Leveraging linguistic and semantic information for relation extraction from domain-specific texts

Anfu Tang

THESE DE DOCTORAT

NNT : 2023UPASG081

Leveraging linguistic and semantic information for relation extraction from domain-specific texts

Exploitation de l'information linguistique et sémantique pour l'extraction de relations à partir de textes en domaine spécialisé

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et Sciences du Numérique. Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **MaIAGE (Université Paris-Saclay, INRAE)**, sous la direction de **Claire Nédellec**, Directrice de recherche, le co-encadrement de **Pierre Zweigenbaum**, Directeur de recherche et **Louise Deléger**, Chargée de recherche

Thèse soutenue à Paris-Saclay, le 6 décembre 2023, par

Anfu Tang

Composition du jury

Membres du jury avec voix délibérative

Vincent Guigue Professeur, Université Paris-Saclay, AgroParistech	Président
Éric Gaussier Professeur, Université Grenoble Alpes	Rapporteur & Examinateur
Xavier Tannier Professeur, Sorbonne Université	Rapporteur & Examinateur
Laure Soulier Maîtresse de conférences, Sorbonne Université	Examinatrice

Titre: Exploitation de l'information linguistique et sémantique pour l'extraction de relations à partir de textes en domaine spécialisé

Mots clés: Traitement automatique des langues, Extraction de relations, Modèle de langue, Analyse syntaxique, Apprentissage profond, Base de connaissances

Résumé: Cette thèse a pour objet l'extraction d'informations relationnelles à partir de documents scientifiques biomédicaux, c'est-à-dire la transformation de texte non structuré en information structurée exploitable par une machine. En tant que tâche dans le domaine du traitement automatique des langues (TAL), l'extraction de relations sémantiques spécialisées entre entités textuelles rend explicite et formalise les structures sous-jacentes. Les méthodes actuelles à l'état de l'art s'appuient sur de l'apprentissage supervisé, plus spécifiquement l'ajustement de modèles de langue pré-entraînés comme BERT. L'apprentissage supervisé a besoin de beaucoup d'exemples d'apprentissages qui sont coûteux à produire, d'autant plus dans les domaines spécialisés comme le domaine biomédical. Les variants de BERT, comme par exemple PubMedBERT, ont obtenu du succès sur les tâches de TAL dans des textes biomédicaux. Nous faisons l'hypothèse que l'injection d'informations externes telles que l'information syntaxique ou la connaissance factuelle dans ces variants de BERT peut pallier le nombre réduit de données d'entraînement annotées. Dans ce but, cette thèse concevra plusieurs architectures neuronales basés sur PubMedBERT qui exploitent des informations linguistiques obtenues par analyse syntaxique ou des connaissances du domaine issues de bases de connaissance.

Title: Leveraging linguistic and semantic information for relation extraction from domain-specific texts

Keywords: Natural language processing, Relation extraction, Language model, Syntactic parsing, Deep learning, Knowledge base

Abstract: This thesis aims to extract relations from scientific documents in the biomedical domain, i.e. transform unstructured texts into structured data that is machine-readable. As a task in the domain of Natural Language Processing (NLP), the extraction of semantic relations between textual entities makes explicit and formalizes the underlying structures. Current state-of-the-art methods rely on supervised learning, more specifically the fine-tuning of pre-trained language models such as BERT. Supervised learning requires a large amount of examples that are expensive to produce, especially in specific domains such as the biomedical domain. BERT variants such as PubMedBERT have been successful on NLP tasks involving biomedical texts. We hypothesize that injecting external information such as syntactic information or factual knowledge into such BERT variants can compensate for the reduced number of annotated training data. To this end, this thesis consists of proposing several neural architectures based on PubMedBERT that exploit linguistic information obtained by syntactic parsers or domain knowledge from knowledge bases.

Synthèse en français

L'extraction de relations est une tâche importante du traitement automatique des langues (TAL) qui a beaucoup attiré l'attention des chercheurs. Dans cette thèse, nous nous concentrons sur l'extraction de relations à partir de textes biomédicaux. Les méthodes actuelles à l'état de l'art s'appuient sur de l'apprentissage supervisé, plus spécifiquement sur l'ajustement de modèles de langue pré-entraînés comme BERT. Cependant, l'annotation de textes biomédicaux destinés à former des exemples d'apprentissage est généralement coûteuse par rapport aux textes de domaine général, et des corpus de petite taille peuvent limiter la performance de l'ajustement. Bien que des variantes biomédicales de modèles telles que PubMedBERT aient été proposées et se soient avérées plus performantes que BERT sur des corpus biomédicaux, le problème de la taille des corpus persiste. Dans cette thèse nous cherchons des solutions différentes de l'augmentation du nombre d'exemples annotés pour améliorer la performance de l'extraction de relations basée sur PubMedBERT. Nous faisons l'hypothèse que l'injection d'informations externes telles que l'information syntaxique ou l'information de base de connaissance (BC) peut compenser l'insuffisance d'exemples annotés et améliorer la performance de PubMedBERT sur la tâche d'extraction de relations.

Pour injecter de l'informations syntaxiques dans PubMedBERT, nous exploitons deux types d'informations syntaxiques : l'information de dépendance et celle de constituant. Ces deux types d'informations syntaxiques peuvent être codés sous forme de données structurées en arbre. Nous proposons d'abord deux méthodes augmentées par l'information de constituant : CE-PubMedBERT et CT-PubMedBERT. L'intuition de CE-PubMedBERT est de segmenter d'abord une phrase en groupes de mots par le parcours en profondeur de l'arbre de constituant correspondant, puis de sommer les plongement de *wordpieces* pour calculer les plongements de groupes de mots. Ces plongements sont ensuite passés à des couches d'attention supplémentaires. L'avantage de CE-PubMedBERT est que les dépendances au niveau des *wordpieces* sont apprises par PubMedBERT, et les dépendances au niveau des groupes de mots sont apprises par des couches d'attention supplémentaires. CT-PubMedBERT consiste à sérialiser d'abord l'arbre de constituant, puis donner directement la séquence obtenue en entrée à PubMedBERT. Nous supposons que PubMedBERT peut apprendre des informations sur la structure de l'arbre de constituant utiles à la prédiction de relations sémantiques. Nous proposons ensuite une architecture d'apprentissage multi-tâche augmentée par l'information de dépendance : MTS-PubMedBERT. En plus de la tâche d'extraction de relations, l'ajustement de PubMedBERT est également guidé par deux tâches de sondage qui sont liées à la syntaxe. L'hypothèse est que l'ajustement sur les deux tâches de sondage injecte implicitement des informations syntaxiques dans PubMedBERT, et que ces informations sont utiles pour mieux prédire la relation sémantique. Les résultats expérimentaux sur trois corpus biomédicaux montrent que CT-PubMedBERT et MTS-PubMedBERT entraînent systématiquement une dégradation des performances, tandis que CE-PubMedBERT montre des améliorations dans certains cas. Cependant, une analyse plus approfondie n'attribue pas l'amélioration de CE-PubMedBERT à l'information syntaxique injectée.

Pour injecter l'information de base de connaissances dans PubMedBERT, nous exploitons les plongements de graphes de connaissances. Nous sélectionnons RotatE, une méthode de plongement de graphe, qui calcule un score de plausibilité pour chaque triplet (entité1, relation, entité2) dans la base de connaissance. Nous proposons ensuite la méthode KB-PubMedBERT, dans laquelle la représentation textuelle chaque paire d'entité candidate pour une relation calculée par PubMedBERT est concaténée avec un vecteur de score de plausibilité calculé par RotatE. Le vecteur de score de plausibilité contient les scores de plausibilité de toutes les relations de la base de connaissances. Notre hypothèse est que ces scores de plausibilité suggèrent des relations sémantiques possibles et donc biaisent PubMedBERT pour faire de meilleures prédictions. Notre méthode est capable d'exploiter des relations des bases de connaissance différentes des relations à prédire. Les résultats expérimentaux montrent que KB-PubMedBERT surpasse systématiquement PubMedBERT.

Dans cette thèse, nous avons proposé quatre méthodes augmentées par des informations externes. Les résultats expérimentaux sur trois corpus d'extraction de relations biomédicales démontrent que l'information syntaxique ne semble pas aider à améliorer la performance de PubMedBERT, tandis que l'information de base de connaissances s'est avérée utile. Cette conclusion n'est pas définitive et je propose plusieurs pistes pour les travaux futurs, y compris l'optimisation des architectures de modèles, le test de plus de corpus et de modèles de base, et la mise en production de KB-PubMedBERT.

To my parents.

Acknowledgements

First and foremost, I would like to express my gratitude to my thesis supervisors, Claire Nédellec, Louise Deléger, and Pierre Zweigenbaum, for their patience and support through the past three years. I also want to give my thanks to Robert Bossy, who actively participated in my thesis supervision.

I am immensely grateful to the members of my thesis committee, Éric Gaussier and Xavier Tannier, who accepted to be rapporteurs of my thesis, Laure Soulier and Vincent Guigue, who accepted to be my thesis examiners. Their insightful feedback motivated me to continue my research from various perspectives.

I am thankful for the financial support provided by Digicosme LabEx, which was crucial for me to complete my thesis.

A special word of thanks to the Saclay-IA platform of Université Paris-Saclay, for providing GPU resources through its Lab-IA cluster.

Lastly, I want to thank my parents for their unwavering support during my thesis.

My sincere appreciation goes out to every one of you. Without your support, this accomplishment would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information extraction (IE) is a topic that has been consistently studied by researchers in the domain of Natural Language Processing (NLP). Building efficient models that can automatically extract structured information such as named entities or relations from unstructured texts is usually less time-consuming and costly compared to manually extracting such information by domain experts. Named Entity Recognition (NER) and Relation Extraction (RE) are two important steps to succeed in an IE task. Generally, entities refer to task-specific nouns that we are interested in such as cities, person names, or more specifically, drugs and genes in biomedical texts. NER consists of locating these entities in the text and determining their entity types. For example, in the sentence "Obama was born in Honolulu", through NER the word *"Obama"* should be tagged as an entity of type *"person name"* entity and *"Honolulu"* should be tagged as an entity of type *"city"*. Relations between these entities are also of our interest. In the previous example, an efficient RE model should be able to identify the relation between *"Obama"* and *"Honolulu"* as a relation of type *"city of birth"*. A more complicated biomedical example is given in Figure 1.1. In this thesis, we focus on the relation extraction task in the biomedical domain.

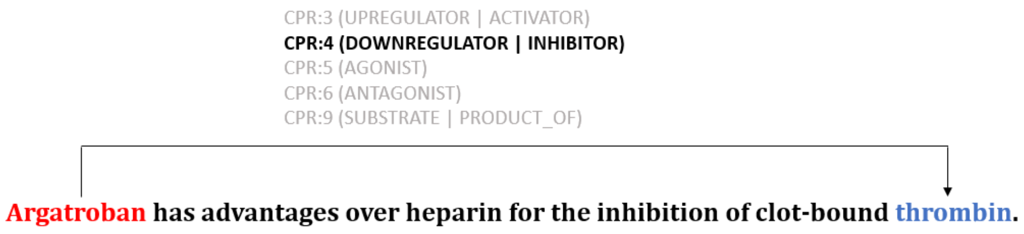

CPR:3 (UPREGULATOR | ACTIVATOR)
CPR:4 (DOWNREGULATOR | INHIBITOR)
CPR:5 (AGONIST)
CPR:6 (ANTAGONIST)
CPR:9 (SUBSTRATE | PRODUCT_OF)
Argatroban has advantages over heparin for the inhibition of clot-bound thrombin.

Figure 1.1: A example of relation extraction from ChemProt (Krallinger et al., 2017): A relation of CPR:4 (DOWNREGULATOR|INHIBITOR) type is annotated between an entity of chemical type "Argatroban" and an entity of gene type "thrombin".

With the rapid development of Artificial Intelligence (AI) and NLP, relation extraction methods have evolved in recent years. Progress of NLP models was made mainly by large Pre-trained Language Models (PLM) such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018, 2019; Brown et al., 2020) and T5 (Raffel et al., 2020). These pre-trained models are supposed to capture basic semantic information during the pre-training and adjusting their weights (fine-tuning) on downstream tasks guided by human-annotated labels has been proved to outperform previous state-of-the-art models on multiple NLP tasks including RE. However, new challenges appear as well with the emergence of these new techniques.

In this introductory chapter, we first briefly describe the challenges that we face in biomedical relation extraction, then more concretely, describe the problem from the perspective of machine learning. The thesis outline will be presented at the end of the chapter.

1.1 Problem Statement

In an information extraction system, relation extraction usually comes after named entity recognition. Since NER is not our focus, in this thesis we always assume that extracted entity information is provided. Given a text and entities, the objective of RE is to identify:

- if a semantic relation exists between given entities;
- the relation type in the case of existence.

In most cases, semantic relation types are pre-defined, For the example in Figure 1.1 there are 5 possible relation types (CPR:3-6, CPR:9). In practice, we usually combine the two goals mentioned above into one by adding a "NULL" relation indicating that no relation exists. Therefore, the objective of RE in Figure 1.1 is to correctly recognize the relation type between "Argatroban" and "thrombin" as CPR:4 out of six classes: *upregulator/activator*, *downregulator/inhibitor*, *angonist*, *antagonist*, *substrate/product_of* and *no_relation*. It is noteworthy that in our work we consider only binary relations, i.e. candidate relations are always between two entities.

In Figure 1.1, due to the existence of a trigger word *"inhibition"* that links *Argatroban* and *thrombin*, it is quite straightforward to classify the candidate relation as *"downregulator / inhibitor"*. However, due to the nature of natural language, real semantic relations may be more implicit and ambiguous. The difficulties are manifold as summarized in Table 1.1: (1) multiple candidate relations in one sentence require the RE model to be able to distinguish different entity pairs in the same context; (2) the RE model needs to understand negation; (3) the existence of coreference requires the model being able to detect long-range relations, e.g. in the last row of Table 1.1, "enzyme" is the coreference of "Delta 6 desaturase".

sentence	difficulty
This study thus demonstrates that the first administration of the recommended starting dose of irbesartan induces a greater and longer lasting Ang II receptor blockade than that of valsartan and losartan in normotensive subjects.	multiple candidate relations
Disodium cromoglycate does not prevent terbutaline-induced desensitization of beta 2-adrenoceptor-mediated cardiovascular in vivo functions in human volunteers.	negation
The use of Delta 6 desaturase (D6D) twice in the conversion of alpha-linolenic acid (ALA; 18:3n-3) to docosahexaenoic acid (DHA; 22:6n-3) suggests that this enzyme may play a key regulatory role in the synthesis and accumulation of DHA from ALA.	coreference

Table 1.1: Sentences with marked entities (ChemProt) and corresponding difficulties for relation extraction.

Challenges also come from fine-tuning large language models (LLMs) on biomedical texts. Since we focus on BERT (Devlin et al., 2019) in our work, we will use it as an example. The performance of BERT may be variable when it comes to domain-specific texts for two reasons: *"First, BERT is trained and tested mainly on datasets containing general domain texts (e.g. Wikipedia), it is difficult to estimate their performance on datasets containing biomedical texts. Also, the word distributions* *of general and biomedical corpora are quite different, which can often be a problem for biomedical text mining models."* (Lee et al., 2020) To handle this problem, researchers have proposed domain-specific BERT variants such as BioBERT (Lee et al., 2020) and SciBERT (Beltagy et al., 2019). These variants have been proved to outperform BERT on biomedical tasks.

For biomedical relation extraction, like many other domain-specific tasks, another challenge is the limited amount of high-quality annotated data. Though benchmark datasets such as ChemProt (Krallinger et al., 2017) exist, their size is relatively small ($< 100,000$ examples). This may also have a negative impact on the performance of LLMs.

To handle these challenges, we investigate the following questions:

1. is it possible to further improve the performance of BERT on biomedical relation extraction without increasing the size of annotated datasets?
2. what resources can be helpful for this improvement?

Syntax trees and knowledge bases (KB) are two resources that are commonly exploited for relation extraction. A syntax tree is a representation of the underlying grammatical structure of texts; while a knowledge base contains factual knowledge usually represented in triples. Both syntax and factual knowledge may play an important role for RE. However, though BERT (Devlin et al., 2019) is pre-trained on a large amount of texts, it is not explicitly pre-trained to capture the syntactic structure of texts or learn factual knowledge in KB. Motivated by this observation, there exist studies dedicated to enhancing BERT by integrating external information during the pre-training or the fine-tuning stage. Some studies (Sachan et al., 2021; Wang et al., 2021) show that integrating external knowledge improves the RE performance of neural models, while other studies (Puccetti et al., 2021; Jawahar et al., 2019; Luo, 2021; Petroni et al., 2019; Li et al., 2022) demonstrate that BERT has encode syntax or factual knowledge from pre-training. These two observations are not contradictory. Even BERT has implicitly learned syntax or factual knowledge, it does not mean that the encoded knowledge is comprehensive enough. It does not exclude the possibility to improve BERT by injecting external knowledge into it. Based on that, we make the principle hypothesis in this thesis: injecting syntactic information or factual knowledge may improve the RE performance of BERT, and negative impacts brought by data insufficiency can be compensated by integrated external knowledge.

1.2 Relation Extraction: A Supervised Text Classification Problem

Though there exist studies that take the relation extraction as a sequence tagging problem (Dai et al., 2019; Fu et al., 2019), in this thesis, we formulate relation extraction as a text classification task: given a training set containing observations $\mathit{\Gamma} = (x_i, y_i), i = 1, ...N$. where $x_i \in X$ refers to a sentence and a candidate pair of entities, $y_i \in Y$ refers to a label (semantic relation type), the goal of relation extraction is to find a function $f : X \rightarrow Y$ that minimizes $L(f(x), y)$ where X, Y denotes respectively the input and output space, and L is a loss function that measures the difference between predictions and true labels.

Since our work focus on pre-trained language models, we skip feature engineering, a crucial step in classical machine learning. We present directly encoder, a neural network that turns input data (texts in our case) to vector representations. Unlike some NLP tasks such as machine translation (Sutskever et al., 2014), for relation extraction we have no need of decoder. Therefore, we plot the general diagram of our supervised relation extraction model in Figure 1.2. In the figure, real lines represent the data flow, while dashed lines represent the back-propagated gradients flow that we use to update model weights (training). Details about back-propagation will be given in Chapter 2.

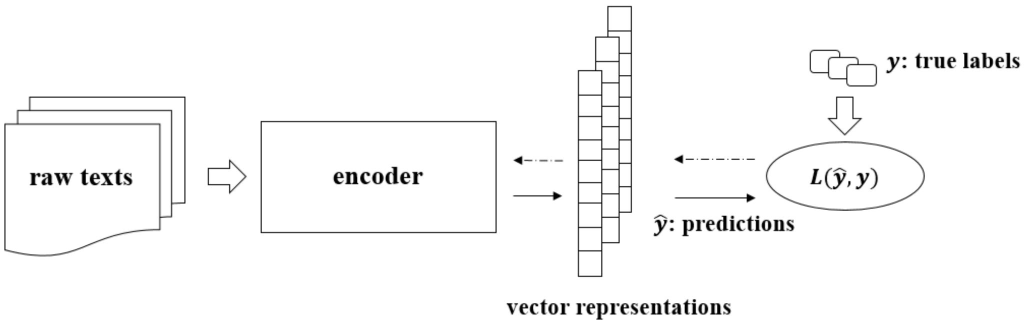


Figure 1.2: Diagram of supervised relation extraction model.

1.3 Thesis Outline

This introductory chapter presents the basic problem that our work is supposed to handle. We also present the motivation and the hypothesis that we make about injecting external information into BERT. In Chapter 2, we present background knowledge about biomedical RE, neural networks and BERT. As our work focuses on injecting two types of external information, syntactic information and KB information, we present respectively relevant work in Chapter 3 and Chapter 4. In Chapter 3, we first present existing syntax-related work, then syntax-enhanced models that we propose with related experiments. In Chapter 4, similarly we present existing KB-enhanced models and our proposed KB-enhanced model with related experiments. Chapter 5 concludes the thesis and presents perspectives for future work.

Chapter 2

Background and Related Work

As mentioned in the first chapter, our work focus on injecting external knowledge into BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) to improve the performance of BERT on biomedical relation extraction (RE) tasks. In this chapter, we present prerequisite knowledge before diving into details about BERT enhancements. This chapter is organized as follows:

1. A historical overview of biomedical RE methods (Section 2.1);
2. Neural network basics (Section 2.2);
3. Key components of BERT (Section 2.3-Section 2.6): WordPiece (Wu et al., 2016) tokenization; the base architecture Transformer (Vaswani et al., 2017); the theoretical base of the pre-training-fine-tuning framework; pre-training details; domain-specific BERT variants;
4. A general neural architecture for RE (Section 2.7);
5. Biomedical RE (Section 2.8): benchmark datasets, evaluation metrics, etc.

2.1 Historical Notes

Before Pre-trained Large Language Models (PLLM), relation extraction had been studied for decades. In this section, we briefly present precedent relation extraction methods in order to help better understand how relation extraction methods have evolved. Classical relation extraction methods can be divided into three categories: feature-based, kernel-based, or embedding-based. The evolution of relation extraction methods from feature or kernel-based models to end-to-end models conforms to the trend in the domain of Natural Language Processing (NLP): while NLP models used to rely on highly task-specific and manually-designed features, they have evolved to automatically extracting useful features from data using vector representations of words. Researchers increasingly search to build first a unified architecture to learn general linguistic knowledge, then adapt this architecture to specific tasks such as relation extraction or machine translation.

2.1.1 Feature-based Methods

A feature refers to a measurable characteristic that can be used as input to machine learning models. For example, if we want to predict the number of customers in a store, the weather might be an important feature (there might be more customers on rainy days than on sunny days). For feature-based relation extraction methods (Kambhatla, 2004; Jiang and Zhai, 2007; Rink and Harabagiu, 2010), we need to determine which features may be helpful and manually design these features. Popular features for RE are mainly lexical, syntactic, or semantic:

1. Bag-of-words (BOW) (Harris, 1954): the BOW consists of representing a text as a counter of words that keeps word frequencies while ignoring the order of words and the syntactic structure. This feature can be extended to prefixes, suffixes, or character N-grams (consecutive N characters).

2. Part-Of-Speech (POS) tags: in grammar, the POS tags are used to describe the syntactic categories of words. Common POS tags in English include noun, verb, adjective, adverb; for some languages POS tags include case or gramatical gender.

3. Named entity types: the category of entity is considered to be important for RE. For example, in the sentence of Figure 1.1, it is useful to know that *"argatroban"* is a chemical, and *"thrombin"* is a protein.

4. Syntactic structure: dependency analysis is a commonly used resource to extract syntactic features. In a dependency tree, words are linked by dependency relations based on the underlying syntactic structure. Details about dependency trees will be given in Chapter 3. A commonly used syntactic features is the Shortest Dependency Path (SDP). Given a dependency tree and a pair of entities e_1 and e_2, the possible semantic relationship $R(e_1, e_2)$ is *"almost exclusively concentrated in the shortest path between e1 and e2 in the undirected version of the dependency graph."* (Bunescu and Mooney, 2005) This hypothesis may not always hold, but it is often useful combine SDP with other features. An example of the dependency tree and SDP is shown respectively in Figure 2.1 and Table 2.1.

5. Hypernyms: in linguistics, the hypernym of a word w refers to a more general term than w. The relation between w and its hypernym is the lexical relation between labels of concepts in a knowledge base where the concepts would be linked by "is_a", or a type-subtype relation. For example, "dog" is the hypernym of "labrador", and "animal" is the hypernym of "dog".

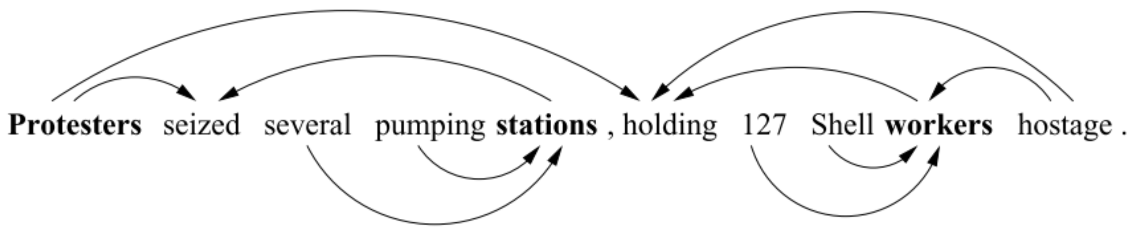

Protesters seized several pumping **stations** , holding 127 Shell **workers** hostage .

Figure 2.1: The dependency graph of a sentence (source: (Bunescu and Mooney, 2005)). Entities are marked in bold.

Relation	Shortest Dependency Path (SDP)
protesters AT stations	**protesters** $\longrightarrow$ seized $\longleftarrow$ **stations**
workers AT stations	**workers** $\longrightarrow$ holding $\longleftarrow$ protesters $\longrightarrow$ seized $\longleftarrow$ **stations**

Table 2.1: Corresponding SDPs from Figure 2.1.

In most cases, these features can be obtained by either data pre-processing (e.g. POS tags from

Feature name	Description
words	the words of both the arguments and all the words in between.
entity types	the entity type of both arguments.
overlap	the number of words (if any) separating the two arguments; the number of other entities in between; a flag indicating whether the two arguments are in the same noun phrase, verb phrase, or prepositional phrase.
dependency tree	the words, POS tags, and chunk labels of the words on which the arguments are dependent in the dependency tree.
constituency tree	similar to SDP, the shortest path connecting the two arguments in the constituency tree, and the path annotated with constituency tags.

Table 2.2: Common features used in feature-based methods (source: (Kambhatla, 2004)).

syntactic parsers) or existing resources (e.g. hypernyms from WordNet). Feature-based methods consist of directly using relevant features, adapting existing features to specific tasks, and exploiting new features. To give an insight into what features are used in practice, Table 2.2 lists the features used in (Kambhatla, 2004).

These features are extracted and then passed to a Maximum Entropy classifier to classify the candidate pairs as being in a relationship or not; if a relationship exists, they are also used to predict the relation type.

2.1.2 Kernel-based Methods

Unlike feature-based methods, instead of feeding features to a classifier, kernel-based methods (Haussler et al., 1999) consist of computing the similarity between examples. For relation extraction, the intuition is to put similar examples in the same category, i.e. an example similar to training examples containing a certain relation is considered more likely to contain the same relation. Different kernels (Zelenko et al., 2003; Zhou et al., 2007; Bunescu and Mooney, 2005; Valsamou, 2017; Tang et al., 2021) are proposed specifically for relation extraction. A good example of kernel-based methods is the method proposed by Bunescu and Mooney (2005). The method consists of first extracting SDPs from dependency trees, then obtaining word attributes as a set of features for each position along the SDP. After that, they compute the Cartesian product over the set of features of each position. Let us take the SDP in Figure 2.1 as an example. By replacing each word with corresponding attributes, we get the enriched SDP as shown in Figure 2.2. Computing the Cartesian product over the SDP generates 48 distinct features in Figure 2.3.

$$\begin{bmatrix}\text{protesters}\\ \text{NNS}\\ \text{Noun}\\ \textsc{Person}\end{bmatrix} \times [\rightarrow] \times \begin{bmatrix}\text{seized}\\ \text{VBD}\\ \text{Verb}\end{bmatrix} \times [\leftarrow] \times \begin{bmatrix}\text{stations}\\ \text{NNS}\\ \text{Noun}\\ \textsc{Facility}\end{bmatrix}$$

Figure 2.2: Enriched SDP from Figure 2.1 with word attributes at each position.

protesters	→	seized	←	stations
Noun	→	Verb	←	Noun
PERSON	→	seized	←	FACILITY
PERSON	→	Verb	←	FACILITY
		... (48 features)		

Figure 2.3: Distinct features extracted from the SDP in Figure 2.1.

However, exhaustively computing features for each SDP is not optimal because the number of features explodes with the increase of SDP length. Since in kernel-based methods we only care about the similarity between examples, we can directly compute the number of common features between each pair of SDPs rather than explicitly extracting high-dimensional features. The kernel function can be written as:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0 & m \neq n \\ \prod_{i=1}^{N} c(x_i, y_i), & m = n \end{cases} \tag{2.1}$$

where $\boldsymbol{x} = x_1 x_2 ... x_m$ and $\boldsymbol{y} = y_1 y_2 ... y_n$ represent two SDPs, and x_i denotes the feature set at the i-th position of $\boldsymbol{x}$ (same for y_i). $c(x_i, y_i)$ denotes the number of the common attributes between x_i and y_i. Since SVM (Cortes and Vapnik, 1995) is kernel-based, it suffices to insert the kernel function 2.1 into SVM for classification.

2.1.3 Embedding-based Methods

Word embeddings refer to continuous vector representations of words. In this section, we only focus on word embeddings learned by neural networks (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017). Compared to manually designed features of texts, word embeddings have the following advantages: (1) as by-products of training, they can be generated from large unannotated texts; (2) unlike manually created features of words, they are not fixed during the training, therefore can be better adapted to different NLP tasks; (3) they encode similarities between words. Embedding-based relation extraction methods use pre-trained word embeddings to initialize the vector representations of words. Similar to BERT-based methods that we will present later in this chapter, embedding-based methods apply the principle of transfer learning (Section 2.5), but the main difference is that for embedding-based methods, only word embeddings are pre-trained, while for BERT-based methods pre-trained weights of intermediate layers are also kept. In this section, we present embedding-based methods without illustrating how neural networks work, a brief introduction to neural networks will be given at the beginning of Section 2.2.

Mikolov et al. (2013) propose two architectures to learn word embeddings: CBOW (Continuous Bag Of Words) and Skip-gram. Given a sequence of words, CBOW consists of predicting the current word using previous and future words within a fixed window as context. Inversely, Skip-gram consists of predicting surrounding words of the current word within a certain range. Diagrams of both architectures are shown in Figure 2.4. Continuous word embeddings learned in this way encode contextual similarities between words, and in most cases using them in neural architectures for downstream NLP tasks (Zhang et al., 2017; Nguyen and Grishman, 2015; Xu et al., 2015) outperforms previous feature-based or kernel-based methods. To include word embeddings in a neural network, an embedding layer is added just behind the network input. If we denote the vocabulary by V and the embedding dimension by d, then the corresponding embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|V| \times d}$. Suppose that a sequence containing integer indexes of length N $(index_1, ..., index_N)$ is fed into an embedding layer, where $index_i$ indicates that the i-th word is indexed by $index_i$ in V, then the output of the embedding layer $\boldsymbol{O} \in \mathbb{R}^{N \times d}$. The function of an embedding layer is equivalent to lookup operations in $\boldsymbol{E}$, each $index_i$ is represented by the corresponding word vector without consideration of its position in the sequence.

Different types of neural architectures can be used in embedding-based RE methods. For example, Nguyen and Grishman (2015) proposes to use Convolutional Neural Network (CNN) for RE.

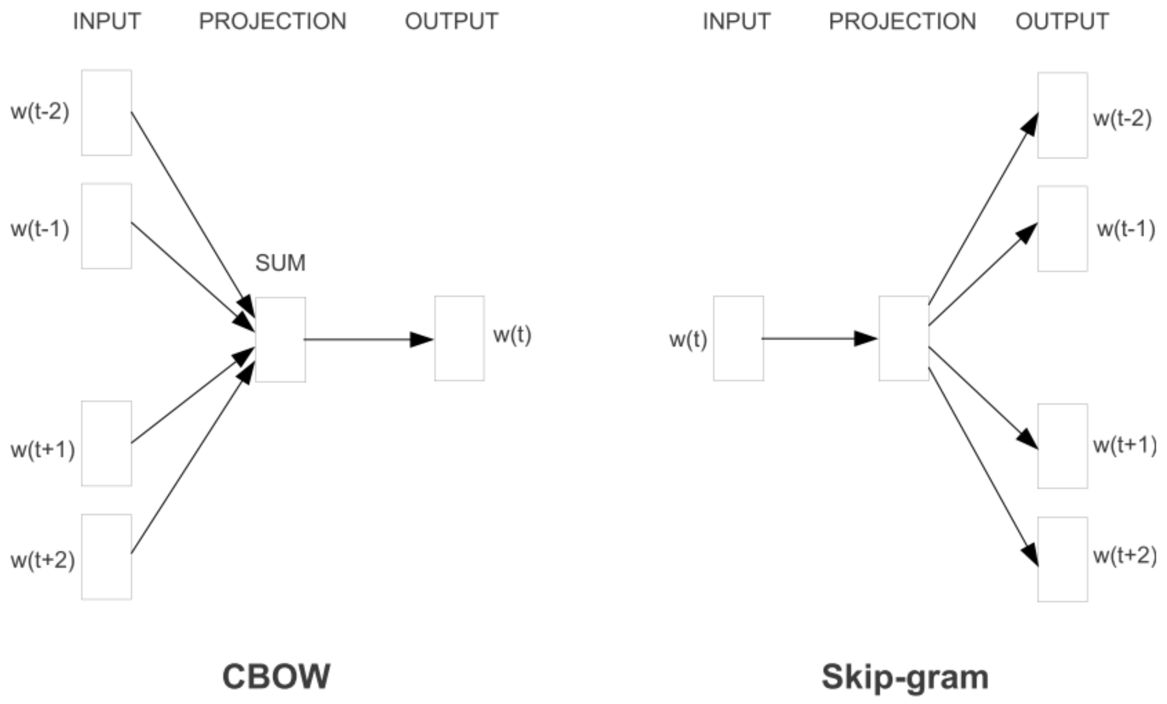


Figure 2.4: Two architectures that are used to learn word embeddings in word2vec (source: (Mikolov et al., 2013)): CBOW and Skip-gram.

The intuition of CNN is to apply the convolutional operator on input vector representations within a sliding window of fixed length. In the NLP scenario, the function of a convolutional operator is similar to extracting features over several continuous words. The CNN RE model proposed by (Nguyen and Grishman, 2015) consists of four parts:

1. An embedding layer to convert words to continuous word embeddings;
2. Convolutional layers to extract features of consecutive words;
3. A pooling layer to extract most important features;
4. A linear layer with a softmax activation to compute probabilities of relation types.

where a softmax function maps an input vector $\boldsymbol{x} = (x_1, ...x_N)$ to a probability vector containing values entre 0 and 1:

$$softmax(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.2}$$

This CNN RE model is very representative because it contains the most essential components of a neural RE model. Modifications can be made to adapt this architecture to relation extraction and obtain better performance: Zhang et al. (2017) propose to use Glove (Pennington et al., 2014) embeddings, introduce position embeddings and replace CNN by LSTM (Hochreiter and Schmidhuber, 1997); Miwa and Bansal (2016) propose to separately encode word sequences by LSTM layers and dependency subtrees by tree-structured LSTM layers; Xu et al. (2015) propose to input directly SDPs and apply LSTM layers to encode different types of information along SDPs such as Part-Of-Speech (POS) tags, dependency relations, etc.

2.2 Neural Networks Basics

In the previous section, we introduced embedding-based methods that belong to the category of neural networks. Analogous to the human brain, a neural network contains connected nodes, and each node receives signals from others. Signals from different nodes are not equally important, therefore edges between nodes should have different weights. The simplest case of a neural network is a one-layer fully-connected network as shown in Figure 2.5; fully-connected indicates that nodes at the output are connected with every input node. Input to this network $\boldsymbol{x} = (x_1, x_2, x_3)$ is a 3-D vector; output $\boldsymbol{y} = (y_1, y_2)$ is a 2-D vector; w_{ij} represents the weight of the edge connecting x_i to y_j. The value of y_j can be written as a weighted sum: $y_j = \sum_{i=1}^{3} x_i w_{ij}$. w_{ij} can be packed into a weight matrix $\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix}$ then we have: $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$. Since this transformation is linear, we want to add non-linearity to the network to expand its expressivity. A common solution is to add a non-linear function σ named "activation" onto the layer output such that $\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x})$. According to the universal approximation theorem (Hornik et al., 1989), theoretically a multi-layer neural network with non-linear activation can approximate any function. Based on that, we create a Multi-Layer Perceptron (MLP) by stacking fully-connected layers with non-linear activation. The architecture of an MLP with two hidden layers is shown in Figure 2.6. There exist different types of neural networks, each designed for different scenarios such as Convolutional Neural Networks (Goodfellow et al., 2016b), LSTM (Hochreiter and Schmidhuber, 1997), Transformer (Vaswani

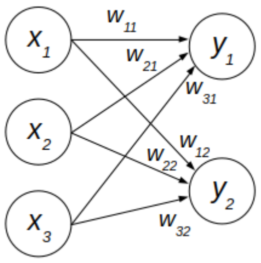


Figure 2.5: A one-layer fully-connected neural network: three-dimension input and two-dimension output.

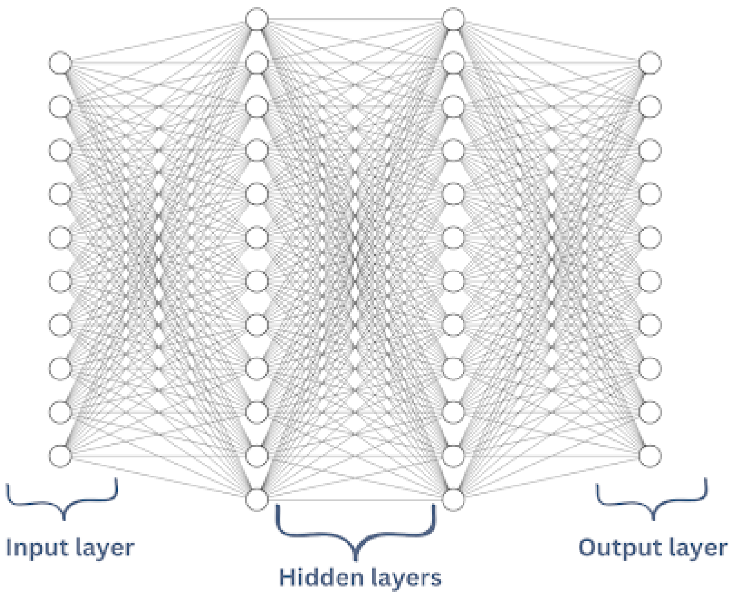


Figure 2.6: An example of Multi-Layer Perceptron with 2 hidden layers.

et al., 2017). Though these neural networks differ in architectures, they are similar in essence: multi-layer structure; layers composed of neurons with different connectivity. For example, the convolutional neural network abandons the full connectivity of MLP and changes to connect nodes only to adjacent nodes of the last layer within a certain range.

