INSTITUTE FOR ADVANCED STUDIES IN BASIC SCIENCES

# Sentence-level Relation Extraction Using Neural Networks

*Author:*
Sohrab Pirhadi

*Supervisor:*
Dr. Ebrahim Ansari

*A thesis submitted in fulfillment of the requirements for the degree of Master of Computer Science*

*in the*

Department of Computer Science and Information Technology

October 14, 2023

# *Abstract*

Information Extraction (IE) refers to the process of automatically extracting structured data from unstructured sources to enable the utilization of such data by other applications. Extracting relations from textual sources, which seeks to detect the semantic relation represented between entities referenced in the texts, is a common sub-problem. The objective of the RE task is to develop automatic extractors that can identify and extract structured, relational information from unstructured sources like natural language text. Assigning a relationship label to a pair of entities may be considered a classification problem. As a result, supervised machine learning methodologies can be employed. It is essential to pre-process the data using methods from natural language processing to organize the textual contents into meaningful data structures before extracting relations from the unprocessed text. In addition, as relations are represented between entities, it is necessary to locate the entities using an entity extraction technique, which is another information extraction sub-problem. Relation extraction methods that use entity-annotated text are called pipeline approaches. Relations can be represented between two or more than two entities which are known as binary relations and N-ary relations, respectively.
This thesis limits our research to binary relations using pipeline approaches.

**Keywords:** Information Extraction, Relation Extraction, Deep Learning, Transfer Learning

# *Acknowledgements*

I would like to express my deepest gratitude to my advisor, Dr. Ansari, for his valuable support. His knowledge and critical analysis have helped me to shape my research. Discussions with him have helped me a lot to improve my research capabilities.
Also, I want to thank my friend, Babak Niroomand, for his help and support. Finally, I want to dedicate this thesis to my parents and my sister for their support and encouragement.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Humankind invented writing thousands of years ago. Since its creation, we have utilized this invention to store information and knowledge in a format that resembles natural language. Throughout much of human history, plain text has been the most efficient and comprehensive method of storing information. As a result, the majority of knowledge that humanity has generated and continues to generate is only accessible through natural language texts. These texts include Millions of books, documents, magazines, manuals, and scientific papers.

The Internet and mobile communication technologies have rapidly expanded and become widely used in recent years. As a result, people all over the world are becoming more interconnected, and natural language is the primary way of exchanging information through these channels. Every day, web pages, blogs, tweets, emails, and text messages from internet users create a massive amount of text data, leading to a significant increase in natural language text data.

One of the vital intellectual capabilities of the human brain is text understanding which is achieved by the creation of a systematic collection of concepts while reading a text. Despite the constant increase in the amount of available text data, our ability to read, process, and absorb knowledge from them remains limited. It is almost impossible for humans to manually read through all of the existing text data and report the extracted knowledge due to its abundance.

In order to access the valuable information contained in these texts, we need to utilize computer processing power to extract meaningful data automatically. Our goal is to equip computers with the ability to comprehend and interpret natural human languages like the human brain. To achieve this, we

must develop some techniques that will enable computers to process and understand language as intelligently as humans can.

Artificial intelligence (AI) is a field of computer science that endeavors to simulate or approximate human intelligence in machines. One of the critical achievements of human intelligence is the development of language and its related skills. Language is a fundamental and vital component of the human intellect, frequently acknowledged as "the hallmark of human intelligence" or "the jewel in the crown of cognition". It is not surprising that a lot of AI research is focused on language and linguistics.

The subfield of text mining is where AI researchers explore the topic of language. Text mining (text data mining or text analytics) is usually considered extracting and analyzing valuable information from text. It is not a standalone technology system and involves the integration of various technologies, such as data mining and machine learning. While data mining and machine learning are commonly used for data analysis, they typically only work with structured or semi-structured data.

Text content differs from structured databases and data warehouses as it is typically unstructured and described using natural language rather than data. Extracting information from unstructured text data is more complex than from structured data, but it is vital in organizing vast amounts of data into valuable knowledge.

In general, the main goal of text mining is to turn text into data to be suitable for analysis. To achieve this goal, we must apply some computationally intensive AI algorithms and statistical techniques to text documents. To perform text mining, various tasks and technologies like Information Extraction, Information Retrieval, and Natural Language Processing are combined to turn unstructured text data into a structured form.

All of these technologies are powerful tools with unique backgrounds and histories. Research related to these fields that involve working with text has been categorized as text mining at some point. Nevertheless, after the workshops (ILMC and IJCA in 1999), Text mining is now considered a sequential workflow of four stages: Information Retrieval, NLP, Information Extraction,

and data mining. In the following, we explain these stages based on their relevance to the subject of this thesis.

The initial step is to select the relevant documents from a vast pool of digital text documents, which is done using information retrieval. Information Retrieval systems aim to identify the subset of documents that match a user's query.

After retrieving a subset of text documents, computers must process and analyze the character strings. To comprehend natural languages as humans do, computers require input in a specific format. When analyzing text, it is possible to use multidimensional representations (also known as "Text as a bag of words") for simple applications. This method does not consider the order of the words during the analysis, but it is still adequate for tasks like classification, topic modeling, and recommender systems. This approach is sufficient for situations where predicting the text's semantic meaning is unnecessary.

However, the true richness of the text representation can be leveraged by treating text as sequences with a semantic meaning. The ordering of words in a document provides a semantic meaning that cannot be inferred from a representation based on only the frequencies of words in that document.

NLP is a research topic focusing on the interactions between computer and human languages. NLP tasks involve processing, understanding, or generating natural languages by analyzing the probability and dependency between words. NLP researchers are primarily concerned with enabling computers to understand human language. In particular, this field discusses how to program computers to process and analyze loads of natural language data.

