Vision, Robotique

Apprentissage par transfert pour l'extraction de relations pharmacogénomiques à partir de textes

Walid Hafiane
hafianewalid1@gmail.com

Soutenue publiquement le 09/09/2020

Encadrants Adrien Coulet
Yannick Toussaint
Joël Legrand

Tuteur Bernard Girau

Résumé

L'extraction des relations entre les entités nommées est une tâche primordiale pour la fouille de textes, notamment dans le domaine biomédicale où cette tâche permet de synthétiser les connaissances disponibles dans les nombreuses publications du domaine. Récemment, les approches de l'apprentissage profond ont considérablement amélioré les performances d'extraction de relation. Cependant, la complexité des données biomédicales met en défi ce type d'approche. Les architectures actuelles ne répondent pas totalement à la problématique de l'extraction des relations particulières (i.e., pharmacogénomiques, protéine-molécule). Dans ce travail, nous proposons des architectures qui apportent une amélioration significative à cette problématique. D'autre part, l'obtention de nombreuses données annotées dans le domaine biomédical est coûteuse, pour pallier ce manque de données nous avons utilisé des méthodes d'apprentissage par transfert. Dans ce cadre, nous avons opté pour deux stratégies d'apprentissage par transfert *frozen* et *fine-tuning*. Notre architecture BERT-CNN-Segmentation avec la stratégie *fine-tuning* dépasse l'état de l'art dans les deux corpus biomédicaux références avec **32.77 %** amélioration absolue en F-macro sur PGxCorpus et **1.73%** amélioration absolue en F-micro sur le corpus ChemProt. Ces résultats montrent l'utilité de l'apprentissage par transfert et l'amélioration des performances du transformateur BERT à travers l'exploitation de l'information locale latente dans les vecteurs de représentation en renforçant cette information par l'information structurale issue de la segmentation des phrases.

mots clés : fouille de données ; fouille de textes ; apprentissage automatique ; l'apprentissage profond ; l'apprentissage par transfert ; traitement automatique du langage naturel ; extraction de relations ; représentation des connaissances ; biomédical

Abstract

The extraction of relationships between named entities is a required task for text mining, particularly in the biomedical field, this task allows to synthesize the available knowledge in a lot of publications. Recently, deep learning approaches have significantly improved performance in relation extraction task. However, the biomedical data complexity is a major challenge facing this approach. Current architectures do not fully perform in particular relation extraction (i.e., pharmacogenomics, protein-molecule). In this work, we propose architectures that bring in a significant improvement to this task. On the other hand, obtaining a large amount of annotated data in the biomedical field is challenging and expensive, in attempt to cover this lack of data transfer learning is used. In this context, we have opted for two transfer learning strategies : frozen and fine-tuning. Our BERT-CNN-segmentation architecture with the fine-tuning strategy achieve new state-of-the-art results on two benchmark biomedical corpora with **32.77 %** absolute improvement in F-macro on PGxCorpus and **1.73%** absolute improvement in F-micro on the ChemProt. These results show the usefulness of transfer learning and the improved performance of the BERT transformers through the exploitation of local latent information in the representation vectors by reinforcing this information with the structural information resulting from sentence segmentation.

Keywords : data mining ; text mining ; machine learning ; deep learning ; transfer learning ; natural language processing ; relation extraction ; knowledge representation ; biomedical

Remerciements

Le domaine de l'informatique m'a toujours passionné, plus précisément l'intelligence artificielle. Mes professeurs, mes encadrants et ma famille m'ont toujours encouragé dans cette voie. Ce qui m'a donné la motivation pour surmonter les difficultés et pour dépasser mes limites. Ce travail est l'aboutissement d'un long cheminement au cours duquel j'ai bénéficié de l'encadrement attentif et du soutien constant de plusieurs personnes, à qui je tiens à dire profondément et sincèrement merci.

Je tiens à remercier toutes les personnes qui ont contribué à ma formation académique et professionnelle tout au long de mon parcours universitaire, à l'université de Batna 2 puis à l'université de Lorraine. Grâce à ces professeurs, j'ai approfondi mes connaissances en informatique et découvert beaucoup de ses applications, notamment dans le domaine de l'intelligence artificielle. En outre, ils m'ont aidé à m'orienter dans mes projets professionnels.

Je suis également très reconnaissant envers l'équipe Orpailleur, au Loria, pour l'accueil chaleureux notamment mes encadrants, messieurs Adrien Coulet, Yannick Toussaint et Joël Legrand, pour le temps et la confiance qu'ils m'ont accordés ainsi que pour leurs précieux conseils. Je remercie également toutes les personnes qui m'ont conseillé et qui ont assuré la relecture lors de la rédaction de ce rapport de stage.

Je remercie mes très chers parents, ma famille et mes amis pour leur présence et leur soutien inconditionnel.

Un proverbe dit " *un voyage de mille lieues commence toujours par un premier pas* ".

Table des matières

Table des figures

Liste des tableaux

Abréviations et notations

TAL : Traitement Automatique des Langues
RNN : *Recurrent Neural Network*
CNN : *Convolutional Neural Network*
MCCN : *Multichannel Convolutional Neural Network*
RGB : *Red, Green, Blue*
FFW : *FeedForWard neural network*
MLP : *MultiLayer Perceptron*
LSTM : *Long Short-Term Memory*
GRU : *Gated Recurrent Unit*
BERT : *Bidirectional Encoder Representations from Transformers*
SciBERT : *A Pretrained Language Model for Scientific Text*
BioBERT : *A pre-trained biomedical language representation model for biomedical text mining*
ALBERT : *A Lite BERT for Self-supervised Learning of Language Representations*
DistilBERT : *A distilled version of BERT: smaller, faster, cheaper and lighter*
ELMo : *Embeddings from Language Models*
OpenAI GPT : *Improving Language Understanding by Generative Pre-Training*
GloVe : *Global Vectors for Word Representation*
OAR : Un gestionnaire de ressources et de tâches polyvalent
GPU : *Graphics Processing Unit*
CPU : *Central Processing Unit*
TPU : *Tensor Processing Unit*
SOTA : *State of the art*
σ : L'écart-type
μ : La moyenne
$\mathbb{R}$: Ensemble de nombres réels
$\mathbb{R}^n$: Ensemble de vecteurs colonnes réels de taille n

Introduction générale

Le projet ANR PractiKPharma (2016-2020) vise à étudier l'extraction, la gestion et la comparaison des connaissances pharmacogénomiques. L'un des piliers de ce projet est de proposer des outils pour l'étude de l'influence des gènes sur la réponse aux médicaments. Cela nécessite un modèle d'extraction des relations pharmacogénétiques performant. Le sujet de ce stage porte sur la tâche d'extraction des relations pharmacogénomiques en étudiant l'architecture du transformateur BERT et ses variantes, ainsi que le développement d'architectures basées sur ce transformateur dans le but d'améliorer significativement les performances d'extraction des relations sur le corpus biomédical PGxCorpus par rapport au premier modèle proposé dans l'article Legrand et al.[15]. Pour pallier le problème de la faible quantité de données biomédicales annotées disponibles, nous avons opté pour l'utilisation des méthodes d'apprentissage par transfert.

Dans ce document nous présentons l'environnement, le contexte scientifique ainsi que la problématique qui sera abordée dans le premier chapitre. Dans le chapitre État de l'art, nous détaillerons le transformateur BERT (et ses variantes) préconisé dans mon cahier des charges ainsi que les mécanismes d'attention et les différentes types et stratégies d'apprentissage par transfert. Dans le chapitre Méthode, pour répondre à notre problématique, nous proposons des architectures basées sur les architectures déjà abordées dans l'état de l'art en utilisant aussi l'apprentissage par transfert. Dans le dernier chapitre nous proposons une analyse et des discussions sur les résultats quant à l'amélioration et la stabilité obtenues par nos modèles.

Chapitre 1

Contexte

Sommaire

1.1 Environnement de stage

1.1.1 Laboratoire d'accueil

Le Loria (laboratoire lorrain de recherche en informatique et ses applications) a été créé en 1997. Les missions du Loria concernent principalement la recherche fondamentale et appliquée en informatique. Ce laboratoire est impliqué dans la démarche du transfert de la technologie du laboratoire à l'industrie à travers des collaborations nationales et internationales. Le Loria est structuré en cinq départements, chacun traitant une thématique particulière : « Algorithmique, calcul, image et géométrie », « Méthodes formelles », « Réseaux, systèmes et services », « Traitement automatique des langues et des connaissances », « Systèmes complexes, intelligence artificielle et robotique ».

1.1.2 Équipe de travail

J'ai réalisé ce stage dans le département « Traitement automatique des langues et des connaissances » au sein de l'équipe Orpailleur. Les thématiques de recherches abordées dans cette équipe portent sur l'extraction et la représentation de connaissances. Ces recherches débouchent sur des applications dans les domaines des sciences de la vie et de la biologie comme par exemple le domaine médical et pharmaceutique.

1.1.3 Méthodologie de travail

Ce stage s'est déroulé en télétravail à cause du confinement dû à l'épidémie de COVID-19. Malgré les contraintes du confinement, nous avons réussi à maintenir un travail d'équipe par l'organisation de réunions bihebdomadaire jusqu'au 15/5/2020, puis hebdomadaire où j'ai présenté plusieurs articles et technologies, ainsi que des démonstrations techniques et des discussions autour des résultats obtenus. J'ai bénéficié également du séminaire MALOTEC et d'une présentation de l'instrument de calcul Grid5000, ce qui m'a permis de m'intégrer rapidement dans l'équipe et de m'adapter aux outils utilisés au sein de l'équipe. En plus de la communication à l'intérieur de l'équipe j'ai également contacté quelques auteurs d'articles pour leur poser des questions et demander des précisions sur leurs articles.

La méthode de travail s'est déroulée en plusieurs étapes. La première était consacrée à l'étude de l'état de l'art afin de déceler les méthodes les plus intéressantes pour notre problématique. Puis, en deuxième temps, vient l'évaluation et les tests des techniques étudiées. La troisième étape a été consacrée à l'amélioration des méthodes existantes en proposant de nouvelles variantes. Enfin, la dernière étape, a été dédiée à l'étude des résultats et à la rédaction de ce rapport.

1.2 Environnement technique

En ce qui concerne l'environnement et les logiciels utilisés dans ce stage : python3 a servi de langage de programmation et nous avons utilisé diverses bibliothèques par exemple, pytorch et scikit-learn pour implémenter les modèles de *deep learning*, et pandas

pour calculer des statistiques sur les corpus et les résultats. En ce qui concerne l'environnement matériel nous avons utilisé Grid5000 qui représente un instrument de calcul distribué sur plusieurs sites, chaque site étant composé des clusters de calcul.
La réservation dans cet instrument se fait par l'ordonnanceur(scheduler) OAR. L'accès au site et job réservé se fait avec une connexion ssh. Pour faciliter la réservation et la réalisation des expériences nous avons implémenté un script de réservation avec une interface graphique cf ; la figure(A1). Avec Grid5000 en peut réserver des CPU(il y en a beaucoup de disponibles) ou des GPU(pas toujours disponibles) et nous avons utilisé les deux selon nos besoins et leurs disponibilités. On note aussi que Grid5000 possède un système d'observation avancé qui permet de reporter des informations détaillées sur les machines utilisées. Nous avons également utilisé GitLab pour héberger et gérer le développement des différents systèmes d'apprentissage automatique.

https://gitlab.inria.fr/whafiane/bert_finetuning_cnn

1.3 L'enjeu d'extraction des relations

1.3.1 Tache d'extraction des relations

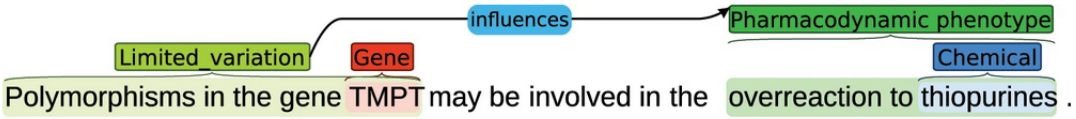

FIGURE 1.1 – Exemple d'une relation pharmacogénomique dans PGxCorpus

Les entités nommées sont des séquences de mots qui peuvent appartenir au domaine général (noms de personnes, d'organisations, de lieux, d'horaires, de dates ...) et au domaine biomédical (gènes, maladie, protéines, ...). La sémantique dans la phrase est généralement représentée par des relations entre des entités nommées.

Dans le cas supervisé avec des relations binaires on peut formuler la tâche d'extraction de relations comme une tâche de classification où l'entrée est définie par le texte et les deux entités nommées. La relation est représentée par une étiquette (la classe) associée à cette entrée. La figure(1.1) présente un exemple d'une relation de type (*influences*) tiré du corpus PGxCorpus.

1.3.2 Contexte scientifique

Le texte brut porte une information non structurée, l'extraction des relations est une tâche qui permet de représenter l'information dans le texte d'une manière structurée sous forme d'entités nommées et de relations liant ces entités, ce qui la transforme en un texte structuré exploitable. Dans le domaine général cette tâche sert de prétraitement à des tâches plus compliquées telles que la réponse aux questions, la traduction, etc. Dans le domaine biomédical cette tâche est utilisée généralement dans la fouille de données (*data*

mining). Avant d'extraire des connaissances présentes dans les données, on les structure à l'aide de l'extraction de relations. Ces connaissances dans le domaine biomédical sont représentées sous forme d'interactions (de causalité ou d'influence ou d'indépendances etc.) entre des entités nommées (gènes, maladie, protéines, ...). Le corpus PGxCorpus a été développé dans le projet ANR PractiKPharma qui vise à étudier l'extraction, la gestion et la comparaison des connaissances pharmacogénomiques. Comme le montre la figure(A2) PractiKPharma utilise plusieurs types de connaissances où PGxCorpus contribue à l'obtention de connaissances sur la façon dont la génétique influe sur la réponse aux médicaments.

1.3.3 Problématique

L'extraction des relations est une tâche primordiale dans la fouille de textes biomédicales. L'amélioration des performances des modèles d'extraction des relations pharmacogénomiques conduit forcément à des améliorations dans l'extraction de connaissances dans la fouille des données médicales à partir de textes structurés, ce qui permet d'étudier les relations pharmacogénomiques (l'influence des gènes sur la réponse aux médicaments). Les méthodes d'apprentissage profond et notamment les transformateurs représentent des méthodes performantes mais qui nécessitent une grande quantité de données. En revanche l'annotation manuelle des corpus pharmacogénomiques est coûteuse (elle nécessite des experts du domaine) ce qui limite la quantité de données. L'architecture du transformateur BERT et ses variantes permet-elle d'aboutir à des performances meilleures que celle du modèle de base proposé dans l'article [15] ? Comment améliorer l'architecture du transformateur dans la tâche d'extraction sur PGxCorpus et sur d'autres corpus biomédicaux ? Quels sont les types et les stratégies d'apprentissage par transfert qui permettent de pallier le problème de la faible quantité des données disponibles sur les corpus du domaine biomédical, notamment sur PGxCorpus (environ un millier de phrases annotées) ? Dans les chapitres suivant nous apportons des réponses à ces questions.

Chapitre 2

État de l'art

Sommaire

Aujourd'hui, dans le domaine du traitement automatique des langues (TAL) nul ne peut ignorer la dernière innovation développée par Google, le transformateur BERT[7] (*Bidirectional Encoder Representations from Transformers*). L'amélioration significative qu'il a apportée dans les performances de la plupart des tâches de TAL permet à BERT de s'imposer dans beaucoup de domaines et d'applications du TAL. BERT atteint ce haut niveau de performance grâce à l'utilisation des différentes techniques de pointe dans le domaine de l'apprentissage profond combinées avec une architecture de transformateur originale (le premier transformateur qui capture le contexte bidirectionnel) sans oublier le mécanisme d'attention et l'apprentissage par transfert qui constituent les piliers du succès de BERT. Dans cette section nous exposerons les différents composants de BERT et nous détaillerons la façon dont ils sont combinés pour composer le modèle BERT. Nous présenterons aussi les différentes variantes de BERT et expliquerons comment ces dernières s'appuient sur BERT pour atteindre des niveaux de performances plus élevés que celles de l'état de l'art pour de nombreuses tâches de TAL, tant dans le domaine général que dans le domaine biomédical.
Si l'on s'intéresse plus particulièrement à la tâche d'extraction de relations à partir de textes, avant l'apparition du *deep learning* on utilisait des méthodes symboliques (par exemple la co-occurence d'entités nommées, l'écriture de règles à l'aide d'éléments syntaxiques), numériques (par exemple les *Conditional Random Fields* ou les réseaux de neurones). Puis est arrivé l'apprentissage profond avec son impressionnante capacité d'apprentissage et de généralisation, l'apprentissage profond détrône en terme de performance les approches précédentes[6]. Avant les transformateurs, les réseaux de neurones récurrents (RNN) et les réseaux de neurones convolutionnels (CNN) étaient les architectures d'apprentissage profond les plus utilisées.

RNN
Les RNN sont des architectures qui peuvent traiter l'information contextuelle à travers des liens de récurrence et peuvent par cette méthode mémoriser l'information dans les séquences. LSTM (*Long short-term memory*) et GRU (*Gated recurrent unit*) sont les types de RNN les plus standards grâce à leur gestion du flux par des portes, procédé qui permet aux LSTM et GRU de pallier le problème de la disparition du gradient (*vanishing gradient*) et d'augmenter ainsi la longueur des séquences mémorisées. Les RNN ont la capacité de capturer le contexte, ce qui présente un avantage dans le domaine de traitement automatique des langues : les blocs LSTM peuvent être utilisés séquentiellement sous forme d'une chaîne ou sous forme d'un arbre de LSTM[36]qui forme un réseau (les liens entre les blocs LSTM dépendent des entrées). Les arbres LSTM peuvent notamment servir à apprendre des structures sous forme d'arbre comme par exemple les graphes de dépendances qui sont des arbres binaires qui représente la structure lexicale d'une phrase : chaque noeud est un mot, chaque arc est une dépendance syntaxique entre les mots. Pour obtenir l'arbre de dépendance on a souvent besoin d'un parseur qui définit les liens de dépendance entre les mots dans la phrase. La structure des LSTM correspond à cet arbre (le stanford-parser[22] est l'un des parseurs les plus utilisés). En ce qui concerne l'extraction de relation, Legrand et al.[16] a utilisé une architecture d'arbre LSTM[36] sur des corpus biomédicaux de référence, SNPPhena [3] par exemple avec 65,5 % F-macro. L'ar-

ticle Geng et al.[8] propose une architecture parallèle composée d'une LSTM séquentielle et d'un arbre LSTM qui sert à capturer les caractéristiques structurelles ; la partie LSTM séquentielle est suivie par l'attention et fournit une information sur la pertinence entre les mots à la deuxième partie (l'arbre LSTM). Dans cet article l'arbre de dépendance est calculé par « stanford parser », cette architecture obtient de relativement bonnes performances avec le corpus référence SemEval[9] (87,1% F-macro). Une autre architecture [19] a exploré la segmentation de la phrase en 5 parties, (une première partie avant la première entité nommée, la 1ère entité nommée, une partie entre les deux entités, la seconde entité nommée, la partie après la seconde entité nommée). Les segments sont traités séparément avec une séquence d'LSTM accompagnée du mécanisme d'attention(2.1).

CNN

Les architectures CNN sont largement utilisées dans le domaine de vision par ordinateur. Le traitement automatique des langues les a adoptées en raison de leur grande capacité d'extraction des caractéristiques locales à l'aide des filtres de convolution. L'extraction des caractéristiques s'opère hiérarchiquement, partant des caractéristiques très locales et très simples, apprises dans les premières couches et allant jusqu'à des caractéristiques compliquées apprises dans les couches profondes. Le CNN présente aussi l'avantage de posséder un nombre réduit de paramètres grâce au partage des paramètres (les filtres de convolution). MCCN (multi channel CNN) [30] est un modèle qui traite indépendamment plusieurs canaux, chacun important une information différente. Dans le domaine de vision, les couleurs RGB par exemple sont souvent utilisées comme canaux. Dans le domaine du traitement automatique des langues les canaux peuvent être des différents plongements. À propos de l'extraction des relations utilisant des architectures base CNN, de nombreux articles ont abordé l'extraction des relations à l'aide d'un CNN, par exemple Legrand et al.[16] qui utilise un MCNN[30] où les canaux sont des différents plongements, ce modèle atteint 63,3 % F-macro sur SNPPhena[3]. L'article [15] aussi utilise l'architecture MCNN[30], ce papier atteint **45.67%** F-macro sur le corpus biomédicale ***PGxCorpus***[15]. Un autre article [5] a exploré l'utilisation d'une segmentation de la phrase (une première partie, la première entité nommée, une partie entre les deux entités, la deuxième entité nommée, une dernière partie) chaque segment correspondant à un canal. Cette segmentation renforce la capacité du modèle en fournissant l'information structurelle. Cette méthode présente des performances relativement bonnes avec le corpus référence SemEval[9] avec 84,56 % F-macro. Un autre article [25] a adopté une architecture mixte qui utilise en parallèle CNN et RNN, la partie RNN permet d'extraire des vecteurs de représentation en exploitant l'information contextuelle à travers un BiLSTM combiné avec les caractéristiques locales extraites par un CNN, un plongement de position relative fournit une information structurelle au modèle et le mécanisme d'attention (2.1) a été également utilisé avec un LSTM pour mieux capturer le contexte. L'article [25] définit l'état de l'art sur le corpus référence SNPPhena[3] avec 73,97 % F-macro.

