



國立臺灣大學管理學院資訊管理學研究所
碩士論文

Department of Information Management
College of Management
National Taiwan University
Master Thesis

利用依存句法於生物醫學關係萃取之表示學習
Representation Learning for Biomedical Relation
Extraction with Dependency Parsing

朱瑤章

Yao-Chang Chu

指導教授：魏志平 博士
Advisor: Chih-Ping Wei, Ph.D.

中華民國 109 年 7 月
July 2020

國立臺灣大學碩士學位論文
口試委員會審定書

利用依存句法於生物醫學關係萃取之表示學習
Representation Learning for Biomedical Relation
Extraction with Dependency Parsing

本論文係朱瑤章君（學號 R07725032）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 109 年 7 月 15 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

魏志平

張詠淳

吳嘉尚

所 長：

魏志平



誌謝

兩年的碩士生涯終於到了尾聲，人生也從迷惘之中漸漸找到方向。能走到現在，真的要感謝很多貴人相助，其中最重要的貴人就是指導教授—魏志平老師。老師學識淵博、對研究非常有熱誠，每每與老師討論論文都十分有收穫，不論是過去熱門的領域或最新的研究老師都能有系統地跟我們分享討論。另外，老師也願意與我們分享人生智慧和各種有趣的故事，讓我也學習到學術領域之外的知識，受益良多。在研究上，老師也十分包容、十分照顧我們，讓我沒有擔憂，能夠有信心地一步步前進，最終完成論文。我覺得能進入老師的實驗室真的是三生有幸。

另外，研究室的同學們也是支持我的最大助力之一。感謝辰宇、晉華、昺倫、芷伊在每個辛苦的日子，與我一起奮鬥、天馬行空地聊天、幫助我完成論文和許多被我忽略的事情。也感謝虹鈞為我們安排研究室大大小小的事情，從設備到時程上，若是沒有虹鈞，實驗室可能不能那麼順利地運作。另外也感謝學弟妹在報論文、打球以及其他實驗室的活動上帶給我的歡笑和學習。也感謝我的家人從小的支持與鼓勵，讓我可以無後顧之憂一路讀書，直到現在可以進入人生下個階段。

謝謝在我身邊的大家，有你們才有今天的我，也才有這篇論文的誕生，請讓我獻上最深的感謝。

朱瑤章 謹識

于國立臺灣大學資訊管理研究所

中華民國一百零九年七月



摘要

關係萃取的任務是從文本中自動學習、抽取兩個實體間的關係。近年來，神經網路模型被廣泛應用在關係萃取上，也取得了優異的表現。然而，神經網路需要大量的訓練資料，而在生醫領域，因為標記成本昂貴，缺乏大量的訓練資料，所以我們進一步探索只需要少量標記資料來微調模型的自監督式學習方法。

MTB 是一個利用自監督式學習方法的關係萃取模型，藉由相同兩實體組成的實體對(entity pair)出現在不同句子也可能隱含相同關係的假設，MTB 得以訓練任意兩實體間的關係向量表示。不像過去許多深度學習之關係萃取模型，MTB 並未利用額外的自然語言特徵，故我們認為若加入兩實體間的依存路徑資訊，有機會讓 MTB 訓練得更好。另外，由於 MTB 僅利用不同的兩實體對是否相同當作訓練依據，負面樣本(非完全相同的實體對)的選定格外重要，因此，我們認為除了 MTB 提出的兩種負面樣本外，還存在使 MTB 訓練更有效的負面樣本。

因此，基於 MTB 模型，我們提出兩個改善方向：(1) 藉由四種網路模組編碼並嵌入實體對之間的依存關係 (2) 藉由行內(inline)負樣本，使 MTB 模型不能只學會關鍵字匹配，而作為真正學到基於上下文的關係表示。在不同設置的實驗下，我們證明了相對於 MTB 原本架構，我們提出的兩個改善方向都能有效地提升關係萃取的效能。我們並探索了在簡單或複雜的句法關係下，更適合的依存神經網路模組，也證明了在更細粒度的方向性關係下，我們的模型仍能有效辨別並超越 MTB 原始架構的表現。

關鍵字: 關係萃取、生醫關係萃取、關係分類、深度學習、非監督式學習、自監督式學習



Abstract

Relation extraction is the task that learns and extracts relations between entities from the text. In recent years, neural network models have been widely used in relation extraction, and have achieved the state-of-the-art performance. However, neural networks require a large amount of training data. In the biomedical domain, because acquiring labeled instances is expensive and the training dataset is often small-sized, we further explore self-supervised learning methods that require only a small amount of labeling data for fine-tuning the model. Matching The Blank (MTB) is a self-supervised based relation extraction model. With the assumption that if two entity pair from two sentences are the same, it also implies that they are having the same relation, MTB can train the vector of relation representation between any two entities. However, unlike many deep learning relationship extraction models in the past, MTB does not use additional natural language features other than text. Hence, we believe that if the dependency parsing information between the two entities in a sentence is taken into account, there is an opportunity for MTB to be trained better. In addition, since negative samples play an important role in MTB training, the selection of negative



samples (non-identical entity pairs) is particularly important. Therefore, we believe there exists a new type of negative samples that is more effective for MTB training. Therefore, based on the MTB model, we propose two directions for improvement: (1) four neural network modules to encode the dependency relationship between entities, and (2) inline negative samples that the MTB model will not just learn to do keyword matching, but will truly learn context-based relation representation. With the various experiment settings for robustness, we prove that compared with the original structure of MTB, the two directions that we propose can improve the effectiveness of relation extraction. We also explore more suitable dependency modules under simple or complex dependency relationships of an entity pair, and also prove that under more fine-grained directional relations, our model can still effectively identify and outperform the original structure of MTB.

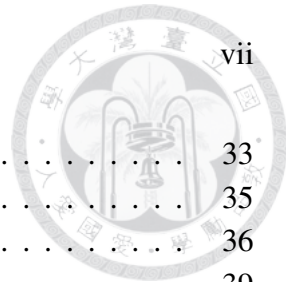
Keywords: Relation extraction, Biomedical relation extraction, Relation classification, Deep learning, Unsupervised Learning, Self-supervised Learning



Table of Contents

口試委員會審定書	i
誌謝	ii
摘要	iii
Abstract	iv
List of Figures	viii
List of Tables	x
Chapter 1 Introduction	1
1.1 Background	1
1.2 Research Motivation	5
1.3 Research Objectives	7
Chapter 2 Literature Review	8
2.1 Pipelined Relation Extraction	8
2.2 Joint learning Relation Extraction	14
2.3 Relation Extraction with few labeled data	16
2.4 Summary	19
Chapter 3 Methodology	20
3.1 Problem Definition of Self-supervised Training	20
3.2 Model Overview	21
3.3 BERT Encoder	22
3.4 Dependency Path Encoder	23
3.5 Merging Layer	29
3.6 Training Process	30
3.7 Few-shot Relation Classification	32
Chapter 4 Empirical Evaluations	33

TABLE OF CONTENTS



4.1	Experiment Setting for Self-supervised Learning	33
4.2	Compared Methods	35
4.3	Experiment setting for Few-shot Relation Classification	36
4.4	Experiment Results	39
Chapter 5 Conclusions		46
5.1	Contributions	46
5.2	Future Works	46
References		48



List of Figures

1.1	Through "remains", we know ENT1 is associated with ENT2 and they have a relationship of <u>ISA</u>	5
1.2	Through "treated with", we know ENT1 is associated with ENT2 and they have a relationship of <u>ADMINISTERED_TO</u>	6
2.1	An demonstration of process of a pipelined method, which starts from recognizing two entities from a given text (NER), then classifying the relation of this entity pair (RC)	8
2.2	An example of neural named entity recognition from (Lample et al., 2016)	10
2.3	An example of RNN relation classification model from (Zhang and Wang, 2015).	12
2.4	An example of CNN relation classification model from (Zhang and Wang, 2015).	12
2.5	An example of position embedding from (Zhang and Wang, 2015), that the position of word "son" is $[3, -2]$ is the relative distance to the two entities and will be further passed through the embedding layer.	13

LIST OF FIGURES



2.6 An example of joint neural relation extraction model from (Wei et al., 2019) 15

2.7 An example of training task of Word2vec (Mikolov et al., 2013) 18

2.8 An example of training task of BERT (Devlin et al., 2018) 19

3.1 An illustration of our relation encoder model. 21

3.2 An illustration of the BERT encoder 22

3.3 An illustration of dependency parsed result of a sentence. 23

3.4 An illustration of Pair base module 26

3.5 An illustration of Pair independent module 27

3.6 An illustration of Pair concatenated module 28

3.7 An illustration of Path module 28

3.8 An illustration of Merging Layer 29

4.1 The result of increasing the shot number on different modules 44



List of Tables

4.1	Statistics of the self-supervised dataset	34
4.2	Hyper parameters for self-supervised learning	35
4.3	Statistics of relation classification	37
4.4	Statistics of the general testing set	38
4.5	Statistics of the directional testing set	39
4.6	Main evaluation result of methods	40
4.7	Experiment results of different setting of five-class classification for two modules	41
4.8	Type-wised performance on which Pair(25)+inl outperforms most	41
4.9	Type-wised performance on which Path(25)+inl outperforms most	41
4.10	Entropy of each relation, where the green color relations represent the top 3 relations of Path module, and the red color relations represent the top 3 relations of Pair module.	43
4.11	Result of binary classification for directional relations	45
4.12	Result of multiclass classification for the general and directional testing sets	45



Chapter 1

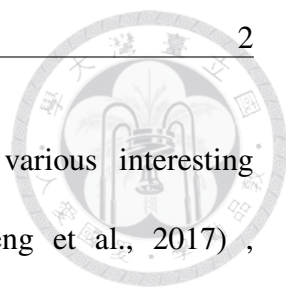
Introduction

1.1 Background

There are more and more unstructured data like texts available online, and we can learn the relations between entities by reading the context. However, reading and arranging the relations between entities by human is expensive, because large amount of corpus need to be processed, especially for some domains that require professional experts to label. Relation extraction (RE) is the task to classify the relation between two named entities (NE) from the text (Pawar et al., 2017), and a relation can be represented as a triple (relation, head, tail). For example:

- **IBM** was founded by **Harlow Bundy** in Binghamton. → (FOUNDED_BY, IBM, Harlow Bundy)
- Only **progestins** have this effect on **insulin receptors**. → (INTERACTS_WITH, progestins, insulin receptors)

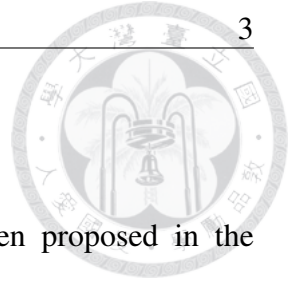
In most cases, the total number of relation types is usually pre-defined based



on the task we work on. Relation extraction can support various interesting applications, including detecting gene-disease relationships (Zheng et al., 2017) , detecting protein-protein interaction (Nadeau and Sekine, 2007) , building knowledge bases (Subasic et al., 2019), supporting question answering (Srihari and Li, 1999) . For example, in the biomedical domain, if we can extract the relation (Treat, Drug, Disease), (Interact, Drug, Drug) from the biomedical literature, we can help researchers better summarize the research results at a glance.

Relation extraction can be divided into two types: (1) pipelined method and (2) joint learning method (Zheng et al., 2017) . Pipelined methods decompose relation extraction into two subtasks: Named Entity Recognition (NER) and Relation Classification (RC). NER is to recognize information units (e.g., names, numeric expressions) (Nadeau and Sekine, 2007) , then RC further classifies the relation between the extracted entities into a pre-defined relation type. In contrast, joint learning methods conduct NER and RC together and these methods use an end-to-end model to detect entities and classify relations.

For the biomedical domain, we can just consider the relation classification task because entities recognition can be easily done by UMLS Metathesaurus mapping (Bodenreider, 2004). For relation classification, many recent works use neural network models with supervised training (Liu et al., 2013; Miwa and Bansal, 2016; Zhao et al., 2019). However, neural supervised training is associated with a main drawback that data labeling for entities and relations is expensive, especially in professional domains, and neural network needs a huge amount of training data. So if we just get into a new field,



few-shot learning based methods are required.