When we apply a neural network to a specific task, we need to adjust the weights inside the network to certain values such that the output is close to or equal to our expectations. This process of weight value adjustment is known as “training”. To understand the training process, we need to respond to the following questions:

- How can we measure the dissimilarity between the output and our expectation?

- How can we automatically adjust weight values to "adapt" a neural network to a specific task?

- Does randomness affect the training process? How can we mitigate its negative impact?

In the rest of this section, we focus on the three questions above.

2.2.1 Loss Function

The goal of training a neural network is to minimize the dissimilarity between its output (prediction) and our expectation (normally presented in the form of per-example labels). To achieve that, a loss function is defined to describe this dissimilarity mathematically. Suppose that a neural network expresses a function $f : \boldsymbol{x} \to \hat{\boldsymbol{y}}$, where $\boldsymbol{x} \in \mathbb{R}^M$ denotes the input vector and $\hat{\boldsymbol{y}} \in \mathbb{R}^N$ denotes the output vector containing probabilities of N classes. Then $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes the parameters of the neural network. We use $\hat{\boldsymbol{y}}$ here to distinguish the prediction from the one-hot vector encoding the ground truth $\boldsymbol{y}$, which is the expected output. A loss function $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ maps a pair of vectors to a real value representing the dissimilarity. Thus the training process is equivalent to solving an optimization problem:

$$\min_{\boldsymbol{\theta}} L(f(\boldsymbol{x}; \boldsymbol{\theta}), \boldsymbol{y}) \tag{2.3}$$

In information theory, cross entropy is commonly used to compute the dissimilarity of two probability distributions. In our case, $\hat{\boldsymbol{y}} = (\hat{y}_1, ..., \hat{y}_N)$ already contains probabilities; $\boldsymbol{y} = (y_1, ..., y_N)$ contains discrete values of either 0 or 1. By taking $\boldsymbol{y}$ as a vector of extreme probability values, the per-example cross entropy loss between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ over N classes is computed by:

$$CrossEntropy(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{2.4}$$

2.2.2 Optimization

Nearly all training processes of neural networks consist of solving Equation 2.3. The difficulty of this optimization problem largely depends on the form of f. Unfortunately, since in a neural network f

is implicitly expressed, mathematically it is hard to give an analytical solution. Intuitively, popular solutions consist of iteratively adjusting $\boldsymbol{\theta}$ such that $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ decreases gradually until either (1) the loss reaches a value that is sufficiently low; (2) a maximum number of iterations is executed; (3) the loss value stops decreasing for a long time. In this subsection, we focus on a popular iterative algorithm to gradually decrease $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$.

2.2.2.1 Gradient Descent

Gradient descent is one of the most widely used first-order gradient-based iterative optimization algorithms to find a local minimum of a differentiable function. Given a differentiable f, the gradient at a certain point p is a vector $\nabla f(p)$ whose direction is the direction in which f increases the most and whose magnitude equals the rate of increases, i.e. $df = |\nabla f(p)| \cdot dr$, where df represents the increase of f and dr represents a displacement.

The intuition of gradient descent consists of always moving the point p in the opposite direction of $\nabla f(p)$ until the magnitude of the gradient is smaller than a certain threshold, i.e. $|\nabla f(p)| < \epsilon$. Take the quadratic function $f(x) = x^2$ for example, the corresponding trajectory of points obtained by gradient descent is plotted in Figure 2.7. Starting from a random point p_0, we move the point always in the opposite direction of the current gradient. As shown in the figure, we approach the minimum (red point) gradually. The magnitude of the gradient becomes smaller and smaller as in the present case $|\nabla f(x_i)| = \frac{df}{dx_i} = 2 * x_i$, $\lim\limits_{x_i \to 0} |\nabla f(x_i)| = 0$ (same for another initial point p_0'). The point p_i is thus expected to reach the red point when the process stops.

We can further improve this process by controlling the amount of "descent" at each step (described by the term "step size") in order to avoid oscillation between points around the minimum. If we do not control the step size, it is possible that p_i repeatedly enters the second quadrant and then jumps back into the first quadrant. This oscillation may repeat infinitely such that we can never reach the red point. To avoid a too large or too small step size, we can add a hyperparameter γ. For the sequence of variables $(x_0, ..., x_i, ...)$ corresponding to the sequence of points obtained by gradient descent $(p_0, ..., p_i, ...)$, let: $x_{i+1} = x_i - \gamma |\nabla f(x_i)|$. The hyperparameter γ is also known as the "learning rate" as it controls the speed of weight updates. Setting an appropriate value for the

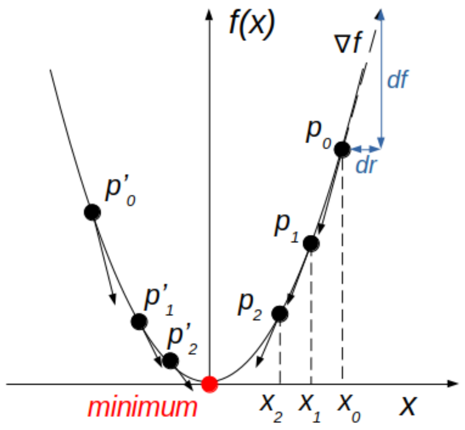


Figure 2.7: Illustration of gradient descent: quadratic function.

learning rate is crucial for succeeding in training a neural network: if the learning rate is too small, it may cost too much time for the network to converge, while a too large learning rate may lead to divergence, i.e. the training process stops at a point far from the minimum.

2.2.2.2 Back-Propagation

We have seen how gradient descent works in a quite simple scenario of a single-variable quadratic function. Though the principle of gradient descent applies to more complicated functions, there remains an unsolved problem unsolved: for neural networks, can we effectively calculate the gradient of the loss L with respect to (w.r.t.) thousands or even millions of weights? Since in a neural network, all trainable weights are variables of the loss function, we need to compute the gradient of a multi-variate function. It is reasonable to think that we can calculate the derivative of L to each variable, then "descend" simultaneously in all directions with corresponding step sizes. The problem is thus turned into computing the derivative of L w.r.t. multiple variables.

Due to the multi-layer structure of neural networks as shown in Figure 2.6, intuitively we want to factorize the derivative $\frac{\partial L}{\boldsymbol{\theta}^{(l)}}$ as a combination of derivatives like $\frac{\partial L}{\partial \boldsymbol{\theta}^{(l+1)}} \frac{\partial \boldsymbol{\theta}^{(l+1)}}{\partial \boldsymbol{\theta}^{(l)}}$, where $\boldsymbol{\theta}^{(l)}$ denotes the weights of the l-th layer. Mathematically this intuition has a theoretical basis: in calculus, if a variable z depends on the variable y, and y depends on the variable x, then y is an intermediate variable between x and z. The chain rule states that:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \tag{2.5}$$

Equation 2.5 can be extended to vector and matrix variables, thus it applies to neural networks. This allows us to calculate $\frac{\partial L}{\partial \boldsymbol{\theta}^{(l)}}$ for each layer and propagate gradients of previous layers to the current layer. Since the gradient propagation is backward (starting from the loss value), this method is known as "backpropagation".

We take MLP as an example to illustrate how backpropagation actually works. Denote the output of the l-th layer after activation σ by $\boldsymbol{a}^{(l)} \in \mathbb{R}^{d_l}$, and the output before σ by $\boldsymbol{z}^{(l)} \in \mathbb{R}^{d_l}$. The feed-forward information flow can be written as:

$$\begin{aligned} \boldsymbol{z}^{(l+1)} &= \boldsymbol{W}^{(l+1)}\boldsymbol{a}^{(l)} + \boldsymbol{b}^{(l+1)} \\ \boldsymbol{a}^{(l+1)} &= \sigma(\boldsymbol{z}^{(l+1)}) \end{aligned} \tag{2.6}$$

Suppose that $\frac{\partial L}{\partial \boldsymbol{z}^{(l)}} = \boldsymbol{\delta}^{(l)} \in \mathbb{R}^{d_l}$, $\boldsymbol{\delta}^{(l)}$ can be used to represent how sensitive the loss value is to neurons in the l-th layer. Compute $\boldsymbol{\delta}^{(l)}$ by the chain rule:

$$\begin{aligned} \boldsymbol{\delta}^{(l)} &= \frac{\partial L}{\partial \boldsymbol{z}^{(l)}} \\ &= \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}}\frac{\partial \boldsymbol{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}}\frac{\partial L}{\partial \boldsymbol{z}^{(l+1)}} \\ &= \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}}\frac{\partial \boldsymbol{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}}\boldsymbol{\delta}^{(l+1)} \end{aligned} \tag{2.7}$$

Recall that in vector calculus (Deisenroth et al., 2020), the derivative of a column vector $\boldsymbol{y} = y_1 ... y_m$ w.r.t. to a column vector $\boldsymbol{x} = x_1 ... x_n$ is (here we use the denominator layout, i.e. elements in $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ are arranged in the same order as $\boldsymbol{x}$):

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \begin{bmatrix} (\frac{\partial \boldsymbol{y}}{\partial x_1})^\top \\ \vdots \\ (\frac{\partial \boldsymbol{y}}{\partial x_n})^\top \end{bmatrix} = \begin{bmatrix} \frac{dy_1}{dx_1} & \cdots & \frac{dy_m}{dx_1} \\ \vdots & \ddots & \vdots \\ \frac{dy_1}{dx_n} & \cdots & \frac{dy_m}{dx_n} \end{bmatrix} \tag{2.8}$$

As σ is a point-wise activation, each element in $\boldsymbol{a}^{(l)}$ depends only on the element at the same position in $\boldsymbol{z}^{(l)}$. $\frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}}$ is thus a diagonal matrix:

$$\frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} = \begin{bmatrix} \frac{d\sigma(z_1^{(l)})}{dz_1^{(l)}} & \cdots & \frac{d\sigma(z_1^{(l)})}{dz_{d_l}^{(l)}} \\ \vdots & \ddots & \vdots \\ \frac{\sigma(dz_{d_l}^{(l)})}{dz_1^{(l)}} & \cdots & \frac{\sigma(dz_{d_l}^{(l)})}{dz_{d_l}^{(l)}} \end{bmatrix} = \begin{bmatrix} \sigma'(z_1^{(l)}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma'(z_{d_l}^{(l)}) \end{bmatrix} = diag(\sigma'(\boldsymbol{z}^{(l)})) \tag{2.9}$$

where diag represents a diagonal matrix and σ' denotes the derivative function of σ.

From Equation 2.6 we have:

$$\begin{aligned} z_i^{(l+1)} &= \sum_{k=1}^{d_l} W_{ik}^{(l+1)} a_k^{(l)} \\ \frac{dz_i^{(l+1)}}{da_j^{(l)}} &= W_{ij}^{(l+1)} \end{aligned} \tag{2.10}$$

Combine Equation 2.8 and Equation 2.10:

$$\frac{\partial \boldsymbol{z}^{(l+1)}}{\partial \boldsymbol{a}^{(l)}} = \begin{bmatrix} \frac{dz_1^{(l+1)}}{da_1^{(l)}} & \cdots & \frac{dz_{d_{l+1}}^{(l+1)}}{da_1^{(l)}} \\ \vdots & \ddots & \vdots \\ \frac{dz_1^{(l+1)}}{da_{d_l}^{(l)}} & \cdots & \frac{dz_{d_{l+1}}^{(l+1)}}{da_{d_l}^{(l)}} \end{bmatrix} = \begin{bmatrix} W_{11}^{(l+1)} & \cdots & W_{d_{l+1}1}^{(l+1)} \\ \vdots & \ddots & \vdots \\ W_{1d_l}^{(l+1)} & \cdots & W_{d_{l+1}d_l}^{(l+1)} \end{bmatrix} = (\boldsymbol{W}^{(l+1)})^\top \tag{2.11}$$

Substitute Equation 2.9 and Equation 2.11 into Equation 2.7, we have:

$$\boldsymbol{\delta}^{(l)} = \sigma'(\boldsymbol{z}^{(l)}) \odot ((\boldsymbol{W}^{(l+1)})^\top \boldsymbol{\delta}^{(l+1)}) \tag{2.12}$$

where $\odot$ denotes the element-wise multiplication. We can see that in Equation 2.12, $\boldsymbol{\delta}^{(l)}$ is dependent on the weight of the next layer $\boldsymbol{W}^{(l+1)}$ and $\boldsymbol{\delta}^{(l+1)}$, and no information from previous layers is needed.

This demonstrates that the gradient flow is backward.

2.2.2.3 Minibatch Stochastic Methods

We have described generally how gradient descent works and how to compute gradients in a neural network effectively. There exists another problem: How do we feed training examples to a neural model? Neither feeding examples one by one nor feeding all examples at once is a good solution: feeding one single example at a time will slow down the training process, and a single example may provide an inaccurate estimation of gradients; feeding all examples at once is also problematic due to memory limitation.

An effective algorithm to handle this problem is Stochastic Gradient Descent (SGD). As an extension of gradient descent, the principle of SGD is to sample a small subset of examples to perform gradient steps instead of using the whole dataset. The small subset of examples extracted at each time is also known as “minibatch”.

The loss function over a training set $\mathbb{T} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)\}$ can be decomposed as a sum of per-example function $L(f(\boldsymbol{x_i}; \boldsymbol{\theta}), y_i)$ (as in Equation 2.3):

$$L_{\mathbb{T}} = \frac{1}{N} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) \tag{2.13}$$

The same applies to the gradient over $\mathbb{T}$:

$$\nabla_{\boldsymbol{\theta}} L_{\mathbb{T}} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) \tag{2.14}$$

It is possible to estimate the gradient over $\mathbb{T}$ using the gradient over a minibatch of examples $\mathbb{B} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_M, y_M)\}$. The estimate of the gradient can be formed as:

$$\boldsymbol{g} = \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i) \tag{2.15}$$

Then the SGD algorithm makes a gradient step:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \gamma \boldsymbol{g} \tag{2.16}$$

In deep learning, a complete iteration over the training set is known as an "epoch". In practice, we usually shuffle the training examples once at the beginning of each epoch, then extract a minibatch in order for each gradient step.

Some studies propose enhancements to SGD (Duchi et al., 2011; Kingma and Ba, 2014; Loshchilov and Hutter, 2017). Among different optimization algorithms, Adam (Kingma and Ba, 2014) is the most commonly used optimizer for training large language models as in studies such as (Devlin et al., 2019; Gu et al., 2021; Lee et al., 2020). We omit details about Adam here since illustrating these advanced optimization algorithms is beyond the scope of neural network basics. In our work, if no special explanation is given, we always use Adam as the optimizer in our experiments.

2.2.3 Seeds and Ensembling

The training process of neural networks is not deterministic, i.e. given the same input and hyperparameters (fixed parameters other than trainable weights of a neural model such as learning rate, and number of hidden layers), the weights of a neural network are different after each training process. The randomness comes from several aspects (Bouthillier et al., 2021):

- Random initialization of weights;
- Random sampling of minibatches;
- Certain strategies introduce extra randomness, e.g. Dropout (Srivastava et al., 2014) randomly chooses a certain number of neurons to remove temporarily for regularization purposes.

Randomness can be beneficial: a good random weight initialization may lead to better convergence; gradient estimation from randomly sampled examples can help optimizers avoid local minima; strategies like Dropout can increase the robustness of neural networks and prevent overfitting. However, the randomness also makes the performance of neural models unstable, it is thus crucial to reduce the variance of predictions obtained from neural models. One solution to handle the high variance of neural networks is the ensembling technique. Ensembling refers to combining predictions of various base models in order to create a robust output. In our work, we choose hard voting as our ensembling strategy, i.e. we always choose the class predicted by most models as the final prediction.

Another problem raised by randomness is that it is hard to reproduce experimental results. A commonly used solution is to use fixed random seeds at the beginning of training. A random seed is an integer that is used to initialize a pseudorandom number generator. Due to the nature of number-generating algorithms, a number sequence generated is fully determined by the initial seed. Since most deep learning frameworks like Pytorch (Paszke et al., 2019a) use the pseudorandom number generator in their implementations, in practice the reproducibility of neural model predictions can be guaranteed by simply setting the same seeds for different training processes. We follow this convention in our experiments.

2.3 Tokenization

Tokenization is an important data preprocessing step in NLP. Most NLP methods treat a piece of text as a sequence of tokens. A token refers to a common sub-sequence of text, which can be a sequence of characters, a symbol, a punctuation mark, etc. Traditional tokenizers such as Stanford CoreNLP (Manning et al., 2014) use a collection of rules to segment texts into English words or symbols. However, since the size of word vocabulary is usually limited due to computation complexity limitations, using complete words as tokens raises the Out-Of-Vocabulary (OOV) problem: tokens encountered in the inference stage may not exist in the vocabulary (the set of words that have been used for training), and these unrecognized tokens may degrade the performance of NLP models. Another challenge for embedding-based methods is that even if we build a vocabulary that is big enough to accommodate all words, vectors of low-frequency words will not be fully trained. To handle these problems, tokenization algorithms based on sub-words are proposed. In this thesis, we focus on the WordPiece tokenization (Wu et al., 2016).

As indicated in (Wu et al., 2016): *"The WordPiece model is generated using a data-driven approach to maximize the language-model likelihood of the training data, given an evolving word definition."* Given a corpus $\mathcal{C}$ and a desired number of word pieces N, the WordPiece algorithm works as follows (Schuster and Nakajima, 2012):

1. Initialize the word piece vocabulary with five special symbols used in BERT: [CLS], [SEP], [UNK], [PAD], [MASK]. The function of special symbols will be presented in Section 2.6.

2. Tokenize the corpus first into words. For each word, add the first letter to the vocabulary; add all subsequent letters prefixed by "##". If there are non-English words or other symbols, add their ASCII values;

3. Build a language model on $\mathcal{C}$ using the initialized word piece vocabulary;

4. Generate a new word piece by combining two word pieces (except the 5 special symbols) in the current vocabulary. Choose the word piece that increases the likelihood over $\mathcal{C}$ the most when added to the vocabulary;

5. Repeat 2 until N is reached or the likelihood increase falls below a certain threshold.

"Wordpieces achieve a balance between the flexibility of characters and efficiency of words." (Wu et al., 2016). With WordPiece tokenization OOV words can be segmented into word pieces and these word pieces are guaranteed to be frequent enough. Wu et al. (2016) also reports that using a total vocabulary between 8k and 32k word pieces achieves both good accuracy on NLP tasks and fast decoding speed. The number of word pieces used in BERT and its variants drops in this (8k,32k) interval. For example, there are 30k word pieces for BERT (Devlin et al., 2019); 28,996 for BioBERT (Lee et al., 2020); 28,895 for PubMedBERT (Gu et al., 2021).

To give an insight into how WordPiece tokenization works, take the sentence from Figure 1.1 as an example. After tokenization (with the tokenizer from PubMedBERT) we obtain a word piece sequence as follows: [CLS], arg, ##atr, ##oba, ##n, has, advantages, over, heparin, for, the, inhibition, of, clot, , bound, thrombin, ., [SEP].

2.4 Transformer

Since its invention in 2017, Transformer (Vaswani et al., 2017) has been ubiquitous as the base architecture for most Large Language Models (LLMs). Unlike the original Transformer (Vaswani et al., 2017) with an encoder-decoder architecture (Sutskever et al., 2014), subsequent LLMs make slight modifications. For example, BERT (Devlin et al., 2019) is an encoder-only Transformer, while GPT (Radford et al., 2018, 2019; Brown et al., 2020) is a decoder-only Transformer. In this section, we explain the attention mechanism and present the encoder-decoder architecture of Transformer.

2.4.1 Attention Mechanism

Bahdanau et al. (2014) first proposed the attention mechanism for machine translation tasks, but their model architecture is a mixture of attention layers and RNN (Recurrent Neural Network) rather than pure attention layers. The attention mechanism mimics the cognitive attention of humans. Given a sequence, the intuition is to assign a weight w_{ij} to each pair of elements (e_i, e_j) representing the importance of e_j to e_i, where e_i refers to the i-th element in the sequence. This mechanism allows thus each element to focus on important segments rather than the whole sequence. This mechanism consists of computing the output of an attention function (Vaswani et al., 2017):

> An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

For simplicity, assume that queries, keys and values have the same dimension d. In the scenario of NLP, the input to Transformer is a sequence of word pieces (tokenized text by the WordPiece tokenizer): $(wp_1, wp_2, ..., wp_N)$. Then we have the corresponding word piece embedding matrix $\boldsymbol{E} \in \mathbb{R}^{N \times d}$. Queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$) and values ($\boldsymbol{V}$) are computed by:

$$\boldsymbol{Q} = \boldsymbol{E}\boldsymbol{W}_{\boldsymbol{Q}}, \boldsymbol{K} = \boldsymbol{E}\boldsymbol{W}_{\boldsymbol{K}}, \boldsymbol{V} = \boldsymbol{E}\boldsymbol{W}_{\boldsymbol{V}} \tag{2.17}$$

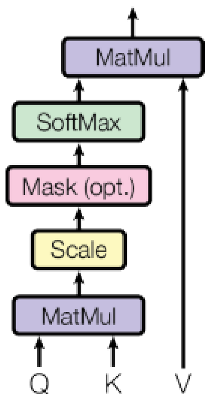


Figure 2.8: Diagram of attention (source: (Vaswani et al., 2017)). $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$ are computed using Equation 2.17.

where $\boldsymbol{W_Q} \in \mathbb{R}^{d\times d}$, $\boldsymbol{W_K} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{W_V} \in \mathbb{R}^{d\times d}$ are trainable weights. The output of the attention function is then computed by:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})V \tag{2.18}$$

Suppose that $\boldsymbol{A} = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})$, then $\boldsymbol{A}$ is a $N \times N$ matrix and a_{ij} is the weight that the i-th word piece wp_i assigns to the j-th word piece wp_j, i.e. the "attention" that wp_i pays to wp_j. It is noteworthy that $\boldsymbol{A}$ is not symmetric since $\boldsymbol{W_Q}$ and $\boldsymbol{W_K}$ are different. The diagram of attention is shown in Figure 2.8. To give an insight into attention coefficients, we compute the matrix containing attention coefficients for the sentence in Figure 1.1 and plot the result in Figure 2.9. Coefficients of each the i-th row represent the attention weights between the i-th token and all other tokens.

Equation 2.18 can be further decomposed to explain the intuition of the attention mechanism. To compute the updated embedding of the i-th word piece v_i' given A and V, we have:

$$(v_i')_j = \sum_{k=1}^{N} a_{ik} V_{kj} \tag{2.19}$$

Omit j in the Equation 2.19, v_i' can be written as a weighted sum of value vectors in V:

$$v_i' = \sum_{k=1}^{N} a_{ik} v_k \tag{2.20}$$

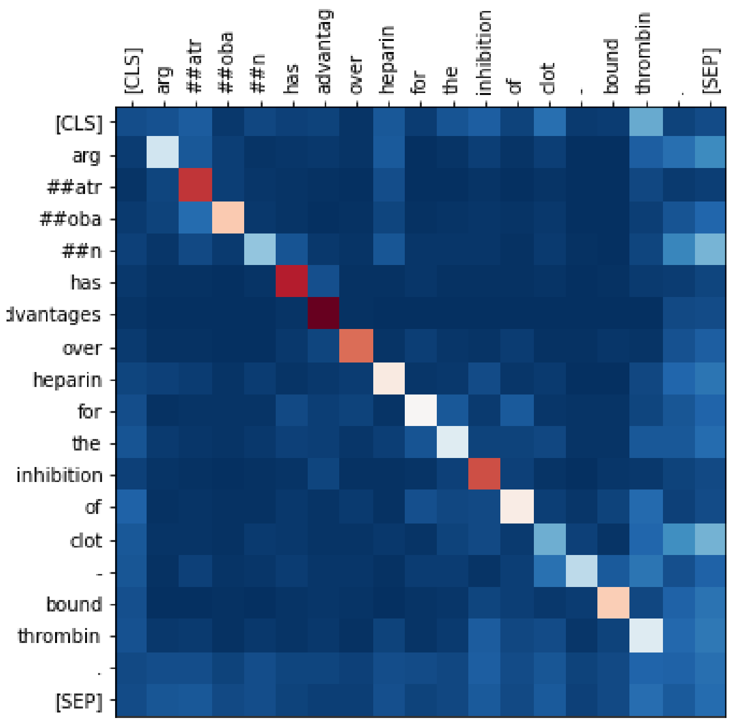


Figure 2.9: Visualisation of the attention coefficient matrix ($L = 12$, $H = 4$) from PubMedBERT (Gu et al., 2021) given the sentence from Figure 1.1 as input. L refers to the layer index, and H refers to the attention head index. A value of 0.0 is marked in deep blue, and 1.0 is marked in deep red. The sentence is tokenized using the tokenizer from PubMedBERT.

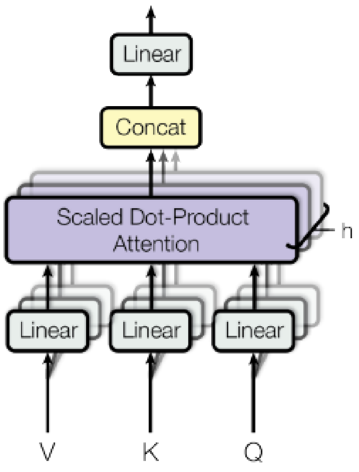


Figure 2.10: Diagram of multi-head attention. (source: (Vaswani et al., 2017)) h refers to the number of attention heads.

where v_k refers to the input embedding of the k-th word piece.

2.4.2 Multi-Head Attention

The attention function consists of computing once linear projections W_Q, W_K, W_V as shown in Equation 2.17. We can further enrich the representations learned by the attention mechanism: by performing the attention function multiple times and learning different linear projections at each time. Outputs of each attention function are then concatenated. *"Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions."* (Vaswani et al., 2017). The diagram of multi-head attention is shown in Figure 2.10.

2.4.3 Model Architecture

In this part, we present the encoder-decoder architecture proposed in the original article of Transformer (Vaswani et al., 2017). The architecture is shown in Figure 2.11. Though current LMs do not inherit exactly the same architecture, most of them use a similar architecture that is composed of stacked attention layers.

As shown in Figure 2.11, each attention layer in the encoder contains two sub-layers: a multi-head self-attention sub-layer and a fully-connected sub-layer. A non-linear activation is used in the fully-connected sub-layer to add non-linearity to the network (same as the function of σ in MLP;

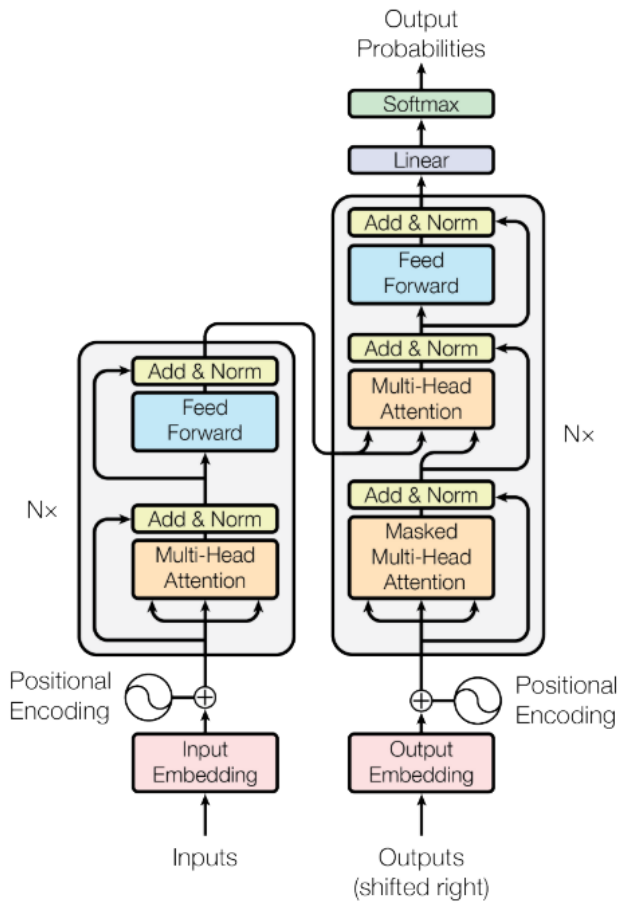


Figure 2.11: The model architecture of Transformer. (source: (Vaswani et al., 2017))

refer to Section 2.2). Both sub-layers connect the input and output by the residual connection (He et al., 2016), a type of skip-connection that is able to ease the optimization of neural networks. The residual connection provides an additional path of data flow by directly connecting the input and output. Suppose that we have an input x and a function $\mathcal{F}$ expressed by a neural sub-layer. The output of residual connection is $\mathcal{F}(x)+x$. Besides, layer normalization (Ba et al., 2016) is performed on the outputs of both sub-layers. The layer normalization consists of re-centering and re-scaling the inputs to neurons of the same layer in order to accelerate the convergence of the network. The operation of residual connection and layer normalization is marked as "Add & Norm" in Figure 2.11. The output of the last attention layer in the encoder is used as input to each attention layer in the decoder.

The attention layer in the decoder is slightly different: each layer contains a masked multi-head attention sub-layer. "Masked" means that the attention coefficient matrix is lower triangular, i.e.

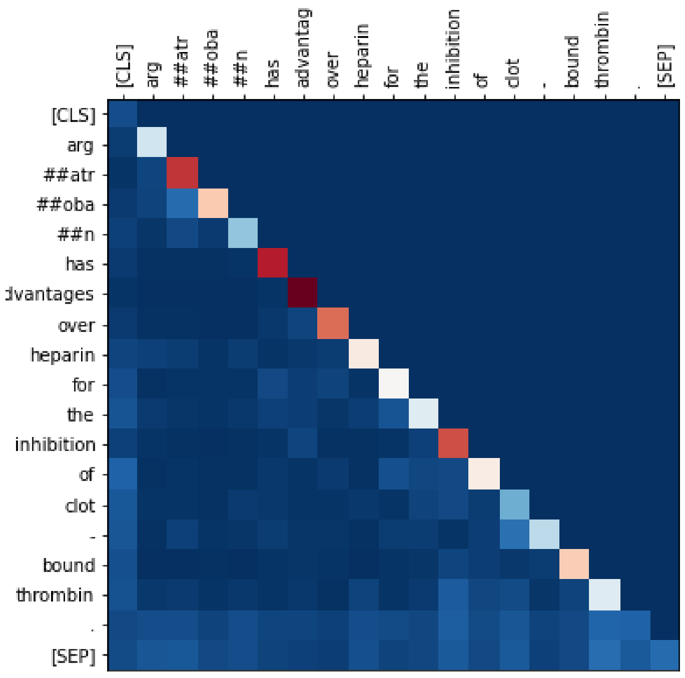


Figure 2.12: The masked version of Figure 2.9.

each token is only allowed to attend to previous tokens. Take the example from Figure 2.9, the corresponding masked matrix should be as shown in Figure 2.12. The masked attention sub-layer is created for the auto-regressive scenario, i.e. when generating texts, previously predicted tokens are used as context to predict the next token. The final output of the decoder is then passed to a linear layer with a softmax activation to compute the probabilities of target classes.

2.5 Transfer Learning

The concept of transfer learning comes from the field of representation learning. Learning good representations is important because the difficulty of information processing tasks may depend on how the information is represented. *"Generally speaking, a good representation is the one that makes a subsequent learning task easier."* (Goodfellow et al., 2016c). For example, dense word vectors computed by context-based prediction tasks (as presented in Subsection 2.1.3) are better representations than one-hot word vectors: one-hot word vectors are less informative as every pair of one-hot word vectors are the same L^2 distance away from each other; while the L^2 distance between each pair of dense word vectors is found to encode their semantic similarity (Mikolov et al., 2013). Experiments also demonstrate that using dense vectors in NLP tasks usually leads to better performance.

The process of training a neural network can be regarded as a kind of representation learning. In the scenario of relation extraction, in most cases the last layer is a linear classifier with a softmax activation, and the rest of the layers actually provides a representation to this classifier. Furthermore, the intermediate representations can be shared by different tasks. Good representations benefit not only RE but also other NLP tasks. This observation leads to the idea of transfer learning: Transfer learned knowledge (shared intermediate representations) to various tasks. The benefits of transfer learning are manifold:

1. It is possible to train a neural network that learns general representations only once, then continue training this network on downstream tasks. This is less time-consuming compared to training a neural network from scratch for each downstream task. The process of learning general knowledge is known as "pretraining", and the process of adapting a general architecture to specific tasks is known as "fine-tuning";

2. Manually-annotated data is not necessary for pre-training. For example, the CBOW model in word2vec (Mikolov et al., 2013) predicts the current word based on its context. Though the prediction task is supervised, labels can be generated automatically from unlabelled texts (which is why the term "unsupervised pre-training" is sometimes used).

The first benefit of transfer learning can be explained by a practical observation: *"The choice of initial parameters for a deep neural network can have a significant regularizing effect on the model (and, to a lesser extent, that it can improve optimization.)"* (Goodfellow et al., 2016a) As mentioned above, this "pretraining plus fine-tuning" strategy uses weights learned from the unsupervised learning stage as the weight initialization for the fine-tuning stage, which is generally a better initialization than a random one. Another explanation is that features learned from unsupervised pretraining may also be useful for the supervised task. It is quite evident for some NLP tasks: understanding semantic similarities between words is helpful to determine how similar two texts are

(Semantic Textual Similarity).

To summarize, transfer learning consists of learning from a large amount of unlabeled data and a relatively small amount of labeled data. In most cases, a general network is pre-trained on unlabeled data, then fine-tuned on labeled data of downstream tasks. Some extreme cases of transfer learning exist such as few-shot (zero-shot) learning: a few (or zero) labeled examples are given in the input to a pre-trained LM, and no weight is updated.

2.6 BERT: Transformer-based Pre-trained LLM

In previous sections, we have presented the self-attention mechanism and discussed the advantages of transfer learning. BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) inherits the encoder architecture of Transformer and is pre-trained based on the principle of transfer learning. The originality of BERT comes from several aspects:

1. Segment embeddings are introduced in BERT along with token embeddings and position embeddings,;

2. BERT proposes two bidirectional pre-training tasks that are found to be effective for downstream NLP tasks;

The success of BERT demonstrates the importance of transfer learning. since BERT, the pre-training and fine-tuning framework has become mainstream in the field of NLP (Devlin et al., 2019):

> BERT advances the state of the art for eleven NLP tasks [...] We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures. BERT is the first fine-tuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.

In this section, we first present the pre-training tasks of BERT, then focus on BERT variants that are proposed to adapt BERT to domain-specific scenarios. Because the architecture of BERT is "almost" identical to the Transformer encoder, we will skip the introduction of the model architecture. The main differences between BERT and the Transformer encoder in terms of model architecture are: BERT uses Gelu (Hendrycks and Gimpel, 2016) as the activation in fully connected sub-layers rather than the previously popular Relu; and BERT contains more attention layers.

2.6.1 Pre-training of BERT

Generative Transformer-based models such as GPT-2 (Radford et al., 2019) use Language modeling as the pre-training task. It consists of modeling the probability of a sentence $s = (t_1, t_2, ..., t_n)$. This joint probability can be factorized as the product of conditional probabilities (Jelinek, 1980; Bengio et al., 2000):

$$p(s) = \prod_{i=1}^{N} p(t_n | t_1, ..., t_{n-1}) \tag{2.21}$$

A main advantage of using language modeling is that it only allows each token to "see" previous tokens, which is mandatory for an auto-regressive model because there is no context from the right side in the decoding stage. However, this characteristic is not necessary for an encoder like BERT. A downside of using language modeling is that when predicting the current token, only contexts from the left side are used. Therefore, BERT chooses to use bidirectional contexts. Besides, in order to prevent each token from "seeing" itself, the authors propose to randomly replace a token with a special symbol "[MASK]" and then ask the model to predict the masked token. This pre-training task is named Masked Language Model (MLM). Unlike Equation 2.21, the objective of MLM changes to maximize:

$$p(t_i) = p(t_i | t_1, ..., t_{i-1}, t_{i+1}, ..., t_n) \tag{2.22}$$

where t_i is selected to be masked. However, simply replacing all target tokens with [MASK] is problematic since no tokens are masked in fine-tuning data. To mitigate this, the authors propose to optimize this strategy further (Devlin et al., 2019):

The training data generator chooses 15% of the token positions at random for prediction.

If the i-th token is chosen, we replace the i-th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time.

Since BERT uses WordPiece tokenization, they mask 15% of word piece tokens in each sequence randomly, *"no special consideration given to partial word pieces."* (Devlin et al., 2019)

Since certain NLP tasks such as QA (Question Answering) require BERT to have the ability to understand relations between sentences, a second binarized pre-training task is proposed: Next Sentence Prediction (NSP). Given a pair of sentences, the NSP task consists of predicting if one sentence follows another in the original document. Pre-training data is generated for NSP tasks such that (Devlin et al., 2019):

when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext).

Though the NSP task is found to be beneficial to some tasks like QA (Devlin et al., 2019), some studies (Liu et al., 2019; Joshi et al., 2020) question the necessity of it. Despite this controversy, all BERT variants used in our experiments are pre-trained with both MLM and NSP tasks. An example of the pre-training and fine-tuning procedure is shown in Figure 2.13.

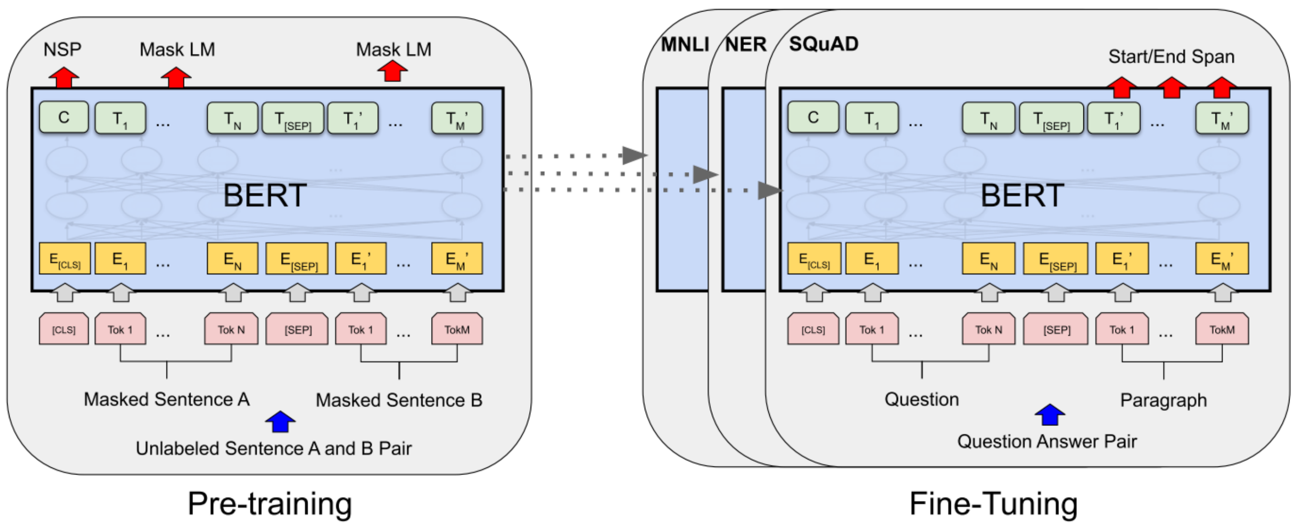


Figure 2.13: The pre-training and fine-tuning framework of BERT. (source: (Devlin et al., 2019))

The pre-training data of BERT is extracted from two sources: BookCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). BookCorpus contains 11,038 free books from the web, each book has more than 20k words. English Wikipedia is an online collaborative encyclopedia, and the total volume of the compressed texts of its articles amounted to 20 GB (November 2022).