2.1 Mécanisme d'attention

Ce concept simple mais puissant a révolutionné le domaine de l'apprentissage automatique. En introduisant une capacité de focalisation sur l'information pertinente il a conduit à une amélioration significative non seulement dans les tâches de TAL, mais aussi dans le traitement d'image, de la reconnaissance vocale, etc.

2.1.1 Séquence à séquence

L'attention a été introduite la première fois pour améliorer la tâche de traduction [35], en utilisant l'architecture séquence à séquence avec des réseaux de neurones récurrents(RNN). Mais cette architectures RNN est mal adaptée aux longues phrases à cause du problème que pose la disparition ou l'explosion du gradient (*vanishing gradient*). Certes les LSTM et GRU sont plus adaptés aux longues phrases que le RNN simple, mais le problème des longues phrases demeure. Une architecture séquence à séquence est une architecture composée d'une partie encodeur et d'une partie décodeur. L'encodeur prend une séquence en entrée et génère un vecteur contextuel en sortie (le dernier vecteur d'état). Ce vecteur résume en quelque sorte la séquence en entrée, puis il deviendra l'entrée du décodeur qui va générer la séquence en sortie au long des itérations : dans chaque itération le décodeur utilise le vecteur de contexte et le vecteur de l'état précédent pour générer la sortie, associée à l'itération en question. Dans le cas de séquence à séquence avec l'attention, le mécanisme d'attention est appliqué dans la phase de décodage, où une fonction de similarité (un produit scalaire est utilisé souvent comme une fonction de similarité) est appliquée entre les vecteurs de contexte et les vecteurs d'état de la phase d'encodage. Cette fonction de similarité fournit des scores de similarité associés aux vecteurs d'état d'encodeur. Ces scores seront normalisés par une fonction softmax afin d'obtenir un vecteur d'attention, ce dernier constituant avec le vecteur de contexte les entrées du décodeur. Un vecteur d'attention est calculé à chaque itération de décodage afin d'obtenir le vecteur de contexte accordé à cette itération. À chaque itération le mécanisme d'attention permet donc d'avoir un vecteur de contexte qui représente un résumé sélectif de la séquence d'entrée. Plus un vecteur d'état est similaire au vecteur de contexte associé à cette itération, plus il contribue au vecteur en sortie du décodeur.

2.1.2 L'auto-attention à plusieurs têtes

L'auto-attention multi-têtes est un moyen de calculer la pertinence d'un ensemble de valeurs en fonction de certaines clés et requêtes selon plusieurs têtes. Le mécanisme d'attention est utilisé pour permettre au modèle de se concentrer sur des informations pertinentes en fonction de ce qu'il traite actuellement, cette focalisation peut porter sur plusieurs caractéristiques différentes, ce qui nécessite de calculer l'attention selon plusieurs têtes parallèles. Chaque tête réalise une focalisation différente, ce qui permet au transformateur de calculer diverses représentations à travers différentes transformations linéaires, et donc de capturer plusieurs aspects sur les entrées.

Les vecteurs d'attention

La première étape dans l'attention multi-têtes consiste à calculer trois transformations sur chaque vecteur de représentation en entrée. Ces transformations linéaires sont calculées à partir des multiplications de vecteurs de représentation avec trois matrices, ces matrices étant des poids appris dans la phase d'entraînement comme les autres poids dans le réseau. Les trois vecteurs sont donc calculés par les équations (2.1,2.2,2.3) tel que X représente les vecteurs des représentation en entrée, Q représente la requête (query), K représente la clé (key), V représente la valeur (value), W_q,W_k,W_v représentent les matrices associées respectivement aux requêtes, clés, valeurs.

$$Q = X.W_q \tag{2.1}$$

$$K = X.W_k \tag{2.2}$$

$$V = X.W_v \tag{2.3}$$

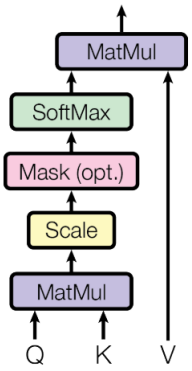

FIGURE 2.1 – Produit scalaire dans l'attention [37]

Attention selon une tête

Mathématiquement parlant, le produit scalaire entre deux vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ est défini par (2.4). Ce produit reflète le degré d'alignement des deux vecteurs, plus le produit scalaire entre deux vecteurs est grand plus les vecteurs sont alignés et donc similaires. Dans l'article [37] le produit scalaire est utilisé comme une fonction de similarité qui sert à calculer les scores de similarité entre les vecteurs requêtes et clés comme l'illustre la figure (3.1). La dimension des vecteurs utilisés impacte la plage des valeurs, une mis à l'échelle est donc appliquée sur le produit scalaire en divisant ce produit par la racine de la dimension, comme le montrent l'équation (2.6) et la figure (3.1). Ensuite la distribution des scores est normalisée par une fonction softmax (softmax est une fonction qui transforme

un vecteur z avec des valeurs réelles arbitraires à un vecteur réel dont les valeurs se situent entre 0 et 1. (2.5) illustre cette fonction telle que z représente le vecteur en entrée avec des valeurs réelles arbitraires et C représente la dimension de ce dernier). Le résultat de cette normalisation est le vecteur d'attention, ce vecteur sera par la suite multiplié par le vecteur des valeurs pour générer le vecteur de représentation, ce calcul est présenté par la figure (3.1). Ces opérations d'attention permettent aux différents sous-mots de contribuer à la représentation de chaque sous-mot selon la similitude entre le sous-mot en question et les autres sous-mots (l'attention est calculée entre la clé de chaque sous-mot et la requête du sous-mot dont on est en train de générer la représentation). Soit deux sous-mots A et B. Plus le sous-mot A est similaire à un sous-mot B, plus il contribue à la représentation de B, ce qui permet au transformateur de capturer des relations de longueur arbitraire entre les mots dans la phrase contrairement au RNN.

$$\overrightarrow{u} \cdot \overrightarrow{v} = \|\overrightarrow{u}\| \cdot \|\overrightarrow{v}\| \, cos(\theta) \tag{2.4}$$

$$\varsigma(\mathbf{z})_c = \frac{e^{z_c}}{\sum_{d=1}^{C} e^{z_d}} \quad \text{pour } c = 1 \cdots C \tag{2.5}$$

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.6}$$

L'attention multi têtes

Pour apprendre diverses représentations, l'attention multi-têtes applique des transformations linéaires aux vecteurs, clés et requêtes pour chaque tête d'attention. Ceci est illustré dans la figure (3.4). Une fois que les vecteurs de représentation sont calculés selon les différentes têtes, ces vecteurs seront concaténés pour construire un vecteur de représentation. Une transformation linéaire est appliquée sur ce dernier pour réduire sa dimension, comme le montrent l'équation (2.7,2.8),W^O représente les poids de la couche linéaire. Les têtes étant indépendantes, le calcul peut être réalisé en parallèle. Ce gain de temps est un avantages appréciable de cette méthode.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.7}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.8}$$

2.2 Apprentissage par transfert

Définitions :

Domaine

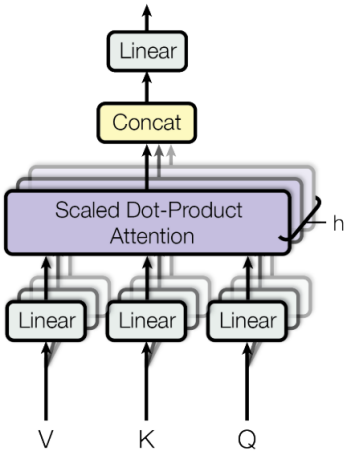

FIGURE 2.2 – Attention multi-têtes [37]

Un domaine, D, est défini par un espace de caractéristiques χ, et d'une probabilité marginale, $P(X)$,X est un échantillon. Donc un domaine est représenté mathématiquement comme D =$\{\chi$,$P(X)$ $\}$

Tâche

Étant donné un domaine $D = \chi, P(X)$, une tâche T définie par un espace d'étiquette γ et une fonction de prédiction $f : \chi \rightarrow \gamma$ apprise à partir des données. Dans le cas supervisé on note la base d'apprentissage par D : $D = (x_i, y_i) \in \chi \times \gamma : i \in (1, .., n)$ et la fonction prédictive $f(.)$ qui est équivalente a $P(y|x)$. D_s et D_t représentent respectivement le domaine source et cible. T_s et T_t représentent respectivement la tâche source et la tâche cible.

Cas de transfert

1. $\chi_s \neq \chi_t$

C'est le cas lorsque les espaces de caractéristiques des domaines sont différents, par exemple des langues différentes (adaptation multilingue).

2. $P(X_s) \neq P(X_t)$

Les espaces des caractéristiques sont les mêmes mais les distributions de données sont différentes. Ce cas correspond à l'adaptation de domaines où les tâches sources et cibles sont les mêmes ($T_s = T_t$) alors que les données proviennent de domaines différents (différentes corpus).

3. $P(Y_s|X_s) \neq P(Y_t|X_t)$

Les distributions source et cible partagent les mêmes étiquettes $Y_t = Y_s$, mais la probabilité conditionnelle de ces classes est différente, donc la répartition de ces classes l'est aussi.

4. $P(Y_s) \neq P(Y_t)$

Les espaces des étiquettes sont différents $Y_s \neq Y_t$, et les tâches sont également diffé-

rentes $T_s \neq T_t$. Il est extrêmement rare de trouver deux tâches différentes avec la même probabilité conditionnelle.

2.2.1 Types d'apprentissage par transfert

Comme c'est évoqué dans l'article [26] l'apprentissage par transfert supervisé se divise en deux grandes catégories : le transfert inductif et le transfert transductif. L'apprentissage par transfert inductif consiste à transférer l'information entre des tâches différentes mais dans le même domaine. Quant à l'apprentissage par transfert transductif, il consiste à transférer de l'information entre des tâches similaires alors que les domaines correspondants sont différents (différentes probabilités marginales). Deux types de transfert transductif (adaptation de domaine, confusion de domaine) et un type de transfert inductif (l'apprentissage multi-tâche) seront évoqués plus loin.

2.2.1.1 Adaptation de domaine

L'adaptation de domaine est généralement évoquée dans le cas où les probabilités marginales entre les domaines source et cible sont différentes $P(X_s) \neq P(X_t)$. Il y a un décalage ou une dérive dans la distribution des données des domaines source et cible qui nécessite des ajustements pour transférer l'apprentissage. Par exemple, la tâche d'extraction des relations dans le domaine biomédical peut être une spécification (adaptation) de l'extraction de relations dans le domaine général. Bien que la distribution dans les deux domaines soit différente, le domaine général (domaine source) peut contribuer à l'apprentissage de la fonction prédictive dans le domaine biomédical (domaine source) via le partage de l'information commune entre les deux domaines. Plus précisément, si on utilise des réseaux de neurones pour apprendre la fonction prédictive (f), les différentes couches d'un réseau d'apprentissage profond capturent différentes caractéristiques, allant de la plus simple à la plus complexe, ce qui permet à un modèle d'apprendre des caractéristiques invariantes au domaine et d'améliorer ainsi leur transférabilité d'un domaine a l'autre.

2.2.1.2 Confusion de domaine

L'application de l'adaptation de domaine nécessite une certaine similarité entre les données sources et cibles. Cette similarité n'est pas toujours présente directement dans les caractéristiques des deux domaines si bien que le modèle est incapable de transférer l'information commune entre les deux domaines à cause de la dissemblance entres les deux distributions. Dans le but d'éviter le transfert négatif, les méthodes de confusion de domaine imposent que les caractéristiques dans les domaines source et cible soient assez similaires pour permettre de transférer l'information, cela peut être réalisé en appliquant certaines étapes de prétraitement sur les représentations. Une méthode d'alignement des caractéristiques a été proposée dans l'article Sun et al. [34]. Cette technique consiste à ajouter un autre objectif au modèle source pour encourager la similarité en confondant les domaines.

2.2.1.3 Apprentissage multi-tâche

Dans le cas de l'apprentissage multi-tâche, plusieurs tâches sont apprises simultanément sans distinction entre la source et les cibles. Le modèle apprend des caractéristiques communes entre les tâches et d'autres caractéristiques apprises sont spécifiques à chaque tâche, il y a donc un co-transfert entre les tâches. Toutes les tâches contribuent au partage de l'information entre elles, ce qui donne au modèle une meilleure capacité de généralisation qui peut améliorer les performances dans toutes les tâches. Dans l'apprentissage multi-tâche deux types d'information sont transférés entre les tâches, l'information commune présente dans les données (les domaines) des différentes tâches, et l'information apprise dans la tâche elle-même. Autrement dit, l'apprentissage simultané de certaines tâches contribue à l'acquisition de l'information partagée entre elles, ce que ne permet pas le modèle uni-tâche.

2.2.2 Stratégies d'apprentissage par transfert

2.2.2.1 *Fine tuning*

Cette stratégie consiste à récupérer un modèle pré-entraîné ou une partie d'un modèle pré-entraîné et de l'utiliser comme un modèle initial en ajoutant un nombre réduit de paramètres (couches) et en reprenant l'apprentissage. Le modèle source (pré-entraîné) aide ainsi le modèle cible à ajuster les poids en fournissant une bonne initialisation des poids. Généralement les modèles pré-entraînés sont entraînés sur plusieurs tâches et sur une grande quantité de données, le modèle transféré a donc déjà une bonne capacité de représentation et ne nécessite que peu de données dans le domaine cible et moins de temps d'apprentissage pour adapter le modèle pré-entraîné à la tâche cible. Cette stratégie est largement utilisée en TAL avec les transformateurs (BERT[7] par exemple).

2.2.2.2 *Frozen*

Les modèles d'apprentissage profond sont des architectures en couches qui apprennent différentes caractéristiques à différents niveaux (représentations hiérarchiques de caractéristiques en couches). Ces couches sont connectées à une dernière couche (généralement une couche entièrement connectée dans le cas de l'apprentissage supervisé) pour obtenir la sortie finale. Ces architectures en couches permettent d'utiliser un réseau pré-entraîné (tel que BERT) comme un extracteur des caractéristiques si l'on enlève la dernière couche. Après l'extraction des caractéristiques on peut utiliser n'importe quel classifieur (SVM par exemple) pour classifier les objets d'intérêt sur la base des caractéristiques fournirent par le modèle pré-entrainé. Une méthode similaire permet de faire un apprentissage en utilisant la stratégie qui consiste à arrêter l'apprentissage sur certains poids ou couches au bout d'un moment, autrement dit à figer une partie du réseau tandis que le reste continue l'apprentissage, ainsi la rétro-propagation du gradient s'applique uniquement sur la partie non gelée du réseau. On peut considérer la partie figée comme un extracteur des caractéristiques et la partie non figée comme un classifieur initialisé à l'aide du pré-entraînement appliqué préalablement.Un réseau dont une partie est figée est appelée avec l'anglicisme *frozen.*

En ce qui concerne le temps d'apprentissage, la stratégie *frozen* est plus avantageuse par rapport au *fine-tuning* car le nombre des poids à ajuster dans le *frozen* est bien moins élevé qu'en *fine-tuning* car le temps d'une seule inférence sur les poids figés est bien moindre que le temps nécessaire pour entraîner ces poids.

2.2.2.3 Distillation

La distillation est une stratégie de transfert entre modèles, on l'utilise essentiellement pour la compression des modèles, il s'agit du transfert des connaissances acquises par un grand modèle vers un petit modèle. Le grand modèle (enseignant) doit en quelque sorte enseigner le petit modèle (élève) sans perte de validité. Même si les deux modèles sont entraînés sur les mêmes données, le petit modèle est incapable d'apprendre une représentation concise des connaissances. Cependant certaines informations sur une représentation concise des connaissances sont codées dans les pseudo-vraisemblances affectées à la sortie du modèle. Les pseudo vraisemblances (ou *soft-label* en anglais) peut être vu comme le processus suivant : après qu'un modèle prédit correctement une classe, il attribue une valeur élevée à la variable de sortie correspondant à cette classe et des valeurs plus petites aux autres variables de sortie. La distribution des valeurs dans le vecteur des pseudo-vraisemblances en sortie de modèle (enseignant) fournit des informations sur la façon dont ce grand modèle représente les connaissances. Par conséquent, l'objectif de faciliter l'apprentissage du petit modèle (élève) peut être atteint en entraînant uniquement le grand modèle sur les données où les étiquettes sont représentées par un vecteur «*one-hot encoding*» (*hard-label*) c'est-à-dire en exploitant sa meilleure capacité à apprendre des représentations de connaissances concises, puis en distillant ces connaissances dans un modèle plus petit, qui sera donc en mesure d'apprendre sur les pseudo-vraisemblances du grand modèle (*soft-label*).

2.3 Représentation des entrées

Le transformateur transforme des vecteurs réels en entrées en vecteurs réels en sorties tout en respectant le même nombre des vecteurs. Le texte doit donc être représenté par des vecteurs réels avant d'être traité par le transformateur. Cette opération est réalisée via un plongement contextuel (*contextual embedding*). Cette représentation sera détaillée dans cette section.

Le plongement lexical classique (word2vec[23], GloVe[27]) est en général basé sur la co-occurrence statistique des mots qui sont projetés indépendamment de leur contexte. Ainsi les relations sémantiques dans la phrase ne contribuent pas au plongement. Contrairement à cela le plongement contextuel projette les mots selon leur contexte dans la phrase et fournit en plus une représentation logique à l'échelle de la phrase. Cette représentation est largement utilisée dans les tâches de classification de texte.

Dans le cas de BERT[7] le plongement est réalisé en plusieurs étapes. On procède d'abord à un prétraitement des textes en insérant des symboles spéciaux dans le texte

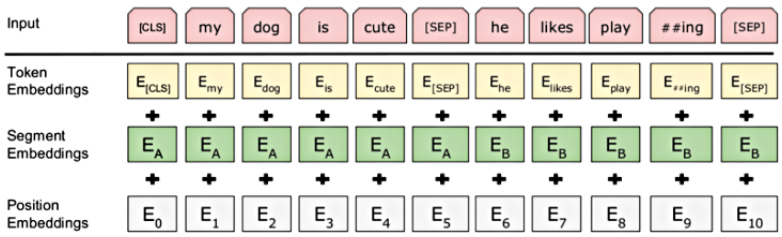

FIGURE 2.3 – La représentation des entrées de BERT [7]

brut, pour indiquer au transformateur certaines informations sur les mots et sur la composition du texte en entrée. Ensuite les mots sont décomposés en sous-mots (*token*) présents dans le vocabulaire et ces derniers seront représentés par leur identifiant dans le vocabulaire. Puis chaque token passera dans la couche du plongement lexical pour obtenir son vecteur de représentation (*token embedding*).
BERT possède en plus deux autres types de plongement : le plongement de position (*position embedding*) et le plongement de segmentation (*segmentation embedding*). Le premier porte l'information structurelle de la phrase tandis que le second porte l'information de positionnement des phrases dans l'entrée. La représentation finale de chaque sous-mot en entrée est la somme des trois vecteurs de représentation qui lui sont associés, la figure3.3 illustre comment ces trois plongement représentent une phrase donnée en entrée.

2.3.1 Prétraitement

Des informations sont indiquées au transformateur par le biais des sous-mots spéciaux insérés dans le texte. Nous en présentons ici les principaux :

$[CLS]$: indique le début de la phrase (le vecteur en sortie correspondant à ce sous-mot est très utilisé dans les tâches de classification car il porte une représentation de la phrase complète).
$[SEP]$: indique la séparation entre les phrases ou la fin de la phrase.
$[PAD]$: est utilisé pour compléter les dimensions vides d'un vecteur dont la dimension est inférieur à celle du plongement (*padding*).
$[MASK]$: est utilisé pour masquer des mots et pour indiquer au transformateur les mots à prédire dans la tâche de prédiction des mots masqués.