To address this drawback, the following approaches have been proposed in the literature:

- Distant supervised labeling approach (Mintz et al., 2009; Yao et al., 2010)
- Semi-supervised learning approach (Agichtein and Gravano, 2000; Brin, 1999; Ravichandran and Hovy, 2002)
- Self-supervised learning approach (Baldini Soares et al., 2019)

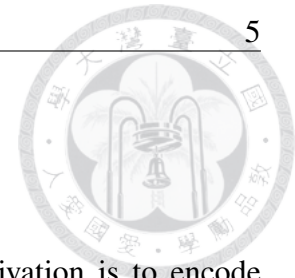
However, in the biomedical domain, there is not a well-established relation knowledge base, so the distant supervised labeling approach cannot be applied effectively. For semi-supervised learning, it often suffers from low precision and semantic drift (Mintz et al., 2009). The self-supervised approach with few-shot learning setting is considered most appropriate in this situation. Furthermore, this approach can facilitate another applications, e.g., relation discovery, because this approach does not need to pre-define relation types, it can easily generalize to different various relations with different granularities. Besides, for few-shot learning, we can define the relation types of interest, provide few labeled instances, and obtain few-shot classification results without re-training the representation model. In addition, we can also cluster the entity-pair embeddings and explore the clusters for new relations.

Matching The Blank (MTB) (Baldini Soares et al., 2019) is a self-supervised model, learning distributional similarity for relation between two entities from a large unlabeled corpus, and using this pre-trained model to conduct further classification tasks. The

intuition behind MTB is that identical entity pairs in different texts (i.e., sentences) are likely to have the same relation, and different entity pairs generally are associated with different relations. For example:

- **Bromocriptine** suppressed **prolactin** to < 3 ng/ml. → INHIBITS
- Suppression of **prolactin** by **bromocriptine** prevented this effect. → INHIBITS
- Ovulation and **pregnancy** were induced with **bromocriptine** in all 17 patients. → AUGMENTS

The training process of MTB is that, MTB uses BERT (Devlin et al., 2018), a multi-layer neural network model, to encode a relation statement, which is a sentence with a pair of entities, as a n -dimensional vector as the relation representation between the two entities. Given a relation statement, another relation statement whose entity pairs are the same as that of the given relation statement is called a positive sample. Accordingly, the vectors of two relation statements belonging to a positive sample should be similar. That is, the inner product of the vectors of the relation statements of a positive sample should be large. On the other hand, for a negative sample (two relation statements with at least one entity being different), the inner product of the corresponding relation statements should be small. Accordingly, MTB uses the binary cross entropy as the loss function, where a positive sample is labeled as 1, a negative sample is labeled as 0.



1.2 Research Motivation

Based on the model structure of MTB, our first research motivation is to encode additional information to improve the training effectiveness of MTB. MTB does not utilize extra information, but just uses BERT to encode inherent relation between two entities. However, many existing relation extraction methods also add dependency parsing as a part of the model or features (Fundel et al., 2007; Ningthoujam et al., 2019; Song et al., 2019), and human discriminate the relation between two entities not only by the meaning of entities and context, but also by the dependency relation. We can think about this process using our two examples shown in Figure 1.1 and Figure 1.2. So we would like to explore different structures to encode and represent dependency information in the model to help the training process of MTB.

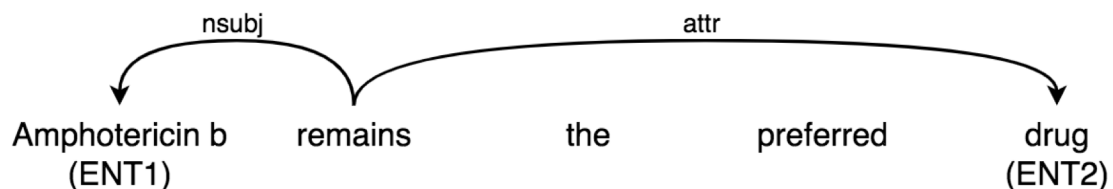


Figure 1.1: Through "remains", we know *ENT1* is associated with *ENT2* and they have a relationship of ISA.

Our second research motivation is to incorporate different type of negative samples to improve training effectiveness. MTB trains with positive samples and two types of negative samples, where the types of negative samples include (easy) negative examples (entity pairs of two relation statements are totally different) and strong negative examples (entity pairs of two relation statements are partially different). But there may still exist another type of negative samples, which are stronger and thus can provide additional clues

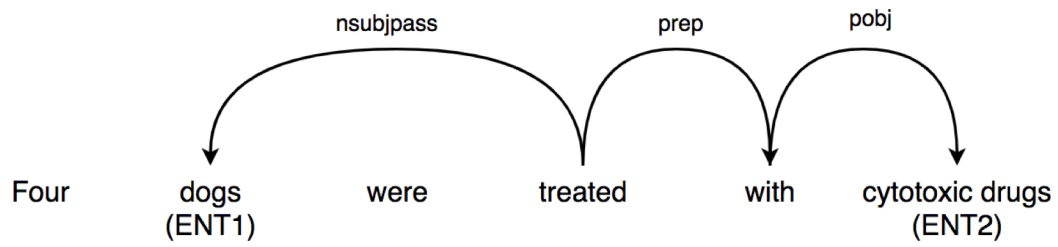


Figure 1.2: Through "treated with", we know *ENT1* is associated with *ENT2* and they have a relationship of ADMINISTERED_TO.

for MTB training.



1.3 Research Objectives

On the basis of the aforementioned research motivation, our research objectives include the following: First, we propose 4 network structures to encode dependency parsing in conjunction with the BERT structure. Second, we propose inline-negative samples for MTB training process as harder examples. Finally, we utilize Bio-BERT (Lee et al., 2020) as the pre-trained language model for the biomedical domain.

For the training process, we train the model by the MTB process with MEDLINE sentences, in which the entities in the sentences are labeled by SemMed (Kilicoglu et al., 2012). For evaluation, we use MEDLINE sentences labeled by a medical doctor as the testing instances of relation classification. Specifically, we extract the relation embeddings of sentences, and then employ the kNN classification method for few-shot learning for relation classification.



Chapter 2

Literature Review

2.1 Pipelined Relation Extraction

The pipelined relation extraction approach decomposes the relation extraction task into the named entity recognition (NER) subtask and the relation classification (RC) subtask (as Figure 2.1 illustrates).

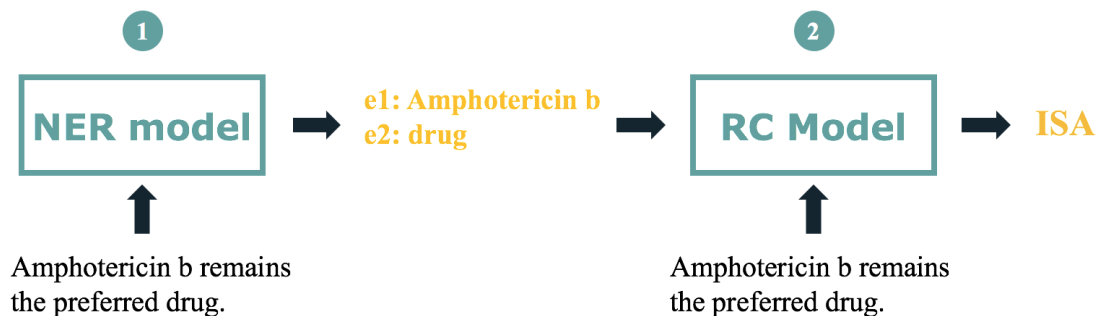
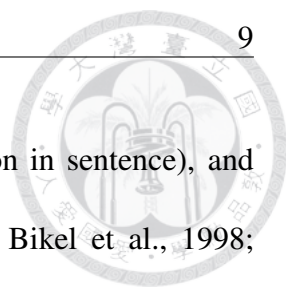


Figure 2.1: An demonstration of process of a pipelined method, which starts from recognizing two entities from a given text (NER), then classifying the relation of this entity pair (RC)

Named entity recognition (NER)

The objective of NER is to find the entities in a sentence. Early works employ hand-crafted features, such as case, digit pattern, punctuation, character, morphology



(prefix, suffix), part-of-speech, ngram pattern, local syntax (position in sentence), and frequency and co-occurrences of words and phrases (Bick, 2004; Bikel et al., 1998; Collins, 2002; Nadeau and Sekine, 2007). To train an NER model, the CRF method (Bundschuh et al., 2008; Sutton et al., 2007), SVM (Asahara and Matsumoto, 2003) and decision tree (Sekine, 1998) are applied for entity classification. Recently, many works consider NER as a sequence labeling problem and then use the neural network approach to recognize named entities from a given collection of sentences (Chiu and Nichols, 2016; Dernoncourt et al., 2017; Ju et al., 2018; Lample et al., 2016). Specifically, sequence labeling is to predict the label of every word token in a given sentence in a unified framework.

For example, Figure 2.2 uses a bi-LSTM with a CRF layer to predict whether each word token is within an entity (e.g., the begin, end or middle token of an entity) using the IOB format (inside, outside, beginning, etc.), and also predict the entity type of this token.

Relation Classification (RC)

The objective of relation classification is to classify the relation of the extracted entity pair in a given sentence into a pre-defined set of relation types. Early studies use hand-crafted features pertaining to the two extracted entities (Kambhatla, 2004; Pawar et al., 2017; Zhou et al., 2005). For example, (Kambhatla, 2004) uses the following features to construct a ruled-based model: word features (entities and words in between), entity types and mention level (e.g., nominal or pronoun) of the entities, overlap (e.g., number of words in between, if two entities are in the same NP, VP), dependency structure (e.g., POS of the word where the entity is dependent on, path between the two entities or

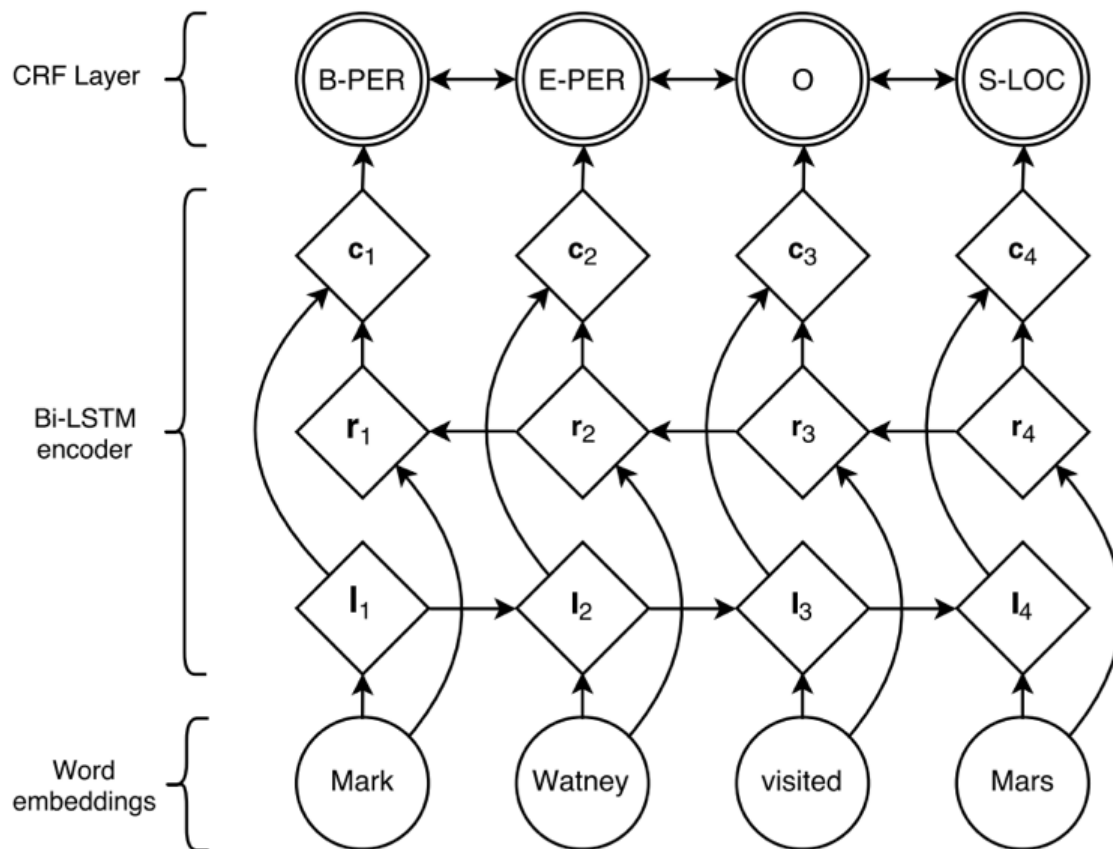
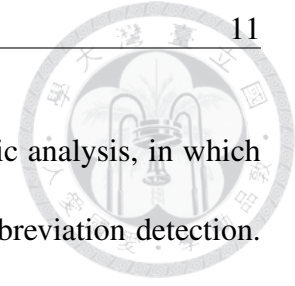


Figure 2.2: An example of neural named entity recognition from (Lample et al., 2016)

the number of links of the focal path).

Recently, many works adopt the neural network approach, such as CNN (Lin et al., 2016; Liu et al., 2013; Zeng et al., 2014, 2015) and RNN (Kavuluru et al., 2017; Miwa and Bansal, 2016; Peng et al., 2018) to extract features and learn a relation classification model.