Except for [MASK], there exist other special symbols used in BERT:

[CLS]: inserted at the beginning of each input sentence; the vector representation of [CLS] is used as the sentence representation for the NSP task;

[SEP]: inserted at the end of each input sentence or between two sentences (used to separate two sentences in a sentence pair for the NSP task);

[UNK]: used to denote an unrecognized word piece;

[PAD]: added at the end of each sentence so that sentences in a minibatch have the same length;

2.6.2 BERT Embeddings

As mentioned in Subsection 2.1.3, a token embedding itself encodes no positional information. However, the lack of positional information can be supplemented by position embeddings. There are three types of embedding in BERT: token embedding, position embedding, and segment embedding. Each type of embedding is represented as a matrix in the architecture of BERT and used to initialize vector representations of the input. The three types of embedding share the same vector dimension d:

- Token embeddings refer to dense vectors corresponding to word pieces in the vocabulary of BERT. Denote the vocabulary by V and token embedding matrix by $\boldsymbol{E}_{token}$, then $\boldsymbol{E}_{token} \in \mathbb{R}^{|V| \times d}$;

- Position embeddings encode the positions of each token in the sequence. Unlike the sinusoid position embeddings used in the original Transformer (Vaswani et al., 2017), BERT uses absolute position embeddings, i.e. positions are represented by integer indexes of tokens. This encoding method fixes the maximum length of input sequences. Therefore, BERT cannot accept sequences longer than a fixed value L (e.g. 512 for BERT$_{base}$). Denote the position embedding matrix by $\boldsymbol{E}_{position}$, then $\boldsymbol{E}_{position} \in \mathbb{R}^{L \times d}$;

- Segment embeddings are created specifically for the NSP task. The segment embedding matrix can be denoted by $\boldsymbol{E}_{segment} \in \mathbb{R}^{2 \times d}$, there are only two vectors in $\boldsymbol{E}_{segment}$, each encoding a segment value 0 or 1. A token that belongs to the first sentence has a value of 0 in the segment sequence, otherwise has a value of 1. For the NSP task, given a pair of sentences, the two sentences are separated by a special symbol [SEP].

An input tokenized sentence is turned into three sequences before being passed to the BERT encoder: a word piece index sequence, a position index sequence, and a segment value sequence. Each of the three sequences is then turned into embeddings by a lookup operation respectively in $\boldsymbol{E}_{token}$, $\boldsymbol{E}_{position}$ and $\boldsymbol{E}_{segment}$. The sum of the three embeddings is then computed and used as the input to the BERT encoder. This procedure is shown in Figure 2.14.

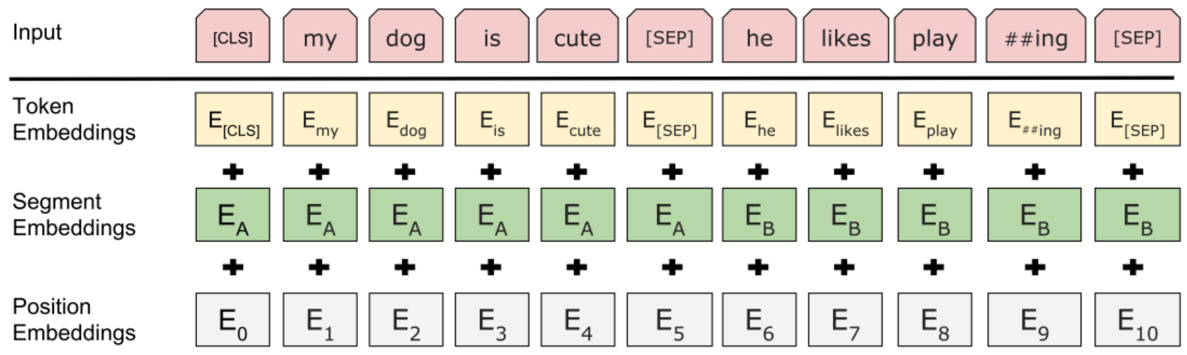

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.14: Embeddings of an input tokenized sentence are the sum of token, position and segment embeddings. (source: (Devlin et al., 2019))

2.6.3 Domain-specific BERT

Though BERT has outperformed previous state-of-the-art models over many downstream NLP tasks, directly fine-tuning it on biomedical texts has limitations (Lee et al., 2020):

First, BERT is trained and tested mainly on datasets containing general domain texts (e.g. Wikipedia), it is difficult to estimate their performance on datasets containing biomedical texts. Also, the word distributions of general and biomedical corpora are quite different, which can often be a problem for biomedical text mining models.

To handle this problem, a solution is to pre-train BERT on biomedical texts. Based on this hypothesis, biomedical BERT variants have been proposed such as BioBERT (Lee et al., 2020), SciBERT (Beltagy et al., 2019), and PubMedBERT (Gu et al., 2021).

Most BERT variants share the same architecture as vanilla BERT. They differ from each other mainly in the following aspects: (1) pre-training corpus; (2) pre-training initialization; (3) WordPiece vocabulary. We list the comparison results between BioBERT, SciBERT and PubMedBERT in Table 2.3.

BERT variant	pre-training corpus	initialized from BERT?	new WordPiece vocabulary?
BioBERT	English Wikipedia Book Corpus PubMed Abstracts PMC Full-text articles	✓	✗
SciBERT	Semantic Scholar	✗	✓
PubMedBERT	PubMed Abstracts	✗	✓

Table 2.3: Comparison between three different BERT variants: BioBERT, SciBERT and PubMedBERT. Since each variant has multiple versions, we choose to compare the following versions: biobert-base-cased-v1.2 (BioBERT); scibert_scivocab_uncased (SciBERT); BiomedNLP-PubMedBERT-base-uncased-abstract (PubMedBERT)[1]. Three aspects are compared: pre-training corpus; if the pre-training starts from BERT checkpoints; and if a new WordPiece vocabulary is created.

Due to the large volume of biomedical data (14 million abstracts, 3.2 billion words used by PubMedBERT) and the existence of long biomedical terms, Gu et al. (2021) emphasizes the importance of domain-specific pre-training from scratch and in-domain WordPiece vocabulary. Though BioBERT is similar to PubMedBERT, PubMedBERT consistently outperforms BioBERT in most downstream tasks (Gu et al., 2021). To give an insight into how in-domain vocabulary helps, we compare the WordPiece tokenization result of BioBERT and PubMedBERT in Table 2.4. Due to

[1]Version names of corresponding pre-trained models of Huggingface (Wolf et al., 2020).

the superior performance of PubMedBERT over BioBERT when applied to biomedical texts, we choose PubMedBERT as the base model in all our experiments.

biomedical term	BioBERT	PubMedBERT
diabetes	✓	✓
DNA	d ##na	✓
hypertension	h ##yper ##tens ##ion	✓
lidocaine	lid ##oc ##aine	✓
chloramphenicol	ch ##lora ##mp ##hen ##ico ##l	✓
acetyltransferase	ace ##ty ##lt ##ran ##s ##fer ##ase	✓

Table 2.4: WordPiece tokenization results of BioBERT and PubMedBERT for some biomedical terms. A tick indicates that the exact word exists as a word piece token in the corresponding vocabulary.

2.7 General Neural Architecture for RE

Now that we have basic knowledge about neural networks, we present a general neural architecture commonly used for relation extraction, as shown in Figure 2.15. Suppose that we have (sentence,label) pairs as input. It takes several steps to compute probabilities of relation types:

- Tokenization: an input sentence is represented as a sequence of word pieces;
- Encoding: the word piece sequence is transformed into a sequence of fixed-length embeddings (represented as a matrix);
- Pooling: obtain the sentence embedding from the word piece embedding matrix. For BERT, we can achieve that by simply taking the embedding of [CLS] (Gu et al., 2021; Lee et al., 2020);
- Linear classifier with softmax (as in Equation 2.2) activation: project the sentence embedding to a vector of dimension N_r, where N_r represents the number of possible relation types.

During the inference stage, we take the most probable relation type as the prediction; during the training stage, a loss value is computed using the prediction $\hat{\boldsymbol{y}}$ and the one-hot vector of true labels $\boldsymbol{y}$ (as mentioned in Subsection 2.2.1):

$$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{N_r} w_i log(\hat{y}_i) y_i \tag{2.23}$$

where w_i is the weight for the i-th relation type. A weighted loss function can better handle imbalanced datasets since we can modify the importance of different relation types by adjusting w_i. w_i can be calculated by:

$$w_i = \frac{\sum_{j=1}^{N_r} N_j}{N_i} \tag{2.24}$$

Relations with fewer examples thus have greater weights. Once $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is defined, gradients of L w.r.t. model weights can be calculated and back-propagated. A gradient-based optimizer can then be used to train the neural model as described in Section 2.2.

2.8 Prerequisites of Biomedical Relation Extraction

We have discussed the challenges of biomedical relation extraction in the first chapter. Before diving into details about adapting BERT to biomedical RE tasks, it is very important to find relevant datasets since we always need data to experimentally verify if we make advances by proposing new models. In this section, we first present benchmark datasets that are used in experiments throughout this thesis, then basic pre-processing steps to extract structured data from raw texts. After that, we introduce the evaluation metrics that we use.

2.8.1 Datasets

We choose three biomedical relation extraction datasets to evaluate the performance of different models: ChemProt (Krallinger et al., 2017), DrugProt (Miranda et al., 2021) and Bacteria Biotope Relation extraction (BB-Rel) (Bossy et al., 2019). This choice is based on multiple factors, e.g. popularity, diversity of entity types, size of the dataset. All datasets are annotated by domain experts, the quality of the datasets is thus guaranteed. Besides, these datasets were all created in the context of relevant NLP challenges, so studies and experimental results prior to our work exist, which makes it easier for us to know where the performance of our proposed models stands.

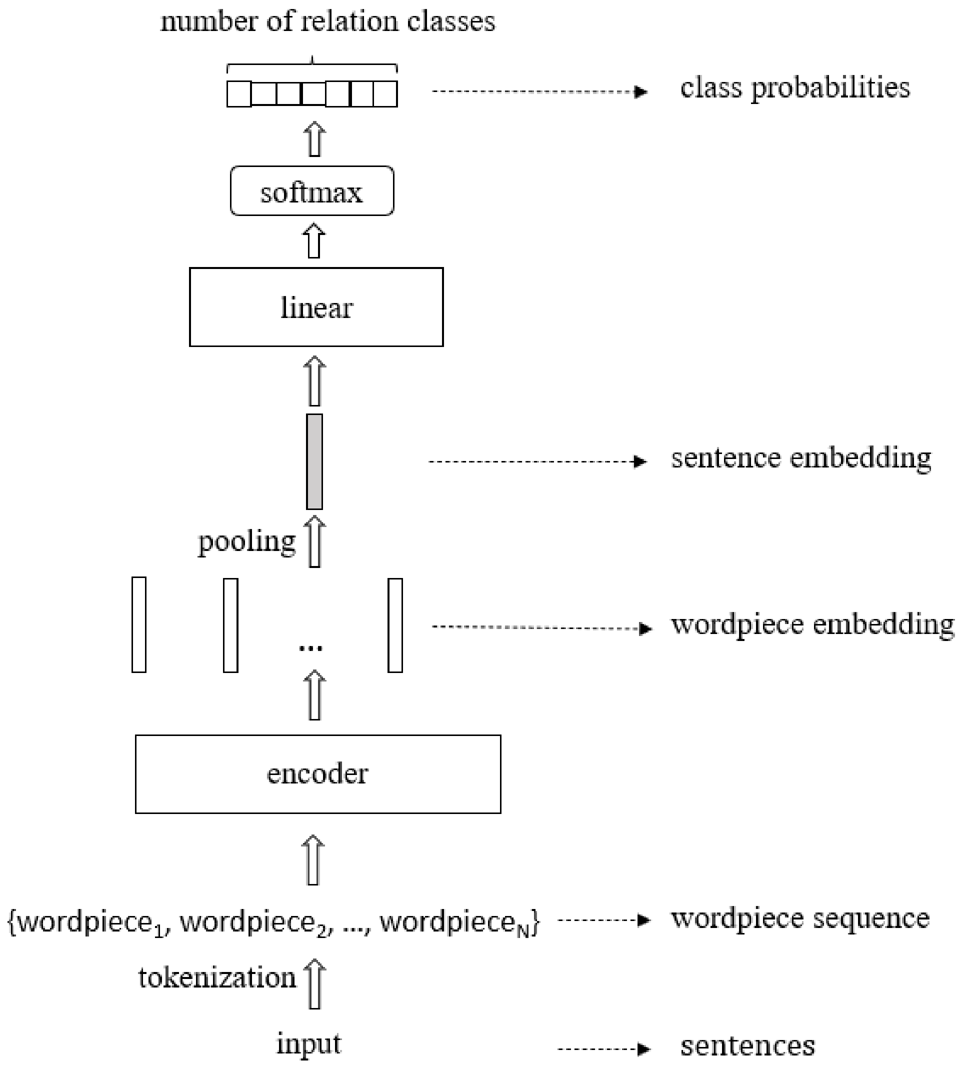


Figure 2.15: A general neural architecture for supervised relation extraction.

Post-challenge studies on these datasets also help us keep abreast of state-of-the-art models that actually work in practical use.

2.8.1.1 ChemProt & DrugProt

We present ChemProt (Krallinger et al., 2017) and DrugProt (Miranda et al., 2021) together as they both contain chemical/drug-protein/gene interactions, and they use the same relation hierarchy as shown in Figure 2.16. All relation types in the figure are directed, i.e. only relations from chemical/drug to protein/gene are annotated, not vice versa. For ChemProt, though all relation types (leaves and nodes in the tree of Figure 2.16) are annotated, they are categorized into 10 groups based on biological semantical similarity, and a label in the form of CPR:X (Chemical-Protein Relation) is assigned to each group ($X = 1, ..., 10$). For example, relations *upregulator*, *activator* and *indirect upregulator* belong to the group labeled by CPR:3. Groups of relation types are shown in Table 2.5. Only 5 out of the 10 CPRs are used for evaluation (CPR:3, CPR:4, CPR:5, CPR:6, CPR:9).

Unlike ChemProt, DrugProt searches to *"facilitate the development of more granular relation extraction systems"* (Miranda et al., 2021) and therefore relation types are not regrouped. In DrugProt, 13 relation types are annotated and used for evaluation: *indirect-downregulator*, *indirect-upregulator*, *direct-regulator*, *activator*, *inhibitor*, *agonist*, *antagonist*, *agonist-activator*, *agonist-inhibitor*, *product-of*, *substrate*, *substrate-product-of*, and *part-of*. It is noteworthy that among the 10,000 documents of the DrugProt test set, only 750 documents are manually annotated and used for evaluation in order to prevent participants from cheating. Statistics of ChemProt and DrugProt are summarized in Table 2.6.

2.8.1.2 Bacteria Biotope 2019

The BB-Rel (Bossy et al., 2019) corpus is curated from PubMed abstracts and PubMed full-text articles. Below we cite the representation scheme from (Bossy et al., 2019) containing four entity types:

- *Microorganism*: names denoting microorganism taxa that correspond to microorganism branches

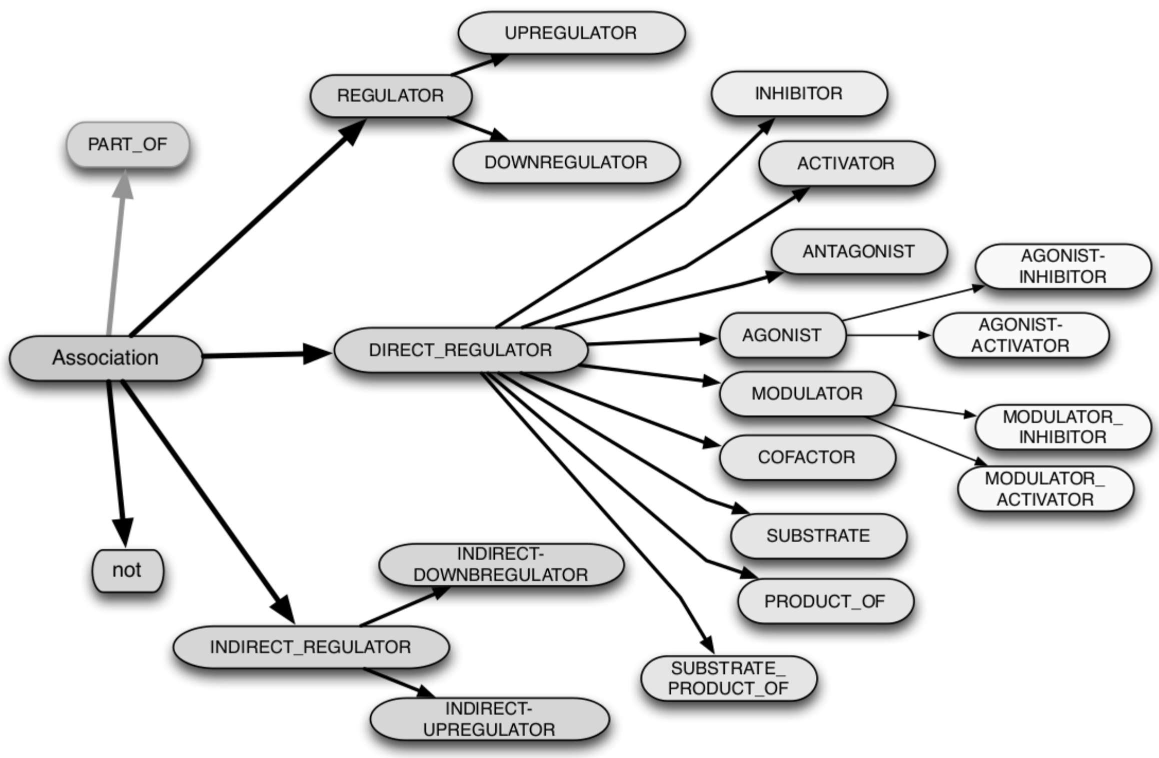


Figure 2.16: ChemProt and DrugProt relation types. (source: (Krallinger et al., 2017))

Group	Eval?	ChemProt relations belonging to the group
CPR:1	✗	part-of
CPR:2	✗	regulator | direct-regulator | indirect-regulator
CPR:3	✓	upregulator | activator | indirect-upregulator
CPR:4	✓	downregulator | inhibitor | indirect-downregulator
CPR:5	✓	agonist | agonist-activator | agonist-inhibitor
CPR:6	✓	antagonist
CPR:7	✗	modulator | modulator-activator | modulator-inhibitor
CPR:8	✗	cofactor
CPR:9	✓	substrate | product-of | substrate-product-of
CPR:10	✗	not

Table 2.5: Groups of ChemProt relation types. A tick in the second column indicates that the corresponding group is used for evaluation, otherwise it is not.

Set	# Docs	# Entity mentions		# Relations
		Chemical/Drug	Gene/Protein	
train	1,020	13,017	12,735	4,157
development	612	8,004	7,563	2,416
test	800	10,810	10,018	3,458

(a) ChemProt statistics.

Set	# Docs	# Entity mentions		# Relations
		Chemical/Drug	Gene/Protein	
train	3,500	46,274	43,255	17,274
development	750	9,853	9,005	3,761
test	10,000	9,434	9,515	-

(b) DrugProt statistics. Relation annotations of the test set are not available.

Set	# Docs	# Entity mentions				# Relations
		Microorganism	Habitat	Geographical	Phenotype	
train	125	730	1,056	34	359	1142
development	64	427	632	46	161	610
test	93	634	920	37	251	-

(c) BB-Rel statistics. Relation annotations of the test set are not available.

Table 2.6: Dataset statistics.

of the NCBI taxonomy;

- *Habitat*: phrases denoting physical places where microorganisms may be observed;
- *Geographical*: names of geographical places;
- *Phenotype*: expressions describing microbial characteristics.

and two directed relation types:

- *lives_in* relations which link a microorganism entity to its location (either a habitat or a geographical entity, or in few rare cases a microorganism entity);
- *exhibits* relations which link a microorganism entity to a phenotype entity.

Descriptive statistics of BB-Rel are summarized in Table 2.6c.

2.8.2 Data Pre-processing

In this section, the objective of data pre-processing is limited to preparing data for BERT. Data processing steps for syntax-enhanced and KB-enhanced models will be presented respectively in Chapter 3 and Chapter 4. Unlike text classification datasets, relation extraction datasets provide not only relation annotations but also named entity annotations. An example of relation and entity annotations is given in Table 2.7. With chemical entities marked in red and gene entities marked in blue, the original document is:

> "Evaluation of animal models for intestinal first-pass metabolism of drug candidates to be metabolized by CYP3A enzymes via in vivo and in vitro oxidation of midazolam and triazolam. Abstract 1. To search an appropriate evaluation methodology for the intestinal first-pass metabolism of new drug candidates, grapefruit juice (GFJ)- and vehicle (tap water)-pretreated mice or rats were orally administered midazolam (MDZ) or triazolam (TRZ), and blood levels of the parent compounds and their metabolites were measured by liquid chromatography/MS/MS. A significant effect of GFJ to elevate the blood levels was observed only for TRZ in mice. 2. In vitro experiments using mouse, rat and human intestinal and hepatic microsomal fractions demonstrated that GFJ suppressed the intestinal microsomal oxidation of MDZ and especially TRZ. Substrate inhibition by MDZ caused reduction in 1'-hydroxylation but not 4-hydroxylation in both intestinal and hepatic microsomal fractions. The kinetic profiles of MDZ oxidation and the substrate inhibition in mouse intestinal and hepatic microsomal fractions were very similar to those in human microsomes but were different from those in rat microsomes. Furthermore, MDZ caused mechanism-based inactivation of cytochrome P450 3A-dependent TRZ 1'-hydroxylation in mouse, rat and human intestinal microsomes with similar potencies. 3. These results are useful information in the analysis of data obtained in mouse and rat for the evaluation of first-pass effects of drug candidates to be metabolized by CYP3A enzymes."

The pre-processing procedure can be divided into four steps:

subject entity id	object entity id	relation
T1	T14	INHIBITOR
T2	T14	SUBSTRATE

(a) Relation annotation

entity id	entity type	span
T1	chemical	[1202,1205)
T2	chemical	[1274,1277)
T3	chemical	[404,413)
T4	chemical	[415,418)
T5	chemical	[423,432)
T6	chemical	[434,437)
T7	chemical	[627,630)
T8	chemical	[807,810)
T9	chemical	[826,829)
T10	chemical	[855,858)
T11	chemical	[997,1000)
T12	chemical	[157,166)
T13	chemical	[171,180)
T14	gene	[1245,1263)
T15	gene	[1536,1541)
T16	gene	[105,110)

(b) Entity annotation

Table 2.7: An annotated document (article id: 23282066) from the original DrugProt development set (`https://biocreative.bioinformatics.udel.edu/tasks/biocreative-vii/track-1/`). Each entity is assigned an id and annotated relations are represented in triples (subject entity id, object entity id, relation type). The text span corresponding to each entity is also given, e.g. a span of [1202,1205) means that entity T1 is a substring of length 3 indexed by 1202 in the original text.

1. Sentence segmentation: we use the off-the-shelf parser Stanza (Qi et al., 2020; Zhang et al., 2021) to segment each document into sentences;

2. Extraction of valid entity pairs: given entity annotations, extract all valid pairs of entities using entity type information. Determining whether a pair of entities is valid depends on the dataset: for example, in DrugProt we consider only chemical/drug-gene/protein relations, therefore only (chemical/drug, gene/protein) pairs are valid. On ChemProt and DrugProt, we only consider intra-sentence relations (since few inter-sentence relations exist), therefore any annotated inter-sentence relation is treated as a false negative during evaluation; On BB-Rel, we extract entity pairs that span over two sentences. In the case of candidate inter-sentence relations, we combine the two sentences in which target entities exist and remove all sentences in-between;

3. Adding entity markers: given each entity pair, add entity markers at the beginning and at the end of each entity using corresponding text spans. Therefore, we have the same number of unique sentences with entity markers as valid entity pairs. In this thesis, we use three types of entity markers for BB-Rel and DrugProt: "@@" for subject entities; "$$" for object entities; "¢¢" in cases when the subject and object entity refer to the same entity or their text spans overlap. The choice of entity markers is empirical: we try to choose tokens that rarely exist in biomedical texts and use different markers for subject and object entities to distinguish them from each other. We keep the original entity markers in ChemProt since it is a widely used dataset, and changing entity markers may make the comparison between different methods less convincing. In ChemProt (Blurb), both subject and object entities are marked as "@ ENTITY $".

4. Labeling: For each entity pair, if a relation exists, attach its relation type as the label to the corresponding sentence. Otherwise, attach a "null" label indicating that no relation exists.

Now we use the example document presented above to illustrate our pre-processing procedure step by step. After sentence segmentation, the document is segmented into 8 sentences. Since in the second step we select only (chemical, gene) entity pairs, we keep only sentences with at least

sentence	label
Evaluation of animal models for intestinal first-pass metabolism of drug candidates to be metabolized by $$ CYP3A $$ enzymes via in vivo and in vitro oxidation of @@ midazolam @@ and triazolam.	NULL
Evaluation of animal models for intestinal first-pass metabolism of drug candidates to be metabolized by $$ CYP3A $$ enzymes via in vivo and in vitro oxidation of midazolam and @@ triazolam @@.	NULL
Furthermore, @@ MDZ @@ caused mechanism-based inactivation of $$ cytochrome P450 3A $$-dependent TRZ 1'-hydroxylation in mouse, rat and human intestinal microsomes with similar potencies.	INHIBITOR
Furthermore, MDZ caused mechanism-based inactivation of $$ cytochrome P450 3A $$-dependent @@ TRZ @@ 1'-hydroxylation in mouse, rat and human intestinal microsomes with similar potencies.	SUBSTRATE

Table 2.8: Pre-processing result of a document (article id: 23282066) from the DrugProt development set.

a chemical entity and a gene entity (i.e. in the above document, only sentences with at least an entity in red and an entity in blue are kept). Therefore, after the first step, two sentences are kept:

1. Evaluation of animal models for intestinal first-pass metabolism of drug candidates to be metabolized by CYP3A enzymes via in vivo and in vitro oxidation of midazolam and triazolam.

2. Furthermore, MDZ caused mechanism-based inactivation of cytochrome P450 3A-dependent TRZ 1'-hydroxylation in mouse, rat and human intestinal microsomes with similar potencies.

In each of the two sentences, two valid entity pairs can be extracted, there are thus four sentences with inserted entity markers. With steps 2-4 combined, the pre-processing result for the example document is presented in Table 2.8.

2.8.3 Evaluation Metric

Though different evaluation metrics exist for relation extraction, in this thesis we report only the micro F1-score as it is the metric used in challenges related to all three datasets that we choose. Since we have added a label *null* during the data pre-processing, this label is taken as the negative class, i.e. correctly predicting *null* does not contribute to the final score. "Micro" here means that we take all classes other than *null* as positive classes. For simplicity of expression, we assume that each relation type is represented by an integer and the relation *null* is always indexed by 0.

Therefore, given predictions $\hat{\boldsymbol{y}} = (\hat{y}_1, ..., \hat{y}_N)$ and true labels $\boldsymbol{y} = (y_1, ..., y_N)$, denote the set of possible relation types by R, then we have $\hat{y}_i \in R$ and $y_i \in R$ with $i \in I$ where $I = \{1, ..., N\}$. Denote the number of true positives, false positives, and false negatives respectively by N_{TP}, N_{FP} and N_{FN}, we have:

$$\begin{aligned} N_{TP} &= |\{i \in I | \hat{y}_i = y_i \,\&\, y_i \neq 0\}| \\ N_{TP} + N_{FN} &= |\{i \in I | y_i \neq 0\}| \\ N_{TP} + N_{FP} &= |\{i \in I | \hat{y}_i \neq 0\}| \end{aligned} \tag{2.25}$$

Then the precision p, recall r, and micro F1-score f_{micro} can be computed as:

$$\begin{aligned} p &= \frac{N_{TP}}{N_{TP} + N_{FP}} \\ r &= \frac{N_{TP}}{N_{TP} + N_{FN}} \\ f_{micro} &= \frac{2 \cdot p \cdot r}{p + r} \end{aligned} \tag{2.26}$$

2.9 Conclusion

In this chapter, we started with an overview of relation extraction methods prior to fine-tuning Large Language Models (LLMs). These methods evolve from relying heavily on manually-designed features or kernels to end-to-end neural models that are partly pre-trained (word embedding). Then we presented neural network basics, including measuring the dissimilarity between predictions and true labels by a loss function, optimization basics about adjusting weights of a neural network iteratively, and the practical use of random seeds and the ensembling technique. After illustrating the prerequisites for neural networks, we presented key elements to understand BERT, one of the most commonly used LLMs: its tokenization algorithm WordPiece, the base architecture Transformer, the transfer learning principle on which the pretraining-fine-tuning framework is built, and BERT itself. A general neural architecture for relation extraction is then presented; it applies to most of the current pre-trained LMs, including BERT. The final part of this chapter covers key elements of biomedical relation extraction prerequisites. We introduced first relevant benchmark datasets, then data pre-processing steps to turn raw texts into ready-to-use structured data, and at the end the most commonly used evaluation metrics for biomedical RE tasks.

Nearly all methods that we will present in Chapter 3 and Chapter 4 are neural methods. In this chapter, we have described general architectures and principles of neural models. In the following two chapters, we will see how we make modifications to existing neural architectures to achieve our objective in this thesis: injecting syntactic or knowledge base (KB) information into BERT to enhance its performance. Generally, existing syntax-enhanced or KB-enhanced models (Subsection 3.2.2; Section 4.3) can be divided into three categories based on the modifications that they make: (1) enriching the input; (2) adding vector representations that encode external information inside the neural architecture; (3) adding learning objectives. This principle applies to our proposed methods as well. We propose three syntax-enhanced models (Section 3.3): CE-PubMedBERT (2nd category), CT-PubMedBERT (1st category); MTS-PubMedBERT (3rd category). The KB-enhanced method that we propose, KB-PubMedBERT, belongs to the 2nd category.

Chapter 3

Injecting Syntactic Information into BERT

Statistical semantics (Delavenay and Delavenay, 1962) consists of applying statistical methods to describe the meaning of words. It makes an assumption that the sense of a word depends on its context words, i.e. words sharing a similar context are supposed to have similar meanings. Both the CBOW model of word2vec (Mikolov et al., 2013) and the MLM pre-training task of BERT use this hypothesis: they consist of using context words to predict the target word. Therefore, both word embeddings and pre-trained language models like BERT are presumed to encode semantic information. However, though under the assumption of statistical semantics, simple syntactic information may be learned such as the POS (part of speech) of words, we are unsure whether more complex syntactic information about the structure of texts can be learned. For natural languages, a sentence is composed of words with different meanings, but these words cannot be randomly arranged. One disadvantage of the bag-of-words (refer to 2.1.1) feature is that it contains only word frequencies, whereas information about how these words are ordered is lost. Yet this lost structural information is also important in understanding the meaning of texts.

Understanding syntactic structure can be beneficial to relation extraction (RE) tasks. Previous studies (Bunescu and Mooney, 2005; Xu et al., 2015; Liu et al., 2015) show that using the shortest dependency path helps extract the most relevant words to the two arguments of a candidate entity pair, therefore improving the performance of RE. Given a long sentence such as “In vitro and in vivo studies have shown that argatroban has advantages over heparin for the inhibition of clot-bound thrombin and for the enhancement of thrombolysis with TPA.", if our goal is to determine the relationship between "argatroban" and "thrombin", then we can obtain from the dependency analysis (which will be presented in Subsection 3.1.1) the shortest dependency path (SDP): "Argatroban", "has", "advantages", "inhibition", "thrombin". We can see that the most important words that suggest the existence of a relation "inhibitor" are included in the SDP. SDP is an extreme case that removes less relevant words entirely, we can naturally extend the idea of paying more attention to words with important syntactic roles to attention-based models. We will discuss how to achieve that later.

In this chapter, we present first two types of commonly used syntactic analyses (Section 3.1), then focus on existing methods that exploit syntactic information to improve the performance of relation extraction (Section 3.2). In Section 3.3 we propose three syntax-enhanced models and test their performances on RE datasets along with an existing syntax-enhanced model. Related experimental results and analysis will also be presented in Section 3.4 and Section 3.5. Section 3.4 and Section 3.5 are highly inspired by Tang et al. (2022).

3.1 Syntactic Analysis

Syntactic analysis consists of investigating the syntactic roles of words or relationships between words based on certain syntactic rules. Since the syntactic structure of a sentence is not explicitly given, we need external tools to obtain it, these external tools are also known as syntactic parsers. In this section, we present two types of syntactic analysis: dependency analysis which consists of discovering syntactic relationships between words, and constituency analysis which consists of segmenting a sentence into nested parts and attributing syntactic roles to each of them.

3.1.1 Dependency Analysis

Given a tokenized sentence, the goal of dependency analysis is to describe the syntactic role of each word by linking it with another word through a directed dependency relation. The two words linked by a dependency relation are also known as dependent and head; each dependency relation starts from the head and points to the dependent. Each dependency relation is a one-to-one correspon-

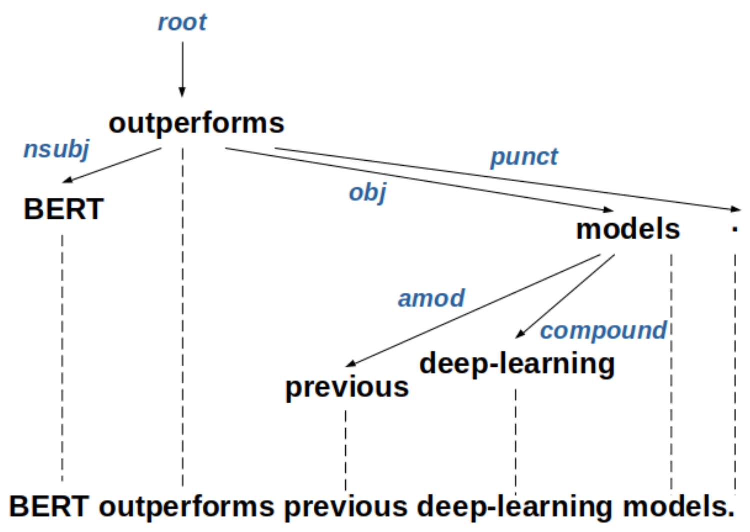


Figure 3.1: The dependency tree of the sentence: "BERT outperforms previous deep-learning models."

dence, and every word has at least one link with another word excluding itself. Since the result of dependency analysis contains word-to-word links, it is also known as the dependency tree (a N-ary tree) or the dependency graph. The number of dependency relations is limited and may vary with different versions. In this thesis, when we mention dependency analysis we always refer to Universal Stanford Dependencies (de Marneffe et al., 2014). To illustrate how dependency analysis works, let us take a simple example using the sentence "BERT outperforms previous deep-learning models.". The corresponding dependency tree is shown in Figure 3.1. The relation "nsubj" between the verb "outperforms" and "BERT" indicates that "BERT" is the nominal subject of the verb; while the relation "obj" indicates that "models" is the nominal object of the verb. "amod" shows that "previous" is the adjectival modifier of "models"; and "compound" suggests that "deep-learning" and "models" make up a two-word compound noun. The detailed explanation of dependency relations shown in Figure 3.1 is summarized in Table 3.1. It is noteworthy that "root" is a special dependency relation as it links a word to a virtual word, which is only used to explicitly indicate that a word is the syntactic root, e.g. in the above example the syntactic root is the verb "outperforms".

A possible way to represent a dependency graph is through its adjacency matrix. The adjacency matrix only saves linkage information while information of the dependency relation type and the direction of dependency relation is discarded, i.e. in each row of the adjacency matrix that corre-

Dependency Relation	Definition
amod	An adjectival modifier of a noun (or pronoun) is any adjectival phrase that serves to modify the noun (or pronoun). The relation applies whether the meaning of the noun is modified in a compositional way (e.g., large house) or an idiomatic way (hot dogs).
compound	The compound relation is used for multiword expressions (MWEs), it is used for: • noun compounds (e.g., phone book); • for particle verbs (e.g., put...up);
nsubj	A nominal subject (nsubj) is a noun which is the syntactic subject and the proto-agent of a clause.
obj	The object of a verb is the second most core argument of a verb after the subject. Typically, it is the noun phrase that denotes the entity acted upon or which undergoes a change of state or motion (the proto-patient).
punct	The punct relation is used for any piece of punctuation in a clause
root	The root grammatical relation points to the root of the sentence. A fake node "root" is used as the governor.

Table 3.1: Definition of a subset of dependency relations (source: (de Marneffe et al., 2014)). There are in total 65 universal dependency relations.

sponds to a certain word, positions of words that are linked to this word are filled with a value of 1, otherwise filled with a value of 0. Besides, self-connections are added to the adjacency matrix for each word. The adjacency matrix of the dependency tree in Figure 3.1 is shown in Figure 3.2. Though the adjacency matrix does not encode the entire information in the dependency graph, it is easy to integrate into neural models, especially attention-based models because both the attention matrix and the adjacency matrix contain word-to-word coefficients. We will see later in this chapter how to exploit the adjacency matrix of the dependency graph for the purpose of injecting syntactic information.