En ce qui concerne la tâche d'extraction des relations réalisée durant ce stage de Master, nous distinguons deux stratégies d'indication des entités nommées au transformateur :

L'encapsulation : les entités nommées sont entourées par des symboles spéciaux pour indiquer le début et la fin de chaque entité. Dans Beltagy et al. SCIBERT[2] par exemple les entités sont encapsulées par $(<< e >>, [[e]])$. Cette stratégie présente l'avantage de bien représenter les entités imbriquées ou celles qui se chevauchent. En outre, le modèle

peut apprendre non seulement le contexte de la phrase mais aussi le contexte dans les sous-phrases qui forment les entités.
Cette méthode est très utile dans les approches qui prennent en considération les entités nommées en sortie (s'il y a un post- traitement sur le contexte capturé par les entités séparément).

L'anonymisation : cette stratégie consiste à remplacer les entités nommées par des sous-mots spéciaux. Dans Lee et al. BioBERT[14], par exemple, les entités sont remplacée par (@$GENE$ @$DISEASE$) et dans [4] [$unused_1$][$unused_2$]. Cette opération permet au modèle d'identifier les sous-mots qui appartiennent aux entités nommées sans connaître les entités nommées, par conséquent elles assurent que le modèle a utilisé uniquement le contexte dans sa décision, ce qui permet d'éviter l'apprentissage des co-occurrences qui pourrait être une forme de sur-apprentissage.

2.3.2 Tokénisation

Il s'agit d'un processus de partition des mots en sous-mots qui appartiennent au vocabulaire de sorte que le mot sera remplacé par le minimum de sous-mots (*tokens*).
Exemple de tokenisation : le mot « embeddings » est représenté par 4 sous-mots dans le vocabulaire ('em', '##bed', '##ding', '##s')

Vocabulaire

Le vocabulaire diffère d'une version de BERT à une autre : par exemple BERT-base english possède 30 000 sous-mots, ce vocabulaire est construit par la méthode WordPiece tokenisation proposée par [40] ; cette méthode consiste à trouver un partitionnement des mots (vocabulaire) avec un bon compromis entre la taille du vocabulaire et le nombre des mots hors vocabulaire.
Ce vocabulaire est calculé à partir de très larges corpus, par exemple le vocabulaire de BERT de base [7] est construit avec Wikipédia (2,5 milliards de mots) et BookCorpus (800 millions de mots). Le vocabulaire de BERT est donc susceptible de bien couvrir l'anglais. Des variantes de BERT proposent de couvrir tout le vocabulaire du domaine cible, comme BioBERT[14], par exemple. En revanche d'autres variantes ont construit leur vocabulaire de façon plus adéquate au domaine cible comme SciBERT [2].

Gestion de mots hors vocabulaire

Contrairement aux approches traditionnelles qui rajoutent un mot au vocabulaire pour distinguer les mots hors vocabulaire, la méthode WordPiece représente les mots hors vocabulaire par une séquence de mots de vocabulaire, donc dans le pire des cas le mot hors vocabulaire sera remplacé par une séquence de caractères.

2.3.3 La segmentation

Le rôle de plongement de segmentation consiste à désigner les appartenances des sous-mots aux phrases, par conséquent à traiter des tâches multi-entrées. La prédiction de séquencement des phrases est une illustration d'une tâche de classification avec deux phrases en entrées, ces dernières étant présentées par un vecteur de segmentation composé d'une séquence de 0 (sous-mots de la première phrase) suivi par une séquence de 1 (sous-mots de la deuxième).

2.3.4 Plongement de position

L'ordre et la position des mots dans la phrase est une information utile pour la compréhension, car la conception de la phrase dans les langages naturels respecte un ordre défini (une phrase n'est pas un sac de mots). Les CNN ou RNN exploitent bien les informations locales hiérarchiques et donc tiennent compte de cette information. Par contre en ce qui concerne la couche d'attention dans le transformateur, aucune règle explicite sur l'ordre des éléments n'est imposée, ce qui nécessite de coder cette information dans les entrées avant de les passer au transformateur ou bien d'introduire cette information dans le mécanisme d'attention afin de l'exploiter. Ce plongement de position peut être fixé ou appris. Les méthodes du plongement de position utilisées avec les architectures transformateur seront présentées par la suite.

A. Plongement de position statique

Vaswani et al. [37]propose des fonctions périodiques qui permettent d'intégrer des informations sur la position relative en utilisant les équations (2.9) (2.10), on note *pos* la position du mot dans la phrase, PE le vecteur de plongement de position $PE \in \mathbb{R}^{d_{PE}}$, $i \in d_{PE}$ et d_{model} la taille maximale de la phrase (512 pour BERT). Les positions sont projetées dans l'espace du plongement par des vecteurs où les dimensions paires de ces vecteurs sont calculées par la formule (2.9) et les impaires par la formule (2.10)

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.9}$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.10}$$

B. Plongement de position adaptative

B.a Intégration du plongement dans l'attention

Le plongement de position peut également être appris comme d'autres paramètres dans le réseau de neurones. Shaw et al. [32] propose une méthode qui modélise les positions sous forme d'un graphe complet orienté en utilisant deux ensembles de paramètres

de position a_i, l'un pour représenter les valeurs et l'autre pour représenter les clés. Ces derniers modélisent le plongement de la position relative entre les positions i et j.
Dans le but d'intégrer cette information dans l'attention nous rajoutons ces paramètres dans la fonction d'attention, comme il est montré dans la figure3.2 ou comme on en voit le détail dans les équations d'attention modifiées (2.11) (2.12) proposées par [32],$x = (x1, ..., xn) : x_i \in \mathbb{R}^{d_x}$ et $z = (z1, ..., zn) : z_i \in \mathbb{R}^{d_z}$ représentent respectivement les séquences d'entrée et de sortie, et $W^Q, W^K, W^V \in \mathbb{R}^{d_x x d_z}$sont les matrices des paramètres d'attention : matrice requête, matrice clé, matrice valeur dans cet ordre, a_{ij}^V et a_{ij}^K sont respectivement les matrices des paramètres du plongement de position matrice clé et matrice valeur tandis que α_{ij} représente les coefficients de pondération calculés en utilisant une fonction $softmax$. Cela permet à la sortie d'attention de dépendre non seulement de la pertinence de la requête mais aussi de la position des mots. Par conséquent, nous pouvons apprendre a_i au cours d'entraînement.

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_i W^V + a_{ij}^V) \tag{2.11}$$

$$e_{ij} = \frac{x_i W^Q (x_i W^k + a_{ij}^K)^T}{\sqrt{d_z}} \tag{2.12}$$

B.b Représentations des positions relatives bornées

Pour les séquences linéaires, les bords peuvent saisir des informations sur les positions relatives entre les éléments d'entrée. L'article [32] part de l'hypothèse que des informations précises sur la position relative ne sont pas utiles au-delà d'une une certaine distance. La position relative maximale que nous considérons est limitée à une valeur absolue maximale k qui représente le nombre des arcs considérés ($2 <= k <= n-4 : n$ longueur maximale de séquences). Cette limitation de la distance maximale permet au modèle de généraliser les longueurs de séquences non vues pendant l'apprentissage, elle permet également de réduire le nombre de paramètres de position a_i a uniquement à $2k + 1$ paramètre à apprendre. Pour ce faire [32] propose une distance relative bornée (2.11)(2.12)(2.15).

$$a_{ij}^K = w_{clip(j-i,k)}^K \tag{2.13}$$

$$a_{ij}^V = w_{clip(j-i,k)}^V \tag{2.14}$$

$$clip(x, k) = max(-k, min(k, x)) \tag{2.15}$$

2.3.5 Plongement contextuel

Le plongement contextuel est une représentation des entrées par la combinaison de trois types différents de plongement, afin de constituer une entrée au transformateur BERT, telle qu'elle inclue les informations de contexte, de position et de segmentation. Les trois types de plongement, le plongement lexical, le plongement de segmentation et le plongement de position utilisent tous une couche linaire en sortie qui permet d'avoir

trois vecteurs de plongement de la même dimension $(n, 768)$ avec n qui définit le nombre de sous-mots en entrée. Une fois que les trois plongements sont calculés, le plongement contextuel équivaut à la somme des trois plongements, la figure3.3 illustre la construction d'entrée à partir des trois plongements.

2.4 L'architecture de BERT

BERT est la première architecture d'encodeur qui capture le contexte passé et futur simultanément, contrairement à OpenAI GPT(*Improving Language Understanding by Generative Pre-Training*) [29] qui capture uniquement le contexte passé ou Elmo(*Embeddings from Language Models*) [28] qui utilise une concaténation des contextes passé et future capturés indépendamment.

L'encodeur est composé d'une pile de 12 blocs identiques. L'architecture des blocs utilisés dans BERT est illustrée par la figure3.2 où chaque bloc est composé de deux sous-couches. La première est un mécanisme d'auto-attention à plusieurs têtes (2.1.2), et la seconde est un simple réseau de *feed-forward* entièrement connecté (FFN). Des connexions résiduelles sont utilisées autour de chacune des deux sous-couches, suivies par une normalisation appliquée après chaque sous-couche.

Le décodeur est également composé d'une pile de 12 (BERT base) blocs identiques. En plus des deux sous-couches d'encodeur, le décodeur insère une troisième sous-couche, qui effectue une attention multi-têtes sur la sortie d'encodeur. Donc comme il est montré dans la figure3.2 les vecteurs clés et valeurs viennent de la sortie de l'encodeur, et la requête est calculée à base d'attention sur les étiquettes dans le décodeur. Les sous-couches de décodeur suivent la même procédure que celle du codeur.

2.4.1 Attention

Ce processus itératif à travers les blocs aidera le réseau neuronal à capturer des relations plus complexes entre les mots dans la séquence d'entrée. Les sous-couches d'attention utilisées dans BERT correspondent à l'auto-attention à plusieurs têtes détaillée dans (2.1.2). Avec 12 têtes d'attention le modèle peut apprendre jusqu'à 12 types de relation entre les sous-mots dans chaque bloc, ce qui constitue déjà une bonne capacité de généralisation. L'indépendance calculatoire élevée est un autre avantage marquant dans cette sous-couche : comme le calcul de l'attention selon chaque tête se fait indépendamment des autres têtes, la performance relative au temps de calcul est appréciable.

2.4.2 *Feed-Forward*

La sous-couche *Feed-Forward* est composées elle-même de deux couches linéaires entièrement connectées avec une fonction d'activation RELU entres ces deux couches. Cette sous-couche permet de calculer les caractéristiques hiérarchiques non linéaires. Pendant cette étape, les vecteurs de représentation des sous-mots n'interagissent pas les uns avec

les autres. La sous-couche *Feed-Forward* est détaillée par la formule (2.16)

$$FFN(x) = max(0, xW1 + b1)W2 + b2 \tag{2.16}$$

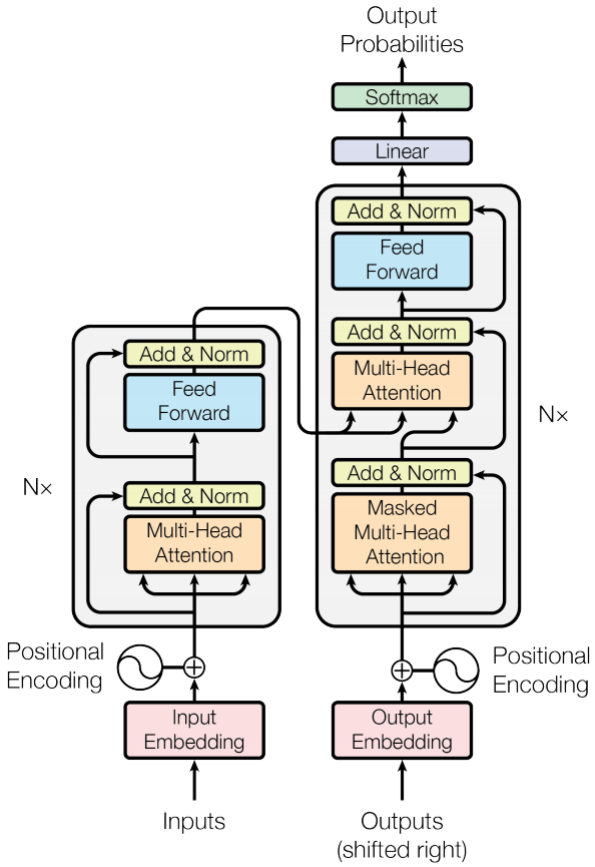

FIGURE 2.4 – L'architecture de transformateur BERT [37]

2.4.3 Liens résiduels

Dans l'encodeur et le décodeur, une connexion résiduelle est utilisée autour de chacune des deux sous-couches, suivie d'une normalisation des couches. Les connexions sautées ou résiduelles sont utilisées pour permettre aux gradients de circuler directement dans le réseau, sans passer par des fonctions d'activation non linéaires. Car ces dernières, par leur nature non linéaire, font exploser ou disparaître les gradients. Les connexions sautées forment conceptuellement un bus qui circule tout au long du réseau, cela permet de réaliser un apprentissage sur une architecture assez profonde, telle que BERT, d'une manière

douce. Autrement dit, les liens résiduels déforment la topologie de la fonction de perte, en appliquant une sorte de lissage sur cette dernière.

2.4.4 Normalisation

La normalisation permet de résoudre le problème de décalage interne des covariables produit dans un réseau de neurones. Ce problème concerne la variation des distributions au niveau des couches et il est dû à l'initialisation des poids ou à l'échelle des caractéristiques en entrée ou bien à la dérive de distribution dans les différentes profondeurs.
Le décalage interne des covariables se manifeste quand le réseau apprend et que les poids sont mis à jour, de sorte que la distribution des sorties d'une couche spécifique dans le réseau change. Cela oblige les couches supérieures à s'adapter à cette dérive, ce qui ralentit l'apprentissage et impacte les performances et la stabilité de réseau.

A. Normalisation par lot

L'article [10] aborde le décalage interne des covariables et propose une normalisation par lot (*batch normalisation*) qui permet de résoudre ce problème. Cette normalisation consiste à calculer la moyenne et la variance de chaque mini-lot et de normaliser chaque caractéristique en fonction des statistiques du mini-lot. Cela signifie que la moyenne et la variance seront différentes pour chaque min-lot. Cette dépendance pose des problèmes en fonction de la taille des lots et de la variation entre les lots. La normalisation est réalisée sur l'axe des mini-lots par les formules (2.17)(2.18)(2.19) xi représentant le $i^{ème}$ vecteur des caractéristiques dans le lot, m la taille du lot, μ la moyenne des vecteurs xi, σ l'écart-type des xi et $\hat{x_i}$ le vecteur xi normalisé.

B. Normalisation par couche

Une autre approche [1] traite le problème du point de vue des poids des couches. La normalisation est faite de la même façon que la normalisation par lot, sauf que les statistiques sont calculées sur l'axe de caractéristique et non pas sur l'axe des exemples. Donc pour chaque couche une moyenne et une variance sont calculées pour chaque exemple en entrée indépendamment des autres, ces deux mesures sont utilisées par la suite dans la normalisation de la couche en question. L'indépendance entre les entrées est l'avantage de cette méthode : chaque entrée a un fonctionnement de normalisation différent, ce qui permet d'utiliser des mini-lots de taille arbitraire. Comme la normalisation par lot, la normalisation par couche est réalisée sur l'axe des caractéristiques par les formules (2.17)(2.18)(2.19), xi représentant la $i^{ème}$ dimension de vecteur de caractéristique, m la dimension des vecteurs de caractéristique (la taille de de la couche), μ la moyenne des vecteurs xi, σ l'écart-type des xi et $\hat{x_i}$ le vecteur xi normalisé. L'architecture de BERT[7] adopte la normalisation par couche comme mécanisme de normalisation et le décrochage comme mécanisme de régularisation, ce dernier sera expliqué plus loin.

$$\mu \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.17}$$

$$\sigma^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{2.18}$$

$$\hat{x_i} \leftarrow \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.19}$$

2.4.5 Le décrochage et l'addition

Le décrochage (Dropout) est une méthode de régularisation qui permet de réduire le sur-apprentissage dans les réseaux de neurones. La co-adaptation entre les neurones peut conduire à un déséquilibre des poids, déséquilibre causé par le fait que le neurone s'appuie uniquement sur quelques caractéristiques en entrée entraînant une élévation des poids associés à ces caractéristiques, ce qui forme des relations de co-adaptations fortes. Ce phénomène de co-adaptation est une forme de sur-apprentissage : afin d'éviter ce type de sur-apprentissage, Srivastava et al.[33] ont proposé une méthode qui s'appelle le décrochage.Il s'agit d'un abandon de neurones choisis aléatoirement dans une couche, ce qui fait que la couche suivante n'a plus qu'une information partielle (uniquement les sorties des neurones conservés). Cette méthode empêche donc les neurones de s'appuyer toujours sur les mêmes caractéristiques, ce qui lui donne une meilleure capacité de généralisation. A chaque itération le neurone sera abandonné avec une probabilité p. Plus le paramètre p est élevé plus le nombre de neurones décrochés est important, la co-adaptation est donc moindre mais le risque de sous-apprentissage est plus élevé, en revanche si p est trop faible, la co-adaptation augmente ainsi que le sur-apprentissage.
En ce qui concerne BERT, un décrochage est appliqué après chaque sous-couche avec une probabilité de 0,1. Les vecteurs résultant de ce décrochage seront additionnés à l'entrée de cette sous-couche. Ce vecteur-somme sera ensuite normalisé utilisant une normalisation par couche. Ces différentes opérations sont illustrées par la formule (2.20)

$$LayerNorm(x + Dropout(Sublayer(x))) \tag{2.20}$$

2.5 Phase de pré-entraînement

Cette phase permet au transformateur d'avoir une certaine compréhension générale sous forme d'une capacité de représentation. Une telle capacité de représentation bidirectionnelle profonde est induite d'un entraînement à des tâches non supervisées (prédiction des mots masqués, prédiction de la phrase suivante) sur de larges corpus (Wikipedia (2,5B words), BookCorpus (800M words)). Cette capacité de représentation sera transférée à la phase en amont. Cette méthode correspond donc à l'apprentissage par transfert, plus précisément au transfert inductif du type multitâche au niveau de la phase de pré-entraînement et du transfert transductif du type d'adaptation de domaine entre la phase de pré-entraînement et la phase d'entraînement.

2.5.1 La tâche de prédiction des mots masqués

Contrairement à Elmo [28] et à OpenAI GPT [29], BERT [7]peut capturer le contexte passé et futur simultanément, comme il est montré dans la figure(2.5). Le problème qui a empêché ses prédécesseurs de le faire, c'est le biais engendré par le partage d'informations entre les deux contextes, autrement dit, chaque contexte (le contexte passé et le contexte futur) à travers l'autre contexte un accès direct à l'information qu'il cherche à généraliser. Cela introduit un biais qui empêche le modèle d'apprendre les deux contextes simultanément. La solution utilisée dans BERT est d'entraîner le modèle sur la généralisation du contexte bidirectionnel, sur une tâche où l'information que le modèle cherche à prédire est absente dans les deux contextes. Ainsi le partage d'informations entre contextes ne pose plus de problème. Il s'agit d'un masquage de mots aléatoirement choisis. Le modèle va ensuite apprendre à prédire ces mots masqués à partir de leur contexte bidirectionnel. La prédiction des mots masqués dans la phrase permet au modèle de BERT d'acquérir une bonne capacité de généralisation du contexte bidirectionnelle. En outre cette tâche est étiquetée automatiquement, non pas manuellement ; ce qui permet d'entraîner le modèle sur une quantité beaucoup plus large de données non supervisées, donc facilement disponibles.