SemRep (Rindflesch and Fiszman, 2003; Rindflesch et al., 2005) is a rule-based NLP system, using lexical and ontological semantics from Unified Medical Language System (UMLS) (Bodenreider, 2004; Lindberg et al., 1993), to extract semantic relations from biomedical literature in PubMed (Kilicoglu et al., 2020). The process of SemRep



(Kilicoglu et al., 2020) is as follows: The first phase is pre-linguistic analysis, in which SemRep performs sentence splitting, tokenization, and acronym/abbreviation detection. The second phase is lexical/syntactic analysis. Specifically, SemRep lookups from UMLS SPECIALIST Lexicon (McCray et al., 1994) to know the POS tags of words. In the third phase, SemRep conducts referential analysis by using MetaMap (Aronson and Lang, 2010) to detect and link the entities in a sentence to the ontology in database. MetaMap is a program linking UMLS Metathesaurus from biomedical text. In the last phase, SemRep conducts relational analysis; i.e., given two entities within a sentence, it predicts predication (relation) of the sentence by lexical, syntactic and semantic features. For example, one of the rules for the **ISA** relation is as follows: Two NPs separated by some keywords (e.g., is, remain, such as), so for example: “ **Non-steroidal anti-inflammatory** drugs such as **indomethacin**.” will be identified as having an **ISA** relation between the two highlighted entities in the sentence.

(Zhang and Wang, 2015) (see Figure 2.3) use $e1$, $/e1$, $e2$, $/e2$ as position indicators, for example: $\langle e1 \rangle$ people $\langle /e1 \rangle$ have been moving back into $\langle e2 \rangle$ downtown $\langle /e2 \rangle$. The model inputs word embedding into a Bi-LSTM model, and takes the global max pooling of the hidden states as the representation of the sentence.

PCNN (Zeng et al., 2015) (see Figure 2.4) takes a sequence of word embedding and position embedding (see Figure 2.5) and segments the sentence into three parts by two entities as input. Accordingly, PCNN performs the convolution of embedding and the max-pooling on each part separately, and then, with a fully connected layer, generates the relation representation vector of a relation statement.

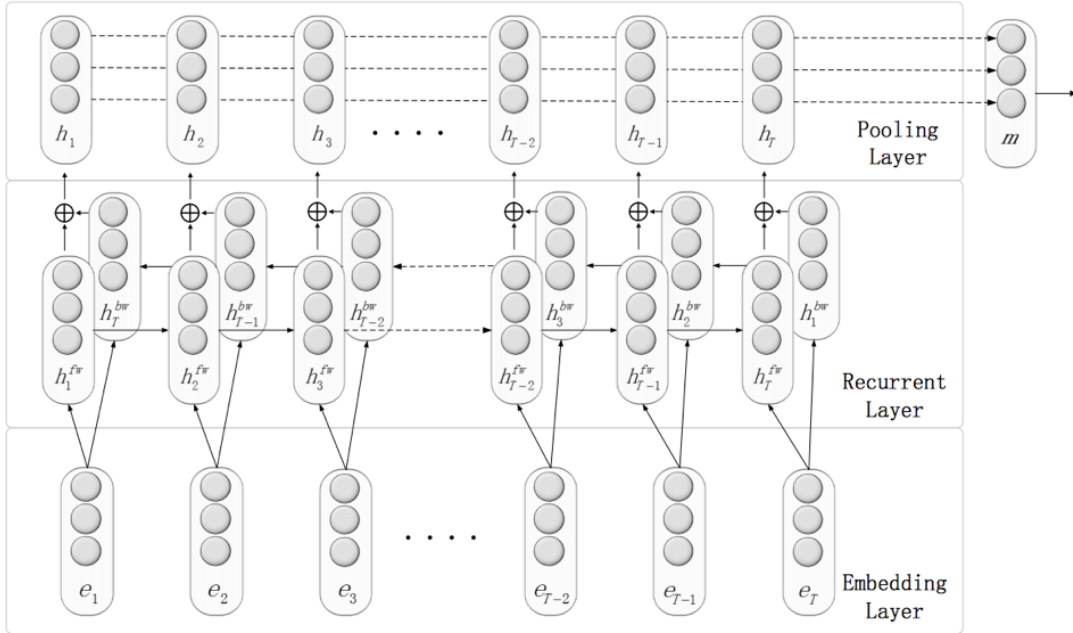


Figure 2.3: An example of RNN relation classification model from (Zhang and Wang, 2015)

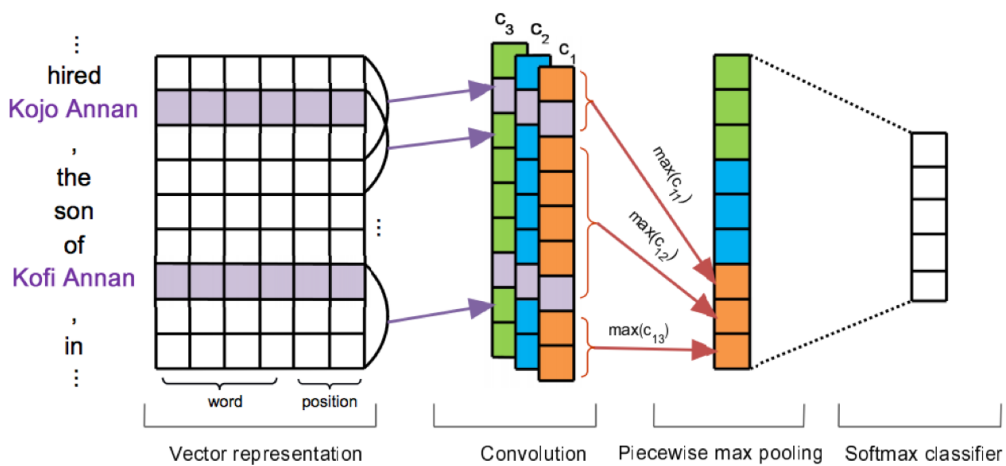


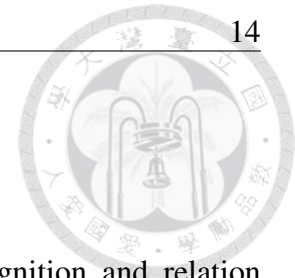
Figure 2.4: An example of CNN relation classification model from (Zhang and Wang, 2015)



... hired **Kojo Annan** , the son of **Kofi Annan** , in ...

A diagram illustrating position embedding. Two curved arrows originate from the word "son" in the sentence. The left arrow points to the word "Kojo Annan" and is labeled with the number "3". The right arrow points to the word "Kofi Annan" and is labeled with the number "-2".

Figure 2.5: An example of position embedding from (Zhang and Wang, 2015), that the position of word "son" is $[3, -2]$ is the relative distance to the two entities and will be further passed through the embedding layer.



2.2 Joint learning Relation Extraction

Joint learning relation extraction conducts named entity recognition and relation extraction subtasks jointly because the pipeline approach is prone to the propagation of errors from the NER task to the RE task (Pawar et al., 2017).

Traditionally, integer linear programming (Roth and Yih, 2004) and graphical model (Roth and Yih, 2002) are adopted to make a global decision. Recently, applying the deep learning, many neural network based joint model also proposed (Fu et al., 2019; Giorgi et al., 2019; Wei et al., 2019; Yu et al., 2019).

The study by (Wei et al., 2019) learns to predict (subject entity, relation, object entity) triples directly with BERT (Devlin et al., 2018) as the base encoder, instead of predicting entities and relations in a sentence separately, as Figure 2.6 illustrates. First, a layer of NN is used to predict whether each token is the start or end of a subject entity with token representations. Then, based on each detected subject entity, object tagger concatenates targeted token representation and the averages of token representations within the subject to predict the start or end position of the corresponding object entity.

(Yu et al., 2019) separate the task as entity pair extraction and relation type classification. First, they use a shared parameters encoder to encode the token representations. Then, they use a HE Extractor to distinguish head entities with max pooled global contextual embedding and token hidden state at current step. Similarly, a TER Extractor is adopted to predict tail entities and relations with global embedding, current hidden state, head entity representation and head-tail relative distance position

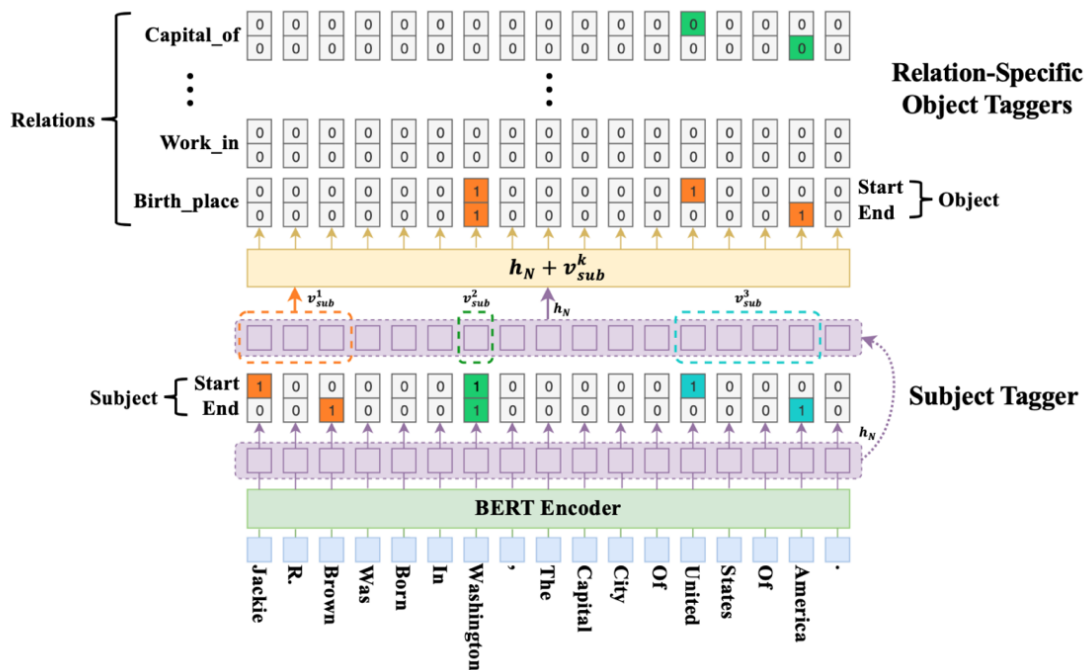
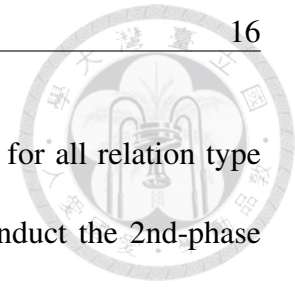


Figure 2.6: An example of joint neural relation extraction model from (Wei et al., 2019)

embedding. The structure of HE Extractor and TER Extractor is a Hierarchical Boundary Tagger (HBT), which use Bi-LSTM to predict the start tags of entities and find the corresponding end tags.

(Giorgi et al., 2019) use BERT to predict NER first. Their proposed method then feeds the predicted result to build entity pair set, take the last word as head and tail entity, and classifies the relation of the pairs (NEG for no relation or wrong pair). (Fu et al., 2019) encode token representation with Bi-LSTM, and build the graph of tokens based on the syntactic dependency relation between tokens with a dependency parser. After this, they use Bi-GCN (Kipf and Welling, 2016) to further combine the neighbor tokens information for each token. Then, based on the token representation, they use a fully connected layer to predict the entity type and relation of the token as the 1st-phase prediction. Using



the predicted relation as edges, they further build relational graphs for all relation type separately, again using Bi-GCN and a fully connected layer to conduct the 2nd-phase prediction, and consider the 1st and 2nd entity losses and relation loss for training.

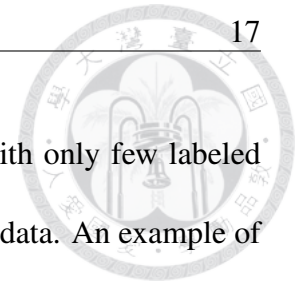
2.3 Relation Extraction with few labeled data

For the biomedical domain, we can detect the entities easily by UMLS linking tools. However, for the relation classification training data, there are many unlabeled sentences from PubMed, but only very limited labeled sentences (each is with two entities and a relation) to serve as the training data for relation classification (Bravo et al., 2015; Krallinger et al., 2017; Van Mulligen et al., 2012). To deal with this problem, this study focuses on relation classification with few labeled data but with ample unlabeled data. To address relation extraction with few labeled data, prior studies have proposed several different approaches that can be classified into three categories: distant supervised labeling, semi-supervised learning and self-supervised learning.