3.1.2 Constituency Analysis

Unlike dependency analysis, constituency analysis consists of finding the underlying hierarchical structure that segments a sentence into constituents, where a constituent refers to a group of words (or a group containing a single word) that functions as a single unit. In a similar way to

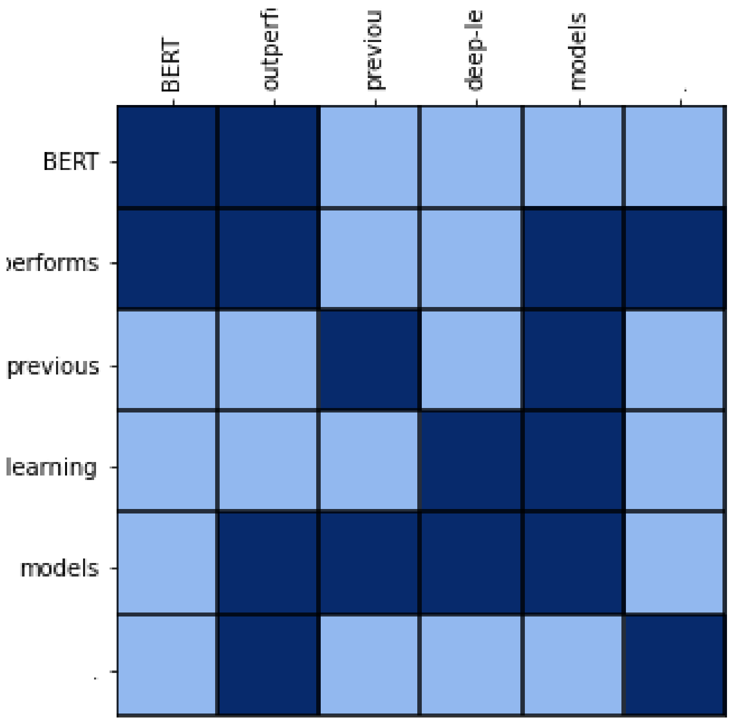


Figure 3.2: The adjacency matrix that corresponds to the dependency tree in Figure 3.1. A cell of (word$_1$,word$_2$) marked in deep blue indicates that word$_1$ and word$_2$ are directly linked to each other in the dependency tree; otherwise, they are not directly linked.

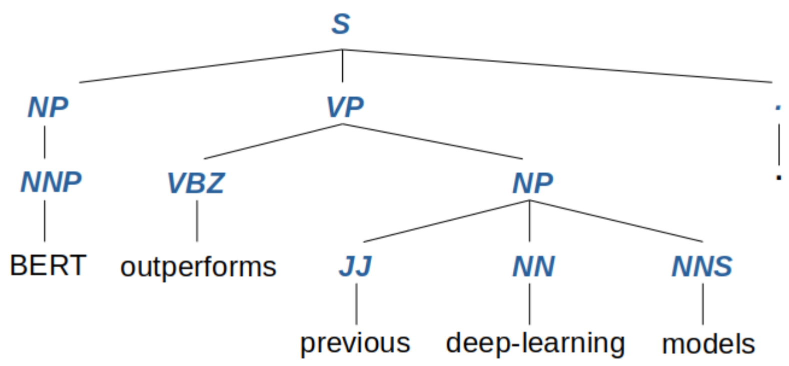


Figure 3.3: The constituency tree of the sentence: “BERT outperforms previous deep-learning models.”

dependency analysis, the result of constituency analysis can also be represented as an N-ary tree. The difference is that in a dependency tree, each node in intermediate levels and each leaf node represents a word; while in a constituency tree, each non-leaf node represents a constituent and each leaf node represents a word. The constituent represented by a non-leaf node n consists of words that correspond to all leaf nodes in the sub-tree rooted at n. A constituency tag is attributed to each non-leaf node describing its syntactic function. Let us take the same sentence as in the previous subsection: “BERT outperforms previous deep-learning models.” The corresponding constituency tree is given in Figure 3.3. If we traverse the constituency tree from the top, the sentence can be summarized as a noun phrase (NP), a verb phrase and a punctuation, where a constituent tagged as NP/VP indicates that the corresponding group of words plays the role of a noun/verb. The verb phrase is then further decomposed into a verb and a noun phrase. Iteratively decomposing each constituent into smaller constituents is the characteristic of constituency analysis. A top-down hierarchical structure is thus built from the biggest constituent (the whole sentence) to the smallest constituents (each word). Constituency tags can be divided into three categories: clause-level, phrase-level, and word-level. Table 3.2 lists the definition of a subset of constituency tags from each category.

Constituency Tag	Definition
	Clause Level
S	simple declarative clause.
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ	Direct question introduced by a wh-word or a wh-phrase.
	Phrase Level
ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
NP	Noun Phrase.
PP	Prepositional Phrase.
VP	Verb Phrase.
	Word Level
DT	Determiner
JJ	Adjective
NN	Noun, singular or mass.
NNS	Noun, plural.
NNP	Proper noun, singular.
RB	Adverb
VB	Verb, base form.
VBD	Verb, past tense.
VBG	Verb, gerund or present participle.
VBZ	Verb, 3rd person singular present.

Table 3.2: Definition of a subset of constituency tags (source: (Bies et al., 1995)). There are in total 82 constituency tags.

3.2 Related Work

We have presented commonly used syntax representation in the previous section. How to integrate these representations into neural models like BERT becomes therefore the next topic that we want to discuss. Studies such as those of (Sachan et al., 2021; Xu et al., 2021; Yu et al., 2020) focus on exploiting syntax representations to enhance the performance of neural models on relation extraction tasks. Before diving into the details of existing syntax-enhanced models, we also want to investigate the following question: Has BERT already learned syntactic information from pre-training? if so, to what extent has it learned syntactic information? In this section, we first present relevant studies of the probing technique that measures to what extent syntactic information is encoded by BERT (Subsection 3.2.1), then focus on existing syntax-enhanced models (Subsection 3.2.2) that are proposed for RE tasks.

3.2.1 Syntactic Probes

A probe refers to a supervised model designed to find information in a representation (Hewitt et al., 2021). If syntactic information is encoded by BERT, it must have been saved in pre-trained weights of BERT. Syntactic probes hypothesize that if we can exploit internal representations of BERT to approximate certain syntax-related properties, it demonstrates the existence of encoded syntactic information in BERT (the hypothesis applies to other neural language models as well). Common targets of probing are word piece vector representation and attention matrices. Clark et al. (2019) propose to probe individual attention heads. A word piece-level attention map is first turned into a word-level map: the attention from a split-up word is computed as the mean of the attention weights from its word pieces; while the attention to a split-up word is the sum of the attention weights to its word pieces. For each word, the word receiving the most attention from it is taken as its syntactic head. The proposed probe is evaluated on the Penn Treebank (Marcus et al., 1993) and compared to fixed-offset baselines, i.e. for each word, the baseline with an offset n predicts always the n-th word next to it as the syntactic head. The per-dependency comparison result shows that for certain dependency relations, the syntactic probe performs substantially better than fixed-offset baselines, showing that certain attention heads specialize to specific dependency relations. Instead of directly using internal representations of BERT for prediction, some studies (Limisiewicz et al., 2020; Hewitt and Manning, 2019; Reif et al., 2019) propose a simple fine-tuning method using internal BERT representations to predict syntax-related properties. The syntactic probe thus consists of two parts: the pre-trained BERT and a fully connected layer. If by simple fine-tuning the syntactic probe outperforms baselines in predicting a syntactic phenomenon of interest, it suggests that syntactic information is embedded in BERT. Hewitt et al. (2021) propose a structural probe that aims to recover the structural properties of dependency trees. *"This can be interpreted as finding the part of the representation space that is used to encode syntax; equivalently, it is finding the distance on the original space that best fits the tree metrics."* (Hewitt et al., 2021). There are two tree metrics to recover: pairwise distance and depth of words in the dependency tree. For a sentence l containing n words $w_{1:n}$, denote the corresponding vector representations by $\boldsymbol{h}_{1:n}$. The objective of the structural probe is to:

- use the L2 distance between a pair of vector representations $(\boldsymbol{h}_i, \boldsymbol{h}_j)$ to estimate the distance between w_i and w_j.
- use the squared norm of vector representations $\|\boldsymbol{h}_i\|$ to estimate the depth of w_i in the dependency tree, i.e. the distance between w_i and the root of the dependency tree.

Their experimental results show that the structural probe based on BERT is surprisingly able to recover the dependency parse structure compared to baseline models such as a right-branching tree or non-contextualized word embeddings. Therefore, they conclude that BERT seems to embed the entire dependency parse tree in the space of vector representations. Similarly, Reif et al. (2019) propose to reconstruct the dependency tree as well, but they change to exploit attention weights. For each pair of words (w_i, w_j), corresponding attention weights are extracted from attention heads across all attention layers, the vector containing these attention weights is then used to train two probes: the first is a binary probe that predicts if a dependency relation exists between w_i and w_j in the dependency tree; the second is a multi-class probe that predicts the type of dependency relation. The architecture of the proposed syntactic probe is shown in Figure 3.4. Experiments on the Penn

Treebank corpus show that the binary probe and the multi-class probe obtain respectively high accuracy of 85.8% and 71.9%, suggesting that the dependency information is encoded in attention matrices.

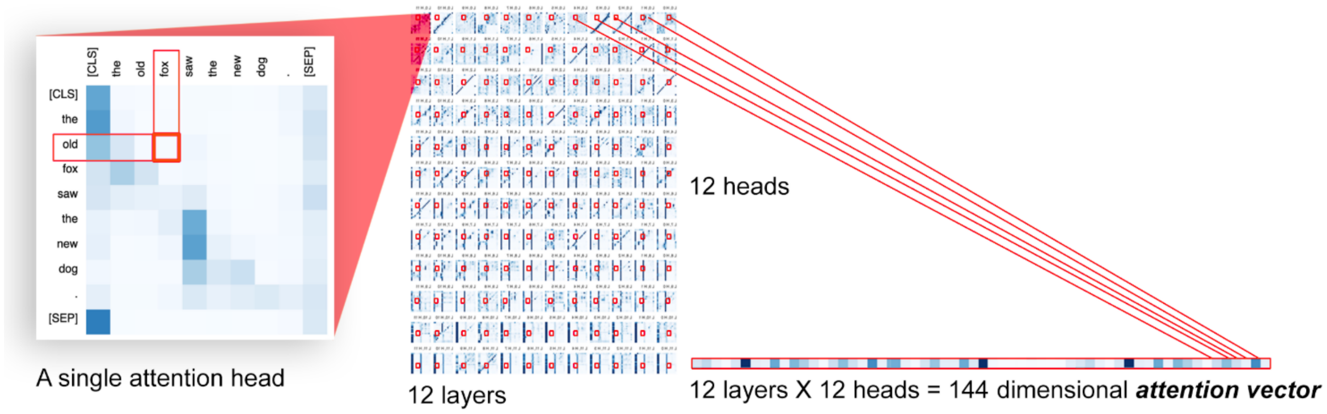


Figure 3.4: The architecture of the syntactic probe proposed by Reif et al. (2019). They use the base version of BERT that consists of 12 layers each containing 12 attention heads. Therefore, for each pair of words, an attention vector of dimension $12 \times 12 = 144$ is extracted.

Relevant studies above based on syntactic probes suggest that pre-trained BERT encodes syntactic information, either by encoding distance information of the dependency tree in the vector representations (depth can also be regarded as a distance) or by encoding word-to-word linkage information in attention matrices. This discovery is important as it inspires us to design syntax-enhanced models in subsequent sections.

3.2.2 Syntax-enhanced Models

Though BERT is found to encode syntactic information, there is no evidence that encoded syntactic information is sufficient and that introducing external syntax is useless or harmful. On the contrary, existing studies (Guo et al., 2021; Xiong et al., 2019; Zhang et al., 2018; Bai et al., 2021; Sachan et al., 2021) show that integrating syntactic information can be helpful. In this section, we focus on studies that successfully introduce syntactic information into neural models to improve the performance on the RE task. We include studies that do not concern pre-trained language models as well, as the methods that are used to inject syntactic information into neural models also apply to pre-trained language models.

3.2.2.1 Adjacency Matrix-based Methods

Many previous syntax-enhanced methods (Sachan et al., 2021; Zhang et al., 2018; Yu et al., 2020; Xiong et al., 2019) consist of using a graph neural network (GNN) specialized to encode dependency information. As presented in Subsection 3.1.1, a dependency tree can be represented by an adjacency matrix in which each entry of value 1 corresponds to a direct link between words in the dependency tree. GNNs of various structures are designed to exploit the adjacency matrix. Zhang et al. (2018) propose a GCN (Graph Convolutional Network) to directly insert the adjacency matrix into convolution layers. For each word w_i, instead of using a sliding window of fixed length and summing word vectors within the window, the upgraded word vector $\boldsymbol{h_i'}$ is calculated as the sum of word vectors that are linked to w_i in the dependency tree:

$$\boldsymbol{h_i'} = \sigma(\sum_{j=1}^{n} \boldsymbol{A_{ij}} \boldsymbol{W} \boldsymbol{h_j} + \boldsymbol{b})$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ denotes the linear transformation and the bias; $\boldsymbol{A}$ denotes the adjacency matrix. The method proposed by Zhang et al. (2018) does not compute self-attention coefficients but directly uses the adjacency matrix as the self-attention matrix. Sachan et al. (2021) use the adjacency matrix as the attention mask in self-attention sublayers. To extend the dependency tree to word pieces, for words that are segmented into multiple word pieces, extra edges are added between the first word piece and the subsequent word pieces. Based on that, they propose syntax-GNN, which is an encoder that consists of multiple modified transformer layers. The self-attention sublayer is replaced by graph attention (Veličković et al., 2018), i.e. each word piece attends only to word pieces that are linked to it in the dependency tree. Original attention weights are kept and the adjacency matrix is used as a mask. Given original attention weights α_{ij} each representing the interaction score between the i-th and the j-th word piece, the graph attention weight g_{ij} is computed as:

$$g_{ij} = \frac{\alpha_{ij}}{\sum_{k \in \mathcal{N}_\rangle} \alpha_{ik}}$$

where $\mathcal{N}_\rangle$ represents the set of word pieces that are linked to the i-th word piece in the dependency tree. Figure 3.5 shows the diagram of a modified transformer layer in Syntax-GNN. Two syntax-GNN-based models are then proposed: Late-Fusion and Joint-Fusion. Both models consist of obtaining syntax-enhanced representations by feeding word piece representations through a syntax-GNN containing 4 modified transformer layers. The difference is that for Late-Fusion, the syntax-GNN is placed after BERT, while in Joint-Fusion the syntax-GNN is added before BERT. For Late-Fusion, the outputs of pre-trained BERT and syntax-GNN are infused using the Highway Gate (Srivastava et al., 2015) method. Instead of using a simple sum or average operation, in Highway Gate a parameter g_i is set to adjust the importance of two inputs:

$$\begin{aligned} \boldsymbol{g}_i &= \sigma(\boldsymbol{v}_i^T \boldsymbol{W}_g + \boldsymbol{b}_g) \\ \boldsymbol{h}_i &= \boldsymbol{g}_i \odot \boldsymbol{v}_i + (\boldsymbol{1} - \boldsymbol{g}_i) \odot \boldsymbol{z}_i \end{aligned} \tag{3.1}$$

where σ denotes the sigmoid function; $\boldsymbol{W}_g$ and $\boldsymbol{b}_g$ are learnable parameters. At each layer of Joint-Fusion, the sum of syntax-GNN representations and the output from the last BERT layer is used as input to compute the key and query vectors. The architecture of Late-Fusion and Joint-Fusion are shown in Figure 3.6. In their experiments, Late-Fusion and Joint-Fusion are tested on Semantic Role Labeling (SRL), NER, and RE tasks. Experimental results show that Late-Fusion provides gains on SRL and RE tasks, and Joint-Fusion improves the performance on SRL tasks. They also find that significant gains obtained on SRL tasks may be due to the fact that gold dependency parses are available, while on RE tasks only parses obtained from in-domain parsers are available: *"drastic impact on the performance of the Syntax-Augmented BERT models, with substantial gains only observed when gold parses are used."* (Sachan et al., 2021)

A drawback of the GCN method proposed by Zhang et al. (2018) is that the integrated adjacency matrix is fixed during the training, it is therefore unable to adjust the importance of words to better classify relations. Yu et al. (2020) propose DP-GCN (Dynamically Pruned Graph Convolutional Network) in which the rethinking mechanism (Li et al., 2018) is introduced to handle this problem. The rethinking mechanism consists of adding loops inside a neural network such that

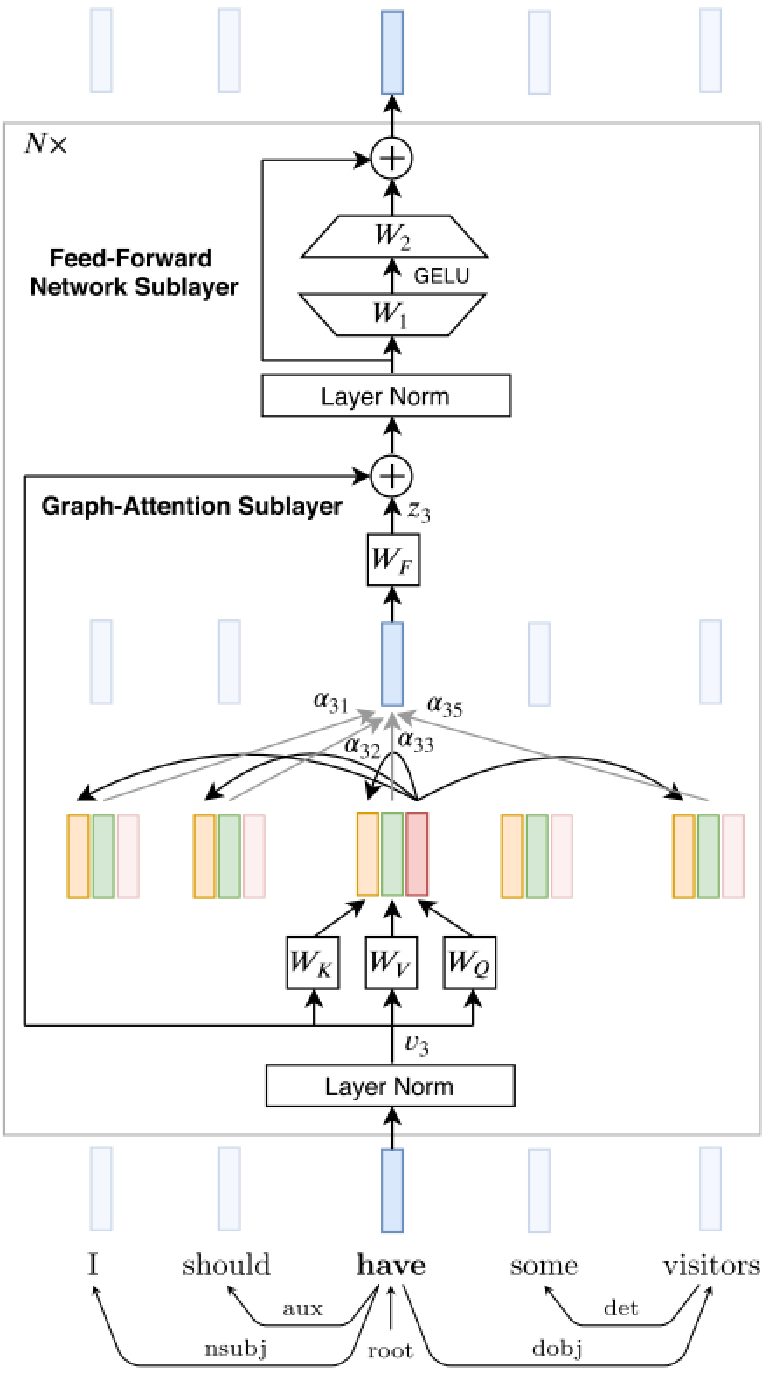
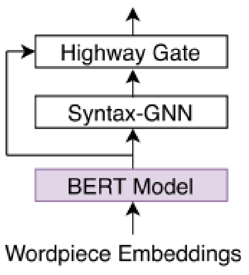
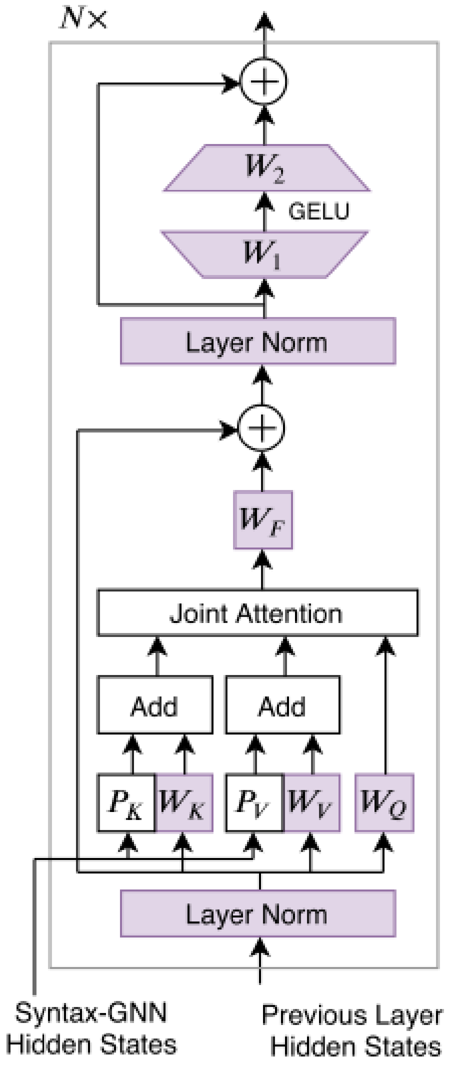


Figure 3.5: The modified transformer layer of syntax-GNN (source:(Sachan et al., 2021)). As "some" is the only word that is not linked to "have", α_{34} in the figure is not used for computing the updated representation of "have".



(a) Late-Fusion



(b) Joint-Fusion

Figure 3.6: Two syntax-enhanced models proposed in (Sachan et al., 2021). Inputs to the two models are word piece embeddings. (source: (Sachan et al., 2021))

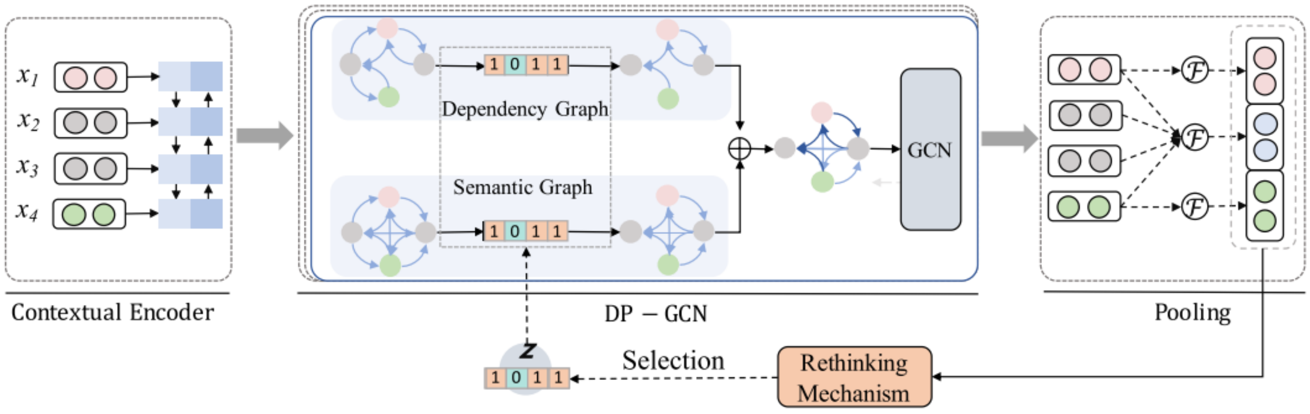


Figure 3.7: The architecture of DP-GCN (source:(Yu et al., 2020)). The dependency graph denotes the pruned adjacency matrix (pruned by applying binary gates over it); the semantic graph denotes the self-attention matrix computed as in the original transformer layer.

chosen intermediate representations are used to compute themselves in the feed-forward pass. Intuitively, vector representations at the output of DP-GCN are used to compute a set of binary gates $\{z_1, ..., z_n\}$, where $z_i \in \{0, 1\}$ and a z_i of value 1 indicates that the information from the i-th node in the dependency graph can be used to update the vector representation of each node. Besides, instead of simply using the adjacency matrix A_{adj} as the self-attention matrix as in (Zhang et al., 2018), DP-GCN also computes the self-attention matrix A_{att} as in the original transformer layer. Binary gates are applied over both A_{adj} and A_{att} and their average is used as the final self-attention matrix in each layer of DP-GCN. The architecture of DP-GCN is shown in Figure 3.7. DP-GCN is proved to outperform previous syntax-enhanced models such as GCN (Zhang et al., 2018) on a general-domain corpus TACRED (Zhang et al., 2017).

3.2.2.2 Syntax-aware Pre-training

Since the original BERT is not pre-trained to explicitly capture syntactic information, another solution to build a syntax-enhanced BERT is to set pre-training tasks that specialize to syntax (Zhang et al., 2022; Wang et al., 2021; Bai et al., 2021; Xu et al., 2021). Xu et al. (2021) propose the distance prediction (DP) task that captures global syntactic distances between words. In a similar way to the structural probe (Hewitt and Manning, 2019), the authors hypothesize that capturing pairwise syntactic distances between words is beneficial for BERT to be aware of the

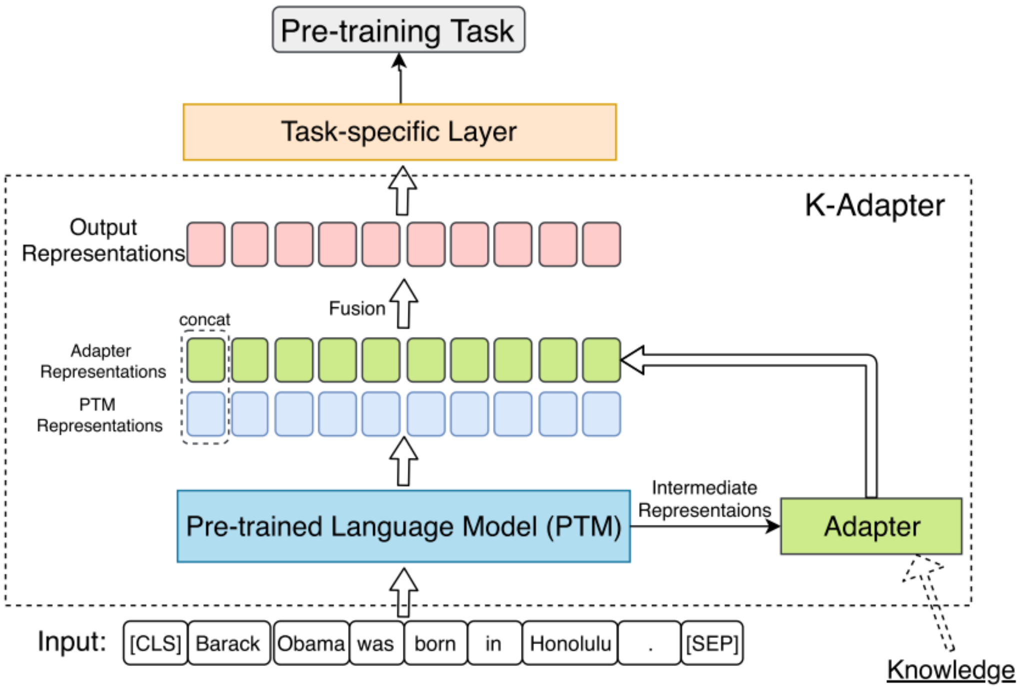


Figure 3.8: The overview of K-adapter (source: (Wang et al., 2021)).

syntactic structure of texts. The DP task consists of predicting the distance matrix $\boldsymbol{D}$ in which $\boldsymbol{D}_{ij} = d(i, j)$, where $d(i, j)$ refers to the length of the Shortest Dependency Path (SDP) between the i-th and the j-th word. Vector representations of words i and j are concatenated and passed to a linear classifier to predict $d(i, j)$. Wang et al. (2021) propose a K-adapter that consists of two parts: pre-trained model, and a neural network (known as an adapter) plugged outside the pre-trained model that specializes in capturing different types of knowledge. Unlike multi-task learning, K-adapter adopts a new training strategy: to prevent the pre-trained model from forgetting original knowledge that is learned, parameters of the original pre-trained model are frozen in the pre-training stage; inversely, the pre-trained model is made trainable and parameters of adapters are frozen in the fine-tuning stage. To capture syntactic knowledge, the corresponding linguistic adapter is pre-trained on the dependency relation prediction task: for each word, the goal of the task is to predict the index of its syntactic head in the dependency tree. The schema of K-adapter pre-training is shown in Figure 3.8.

3.2.3 Discussion

Though previous syntax-enhanced methods are found to improve the RE performance, there still exist several limitations. Firstly, most of syntax-enhanced methods are tested on general-domain corpora such as NYT (Riedel et al., 2010) or TACRED (Zhang et al., 2017), few studies focus on systematically applying syntax-enhanced methods on biomedical RE tasks. Secondly, existing syntax-enhanced methods mostly exploit dependency information while constituency parsers are also available. Lastly, most BERT-based syntax-enhanced models consist of pre-training which requires more computing resources compared to fine-tuning. These limitations lead us to propose syntax-enhanced models that are different from existing ones. We wish to design syntax-enhanced models that require no pre-training, injecting both dependency and constituency information, and working on biomedical RE corpora.

3.3 Contribution: Syntax-enhanced Models

In this section, we present several approaches to inject syntax into BERT models, looking at methods that use syntactic information in the form of dependencies and constituents. We first test Late-Fusion (Subsection 3.2.2.1) that explicitly injects dependency-based syntactic information and is not previously evaluated on biomedical RE tasks, then we propose two novel constituency-based explicit approaches. Finally, we introduce a multi-task method that implicitly injects dependency-based syntactic information.

3.3.1 CE-PubMedBERT

The first method we propose does not use complete constituency trees, but aims first to extract chunks from the constituency tree and then enhance PubMedBERT (which is the model we selected to use as our base model) with chunk information. We name this method CE-PubMedBERT (Chunk-Enhanced-PubMedBERT). Intuitively, in addition to applying the attention mechanism at the word piece level (which is done by PubMedBERT), we aim to compute attention weights at the chunk level. We hypothesize that computing attention at the chunk level is beneficial for relation extraction, as sentences are segmented into chunks that play the same role in constituency grammar, therefore the phrase-level structure of texts is explicitly provided to PubMedBERT. In this method, we first obtain chunks by grouping together word pieces that are consecutive leave nodes in the dependency tree and share the same lowest common ancestor. For example, in the sentence "Caffeine inhibits the checkpoint kinase ATM.", given the corresponding constituency tree shown in Figure 3.9, we only group together the word pieces of "the checkpoint kinase ATM", which share the same parent node Noun Phrase (NP) and NP is their lowest common ancestor. We do not group together the word pieces of "inhibits the checkpoint kinase ATM", because though they are in the subtree rooted at the node of Verb Phrase (VP), VP is not the lowest ancestor of nodes "the", "checkpoint", "kinase", "ATM". This process can be regarded as a kind of shallow chunking, through which we can simplify the sentence representation by regarding chunks as individual tokens.

Figure 3.9 shows the architecture of CE-PubMedBERT. Atop the pre-training PubMedBERT, we add a block wp2const that contains no trainable parameter but sums only word piece embeddings that belong to the same chunk to generate chunk embeddings. Suppose that there are N word pieces in a sentence and M chunks are extracted from the constituency tree: at the input of the wp2const block, we have a sequence of word piece embeddings $[\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N]$. Inside the wp2const block, we compute chunk embeddings by:

$$u_i = \sum_{j \in \mathcal{C}_i} \boldsymbol{v}_j \tag{3.2}$$

where $i = 1, 2, ..., M$. $\mathcal{C}_i$ denotes the set of word piece indexes that belong to the i-th chunk and u_i the embedding of the i-th chunk. Chunk embeddings are then fed into extra attention layers, which share the same architecture as the transformer layers as proposed by Vaswani et al. (2017). The output of extra attention layers is infused with the output of pre-trained PubMedBERT by Highway Gate (Srivastava et al., 2015) as in the Late-Fusion method described in Subsection 3.2.2.1.

3.3.2 CT-PubMedBERT

While CE-PubMedBERT relies mostly on chunks, our second method, named CT-PubMedBERT (Constituency-Tree-enhanced-PubMedBERT), aims to embed complete constituent trees. However,

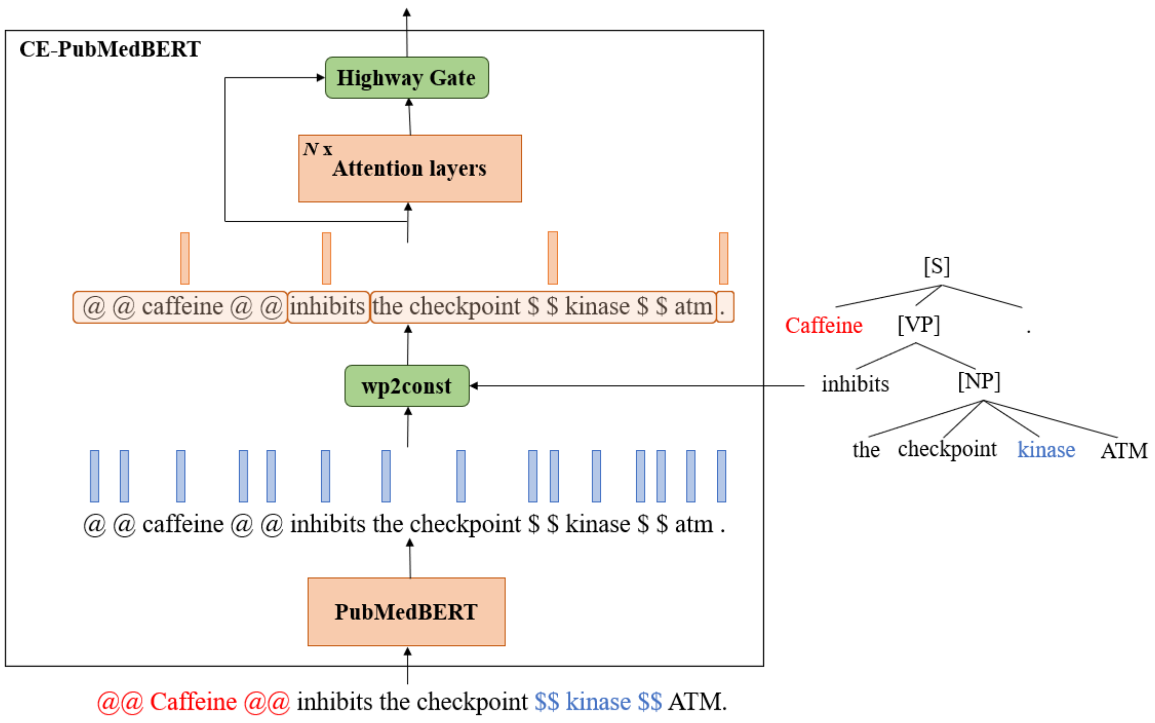

CE-PubMedBERT
Highway Gate
N x Attention layers
@ @ caffeine @ @ inhibits the checkpoint $ $ kinase $ $ atm .
wp2const
@ @ caffeine @ @ inhibits the checkpoint $ $ kinase $ $ atm .
PubMedBERT
@@ Caffeine @@ inhibits the checkpoint $$ kinase $$ ATM.
[S]
Caffeine
[VP]
.
inhibits
[NP]
the
checkpoint
kinase
ATM

Figure 3.9: the architecture of CE-PubMedBERT: the wp2const block at the output of the PubMedBERT model groups together the word pieces that belong to a pre-defined chunk to compute chunk embeddings.

since it is impossible to feed constituent trees directly to PubMedBERT, we choose to first turn constituent trees into sequences. An example of the linearization of the constituency tree is shown in Figure 3.10. By converting a constituency tree to its DFS (Depth First Search) traversal, the resulting sequence is supposed to encode hierarchical information of the tree structure; furthermore, we keep the constituency tags of non-leaf nodes, which provides additional information about the tree. Besides, we keep the wordpiece-to-wordpiece attention pattern, which is the same as in standard attention layers. Constituency tags are treated as independent word pieces and are added to the vocabulary of the word piece tokenizer. This operation certainly increases the difficulty of fine-tuning due to the fact that word piece embeddings of these newly added constituency tags are randomly initialized and the pre-trained PubMedBERT does not see any of them during the pre-training phase. We still keep this model in experiments however, hypothesizing that the additional information brought by constituency tags may compensate for the negative effect caused by introducing new word pieces.

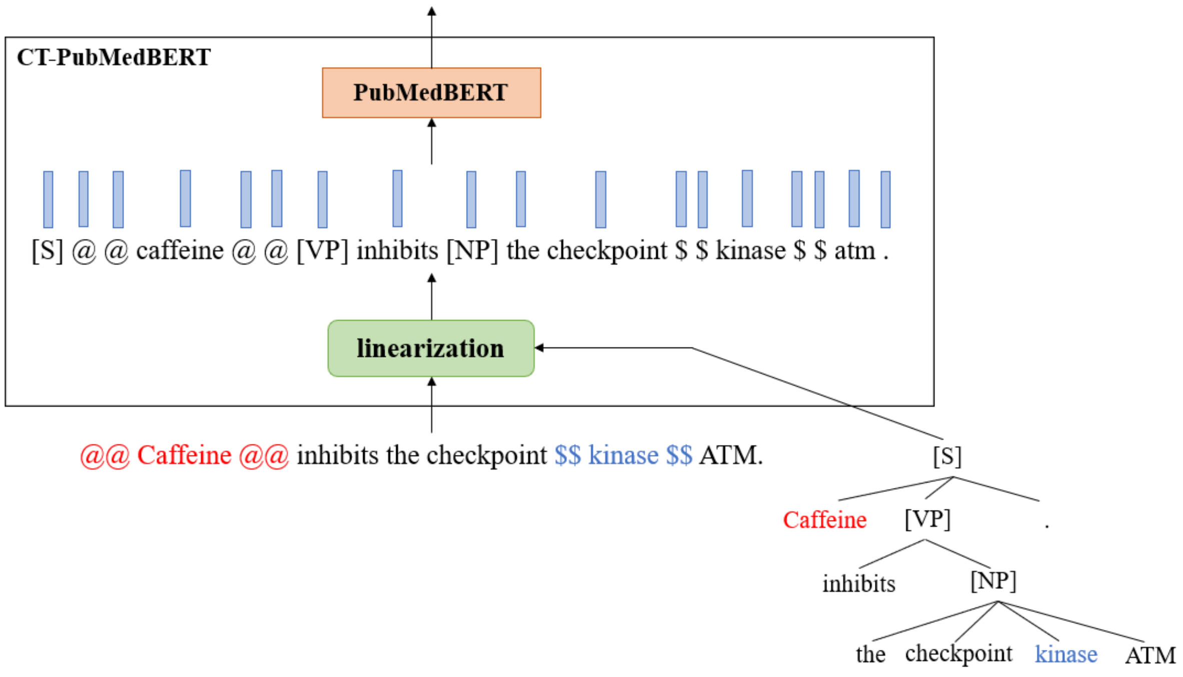

CT-PubMedBERT
PubMedBERT
[S] @ @ caffeine @ @ [VP] inhibits [NP] the checkpoint $ $ kinase $ $ atm .
linearization
@@ Caffeine @@ inhibits the checkpoint $$ kinase $$ ATM.
[S]
Caffeine
[VP]
.
inhibits
[NP]
the
checkpoint
kinase
ATM

Figure 3.10: the architecture of CT-PubMedBERT: the linearization turns a constituency tree to a sequence by DFS traversal.