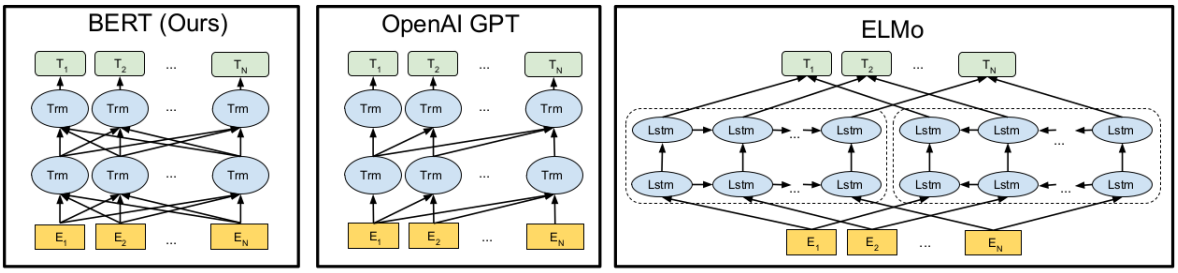

Figure 2.5 – Les différences entres les architecture des transformateur BERT, ELMo et OpenAI GPT [7]

Le but principal de la tâche de prédiction des mots masqués n'est pas d'obtenir la meilleure performance sur cette tâche, mais de transférer une bonne généralisation du contexte aux tâches en aval où il n'y a pas de masquage de mots. Delvin et al.[7] propose deux autres modes de remplacement (remplacement par un sous-mot aléatoire ou par le mot original) pour aider le modèle à séparer le vrai du faux selon le contexte bidirectionnel. Delvin et al. BERT[7] masque 15% des sous-mots. 80% des sous-mots masqués seront remplacés par le sous-mot [MASK], 10% par un sous-mot aléatoire et 10% conserveront le mot original. La perte se détermine sur la façon dont BERT prédit ou pas le mot manquant, et non sur l'erreur de reconstruction de la séquence entière.

2.5.2 La tâche de prédiction de la phrase suivante

De nombreuses tâches importantes en aval, telles que la réponse à une question (QA), sont basées sur la compréhension de la relation entre deux phrases, ce qui n'est pas directement capturé par la modélisation du langage utilisant la tâche de prédiction des mots masqués. Dans le but de former un modèle qui généralise la relation entre phrases, Delvin et al.[7] propose une seconde tâche étiquetée automatiquement. La tâche de prédiction de la phrase suivante est une tâche de classification binaire impliquant la prédiction pour dire si la deuxième phrase succède ou pas à la première dans le corpus. Il y a deux séquences en entrée séparées par le sous-mot [SEP] et les plongements de segmentation sont également utilisés pour indiquer l'appartenance des sous-mots aux phrases. Le corpus est construit d'une manière équilibrée, dans 50% des cas, la phrase suivante est correctement utilisée comme phrase suivante, et dans 50% des cas, une phrase est tirée aléatoirement du corpus. Cela garantit que le modèle s'adapte à la généralisation des relations entre les séquences multiples.

2.6 Tâches en aval

2.6.1 Classification des séquences

Le modèle pré-entraîné sera entraîné sur un ensemble de données supervisées pour prédire la classe d'une séquence donnée. Le sous-mot de classification ([CLS]) est utilisé ici. La sortie de ce sous-mot est considérée comme la sortie groupée du classificateur et elle est ensuite placée dans une couche de classification entièrement connectée pour obtenir la sortie étiquetée.

2.6.2 Reconnaissance des entités nommées

Les vecteurs de représentation en sortie du transformateur sont directement placés dans une couche de classification, avec le nombre d'étiquettes comme unités de sortie pour chacun des sous-mots. Ensuite une $softmax$ est appliquée sur les vecteurs en sortie de la couche de classification. Les valeurs dans ces vecteurs sont interprétées comme la probabilité de prédiction de chaque classe (types d'entités nommées). Ces derniers sont utilisés pour obtenir la classe prédite de chaque sous-mot en utilisant $argmax$.

2.6.3 Inférence en langage naturel

Le modèle pré-entraîné est entraîné de la même manière que pour la tâche de prédiction de la phrase suivante. Le texte et l'hypothèse sont séparés à l'aide du sous-mot [SEP] et identifiés à l'aide du plongement de segmentation. Le sous-mot [CLS] est utilisé pour obtenir le résultat de la classification, comme cela a été expliqué dans la partie classification des séquences.
A partir d'une phrase, la tâche consiste à choisir la suite la plus plausible parmi quatre choix. Donc il y 4 séquences d'entrée, chacune contenant la phrase originale et le choix

correspondant concaténé à celle-ci. Le vecteur en sortie associé au sous-mot [CLS] est transmis à une couche entièrement connectée pour obtenir les scores de chacun des choix, qui sont ensuite normalisés à l'aide d'une couche $softmax$.

2.6.4 Réponse aux questions

Un paragraphe est utilisé comme une séquence et la question est utilisée comme une autre séquence. Il y deux cas de figure à propos de cette tâche :

— Dans le premier cas, la réponse se trouve dans le paragraphe. Dans ce cas, la tâche consiste à trouver le début et la fin de la réponse dans le paragraphe. Pour les trouver, Devlin et al.[7] introduit un vecteur de début $S \in \mathbb{R}^H$ et un vecteur de fin $E \in \mathbb{R}^H$, soit H la taille des vecteurs en sortie du transformateur. Le produit de chaque sous-mot Ti et du vecteur de début S est utilisé pour obtenir la probabilité que le sous-mot i soit le début de la réponse. De même, nous obtenons la probabilité du sous-mot de fin j. Le score d'un intervalle candidat de la position i à la position j est défini comme $S.Ti + E.Tj$, et l'intervalle de score maximum où $j \geq i$ est utilisé comme prédiction.

— Dans le second cas, il n'y pas de réponse courte à la question présente dans le paragraphe, ce qui représente un cas plus réaliste. Pour cela, on utilise les scores de probabilité pour le début et la fin de la réponse, calculés avec le vecteur de représentation correspondant au $[CLS]$ et les vecteurs de représentation des sous-mots. Ces scores (s_null) seront comparés avec le score maximum obtenu à la meilleure plage de candidats (c'est-à-dire le meilleur score pour le premier cas). Si le score obtenu est supérieur à s_null d'un seuil suffisant τ, nous utilisons le meilleur score du candidat comme réponse. Le seuil τ peut être réglé pour maximiser le score F1 sur la base de développement.

2.7 Variantes de BERT

Après la sortie de l'article [7], d'autres recherches ont adopté l'architecture et la méthodologie proposées, mais elles ont utilisé d'autres données BioBERT[14],SciBERT[2] par exemple, ou d'autres méthodes d'entraînement comme DistilBERT, ou d'autres langues. Cela a donné naissance à des variantes de BERT, qui peuvent être plus performantes que BERT, ALBERT[13][ALBERT] par exemple dans le domaine général ou BioBERT [14], et SciBERT [2] dans le domaine biomédical. Elles peuvent générer un gain du temps important par rapport à BERT [7] telles DistilBERT [DistilBERT] [31]. Des variantes de BERT seront présentées plus loin.

2.7.1 Domaine générale

BERT-large

L'article [7] a proposé deux variante de BERT, BERT-base avec 12 blocs transformateurs, avec une taille de 768 pour les vecteur des représentations, avec un nombre total des paramètres valant 110M de paramètres. Une version plus large (BERT-large) a été proposée dans le même article avec 24 blocs de transformateurs, une taille de représentation de 1024 avec 340M de paramètres. Cette dernière est légèrement plus performante que BERT base sur GLUE, mais elle est plus coûteuse en terme de temps de calcul.

BERT-uncased

Le papier [7] a proposé également deux variantes BERT-cased et BERT-uncased. Cette dernière convertit tous les caractères en minuscule, c'est la variante la plus utilisée car dans la plupart des tâches la majuscule n'apporte aucune information supplémentaire, en revanche la variante BERT-cased préserve les majuscules dans le texte pour les tâches où les majuscules portent un supplément d'information, la détection des entités nommées en est une illustration, car la majuscule est largement présente avec les noms propres, les références et d'autres entités nommées.

AlBERT

L'augmentation de la taille de BERT (BERT-large 340M de paramètres) pose de plus en plus de problèmes d'entraînement sur les GPU/TPU. Ces problèmes sont essentiellement des problèmes de limite de mémoire et des problèmes de communication dans le cas d'entraînement distribué. ALBERT Lan et al.[13] a montré que l'augmentation de la taille du modèle ne conduit pas forcément à un modèle plus performant, il peut se produire une dégradation des performances lorsqu'on augmente le nombre de paramètres, on peut constater cet effet en comparant les performances de BERT-large et BERT-xlarge (BERT-xlarge est un BERT deux fois plus large que BERT-large). L'article [13] a proposé une méthode qui améliore BERT sur le plan des performances et sur celui du nombre de paramètres et donc sur le temps de calcul. La solution proposée dans [13] consiste à partager les paramètres, ce qui en réduit le nombre. Plusieurs méthodes de partage des paramètres sont proposées dans l'article : ou un partage entres les sous-couches feed-forward ou bien entre les sous-couches d'attention ou encore entres les blocs entiers. Cet article a montré aussi que la diminution de la dimension de plongement peut améliorer les performances. Lan et al. [13] propose aussi une nouvelle tâche de pré-entraînement, il s'agit de la prédiction de l'ordre des phrases à la place de la prédiction du séquencement des phrases utilisé dans [7]. La prédiction de l'ordre est aussi une tâche binaire où le modèle doit prédire si l'ordre des phrases est inversé ou non. Cette tâche permet au modèle d'apprendre une distinction plus fine sur les propriétés de cohérence entre les phrases. Comme Lan et al.[13] a réussi à réduire le nombre des paramètres il peut facilement augmenter la taille des vecteurs de représentation, il propose donc des modèles présentant de larges vecteurs de représentation tout en gardant un nombre raisonnable de paramètres. (ALBERT-base, ALBERT-large, ALBERT-xlarge, ALBERT-xxlarge avec les nombres des paramètres 12M 18M 60M 235M respectivement)

DistilBERT

BERT de base est un modèle très profond disposant d'un grand nombre de paramètres, ce qui le rend non utilisable sur certains supports physiques, du fait de leur temps d'inférence et par manque d'une mémoire suffisante. Il existe de nombreuses techniques pour pallier cette difficulté : la quantification (approximation des poids d'un réseau), l'élagage des poids (suppression de certaines connexions dans le réseau) et la distillation basée sur l'apprentissage par transfert, point déjà évoqué dans la section (2.2.2) L'article [31] aborde l'application de la distillation sur BERT [7]et propose une variante de BERT appelée DistilBERT.
DistilBERT est un modèle de langage présentant beaucoup moins de paramètres et entraîné à partir de la supervision de BERT. Le modèle élève (DistilBERT) possède la même architecture que BERT mais réduit le nombre de couches par un facteur de deux, et supprime le plongement de segmentation (utilisé pour la tâche de prédiction de phrase suivante). En ce qui concerne le temps d'inférence, DistilBERT, possédant moins de paramètres, est 60% plus rapide que BERT et 120% plus rapide et plus petit que ELMo + BiLSTM.

2.7.2 Domaine biomédicale

Il existe des variantes de BERT qui reprennent le modèle pré-entraîné de BERT-base [7] et qui appliquent un deuxième transfert de domaine dans le but de rendre BERT plus performant dans un domaine spécifique tel que le domaine biomédical. Deux variantes BioBERT [14], et SciBERT [2] seront expliquées plus loin.

BioBERT

BioBERT [14], est une variante de BERT [7] qui reprend l' architecture de BERT et l'entraîne sur des données biomédicales afin d'acquérir une bonne capacité de représentation dans le domaine biomédical.

Données biomédicales
Les données biomédicales utilisées dans BioBERT [14], sont essentiellement des articles et des extraits d'articles dans le domaine biomédical. Ces données sont composées de deux larges corpus PubMed qui compte plus de 29 millions d'extraits d'articles avec environ 4.5 milliards de mots et MPC où les articles sont entièrement utilisés et contient environ 13.5 milliards de mots.

Vocabulaire
Le vocabulaire utilisé dans cette variante est le vocabulaire de BERT [7] (BASEVOCAB) sous l'hypothèse qu'il couvre le domaine biomédical.

Modèles et Méthodes

BioBERT [14] a réussi à adapter BERT [7] au domaine biomédical et donc à améliorer sa performance dans les tâches de TAL appliquées à des données biomédicales. BioBERT propose plusieurs versions qui diffèrent par les données utilisées et par le nombre d'itérations utilisé dans la phase de pré-entraînement pour adapter BERT au domaine biomédicale. Par rapport au nombre d'itérations. il existe deux versions : v1.0 avec 470k d'itérations et v1.1 avec un million d'itérations. La première version présente trois sous-versions selon les données utilisées (PubMed, PMC , PubMed + PMC).

Prétraitement

Le prétraitement consiste à anonymiser les entités nommées dans le texte par les sous-mots (@GENE$) pour la première entité et par (@DISEASE$) pour la deuxième.

Pré-entraînement

Dans la phase de pré-entraînement, les poids du modèle sont initialisés par les poids de BERT-base [7] avant d'entraîner ce modèle sur les tâches non supervisées (prédiction de la phrase suivante et prédiction des mots masqués) en utilisant des données biomédicales dans le but d'acquérir une bonne généralisation de représentation des textes biomédicaux.

Tâches en aval

La stratégie de transfert utilisée dans BioBERT [14], est le *fine-tuning* expliqué dans la section (2.2.2). Comme la rétro-propagation du gradient s'applique à tous les poids, on n'ajoute qu'une seule couche pour les tâches en aval, le réseau étant déjà très profond.

Le *fine-tuning* est appliqué sur trois tâches (la reconnaissance des entités nommées, l'extraction des relations et la réponse aux questions) sur des corpus biomédicaux citant ChemProt [12] pour l'extraction des relations.

Optimisation

Comme dans BERT [7] BioBERT [14], utilise la fonction *cross-entropy* comme une fonction de perte, avec une optimisation réalise par l'algorithme AdamWithDecay [21] utilisant la technique de gestion de pas d'apprentissage adaptative appelée « linear warmup ». Différentes tailles de lot (*batch*) sont utilisées (10,16,32,64), avec différentes initialisations de pas d'apprentissage ($5.10^{-5}, 3.10^{-5}, 10^{-5}$).

SciBERT

SCIBERT [2] est un modèle base BERT [7] de langage pré-entraîné, entraîné sur plusieurs corpus scientifiques pour effectuer différentes tâches de traitement automatique des langues sur des données scientifiques. Ces tâches incluent, entre autres, l'extraction des relations, le balisage de séquences, la classification de textes etc. SCIBERT définit l'état de l'art sur quelques-unes de ces tâches en aval, notamment l'extraction des relations sur le corpus ChemProt [12]. SciBERT effectue également des expérimentations approfondies sur la stratégie de transfert *frozen* déjà expliquée dans la section (2.2.2) et sur l'effet du vocabulaire dans le domaine. Une comparaison avec des variantes précédentes BERT [7]et BioBERT [14], est également proposée dans cet article.

Données scientifiques

SCIBERT utilise 1.14M articles de « Semantic Scholar. » et il exploite le texte des article, y compris celui des résumés. Les articles ont une longueur moyenne de 154 phrases, la taille totale de ces données est d'environ 3,17 milliard mots approchant ainsi la taille des données utilisées dans BERT [7] avec 3,3 milliard mots. Les données scientifiques sont utilisées dans la construction de vocabulaire et dans la phase de pré-entraînement. Signalons que 82% de ces données scientifiques appartiennent au domaine biomédical.

Vocabulaire

En utilisant la méthode « Wordpiece » [40] l'article SciBERT [2] a construit un vocabulaire scientifique dont la taille correspond à la taille du vocabulaire de BERT [7] avec 30K sous-mots avec une intersection de 42% entres les deux vocabulaires (BASEVOCAB, SCIVOCAB). Ceci illustre la différence substantielle entre le vocabulaire utilisé dans le domaine générale et le domaine scientifique.

Modèles et méthodes

SciBERT [2] définit l'état de l'art en ce qui concerne plusieurs tâches de traitement automatique des langues dans le domaine scientifique en générale et dans le domaine biomédical en particulier. Utilisant l'architecture de BERT, SciBERT propose 4 modèles qui diffèrent selon le vocabulaire utilisé (BASEVOCAB ou bien SCIVOCAB) et ou selon qu'ils prennent en compte ou non les caractères majuscules/minuscules (*Cased* ou bien *uncased*).

Prétraitement

Le prétraitement consiste à encapsuler les entités nommées dans le texte par les sous-mots (< < , > >) pour la première entité et par ([[,]]) pour la deuxième.

Pré-entraînement

Dans la phase de pré-entraînement, les poids de ces modèles sont initialisés par les poids de BERT-base [7] avant d'être entraînés sur des tâches non supervisées (prédiction de la phrase suivante et prédiction des mots masqués) en utilisant des données scientifiques dans le but d'acquérir une bonne généralisation de représentation des textes scientifiques.

Tâches en aval

Dans la phase d'entraînement sur les tâches en aval (Reconnaissance d'entités nommées, Extraction PICO, Classification de texte, Extraction des relations, Analyse des dépendances) deux stratégies de transfert sont employées : le *fine-tuning* exposé dans la section (2.2.2) et le *frozen* expliqué dans la même section.

Dans le cas de *frozen* les poids du transformateur sont figés (pas de mise à jour des poids dans cette partie du réseau). Comme il n'y pas de rétro-propagation du gradient dans cette partie de l'architecture le transformateur joue le rôle d'un extracteur des caractéristiques plus précisément le rôle d'un plongeur contextuel bidirectionnel. Pour classifier les caractéristiques extraites, on utilise un classificateur composé de deux couches de types

LSTM suivi par un PCM (perceptron multi-couche). Le dernier vecteur de représentation en sortie de ces couches LSTM est placé dans un PCM pour calculer la sortie. Cette architecture combine le transformateur qui a une bonne capacité de représentation contextuelle mais une faible généralisation des informations structurelles, avec un réseau de neurones récurrent qui peut capturer l'information structurelle.

Dans le cas de *fine tuning* comme BERT [7], SciBERT [2] ajoute une seule couche en sortie pour appliquer le *fine-tuning*. Comme dans ce cas la rétro-propagation du gradient s'applique sur le transformateur qui est déjà assez profond, le nombre de couches ajoutées sera très restreint pour éviter des problèmes liés à la profondeur du réseau et les problèmes de stabilité liés à l'initialisation des nouvelles couches.

Optimisation

SciBERT [2] utilise la fonction *cross-entropy* comme une fonction de perte, avec une optimisation réalisée par l'algorithme Adam [11], avec une taille de lot (*batch*) valant 32 avec différentes initialisations de pas d'apprentissage (5.10^{-5} , $3.10^{-5}, 10^{-5}$).

2.7.3 les performances sur l'extraction des relations

Comme dans la plupart des tâches de traitement automatique de la langue, BERT, les variantes de BERT et les architectures de base BERT définissent l'état de l'art sur l'extraction des relations dans le domaine général et dans le domaine biomédical. La combinaison de toutes les techniques de pointe dans le domaine de l'apprentissage profond et dans l'apprentissage par transfert ainsi que l'utilisation des architectures issues de l'état de l'art dans le traitement automatique de la langue abordé dans cette section, permettent à BERT et aux modèles de base BERT d'atteindre des résultats très supérieurs à ceux des approches précédentes.

Dans le domaine général et sur le corpus de référence SemEval [9] l'article Li et al.[17]qui est un modèle base BERT[7] représente l'état de l'art avec une F-mesure macro égale à **91,0%** en dépassant les performances de Manning et al.[39] avec **89,25%.**

Dans le domaine biomédical sur le corpus référence ChemProt [12], c'est SciBERT [2] qui définit l'état de l'art avec **83,64%** de F-mesure micro en dépassant largement les performances de ses prédécesseurs BERT (BioBERT [14], avec **76,46%** , BERT [7] et avec **73,74%**).

Chapitre 3

Méthodes

Sommaire

3.1 Exploration des données

3.1.1 Compositions des corpus

Dans ce travail nous avons utilisé 4 corpus (PGxCorpus[15], SNPPhena[3], ChemProt[12], SemEval[9]), en nous focalisant sur PGxCorpus et ChemProt qui sont des corpus biomédicaux. Nous avons exploré ces corpus à travers l'étude de la composition, de la hiérarchie, des fréquences lexicales, de l'écartement entre les entités nommées, de l'adaptation au vocabulaire et de la visualisation des caractéristiques de représentation.

PGxCorpus

PGxCorpus est composé de 2,875 exemples, répartis sur 7 différents types de relation (*influences, causes, decreases, increases, treats, isEquivalentTo*). Le pourcentage et le nombre des exemples associés à chaque type de relation sont présentés dans la figure (3.1). On remarque que la répartition des relations est déséquilibrée (la relation la plus fréquente est (*influences*), relation qui représente 32.6 % du corpus tandis que (*causes*) la relation la moins fréquente représente 5.8 % du corpus) ; certes il existe des types de relation beaucoup plus fréquents que les autres mais le nombre des exemples dans les classes les moins fréquentes est notable (168 exemples dans la classe la moins fréquente). Il existe une hiérarchie entre les relations : elle est exposée dans la figure (3.2) et même si on prend cette hiérarchie en considération, le déséquilibre entre le nombre des relations demeure.