Distant supervised labeling

Distant supervision (Mintz et al., 2009; Yao et al., 2010) based methods deal with the problem of lack of labeled examples. Distant supervision method uses a knowledge base to obtain relation tuples, assuming that when a sentence contains both entities in a tuple, we assume that this sentence depicts the relation defined in the tuple. But in some specific or fast-changing domains (e.g., biomedical domain), they are not associated with a well-established knowledge base (e.g., WikiData).

Semi-supervised learning



Semi-supervised learning methods use lots of unlabeled data with only few labeled data for training with various assumptions to utilize these unlabeled data. An example of such assumption is smoothness assumption: Two data may have the same label if they are close in a high density region.

Some semi-supervised works for relation extraction using bootstrapping are adopted (Agichtein and Gravano, 2000; Brin, 1999; Ravichandran and Hovy, 2002). In (Brin, 1999), they iteratively find patterns from a set of tuples (each contains an entity pair and a relation) or find tuples from a set of patterns. Despite the fact that bootstrap only needs a few seed samples, because this approach generates initial patterns from a very limited labeled data, the initial patterns often suffer from low precision and semantic drift (Mintz et al., 2009).

Self-supervised learning

Self-supervised learning (Baldini Soares et al., 2019; Devlin et al., 2018; Lan et al., 2019; Mikolov et al., 2013; Pennington et al., 2014; Vaswani et al., 2017), sometimes called unsupervised learning or pre-training, requires only unlabeled data in order to formulate a pretext learning task (Kolesnikov et al., 2019). It uses the data itself (withhold some of the data) to generate labels, and then trains a model in a supervised manner to predict the label (withheld part). After the pre-training process, most studies further fine-tune their models for the target task which is only few labeled data available. Self-supervised learning tasks have been popular and have been used in many state-of-the-art works from Word2Vec (Mikolov et al., 2013), Skip-Thought (Kiros et al., 2015), GPT (Radford et al., 2018), Transformer (Vaswani et al., 2017), BERT (Devlin

et al., 2018), to MTB (Baldini Soares et al., 2019).

Word2Vec (Mikolov et al., 2013) uses surrounding words to predict the central word (CBOW) or uses a central word predict its surrounding words (Skip-gram), as Figure 2.7 illustrates.

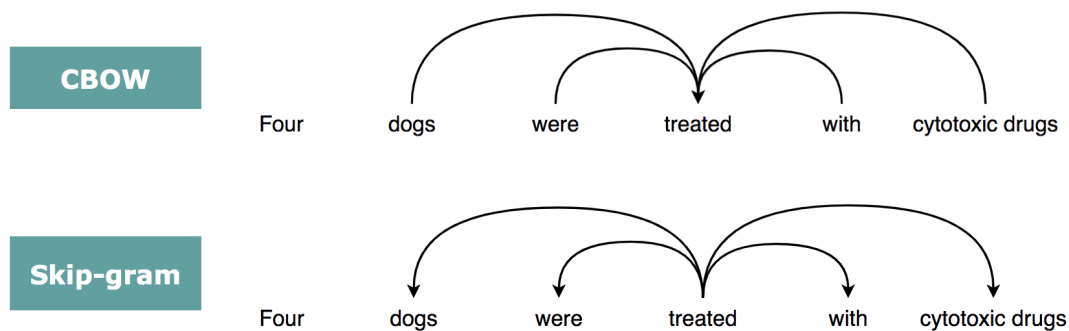


Figure 2.7: An example of training task of Word2vec (Mikolov et al., 2013)

BERT (Devlin et al., 2018) trains a language model with Masked Language Model (MLM) that predicts the masked token by its context and Next Sentence Prediction (NSP) that predicts whether two sentences are consecutive sentences or not with a deep self-attention model (please see Figure 2.8). BERT's model is the encoder of Transformer (Vaswani et al., 2017), which is a multi-layer neural network model, using position embedding and multi-head self-attention to replace RNN structure to build a deep model. BERT's model pre-trains on large scaled corpus, which makes it a powerful language model.

MTB (Baldini Soares et al., 2019) trains a language model with the same entity pairs in different sentences as positive samples and different entity pairs as negative samples. If two entity pairs are identical, their vector representations should be similar, otherwise

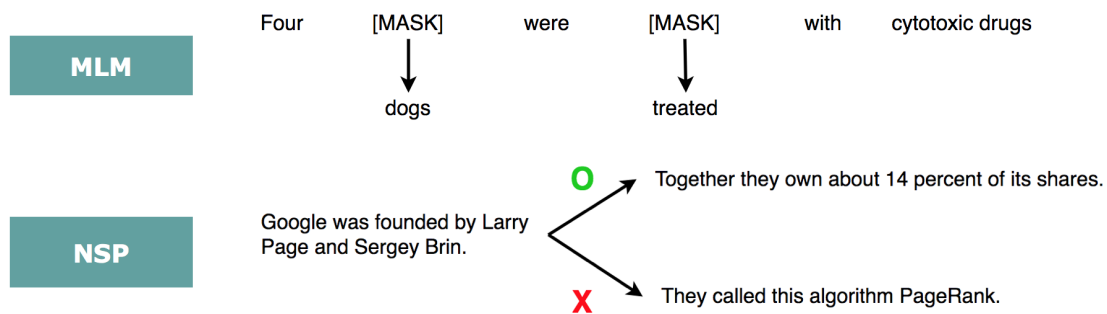


Figure 2.8: An example of training task of BERT (Devlin et al., 2018)

should be dissimilar. To train a model to learn the relation from context instead of entity linking, MTB randomly masks some entities with a `[BLANK]` token.

2.4 Summary

Because the self-supervised learning approach has achieved the state-of-the-art performance (Baldini Soares et al., 2019), and can deal with the challenge of the availability of only few labeled data, we will focus on improving MTB, the self-supervised relation representation model. We believe that there still exists some room for MTB model to improve, which can be split by two folds:

- Encode additional information to improve training effectiveness: MTB model can take into account dependency parsing information of the focal sentence when learning relation representation.
- Incorporate different type of negative samples to improve training effectiveness: We develop harder negative samples that help MTB train better.



Chapter 3

Methodology

3.1 Problem Definition of Self-supervised Training

Given a relation statement $r = (seq, s1, s2)$, $seq = [t_0, t_1, \dots, t_n]$ is a sentence, formed by a sequence of tokens (a token is part of a term), which is tokenized by BERT tokenizer. $s1 = [ent1_start, ent1_end]$ and $s2 = [ent2_start, ent2_end]$ which mean the start and the end positions of the first entity and those of the second entity, respectively. The entity in the front is called the first entity (entity 1 or ent1), and the other is called the second entity (entity 2 or ent2). For example:

amphotericin b remains the preferred **drug**.

→ $seq = [CLS] am \ ##ph \ ##oter \ ##ici \ ##n \ b \ remains \ the \ preferred \ drug \ . \ [SEP]$ is the token sequence after tokenizing process.

→ $s1 = [2, 6]$, $s2 = [10, 10]$ are the span of the first entity and the second entity.

The task of self-supervised is to learn a model to encode a relation statement to a n dimensional vector.



3.2 Model Overview

As Figure 3.1 shows, our main model, relation encoder, consists of three components:

BERT encoder, dependency path encoder, and merging layer.

Relation encoder

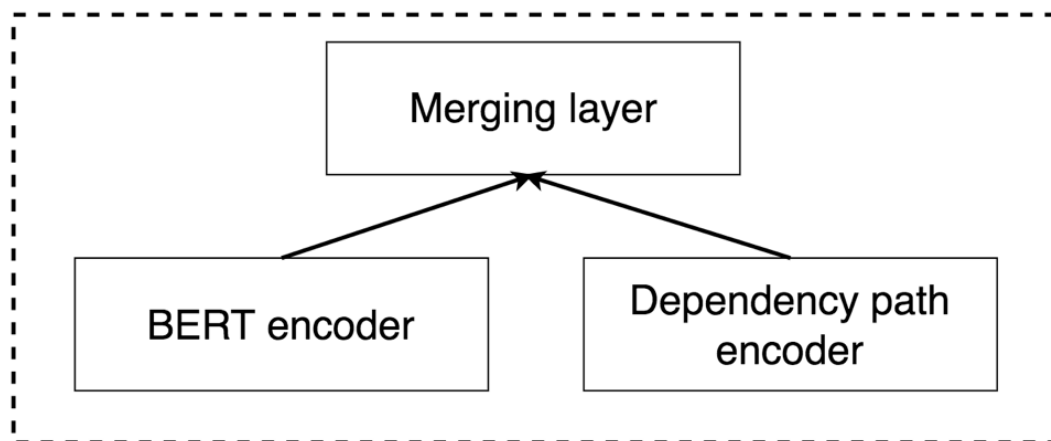


Figure 3.1: An illustration of our relation encoder model.

- BERT encoder
- Dependency path encoder
- Merging layer

For the BERT encoder, we follow the setting of MTB. For the dependency path encoder, we propose four different modules to encode the dependency path between two entities in a sentence. For the merging layer, we use a linear layer to merge two representations generated by the two encoders and output the final relation representation.



3.3 BERT Encoder

Following the best experimental result reported in (Baldini Soares et al., 2019), we use **ENTITY MARKERS – ENTITY START** setting with BERT (see Figure 3.2) to encode a relation statement as the base relation encoder

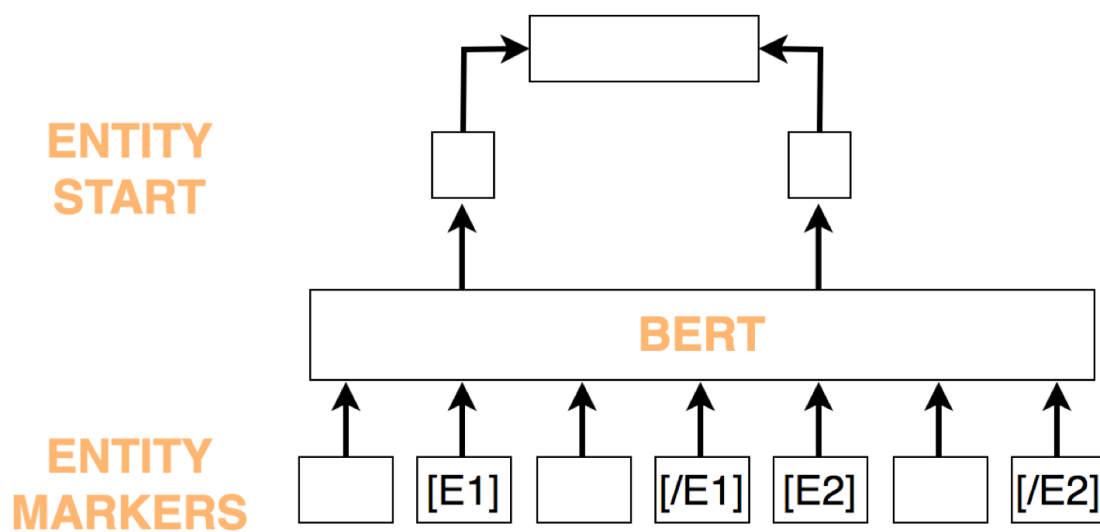


Figure 3.2: An illustration of the BERT encoder

ENTITY MARKERS setting add $[E1]$, $[/E1]$, $[E2]$, $[/E2]$ as the special tokens of the start ($[E1]$, $[E2]$) and end ($[/E1]$, $[/E2]$) of an entity to encode the position spans of the two entities. For example:

Amphotericin b remains the preferred drug. \rightarrow [CLS] $[E1]$ am ##ph ##oter ##ici ##n
b $[/E1]$ remains the preferred $[E2]$ drug $[/E2]$. [SEP]

The **ENTITY START** setting concatenates the hidden state (representation) vectors of $[E1]$ and $[E2]$, $r = [h_{e1}|h_{e2}]$

The BERT model takes the whole sequence of tokens as input, conducts multi-layer

embedding, self-attention and linear transformation to get the representation of every token, and we can add a fully connected layer on this pretrained model and fine-tune this layer for the target task.