3.3.3 MTS-PubMedBERT

Aside from explicitly injecting syntax into PubMedBERT by modifying either the input or the internal representations (such as attention weights), another option to integrate syntactic information is to set syntax-aware tasks as presented in Subsection 3.2.2.2. Instead of setting syntax-aware tasks in the pre-training stage, we choose to adopt a multi-task learning architecture that seeks to encode syntactic information and at the same time classify relations, i.e. we add syntax-aware tasks in the fine-tuning stage. Inspired by the structural probe (Hewitt and Manning, 2019) (Subsection 3.2.1), our proposed architecture MTS-PubMedBERT (Multi-Task-Syntax-enhanced-PubMedBERT) is jointly trained on:

- RE tasks: assign relation labels for the given sentence;
- distance probe task: predict pairwise word distances in the dependency tree;
- depth probe task: predict the depth of each word in the dependency tree.

The architecture of MTS-PubMedBERT is illustrated in Figure 3.11. We hypothesize that using the two probing tasks in (Hewitt and Manning, 2019) as supervised objectives will force the model to encode syntactic information and that the word piece representations enriched with syntactic information will help better categorize relations. Unlike Hewitt et al. (2021), in our architecture the two probing tasks are treated as classification tasks, i.e. each distance corresponds to a class $c \in \mathcal{C}_{dist}$. Similarly, each depth corresponds to a class $c \in \mathcal{C}_{depth}$. Empirically, syntactic distances are divided into 12 classes and syntactic depths are divided into 8 classes as shown in Table 3.3.

Formally, for a sentence s represented as a word piece sequence of $wp^s_{1:n}$, we first sum PubMed-BERT vector representations of word pieces that belong to the same word to obtain word vector representations $\boldsymbol{h}^s_{1:n} \in \mathbb{R}^m$. Denote the true syntactic distance between (w^s_i, w^s_j) by $\hat{d}^s_{ij}$ and its one-hot encoding by $\hat{\boldsymbol{d}}^s_{ij}$; the predicted syntactic distance by d^s_{ij} and its one-hot encoding by $\boldsymbol{d}^s_{ij}$. We first apply a linear transformation to project $\boldsymbol{h}^s_{1:n}$ into a subspace specialized to capture syntactic distance information, then use the element-wise square of the difference vector between transformed $(\boldsymbol{h}^s_i, \boldsymbol{h}^s_j)$ to predict $\hat{d}^s_{ij}$:

$$\begin{aligned} \boldsymbol{h}'^s_i &= (\boldsymbol{h}^s_i)^T \boldsymbol{W}_{dist} + \boldsymbol{b}_{dist} \\ \boldsymbol{d}^s_{ij} &= ((\boldsymbol{h}'^s_i - \boldsymbol{h}'^s_j) \odot (\boldsymbol{h}'^s_i - \boldsymbol{h}'^s_j))^T \boldsymbol{W_1} + \boldsymbol{b}_1 \end{aligned} \tag{3.3}$$

where $\boldsymbol{W}_{dist} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{b}_{dist} \in \mathbb{R}^m$ refer to the weight and bias of the first linear transformation; $\boldsymbol{W}_1 \in \mathbb{R}^{m \times |\mathcal{C}_{dist}|}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{|\mathcal{C}_{dist}|}$ the weight and bias of the second linear transformation; $\odot$ denotes the element-wise multiplication (Hadamard product). Suppose that $\hat{\boldsymbol{D}}^s$ is the true syntactic distance matrix such that $\hat{\boldsymbol{D}}^s_{i,j} = \hat{d}^s_{ij}$. $\hat{\boldsymbol{D}}^s$ is a symmetric matrix and values on the diagonal line are 0. Since predicting 0 or values at symmetric positions is trivial, we take the upper triangular matrix of $\hat{\boldsymbol{D}}^s$ as prediction targets. The loss function of the pairwise distance probe can thus be written as:

$$\mathcal{L}_{dist} = \sum_s \frac{2}{|s|(|s|-1)} \sum_{i=1}^{|s|} \sum_{j=i+1}^{|s|} CE(\hat{\boldsymbol{d}}^s_{ij}, \boldsymbol{d}^s_{ij}) \tag{3.4}$$

where $|s|$ denotes the length of s, and CE denotes the cross-entropy loss as described in Equation 2.4

distance (interval)	class
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
$[10, 15)$	9
$[15, 20)$	10
≥ 20	11

(a) Syntactic distances are divided into 12 classes.

depth (interval)	class
0	0
1	1
2	2
3	3
4	4
5	5
$[6, 10)$	6
≥ 10	7

(b) Syntactic depths are divided into 8 classes.

Table 3.3: Syntactic pairwise distances and depths are regrouped into classes. 0 is not used as a label for syntactic distances but is kept for syntactic depths (syntactic root).

(Subsection 2.2.1).

For the depth probe task, denote the true syntactic depth of w_i^s by $\hat{p}_i^s$ and its one-hot encoding by $\hat{\boldsymbol{p}}_i^s$; the predicted syntactic distance by p_i^s and its one-hot encoding by $\boldsymbol{p}_i^s$. Similarly, we first apply a linear transformation to project $\boldsymbol{h}_{1:n}$ to a subspace that is specialized in capturing syntactic depth information, then use transformed $\boldsymbol{h}_i$ to predict $\hat{p}_i^s$:

$$\begin{aligned} \boldsymbol{h}_i''^s &= (\boldsymbol{h}_i^s)^T \boldsymbol{W}_{depth} + \boldsymbol{b}_{depth} \\ \boldsymbol{p}_i^s &= \boldsymbol{h}_i''^s \boldsymbol{W}_2 + \boldsymbol{b}_2 \end{aligned} \tag{3.5}$$

where $\boldsymbol{W}_{depth} \in \mathbb{R}^{m \times m}$, $\boldsymbol{b}_{depth} \in \mathbb{R}^m$, $\boldsymbol{W}_2 \in \mathbb{R}^{m \times |\mathcal{C}_{depth}|}$, $\boldsymbol{b}_2 \in \mathbb{R}^{|\mathcal{C}_{depth}|}$ are learnable parameters. The loss function of the depth probe task can be written as:

$$\mathcal{L}_{depth} = \sum_s \frac{1}{|s|} \sum_{i=1}^{|s|} CE(\hat{\boldsymbol{p}}_i^s, \boldsymbol{p}_i^s) \tag{3.6}$$

For the RE task, we take the vector representation of [CLS] $\boldsymbol{h}_{[CLS]}^s$ to predict the true label of relation $\hat{y}^s$:

$$\boldsymbol{y}^s = \boldsymbol{h}_{[CLS]}^s \boldsymbol{W}_3 + \boldsymbol{b}_3 \tag{3.7}$$

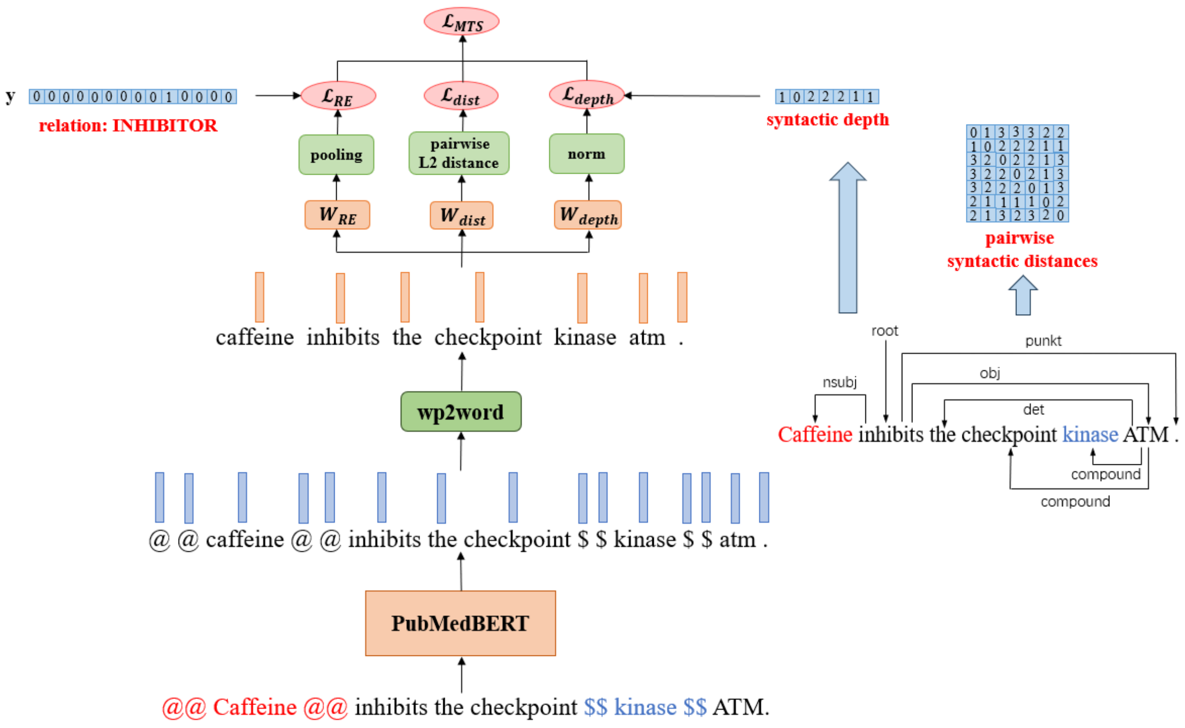


Figure 3.11: the architecture of MTS-PubMedBERT.

where $\boldsymbol{y}^s$ is the one-hot encoding of predicted relation; $\boldsymbol{W}_3 \in \mathbb{R}^{m\times|\mathcal{R}|}$ and $\boldsymbol{b}_3 \in |\mathcal{R}|$ are learnable parameters; $\mathcal{R}$ denotes the set of relation labels. The loss function of the RE task is:

$$\mathcal{L}_{RE} = \sum_{s} CE(\hat{\boldsymbol{y}}^s, \boldsymbol{y}^s) \tag{3.8}$$

The loss of MTS-PubMedBERT is the sum of the above three losses. We add a parameter α to control the importance of syntactic information, the final loss function is:

$$\mathcal{L}_{MTS} = \frac{\mathcal{L}_{RE} + \alpha(\mathcal{L}_{dist} + \mathcal{L}_{depth})}{1+\alpha} \tag{3.9}$$

3.4 Experimentation

In this section, we present details about our experiments of applying four syntax-enhanced models CE-PubMedBERT, CT-PubMedBERT, Late-Fusion and MTS-PubMedBERT, along with two baseline models (Subsection 3.4.2) on three corpora: ChemProt, DrugProt and BB-Rel. We start with data pre-processing (Subsection 3.4.1) that turns texts into data readable by each of the syntax-enhanced models, then present implementation details (Subsection 3.4.3), and hyperparameter search (Subsection 3.4.4). At the end of the section, we present experimental results (Subsection 3.4.5) on each of the three corpora. Our code is available at: `https://github.com/Maple177/syntax-enhanced-RE`.

3.4.1 Data Pre-processing

We start from datasets that are pre-processed as presented in Subsection 2.8.2. To begin with, we use off-the-shelf syntactic parsers to perform the dependency analysis and the constituency analysis on each of the datasets. For the dependency analysis, we use a biomedical version of Stanza (Zhang et al., 2021), which is a variant of the BiLSTM-based deep biaffine model trained on the CRAFT treebank (Verspoor et al., 2012)[1]. Since the CRAFT treebank contains full-text PubMed articles in which entity mentions of chemicals, genes and proteins are annotated, we choose this version of Stanza because we think that texts in the CRAFT treebank are similar to the datasets on which we conduct experiments. For the constituency analysis, we choose a previous state-of-the-art model the Berkeley neural parser (Kitaev and Klein, 2018), which is a self-attentive encoder trained on the Penn Treebank (Marcus et al., 1993). For each of the four syntax-enhanced models, we take the sentence in Figure 1.1 as an example to illustrate the pre-processing steps.

CE-PubMedBERT. The goal of pre-processing for CE-PubMedBERT is to obtain chunks from a constituency tree. Since the constituency analysis splits a sentence into nested parts, the difficulty is to select which chunks to extract from nested constituents. We design an algorithm based on DFS (Depth-First Search) to solve this problem. Iteratively, the DFS traversal of a tree starts at the root node of the tree (or a subtree), then explores its leftmost subtree until the deepest node in the subtree is visited. After that, the algorithm returns to the root node and explores the next subtree from the left. This step is repeated until every node in the tree is visited. As shown in Figure 3.12, in the case of a constituency tree, a DFS traversal retrieving the values of leaf nodes

[1] `https://bionlp-corpora.sourceforge.net/CRAFT/`

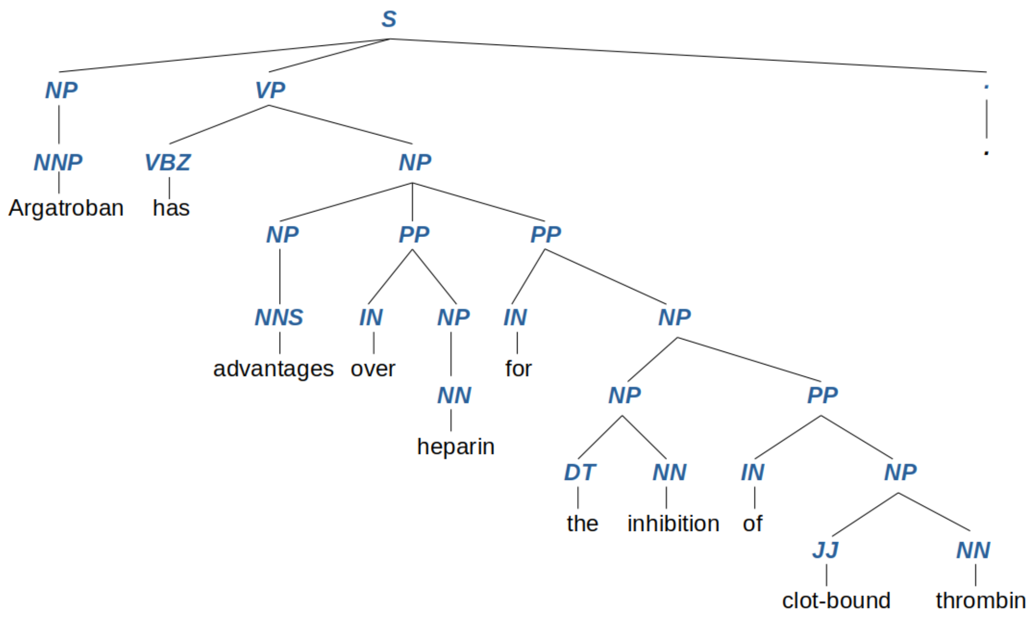


Figure 3.12: The constituency tree that corresponds to the sentence in Figure 1.1.

simply restores the original list of words. We make modifications to the DFS algorithm such that contiguous leaf nodes that share the same lowest ancestor (LCA) node are considered as a chunk (Subsection 3.3.1). For example, in Figure 3.12, “the” and “inhibition” both belong to a constituent “NP” and “NP” is their LCA, they are thus grouped to make up a chunk “the inhibition”. Though “of”, “clot-bound” and “thrombin” belong to the constituent “PP”, “of” is not regrouped with “clot-bound” and “thrombin” since “clot-bound” and “thrombin” share “NP” as their LCA while “of” does not. Besides, nodes labeled by word-level constituent tags are removed, e.g. “heparin” is a word-level constituent “NN” as well as a phrase-level constituent “NP”, but it is only considered as “NP” and thus regrouped with “over”. Therefore, the result of pre-processing for the sentence in Figure 1.1 is: “Argatroban”, “has”, “advantages”, “over heparin”, “for”, “the inhibition”, “of”, “clot-bound thrombin”, “.”. The pseudocode of the modified DFS algorithm is presented in Algorithm 1. After we obtain chunks from the constituency tree, we add entity markers to mark the entity spans. The final pre-processing result that we obtain should be: “@ @ Argatroban @ @”, “has”, “advantages”, “over heparin”, “for”, “the inhibition”, “of”, “clot-bound \$ \$ thrombin \$ \$”, “.”. In practice, we output the list of index spans that correspond to each chunk.

Algorithm 1 Modified DFS algorithm for data pre-processing of CE-PubMedBERT.

```
function DFS($tree$,$index$)
    $L \leftarrow 0$                                        ▷ number of leaf nodes in $tree$
    $S_t \leftarrow []$                                     ▷ index spans of $tree$
    $S_c \leftarrow []$                                     ▷ temporary index spans
    for all $subtree \in tree$ do
        if $subtree$.length=1 then
            $S_c$.append($index$)
            $L \leftarrow L + 1$
            $index \leftarrow index + 1$
        else
            if $S_c$ is not empty then
                $S_t$.append($S_c$)
                $S_c \leftarrow []$
            end if
            $L_{subtree}, S_{subtree} \leftarrow$ DFS($subtree$,$index$)
            $index \leftarrow index + L_{subtree}$
            $L \leftarrow L + L_{subtree}$
            $S_t$.extend($S_{subtree}$)
        end if
    end for
    if $S_c$ is not empty then
        $S_t$.append($S_c$)
    end if
    return $L$, $S_t$
end function
```

CT-PubMedBERT. For CT-PubMedBERT we search to insert constituency tags before the first word of each constituent. We ignore word-level constituency tags to avoid producing too long sequences (due to the length limitation of BERT input). Therefore, word-level constituent tags are ignored. Given the constituency tree in Figure 3.12, the pre-processing result for CT-PubMedBERT is: "[S]", "[NP]", "Argatroban", "[VP]", "has", "[NP]", "advantages", "[PP]", "over", "inhibition", "[PP]", "for", "[NP]", "[NP]", "the", "inhibition", "[PP]", "of", "[NP]", "clot-bound", "thrombin". Constituency tags are wrapped in "[]" because they are added to the vocabulary of BERT and treated as special symbols like [CLS]. We use a DFS-based algorithm similar to Algorithm 1 for CT-PubMedBERT as presented in Algorithm 2.

Algorithm 2 Modified DFS algorithm for data pre-processing of CT-PubMedBERT.

```
1:  function DFS($tree$,$index$)
2:      $L \leftarrow 0$                                ▷ number of leaf nodes in $tree$
3:      $T \leftarrow [tree.label]$                     ▷ linearized $tree$
4:      for all $subtree \in tree$ do
5:          if $subtree$.length=1 then
6:              $T$.append($index$)
7:              $L \leftarrow L + 1$
8:              $index \leftarrow index + 1$
9:          else
10:             $L_{subtree}, T_{subtree} \leftarrow$ DFS($subtree$,$index$)
11:             $index \leftarrow index + L_{subtree}$
12:             $L \leftarrow L + L_{subtree}$
13:             $T$.extend($T_{subtree}$)
14:         end if
15:     end for
16:     return $L$, $T$
17: end function
```

Late-Fusion. The Late-Fusion method consists of injecting dependency information by applying an adjacency matrix as the attention mask in transformer layers (Subsection 3.2.2). Since the adjacency matrix is similar to the attention matrix, it is easy to integrate it into neural models. However, the adjacency matrix is symmetric as it takes dependency relations as undirected, therefore partial information about the dependency tree is lost. Figure 3.13 shows the dependency tree corresponding to the sentence in Figure 1.1. The dependency tree is first extended to word pieces as in (Sachan et al., 2021): for a word tokenized into multiple word pieces, the first word piece is

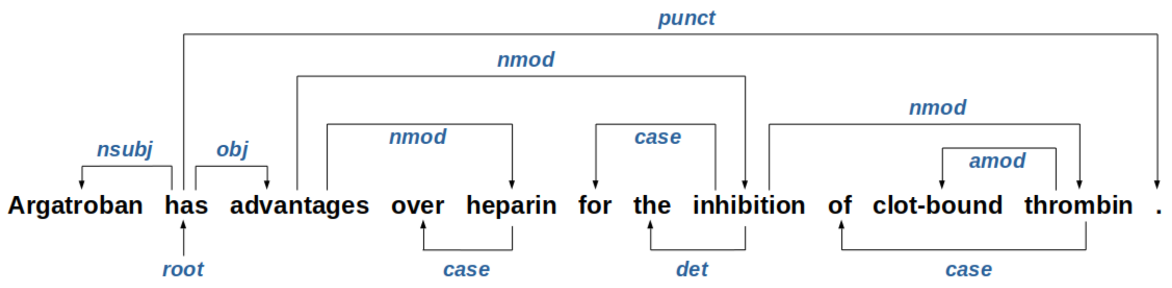

punct
nmod
nmod
nsubj
obj
nmod
case
amod
Argatroban has advantages over heparin for the inhibition of clot-bound thrombin .
root
case
det
case

Figure 3.13: The dependency tree that corresponds to the sentence in Figure 1.1.

used as the representative of the word and is linked to other head word pieces in the adjacency matrix. Subsequent word pieces are linked to the first word piece. We further link [CLS] and [SEP] to the syntactic root of the sentence. Besides, entity markers are linked to the entity. In case of a multi-word entity, entity markers are linked to the syntactic head of the entity. The word piece-level adjacency matrix corresponding to the dependency tree in Figure 3.13 is shown in Figure 3.14.

MTS-PubMedBERT. MTS-PubMedBERT requires pairwise word distances and the depth of each word in the dependency graph as labels. It is common to use the Dijkstra algorithm (Dijkstra, 1959) to find the minimal distance between two nodes in a graph. However, since in the dependency graph edges are regarded as equally weighted, we can use a simpler BFS (Breadth First Search) traversal to solve this problem. Starting from a given node n_s, we first traverse nodes in its 1-hop neighborhood $\mathcal{N}_1$, i.e. nodes that are directly linked to the n_s and are therefore 1 distance away from n_s. Then we iteratively traverse nodes in the i-hop neighborhood of n_s until the target node n_t is reached. The BFS algorithm is illustrated in Algorithm 3. Once we obtain the pairwise distance matrix, we can obtain the depths of words in the dependency tree by retrieving the row that corresponds to the syntactic root in the distance matrix. Figure 3.15 shows the pairwise distance matrix that corresponds to the example in Figure 1.1. As “has” is the root of the dependency tree (Figure 3.13), values in the second row in the matrix represent the depths of words in the dependency tree.

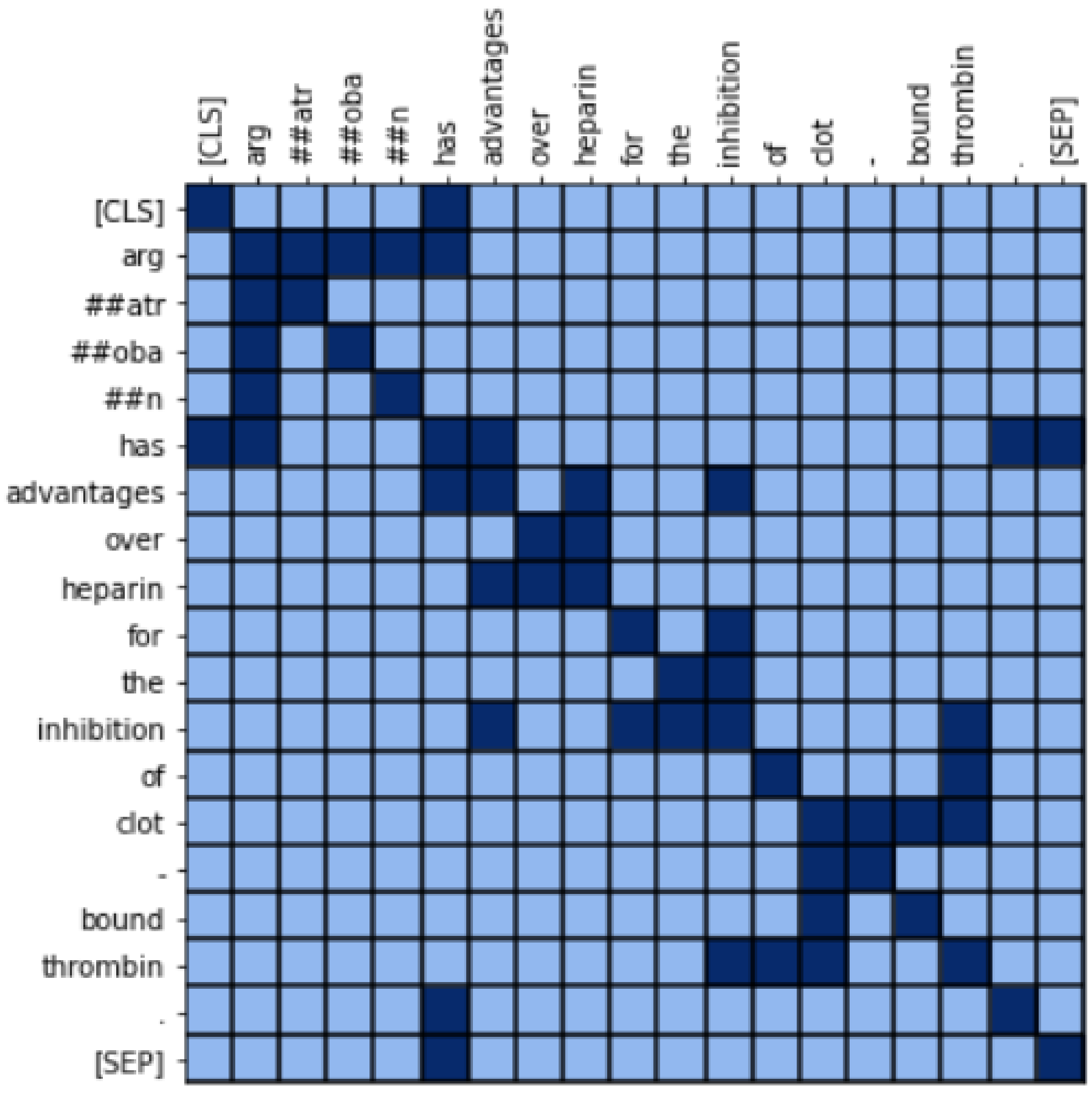
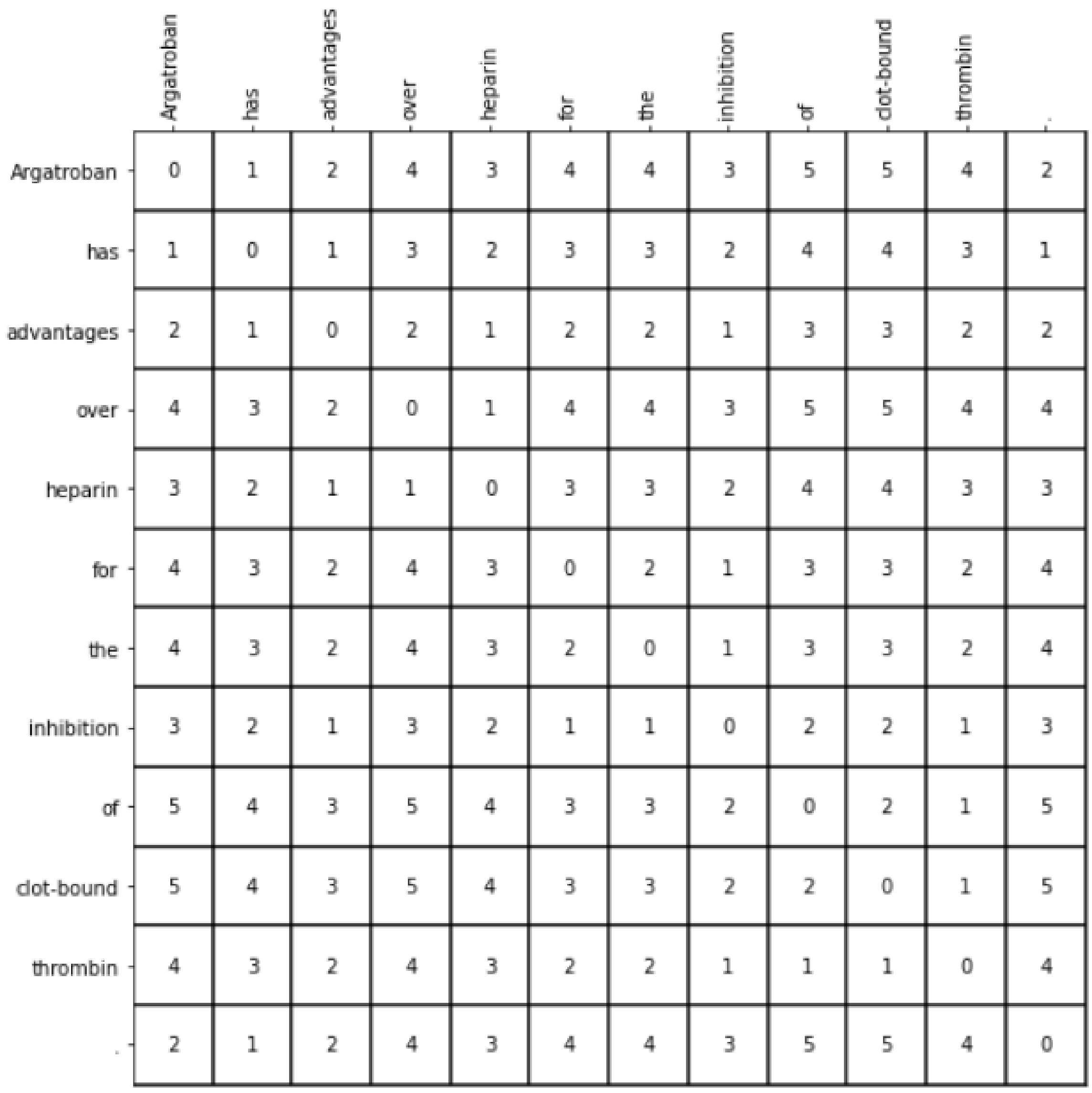


Figure 3.14: The adjacency matrix that corresponds to the dependency tree in Figure 3.13.



	Argatroban	has	advantages	over	heparin	for	the	inhibition	of	clot-bound	thrombin	.
Argatroban	0	1	2	4	3	4	4	3	5	5	4	2
has	1	0	1	3	2	3	3	2	4	4	3	1
advantages	2	1	0	2	1	2	2	1	3	3	2	2
over	4	3	2	0	1	4	4	3	5	5	4	4
heparin	3	2	1	1	0	3	3	2	4	4	3	3
for	4	3	2	4	3	0	2	1	3	3	2	4
the	4	3	2	4	3	2	0	1	3	3	2	4
inhibition	3	2	1	3	2	1	1	0	2	2	1	3
of	5	4	3	5	4	3	3	2	0	2	1	5
clot-bound	5	4	3	5	4	3	3	2	2	0	1	5
thrombin	4	3	2	4	3	2	2	1	1	1	0	4
.	2	1	2	4	3	4	4	3	5	5	4	0

Figure 3.15: The pairwise distance matrix that corresponds to the dependency tree in Figure 3.13.

Algorithm 3 BFS algorithm for data pre-processing of MTS-PubMedBERT.

```
 1: function BFS($G$,$n_s$,$n_t$)
 2:     $V \leftarrow \{n_s\}$                          ▷ A set saving nodes that are already visited
 3:     $Q \leftarrow [(n_s, 0)]$                       ▷ A queue saving nodes to visit
 4:     while $Q$ is not empty do
 5:         $u, d \leftarrow Q$.popleft()
 6:         for all edges from $u$ to $v \in G$ do
 7:             if $v$ is $n_t$ then
 8:                 return $d + 1$
 9:             end if
10:             if $v \notin V$ then
11:                 $V$.add($v$)
12:                 $Q$.append(($v$,$d + 1$))
13:             end if
14:         end for
15:     end while
16: end function
```

3.4.2 Baseline Models

We set two baseline models in our experiments: PubMedBERT and PubMedBERT-extra. Since extra attention layers are added in CE-PubMedBERT and Late-Fusion, we need to exclude the impact of extra attention layers. We thus build PubMedBERT-extra by adding a certain number of attention layers atop the pre-trained PubMedBERT without introducing any syntactic information. Similar to CE-PubMedBERT and Late-Fusion, the outputs of the pre-trained PubMedBERT and added attention layers are passed to the Highway Gate (Srivastava et al., 2015) as shown in Figure 3.16.

3.4.3 Implementation Details

Implementation of all models is based on HuggingFace's transformer (Wolf et al., 2020) and Pytorch (Paszke et al., 2019b). For all of the six models (four syntax-enhanced models and two baseline models), we initialize parameters from PubMedBERT checkpoints released on Huggingface[2]. In cases where extra attention layers (as in CE-PubMedBERT or Late-Fusion) are added atop PubMedBERT, extra layers are randomly initialized. Specifically, for CT-PubMedBERT, we add 24

[2] `https://huggingface.co/microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract`

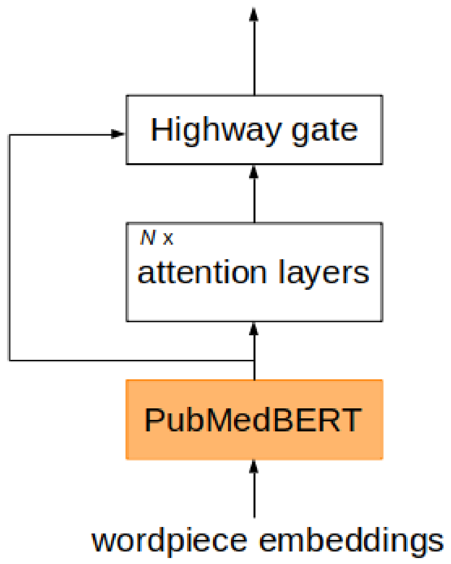


Figure 3.16: The architecture of PubMedBERT-extra (one of our baseline models).

constituency tags as new word pieces into the vocabulary of the PubMedBERT word piece tokenizer, and randomly initialize corresponding word piece embeddings. We fine-tune each model with mono 32 G NVIDIA V100 Graphics processing unit (GPU) for a fixed number of epochs, and save the checkpoint that gives the best performance on the validation set during fine-tuning. Table 3.4 summarizes the number of epochs that are used for each (corpus, model type) combination. We use the AdamW (Loshchilov and Hutter, 2017) optimizer in all experiments with a learning rate scheduler. The learning rate is linearly increased to a target value during the first 10 % steps and then linearly decays to 0 at the end of training. In Subsection 2.2.3 we have introduced the ensembling technique, we adopt this strategy in our experiments. For each model type, we fine-tune the same neural architecture 5 times with different random seeds and combine the results of all models by majority voting. Random seeds are fixed across different model types to exclude the influence of random initializations.

3.4.4 Hyperparameters

For each type of model, we use grid search to determine the optimal hyperparameters. Instead of applying the same grid for all models, we establish first a hyperparameter grid and then slightly

	BB-Rel	ChemProt	DrugProt
PubMedBERT	60	60	20
PubMedBERT-extra	60	60	20
CE-PubMedBERT	60	60	20
CT-PubMedBERT	60	60	20
Late-Fusion	60	60	20
MTS-PubMedBERT	60	30	10

Table 3.4: Fixed number of epochs for each (corpus,model type) combination.

adjust the search space for each model according to preliminary experiments. Due to the limitation of computing resources, we fix the batch size to 16 for all model types except MTS-PubMedBERT. The batch size for MTS-PubMedBERT is fixed at 8 due to the RAM (Random Access Memory) limitation. Our hyperparameter grid search therefore focuses on three hyperparameters: learning rate (LR), the number of extra attention layers (EAL) and the weighting coefficient (α) of syntactic loss (for MTS-PubMedBERT only). Adapted hyperparameter grids of each model are summarized in Table 3.5. '-' indicates that the corresponding hyperparameter is not useful for the corresponding model type. We test hyperparameter combinations in the Cartesian product of the available parameter set: for example, for the Late-Fusion model, we construct 9 ensembles each with one parameter combination out of 9. On each corpus, the hyperparameter combination that leads to the best performance on the validation set is used for inference on the test set. For all our experiments, we use a single NVIDIA Tesla V100 GPU of the Lab-IA GPU cluster (provided by the Saclay-IA platform of Université Paris-Saclay).

Model type	LR	EAL	α
PubMedBERT	$\{1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5}\}$	-	-
PubMedBERT-extra	$\{1e^{-5}, 3e^{-5}, 5e^{-5}\}$	$\{1, 2, 4\}$	-
CE-PubMedBERT	$\{1e^{-5}, 3e^{-5}, 5e^{-5}\}$	$\{1, 2, 4\}$	-
CT-PubMedBERT	$\{1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5}\}$	-	-
Late-Fusion	$\{1e^{-5}, 3e^{-5}, 5e^{-5}\}$	$\{1, 2, 4\}$	-
MTS-PubMedBERT	$\{3e^{-5}, 5e^{-5}, 7e^{-5}\}$	-	$\{0.1, 0.5, 1.0\}$

Table 3.5: Hyperparameter optimization with grid search: customized grids of each model.

Model type	LR	EAL	α
PubMedBERT	$3e^{-5}$	-	-
PubMedBERT-extra	$3e^{-5}$	4	-
CE-PubMedBERT	$3e^{-5}$	2	-
CT-PubMedBERT	$2e^{-5}$	-	-
Late-Fusion	$3e^{-5}$	1	-
MTS-PubMedBERT	$3e^{-5}$	-	0.1

(a) Optimal hyperparameters on BB-Rel.