ChemProt

ChemProt représente un large corpus avec 10 031 exemples (base d'apprentissage, base de test, base de validation). La figure (3.1) montre la distribution des exemples sur les relations dans la base d'apprentissage. ChemProt est un corpus très déséquilibré où la relation la plus fréquente dans la base d'apprentissage présente 39.4 % alors que la relation la moins fréquente, ne représente que 0.1 % . Il y a donc des classes très fréquentes et des classes rares, telle (*AGONIST INHIBITOR*) avec uniquement 4 exemples dans la base d'apprentissage. Cela représente une difficulté pour le modèle qui peinera à généraliser ces classes rares.

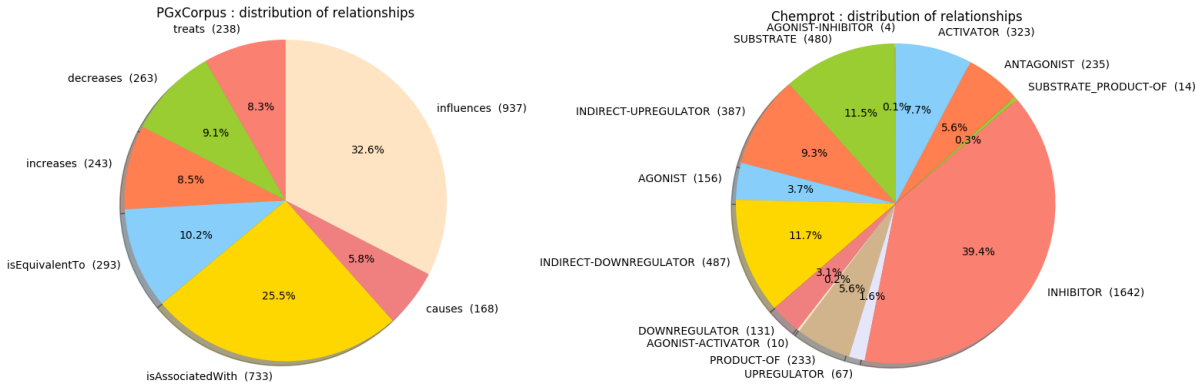

FIGURE 3.1 – Distribution des relations dans PGxCorpus et ChemProt

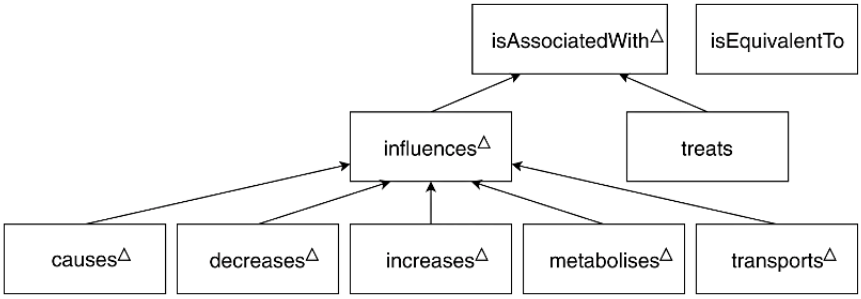

FIGURE 3.2 – La hiérarchie des relations en PGxCorpus [15]

3.1.2 La Fréquence lexicale

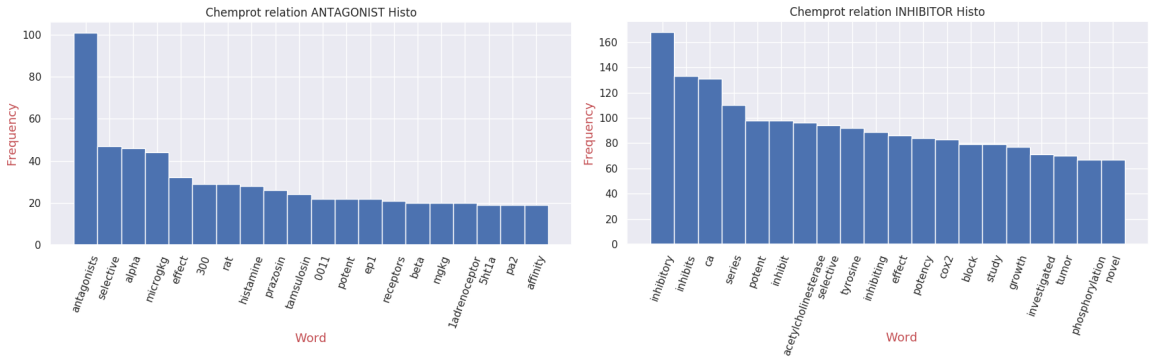

FIGURE 3.3 – Histogramme des relation (ANTAGONIST,INHIBITOR) en ChemProt

L'étude des statistiques de la fréquence des mots dans le corpus peut faire émerger des mots-clés dans le vocabulaire utilisé dans le corpus, ce qui nous donne une idée sur la composition des phrases dans les corpus.

Pour extraire le vocabulaire associé à une relation nous avons procédé de la manière suivante : on supprime d'abord les mots les plus utilisés dans la langue (liste des *stop-words*), ensuite on calcule l'histogramme des corpus, puis l'histogramme selon chaque type de relation en excluant les 20 mots les plus fréquents dans le corpus. Donc selon chaque type relation on calcule l'histogramme des 20 mots les plus fréquents dans la relation mais qui ne figurent pas parmi les plus fréquents dans le corpus ou dans la langue.

La figure (3.4) montre deux exemples des histogrammes des relations de type (*decreases*) et (*increases*) dans le corpus PGxCorpus. Dans cette figure on peut remarquer quelques mots clés qui émergent, dans l'histogramme (*decreases*) par exemple (*reduced, lower, decreased, reduction*). On trouve aussi des mots-clés dans l'histogramme de la relation « *increases* » par exemple (*increased, increase, higher, greater*). On peut également observer des mots-clés dans le corpus ChemProt, dans la figure (3.3) on trouve les histogrammes des relations (*INHIBITOR, ANTAGONIST*) où figurent aussi des mots liés à ces relations, (*antagonists, inhibitory*) par exemple.

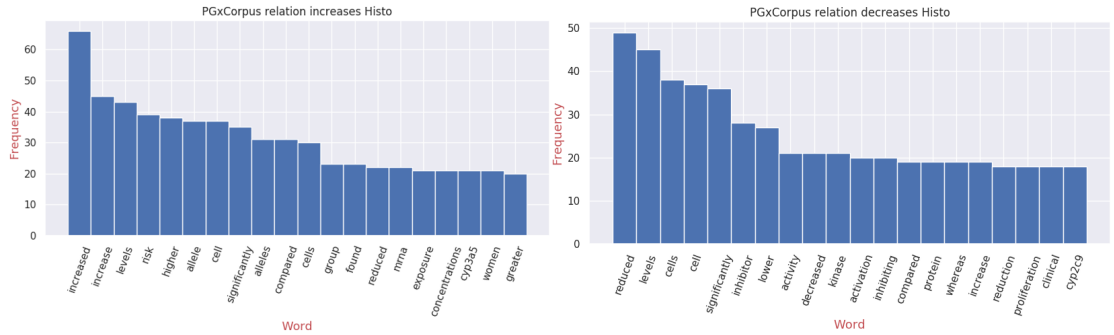

FIGURE 3.4 – Histogramme des relation (increases,decreases) en PGxCorpus

Une remarque intéressante qu'on peut tirer à partir des histogrammes, c'est qu'il existe dans les corpus des exemples où apparaissent des mots liés aux relations, ces relations s'expriment donc avec un vocabulaire explicite et on peut considérer ces exemples comme des exemples relativement faciles pour le modèle.

3.1.3 L'écartement des entités nommées

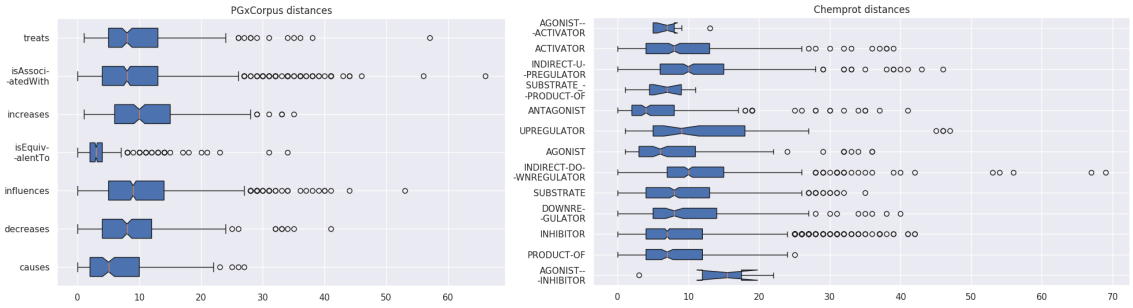

FIGURE 3.5 – Les boîtes à moustache des distances entre entités nommées selon chaque relation en PGxCorpus et ChemProt

La distance entre les entités nommées est l'une des caractéristiques importantes du corpus car le choix du modèle peut dépendre de cette distance (choisir ou non un modèle adapté aux longues relations), Dg-spanbert [4] est un exemple de modèle adapté au longues relations. La figure (3.5) expose les boîtes à moustache d'écartement des entités nommées selon chaque relation. On remarque que la distance ne dépasse pas 30, et que la médiane ne dépasse pas 10 dans la majorité des relations pour les deux corpus PGxCorpus et ChemProt. Cela signifie que l'écartement est relativement faible. Les boîtes à moustache des différentes relations se chevauchent, ce qui signifie que l'écartement entre les entités nommées est une caractéristique non discriminante. A la différence des boîtes à moustache des relations, celle de la relation (*isEquivalentTo*) dans le corpus PGxCorpus est une petite boîte (relation avec des écarts faibles) cela s'explique par la façon dont est construit le corpus PGxCorpus[15] où cette relation est utilisée pour définir des acronymes ou des entités équivalentes, d'où un écartement réduit dans cette relation.

3.1.4 Tokénisation

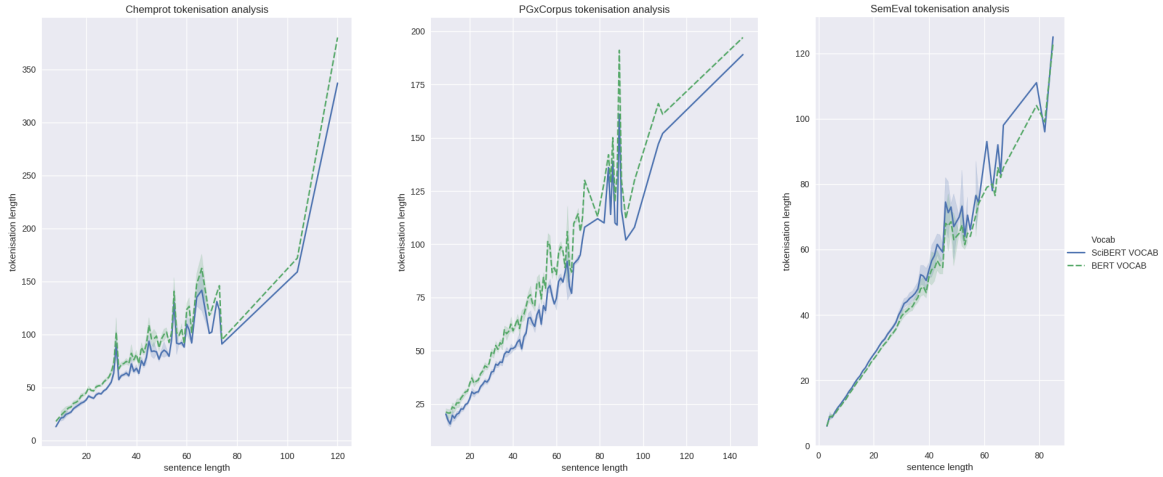

FIGURE 3.6 – Longueurs des tokénisation avec Sci-Vocab et BERT-Vocab

Dans le but d'analyser l'adaptation des vocabulaires au corpus, nous observons les longueurs des séquences des sous-mots générées par un vocabulaire (nombre de sous-mots) par rapport à la longueur de la phrase (nombre de mots). La figure (3.6) expose la comparaison entre deux vocabulaires BERTVOCAB[7] et SciVOCAB[2] sur trois corpus dont deux appartiennent au domaine biomédical (PGxCoprus, ChemProt) et dont le dernier appartient au domaine général (SemEvale). L'axe des abscisses représente la longueur de la phrase et l'axe des ordonnées la longueur de la tokénisation. Plus la tokénisation est courte, plus le vocabulaire utilisé est adapté. On remarque dans la figure (3.6) que la courbe de SciVOCAB est toujours en dessous de la courbe de BERTVOCAB dans les corpus biomédicaux alors que SciVOCAB est toujours est en dessus de BERTVOCAB sur SemEval, ce qui signifie que SciVOCAB est plus adapté aux corpus biomédicaux (PGxCorpus, Chemprot) et que BERTVOCAB est plus adapté aux corpus du domaine général (SemEval).

3.1.5 Visualisation des données

Pour visualiser les vecteurs de représentation calculés par le transformateur avant et pendant l'apprentissage, on utilise trois méthodes différentes de réduction de la dimensionnalité (ACP, T-SNE, T-SNE sur ACP).

ACP

L'analyse en composantes principales (ACP) est une méthode linéaire de réduction de la dimensionnalité qui est souvent utilisée pour réduire la dimensionnalité de grands ensembles de données, en transformant un grand ensemble de variables en un ensemble réduit qui contienne encore la plupart des informations du grand ensemble.
L'ACP consiste en une projection des données sur les axes où la dispersion est maximale, ce qui permet à cette transformation de réduire le nombre de variables d'un ensemble de

données tout en préservant le plus d'informations possible.

T-SNE

La méthode T-SNE (*T-distributed Stochastic Neighbor Embedding*) est une méthode non linéaire de réduction de la dimensionnalité, l'algorithme de T-SNE comprend deux étapes :
- Dans la première étape : l'algorithme construit une distribution de probabilités sur des paires d'objets de grande dimension de telle sorte que les objets similaires se voient attribuer une probabilité plus élevée tandis que les points dissemblables se voient attribuer une probabilité très faible.
- Dans la deuxième étape : l'algorithme définit une distribution de probabilités similaires sur les vecteurs de dimension réduite, en minimisant la divergence de Kullback-Leibler entre les deux distributions. L'algorithme utilise la distance euclidienne entre les objets dans sa métrique de similarité.

3.2 Architectures proposés

3.2.1 BERT-MLP

Dans les articles BERT[7], BioBERT[14], SciBERT[2] l'extension ajoutée après le transformateur avec la stratégie *fine-tuning* est un simple perceptron multicouche composé d'une seule couche cachée entièrement connectée. Nous avons utilisé cette architecture dans la reproduction des résultats dans l'article SciBERT[2] sur ChemProt[12], nous avons aussi testé cette architecture simple avec les différentes variantes de BERT sur PGxCorpus, cette architecture est également utilisée dans le transfert de domaine expliqué plus loin.

3.2.2 BERT-RNN

L'article [2] a proposé une architecture base RNN comme un classifieur utilisé avec le transformateur BERT en appliquant un transfert de domaine par la stratégie de transfert *frozen*. L'architecture du classifieur utilisée est composée de deux couches LSTM bidirectionnelles suivies par une couche entièrement connectée. Les transformateurs (BERT[7], SciBERT[2]) sont utilisés comme des extracteurs des caractéristiques. Les caractéristiques extraites par le transformateur constituent des séquences des vecteurs de représentation, comprenant un vecteur par sous-mot. Ainsi la partie RNN dans cette architecture sert à exploiter l'information contextuelle dans cette séquence afin d'extraire un vecteur contextuel qui représente la séquence en entrée. Ce dernier est placé dans une dernière couche entièrement connectée, puis une régularisation par décrochage est appliquée pour éviter le sur-apprentissage (co-adaptation). La rétro-propagation du gradient s'applique seulement sur le classifieur, donc l'impact du nombre des paramètres ajoutés sur la stabilité du modèle est limité, c'est pourquoi cet article a utilisé cette architecture avec la stratégie *frozen* et non pas avec la stratégie *fine-tuning* dans l'article SciBERT[2]. Cette architecture présente l'avantage d'un temps de calcul moindre, avantage important dans les applications où les ressources de calcul sont limitées. Nous avons implémenté cette architecture dans

le but de reproduire les expériences de l'article SciBERT[2] et de tenter d'autres expériences non présentées dans cet article, par exemple l'utilisation de BioBERT[14] comme extracteur des caractéristiques.

3.2.3 BERT-CNN

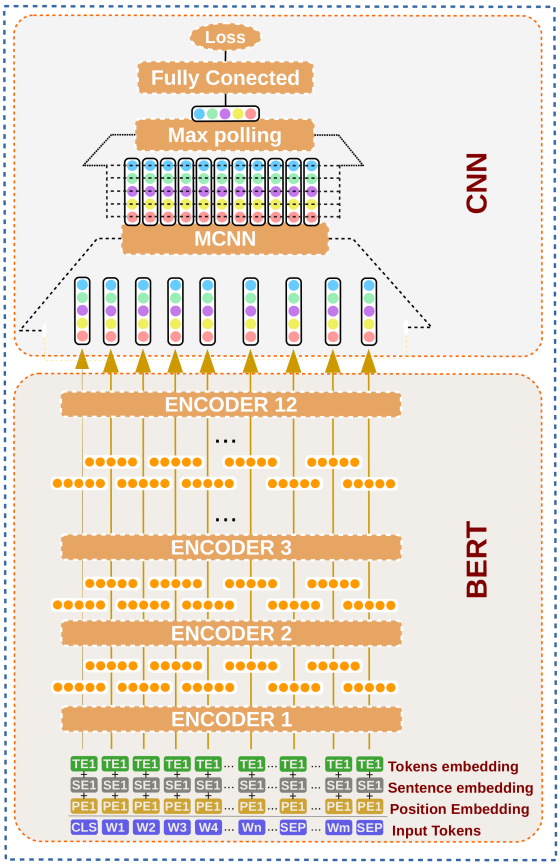

FIGURE 3.7 – Architecture BERT-CNN

Le transformateur BERT a une capacité de représentation de l'information contextuelle des sous-mots par l'utilisation de l'attention multi-têtes, dans le but d'extraire l'information locale dans les vecteurs de représentation en sortie de BERT. Pour ce faire nous proposons une architecture MCNN. L'absence d'ordre explicite entre les différentes dimensions dans les vecteurs de représentation nous incite à les considérer comme des caractéristiques indépendantes, on traite donc chaque dimension comme un plongement différent. Ainsi chaque dimension représente un canal pour notre MCNN, de sorte que les filtres de convolution s'opèrent entre les mots selon chaque canal comme le montre la figure (3.7). Dans cette architecture la partie CNN doit calculer des vecteurs de caractéristique à partir des vecteurs de représentation, c'est-à-dire l'extraction de l'information locale latente dans les vecteurs de représentation. Un *maxpolling* est appliqué pour réduire la taille des caractéristiques et pour rendre le modèle invariant au léger décalage dans les

caractéristiques en entrée. Les vecteurs en sortie du *maxpolling* sont considérés comme des vecteurs de représentation au niveau de la phrase, ils servent donc à la classification des relations. Ces vecteurs sont placés dans une dernière couche entièrement connectée qui les classifie, comme le montre la figure(3.7). Une normalisation par couche est également appliquée au niveau de MCNN comme au niveau de la couche entièrement connectée. Le nombre réduit de paramètres (paramètre partagés) rend cette architecture utilisable avec les deux stratégies *frozen* et *fine-tuning*. Avec *fine-tuning* le modèle reste assez stable (il y peu nouveaux de paramètres ajoutés et l'impact de leur initialisation est limité). Plus précisément le nombre réduit de paramètres dans le CNN permet d'utiliser une architecture profonde comme une extension au transformateur tout en évitant le décalage interne des co-variables. Afin d'éviter le sur-apprentissage (co-adaptation) on applique encore une régularisation par décrochage.