3.4 Dependency Path Encoder

The dependency path between two entities is the path from entity 1 to the lowest common ancestor of entity 1 and entity 2 and then to entity 2, for example:

- Four dogs were treated with cytotoxic drugs (Figure 3.3)
- Entity 1: dogs Entity 2: cytotoxic drugs
- Lowest common ancestor: treated
- Path: dogs ← treated → with → cytotoxic drugs

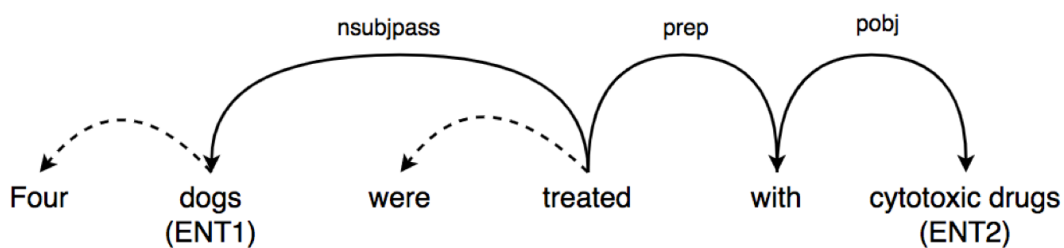
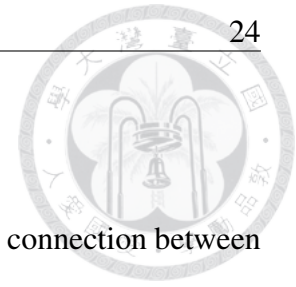


Figure 3.3: An illustration of dependency parsed result of a sentence.

We use SpaCy ¹ dependency parser to parse our data. SpaCy is a fast and accurate (Honnibal and Johnson, 2015; Neumann et al., 2019) industrial-strength natural language processing package in Python.

¹<https://spacy.io/>



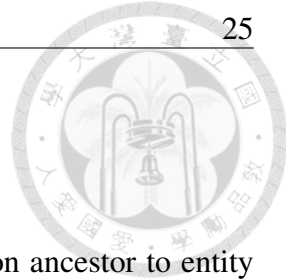
Dependency path

A dependency path is formed by edges (where each edge denotes connection between two terms as head and tail) in the path (e.g., “treated with” is one of the edges in the example above), which includes:

- Relation type: The dependency labels defined by spaCy (e.g., nsubjpass, prep, pobj) plus “END” label which means the tail term is the lowest common ancestor of the two entities. In this study, we also add “LONG” label, meaning the path is longer than the maximum path length allowed.
- POS type of tail: The part-of-speech labels of tail term defined by spaCy (e.g., NOUN, VERB, ADP), plus “ENT” label which means the tail term is one of the two entities.
- Identifier of start entity: If the edge starts from entity 1 to the lowest common ancestor, the Identifier should be 0. If the edge points to entity 2, Identifier should be 1.

An example of dependency path is as follows (Four **dogs** were treated with **cytotoxic drugs**):

- **dogs** ← treated (‘nsubjpass’, ‘ENT’, 0)
- treated ← X (‘END’, ‘VERB’, 0)
- **cytotoxic drugs** ← with (‘pobj’, ‘ENT’, 1)
- with ← treated (‘prep’, ‘ADP’, 1)



- treated ← X ('END', 'VERB', 1)

The reason that we reverse the sub-path from the lowest common ancestor to entity 2 (in the above example, treated ← with ← cytotoxic drugs) is that, we think keeping the same direction for both sub-paths from entity 1 and entity 2 to the lowest common ancestor can simplify the encoding complexity that the model needs to learn and just focuses on encoding only one structure. For discriminating the direction of an edge, model still can utilize identifier of start entity to identify from the two directions.

For a dependency path longer than 14 edges, we replace the whole path by: [('LONG', 'ENT', 0), ('LONG', 'ENT', 1)]. We set the maximum dependency path length as 14 because only 0.4% dependency paths of our testing data is longer than 14.

Dependency pair

Alternatively, to prevent encoding information that is too complicated, we also develop a dependency pair encoder, which is just the starting edge of entity 1 and that of entity 2. For example:

[('nsubjpass', 'ENT', 0), ('END', 'VERB', 0), ('pobj', 'ENT', 1), ('prep', 'ADP', 1), ('END', 'VERB', 1)]
 → ('nsubjpass', 'pobj')

Base module

To encode the information of dependency path, we propose four different modules to map a path or a part of a path to vector representation. The first model is Base module. It simply does nothing, i.e., does not encode dependency information (i.e., without using



dependency information).

Pair base module

As Figure 3.4 illustrates, pair base module encodes the dependency pair for the two entities using the same embedding space and add them together. $W_{pb} \in R^{L_r \times d_{pb}}$ is the embedding layer for both edges of the entity pair, L_r is the total number of relation types, and d_{pb} is the embedding dimension of the pair base encoder. $edge_h, edge_t \in R^{L_r}$ is the head edge and tail edge for the entity pair. $h_{pb} = edge_h W_{pb} + edge_t W_{pb}$ is the embedding representation of the entity pair.

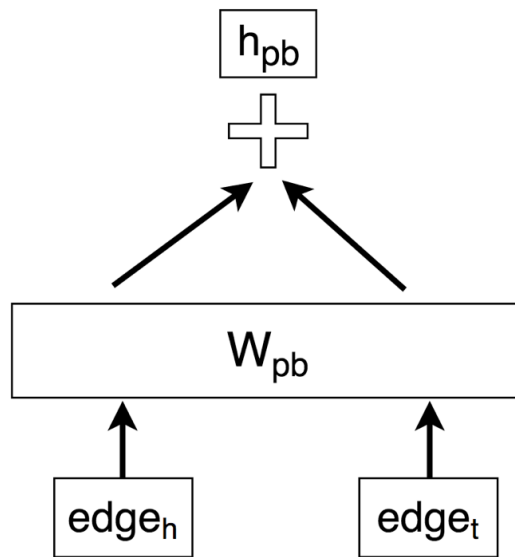


Figure 3.4: An illustration of Pair base module

Pair independent module

Pair independent module (see Figure 3.5) encodes dependency pair using different embedding spaces and adds them together. $W_{pih}, W_{pit} \in R^{L_r \times d_{pi}}$ is the embedding layer for head edge and tail edge of the entity pair, respectively. d_{pi} is the embedding dimension of the pair independent encoder. $h_{pi} = edge_h W_{pih} + edge_t W_{pit}$ is the embedding representation



of the entity pair.

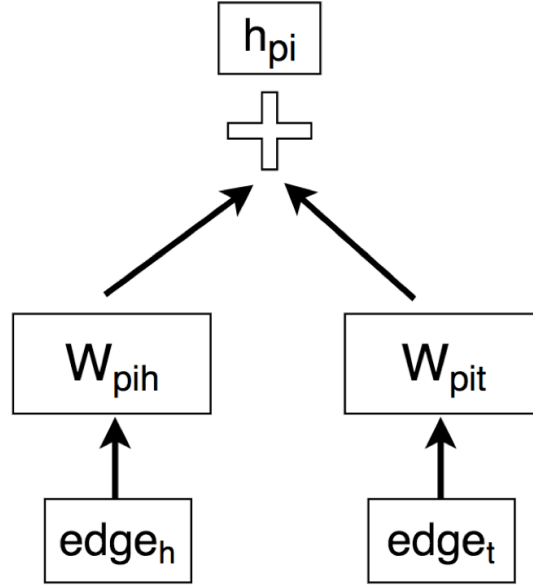


Figure 3.5: An illustration of Pair independent module

Pair concatenated module

Pair concatenated module encode (as Figure 3.6 shows) encodes the dependency pair using different embedding space and concatenates them together. $W_{pch} \in \mathbb{R}^{L_r \times d_{pch}}$, $W_{pct} \in \mathbb{R}^{L_r \times d_{pct}}$ is the embedding layer for head edge and tail edge of the entity pair, and d_{pch}, d_{pct} is the embedding dimension of head and tail edge of the pair concatenated encoder. $h_{pi} = edge_h W_{pch} \oplus edge_t W_{pct}$ is the embedding representation of the entity pair.

Path module

Path module (see Figure 3.7) encodes the dependency path using LSTM. $W_{rel} \in \mathbb{R}^{L_r \times d_{rel}}$, $W_{pos} \in \mathbb{R}^{L_r \times d_{pos}}$, $W_{idf} \in \mathbb{R}^{2 \times d_{idf}}$, where L_p is the total number of POS tagging types, and $d_{rel}, d_{pos}, d_{idf}$ is the embedding dimensions for relation type, POS tagging and identifier of start entity of an edge in entity path, and d_{lstm} is the hidden state dimension of

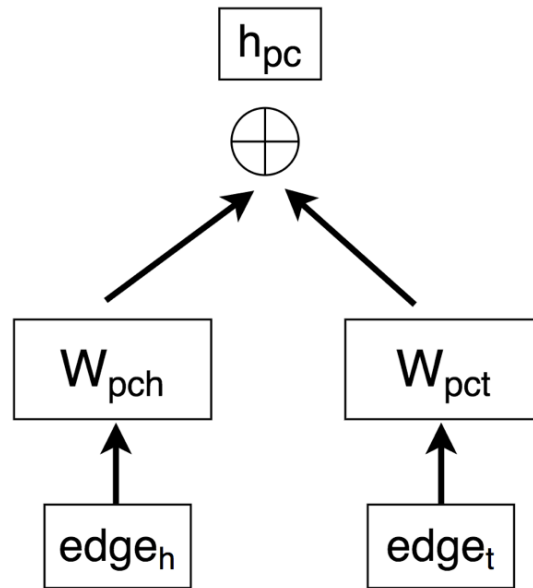


Figure 3.6: An illustration of Pair concatenated module

LSTM module. $h_{path} = LSTM(rW_{rel} \oplus pW_{pos} \oplus iW_{idf})$ is the representation of dependency path, where r, p, i is relation type, POS tagging, and identifier of an edge. We then input the edges to the LSTM model and take the hidden state of last token as the representation.

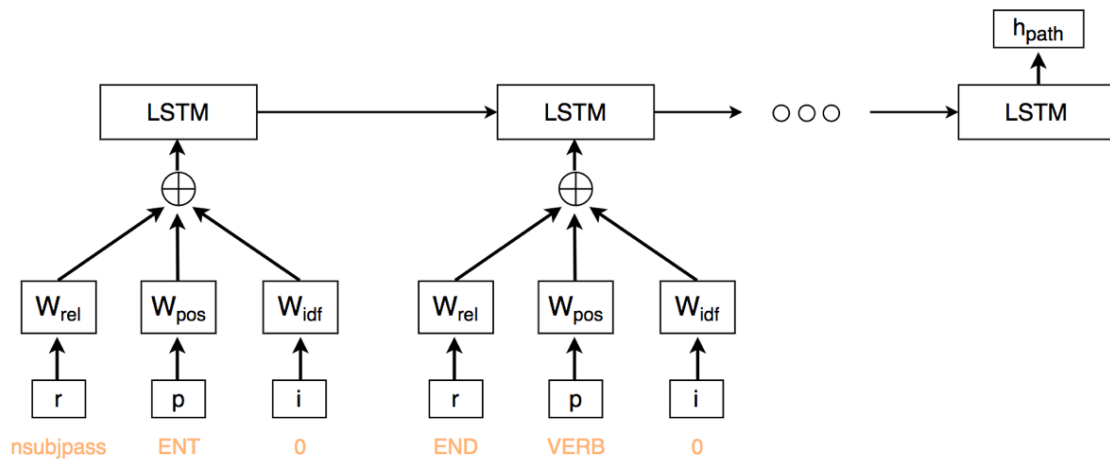
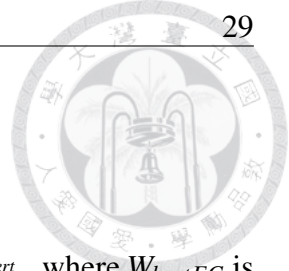


Figure 3.7: An illustration of Path module



3.5 Merging Layer

The Merging Layer is shown in Figure 3.8. $W_{bertFC} \in R^{(d_{bert} * 2) \times d_{bert}}$, where W_{bertFC} is the fully connected layer for MTB output. d_{bert} is the hidden dimension of BERT model output. The reason that d_{bert} is multiplied by 2 is because ENTITY START structure concatenates the start of two entities ([E1] and [E2]) as the relation representation. $rel = h_{dp} \oplus (h_{bert} W_{bertFC})$, where h_{dp}, h_{bert} are hidden state (embedding) of the dependency path encoder and the output of BERT, and rel is the representation of the whole relation statement.