Model type	LR	EAL	α
PubMedBERT	$1e^{-5}$	-	-
PubMedBERT-extra	$1e^{-5}$	2	-
CE-PubMedBERT	$3e^{-5}$	4	-
CT-PubMedBERT	$1e^{-5}$	-	-
Late-Fusion	$1e^{-5}$	1	-
MTS-PubMedBERT	$3e^{-5}$	-	0.1

(b) Optimal hyperparameters on ChemProt.

Model type	LR	EAL	α
PubMedBERT	$2e^{-5}$	-	-
PubMedBERT-extra	$1e^{-5}$	2	-
CE-PubMedBERT	$1e^{-5}$	4	-
CT-PubMedBERT	$2e^{-5}$	-	-
Late-Fusion	$1e^{-5}$	4	-
MTS-PubMedBERT	$3e^{-5}$	-	0.1

(c) Optimal hyperparameters on DrugProt.

Table 3.6: Optimal combination of hyperparameters for each (corpus, model type) based on the performance on the validation set: learning rate (LR), number of extra attention layers (EAL), α.

3.4.5 Results

In this part, we present the results that we obtain for each of the six model types: PubMedBERT, PubMedBERT-extra, CE-PubMedBERT, CT-PubMedBERT, Late-Fusion and MTS-PubMedBERT. For each model type, we report the performance on the test set using the optimal combination of hyperparameters. The summary of optimal hyperparameters is shown in Table 3.6. The detailed summary of experimental results is presented in Table 3.7. We present voting F1-scores for all model types. To give an overview of the variance between models in the same ensemble, we also show averaged F1-scores with standard deviation for each model type.

Model	**F1** (vote)	%Δ	**F1** (ave ± std)
Baseline Models			
PubMedBERT	68.8	-	66.9 ± 1.2
PubMedBERT-extra	**70.6**	**+1.8**	$\mathbf{68.1 \pm 1.2}$
Syntax-enhanced Models			
CE-PubMedBERT	70.2	+1.4	67.6 ± 1.8
CT-PubMedBERT*	65.9	-2.9	64.3 ± 1.7
Late-Fusion	68.9	+0.1	66.6 ± 1.5
MTS-PubMedBERT	68.2	-0.6	67.1 ± 1.5

(a) Results on BB-Rel.

Model	**F1** (vote)	%Δ	**F1** (ave ± std)
Baseline Models			
PubMedBERT	78.2	-	77.1 ± 0.4
PubMedBERT-extra	**78.4**	**+0.2**	$\mathbf{77.3 \pm 0.5}$
Syntax-enhanced Models			
CE-PubMedBERT†	78.0	-0.2	76.6 ± 0.5
CT-PubMedBERT*	75.6	-2.6	73.6 ± 1.2
Late-Fusion	77.9	-0.3	76.9 ± 0.2
MTS-PubMedBERT*	73.9	-4.3	72.2 ± 0.6

(b) Results on ChemProt.

Model	**F1** (vote)	%Δ	**F1** (ave ± std)
Baseline Models			
PubMedBERT	77.2	-	75.8 ± 0.5
PubMedBERT-extra*	77.8	+0.6	76.6 ± 0.4
Syntax-enhanced Models			
CE-PubMedBERT	77.3	+0.1	76.3 ± 0.7
CT-PubMedBERT*	74.4	-2.8	73.3 ± 0.3
Late-Fusion*	**78.0**	**+0.8**	$\mathbf{76.8 \pm 0.4}$
MTS-PubMedBERT*	72.5	-4.7	71.2 ± 0.5

(c) Results on DrugProt.

Table 3.7: Voting F1-score for each model type (in the second column) on the three corpora. %Δ denotes the relative gain in voting F1 over PubMedBERT. The averaged F1-score and standard deviation calculated over 5 independent runs within ensembles are presented in the rightmost column. * and † indicate that the performance of the corresponding model is significantly different from PubMedBERT and PubMedBERT-extra respectively under a one-sided t-test with $p < 0.05$ (we only compare CE-PubMedBERT and Late-Fusion to PubMedBERT-extra).

3.5 Analysis & Discussion

Our first observation based on Table 3.7 is that CT-PubMedBERT and MTS-PubMedBERT give worse performances than PubMedBERT in terms of voting F1-scores on the three corpora: BB-Rel, ChemProt and DrugProt. Regarding average F1-scores, CT-PubMedBERT degrades the performance by a significant margin on the three corpora, and MTS-PubMedBERT degrades the performance significantly on ChemProt and DrugProt. Though MTS-PubMedBERT outperforms PubMedBERT on BB-Rel regarding average F1-scores, the slight gain is not statistically significant. Surprisingly, PubMedBERT-extra consistently outperforms PubMedBERT, and the difference is even statistically significant on DrugProt. CE-PubMedBERT and Late-Fusion obtain high F1-scores and outperform PubMedBERT on BB-Rel and DrugProt, but their performance is never statistically better than PubMedBERT-extra, therefore the improvement is not caused by syntactic information but is more likely due to added attention layers. Based on the general performances, our first conclusion is that integrated syntactic information, either constituency or dependency information, does not help improve the performance on the RE task. It is noteworthy that for Late-Fusion, we obtain similar results to Sachan et al. (2021). Their experimental results show that Late-Fusion slightly improves the performance of RE over BERT (+0.29) on a general-domain RE corpus. In our case, the gain obtained by Late-Fusion over PubMedBERT is also not significant (respectively +0.1, -0.3, and +0.8 respectively on BB-Rel, ChemProt and DrugProt). However, since they do not set a baseline like PubMedBERT-extra as we do, they attribute this improvement to integrated dependency information. We dispute this conclusion and think that the slight improvement is due to extra attention layers added in Late-Fusion.

3.5.1 Impact of Parsing Quality

Sachan et al. (2021) study the impact of parsing quality and report that substantial gains are observed only when gold parses are used. We thus hypothesize that the degradation (CT-PubMedBERT, MTS-PubMedBERT) or the ineffectiveness (CE-PubMedBERT, Late-Fusion) of injected syntactic information might be partly due to the quality of syntactic parses obtained from the off-the-shelf parsers. Since manually performing constituency analysis and comparing two constituency parses are more complicated compared to dependency analysis, we choose to only analyze the parsing quality of Stanza (Zhang et al., 2021), which is the dependency parser that is used in our experiments. We manually evaluated dependency analysis on a small subset of sentences in the validation set of BB-Rel and DrugProt (37 sentences for BB-Rel; 48 sentences for DrugProt). We exclude ChemProt due to the similarity between ChemProt and DrugProt: they both contain PubMed abstracts and both consist of identifying chemical-gene interactions. Firstly, we divide sentences in the validation set into groups according to the length of the dependency path between the two arguments of the candidate relation. Then we randomly sample sentences from each group. For each sentence, we manually examine the dependency parse provided by Stanza and report two values: the number of erroneous dependency links on the full parses and the number of erroneous dependency links in the dependency path between the two arguments of the candidate relation. The number of errors on the full-dependency parses allowed us to compute an "Unlabelled Attachment" Precision score of 91.13 (on BB-Rel) and 94.96 (on DrugProt). Compared to Stanza reporting 91.09 as Unlabeled Attachment Score on the CRAFT treebank (on which the version of Stanza that we use is trained), although we cannot make a direct comparison (since their score is an F1-score), it is fair to conclude that the quality of dependency parses from Stanza is acceptable on the subset of sentences that we extracted.

During the manual analysis, we identified two issues in BB-Rel that may cause unnecessary errors. Firstly, as mentioned in Subsection 2.8.2, processed data of BB-Rel contains examples that consist of two sentences (for inter-sentence relations). As we input these examples directly to Stanza, they are treated as a single sentence. Stanza has difficulty analyzing dependency relations between words originally from different sentences. To verify the impact of having double-sentence examples, we correspondingly divide the examples in the validation set of BB-Rel into two groups, and report the RE performance of PubMedBERT, Late-Fusion and MTS-PubMedBERT on each group. We report the average performance of Late-Fusion and MTS-PubMedBERT (for each architecture we train multiple models) respectively on the two groups. Denote the performance on two-sentence examples by ave_{ds}; the performance on single-sentence examples by ave_{ss}. The result is given in Table 3.8. We observe that the performance of double-sentence examples is consistently much lower than that of single-sentence examples, which is not out of our expectation due to the difficulty of predicting inter-sentence relations. However, the value of $\frac{ave_{ss}}{ave_{ds}}$ of Late-Fusion and MTS-PubMedBERT drop compared to that of PubMedBERT, indicating the negative impact of having double-sentence examples for syntax-enhanced models. For Late-Fusion, the injected dependency information helps improve the RE performance on single-sentence examples, but degrades the RE performance on double-sentence examples. For MTS-PubMedBERT, the dependency information degrades performance on all examples, but the degradation is greater on double-sentence examples than on single-sentence ones. The second issue in the BB-Rel corpus is the existence of many abbreviation-related periods. Since in BB-Rel abbreviations of microorganism names such as "E. coli" and "B. fragilis" are frequent, Stanza is likely to be confused by the periods contained in the abbreviations.

	ave_{ds}	ave_{ss}	$\frac{ave_{ss}}{ave_{ds}}$
PubMedBERT	0.20	0.77	3.87
Late-Fusion	0.17	0.78	4.45
MTS-PubMedBERT	0.05	0.76	15.91

Table 3.8: Stratified results on the validation set of BB-Rel: ave_{ds} denotes the average micro F1-score of the corresponding model on the group of examples that consist of two sentences; ave_{ss} denotes that on the group on examples that consist of a single sentence.

We further analyze whether there exists a negative correlation between the number of errors (in full sentences and in SDPs) in the dependency parse and the RE performance of Late-Fusion and MTS-PubMedBERT, which are enhanced with dependency information. We gather sentences that share the same number of errors and calculate micro F1-scores on sentences from each group. Note that "no_relation" is not counted in the calculation of micro F1-scores, so we remove groups that include only examples labeled as "no_relation" (otherwise, in the stratified results, zero micro F1-scores would be reported for these groups, which are unrepresentative). The stratified results on BB-Rel and DrugProt are respectively shown in Figure 3.17 and Figure 3.18. We observe that on both BB-Rel and DrugProt, the stratified F1-score shows no sign of being correlated to the number of total errors; meanwhile the stratified score decreases with the number of errors in the SDP, though

Model	r_s	p-value
# total errors		
Late-Fusion	-0.12	0.41
MTS-PubMedBERT	-0.12	0.41
# errors in the SDP		
Late-Fusion	-0.5	0.33
MTS-PubMedBERT	-0.5	0.33

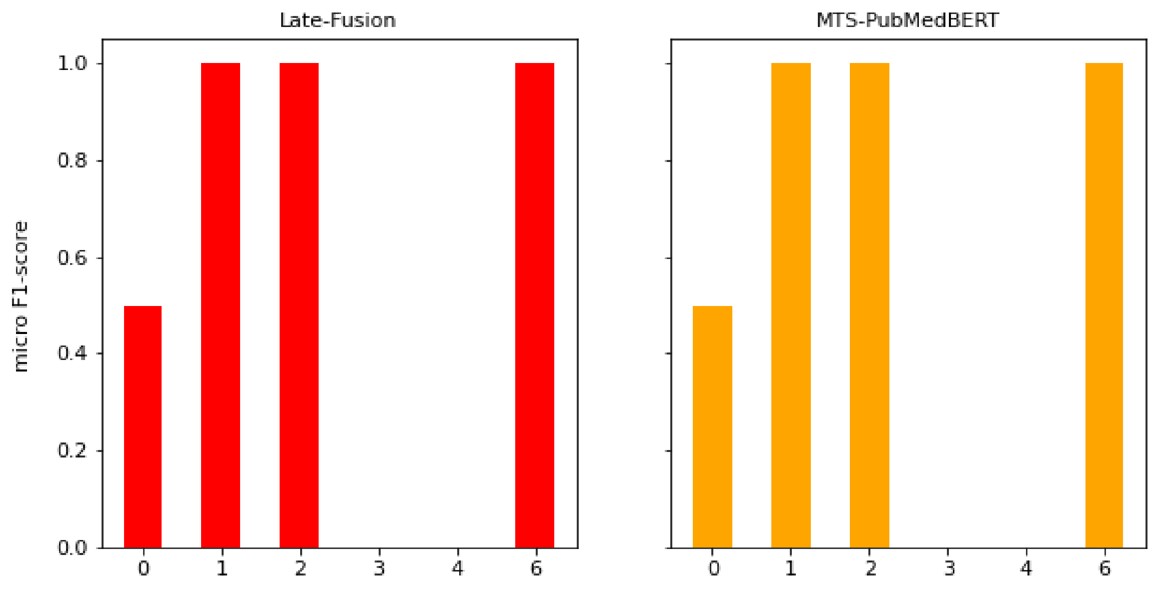
(a) Analysis using the Spearman's rank coefficient on BB-Rel.

Model	r_s	p-value
# total errors		
Late-Fusion	0.27	0.70
MTS-PubMedBERT	0.09	0.56
# errors in the SDP		
Late-Fusion	-0.80	0.10
MTS-PubMedBERT	-0.63	0.18

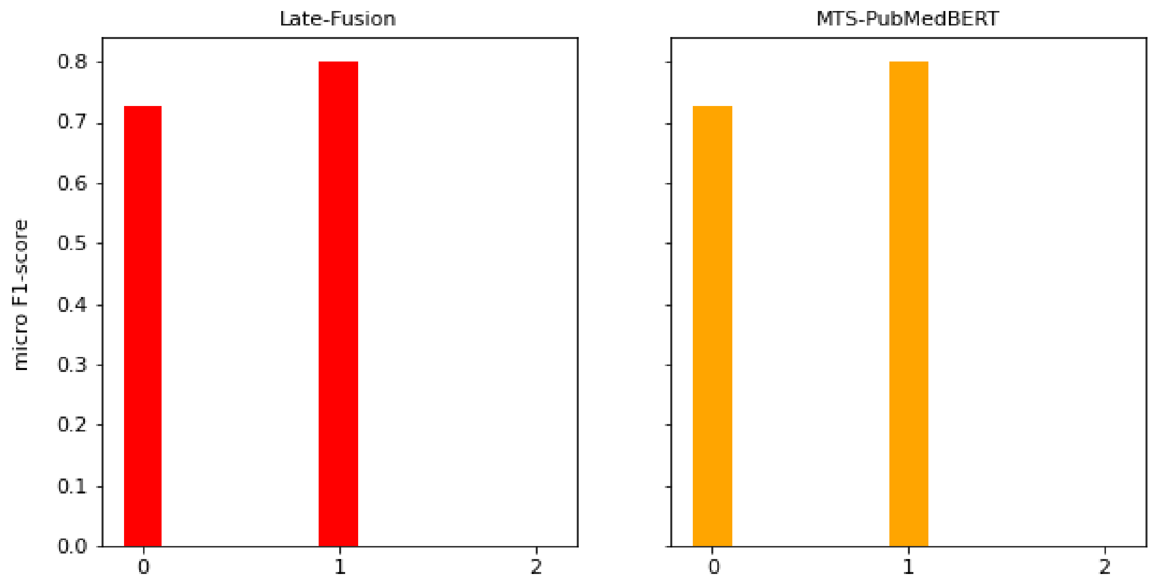
(b) Analysis using the Spearman's rank coefficient on DrugProt.

Table 3.9: Spearman's rank coefficient r_s and p-value between the number of errors (in full parses and in SDPs) and the stratified micro F1-score obtained by Late-Fusion and MTS-PubMedBERT. The analysis is performed on BB-Rel and DrugProt.

not monotonically. To confirm that, we further compute the Spearson's rank coefficient between the two variables, the result is given in Table 3.9. We report the correlation coefficient r_s for which a value of 1 indicates an exact positive correlation, -1 indicates an exact negative correlation and 0 indicates no correlation. Besides, a p-value is computed, representing the possibility of rejecting the hypothesis that input two variables are correlated. In our case, we expect a negative r_s close to -1 and a small p-value (< 0.05), which indicates that a strong negative correlation exists. Our first observation from Table 3.9 is that on both BB-Rel and DrugProt, the number of total errors is not correlated to the RE performance since r_s values are close to 0 and p-values are high. In terms of the number of errors in the SDP, for BB-Rel there is no sign of negative correlation due to the r_s of -0.5 and p-values of 0.33; for DrugProt, both r_s of Late-Fusion and MTS-PubMedBERT are close to -1 ($<$-0.5), and both p-values are relatively small. However, both p-value are still above 0.05, which means the null hypothesis does not hold and there is no statistically negative correlation between the number of errors in the SDP and the RE performance. In a nutshell, we find no statistically negative correlation between the parser quality and the RE performance.

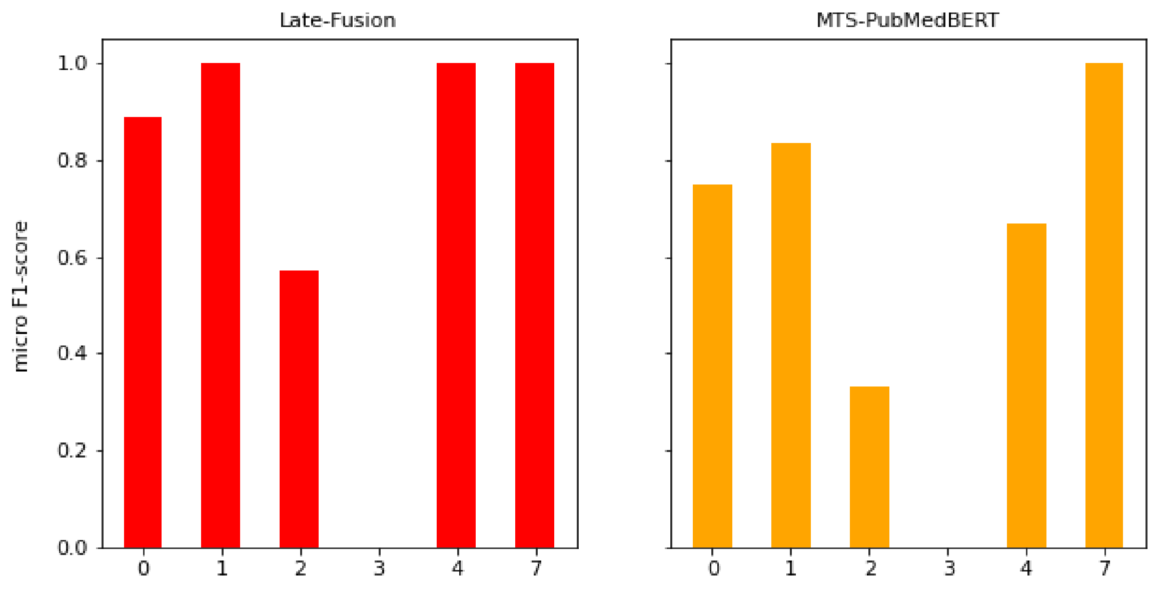


(a) Stratified results on BB-Rel: by the number of total errors in the dependency parse.

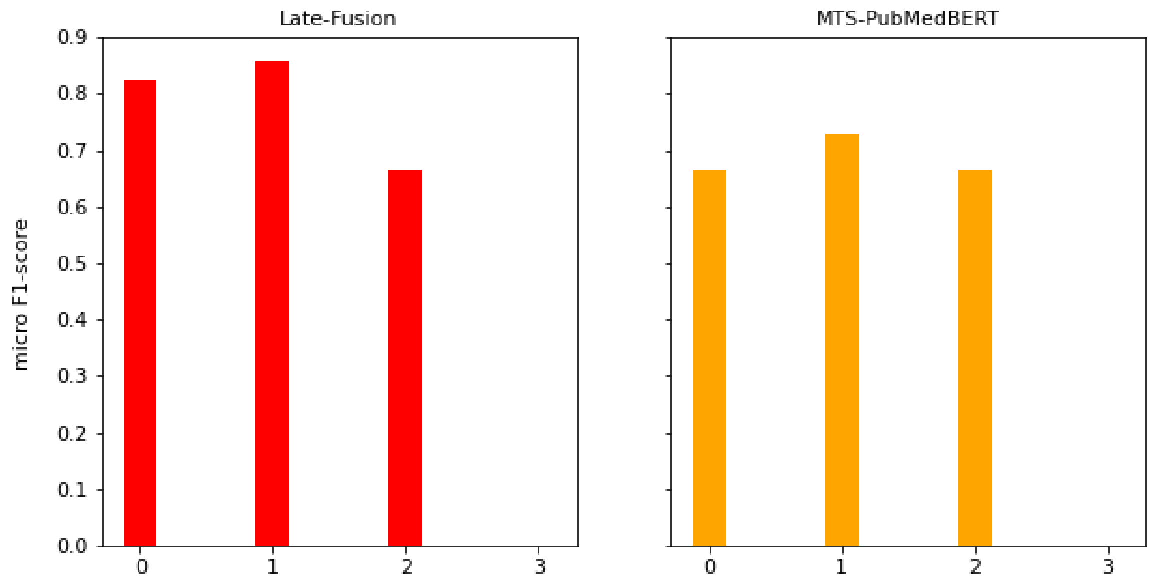


(b) Stratified results on BB-Rel: by the number of errors in the SDP.

Figure 3.17: Stratified RE performance on the subset of sentences used for manual dependency analysis in BB-Rel.



(a) Stratified results on DrugProt: by the number of total errors in the dependency parse.



(b) Stratified results on DrugProt: by the number of errors in the SDP.

Figure 3.18: Stratified RE performance on the subset of sentences used for manual dependency analysis in DrugProt.

3.5.2 Training difficulties

Besides the quality of syntactic parsers, we also hypothesize that the degradation in performance of the CT-PubMedBERT and MTS-PubMedBERT models may be due to the difficulty of training neural networks. We had speculated that constituency tag information and hierarchical information of the constituency tree encoded in the linearized sequence would compensate the possible degradation caused by adding constituent tags as new word pieces, but this hypothesis is not supported by the experimental results. For MTS-PubMedBERT, using the same learning rate to jointly train the neural network on three tasks may not be ideal for making a balance between the three training objectives.

3.5.3 Difference Between Baseline Models and Syntax-enhanced Models

Though the improved performance of CE-PubMedBERT and Late-Fusion is likely due to added attention layers, and CT-PubMedBERT and MTS-PubMedBERT consistently degrade the performance of RE, we are unsure about if syntax-enhanced models behave similarly to baseline models. More specifically, we want to figure out if syntax-enhanced models give better performance than baseline models on long-distance relations. We examine the following questions:

1. Do syntax-enhanced models make unique errors that differ from errors of baseline models?
2. Do syntax-enhanced models outperform baseline models on long-distance relations?

To answer the first question, we choose to identify erroneous predictions made by each of the six models in the validation set of each corpus. To examine the difference between groups of wrong predictions, common mistakes made by all models are removed. Because it is hard to manually examine these errors and find the differences, for each group of mistakes we choose to calculate statistics related to the inter-argument distance: the number of words between the arguments of a candidate relation denoted by $d_{subj-obj}$; the length of the dependency path between the arguments denoted by d_{SDP}. The result is shown in Table 3.10.

We observe that on ChemProt, for all syntax-enhanced models, $d_{subj-obj}$ and d_{SDP} are greater than that of PubMedBERT. Though on DrugProt, except for MTS-PubMedBERT, $d_{subj-obj}$ and

Model	# unique errors	$\bar{d}_{subj-obj}$	$\bar{d}_{SDP}$
PubMedBERT	133	20.3	5.8
PubMedBERT-extra	117	17.8	5.6
CE-PubMedBERT	122	21.6†	5.9
CT-PubMedBERT	164	22.0	5.5
Late-Fusion	112	23.5†	6.1†
MTS-PubMedBERT	146	20.1	5.4*

(a) Statistics on the validation set of BB-Rel.

Model	# unique errors	$\bar{d}_{subj-obj}$	$\bar{d}_{SDP}$
PubMedBERT	365	16.2	5.9
PubMedBERT-extra	359	15.8	6.0
CE-PubMedBERT	305	17.7†	6.1
CT-PubMedBERT	580	18.7*	6.1
Late-Fusion	349	16.7	6.1
MTS-PubMedBERT	554	18.6*	5.9

(b) Statistics on the validation set of ChemProt.

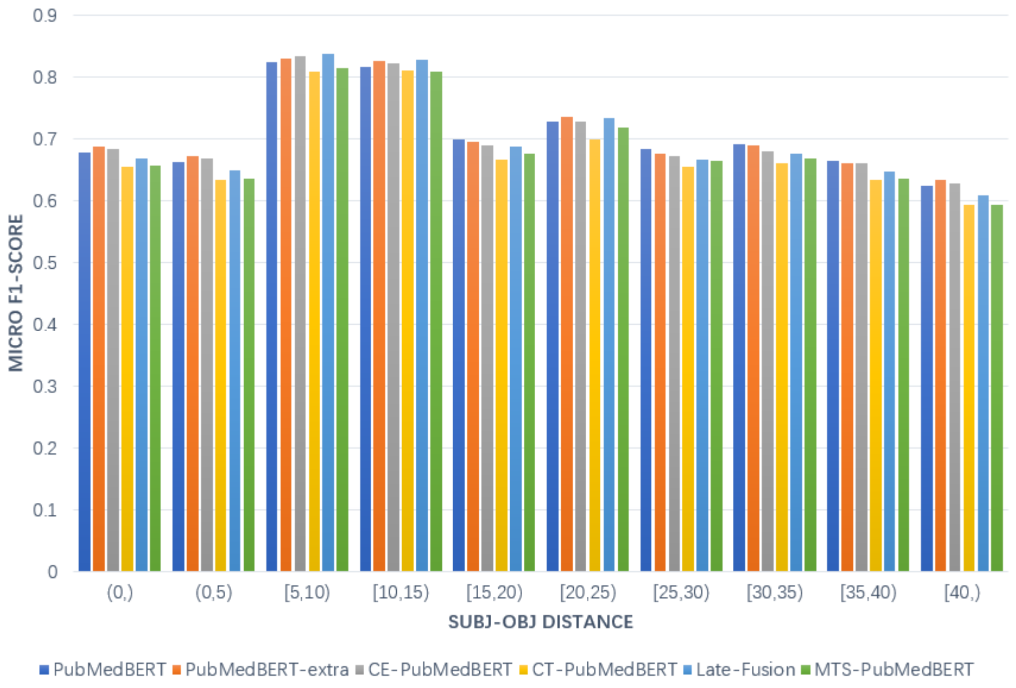
Model	# unique errors	$\bar{d}_{subj-obj}$	$\bar{d}_{SDP}$
PubMedBERT	602	20.8	6.3
PubMedBERT-extra	545	19.7	6.1*
CE-PubMedBERT	545	20.2	6.3
CT-PubMedBERT	671	20.4	6.1
Late-Fusion	564	19.9	6.2
MTS-PubMedBERT	860	21.3	6.2

(c) Statistics on the validation set of DrugProt.

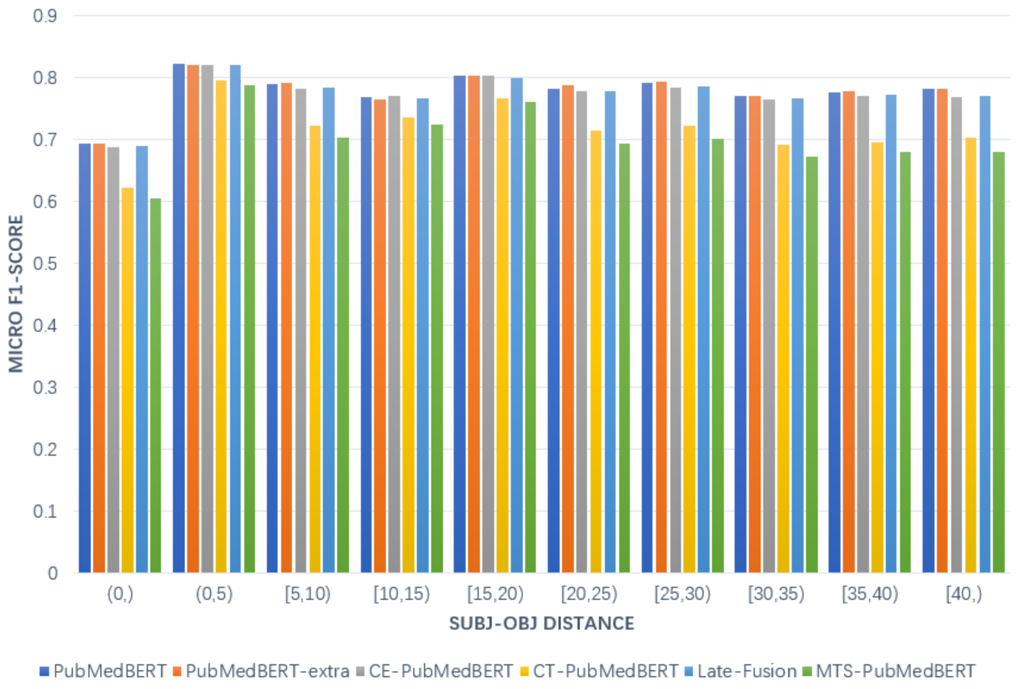
Table 3.10: Statistics of unique errors made by each of the following models: PubMedBERT, PubMedBERT-extra, CE-PubMedBERT, CT-PubMedBERT, Late-Fusion and MTS-PubMedBERT. $\bar{d}$ denotes the average value of the corresponding group of distances. * and † respectively indicate that the value of the corresponding model is significantly different from that of PubMedBERT and PubMedBERT-extra under a one-sided t-test with $p < 0.05$ (we only compare CE-PubMedBERT and Late-Fusion to PubMedBERT-extra).

d_{SDP} of syntax-enhanced models are slightly smaller compared to PubMedBERT, the difference is not statistically significant. On BB-Rel, inversely only $d_{subj-obj}$ and d_{SDP} of MTS-PubMedBERT are smaller compared to PubMedBERT; in terms of d_{SDP}, the difference is even statistically significant. To our surprise, on the three corpora, $d_{subj-obj}$ or d_{SDP} of PubMedBERT-extra is smaller than PubMedBERT; the difference is even statistically significant in terms of d_{SDP} on DrugProt. By comparing CE-PubMedBERT, Late-Fusion to PubMedBERT-extra, we observe that $d_{subj-obj}$ and d_{SDP} of CE-PubMedBERT and Late-fusion are consistently greater than PubMedBERT-extra, in some cases the difference is statistically significant. Therefore, our response to question (1) is that unique errors made by syntax-enhanced models may differ from baseline models, but contrary to our expectation they are slightly more likely to make mistakes on long-distance relations than baseline models.

To confirm if syntax-enhanced models behave differently from baseline models on long-distance relations, we further propose to divide examples in the validation set into groups by $d_{subj-obj}$ and compute per-group micro-average F1-scores of the six models. The result is shown in Figure 3.19. Comparison between baseline models and syntax-enhanced models overturns our hypothesis that integrated syntactic information may improve performance on long-distance relations. CT-PubMedBERT and MTS-PubMedBERT are consistently worse than PubMedBERT in each distance interval; while the performance of CE-PubMedBERT and Late-Fusion are highly correlated to the performance of PubMedBERT-extra. Therefore, we conclude that improvement of CE-PubMedBERT and Late-Fusion is caused by added attention layers rather than injected syntactic information. There is another argument that supports this conclusion: In MTS-PubMedBERT we add a parameter α to control the importance of the losses of syntactic probe tasks. As shown in Table 3.6 the optimal α for MTS-PubMedBERT on each corpus is 0.1 (out of $\{0.1, 0.5, 1.0\}$), proving that MTS-PubMedBERT tends to learn less syntactic information in order to obtain better performance on the validation set.

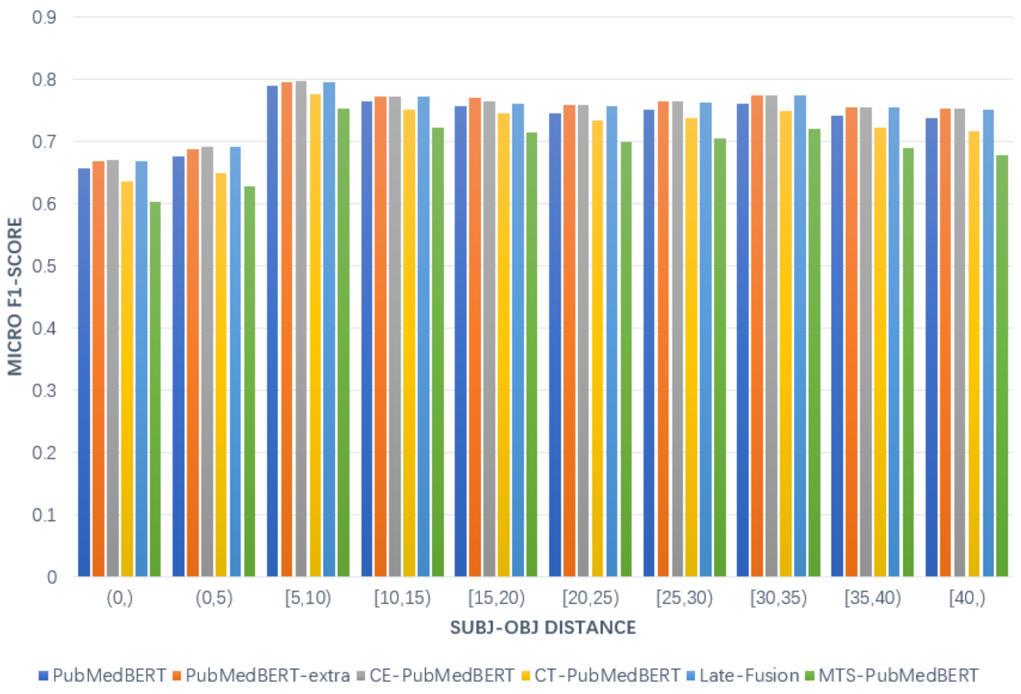


(a) Stratified results on BB-Rel.



(b) Stratified results on ChemProt.

Figure 3.19: Stratified results on the validation set of three corpora: BB-Rel, ChemProt, DrugProt. Examples in the validation set are regrouped based on their subject–object surface distances. Intervals are of length 5 except two special cases 0 and ≥ 40.



(c) Stratified results on DrugProt.

Figure 3.19: Stratified results on the validation set of three corpora: BB-Rel, ChemProt, DrugProt. Examples in the validation set are regrouped based on their subject–object surface distances. Intervals are of length 5 except two special cases 0 and ≥40.

3.6 Conclusion

In this chapter, we have studied the effect of integrating syntactic information into pre-trained PubMedBERT on biomedical RE tasks. We first reviewed studies on the syntactic probes (Subsection 3.2.1) that demonstrate the ability of BERT to encode syntactic information, then introduced existing syntax-enhanced methods (Subsection 3.2.2). A limitation of existing methods is that they focus on general-domain corpora and most of them search to inject dependency information in the form of the adjacency matrix. This motivated us to propose our syntax-enhanced models that inject both constituency and dependency information, and systematically evaluate our models on three biomedical RE corpora: BB-Rel, ChemProt, DrugProt. We conducted experiments (Section 3.4) using three proposed models and an existing model: CE-PubMedBERT, CT-PubMedBERT, Late-Fusion and MTS-PubMedBERT. Aside from PubMedBERT, we set another baseline model PubMedBERT-extra (Subsection 3.4.2) as contrast to CE-PubMedBERT and Late-Fusion. Experimental results show that among syntax-enhanced models, PubMedBERT-extra, CE-PubMedBERT and Late-Fusion consistently give better or close performance compared to PubMedBERT; while CT-PubMedBERT and MTS-PubMedBERT consistently degrade the RE performance possibly due to training difficulties (Subsection 3.5.2).

We then perform an analysis that focuses on two aspects: (1) Does the quality of syntactic parses influence the RE performance? (Subsection 3.5.1) (2) Do syntax-enhanced models behave differently compared to baseline models? (Subsection 3.5.3) We manually identified errors in the dependency parses to respond to the first question. Due to the double-sentence problem (Subsection 3.5.1) that we found existing in BB-Rel, we perform a stratification analysis and find that injected dependency information degrades the RE performance on double-sentence examples (on which Stanza is considered likely to produce errors). This observation seems to prove our hypothesis that the parsing quality is positively correlated to the RE performance of syntax-enhanced models. We further conduct a quantitative analysis by statistically identifying if a negative correlation exists between the number of errors in dependency parses and the RE performance of syntax-enhanced models. To our surprise, statistical results show that no significant negative correlation exists. The two observations from the stratified results and statistical results seem to be contradictory but can be explained by the fact that the statistical results are built on small-sized examples (85 examples). This opens up a perspective for more extensive analysis that would involve manual analysis of more examples. Manually examining errors in constituency parses is also worth investigating. For the second question, our hypothesis was that syntax-enhanced models may better handle long-distance relations. Therefore, we first examined unique errors made by each model and expected that baseline models tend to make errors in long-distance relations. However, our statistical results prove the opposite. We then stratified the examples on the validation set of each corpus based on the subject-object distance of the candidate relation. We observe that CE-PubMedBERT and Late-Fusion outperform PubMedBERT for long-distance relations (subj-obj distance ≥ 40); however, their performances are highly correlated to another baseline model, PubMedBERT-extra. Based on this observation, we attribute the improvements made by CE-PubMedBERT and Late-Fusion to added attention layers rather than injected syntactic information.

Our proposed methods do not succeed in improving the RE performance by injecting syntactic information. Nevertheless, it does not mean that syntactic information can never be helpful for biomedical RE, only that it isn't so in the settings that we investigated. Our experimental results may be biased to the neural architectures that we designed. Training strategies also play an important role. Our analysis needs further investigation to consolidate the conclusions that we obtained. Besides, increasing the number of corpora and testing different base models is also important for future work.

Chapter 4

Injecting KB Information into BERT

Factual knowledge refers to common knowledge in any domain. Since the pre-training of BERT does not explicitly incorporate any factual knowledge, we hypothesize that injecting factual knowledge may improve the ability of BERT. Mastering factual knowledge can be beneficial. For example, it helps eliminate ambiguity. Given the sentence "Apple has increased its investment in AI technology", it is important for BERT to understand that "Apple" is a technology company rather than a fruit. With the help of factual knowledge, it would be easy for BERT to distinguish between the two concepts. This distinction is easy for a layman, but factual knowledge in specific domains such as the biomedical domain is less obvious to non-experts. In some cases, it is hard for a non-expert to deduce if a biomedical term refers to a protein or a chemical without prior knowledge. However, this information can be easily accessed using knowledge bases (KB), which is a set of facts verified by domain experts.