3.2.4 BERT-RNN-CNN

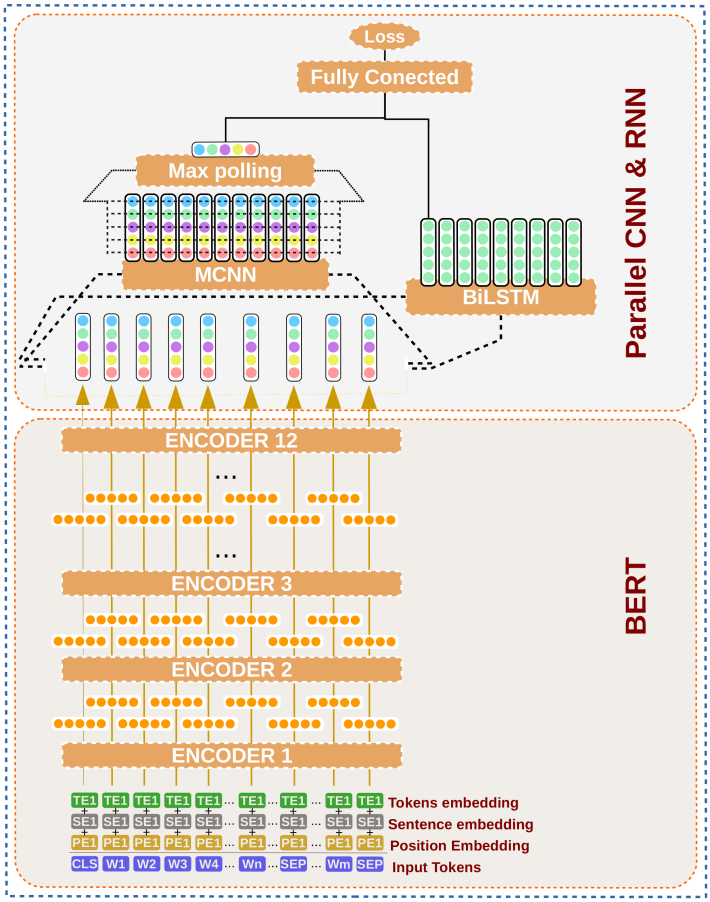

FIGURE 3.8 – Architecture en parallèle BERT-CNN-RNN

Nous proposons deux architectures extensions au transformateur en combinant RNN(3.2.2)

et CNN(3.2.3). Comme le nombre de paramètres des poids ajoutés est important, nous utilisons avec ces architectures la stratégie *frozen* :

RNN-CNN en chaîne

Comme l'illustre la figure (3.9) on applique une architecture en chaîne (trans-

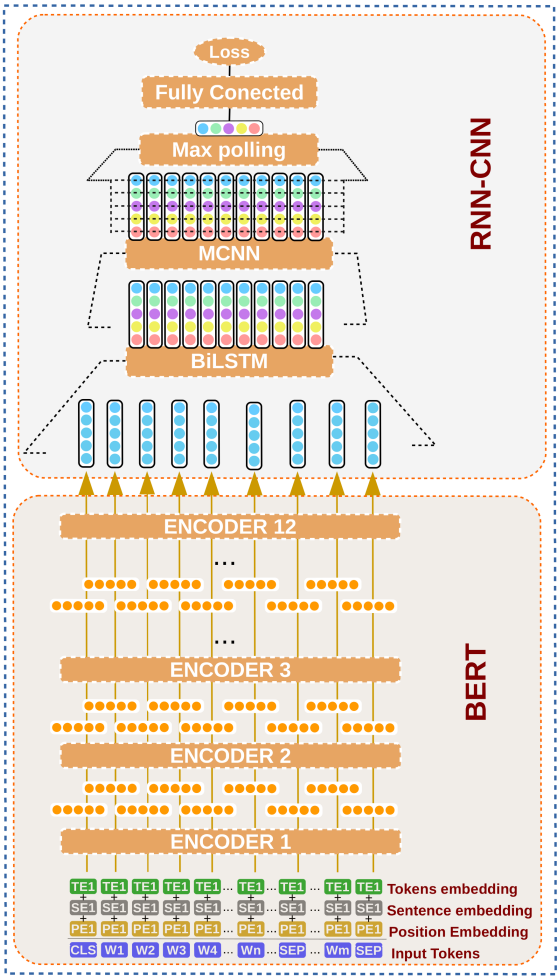

FIGURE 3.9 – Architecture en chaîne BERT-CNN-RNN

formateur BERT, RNN et CNN) : la partie CNN de l'architecture sert à extraire les caractéristiques locales latentes dans les vecteurs d'état en sortie de RNN. Ces derniers résultent de la transformation des vecteurs de représentation du transformateur.

Archetectures parallèle

Nous avons aussi proposé une architecture parallèle ou un CNN(3.2.3) et un RNN(3.2.2) qui prennent comme entrées les vecteurs de représentation calculés par le transformateur, chaque partie (CNN, RNN) calcule séparément un vecteur de représentation. La concaténation de ces deux vecteurs de représentation est placée dans une dernière couche entiè-

rement connectée, ainsi les caractéristiques extraites par RNN (caractéristiques contextuelles) et CNN (caractéristiques locales) servent à la classification. cf. la figure (3.8).

3.2.5 BERT-CNN Segmentation

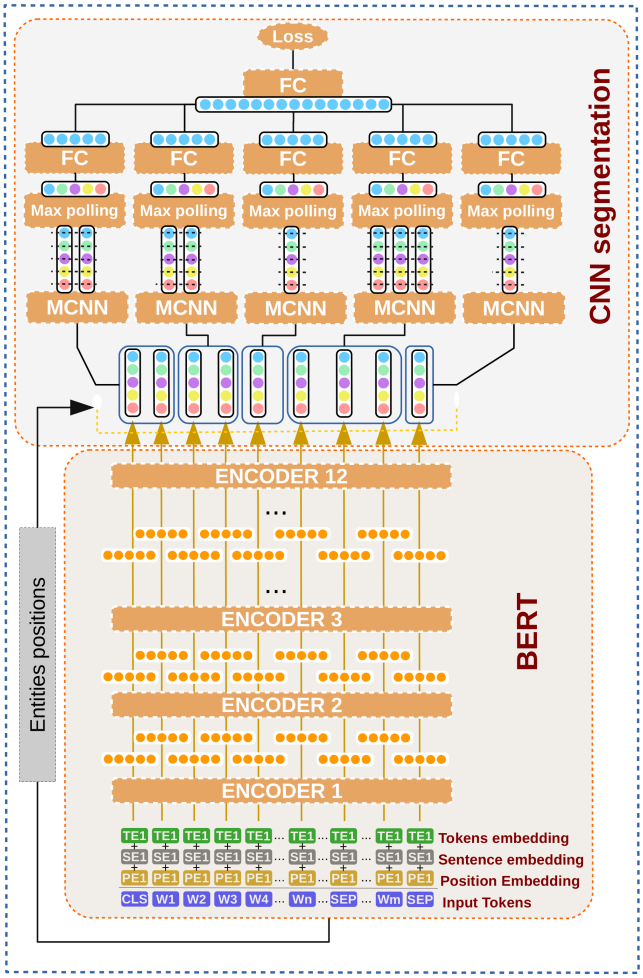

FIGURE 3.10 – Architecture CNN avec la ségmentation

Comme aucun ordre n'est imposé dans le transformateur entre les vecteurs de représentation, l'information structurelle de la phrase est peu présente dans les vecteurs de représentations en sortie de transformateur. Dans le but de renforcer notre modèle par cette information nous avons proposé une architecture composée d'un transformateur, suivi par cinq CNN(3.2.3) en parallèle, chacun traitant indépendamment une partition des vecteurs de représentation calculés par le transformateur.

L'idée de segmentation a été suggérée par les articles [5] [19] qui appliquent une segmentation sur la phrase (une première partie avant la première entité nommée, la 1ère entité nommée, une partie entre les deux entités, la deuxième entité nommée, la partie

après la seconde entité nommée), ces articles utilisent la segmentation comme un prétraitement. Quant à nous, nous utilisons la segmentation comme un post-traitement où les positions des entités sont utilisées pour partitionner la séquence des vecteurs de représentation, comme le montre la figure (3.10).
Chaque partition est traitée indépendamment l'une de l'autre par le CNN associé à cette partition pour extraire les caractéristique locales dans les vecteurs de représentation associés à cette partition. Ensuite la concaténation des vecteurs de représentation calculés par les cinq CNN est placée dans une dernière partie *feed-forward*. Comme le montre la figure (3.10), le dernier classifieur *feed-forward* utilise l'information des caractéristiques locales extraites par les CNN ainsi que l'information de la provenance de ces caractéristiques. Cela permet d'enrichir les caractéristiques par une information structurelle qui peut améliorer les performances du modèle tout en gardant un nombre réduit de paramètres et une bonne stabilité avec le *fine-tuning*.

3.3 Paramètres

Paramètres Nous avons testé différentes techniques et paramètres afin de proposer des architectures performantes et stables. Dans nos architectures on ajoute des nouveaux paramètres aux réseaux pré-entrainés, ce qui peut engendrer un décalage interne des covariables. Pour éviter ce problème nous avons utilisé deux types de normalisation (normalisation par couche, normalisation par lot). Nous utilisons différentes régularisations (L2-regularization, Dropout[33]) les valeurs utilisées pour le décrochage (0.1, 0.2, 0.25, 0.5) et pour la L2-regularisation (0.01). L'optimisation est un facteur important dans le processus d'apprentissage et nous utilisons un algorithme d'optimisation avec un pas d'apprentissage adaptatif Adam[11], nous avons aussi exploré une variante de cet algorithme AdamWithDecay[21] avec un *decay* valant 0.01. Les pas d'apprentissage initiaux utilisés avec Adam sont : (0.001) pour la stratégie de transfert *frozen*, (5.10 -5 , 3.10 -5 , 10 -5) pour la stratégie fine-tuning. Nous avons employé également des méthodes de gestion de pas d'apprentissage (*Linear-warm-up*, *polynomial-warm-up*). La fonction de perte utilisée est l'entropie croisée (*cross entropy*) et les fonctions d'activation utilisées (RELU, GELU, Tanh, hardTanh). Nous avons utilisé une taille de lot (batch) valant 32. Nous avons initialisé les poids ajoutés par une loi normale (μ= 0.0 et σ=0.02). Le nombre d'époques est (30, 65) pour la stratégie *frozen*, et (5, 8, 10) pour la stratégie fine-tuning. Pour les architecture basées sur le CNN les filtres de convolution utilisés sont de taille (3, 5, 7) et le nombre de filtres (3, 6, 7,16). Nous avons également utilisé deux types de *polling* (*average pooling*, *max pooling*). Pour le corpus ChemProt une base de validation et une base de test sont fournies, on les utilise donc pour la sélection et pour l'évaluation du modèle. Pour le corpus PGxCorpus une seule base est fournie, nous avons adopté donc une validation croisée (*k-fold cross validation*) avec k (10, 5). La procédure d'apprentissage se fait à partir de différentes initialisations aléatoires, nous rapportons les performances moyennes, nous calculons également l'écart-type et les boîtes à moustaches associées aux distributions des performances pour analyser la stabilité des modèles.

Chapitre 4

Résultat et discussion

Sommaire

4.1 *Frozen*

4.1.1 ChemProt

ChemProt frozn			
	Model	F-score micro	σ
SOTA	SciBERT LSTM[2]	75.03	–
SOTA Reproducing	SciBERT LSTM	75.10	1.00
BERT+Ours	BERT LSTM	64.41	2.42
	BERT CNN	68.26	1.54
	BERT LSTM-CNN	75.35	1.02
	BERT LSTM-CNN parallel	62.07	1.33
BioBERT+Ours	BioBERT LSTM	72.86	1.36
	BioBERT CNN	77.57	0.70
	BioBERT LSTM-CNN	**80.08**	0.80
	BioBERT LSTM-CNN parallel	74.85	0.96
SciBERT+Ours	SciBERT CNN	77.85	**0.68**
	SciBERT LSTM-CNN	79.24	0.76
	SciBERT LSTM-CNN parallel	70.45	1.18

TABLE 4.1 – Les résultats obtenus par la stratégie frozen sur ChemProt

Comme le montre le tableau (4.1) nous avons pu reproduire les résultats obtenus par SciBERT Beltagy et al.[2] avec un écart de 0.07 % F-micro. Cet article présente l'état de l'art dans le corpus ChemProt en fine-tuning. Considérant les résultats de cet article comme référence dans les deux stratégies *frozen* et *fine-tuning*, cet article utilise le modèle SciBERT comme extracteur des caractéristiques. Nous avons utilisé nos architectures *frozen* avec différentes variantes de BERT (BERT base, BioBERT, SciBERT) comme extracteurs des caractéristiques avec les différentes architectures que nous avions proposées pour la stratégie *frozen*. En utilisant le modèle BERT de base comme extracteur des caractéristiques nous avons pu atteindre des résultats similaires à ceux de la référence SciBERT(75.03 % F-micro) avec l'architecture BERT+LSTM-CNN(75.35 % F-micro). Avec l'extraction des caractéristiques utilisant SciBERT nous avons obtenu des résultats bien meilleurs que l'architecture-référence SciBERT+LSTM proposée par Beltagy et al.[2], ces améliorations sont obtenues grâce à l'architecture CNN avec 77.85% F-micro (2.82 % amélioration absolue) ainsi que par l'architecture en chaîne SciBERT+LSTM-CNN avec 79.24 % F-micro (4.21 % amélioration absolue par rapport au Beltagy et al.[2]). En utilisant BioBERT comme extracteur des caractéristiques nous avons pu avoir des écarts plus importants par comparaison avec la référence SciBERT, les deux architectures BioBERT-

CNN et **BioBERT-LSTM-CNN** avec respectivement des F-micro 77.57% et **80.08 %**, ce qui représente une amélioration absolue respectivement de 2.54 % et **5.05 %**. En ce qui concerne la stabilité, comme le montrent la figure (4.2) et les écarts-types dans le tableau (4.1), les modèles CNN sont les plus stables. L'écart-type minimal (0.68) est obtenu par SciBERT CNN. LSTM-CNN représente l'architecture la plus performante sur le corpus Chemprot avec les trois variantes de BERT. BioBERT LSTM-CNN représente le modèle le plus performant avec **80.08 %** F-micro.

4.1.2 PGxCorpus

PGxCorpus frozn					
	Model	Precision	Recall	F-score	σ
Baseline model	MCNN[15]	–	–	45.67	4.51
BERT+Ours	BERT LSTM	54.29	54.82	54.29	0.84
	BERT CNN	57.64	53.84	53.73	1.33
	BERT LSTM-CNN	71.85	70.47	70.63	1.38
	BERT LSTM-CNN parallel	54.48	54.37	53.99	**0.28**
BioBERT+Ours	BioBERT LSTM	57.16	57.32	56.93	0.89
	BioBERT CNN	69.52	66.25	67.02	2.43
	BioBERT LSTM-CNN	**73.05**	71.81	72.09	1.71
	BioBERT LSTM-CNN parallel	67.46	64.64	65.39	0.57
SciBERT+Ours	SciBERT LSTM	62.60	62.26	62.18	1.18
	SciBERT CNN	69.59	66.57	67.35	1.74
	SciBERT LSTM-CNN	72.65	**72.58**	**72.38**	1.04
	SciBERT LSTM-CNN parallel	61.54	61.11	61.00	1.45

TABLE 4.2 – Les résultats obtenus par la stratégie frozen sur PGxCorpus

Un premier modèle base MCNN[30] est proposé dans l'article [15]. Ce modèle a atteint 45.67 % F-macro sur le corpus PGxCorpus et nous utilisons ce modèle comme référence. Toutes nos architectures dépassent significativement le modèle de référence Monnin et al.[24] avec tous les transformateurs. Avec BERT de base comme extracteur des caractéristiques, nous avons pu dépasser cette référence avec des performances 70.63 % grâce au modèle BERT LSTM-CNN, ce qui représente déjà un écart de 24.96 % par rapport au référent ainsi qu'un écart de 16.34 % par rapport au BERT LSTM. En utilisant BioBERT comme extracteur des caractéristiques, les trois modèles BioBERT CNN, BioBERT LSTM-CNN et BioBERT LSTM-CNN en parallèle obtiennent des résultats bien meilleurs que la référence (67.02 %, 72.09 %, 65.39 % F-macro respectivement), ce qui représente une amélioration absolue respectivement de (21.35 %, 26.42 %, 19.72%) par rapport à la référence ainsi qu'une amélioration absolue respectivement de (10.9 %, 15.16 % , 6.7 %)

par rapport au BioBERT LSTM. Adoptant sciBERT comme extracteur des caractéristiques notre architecture améliore encore davantage les résultats avec 67.35 %, 72.38 % pour les modèles SciBERT CNN, SciBERT LSTM-CNN avec respectivement une amélioration de 21.68 % et 26.71 % ainsi qu'une amélioration (respectivement de 5.17 %, 10.2 %) par rapport au SciBERT-LSTM. BioBERT LSTM-CNN offre la précision le plus élevée (73.05 %), en revanche **SciBERT LSTM-CNN** présente le meilleur rappel (72.58 %) et F-macro (**72.38 %**). Comme le montrent la figure(4.2) et le tableau(4.2), le modèle le plus stable avec ce corpus est BERT LSTM-CNN-parallèle. Le modèle LSTM-CNN réalise les meilleures performances avec les trois transformateurs (BERT, BioBERT, SciBERT).

4.2 *Finetuning*

4.2.1 ChemProt

ChemProt fine tune			
	Model	F-score micro	σ
SOTA	SciBERT MLP[2]	83.64	–
SOTA Reproducing	SciBERT MLP	82.44	1.47
BERT+Ours	BERT MLP	79.28	1.21
	BERT CNN	72.18	1.73
	BERT CNN Segmentation	81.43	0.91
BioBERT+Ours	BioBERT MLP	82.28	3.27
	BioBERT CNN	83.90	1.11
	BioBERT CNN Segmentation	**85.37**	**0.67**
SciBERT+Ours	SciBERT CNN	82.98	0.96
	SciBERT CNN Segmentation	84.77	**0.67**

TABLE 4.3 – Les résultats obtenus par la stratégie fine-tuning sur ChemProt

SciBERT Beltagy et al.[2] présente l'état de l'art sur Chemprot avec 83.64 % micro-F en utilisant la stratégie de transfert *fine-tuning*. L'architecture utilisée dans cet article consiste en MLP avec SciBERT, on considère donc ce modèle comme le modèle de référence. Le tableau (4.3) présente les résultats de nos architectures de *fine-tuning* ainsi que nos reproductions des résultats de référence. Nous avons reproduit les résultats de SciBERT Beltagy et al.[2] en utilisant la stratégie de transfert *fine-tuning*, le résultat de reproduction obtenu est 82.44 % avec un écart de 1.2 % par rapport à la référence. Nous avons pu dépasser légèrement le modèle de référence avec BioBERT-CNN (83.90%), ce qui présente une amélioration absolue de 0.46%. Une autre amélioration plus significative est atteinte avec notre architecture SciBERT-CNN Segmentation (84.77 %), ce qui présente une amélioration absolue de 1.13 %. Notre architecture **BioBERT-CNN-Segmentation**

définit le nouvel état de l'art avec **85.37 %** F-micro sur ChemProt, ce qui présente une amélioration absolue de **1.73%**. En plus ce modèle offre une meilleure stabilité avec un écart-type valant 0.67. Notre architecture BERT-CNN apporte une légère amélioration par rapport au PMC avec les trois transformateurs (BERT, BioBERT, SciBERT). Le tableau (4.3) et la figure(4.2) indiquent que notre architecture CNN-Segmentation est l'architecture la plus performante et la plus stable avec les trois transformateurs.

4.2.2 PGxCorpus

PGxCorpus Fine tune					
Trensfer	Model	Precision	Recall	F-score	σ
Baseline model	MCNN[15]	–	–	45.67	4.51
BERT+Ours	BERT MLP	70.80	69.70	69.88	4.03
	BERT CNN	71.73	73.39	72.22	**0.98**
	BERT CNN Segmentation	74.31	74.53	74.17	1.08
BioBERT+Ours	BioBERT MLP	73.49	68.84	70.47	5.87
	BioBERT CNN	74.40	71.16	72.23	4.30
	BioBERT CNN Segmentation	77.38	77.24	77.00	2.56
SciBERT+Ours	SciBERT MLP	75.61	75.44	75.32	2.11
	SciBERT CNN	75.88	76.08	75.70	1.04
	SciBERT CNN Segmentation	**78.15**	**79.17**	**78.44**	1.07

TABLE 4.4 – Les résultats obtenus par la stratégie fine-tuning sur PGxCorpus

Comme le montre le tableau (4.4) qui utilise la stratégie *fine-funing*, toutes nos architectures dépassent significativement le modèle de référence Legrand et al.[15] avec tous les transformateurs. En outre elles présentent moins de variations que la référence. Notre architecture **CNN** offre une amélioration significative par rapport au MLP avec **2.34 %** F-macro avec BERT, **1.76 %** avec BioBERT et **0.38 %** avec SciBERT. Notre architecture CNN segmentation présente l'amélioration la plus importante par rapport au MLP avec 4.29 % en utilisant BERT, 6.53 % en utilisant SciBERT et 3.12 % avec BioBERT. Les meilleures performances obtenues sur PGxCorpus le sont avec **SciBERT CNN Segmentation** avec une amélioration absolue de **32.77 %** sur F-macro (avec une amélioration absolue de rappel valant 3.73 % ainsi qu'une amélioration du 2.54 % quant à la précision par rapport au SciBERT PMC). SciBERT CNN Segmentation représente donc le nouvel état de l'art sur PGxCorpus avec **78.44 %** F-macro.