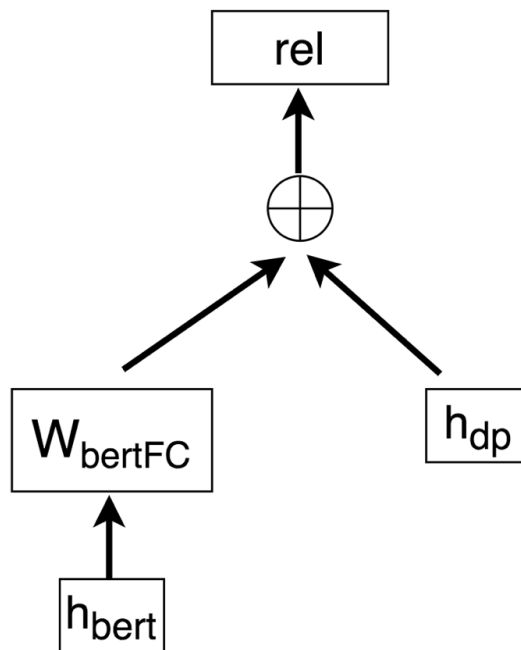


Figure 3.8: An illustration of Merging Layer



3.6 Training Process

An example for self-supervised learning is comprised of two relation statements, which can be simplified as two entity pairs, $r1 = (e11, e12)$, $r2 = (e21, e22)$, and there are three types of examples proposed by MTB on the basis of the two pairs:

- Positive example: $e11 = e21$ and $e12 = e22$
- (Easy) Negative example: $e11 \neq e21$ and $e12 \neq e22$
- Hard negative example: $(e11 = e21$ and $e12 \neq e22)$ or $(e11 \neq e21$ and $e12 = e22)$

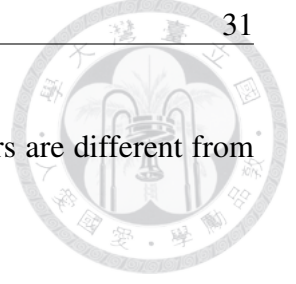
We propose a new type of negative examples called inline negative example, and the definition is as follows:

- Inline negative example: $(e11 = e21$ and $e12 \neq e22)$ or $(e11 \neq e21$ and $e12 = e22)$
and $r1, r2$ are in the same sentence

The intuition behind inline negative examples is that, for the relation statements in the same sentence, the model cannot just discriminate whether two statements are the same relation solely based on matching keywords. In this case, the model also needs to know the dependency relation among the entities and keywords, so inline negative examples serve as harder examples.

For a step during the training process, we randomly choose an entity pair and form a positive group and a negative group for the focal entity pair:

- Positive group: includes relation statements whose entity pairs are the same as the focal entity pair



- Negative group: includes relation statements whose entity pairs are different from the focal entity pair

For a batch in one step, we generate b examples, $b = b_p + b_n + b_{hn} + b_{in}$, where b_p, b_n, b_{hn}, b_{in} are the batch size of positive, negative, hard negative and inline negative examples. For a positive example, we randomly draw two statements from the positive group and label it as 1. For a negative, hard negative or inline negative example, we randomly draw a statement from the positive group and a statement from the negative group and label it as 0

The loss function of self-supervised learning is the binary cross entropy loss, which is calculated as follows:

$$L = - \sum_{i=1}^n y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i)) \quad (3.1)$$

x_i is an example, y_i is the label of this example, and n is the batch size. $f(x) = \frac{1}{1 + e^{(-RE(r1) \cdot RE(r2))}}$, which RE is the relation encoder model, and $x = (r1, r2)$ is relation statements pair of an example. With this binary cross entropy loss, the model learns to map the representations of statements with the same entity pair to similar locations in the space and map the representations of statements with different entity pairs to different places.



3.7 Few-shot Relation Classification

The task of few-shot relation classification is that with only few labeled data (for example, from 1 to 100) for each pre-defined relation type, we try to determine the label of each testing instance. This task is useful and practical because when getting into a new domain, we do not have so many labeled data or it costs too much time or money to label. So, if the few-shot task can be learned well, it can help us explore a novel domain quickly. The input and the output of few-shot relation classification is as follows:

- Input: Relation types (e.g., LOCATION_OF, ADMINISTERED_TO, ISA), shot number of labeled training data (relation statements and relation labels), labeled testing data
- Output: Predicted labels for the testing data, accuracy score of predicted results on the testing data

Following MTB and FewRel (Han et al., 2018), we adopt kNN as the classification method. Specifically, the learning and prediction process is that, we first use the relation encoder to get the vector representation of training and testing instances. Subsequently, we feed the few-shot training data to kNN and use the cosine similarity function as the metric. The testing instance will be classified to the majority class of the k instances in the training set that are closest to the target testing instance. Finally, we evaluate all instances in the testing data and calculate the accuracy score.



Chapter 4

Empirical Evaluations

4.1 Experiment Setting for Self-supervised Learning

For data collection, we download 93,825,701 relation statements from SemMed (Kilicoglu et al., 2012). We randomly sub-sample the whole dataset with 1% probability for each sample, and get 726,685 statements from the data as the training dataset.

For named entity recognition for inline negative examples, we extract 3,048,683 distinct entities as the entity set from all statements and build an entity recognition system (i.e., if a term in a sentence is exactly matched an entity in the set, we label the term as the matched entity) to generate inline negative examples. For entities in the entity set, we lowercase them, replace consecutive blanks with only one blank, and remove entities whose POS-tags are not 'PROPN' or 'NOUN' by spaCy.

Then, we preprocess the dataset. First, we remove those relation statements if one of their two entities labeled by SemMed is not in the sentence. We also remove the relation statements if the total count of their entity pairs are less than 5 in our dataset, because there are not enough sentences containing this entity pair so that the positive examples for

this entity pair will be not enough. Then, we tokenize sentences by BERT tokenizer and remove those relation statements if the length of the tokenized sentences is longer than 60 (constrained by the limited memory of our GPU). Table 4.1 shows the statistics of our final dataset for self-supervised learning.

Table 4.1: *Statistics of the self-supervised dataset*

Number of relation statements	535,339
Distinct group of entity pairs	390,757
Average token length of relation statements	39.66
Average dependency path length of entity pairs	5.70

The values for the hyperparameters for self-supervised learning in our work are given in Table 4.2. We first freeze the BioBERT pretrained BERT layers untrainable, which is the BERT encoder in our relation encoder, and we only train the dependency path encoder and merging layer. After 10 steps, we unfreeze the pretrained BERT layers and make all the layers in the model trainable. We also apply decaying layer-wised learning rates; that is, for a layer in BERT, it will get a learning rate which is 0.95 multiplied by the learning rate of its upper layer. The batch sizes of each type of examples are as follows:

- $b = 32$, $b_p = 16$, $b_n = 8$, $b_{hm} = 6$, $b_{in} = 2$ (6.25% inline negative examples).
- $b = 32$, $b_p = 16$, $b_n = 8$, $b_{hm} = 4$, $b_{in} = 4$ (12.5% inline negative examples)

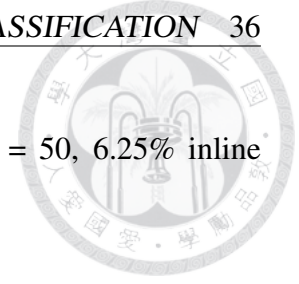
Table 4.2: Hyper parameters for self-supervised learning

BERT module	BioBERT-Base
Batch size	32
Max sequence length	60
Learning rate	0.00005 with Adam
Epoch	1
Layer-wised learning rate decay	0.95
Warm up steps for linear learning rate scheduler	50
Unfreeze steps	10
Random seed	1126

4.2 Compared Methods

In this study, we conduct experiments on 12 different methods for comparing the effect of different dependency encoding modules, different number of hidden dimensions and different ratios of inline negative examples. The methods compared in this research is as follows:

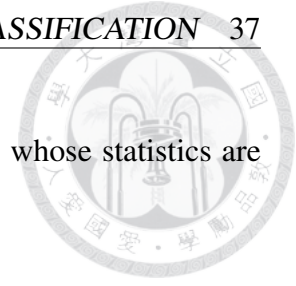
- Base: Base module with $d_{bertFC} = 525$, which is the baseline method
- Base+inl: Base module with $d_{bertFC} = 525$, replacing 6.25% hard negative examples with inline negative examples in a batch
- Pair(25): Pair base module with $d_{bertFC} = 500$ and $d_{pb} = 25$
- Pair(25)+inl: Pair base module with $d_{bertFC} = 500$ and $d_{pb} = 25$, 6.25% inline negative examples
- Pair(25)+inl2: Pair base module with $d_{bertFC} = 500$ and $d_{pb} = 25$, 12.5% inline negative examples



- Pair(50)+inl: Pair base module with $d_{bertFC} = 500$ and $d_{pb} = 50$, 6.25% inline negative examples
- Pair-Ind(25)+inl: Pair independent module with $d_{bertFC} = 500$ and $d_{pi} = 25$, 6.25% inline negative examples
- Pair(13+12)+inl: Pair concatenated module with $d_{bertFC} = 500$ and $d_{pch} = 13$ and $d_{pct} = 12$, 6.25% inline negative examples
- Path(25): Path module with $d_{bertFC} = 500$ and $d_{rel} = 10$ and $d_{pos} = 10$ and $d_{idf} = 5$ and $d_{lstm} = 25$
- Path(25)+inl: Path module with $d_{bertFC} = 500$ and $d_{rel} = 10$ and $d_{pos} = 10$ and $d_{idf} = 5$ and $d_{lstm} = 25$, 6.25% inline negative examples
- Path(25)+inl2: Path module with $d_{bertFC} = 500$ and $d_{rel} = 10$ and $d_{pos} = 10$ and $d_{idf} = 5$ and $d_{lstm} = 25$, 12.5% inline negative examples
- Path(50)+inl: Path module with $d_{bertFC} = 500$ and $d_{rel} = 20$ and $d_{pos} = 20$ and $d_{idf} = 10$ and $d_{lstm} = 50$, 6.25% inline negative examples

4.3 Experiment setting for Few-shot Relation Classification

For our data collection, we invited a medical doctor to help us label the data from SemMed, and we obtain 15,641 statements with 37 relation type labels. We follow the same pre-processing steps as those for the self-supervised learning dataset.



After pre-processing, we obtain a dataset for relation classification, whose statistics are summarized in Table 4.3.

Table 4.3: *Statistics of relation classification*

Number of relation statements	10,710
Average token length of relation statements	41.97
Average dependency path length of entity pairs	5.89

For different research purposes, we further process our original dataset and build 2 subsets. The first one is general testing set, where we choose the top 15 relation types out of 37 relation types to build our testing dataset, and there are a total of 10,298 relation statements remained. The other dataset is directional testing set, where the object and subject relation of (entity 1, entity2) and (entity 2, entity 1) was labeled as different relations. We build this dataset because our domain expert has identified the object and subject of each entity pair, so more fine-grained directional relations between the two entities of some relations can be studied.

For directional relations, an entity being a subject or an object matters and has different meaning in this relation. We modify the labels of examples of these relation types. That is, if object entity is in front of subject entity, we replace the label with the original label plus '_r', which means a reverse relation (e.g., ISA → ISA_r). The directional relations include: 'LOCATION_OF', 'ADMINISTERED_TO', 'ISA', 'PART_OF', 'INHIBITS', 'AFFECTS', 'PROCESS_OF', 'STIMULATES', 'DISRUPTS', 'AUGMENTS', 'PRODUCES'.

For non-directional relations, the order of subject and object does not matter.

They include ‘INTERACTS_WITH’, ‘COMPARED_WITH’, ‘COEXISTS_WITH’, ‘NEG_INTERACTS_WITH’

Two examples for directional relation and non-directional relation is as follows:

- amphotericin b was more efficacious than fluconazole → HIGHER_THAN
- only progestins have this effect on insulin receptors → INTERACTS_WITH

The statistics of the general testing set is shown in Table 4.4, and the statistics of the directional testing set is listed in Table 4.5.

Table 4.4: *Statistics of the general testing set*

Relation type	#	Relation type	#	Relation type	#
LOCATION_OF	2194	INHIBITS	571	STIMULATES	284
ADMINISTERED_TO	1606	AFFECTS	457	DISRUPTS	212
ISA	1576	PROCESS_OF	428	AUGMENTS	152
PART_OF	1084	COMPARED_WITH	336	NEG_INTERACTS_WITH	124
INTERACTS_WITH	861	COEXISTS_WITH	295	PRODUCES	118

For each experiment, we will average 60 different results to obtain a more robust result. We choose kNN seeds from 30 to 49 and average 20 experiment results as the score, and we average 1, 3, 5 neighbors of kNN experiment results as the score. And for the similarity metric, we choose the cosine similarity.

Table 4.5: *Statistics of the directional testing set*

Relation type	#	#_r	Relation type	#	#_r	Relation type	#	#_r
LOCATION_OF	568	1626	INHIBITS	419	152	STIMULATES	191	93
ADMINISTERED_TO	826	780	AFFECTS	386	71	DISRUPTS	162	50
ISA	671	905	PROCESS_OF	333	95	AUGMENTS	110	42
PART_OF	343	741	COMPARED_WITH	336	—	NEG_INTERACTS_WITH	124	—
INTERACTS_WITH	861	—	COEXISTS_WITH	295	—	PRODUCES	70	48

4.4 Experiment Results

First, we evaluate the overall performance of each model, using accuracy (%) as the evaluation metric, and then, we compare models in different relation types [5, 10, 15] classification task, for which we choose top 5, top 10, top 15 most common relations from the general testing set. We further compare models in different shot number (number of training examples, in the range of [5, 10, 25, 50, 100]) for more robust comparisons. The main evaluation result of our proposed methods are shown in Table 4.6.