In recent years, injecting factual knowledge from domain KBs into BERT is a topic that has received much attention (Zhang et al., 2019b; Hao et al., 2020). In this chapter, we first introduce knowledge base basics (Section 4.1), then present graph embeddings that vectorize KB information (Section 4.2). Since exploiting graph embeddings is not the only solution to inject factual knowledge, in Section 4.3 we review existing methods that enhance pre-trained BERT-like LMs with KB information (Section 4.3). At the end of the chapter, we propose our KB-enhanced model KB-PubMedBERT, experimental results and analysis (Section 4.4, Section 4.5 and Section 4.6).

Section 4.4, Section 4.5 and Section 4.6 are inspired by Tang et al. (2023).

4.1 Knowledge Base Basics

A knowledge base is often annotated and maintained by a group of domain experts, and frameworks like Semantic Web (Berners-Lee et al., 2023) allow us to publish and share knowledge bases. A KB stores factual knowledge about a specific domain by interpreting knowledge as relations between entities or concepts. For example, knowing that "Apple" is a technology company, in a KB we link the entity "Apple" and the concept "technology company" by a relation "is_a". The difference between entities and concepts is that concepts are abstract while entities are concrete instances of concepts. Therefore, a KB may be hierarchical due to relations between entities and concepts, or relations between low-level concepts and high-level concepts. For example, the entity "Labrador" is an example of the concept "Dog"; and "Dog" is a subtype of a more abstract concept "Animal". In this section, for simplicity we use "entity" to refer to both entities and concepts in KBs.

A common representation of factual knowledge is a triplet in the form of (e_{subj}, r, e_{obj}), where e_{subj} and e_{obj} refer to the subject and object entities respectively, and r refers to the relation in between. For example, ("Apple","is_a","company") represents that "Apple" is an entity of type "company"; while ("Steve Jobs","founder_of","Apple") corresponds to the fact that Steve Jobs created the Apple company. Since entities and relations are shared among triplets, a knowledge base can also be regarded as a knowledge graph in which entities are vertices and relations are edges. Each triplet thus represents a connection in the graph.

Commonly used knowledge bases in the biomedical domain include DrugBank (Wishart et al., 2018), the Comparative Toxicogenomics Database (CTD) (Davis et al., 2021), the UMLS (Bodenreider, 2004). In Table 4.1 we list several triplets from CTD to give insight into the composition of a domain KB. CTD contains interactions between entities of different types such as chemicals, genes, diseases, and phenotypes. We use CTD in subsequent experiments for our proposed KB-enhanced model on ChemProt and DrugProt, as in the two corpora we focus on extracting relations between chemicals and genes.

[1] `https://www.ncbi.nlm.nih.gov/mesh/`

Subject entity	Subject ID	Relation	Object entity	Object ID
Cellulose	D002482	increases expression	PTGS2	5743
5-methylurapidil	C057446	affects binding	ADRA1A	148
Chlorthalidone	D002752	increases activity	REN	5972
Tretinoin	D014212	decreases expression	A1BG	1
Hydroxyl Radical	D017665	increases abundance	XDH	7498
5-fluoro-2'-deoxyuridine	C576827	affects response to substance	DHFR	1719
Glimepiride	C057619	decreases activity	ABCB11	8647
Lactose	D007785	affects binding	LGALS3	3958
Labetalol	D007741	decreases reaction	PRL	5617
Sulfates	D013431	decreases activity	CA1	759

Table 4.1: A subset of triplets in CTD (Davis et al., 2021). The subject entity is of type "chemical" and the object entity is of type "gene". Chemical entities are linked to MESH[1] identifiers; gene entities are linked to NCBI (Federhen, 2011) identifiers.

4.2 Graph Embedding Methods

The success of word embedding methods has inspired researchers to investigate how to vectorize graph-structured data. A knowledge base can be represented as a graph: each entity is taken as a vertice; each relation is taken as an edge; a triplet is therefore an edge linking two vertices. Graph embedding methods are thus helpful in learning vector representations of KB entities and relations. The objective of graph embedding methods can be formulated as follows: Given a graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, compute embedding matrices $E_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d_{\mathcal{E}}}$ and $E_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times d_{\mathcal{R}}}$ respectively for every entity $e \in \mathcal{E}$ and relation $r \in \mathcal{R}$ in the corresponding KB. Entity and relation vectors are of dimensions $d_{\mathcal{E}}$ and $d_{\mathcal{R}}$ respectively. Based on the difference between learning objectives, existing graph embedding methods can be divided into two types: distance-based methods and similarity-based methods. To put it simply, distance-based methods try to minimize the distance between connected vertices in the vector space. Similarity-based methods try to put vertices that share similar neighborhoods close to each other in the vector space.

4.2.1 Distance-based Methods

The core of distance-based methods (Bordes et al., 2013; Sun et al., 2019; Lin et al., 2015; Nguyen et al., 2016) is to define a distance function for each triplet $d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj})$, where $\boldsymbol{e}_{subj}$, $\boldsymbol{e}_{obj}$, $\boldsymbol{r}$ denote vectors of e_{subj}, e_{obj} and r. Bordes et al. (2013) propose TransE, which takes relations as translations in the vector space. By defining $d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj}) = \|\boldsymbol{e}_{subj} + \boldsymbol{r} - \boldsymbol{e}_{obj}\|$, the plausibility of each triplet (e_{subj}, r, e_{obj}) is inversely proportional to the distance between $\boldsymbol{e}_{obj}$ and the vector of the subject entity plus a relation-specific translation $\boldsymbol{e}_{subj} + \boldsymbol{r}$. An SGD optimizer is then used to minimize $d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj})$ and maximize $d(\boldsymbol{e}'_{subj}, \boldsymbol{r}, \boldsymbol{e}'_{obj})$, where (e_{subj}, r, e_{obj}) exists in KB and (e'_{subj}, r, e'_{obj}) does not. Since a pair of entities may correspond to multiple relations, extensions of TransE such as those of (Lin et al., 2015; Nguyen et al., 2016) are proposed. For example, Nguyen et al. (2016) propose STransE, which enriches the distance function of TransE with two relation-specific matrices: $d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj}) = \|\boldsymbol{W}_{r,1}\boldsymbol{e}_{subj} + \boldsymbol{r} - \boldsymbol{W}_{r,2}\boldsymbol{e}_{obj}\|$. By projecting $\boldsymbol{e}_{subj}$, $\boldsymbol{e}_{obj}$ into a relation-specific subspace before translation, the expressivity of STransE is further improved compared to TransE.

In addition to translation, to better handle symmetric relations, Sun et al. (2019) proposes to take relations as rotation in the complex space. Given a triplet (e_{subj}, r, e_{obj}), the distance function is defined as:

$$d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj}) = \|\boldsymbol{e}_{subj} \circ \boldsymbol{r} - \boldsymbol{e}_{obj}\| \tag{4.1}$$

where $\circ$ is the element-wise product. To guarantee that rotation does not affect the modulus of entity embeddings, $\boldsymbol{r}$ is restrained such that $r_i \in \mathbb{C}$ and $|r_i| = 1$. Therefore, each element of $\boldsymbol{r}$ can be represented as $r_i = e^{i\theta_{r,i}}$, which represents a rotation by $\theta_{r,i}$ radians as shown in Figure 4.1. Similar to TransE, RotatE also uses a loss function that maximizes $d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj})$ for existing triplets in KB and $d(\boldsymbol{e}'_{subj}, \boldsymbol{r}, \boldsymbol{e}'_{obj})$ for non-existing triplets. The loss function of RotatE is formulated as:

$$L = -log\sigma(\gamma - d(\boldsymbol{e}_{subj}, \boldsymbol{r}, \boldsymbol{e}_{obj})) - \frac{1}{k}\sum_{i=1}^{k} log\sigma(d(\boldsymbol{e}'^{i}_{subj}, \boldsymbol{r}, \boldsymbol{e}'^{i}_{obj}) - \gamma) \tag{4.2}$$

where γ is a fixed margin, σ is the sigmoid function and $(e'_{subji}, \boldsymbol{r}, e'_{obji})$ is the i-th non-existing triplet in KB generated by negative sampling.

Due to the superior performance of RotatE on multiple benchmark knowledge graphs (Sun et al., 2019), in this thesis we use RotatE as the method to embed knowledge bases in subsequent experiments.

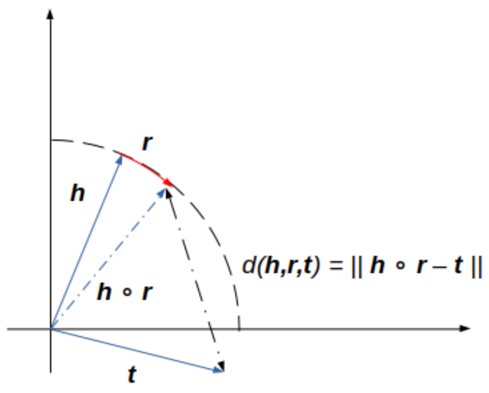


Figure 4.1: RotatE treats relations as rotation in the complex plane (source: (Sun et al., 2019).

4.2.2 Similarity-based Methods

Inspired by word embedding methods like word2vec (Mikolov et al., 2013), the intuition of similarity-based methods (Perozzi et al., 2014; Grover and Leskovec, 2016; Ribeiro et al., 2017; Hamilton et al., 2017) is that nodes with similar contexts should be close to each other in the vector space. However, unlike natural languages, nodes in graphs are not naturally ordered. To handle this problem, Perozzi et al. (2014) proposes an algorithm "random walk". Starting from a random node, the algorithm consists of sampling uniformly nodes from the neighborhood of the current node, therefore generating a sequence of nodes that are "contiguous" in the graph. Node sequences are then treated like texts as in word2vec, and the Skip-Gram architecture is used to learn node features. Subsequent studies improve the random walk algorithm. Instead of uniform sampling, Grover and Leskovec (2016) propose to perform the random walk using probability-based rules. Given a previous jump from node $t \rightarrow u$, probabilities of the next transition $u \rightarrow v$ are modulated by $d(t, v)$, which denotes the distance between t and v. Adjustable parameters are added for $d(t, v) = 0, 1, 2$, in order to bias the random walk towards capturing either local or global structures of the graph. To take structural similarities between nodes into consideration, Ribeiro et al. (2017) further proposes to build a multi-layer graph based on the original graph and perform a random walk that is allowed to traverse between layers. In the k-th layer, the transition probability between node u and v depends on their k-hop neighborhood similarity, i.e. similarity between ordered degree sequences of nodes that are k-distance away from u and v. Intuitively, if a node has many similar nodes in the current layer, it is encouraged to move to higher layers where nodes are more likely to share the same neighborhood structure with it. Therefore, structurally similar nodes have more chances to be put in adjacent positions in node sequences generated by the proposed optimized random walk.

Since most random walk-based methods only generate node sequences, no relation embedding is learned. This is one of the reasons why we choose distance-based graph embedding methods over similarity-based methods. Because distinguishing between different KB relations is important, we think that relation embeddings are very important resources.

4.3 Related Work: KB-enhanced Methods

Many attempts have been made to enhance pre-trained language models with knowledge base information. In this section, we present previous studies that can be divided into three categories:

- Distant Supervision (DS) (Subsection 4.3.1): The intuition is to improve model performance by training on large-scale data generated by aligning triplets in KB with unlabeled texts. In this section, we explain the principle of DS and also present methods that are proposed to improve the quality of DS data;

- Fusion of Graph Embeddings (Subsection 4.3.2): The intuition is to integrate entity or relation embeddings obtained using a graph embedding method into a neural model to enhance its performance;

- KB-related pre-training tasks (Subsection 4.3.3): The intuition is to pre-train language models to capture factual knowledge by introducing pre-training tasks such as entity linking or entity classification.

In the next section, we propose our KB-enhanced model that falls in the second class, i.e. we enhance BERT using pre-trained graph embeddings in the fine-tuning stage.

4.3.1 Distant Supervision

As mentioned in Section 1.1, a main challenge for domain-specific RE tasks is the limited amount of available annotated data. First introduced by Mintz et al. (2009) for relation extraction, distant

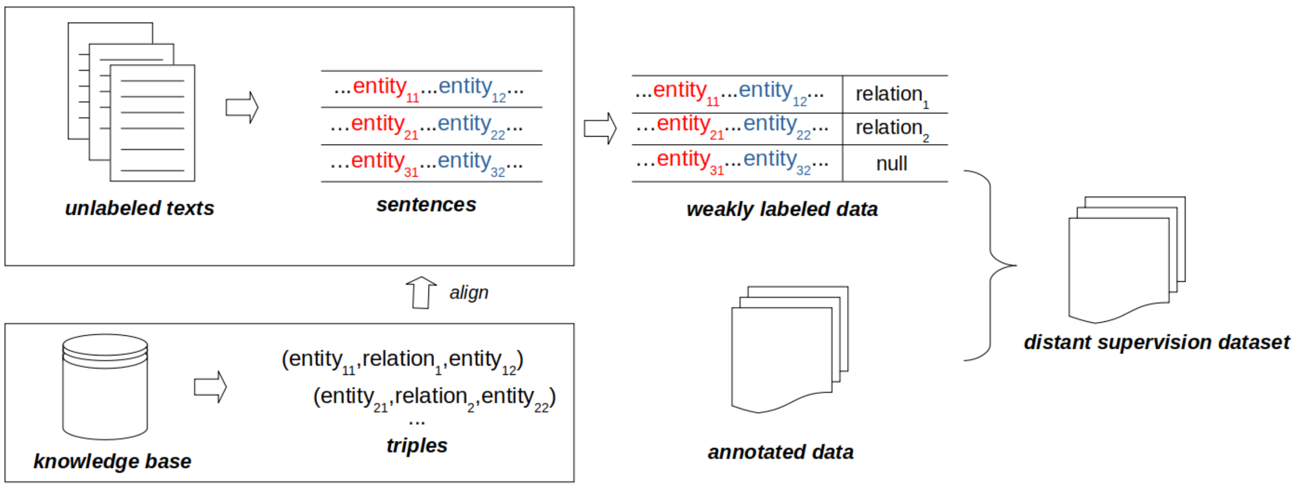

unlabeled texts
...entity$_{11}$...entity$_{12}$...
...entity$_{21}$...entity$_{22}$...
...entity$_{31}$...entity$_{32}$...
sentences
relation$_1$
relation$_2$
null
weakly labeled data
align
knowledge base
(entity$_{11}$,relation$_1$,entity$_{12}$)
(entity$_{21}$,relation$_2$,entity$_{22}$)
...
triples
annotated data
distant supervision dataset

Figure 4.2: Generation of distant supervision data.

supervision (DS) is a technique that exploits knowledge bases to handle this limitation. The intuition of distant supervision is that if a pair of entities (e_{subj},e_{obj}) has a relation r in an existing knowledge base, then any sentence that contains (e_{subj},e_{obj}) is likely to express the relation r, where e_{subj} and e_{obj} refer to the head and tail entity respectively. Based on this assumption, it is possible to collect a large amount of weakly labeled data. The diagram of DS data generation is shown in Figure 4.2. Though this method does not guarantee the accuracy of labeling, training a RE classifier with a mixture of human-annotated data and a large amount of weakly-labeled data was found to give better performance than only with a small amount of human-annotated data as shown in (Su et al., 2019; Iinuma et al., 2022; Hao et al., 2020). For example, Iinuma et al. (2022) propose first to align triplets from three biomedical knowledge bases with entity pairs from PubMed texts to create DS data, then train two BioRoberta (Lewis et al., 2020b) models respectively on DS data and the human-annotated DrugProt (Miranda et al., 2021) training set. Experiments show that the concatenation of text representations from both models gives the best performance on the DrugProt test set compared to baselines trained without or only with DS data.

However, the basic assumption of distant supervision does not always hold. Also, it is possible that only part of the triplets are actually expressed in sentences. Thus part of the knowledge base remains unused. Two common errors are frequently encountered in DS data:

- False positives: A sentence containing an entity pair may not express the relation between the two entities even though it exists in the knowledge base. For example, though there exists a relation *capital_ of* between *“Paris”* and *“France”*, this relation is not expressed by the sentence “France increases the security level in Paris.”;

- False negatives: An entity pair may be wrongly annotated as having no relation between the two entities due to incomplete knowledge bases.

To reduce wrong labels, a common way is to build a dataset annotated by DS, then hold out a split of the data as the test set and use the rest of the data to train a classifier that assigns relations to entity pairs. A benchmark dataset for DS relation extraction is New York Times (NYT) (Riedel et al., 2010), which is created by aligning Freebase (Bollacker et al., 2008) with the NYT corpus automatically. Previous studies such as (Zeng et al., 2015; Min et al., 2013; Yang et al., 2019; Ye and Ling, 2019; Takamatsu et al., 2012) focus on de-noising DS data by evaluating their model performance on benchmark datasets such as NYT. Zeng et al. (2015) proposes to apply multi-instance learning, i.e. treat sentences containing the same entity pair as a bag and only use the most correct sentence from each bag to train the DS classifier. In the inference stage, a bag is positively labeled only when at least one sentence in the bag is positively labeled by a certain relation. Ye and Ling (2019) proposes to combine attention mechanism with multi-instance learning. Intra-bag attention weights between sentences sharing the same entity pair are calculated and then used to calculate the bag representation. Furthermore, bags labeled by the same relation are regrouped into a group, and inter-bag attention weights are calculated to assign higher weights to those bags that are close to others. Min et al. (2013) focuses on handling the false negatives. They propose to add latent variables l which model true bag-level labels and treat existing labels z assigned by DS as observations, then relate l with z by conditional probabilities such that wrong labels caused by KB incompleteness have a chance to be corrected during the Expectation-Maximization (EM) training. Wang et al. (2018) proposes to abandon hard labels assigned by DS, but changes to only use prior KB information derived from graph embedding methods. Similar to TransE (Bordes et al., 2013), for an existing triplet (e_{subj},r,e_{obj}), the intuition is to replace the relation embedding of $\boldsymbol{r}$ by embeddings of sentences sharing (e_{subj}, e_{obj}). The assumption of TransE is thus upgraded: $\boldsymbol{e_{subj}} + \boldsymbol{s} \approx \boldsymbol{e_{obj}}$, where $s \in S$ and S refers to the bag of sentences that share the entity tuple

(e_{subj}, e_{obj}).

Another disadvantage of distant supervision methods is that in most cases, we require the knowledge base to contain the target relations of RE tasks. However, in practical use it may be difficult to find an available KB that contains exactly the target relations of the RE task. To solve this problem, a possible solution is to create a mapping between KB relations and target relations of RE tasks. For example, Iinuma et al. (2022) manually link Drugbank (Wishart et al., 2018) relations to target relations in DrugProt. A drawback of this method is that errors in the relation mapping may add extra noise to data generated by DS. Mapping is not always possible, or does not convey equivalence.

4.3.2 Fusion of Graph Embeddings

As shown in 4.2, it is possible to obtain vector representations of nodes and edges in a KB. Therefore, a possible solution for the integration of KB information into pre-trained LMs is to use pre-trained KB graph embeddings. In this section, we focus on methods that integrate pre-trained graph embeddings with no need of pre-training. Papaluca et al. (2022) proposes to first feed complete sentences to BERT, then average textual representations of tokens that belong to each entity to obtain entity textual representations $\boldsymbol{x}_{entity}^{BERT}$. Final entity representations are the concatenation of textual representations and pre-trained TransE entity embeddings: $\boldsymbol{x}_{entity} = [\boldsymbol{x}_{entity}^{BERT}, \boldsymbol{x}_{entity}^{graph}]$. Finally, the representations of subject and object entity are passed to a biaffine layer $\boldsymbol{\mathcal{B}}$ for relation classification, where the function of a biaffine layer is: $\boldsymbol{\mathcal{B}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T \boldsymbol{U} \boldsymbol{x}_2 + \boldsymbol{W}(\boldsymbol{x}_1 \| \boldsymbol{x}_2) + \boldsymbol{b}$. Roy and Pan (2021) tests multiple KB-enhanced methods including ClinicalBERT-EE-KGE, which consists of using first three types of KB embeddings (concept embeddings, semantic type embeddings, semantic group embeddings) related to candidate entities to predict the KB relation (from UMLS (Bodenreider, 2004)) between them, then concatenate the corresponding relation embedding and all entity embeddings with the sentence embedding obtained from ClinicalBERT (Alsentzer et al., 2019) for relation classification. Fei et al. (2021) proposes BioKGLM by post-training BERT to predict entity tokens that are masked. BERT layers are divided into three tiers, at the output of each tier pre-trained entity embeddings are infused with token representations at the masked positions. The proposed

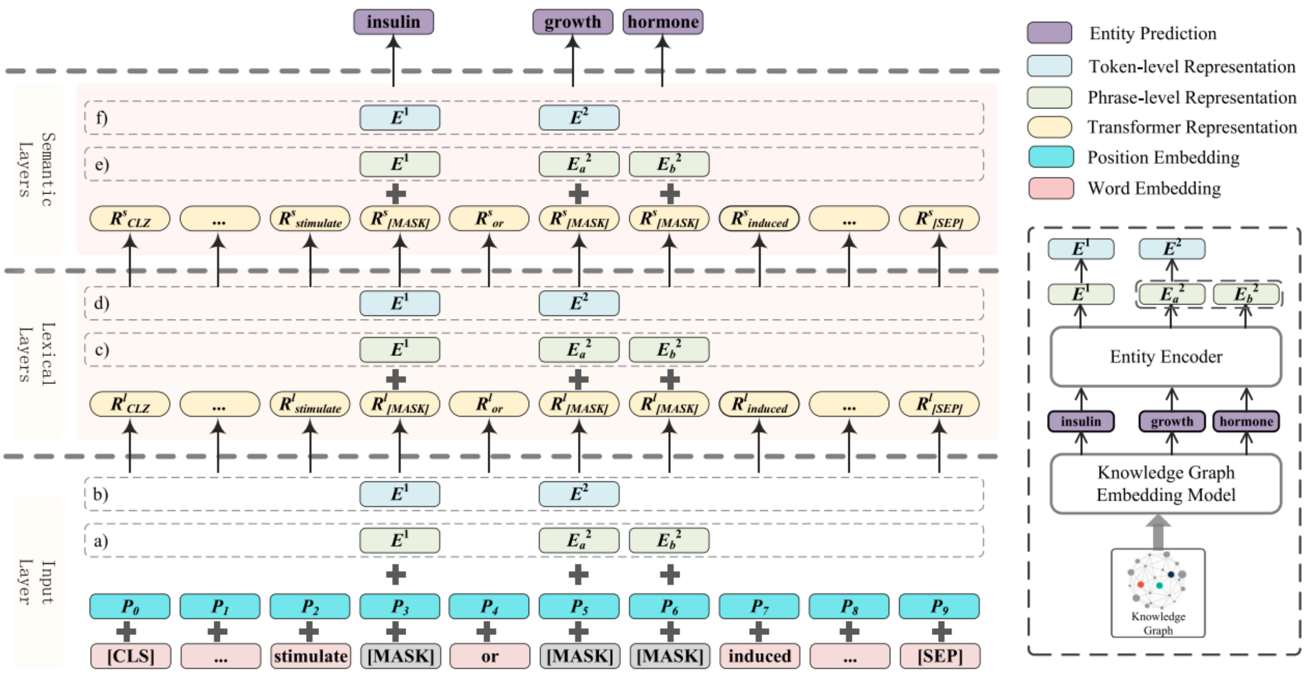


Figure 4.3: Architecture of BioKGLM (source: (Fei et al., 2021)).

architecture of BioKGLM is shown in Figure 4.3.

4.3.3 KB-related Pre-training Tasks

As mentioned in Section 2.6, BERT is pre-trained on MLM and NSP tasks. None of these pre-training tasks is designed to learn factual knowledge. Therefore, researchers propose to enhance BERT by introducing additional pre-training tasks to learn factual knowledge from KB. Peters et al. (2019) proposes KnowBERT (Knowledge-enhanced BERT), which computes first entity representations by combining pre-trained entity embeddings obtained from KB and word piece embeddings of BERT selected by entity spans, then updates entity representations using span-level attention. Besides MLM and NSP, an Entity Linking (EL) loss is added for pre-training which consists of predicting correct KB concepts for each entity. Similarly, Zhang et al. (2019b) proposes ERNIE (Enhanced Language Representation with Informative Entities), which computes the sum of pre-trained entity embeddings and corresponding word piece embeddings as entity representations, then randomly masks token-entity alignments and pre-trains BERT to predict them. Michalopoulos et al. (2021) tries to “soften” the MLM tasks by making use of Concept Unique Identifier (CUI) in UMLS.

They propose to perform multi-label classification for MLM tasks, predicting not only the masked word but also UMLS synonyms that share the same CUI, e.g. if "lung" is masked, then BERT is asked to predict "pulmonary" as well since "lung" and "pulmonary" have the same CUI. Hao et al. (2020) propose to generate pre-training data directly using triplets in KB. They propose to first generate examples in the form of "[CLS] entity$_1$ relation entity$_2$ [SEP]" (positive examples if (entity$_1$,relation,entity$_2$) exists in the KB; otherwise negative examples), then build a binary classifier based on BERT that predicts if the relation really exists between entity$_1$ and entity$_2$ in KB.

4.4 Contribution: KB-PubMedBERT

4.4.1 Hypothesis

Most previous KB-enhanced models such as (Papaluca et al., 2022; Fei et al., 2021; Zhang et al., 2019b) focus on integrating KB entity information into pre-trained language models. However, we argue that incorporating KB relation information is important too, especially for RE tasks. As mentioned at the end of 4.3.1, one challenge that we are facing with KB relations is that in most cases, relations in available KBs are different from the target relations of RE tasks. Nevertheless, relations in a domain-specific KB are likely to be relevant to those of the RE task. Therefore, finding a way to relate KB relations to task relations is crucial in building a KB-enhanced model for RE. Iinuma et al. (2022) used a manually created map to convert KB relations to target relations. We propose KB-PubMedBERT, which takes this idea further by removing the manual mapping step and having the neural model learn the mapping automatically. We hypothesize that our proposed model is capable of building a soft mapping between KB relations and task relations, and that adding these suggested, hypothetical relations on top of the PubMedBERT encoding of the text can improve the performance on RE tasks.

Compared to existing KB-enhanced methods, KB-PubMedBERT brings the following advantages:

- No identity constraint on relation types in KB: Our method applies in scenarios where KB relations are not exactly the same as target RE task relations;

- Low cost: Our method requires no additional pre-training, and uses the existing RotatE graph embedding method to integrate KB information.

4.4.2 Model Architecture

Figure 4.4 shows an overview of our model architecture. The model takes two inputs: the input sentence s, and the concept identifiers of the subject and object entities e_{subj}, e_{obj}. The concept embedding layer and the relation embedding layer are respectively initialized with pre-trained concept embeddings and relation embeddings using RotatE. After initialization, concept and relations embeddings are fine-tuned during model training. The data flow in our model is as follows. First we obtain concept embeddings for the subject and object $\boldsymbol{e_{subj}}$, $\boldsymbol{e_{obj}}$ by looking up the concept embedding layer; then we obtain the M scores for each KB relation:

$$score_i = \gamma - \|\boldsymbol{e_{subj}} \circ \boldsymbol{r_i} - \boldsymbol{e_{obj}}\| \tag{4.3}$$

where γ is a fixed margin (as presented in subsection 4.2.1), $i = 1, 2, ..., M$, and M denotes the number of KB relations. According to the definition of RotatE, the distance $\|\boldsymbol{e_{subj}} \circ \boldsymbol{r_i} - \boldsymbol{e_{obj}}\|$ should be small for existing triplets in KB, thus $score_i$ reflects the plausibility of the triplet (e_subj, r_i, e_obj). Following the convention of using the [CLS] embedding as the pooling vector to represent the sentence, denoting that vector by $\boldsymbol{h}_{[CLS]}$, we get the mixed representation:

$$\boldsymbol{h}_{concat} = [\boldsymbol{h}_{[CLS]}; \boldsymbol{h}_{score}] \tag{4.4}$$

where $[;]$ denotes vector concatenation, and h_{score} is an M-dimensional vector containing the KB relation scores. The mixed representation is then passed to a fully connected layer with softmax activation that computes the probabilities of task relations. The whole model is fine-tuned to minimize the cross entropy loss, with non-frozen PubmedBERT weights.

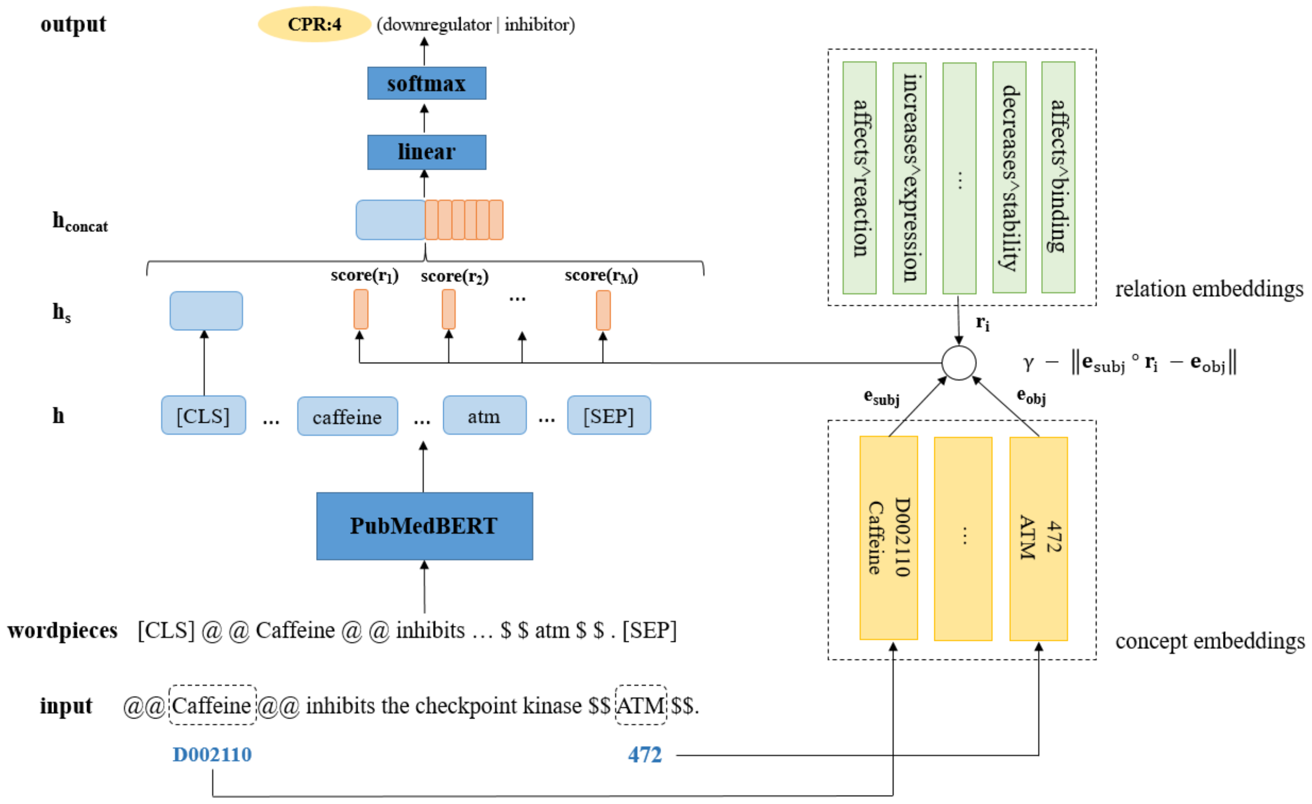


Figure 4.4: Global architecture of proposed KB-PubMedBERT model.

4.5 Experimentation

4.5.1 Datasets

We use the three benchmark datasets presented in subsection 2.8.1: BB-Rel, ChemProt and Drug-Prot. Though we do not require that the chosen KB contains exactly the same relations of RE tasks, relations in KB still need to be relevant. Besides, if a certain entity type does not exist, the performance of our KB-enhanced method can not be evaluated on examples containing entities of the corresponding type. Therefore, for BB-Rel we only keep examples containing entity types that are present in the chosen KB, i.e., microorganism and habitat entities. This subset of BB-Rel is named BB-Rel$_p$ and contains only one type of semantic relation, the lives_in relation (plus the "null" relation used for negative examples).

4.5.2 Domain Knowledge Bases

We select appropriate KBs for different corpora. For ChemProt and DrugProt, we choose CTD (Davis et al., 2021) (presented in Section 4.1), which contains normalized entities such as chemicals (normalized to MESH concepts), genes (normalized to NCBI Gene concepts) and diseases (normalized to MESH or OMIM concepts). Multiple types of relations exist in CTD such as chemical-gene interactions, chemical-phenotype interactions and gene-disease associations. Since target relations of ChemProt and DrugProt are chemical-gene interactions, we extract a subset of CTD containing only chemical-gene interactions. This subset is named by CTD_s. For BB-Rel, we choose Omnicrobe (Dérozier et al., 2023), which contains normalized entities such as microorganisms (normalized to NCBI concepts), habitat (normalized to OntoBiotope concepts), and phenotypes (normalized to OntoBiotope concepts). Three relations exist in Omnicrobe: microorganism-habitat relation "lives_in", microorganism-phenotype relation "exhibits" and microorganism-use relation "studied_for". Similarly, we extract a subset of Omnicrobe, Omnicrobe_s, that contains only "lives_in" relations. The data of Omnicrobe_s comes from three manually curated databases: CIRM[2], BacDive[3] and GenBank[4]. It is worth noticing that Omnicrobe_s is independent of the Bacteria Biotope dataset (Bossy et al., 2019). To incorporate hierarchical information between entities, we link existing microorganism and habitat entities in Omnicrobe with their parent concepts by a relation "is_a" respectively from NCBI (Federhen, 2011) and OntoBiotope (Nédellec et al., 2018), then add obtained triplets to Omnicrobe_s. Statistics of CTD_s and Omnicrobe_s are given in Table 4.2.

KB name	# unique entities	# unique relations	relation type
CTD_s	68178	134	chemical-gene interactions
Omnicrobe_s	2056587	2	lives_in, is_a

Table 4.2: Statistics of CTD_s and Omnicrobe_s.

[2]https://www6.inrae.fr/cirm_eng/The-CIRM
[3]https://bacdive.dsmz.de/
[4]https://www.ncbi.nlm.nih.gov/genbank/

4.5.3 Entity Normalization

Though KB-PubMedBERT does not require KB relations to be the same as those of RE tasks, we still need to align entities in texts to concepts. In our experiments, in cases where entity normalization is given, we use gold normalization. Otherwise, we use existing pre-trained models to perform entity normalization. It is noteworthy that entities may be normalized to concepts that do not exist in the KB: a KB may not cover all concepts that are used as labels for a pre-trained entity normalization model. On Chemprot and Drugprot, we normalize entities using BioSyn[5] (Sung et al., 2020). On BB-Rel$_p$, we directly use gold normalization on the train and validation sets since they are available. On the test set, we use a regression model from the best participant (Mao and Liu, 2019) in the BB-Norm task (Bossy et al., 2019) to normalize microorganism entities to the NCBI (Federhen, 2011) taxonomy of species, and the state-of-the-art model C-Norm (Ferré et al., 2020) to normalize habitat entities to concepts from Onto-Biotope (Nédellec et al., 2018). Table 4.3 summarizes the sources of entity normalization for each corpus.

	ChemProt & DrugProt	**BB-Rel$_p$**
train	BioSyn	*gold*
dev	BioSyn	*gold*
test	BioSyn	C-Norm, *regression*

Table 4.3: Sources of entity normalization for each corpus. "*gold*" refers to gold normalization annotations provided in BB-Norm (Bossy et al., 2019); "*regression*" refers to the regression model proposed in (Mao and Liu, 2019).

4.5.4 Baseline

We use the pre-trained PubMedBERT as a baseline, since it is the model from which our model is derived. On each task, that baseline model is fine-tuned to classify target relations. Comparing KB-PubMedBERT to the baseline directly shows whether integrating KB information helps to classify relations.

[5]We use two public pre-trained models: biosyn-sapbert-bc5cdr-chemical for chemicals; biosyn-sapbert-bc2gn for genes.

4.5.5 Implementation Details

We use the official implementation [6] of RotatE (Sun et al., 2019) to calculate KB concept and relation embeddings. Empirically, we keep the dimension of concept and relation embeddings at 200, γ at 24.0. It might occur that some entities are normalized to concepts that do not exist in the KB, in this case we randomly initialize the embeddings for these concepts. For all datasets, we use the model performance on the development set as the metric to find optimal hyperparameters. We only search the optimal learning rate from the set $(1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5})$. Table 4.4 provides the optimal learning rates. In the same way as in previous works such as (Gu et al., 2021), we use a slanted triangular scheduler for the learning rate, which consists of increasing the learning rate from 0 to a target value, then linearly decreasing to 0 at the end of training. We always use the first 10% steps for this learning rate warmup. For all models, we perform 5 runs using the same model architecture with different random seeds and using the majority voting to compute the voting score of each model. We use a single NVIDIA Tesla V100 GPU for all our experiments. It is worth noting that changing GPU cards may lead to minor or, in some cases, even a 0.1 to 0.2 difference in F1-score, i.e. affecting the reproducibility of experimental results. Our code is available at: `https://github.com/Maple177/RE_with_RotatE_graph_embs`.

	ChemProt	**DrugProt**	**BB-Rel$_p$**
PubMedBERT	-	$3e^{-5}$	$5e^{-5}$
KB-PubMedBERT	$2e^{-5}$	$2e^{-5}$	$2e^{-5}$

Table 4.4: Best learning rate for each (model, corpus) combination.

4.5.6 Results

We compare KB-PubMedBERT to the state-of-the-art (SOTA) models on each corpus:

1. For ChemProt: BioM-BERT (Alrowili and Shanker, 2021) which is a BERT model pre-trained on PubMed and PubMed Central (PMC) literature;

[6]`https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding`

	ChemProt	**DrugProt**	**BB-Rel$_p$**
PubMedBERT	77.1 ± 0.4 (77.2$^\Delta$) / 78.2	75.8 ± 0.5 / 77.2	64.4 ± 0.7 / 65.3
KB-PubMedBERT	77.8 ± 0.1 * / 79.2	77.6 ± 0.4 * / 77.9	**65.7 ± 1.0 * / 66.5**
SOTA	**80.0 / -**	**- / 79.7**	- / 64.8

Table 4.5: F1 scores on RE tasks. We report a/b where a represents the average score of 5 runs with different random initializations; b represents the majority voting score. $^\Delta$ indicates the score reported by Gu et al. (2021) on ChemProt. We report the two scores to better compare our results to the SOTA results. * indicates statistically significant improvements with $p < 0.05$ under a t-test.