FIGURE 4.1 – La précision globale (accuracy) et la fonction de perte sur la base d'apprentissage sur PGxCorpus et ChemProt

4.3 Stabilité et convergence

La figure(4.1) montre la convergence des différentes architectures utilisées avec la stratégie *frozen*, l'axe des abscisses représente les époques, l'axe des ordonnées représente la précision globale (accuracy) et la valeur de la fonction de perte (cross entropy) qu'on cherche à minimiser. On remarque que le CNN converge rapidement (environ 12 époques) par rapport aux autres modèles, ce qui présente un avantage de temps de calcul. Les architectures mixtes (CNN/RNN) nécessitent un nombre moyen d'époques pour converger (environ 22 époques) tandis que l'architecture RNN demande plus de 30 époques pour converger. En plus de l'amélioration des performances, nos architectures offrent donc un gain de temps de convergence.

On remarque dans la figure(4.2) que les modèles base BioBERT sont moins stables avec PGxCorpus (boîtes à moustache plus large) que les modèles SciBERT et BERT. La stratégie *frozen* est généralement plus stable que le *fine-tuning*, à l'exception du modèle BERT-CNN avec une précision qui varie beaucoup par rapport aux autres modèles appliqués avec la stratégie *frozen*. On remarque que notre architecture CNN-segmentation améliore la stabilité avec la stratégie *fine-tuning* avec BERT, BioBERT et SciBERT pour le corpus PGxCorpus ainsi qu'avec BioBERT et SciBERT pour le corpus ChemProt.

4.4 Visualisation

Comparaison des espaces des caractéristiques

La figure (A3) représente les réductions de la dimensionnalité d'espace des caractéristiques de PGxCorpus avant le *fine-tuning*, autrement dit l'espace des caractéristiques utilisées dans la stratégie *frozen*. La figure (A4) représente les réductions de la dimensionnalité d'espace des caractéristiques après cinq époques de *fine-tuning* (avec un modèle SciBERT MLP). On remarque que les vecteurs de représentation sont distribués d'une manière arbitraire dans la figure (A3), cela reflète la difficulté de la classification avec la stratégie *frozen* et explique pourquoi l'écart de performance réalisé par nos archi-

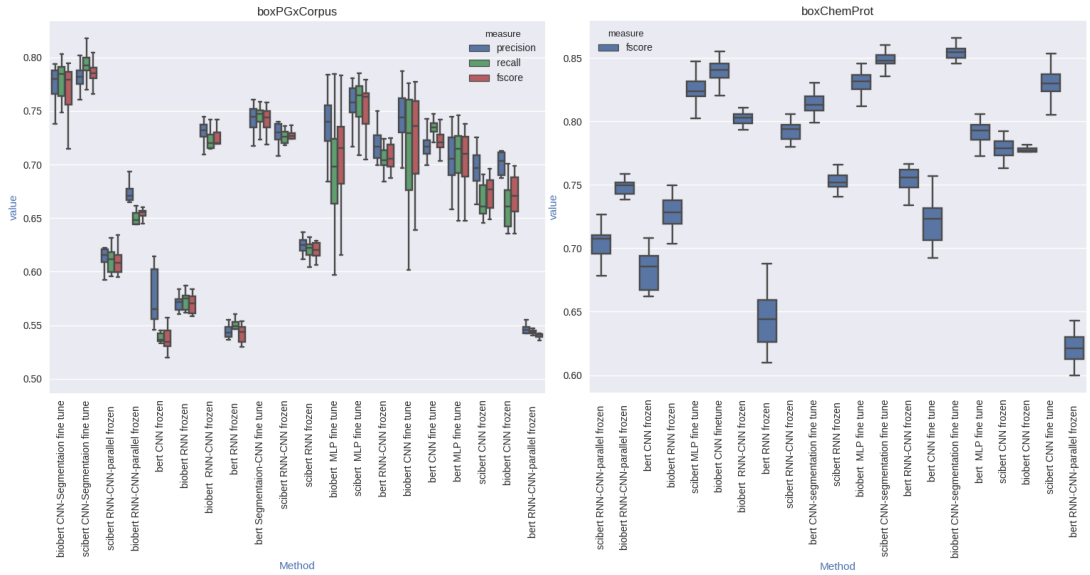

FIGURE 4.2 – Les boîtes à moustache des résultats obtenus sur PGxCorpus et ChemProt

tectures en *frozen* est beaucoup plus important qu'en *fine-tuning*. Dans la figure (A4) les caractéristiques sont groupées, ce qui reflète la facilité de classification avec le *fine-tuning* et explique pourquoi notre CNN atteint des performances presque similaires à un simple MLP en *fine-tuning*. Donc il est nécessaire d'ajouter davantage d'informations (information structurelle via la segmentation) pour obtenir des améliorations significatives (CNN-segmentation).

L'apprentissage et la hiérarchie des relations

Nous avons analysé les visualisations des réductions de la dimensionnalité des vecteurs de représentation avec le modèle SciBERT MLP appliqué sur le corpus PGxCorpus. Les figures (A5), (A6), (A7), (A8), (A4) représentent la visualisation des vecteurs de représentation dans l'époques 1, 2, 3, 4, 5 respectivement. La figure (3.2) expose la hiérarchie dans PGxCorpus. Nous avons remarqué que le modèle semble suivre la hiérarchie dans son apprentissage : la figure (A5) montre que le modèle a appris à séparer la classe *isEquivalentTo* des autres classes, ce qui constitue le premier niveau de la hiérarchie dans la figure (3.2). Ensuite dans la figure (A6) le modèle sépare (influences) des autres classes, enfin dans les figures (A7), (A8), (A4) le modèle sépare les classes feuilles (les niveaux terminaux) dans la hiérarchie (3.2).

4.5 Analyse d'erreur

ChemProt

Dans la figure (4.3) on remarque que le profil de BioBERT et de SciBERT sont très similaires avec un rappel et une précision nuls pour les classes rares qui sont donc mal

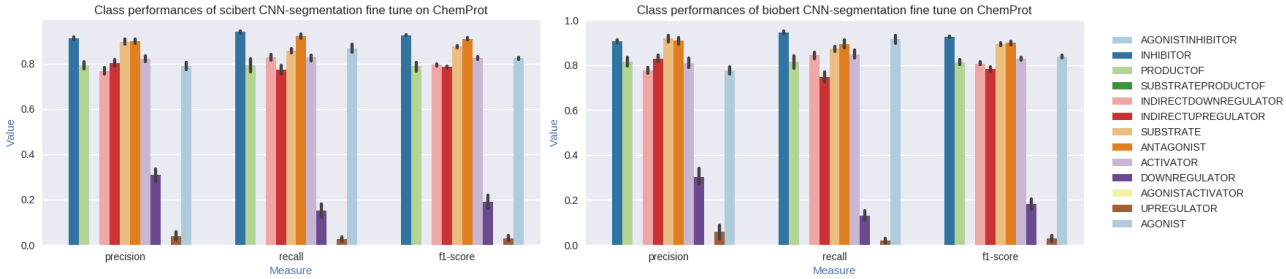

FIGURE 4.3 – Les performances détaillées des modèle SciBERT-CNN-segmation et BioBERT-CNN-segmation sur ChemProt

prédites. On peut le constater clairement dans la matrice de confusion (A9) où tous les exemples appartiennent aux classes rares (*AGONISTACTIVATOR, AGONISTINHIBITOR*) et sont prédits par *AGONIST*, cela s'explique par le nombre insuffisant des exemples pour discriminer ces classes rares. Comme le montre la figure (4.3) les classes (*UPREGULATOR, DOWNREGULATOR*) sont les classes non rare où CNN-segmentation est moins performant : la matrice de confusion (A9) montre que 46 % des exemples appartenant à (*DOWNREGULATOR*) sont confondus avec (*INHIBITOR*) de même que 39 % des exemples de (*DOWNREGULATOR*) sont confondus avec (*INDIRECTDOWNREGULATOR*). Certes (*UPREGULATOR*), (*DOWNREGULATOR*) ne sont pas des classes rares mais sont les classes les moins fréquentes parmi les classes non rares, ce qui explique le nombre élevé d'erreurs sur ces classes.

PGxCorpus

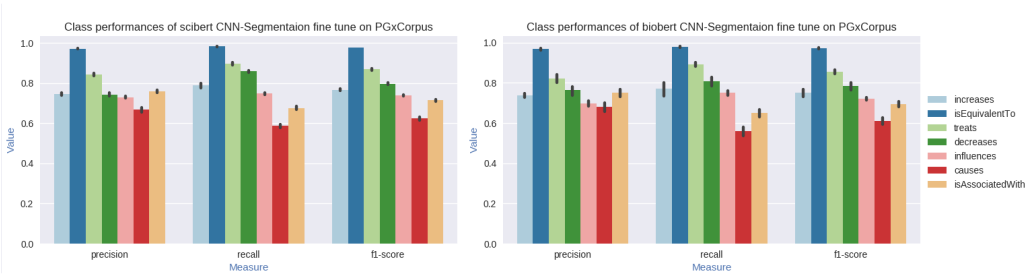

FIGURE 4.4 – Les performances détaillées des modèle SciBERT-CNN-segmation et BioBERT-CNN-segmation sur PGxCorpus

Dans la figure (4.4) on remarque que les modèles BioBERT-CNN-segmentation et SciBERT-CNN-segmentation ont des profils similaires mais que SciBERT-CNN-segmentation est légèrement plus performant quant à la précision des classes (*treats, influences, isAssociatedWith*) et quant au rappel des classes (*increases*, *decreases, causes, isAssociatedWith*). Comme le montre la figure (4.4), (*isAssociatedWith, influences, causes*) sont les classes où CNN-segmentation fait le plus d'erreurs, ce qui découle peut-être de la hiérarchie (3.2). Nous avons constaté que les erreurs hiérarchiques représentent la plupart des erreurs faites par CNN-segmentation. En effet sur la matrice de confusion (A10), 23 % des exemples de (*isAssociatedWith*) sont attribués à la classe-fille (*influences*), 14 % des exemples de (*influences*) sont attribués à la classe-mère de la hiérarchie (*isAssociatedWith*) de même que 12% des exemples de la classe (*causes*) sont confondus avec leurs parents-directs de la hiérarchie (*influences*) et 13 % avec leurs parents-indirects (*isAssociatedWith*).

Conclusion et perspectives

L'amélioration des performances d'extraction de relation sur PGxCorpus est une étape importante dans l'étude de l'influence des gènes sur la réponse aux médicaments dans le cadre du projet PractiKPharma.

Pour répondre à la problématique du stage, nous avons utilisé des méthodes d'apprentissage par transfert transductif de « type transfert de domaine » avec deux stratégies différentes (*frozen* et *fine-tuning*). Nous avons proposé trois architectures pour la stratégie frozen (BERT-CNN, BERT-CNN-RNN-parallèle, BERT-CNN-RNN), et deux architectures pour la stratégie *fine-tuning* (BERT-CNN, BERT-CNN-segmentation). Nous avons utilisé trois variantes de BERT (BERT de base, BioBERT, SciBERT) BioBERT, SciBERT qui sont plus adaptées pour les données biomédicales (PGxCorpus, ChemProt).

L' utilisation de l'architecture de l'état de l'art BERT et de ses variantes, avec l'extraction des informations locales latentes dans les vecteurs de représentation à travers un CNN, et l'exploitation de l'information structurale à travers la segmentation a permis à notre architecture **CNN-segmentation** de dépasser l'état de l'art avec des écarts significatifs : **1.73 %** amélioration absolue en F-micro par rapport à l'état de l'art Beltagy et al. [2] sur le corpus ChemProt et **32.77 %** amélioration absolue en F-macro par rapport au premier modèle Legrand et al.[15] proposé (*baseline model*). L'architecture CNN-segmentation montre aussi une amélioration au niveau de la stabilité de modèle (CNN-segmentation varie peu).

Dans nos analyses nous avons remarqué que le vocabulaire SciVocab est plus adapté au corpus biomédical que BERTVocab. La visualisation des données par la méthode T-SNE et ACP a révélé la difficulté de la classification en espaces des caractéristiques accordée à la stratégie *frozen* par rapport au *fine-tuning*. Nous avons aussi constaté via la visualisation que notre modèle semble suivre la hiérarchie des relations dans son apprentissage. Par l'analyse d'erreurs nous avons remarqué que nos modèles font plus d'erreurs sur les classes rares pour ChemProt et que la plupart des erreurs faites dans PGxCorpus sont des erreurs liées à la hiérarchie des relations.

Le transfert a prouvé sa capacité d'amélioration, l'application du transfert appliquée à plusieurs reprises avec des données différentes et des tâches différentes peut conduire à une amélioration supérieure à celle déjà atteinte. Nous avons entamé des tests consistant à appliquer un deuxième transfert du domaine en utilisant un modèle multi-têtes. Les premiers résultats obtenus pour PGxCorpus et ChemProt respectivement sont présentés dans les tableaux (A1) et (A2), et ils montrent un transfert négatif, qui peut être dû au décalage dans les distributions sources et cibles. Nous envisageons donc comme perspective d'appliquer un apprentissage par transfert de type « confusion de domaine » à la place

d' « adaptation de domaine », ce qui permettra d'imposer une certaine similitude entre les distributions sources et cibles et pourra améliorer les performances dans la tâche d'extraction des relations pharmacogénomiques grâce à une autre couche d'apprentissage par transfert.

Annexes

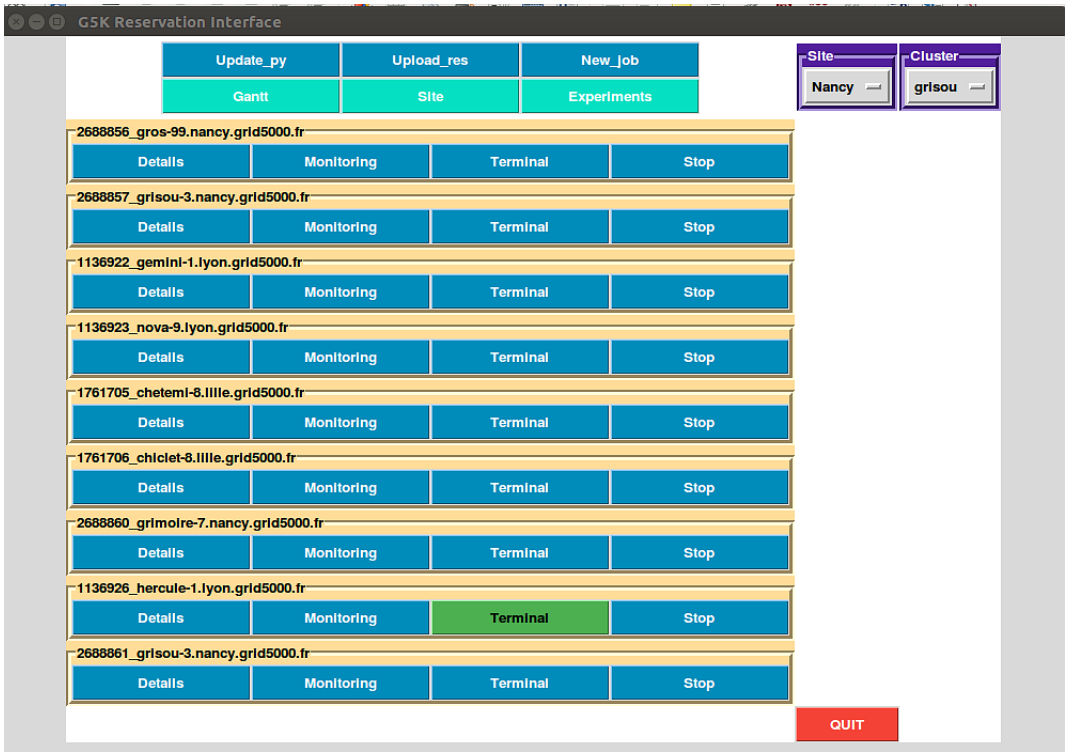

FIGURE A1 – L'interface graphique du scripte développé pour la réservation sur Grid5000

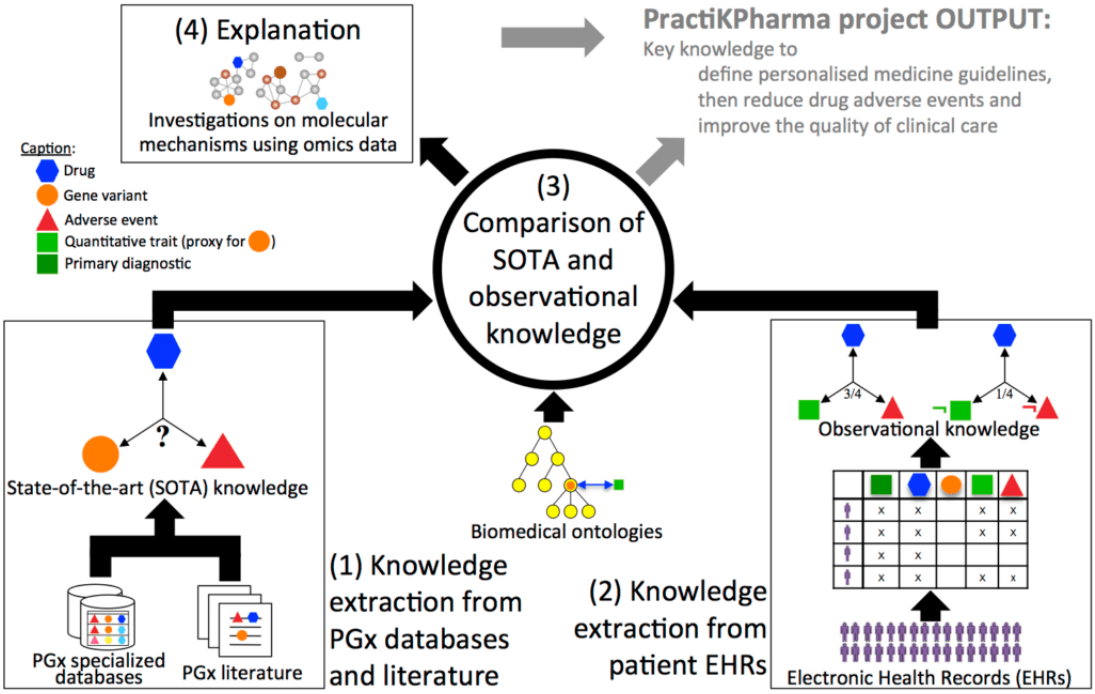

FIGURE A2 – Vue générale des taches étudiées par le projet ANR PractiKPharma

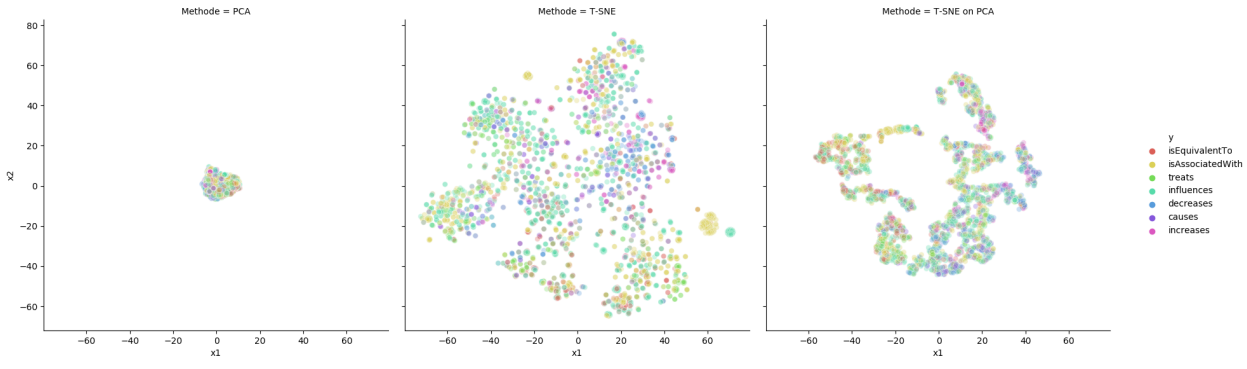

FIGURE A3 – Visualisation de l'espace des caractéristiques avant l'apprentissage sur le corpus PGxCorpus