In our main result, Pair(25)+inl module and Path(25)+inl module perform best among all the models. Comparing with the baseline model (Base module), these two models have an average of 1.62% of performance improvement. However, Pair-Ind(25)+inl only has 0.74% improvement and Pair(13+12)+inl models does not have any improvement, so we will skip these two models and dig deeper into Pair(25)+inl and Path(25)+inl models.

First, we would like to investigate why Pair(25)+inl performs better on 5 relation

Table 4.6: Main evaluation result of methods

# relType # shot	5					10					15					diff
	5	10	25	50	100	5	10	25	50	100	5	10	25	50	100	
Base	45.01	53.55	63.27	70.08	75.16	34.24	41.40	50.10	56.64	62.72	30.23	36.54	44.55	51.01	57.62	0.00
Base+inl	43.78	52.16	62.91	69.63	75.51	33.55	40.07	49.72	56.52	62.54	29.46	35.13	43.78	50.40	56.95	-0.67
Pair(25)	45.50	53.93	64.30	70.81	75.96	35.36	42.46	51.94	58.12	63.61	31.08	37.40	46.08	52.20	58.39	1.00
Pair(25)+inl	46.54	55.26	65.88	72.05	77.38	35.82	42.49	51.70	58.24	64.44	31.64	37.38	45.98	52.40	59.26	1.62
Pair(25)+inl2	44.01	52.34	62.64	69.60	75.63	34.31	40.61	50.05	57.12	63.30	30.26	35.67	44.13	50.86	57.48	-0.27
Pair(50)+inl	44.04	51.58	61.38	67.81	73.61	33.80	39.61	48.52	54.93	61.16	29.67	34.91	43.05	49.35	56.15	-1.50
Pair-Ind(25)+inl	45.13	54.21	64.57	71.09	76.34	35.27	41.90	51.08	57.58	63.22	30.89	36.77	45.48	51.82	57.83	0.74
Pair(13+12)+inl	43.40	50.74	62.04	68.56	73.99	33.15	39.51	48.90	55.36	61.16	29.11	34.79	43.73	50.20	56.51	-1.40
Path(25)	46.34	54.45	64.92	71.65	76.35	34.61	41.56	51.07	57.83	63.71	30.88	37.05	45.87	52.54	58.78	1.03
Path(25)+inl	45.30	55.25	64.84	71.46	77.04	35.63	43.32	52.22	58.56	64.52	31.35	38.16	46.44	52.86	59.48	1.62
Path(25)+inl2	43.45	51.65	61.87	69.36	75.24	33.72	40.48	49.95	56.77	63.07	29.52	35.44	44.40	51.01	57.71	-0.57
Path(50)+inl	45.90	54.10	64.88	71.30	76.38	35.02	42.06	51.34	57.63	63.19	31.00	37.21	45.66	51.88	58.13	0.90

types and Path(25)+inl performs better on 10 and 15 relation types. We want to know if the performance changes because Pair(25)+inl works better for few relation types (5 types) or there exist some relation types in our target 10 or 15 relation types that is hard for Pair(25)+inl to discriminate. So we control the total number of relation types for classification to 5 classes, and conduct experiments on 1~5, 6~10, 11~15, 16~20, 21~25 most common relation type sets from the whole testing dataset. After the experiments, the result is displayed in Table 4.7. We find that for each model, there exists different sets of relations that each model can perform better on. For Pair(25)+inl, it performs better on 1~5, 6~10 and 16~20 sets, and for Path(25)+inl, it performs better on 11~15 and 21~25 sets.

Then, we want to know that on which relation types Pair(25)+inl can outperform Path(25)+inl most and on which relation types Path(25)+inl outperforms Pair(25)+inl

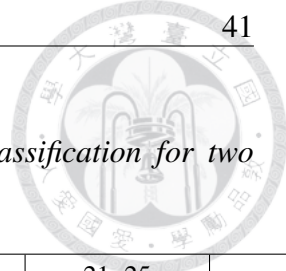


Table 4.7: Experiment results of different setting of five-class classification for two modules

# relType	1~5		6~10		11~15		16~20		21~25		diff
	5	10	5	10	5	10	5	10	5	10	
Pair(25)+inl	46.54	55.26	51.81	56.89	31.03	33.74	45.88	54.26	55.24	60.00	0.00
Path(25)+inl	45.30	55.25	51.80	56.51	32.77	35.29	43.92	52.02	55.36	63.55	0.11

most, so we evaluate class-wised performance of each relation type with f1 score, and we sort the difference of the performance between two models (how much one model performs better than the other model). As Table 4.8 and Table 4.9 illustrate, the top 3 relations that Pair(25)+inl outperforms most are INTERACTS WITH, PROCESS OF and AFFECTS relations, and those that Pair(25)+inl outperforms most are ISA, AUGMENTS and INHIBITS relations.

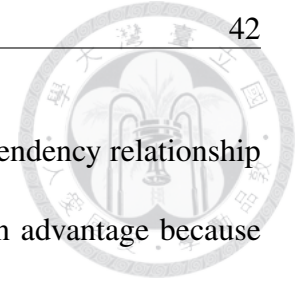
Table 4.8: Type-wised performance on which Pair(25)+inl outperforms most

module	Pair(25)+inl	Path(25)+inl	Pair - Path
INTERACTS_WITH	32.10	29.42	2.68
PROCESS_OF	62.85	60.23	2.62
AFFECTS	28.13	25.70	2.43

Table 4.9: Type-wised performance on which Path(25)+inl outperforms most

module	Pair(25)+inl	Path(25)+inl	Pair - Path
ISA	79.50	82.31	-2.81
AUGMENTS	10.71	13.29	-2.57
INHIBITS	33.44	35.95	-2.50

Then, we want to know the reason why there exists type-wised difference among relations. We assume that because the Path module uses more dependency information comparing with the Pair module, so it has an advantage on relations that are more



complicated linguistically or on relations having a more diverse dependency relationship between two entities. For simpler relations, the Pair module has an advantage because it can avoid overfitting from encoding too much useless information. So we calculate the entropy of a relation, which is the entropy between two entities of all the relation statements belong to this relation. For example, assume there are 3 examples in a relation type: ('nsubjpass', 'pobj'), ('nsubjpass', 'END'), ('pobj', 'prep'). The probability distribution of entities is: nsubjpass=2/6, pobj=2/6, END=1/6, prep=1/6. Accordingly, the entropy of this relation is 1.918. We calculate the entropy for each relation, and sort the relations by their entropy in the descending order. As Table 4.10 illustrates, we put the top 3 relations that Pair(25)+inl outperforms most with red color and the top 3 relations that Path(25)+inl outperforms most with green color. After observing the table, we find that Pair(25)+inl outperforms most on the relation types with lower entropy, which means simpler relations, and Pair(25)+inl outperforms most on the relation types with higher entropy, which means more complicated relations. This match with our assumption that because Path module encodes more detailed information of dependency, so it has an advantage on more complicated relation statements, and Path module encodes simplified information that just enough for simpler relation statements, so it has an advantage on simpler relation statements.

We further examine the effect of inline negative examples. From Table 4.6, we compare Base, Pair(25) and Path(25) modules with Base+inl, Pair(25)+inl and Path(25)+inl modules. We find that Base+inl module has a 0.67% worse than Base module, but Pair(25) and Path(25) modules have 0.62% and 0.59% improvement

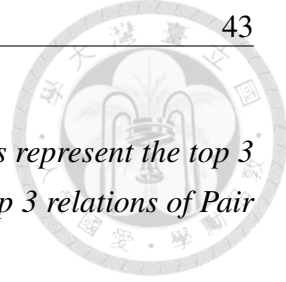


Table 4.10: Entropy of each relation, where the green color relations represent the top 3 relations of Path module, and the red color relations represent the top 3 relations of Pair module.

relation	entropy	relation	entropy
INHIBITS	2.947	NEG_INTERACTS_WITH	2.506
PRODUCES	2.934	CAUSES	2.505
ISA	2.875	INTERACTS_WITH	2.452
DISRUPTS	2.857	COMPARED_WITH	2.451
STIMULATES	2.725	COEXISTS_WITH	2.442
AUGMENTS	2.699	ASSOCIATED_WITH	2.435
NEG_INHIBITS	2.685	ADMINISTERED_TO	2.415
NEG_STIMULATES	2.579	AFFECTS	2.398
NEG_AFFECTS	2.527	LOCATION_OF	2.382
NEG_ADMINISTERED_TO	2.524	PROCESS_OF	2.364

respectively. So in general speaking, inline negative examples make a slight improvement on the accuracy of few-shot relation classification.

Then, we further investigate the effect of the number of inline negative examples. We adjust the number of inline negative examples from 2 to 4 (6.25% to 12.5%) in a batch and compare Pair(25)+inl and Path(25)+inl modules with Pair(25)+inl2 and Path(25)+inl2 modules. We find that Pair(25)+inl2 has a 1.9% worse than Pair(25)+inl and Path(25)+inl2 has a 2.19% worse than Path(25)+inl. Thus, increasing the number of inline negative examples does not lead to better performance, only few replacement from hard negative examples to inline negative examples is enough for the training.

We also examine the effect of the number of embedding dimensions of Pair and Path that perform best in the previous experiment. We compare Pair(25)+inl and Path(25)+inl modules with Pair(50)+inl and Path(50)+inl modules. We find that Pair(50)+inl has a

3.13% worse than Pair(25)+inl and Path(50)+inl has a 0.72% worse than Path(25)+inl. Accordingly, we can conclude that increasing the number of dimensions from 25 to 50 does not have a positive effect on both models.

We further study the trend of increasing the number of shots (number of training data of few-shot classification) for Base, Pair and Path modules over different numbers of relation types. We show the experimental results in Figure 4.1. We find that for every module and every number of relation types, increasing the number of shots will first lead to a huge increase in accuracy, while the increase will gradually become flattening, so a small amount of labeled examples is enough for a comparable performance.

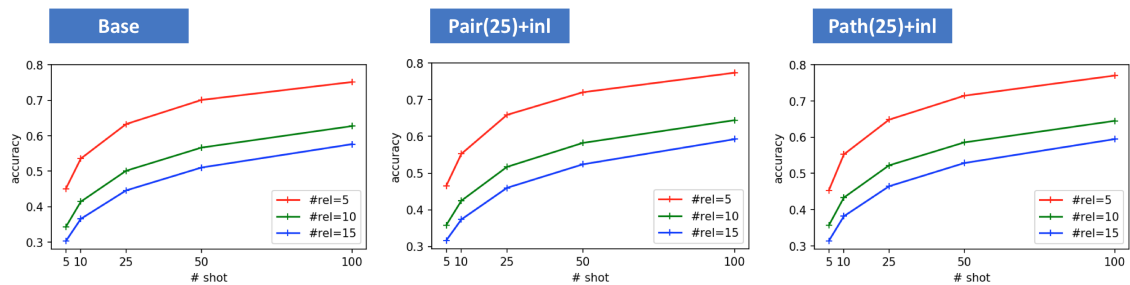
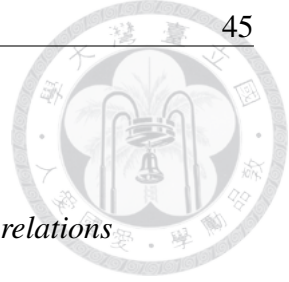


Figure 4.1: The result of increasing the shot number on different modules

We further investigate the fine-grained directional relations in the directional testing set. We want to know whether our models can discriminate directional relation types. We first conduct a binary classification for the top 3 most common directional relations. For example, for the ISA relation, we perform a binary classification on ISA and ISA_r. We show the experimental results in Table 4.11, and find that Base module alone can achieve a high accuracy over 90% with 100 shots, and Path module can further improve the accuracy by more 1.52% on average. This result suggests that our modules have the



ability to handle fine-grained directional relation types.

Table 4.11: Result of binary classification for directional relations

rel # shot	ADMINISTERED_TO					ISA					LOCATION_OF					diff
	5	10	25	50	100	5	10	25	50	100	5	10	25	50	100	
Base	92.02	95.01	96.67	98.14	98.92	67.89	78.45	88.18	91.80	93.30	86.78	90.32	93.77	95.82	97.38	0.00
Pair(25)+inl	95.24	96.82	97.44	98.20	99.03	71.56	79.74	86.11	89.13	91.71	87.02	90.73	94.86	96.91	98.40	0.56
Path(25)+inl	96.16	97.24	97.54	97.97	98.72	73.81	82.04	90.47	92.61	93.97	86.19	90.89	94.92	96.73	98.01	1.52

Then we want to know if we conduct an experiment on the complete directional testing set, whether the performance will change as compared to that of the general testing set (without the directions). For example, for the original 5 relations classification task, we split the directional relations (e.g., ISA and ISA_r) and reframe the task to 9 relations classification task (because among the top 5 relations, only INTERACTS_WITH is the non-directional relation and it should not be split), and compare the performance between 5 and 9 relations classification. Table 4.12 shows the experimental results.