2. For DrugProt: an ensemble of 10 pre-trained RoBERTa-large-PM-M3-Voc (Lewis et al., 2020a) with input enriched by chemical definitions curated from CTD (Weber et al., 2022);

3. For BB-Rel$_p$: A 12-layer Transformer model pre-trained on BooksCorpus, English Wikipedia, PubMed and PMC corpus. (Zhang et al., 2019a)

Table 4.5 summarizes the experimental results. We observe that our KB-PubMedBERT consistently outperforms the baseline model on all three corpora, which shows the effectiveness of our method of KB information injection.

Comparing our method to existing SOTA models, KB-PubMedBERT outperforms previous SOTA on BB-Rel$_p$. Though our model does not outperform SOTA models on ChemProt and DrugProt, the gap in performance might be explained by model sizes: both SOTA models have more layers and more parameters than KB-PubMedBERT.

4.6 Analysis & Discussion

4.6.1 Precision, Recall and F1-score

To further investigate the behavior of our proposed model, aside from the F1-score, we also calculate precision and recall on the validation set and show them in Table 4.6. We observe that on ChemProt, both precision and recall are improved, which naturally leads to an improvement in F1-score. On DrugProt, precision is slightly degraded while recall is improved: it seems that the integrated KB information helps to balance precision and recall. On BB-Rel$_p$, only precision is improved while recall stays unchanged, again helping to balance the two. Therefore, in all three cases, KB

	precision	recall	F1
ChemProt			
PubMedBERT	80.7	80.4	80.5
KB-PubMedBERT	81.5	81.9	81.7
DrugProt			
PubMedBERT	80.2	76.4	78.2
KB-PubMedBERT	79.5	78.8	79.2
BB-Rel$_p$			
PubMedBERT	65.9	74.8	70.0
KB-PubMedBERT	66.8	74.8	70.6

Table 4.6: Precision, Recall and F1 scores on the validation sets of the three RE tasks.

information injected by graph embeddings improves F1-score by balancing (or keeping the balance of) precision and recall.

4.6.2 Direct Links in KB

Given two concepts associated with two input entity mentions, RotatE graph embeddings make it possible to compute scores for all KB relations. However, direct links between these concepts, i.e., relations that actually exist between them in the KB, are likely to be more reliable than inferred relations that do not explicitly exist in the KB between these concepts.

One might thus hypothesize that the existence in the KB of direct links between these concepts might be the main source of information that helps improve predictions. We thus test whether the improved cases, i.e., those predicted wrongly by PubMedBERT but correctly by KB-PubMedBERT, are linked to a higher proportion of direct links.

For this purpose, we divide examples in the validation sets of RE tasks into four categories:

1. easy: both PubMedBERT and KB-PubMedBERT give correct predictions;
2. hard: both models give wrong predictions;
3. improved: PubMedBERT gives wrong predictions while KB-PubMedBERT gives correct predictions;
4. degraded: PubMedBERT gives correct predictions while KB-PubMedBERT gives wrong predictions.

	ChemProt	**DrugProt**	**BB-Rel$_p$**
% easy	15.7	27.0	13.2
% hard	17.0	**31.2**	15.7
% improved	**22.8**	30.1	11.9
% degraded	14.2	28.5	13.2

Table 4.7: Percentages of examples for which the subject and object entities are directly linked in the KB. Bold values are significantly different from the overall percentage of the corresponding corpus with $|Z| > 2$ under a Z-test.

We then calculate the percentage of examples with directly linked subject and object entities in each of the four groups in each corpus and test whether the percentage in each group is significantly different compared to the overall percentage in the corresponding corpus. Table 4.7 displays these percentages. The results reject the above hypothesis: although in ChemProt, the "improved" group contains significantly more examples with directly linked subject and object entities, in DrugProt this difference is not significant, while in BB-Rel$_p$ the "improved" group contains even fewer examples with direct links. Besides, in DrugProt, the "hard" group has a significantly higher percentage.

In conclusion, the observed improvement in RE performance is not explained by the percentage of examples with direct links that exist in the KB. It seems that non-direct links play a role in the improvement of KB-PubMedBERT, and more factors are involved here, but we need further investigation to elucidate.

4.6.3 Ablation Study

To verify how much the graph embedding module in our model is able to detect the target RE relations by itself, we conduct experiments in which we completely remove PubMedBERT from our model. This means that we only use the (subject, object) entity pair to predict the interaction type via RotatE graph embeddings. We compare the resulting relation classifier to a naive model that always predicts the most frequent non-null relation. Table 4.8 shows the results on the test set of the three corpora.

We observe that even with no context, our model significantly outperforms the naive model on ChemProt and DrugProt. This shows that the scores of fine-grained KB relations obtained from RotatE graph embeddings are helpful. On BB-Rel$_p$, the naive model can easily outperform our

	ChemProt	**DrugProt**	**BB-Rel$_p$**
KB-Pred	23.8 ± 1.6	19.5 ± 1.0	26.6 ± 0.3
naive	17.3	12.3	38.3

Table 4.8: KB-Pred denotes our proposed model without the PubMedBERT embedding module, thus using only KB-derived information in its prediction. We report the average score of 5 runs. *naive* refers to a model that always predicts the most frequent non-null relation.

model because there is only one non-null relation.

4.6.4 Case Study

To get an insight into the behavior of our proposed model, we manually examine improved and degraded examples, where the definition of improved and degraded examples remains the same as in the previous subsection (Subsection 4.6.2). Selected improved and degraded examples are respectively given in Table 4.9 and Table 4.10. We observe that PubMedBERT predicts "no_relation" for most of the examples improved by KB information, i.e. false negatives originally predicted by PubMedBERT can be corrected by KB-PubMedBERT. Besides, the KB relation between arguments of an improved example is found to be close to the relation of the RE task as shown in Table 4.9. For example, KB relations "decreases^activity" and "decreases^reaction" are found to help predict the relation "inhibitor"; "increases^expression" helps predict the relation "activator". It demonstrates the effectiveness of introducing KB information. However, KB relations are not always useful. We find that in some cases, KB relations can be harmful:

- False positives: KB-PubMedBERT can be biased to predict a relation that is close to a KB relation. In some cases, KB-PubMedBERT favors KB information over textual information, and predicts a non-null relation even though the relation is not expressed in texts (4-th and 5-th examples in Table 4.10);
- Multiple KB relations: There may exist multiple KB relations for a given pair of entities. These KB relations seem to confuse KB-PubMedBERT (such as the 1st, 2nd and 3rd examples in Table 4.10) if they are quite different from each other. For example, in the third example in Table 4.10, "decreases^reaction" and "increases^activity" describe two opposite chemical-

gene interactions. The existence of "increases^activity" may prevent KB-PubMedBERT from predicting "downregulator | inhibitor".

- Lack of precision: For the last example in Table 4.10, KB-PubMedBERT predicts "indirect-downregulator" while the true relation is "inhibitor". In ChemProt, the two relations are regrouped into a single class, while in DrugProt they are taken as two separate classes. The prediction of KB-PubMedBERT may be biased to "indirect-downregulator" due to the KB suggestion "decreases^reaction", which is not false but is not accurate enough.

4.7 Conclusion

In this chapter, we review the composition of knowledge bases and existing graph embedding methods that can be used to vectorize knowledge base information. Then we focus on previous KB-enhanced methods that can be divided into three classes: distant supervision methods, which consist of extracting large-scale training data from unlabeled data; graph embedding-based methods explicitly integrate knowledge graph embeddings into neural architectures; methods belonging to the third class make language models capture knowledge base information implicitly by adding a pre-training objective.

Inspired by the graph embedding method RotatE, we propose our KB-enhanced model KB-PubMedBERT. Since RotatE provides a way to compute the plausibility of any triplet (e_{subj}, r, e_{obj}), we search to exploit RotatE graph embeddings to compute scores of each KB relation given a candidate entity pair, then integrate the scores into PubMedBERT by concatenation. Our experimental results show that KB-PubMedBERT consistently outperforms PubMedBERT, and even outperforms the state-of-the-art model on BB-Rel$_p$. The ablation study and case study show the effectiveness of injected KB information. The main advantage of our proposed method is that we do not require the selected knowledge base to contain the same relations as those of the RE task, therefore the applicability is improved compared to previous KB-enhanced models. Besides, our proposed method requires no pre-training or extra training data, and it can be easily transferred to other domain-specific corpora.

sentence	with KB?	predictions	KB relations
Imatinib inhibits RET-mediated MTC cell growth affecting RET protein levels in vitro in a dose-dependent manner.	✗	no_relation	
	✓	**downregulator | inhibitor**	decreases^phosphorylation
Binding of cGMP to GAF-A increases cNPK phosphorylation of PDE5 and improves catalytic site affinity for cGMP or inhibitors.	✗	no_relation	
	✓	**upregulator | activator**	increases^abundance
The CYP3A4 activity could be induced 2-fold by rifampicin, whereas CYP2C9 activity remained equally high.	✗	no_relation	
	✓	**activator**	increases^expression
Binding and transactivation assays were used to compare affinities and transcriptional activities of adapalene and tretinoin for the nuclear transcription factors, retinoic acid receptors (RARs).	✗	no_relation	
	✓	**direct-regulator**	decreases^expression
we investigated the role of PPAR-alpha in gemfibrozil-mediated inhibition of iNOS.	✗	no_relation	
	✓	**inhibitor**	affects^cotreatment, decreases^reaction
Selective inhibition of PDE5 is a rational therapeutic approach in ED, as proved by the clinical success of sildenafil.	✗	no_relation	
	✓	**inhibitor**	decreases^activity
The molecular mechanism studies suggested that neoechinulin A may block the phosphorylation of mitogen-activated protein kinase (MAPK) molecule p38, apoptosis signal-regulating kinase 1 (ASK-1) and nuclear translocation of nuclear factor-kB (NF-kB) p65 and p50 subunits.	✗	no_relation	
	✓	**indirect-downregulator**	affects^localization, decreases^reaction
Among the possible transporters involved in the uptake of Cd(2+) and Mn(2+), the expression of ZIP8 (Zrt-, Irt-related protein 8), encoded by Slc39a8, showed a marked suppression in both RBL-Cdr and RBL-Mnr cells.	✗	no_relation	
	✓	**substrate |product_of**	increases^import

Table 4.9: Case study: improved examples on the validation set of ChemProt and DrugProt. Red words refer to the subject entity (chemical) and blue words refer to the object entity (gene). The column "predictions" contains the relations predicted by the corresponding model, where a bold relation refers to a correct prediction. The last column contains relations found in the KB given the corresponding pair of entities.

sentence	with KB?	predictions	KB relations
The following alpha(2)-adrenoceptor antagonists were applied: BRL44408 (alpha(2A)-selective), ARC239 (alpha(2B)- and alpha(2C)-selective).	✗	**no_relation**	
	✓	antagonist	decreases^activity, affects^binding
Arsenic inhibits autophagic flux activating the Nrf2-Keap1 pathway in a p62-dependent manner.	✗	**upregulator |activator**	
	✓	downregulator |inhibitor	decreases^expression, increases^abundance
Known VR1 antagonists (BCTC, thio-BCTC and capsazepine) were also able to block the response of TRPM8 to menthol (IC(50): 0.8+/-1.0, 3.5+/-1.1 and 18+/-1.1 microM, respectively).	✗	**downregulator |inhibitor**	
	✓	antagonist	decreases^reaction, increases^activity
Ponatinib (AP24534) is a multikinase inhibitor with in vitro and clinical activity in tyrosine kinase inhibitor (TKI)-resistant chronic myeloid leukemia, irrespective of BCR-ABL KD mutation.	✗	**no_relation**	
	✓	downregulator |inhibitor	decreases^activity
These findings suggest that troglitazone inhibits antigen-induced LT production in the IgE-sensitized RBL-2H3 cells and A23187-stimulated rat peritoneal neutrophils by direct inhibition of 5-LOX activity.	✗	**no_relation**	
	✓	activator	increases^activity
Finally, PLA2 inhibitor methyl arachidonyl fluorophosphonate blocked the PUFA effects on COX-2 induction, promoter activity and arachidonic acid mobilization suggesting involvement of AA metabolites in PPAR activation.	✗	**inhibitor**	
	✓	indirect-downregulator	decreases^reaction

Table 4.10: Case study: degraded examples on the validation set of ChemProt and DrugProt. Red words refer to the subject entity (chemical) and blue words refer to the object entity (gene). The column "predictions" contains the relations predicted by the corresponding model, where a bold relation refers to a correct prediction. The last column contains relations found in the KB given the corresponding pair of entities.

We conducted case studies respectively for improved examples (on which KB-PubMedBERT corrects erroneous predictions made by PubMedBERT) and degraded examples (on which KB-PubMedBERT spoils originally correct predictions made by PubMedBERT). The case study on improved examples demonstrates concretely how injected KB information helps; while the case study reveals the limitations of KB-PubMedBERT and opens up perspectives for future work. As presented in Subsection 4.6.4, the analysis of degraded examples shows that KB-PubMedBERT may focus more on suggestions from the KB and make erroneous predictions, ignoring textual information. This is likely due to the fact that in KB-PubMedBERT, we simply concatenate vector representations summarizing KB information and textual information, no learnable weights are added to control the quantity of the introduced KB information. Possible solutions would be to introduce a neural layer such as Highway Gate (Srivastava et al., 2015) that specializes in infusing two vector representations, or compute attention coefficients between token embeddings and most plausible KB relation embeddings. Secondly, the case study on degraded examples also shows the negative impact of having multiple KB relations between a pair of entities. In these cases, KB-PubMedBERT seems to be confused by KB suggestions, especially when KB relations are quite different from each other, e.g. "increases^reaction" and "decreases^reaction" may both exist between a chemical and a gene in a KB. This impact may be partly explained by the choice of the graph embedding method. Since RotatE learns a relation-specific rotation in the complex space, it does not handle well the cases where multiple KB relations exist between the same entity pair. However, changing graph embedding methods cannot handle the case of having KB relations that express opposite interactions. In this case, KB suggestions are required to change with contexts and can not be fixed. For now, we have no clue how to solve this problem and we leave it for future work. Lastly, we have not yet exhaustively investigated the effectiveness of our model architecture since we have only tested PubMedBERT as the base language model and RotatE as the graph embedding method. More extensive experiments using combinations of different BERT variants and graph embedding methods would give more insight into the effectiveness of the method. It is also worth testing more biomedical RE corpora such as i2b2 (Uzuner et al., 2011) and DDI (Herrero-Zazo et al., 2013).

Chapter 5

Conclusions and perspectives

In this chapter, we first summarize what has been studied in this thesis, then present perspectives for future work.

5.1 Conclusions

The invention of pre-trained large language models (LLM) like BERT improves the performance of multiple NLP tasks including relation extraction (RE). Domain-specific BERT variants further extend this improvement to texts of specific domains such as biomedical texts. However, since the pre-training of BERT does not involve any syntactic or knowledge base (KB) information, our objective in this thesis was to investigate if introducing this external information may improve the performance of BERT on biomedical relation extraction tasks. We chose PubMedBERT, a biomedical BERT variant, as the cornerstone of our proposed methods and chose three biomedical RE corpora as benchmarks: BB-Rel, ChemProt and DrugProt. We then proposed several BERT-based models enhanced by either syntactic (Section 3.3) or KB (Section 4.4) information:

- CE-PubMedBERT: word piece embeddings that correspond to a chunk are summed to obtain chunk embeddings; then chunk embeddings are passed to extra attention layers after PubMedBERT;

- CT-PubMedBERT: constituency trees are linearized into sequences and then passed to Pub-

MedBERT;

- MTS-PubMedBERT: the RE task is jointly trained with two tasks that consist of recovering structural properties of dependency trees;

- KB-PubMedBERT: the textual representation obtained from PubMedBERT is concatenated with a vector containing plausibility scores of KB relations obtained by RotatE, a graph embedding method.

Two baseline models were set in experiments: PubMedBERT and PubMedBERT-extra (Subsection 3.4.2). We also tested an existing syntax-enhanced model Late-Fusion that has not yet been evaluated on biomedical corpora. Experimental results on syntax-enhanced models show that CE-PubMedBERT and Late-Fusion outperform PubMedBERT on two of the datasets, BB-Rel and DrugProt. However, further stratified results demonstrate that the performance of CE-PubMedBERT and Late-Fusion are highly correlated to PubMedBERT-extra, leading us to conclude that observed improvements of CE-PubMedBERT and Late-Fusion are not due to integrated syntactic information but to extra layers added to the base model. The other two syntax-enhanced models, CT-PubMedBERT and MTS-PubMedBERT, do not work as expected, possibly due to training difficulties. Therefore, we conclude that syntactic information does not help improve biomedical RE performance. This conclusion is not definitive, and further experiments with new neural architectures and training strategies are needed to clarify whether syntactic information may be useful for biomedical RE.

KB-PubMedBERT, the KB-enhanced model that we proposed, succeeded in improving the biomedical RE performance on the three corpora that we selected: BB-Rel, ChemProt, DrugProt. In general, experimental results show that plausibility scores of KB relations help PubMedBERT better classify relations. Our case study confirms the effectiveness of integrated KB information, though it also shows certain negative impacts of exploiting KB suggestions.

5.2 Perspectives

We have mentioned in this thesis that possible improvements can be made by modifying model architectures or training strategies. We summarize them in this section along with other future works.

5.2.1 Improvements to Proposed Methods

5.2.1.1 MTS-PubMedBERT

The degradation of MTS-PubMedBERT may be explained in two ways: (1) since predicting syntactic pairwise distances and depths is treated as a classification task, mapping from distance or depth values to classes may have an impact on the performance of MTS-PubMedBERT. We conducted experiments using a relatively strict mapping, i.e., each distance or depth value corresponds to a class (except for values larger than a certain threshold is mapped to a class). A clue of investigation is to use a less strict mapping. (2) We used the same learning rate for three tasks of MTS-PubMedBERT during fine-tuning, which may not be optimal for the neural network to converge. A second interesting enhancement would consist of trying different learning rates respectively for the three tasks.

5.2.1.2 KB-PubMedBERT

As presented in Section 4.7, subsequent studies following KB-PubMedBERT may consist of adding a neural layer such as HighWay Gate (Srivastava et al., 2015) (used in Late-Fusion) that specializes in infusing the textual representation and the KB representation. Using the attention mechanism over token embeddings and KB relation embeddings is another option: instead of concatenating the vector containing plausibility scores of KB relations, tokens and most plausible KB relations would attend to each other, and both token embeddings and relation embeddings would be updated.

A novelty of KB-PubMedBERT is that the selected KB is not required to contain the exact same relations as the RE task. Experimental results show that our method did improve the RE performance using a KB containing relations that are different from those of the RE task, but KB relations and the target relations are still similar (they are the same in the case of BB-Rel). A further investigation direction is to identify to what extent KB relations can be different from those of the RE task and at the same time provide gains for KB-PubMedBERT over PubMedBERT. A possible objective for this study is to find an indicator that measures the “usefulness” of a KB with respect to a RE task. For example, due to the non-existing entity problem, we can use the percentage of entities that exist in the KB as an indicator. We believe that this study would be helpful for KB selection and may further increase the applicability of KB-PubMedBERT.

5.2.2 Improvements to Preprocessing

We have mentioned in Subsection 3.5.1 several problems in the dependency parses for BB-Rel: double-sentence problem; periods; abbreviations. These problems can be mitigated by pre-processing. For example, to handle the double-sentence problem, we can input the two sentences separately and combine the two dependency trees by linking their syntactic roots, or linking coreferences (if exist) to their antecedents. For problematic abbreviations, we can create a mapping to replace abbreviations with full names of entities. As mentioned in Subsection 4.5.3, entities may be normalized to concepts that do not exist in the KB. Since in KB-PubMedBERT, we create random embeddings for these entities, we believe that the performance of KB-PubMedBERT can be further improved if we can decrease the number of non-existing entities. This can be achieved by pre-processing: for a given entity, pre-trained entity normalization models usually predict top-K (the value of K varies with different models) concepts, but in our experiments, we always chose the most probable concept as the prediction. In the case that the most probable concept does not exist in the KB, using instead less probable concepts that exist in the KB would mitigate the problem of non-existing entities. It would be a tradeoff between the normalization accuracy and the number of non-existing entities.

5.2.3 Resource Choice

A limitation of our work is that we use PubMedBERT as the base model through our experiments and we test only three corpora, therefore our observation and conclusion may not be representative enough and may be biased to the choice of corpus and base model. Further systematic investigation is needed to obtain a global insight. Future work consists of testing different combinations of resources: additional biomedical RE corpora such as i2b2 (Uzuner et al., 2011) and DDI (Herrero-Zazo et al., 2013); additional domain-specific BERT variants such as BioBERT (Lee et al., 2020), SciBERT (Beltagy et al., 2019) and BioLinkBERT (Yasunaga et al., 2022). For KB-PubMedBERT, extensive experiments may consist of testing different combinations of (corpus, base model, graph embedding model): additional graph embedding models exist such as TransE and STranE. Since in specific domains, knowledge bases are often small-sized and cover only part of the domain knowledge, it may also be interesting to try merging different domain-specific KBs to form a more extensive KB. This may also help mitigate the problem of non-existing entities.

5.2.4 Model Deployment

Since KB-PubMedBERT gives promising results, we are considering deploying it to production and integrating it into AlvisNLP (Ba and Bossy, 2016), an NLP pipeline developed and maintained by the Bibliome group of INRAE that consists of multiple modules such as Named Entity Recognition and Entity Normalization. Once KB-PubMedBERT is integrated into AlvisNLP, it can be used by domain-specific applications such as Omnicrobe[1]. Omnicrobe gathers comprehensive information on microbial biodiversity (habitats, phenotypes and usages of microorganisms) as automatically extracted from text sources of publications and databases. Because our current implementation of KB-PubMedBERT is only for experimental usage, extra work is needed for deployment including wrapping our model in the pipeline and possible optimization of codes to reduce training and inference time.

[1]`https://omnicrobe.migale.inrae.fr/`

Publications During the Thesis

Tang, A., Nédellec, C., Zweigenbaum, P., Deléger, L., & Bossy, R. 2021. Global alignment for relation extraction in Microbiology. In *Junior Conference on Data Science and Engineering.*

Tang, A., Deleger, L., Bossy, R., Zweigenbaum, P., & Nédellec, C. 2022. Do syntactic trees enhance Bidirectional Encoder Representations from Transformers (BERT) models for chemical–drug relation extraction?. *Database*, 2022, baac070.

Tang, A., Bossy, R., Deléger, L., Nédellec, C., & Zweigenbaum, P. 2023. Exploitation de plongements de graphes pour l'extraction de relations biomédicales. In *18e Conférence en Recherche d'Information et Applications-16e Rencontres Jeunes Chercheurs en RI-30e Conférence sur le Traitement Automatique des Langues Naturelles (TALN)-25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues* (pp. 298-310). ATALA.

Bibliography

S. Alrowili and V. Shanker. BioM-transformers: Building large biomedical language models with BERT, ALBERT and ELECTRA. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 221–227, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.bionlp-1.24. URL `https://aclanthology.org/2021.bionlp-1.24`.

E. Alsentzer, J. Murphy, W. Boag, W.-H. Weng, D. Jindi, T. Naumann, and M. McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-1909. URL `https://aclanthology.org/W19-1909`.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

M. Ba and R. Bossy. Interoperability of corpus processing workflow engines: the case of alvisnlp/ml in openminted. *Cross-Platform Text Mining and Natural Language Processing Interoperability*, pages 15–18, 2016.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

J. Bai, Y. Wang, Y. Chen, Y. Yang, J. Bai, J. Yu, and Y. Tong. Syntax-BERT: Improving pre-trained transformers with syntax trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020, Online, Apr.

2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.262. URL `https://aclanthology.org/2021.eacl-main.262`.

I. Beltagy, K. Lo, and A. Cohan. Scibert: Pretrained language model for scientific text. In *EMNLP*, 2019.

Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In *Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web*, pages 91–103. Scientific American, 2023.

A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100, 1995.

O. Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL `https://doi.org/10.1145/1376616.1376746`.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

R. Bossy, L. Deléger, E. Chaix, M. Ba, and C. Nédellec. Bacteria biotope at BioNLP open shared tasks 2019. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 121–131, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5719. URL `https://aclanthology.org/D19-5719`.

X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada, Oct. 2005. Association for Computational Linguistics. URL `https://aclanthology.org/H05-1091`.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL `https://aclanthology.org/W19-4828`.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

D. Dai, X. Xiao, Y. Lyu, S. Dou, Q. She, and H. Wang. Joint extraction of entities and overlapping relations using position-attentive sequence labeling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6300–6308, 2019.

A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, J. Wiegers, T. C. Wiegers, and C. J. Mattingly. Comparative toxicogenomics database (ctd): update 2021. *Nucleic acids research*, 49(D1):D1138–D1143, 2021.

M.-C. de Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf`.

M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for Machine Learning*, pages 155–157. Cambridge University Press, 2020.

E. Delavenay and K. M. Delavenay. An introduction to machine translation. *Modern Language Review*, 57:73, 1962. URL `https://api.semanticscholar.org/CorpusID:161770638`.

S. Dérozier, R. Bossy, L. Deléger, M. Ba, E. Chaix, O. Harlé, V. Loux, H. Falentin, and C. Nédellec. Omnicrobe, an open-access database of microbial habitats and phenotypes using a comprehensive text mining and data fusion approach. *PloS one*, 18(1):e0272473, 2023.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1): 269–271, 1959.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

S. Federhen. The NCBI Taxonomy database. *Nucleic Acids Research*, 40(D1):D136–D143, 12 2011. ISSN 0305-1048. doi: 10.1093/nar/gkr1178. URL `https://doi.org/10.1093/nar/gkr1178`.

H. Fei, Y. Ren, Y. Zhang, D. Ji, and X. Liang. Enriching contextualized language model from knowledge graph for biomedical information extraction. *Briefings in bioinformatics*, 22(3):bbaa110, 2021.

A. Ferré, L. Deléger, R. Bossy, P. Zweigenbaum, and C. Nédellec. C-Norm: a neural approach to few-shot entity normalization. *BMC bioinformatics*, 21(23):579, 2020. doi: 10.1186/s12859-020-03886-8. URL `https://doi.org/10.1186/s12859-020-03886-8`.

T.-J. Fu, P.-H. Li, and W.-Y. Ma. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1136. URL `https://aclanthology.org/P19-1136`.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, chapter Representation Learning, pages 521–522. MIT Press, Cambridge, MA, USA, 2016a. URL `http://www.deeplearningbook.org`.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, chapter Representation Learning, pages 321–334. MIT Press, Cambridge, MA, USA, 2016b. URL `http://www.deeplearningbook.org`.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, chapter Representation Learning, pages 517–518. MIT Press, Cambridge, MA, USA, 2016c. URL `http://www.deeplearningbook.org`.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23, 2021.

Z. Guo, G. Nan, W. Lu, and S. B. Cohen. Learning latent forests for medical relation extraction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

B. Hao, H. Zhu, and I. C. Paschalidis. Enhancing clinical bert embedding using a biomedical knowledge base. In *28th International Conference on Computational Linguistics (COLING 2020)*, 2020.

Z. S. Harris. Distributional structure. *<i>WORD</i>*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL `https://doi.org/10.1080/00437956.1954.11659520`.

D. Haussler et al. Convolution kernels on discrete structures. Technical report, Citeseer, 1999.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL `http://arxiv.org/abs/1606.08415`.

M. Herrero-Zazo, I. Segura-Bedmar, P. Martínez, and T. Declerck. The ddi corpus: An annotated corpus with pharmacological substances and drug–drug interactions. *Journal of Biomedical Informatics*, 46(5):914–920, 2013. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2013.07.011. URL `https://www.sciencedirect.com/science/article/pii/S1532046413001123`.

J. Hewitt and C. D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1419. URL `https://aclanthology.org/N19-1419`.

J. Hewitt, K. Ethayarajh, P. Liang, and C. Manning. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.122. URL `https://aclanthology.org/2021.emnlp-main.122`.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

K. Hornik, M. B. Stinchcombe, and H. L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

N. Iinuma, M. Miwa, and Y. Sasaki. Improving supervised drug-protein relation extraction with distantly supervised models. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, pages 161–170, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bionlp-1.16. URL `https://aclanthology.org/2022.bionlp-1.16`.

G. Jawahar, B. Sagot, and D. Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL `https://aclanthology.org/P19-1356`.

F. Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.

J. Jiang and C. Zhai. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, Rochester, New York, Apr. 2007. Association for Computational Linguistics. URL `https://aclanthology.org/N07-1015`.

M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

N. Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL interactive poster and demonstration sessions*, pages 178–181, 2004.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

N. Kitaev and D. Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1249. URL `https://aclanthology.org/P18-1249`.

M. Krallinger, O. Rabal, S. A. Akhondi, M. P. Pérez, J. Santamaría, G. P. Rodríguez, G. Tsatsaronis, A. Intxaurrondo, J. A. López, U. Nandal, et al. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, volume 1, pages 141–146, 2017.

J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

P. Lewis, M. Ott, J. Du, and V. Stoyanov. Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-the-art. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157, Online, Nov. 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.clinicalnlp-1.17. URL `https://aclanthology.org/2020.clinicalnlp-1.17`.

P. Lewis, M. Ott, J. Du, and V. Stoyanov. Pretrained language models for biomedical and clinical tasks: understanding and extending the state-of-the-art. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157, 2020b.

S. Li, X. Li, L. Shang, Z. Dong, C. Sun, B. Liu, Z. Ji, X. Jiang, and Q. Liu. How pre-trained language models capture factual knowledge? a causal-inspired analysis. In *Findings of the*

Association for Computational Linguistics: ACL 2022, pages 1720–1732, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.136. URL `https://aclanthology.org/2022.findings-acl.136`.

X. Li, Z. Jie, J. Feng, C. Liu, and S. Yan. Learning with rethinking: Recurrently improving convolutional neural networks through feedback. *Pattern Recognition*, 79:183–194, 2018.

T. Limisiewicz, D. Mareček, and R. Rosa. Universal Dependencies According to BERT: Both More Specific and More General. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2710–2722, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.245. URL `https://aclanthology.org/2020.findings-emnlp.245`.

Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2047. URL `https://aclanthology.org/P15-2047`.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

Z. Luo. Have attention heads in bert learned constituency grammar? *EACL 2021*, page 8, 2021.

C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

J. Mao and W. Liu. Integration of deep learning and traditional machine learning for knowledge extraction from biomedical literature. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 168–173, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5724. URL `https://aclanthology.org/D19-5724`.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL `https://aclanthology.org/J93-2004`.

G. Michalopoulos, Y. Wang, H. Kaka, H. Chen, and A. Wong. UmlsBERT: Clinical domain knowledge augmentation of contextual embeddings using the Unified Medical Language System Metathesaurus. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1744–1753, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.139. URL `https://aclanthology.org/2021.naacl-main.139`.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `https://aclanthology.org/N13-1095`.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, Aug. 2009. Association for Computational Linguistics. URL `https://aclanthology.org/P09-1113`.

A. Miranda, F. Mehryary, J. Luoma, S. Pyysalo, A. Valencia, and M. Krallinger. Overview of drugprot biocreative vii track: quality evaluation and large scale text mining of drug-gene/protein relations. In *Proceedings of the seventh BioCreative challenge evaluation workshop*, pages 11–21, 2021.

M. Miwa and M. Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1105. URL `https://aclanthology.org/P16-1105`.

C. Nédellec, R. Bossy, E. Chaix, and L. Deléger. Text-mining and ontologies: new approaches to knowledge discovery of microbial diversity. *arXiv preprint arXiv:1805.04107*, 2018.

D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1054. URL `https://aclanthology.org/N16-1054`.

T. H. Nguyen and R. Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.3115/v1/W15-1506. URL `https://aclanthology.org/W15-1506`.

A. Papaluca, D. Krefl, H. Suominen, and A. Lenskiy. Pretrained knowledge base embeddings for improved sentential relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 373–382, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-srw.29. URL https://aclanthology.org/2022.acl-srw.29.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Curran Associates Inc., Red Hook, NY, USA, 2019a.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1005. URL `https://aclanthology.org/D19-1005`.

F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL `https://aclanthology.org/D19-1250`.

G. Puccetti, A. Miaschi, and F. Dell'Orletta. How do bert embeddings organize linguistic knowledge? In *Proceedings of deep learning inside out (DeeLIO): the 2nd workshop on knowledge extraction and integration for deep learning architectures*, pages 48–57, 2021.

P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020. URL `https://nlp.stanford.edu/pubs/qi2020stanza.pdf`.

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

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

E. Reif, A. Yuan, M. Wattenberg, F. B. Viegas, A. Coenen, A. Pearce, and B. Kim. Visualizing and measuring the geometry of bert. *Advances in Neural Information Processing Systems*, 32, 2019.

L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, page 148–163, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642159389.

B. Rink and S. Harabagiu. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 256–259, 2010.

A. Roy and S. Pan. Incorporating medical knowledge in BERT for clinical relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5357–5366, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.435. URL `https://aclanthology.org/2021.emnlp-main.435`.

D. Sachan, Y. Zhang, P. Qi, and W. L. Hamilton. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.228. URL `https://aclanthology.org/2021.eacl-main.228`.

M. Schuster and K. Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

P. Su, G. Li, C. Wu, and K. Vijay-Shanker. Using distant supervision to augment manually annotated data for relation extraction. *PloS one*, 14(7):e0216913, 2019.

Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

M. Sung, H. Jeon, J. Lee, and J. Kang. Biomedical entity representations with synonym marginalization. In *ACL*, 2020.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–729, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `https://aclanthology.org/P12-1076`.

A. Tang, C. Nédellec, P. Zweigenbaum, L. Deléger, and R. Bossy. Global alignment for relation extraction in microbiology. In *Junior Conference on Data Science and Engineering*, 2021.

A. Tang, L. Deléger, R. Bossy, P. Zweigenbaum, and C. Nédellec. Do syntactic trees enhance Bidirectional Encoder Representations from Transformers (BERT) models for chemical–drug relation extraction? *Database*, 2022:baac070, 08 2022. ISSN 1758-0463. doi: 10.1093/database/baac070. URL `https://doi.org/10.1093/database/baac070`.

A. Tang, R. Bossy, L. Deléger, C. Nédellec, and P. Zweigenbaum. Exploitation de plongements de graphes pour l'extraction de relations biomédicales. In C. Servan and A. Vilnat, editors, *18e Conférence en Recherche d'Information et Applications – 16e Rencontres Jeunes Chercheurs en RI – 30e Conférence sur le Traitement Automatique des Langues Naturelles – 25e Rencontre* *des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*, pages 298–310, Paris, France, 2023. ATALA. URL https://hal.science/hal-04130138.

Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5): 552–556, 2011.

D. Valsamou. *Extraction d'Information pour les réseaux de régulation de la graine chez Arabidopsis Thaliana.* PhD thesis, 2017. URL http://www.theses.fr/2017SACLS027. Thèse de doctorat dirigée par Zweigenbaum, Pierre et Nédellec, Claire Informatique Université Paris-Saclay (ComUE) 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

K. Verspoor, K. B. Cohen, A. Lanfranchi, C. Warner, H. L. Johnson, C. Roeder, J. D. Choi, C. Funk, Y. Malenkiy, M. Eckert, et al. A corpus of full-text journal articles is a robust evaluation tool for revealing differences in performance of biomedical natural language processing tools. *BMC bioinformatics*, 13(1):1–26, 2012.

G. Wang, W. Zhang, R. Wang, Y. Zhou, X. Chen, W. Zhang, H. Zhu, and H. Chen. Label-free distant supervision for relation extraction via knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2246–2255, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1248. URL https://aclanthology.org/D18-1248.

R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, J. Ji, G. Cao, D. Jiang, and M. Zhou. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online, Aug.

2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.121. URL `https://aclanthology.org/2021.findings-acl.121`.

L. Weber, M. Sänger, S. Garda, F. Barth, C. Alt, and U. Leser. Chemical–protein relation extraction with ensembles of carefully tuned pretrained language models. *Database*, 2022, 2022.

D. S. Wishart, Y. D. Feunang, A. C. Guo, E. J. Lo, A. Marcu, J. R. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. R. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

W. Xiong, F. Li, M. Cheng, H. Yu, and D. Ji. Bacteria biotope relation extraction via lexical chains and dependency graphs. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 158–167, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5723. URL `https://aclanthology.org/D19-5723`.

Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1206. URL `https://aclanthology.org/D15-1206`.

Z. Xu, D. Guo, D. Tang, Q. Su, L. Shou, M. Gong, W. Zhong, X. Quan, D. Jiang, and N. Duan. Syntax-enhanced pre-trained model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5412–5422, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.420. URL `https://aclanthology.org/2021.acl-long.420`.

K. Yang, L. He, X.-y. Dai, S. Huang, and J. Chen. Exploiting noisy data in distant supervision relation classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3216–3225, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1325. URL `https://aclanthology.org/N19-1325`.

M. Yasunaga, J. Leskovec, and P. Liang. LinkBERT: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.551. URL `https://aclanthology.org/2022.acl-long.551`.

Z.-X. Ye and Z.-H. Ling. Distant supervision relation extraction with intra-bag and inter-bag attentions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2810–2819, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1288. URL `https://aclanthology.org/N19-1288`.

B. Yu, X. Mengge, Z. Zhang, T. Liu, W. Yubin, and B. Wang. Learning to prune dependency trees with rethinking for neural relation extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3842–3852, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.341. URL `https://aclanthology.org/2020.coling-main.341`.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.

D. Zeng, K. Liu, Y. Chen, and J. Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1203. URL `https://aclanthology.org/D15-1203`.

Q. Zhang, C. Liu, Y. Chi, X. Xie, and X. Hua. A multi-task learning framework for extracting bacteria biotope information. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 105–109, Hong Kong, China, Nov. 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-5716. URL `https://aclanthology.org/D19-5716`.

S. Zhang, W. Lijie, X. Xiao, and H. Wu. Syntax-guided contrastive learning for pre-trained language model. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2430–2440, 2022.

Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1004. URL `https://aclanthology.org/D17-1004`.

Y. Zhang, P. Qi, and C. D. Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1244. URL `https://aclanthology.org/D18-1244`.

Y. Zhang, Y. Zhang, P. Qi, C. D. Manning, and C. P. Langlotz. Biomedical and clinical English model packages for the Stanza Python NLP library. *Journal of the American Medical Informatics Association*, 06 2021. ISSN 1527-974X.

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1139. URL `https://aclanthology.org/P19-1139`.

G. Zhou, M. Zhang, D. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736, 2007.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.