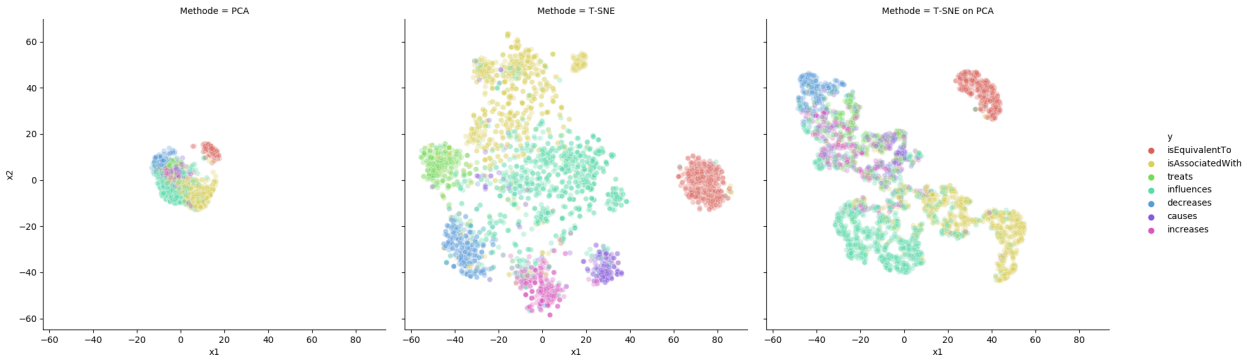

FIGURE A4 – Visualisation de l'espace des caractéristiques après 5 epoch d'apprentissage sur le corpus PGxCorpus

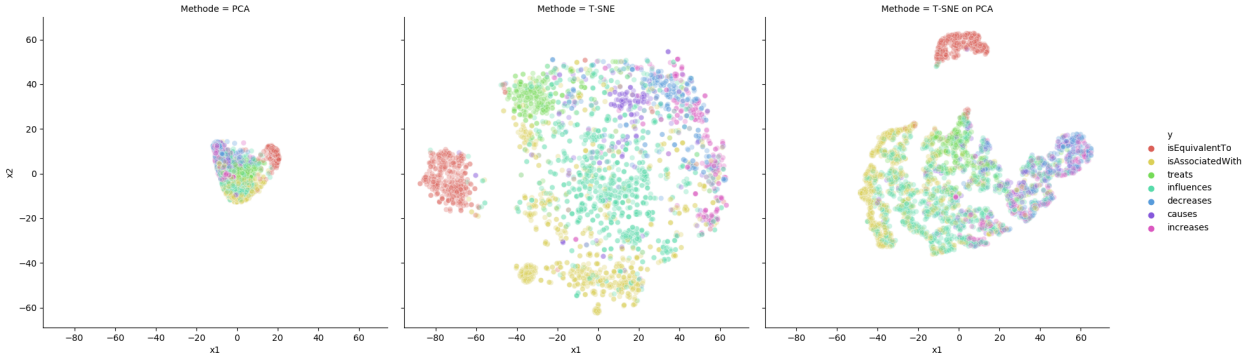

FIGURE A5 – Visualisation de l'espace des caractéristiques après un epoch d'apprentissage sur le corpus PGxCorpus

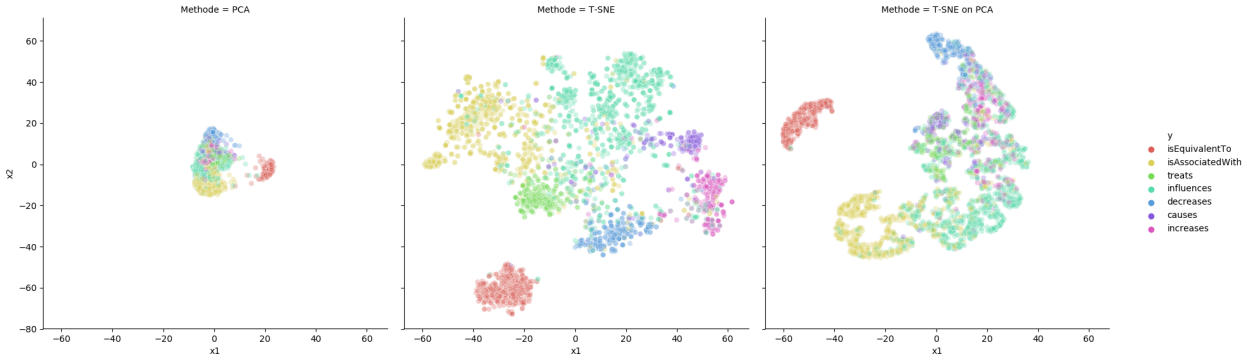

FIGURE A6 – Visualisation de l'espace des caractéristiques après 2 epoch d'apprentissage sur le corpus PGxCorpus

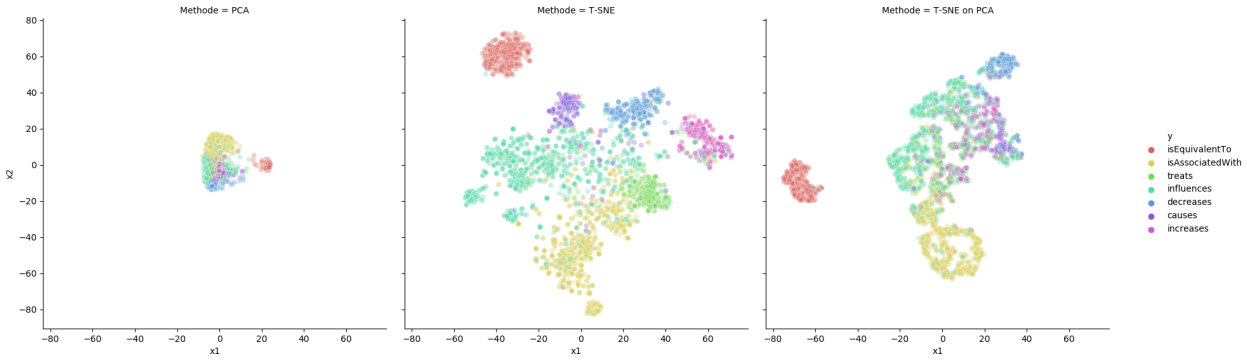

FIGURE A7 – Visualisation de l'espace des caractéristiques après 3 epoch d'apprentissage sur le corpus PGxCorpus

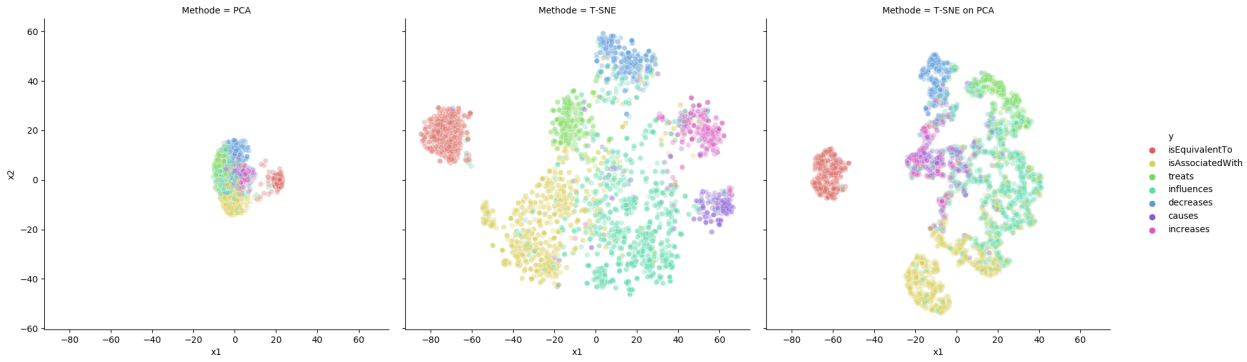

FIGURE A8 – Visualisation de l'espace des caractéristiques après 4 epoch d'apprentissage sur le corpus PGxCorpus

Chemprot Fine tune & domain transfer					
BERT model	Corpora	Precision	Recall	F-score	σ
BioBERT MLP	ChemProt only	**82.28**	**82.28**	**82.28**	**3.27**
	+PGxCorpus	60.12	60.12	60.12	20.8
SciBERT MLP	ChemProt only	**82.44**	**82.44**	**82.44**	1.47
	+SemEval	71.51	71.51	71.51	**1.39**
	+PGxCorpus	75.34	75.34	75.34	2.89
	+SNPPhena	74.23	74.23	74.23	4.59

TABLE A1 – les premiers résultats de l'application d'un deuxième transfert de domaine sur ChemProt

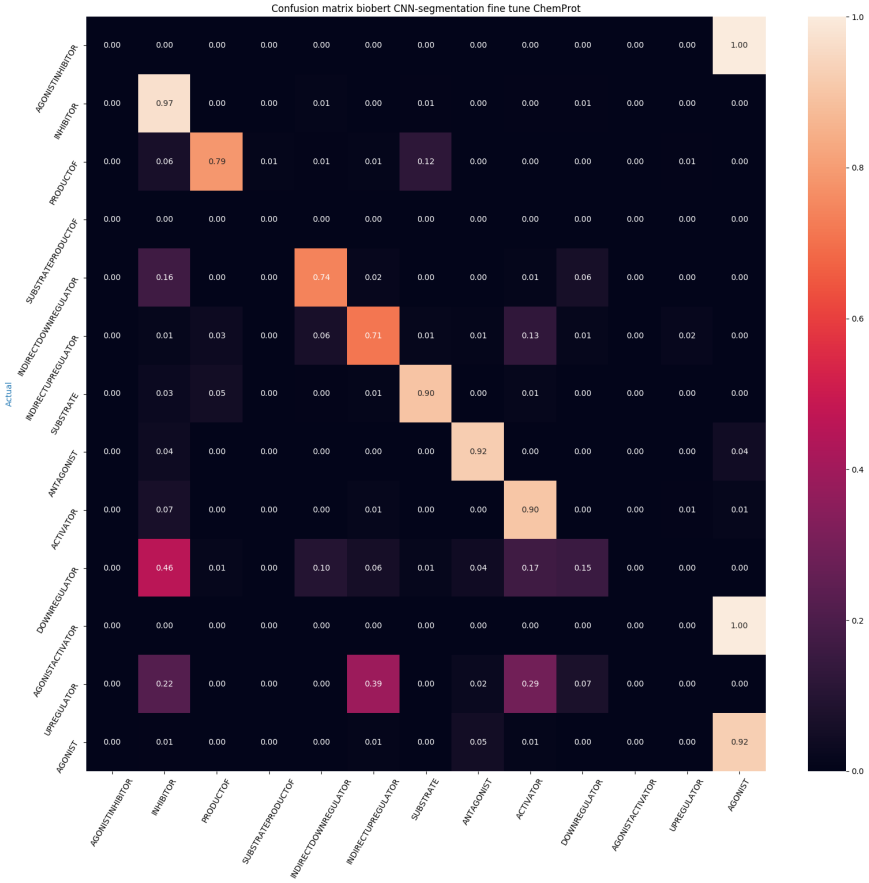

FIGURE A9 – La matrice de confusion du modèle BioBERT-CNN-segmentation sur ChemProt utilisant la stratégie fine-tune

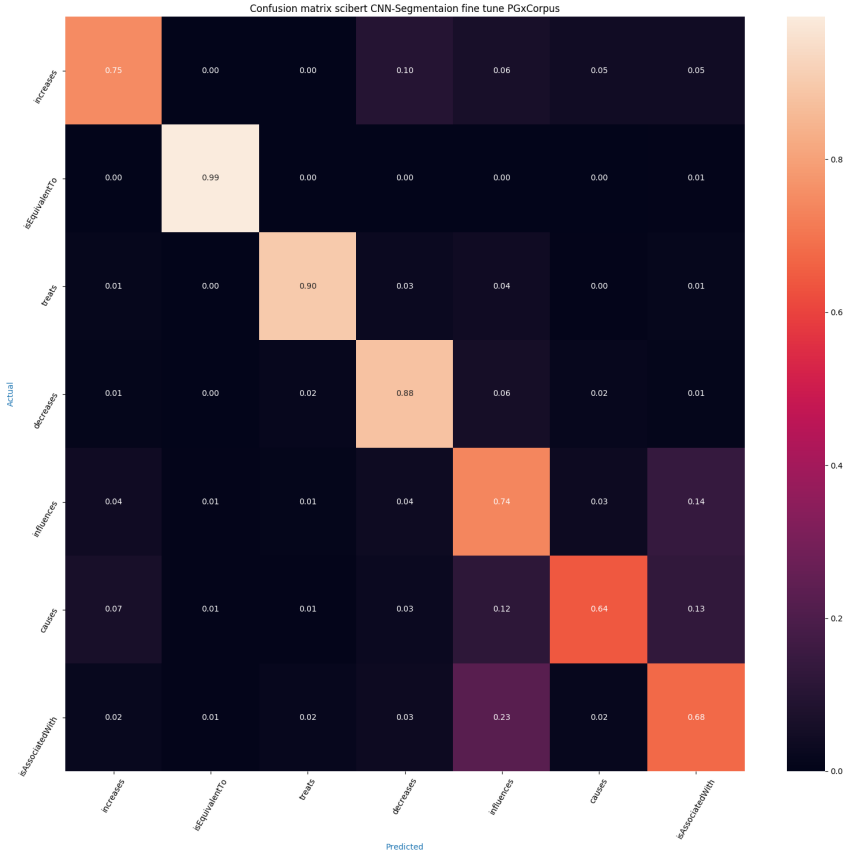

FIGURE A10 – La matrice de confusion du modèle SciBERT-CNN-segmentation sur PGx-Corpus utilisant la stratégie fine-tune

PGxCorpus Fine tune & domain transfer					
Model	Corpora	Precision	Recall	F-score	σ
BERT MLP	PGxCorpus only	**70.80**	**69.70**	**69.88**	4.03
	+SemEval	52.39	49.40	49.19	10.66
	+ChemProt	51.81	50.18	49.35	**2.50**
BioBERT MLP	PGxCorpus only	**73.49**	**68.84**	**70.47**	**5.87**
	+SemEval	51.61	44.03	44.37	6.21
	+ChemProt	54.87	45.69	47.54	7.69
SciBERT MLP	PGxCorpus only	**75.61**	**75.44**	**75.32**	2.11
	+SemEval	52.80	52.80	52.80	**1.34**
	+ChemProt	59.24	56.27	56.48	4.31
	+SNPPhena	69.82	67.51	68.28	4.56

TABLE A2 – les premiers résultats de l'application d'un deuxième transfert de domaine sur PGxCorpus

Bibliographie

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *CoRR*, 2016.

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. SCIBERT : A Pretrained Language Model for Scientific Text. *EMNLP/IJCNLP*, pages 3615–3620, 2019.

[3] Behrouz Bokharaeian, Alberto Diaz, Nasrin Taghizadeh, Hamidreza Chitsaz, and Ramyar Chavoshinejad. SNPPhenA : A corpus for extracting ranked associations of single-nucleotide polymorphisms and phenotypes from literature. *Journal of Biomedical Semantics*, 8(1) :1–13, 2017.

[4] Jun Chen and Robert Hoehndorf. Efficient Long-distance Relation Extraction With Dg-spanbert. *CoRR abs/2004.03636*, 2020.

[5] Yanping Chen, Kai Wang, Weizhe Yang, Yongbin Qing, Ruizhang Huang, and Ping Chen. A Multi-Channel Deep Neural Network for Relation Extraction. *IEEE Access*, 8 :13195–13203, 2020.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 999888 :2493–2537, November 2011.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*, (Mlm), 2018.

[8] Zhi Qiang Geng, Guo Fei Chen, Yong Ming Han, Gang Lu, and Fang Li. Semantic relation extraction using sequential and tree-structured LSTM with attention. *Information Sciences*, 509 :183–192, 2020.

[9] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 task 8 : Multi-way classification of semantic relations between pairs of nominals. *ACL 2010 - SemEval 2010 - 5th International Workshop on Semantic Evaluation, Proceedings*, (July) :33–38, 2010.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1 :448–456, 2015.

[11] Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[12] Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. ChemProt-3.0 : A global chemical biology diseases mapping. *Database*, pages 1–7, 2016.

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT : A Lite BERT for Self-supervised Learning of Language Representations. *ICLR*, pages 1–17, 2019.

[14] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT : A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4) :1234–1240, 2020.

[15] Joël Legrand, Romain Gogdemir, Cédric Bousquet, Kevin Dalleau, Marie Dominique Devignes, William Digan, Chia Ju Lee, Ndeye Coumba Ndiaye, Nadine Petitpain, Patrice Ringot, Malika Smaïl-Tabbone, Yannick Toussaint, and Adrien Coulet. PGx-Corpus, a manually annotated corpus for pharmacogenomics. *Scientific Data*, 7(1) :1–13, 2020.

[16] Joël Legrand, Yannick Toussaint, Chedy Raïssi, and Adrien Coulet. Syntax-based Transfer Learning for the Task of Biomedical Relation Extraction. *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, (Louhi) :149–159, 2019.

[17] Cheng Li and Ye Tian. Downstream Model Design of Pre-trained Language Model for Relation Extraction Task. *ArXiv*, pages 1–15, 2020.

[18] Qing Li, Lili Li, Weinan Wang, Qi Li, Jiang Zhong, and L Li. A comprehensive exploration of semantic relation extraction via pre-trained CNNs. *Knowledge-Based Systems*, 2020.

[19] Zhi Li, Jinshan Yang, Xu Gou, and Xiaorong Qi. Recurrent neural networks with segment attention and entity description for relation extraction from clinical texts. *Artificial Intelligence in Medicine*, 97(December 2018) :9–18, 2019.

[20] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 4 :2124–2133, 2016.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[22] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. *ACL (System Demonstrations)*, pages 55–60, 2014.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *International Conference on Learning Representations*, 2013.

[24] Pierre Monnin, Joël Legrand, Graziella Husson, Patrice Ringot, Andon Tchechmedjiev, Clément Jonquet, Amedeo Napoli, and Adrien Coulet. PGxO and PGxLOD : A reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison. *BMC Bioinformatics*, 20 :1–23, 2019.

[25] Esmaeil Nourani and Vahideh Reshadat. Association extraction from biomedical literature based on representation and transfer learning. *Journal of Theoretical Biology*, 488 :110112, 2020.

[26] Sinno Jialin Pan and Qiang Yang Fellow. A Survey on Transfer Learning. *IEEE Xplore*, pages 1–15, 2009.

[27] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove : Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[28] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies - Proceedings of the Conference*, 1 :2227–2237, 2018.

[29] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Improving Language Understanding by Generative Pre-Training. *OpenAI*, pages 1–10, 2018.

[30] Chanqin Quan, Lei Hua, Xiao Sun, and Wenjun Bai. Multichannel convolutional neural network for biological relation extraction. *BioMed Research International*, 2016.

[31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT : smaller, faster, cheaper and lighter. *CoRR*, pages 2–6, 2019.

[32] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies - Proceedings of the Conference*, 2 :464–468, 2018.

[33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Sutskever Ilya, and Ruslan Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 299(3-4) :345–350, 2014.

[34] Baochen Sun and Kate Saenko. Return of Frustratingly Easy Domain Adaptation. *AAAI*, 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 4(January) :3104–3112, 2014.

[36] C Tanriverdi. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *Journal of Science and Engineering*, 9 :69–76, 2006.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Uszkoreit Jakob, Llion Jones, Aidan N. Gomez, Kaiser Lukasz, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, (Nips), 2017.

[38] Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. Extracting multiple-relations in one-pass with pre-trained

transformers. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, (1) :1371–1377, 2020.

[39] Shanchan Wu and Yifan He. Enriching pre-trained language model with entity information for relation classification. *International Conference on Information and Knowledge Management, Proceedings*, pages 2361–2364, 2019.

[40] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation. *CoRR*, pages 1–23, 2016.

[41] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical Attention Networks for Document Classification. *HLT-NAACL*, pages 1480–1489, 2016.