Table 4.12: Result of multiclass classification for the general and directional testing sets

# relType # shot	5→9			10→17			15→26			diff
	5	10	25	5	10	25	5	10	25	
Base	45.00	53.55	63.27	34.24	41.40	50.10	30.24	36.54	44.55	0.00
Pair(25)+inl	46.54	55.26	65.88	35.82	42.49	51.70	31.64	37.38	45.98	1.62
Path(25)+inl	45.30	55.25	64.84	35.63	43.32	52.22	31.35	38.16	46.44	1.62
Base_r	47.07	55.79	65.55	36.39	43.52	51.63	32.23	38.60	46.31	1.63
Pair(25)+inl_r	49.63	58.08	67.41	38.40	45.57	54.07	34.08	40.12	48.50	3.79
Path(25)+inl_r	49.50	57.84	67.39	39.21	46.33	54.24	35.09	41.02	48.78	4.08



Chapter 5

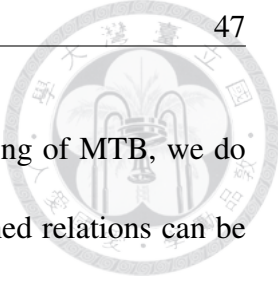
Conclusions

5.1 Contributions

Based on MTB, we develop four modules to encode dependency structure and propose inline negative examples as harder examples to improve self-supervised learning. We conduct a range of experiments to ensure the evaluation results are robust. Our Pair(25)+inl and Path(25)+inl modules outperform the original MTB model. We investigate the difference between Pair and Path modules and find Pair module fits better on simple dependency structure while Path module performs better on complicated structure. We find that our model has ability to learn fine-grained directional relations. When considering relation directions, Path module gets a substantial improvement over 4% in accuracy, and Base module and Pair module also have an improvement.

5.2 Future Works

In this study, we only use a subset (about 1%) of SemMed dataset because the limitation of training resource. Using more training data on self-supervised pre-training



task may further improve the classification performance. In the setting of MTB, we do not need to define the relation types beforehand, so more fined-grained relations can be detected by clustering the relation representation vectors. Future research can apply our proposed models for discovering new relation types. In this study, we do not fine-tune our model on all labeled data and compare with the state-of-the-art relation classification models that are trained only on labeled data. Future study can take our proposed models as the relation statement representation model and develop a more effective relation classification technique that learns a classification model from labeled data.



References

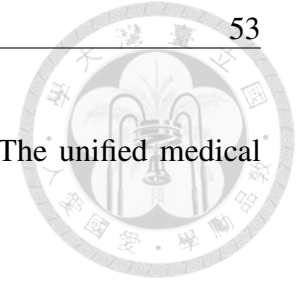
- Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. Proceedings of the Fifth ACM Conference on Digital Libraries (DL).
- Aronson, A. R. and Lang, F.-M. (2010). An overview of Metamap: historical perspective and recent advances. Journal of the American Medical Informatics Association, 17(3):229–236.
- Asahara, M. and Matsumoto, Y. (2003). Japanese named entity extraction with redundant morphological analysis. In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pages 8–15.
- Baldini Soares, L., FitzGerald, N., Ling, J., and Kwiatkowski, T. (2019). Matching the blanks: Distributional similarity for relation learning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.
- Bick, E. (2004). A named entity recognizer for Danish. In Proceedings of the 4th International Conference on Language Resources and Evaluation, pages 305–308.

- Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1998). Nymble: a high-performance learning name-finder. [arXiv preprint cmp-lg/9803003](https://arxiv.org/abs/cmp-lg/9803003).
- Bodenreider, O. (2004). The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl_1):D267–D270.
- Bravo, À., Piñero, J., Queralt-Rosinach, N., Rautschka, M., and Furlong, L. I. (2015). Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. *BMC Bioinformatics*, 16(1):55.
- Brin, S. (1999). Extracting patterns and relations from the world wide web. In Atzeni, P., Mendelzon, A., and Mecca, G., editors, *The World Wide Web and Databases*, pages 172–183, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Bundschus, M., Dejori, M., Stetter, M., Tresp, V., and Kriegel, H.-P. (2008). Extraction of semantic biomedical relations from text using conditional random fields. *BMC Bioinformatics*, 9(1):207.
- Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.
- Collins, M. (2002). Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 489–496.

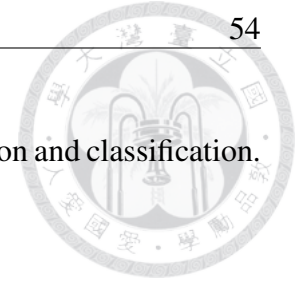
- Dernoncourt, F., Lee, J. Y., and Szolovits, P. (2017). Neuroner: an easy-to-use program for named-entity recognition based on neural networks. [arXiv preprint arXiv:1705.05487](#).
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. [arXiv preprint arXiv:1810.04805](#).
- Fu, T.-J., Li, P.-H., and Ma, W.-Y. (2019). 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.
- Fundel, K., Küffner, R., and Zimmer, R. (2007). Relex—relation extraction using dependency parse trees. [Bioinformatics](#), 23(3):365–371.
- Giorgi, J., Wang, X., Sahar, N., Shin, W. Y., Bader, G. D., and Wang, B. (2019). End-to-end named entity recognition and relation extraction using pre-trained language models. [arXiv preprint arXiv:1912.13415](#).
- Han, X., Zhu, H., Yu, P., Wang, Z., Yao, Y., Liu, Z., and Sun, M. (2018). Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. [arXiv preprint arXiv:1810.10147](#).
- Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In [Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing](#), pages 1373–1378.
- Ju, M., Miwa, M., and Ananiadou, S. (2018). A neural layered model for nested

- named entity recognition. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 1 (Long Papers), pages 1446–1459.
- Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, pages 22–es.
- Kavuluru, R., Rios, A., and Tran, T. (2017). Extracting drug-drug interactions with word and character-level recurrent neural networks. In 2017 IEEE International Conference on Healthcare Informatics (ICHI), pages 5–12. IEEE.
- Kilicoglu, H., Roseblat, G., Fiszman, M., and Shin, D. (2020). Broad-coverage biomedical relation extraction with SemRep. BMC Bioinformatics, 21:1–28.
- Kilicoglu, H., Shin, D., Fiszman, M., Roseblat, G., and Rindfleisch, T. C. (2012). SemMedDB: a PubMed-scale repository of biomedical semantic predications. Bioinformatics, 28(23):3158–3160.
- Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
- Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In Advances in Neural Information Processing Systems, pages 3294–3302.
- Kolesnikov, A., Zhai, X., and Beyer, L. (2019). Revisiting self-supervised visual

- representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1920–1929.
- Krallinger, M., Rabal, O., Akhondi, S. A., Pérez, M. P., Santamaría, J., Rodríguez, G., et al. (2017). Overview of the biocreative vi chemical-protein interaction track. In Proceedings of the sixth BioCreative Challenge Evaluation Workshop, volume 1, pages 141–146.
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.
- Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4):1234–1240.
- Lin, Y., Shen, S., Liu, Z., Luan, H., and Sun, M. (2016). Neural relation extraction with selective attention over instances. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, volume 1 (Long Papers), pages 2124–2133.



- Lindberg, D. A., Humphreys, B. L., and McCray, A. T. (1993). The unified medical language system. Yearbook of Medical Informatics, 2(01):41–51.
- Liu, C., Sun, W., Chao, W., and Che, W. (2013). Convolution neural network for relation extraction. In International Conference on Advanced Data Mining and Applications, pages 231–242. Springer.
- McCray, A. T., Srinivasan, S., and Browne, A. C. (1994). Lexical methods for managing variation in biomedical terminologies. In Proceedings of the Annual Symposium on Computer Application in Medical Care, page 235. American Medical Informatics Association.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119.
- Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). 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. Association for Computational Linguistics.
- Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using LSTMs on sequences and tree structures. arXiv preprint arXiv:1601.00770.



Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes, 30.

Neumann, M., King, D., Beltagy, I., and Ammar, W. (2019). Scispacy: Fast and robust models for biomedical natural language processing. arXiv preprint arXiv:1902.07669.

Ningthoujam, D., Yadav, S., Bhattacharyya, P., and Ekbal, A. (2019). Relation extraction between the clinical entities based on the shortest dependency path based LSTM. arXiv preprint arXiv:1903.09941.

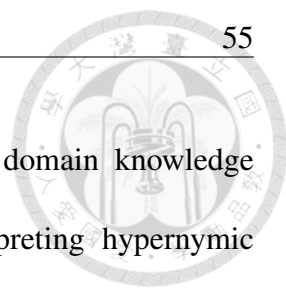
Pawar, S., Palshikar, G. K., and Bhattacharyya, P. (2017). Relation extraction: A survey. arXiv preprint arXiv:1712.05191.

Peng, Y., Rios, A., Kavuluru, R., and Lu, Z. (2018). Chemical-protein relation extraction with ensembles of svm, cnn, and rnn models. arXiv preprint arXiv:1802.01255.

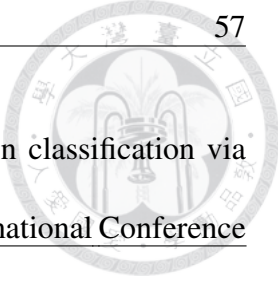
Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.

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

Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 41–47, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

- 
- Rindflesch, T. C. and Fiszman, M. (2003). The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. Journal of Biomedical Informatics, 36(6):462–477.
- Rindflesch, T. C., Fiszman, M., and Libbus, B. (2005). Semantic interpretation for the biomedical research literature. In Medical Informatics. Integrated Series in Information Systems, volume 8, pages 399–422. Springer.
- Roth, D. and Yih, W.-t. (2002). Probabilistic reasoning for entity & relation recognition. In Proceedings of the 19th International Conference on Computational Linguistics.
- Roth, D. and Yih, W.-t. (2004). A linear programming formulation for global inference in natural language tasks. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign.
- Sekine, S. (1998). Description of the Japanese NE system used for MET-2. In Proceedings of Seventh Message Understanding Conference (MUC-7).
- Song, L., Zhang, Y., Gildea, D., Yu, M., and Su, Z. W. J. (2019). Leveraging dependency forest for neural medical relation extraction. arXiv preprint arXiv:1911.04123.
- Srihari, R. and Li, W. (1999). Information extraction supported question answering. Technical report, Cymfony Net Inc., Williamsville, NY.
- Subasic, P., Yin, H., and Lin, X. (2019). Building knowledge base through deep learning relation extraction and wikidata. In Proceedings of AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering.

- Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. Journal of Machine Learning Research, 8(Mar):693–723.
- Van Mulligen, E. M., Fourier-Reglat, A., Gurwitz, D., Molokhia, M., Nieto, A., Trifiro, G., Kors, J. A., and Furlong, L. I. (2012). The eu-adr corpus: annotated drugs, diseases, targets, and their relationships. Journal of Biomedical Informatics, 45(5):879–884.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. CoRR, abs/1706.03762.
- Wei, Z., Su, J., Wang, Y., Tian, Y., and Chang, Y. (2019). A novel hierarchical binary tagging framework for joint extraction of entities and relations. arXiv preprint arXiv:1909.03227.
- Yao, L., Riedel, S., and McCallum, A. (2010). Collective cross-document relation extraction without labelled data. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1013–1023, Cambridge, MA. Association for Computational Linguistics.
- Yu, B., Zhang, Z., and Su, J. (2019). Joint extraction of entities and relations based on a novel decomposition strategy. arXiv preprint arXiv:1909.04273.
- Zeng, D., Liu, K., Chen, Y., and Zhao, J. (2015). 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.

- 
- Zeng, D., Liu, K., Lai, S., Zhou, G., and Zhao, J. (2014). Relation classification via convolutional deep neural network. In Proceedings of the 25th International Conference on Computational Linguistics, pages 2335–2344.
- Zhang, D. and Wang, D. (2015). Relation classification via recurrent neural network. arXiv preprint arXiv:1508.01006.
- Zhao, Y., Wan, H., Gao, J., and Lin, Y. (2019). Improving relation classification by entity pair graph. In Proceedings of the Eleventh Asian Conference on Machine Learning, pages 1156–1171.
- Zheng, S., Wang, F., Bao, H., Hao, Y., Zhou, P., and Xu, B. (2017). Joint extraction of entities and relations based on a novel tagging scheme. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, volume 1 (Long Papers).
- Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring various knowledge in relation extraction. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pages 427–434.