Modern text mining systems apply extensive NLP techniques, such as parts of speech tagging, syntactic parsing, and other linguistic analysis types, to understand a text's semantic meaning. In other words, most text mining applications turn text into structured data for analysis by applying NLP methods.

During this stage of TM, the linguistic data about the text are extracted

from and marked up to the documents, which still hold an unstructured form of data.

In order to be mined as any other kind of data, the unstructured natural language document must be turned into data in a structured form. This stage is called Information Extraction, the data generated by NLP systems. The goal of information extraction is to automatically obtain specific information from unstructured text data, which can then be analyzed or utilized for more advanced natural language processing tasks like question-answering or populating knowledge bases. The process of extracting information from text involves multiple steps, which are illustrated as a pipeline in Figure 1.1. As the process continues, more and more abstract annotations are incorporated into the original text, resulting in a highly abstract representation.

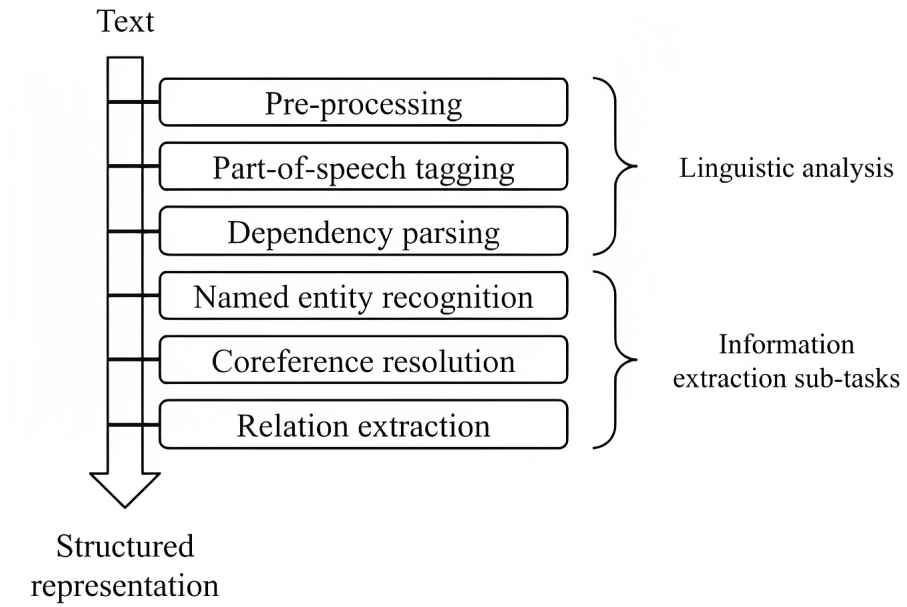


FIGURE 1.1: An information extraction process that involves several steps. Each step adds more abstract annotations to the input text, resulting in a progressively abstract representation of the original text. [1]

First, in the pre-processing step, the text is broken down into individual sentences and then each sentence is divided into smaller units such as words or sub-words through a process called tokenization. The next steps in linguistic analysis involve part-of-speech tagging and dependency parsing. Once the linguistic analysis is done, the named entity recognition identifies the types of entities mentioned in the text. Then, coreference resolution is used to locate and group all expressions that refer to the same entity. Sometimes, an entity mention may be unclear as it refers to more than one entity.

An instance of this confusion is when the term "Washington" is used, which could indicate the state or a person's name, such as "George Washington", depending on the context. The goal of entity linking is to identify and connect references to specific entities within a knowledge base that is organized in a structured manner. Currently, the only missing element is the structure connecting the entities. The primary goal of relation extraction is to establish a structure for the connections between mentioned entities.

Relation extraction is essential in Knowledge Graph, Text Data Mining, and Information Extraction. The task of relation extraction has garnered significant interest in AI and NLP conferences worldwide, resulting in numerous advancements in research. There are different approaches to traditional relation extraction methods, which include rule-based learning, weakly supervised learning, supervised learning, and unsupervised learning. Supervised learning specifically involves manually labeling a considerable amount of data, which demands extensive human and financial resources. As a solution to the difficult task of manual labeling, many methods have been proposed utilizing weak supervision and unsupervised learning.

Throughout the years, various traditional methods for relation extraction have been developed [2], [3], [4]. As a result, a few issues have arisen, including the need to manually label a significant amount of data, manually create Part-of-Speech tagging or dependency analysis, and lack of accuracy.
Thanks to the fast progress in deep learning technologies, researchers now widely rely on deep learning methods to address relation extraction issues. Over time, researchers have suggested different techniques using either Convolutional Neural Networks (CNN) [5], [6], [7] or Recurrent Neural Network (RNN) models [8].

It is worth noting that the accuracy of relation extraction has been enhanced even more since the introduction of the pre-trained language model BERT by Devlin et al. [9]. Relation extraction methods that are based on pre-trained language models tend to have higher accuracy compared to those based on CNN or RNN models. This is because pre-trained language models are capable of capturing more lexical, syntactic, and semantic features.

## 1.1 Development of Relation Extraction

The 1970s was the beginning of **information extraction (IE)** research. The first IE system called JASPER was developed in the 1980s by Carnegie Group [1]. The U.S. **Defense Advanced Research Projects Agency(DARPA3)** [2] provided the majority of the money for IE research in the middle of the 1980s, which was sparked by a series of Message Understanding Conferences [3].
The area of information extraction has received a lot of interest recently due to the increase of unstructured texts on the Web. The Web is now regarded as a collection of documents. From these materials, users must independently extract the key information. It is the responsibility of the user to extract crucial information from these pages on their own. Unstructured data may be transformed into structured data thanks to advances in IE research, and an automated system can provide users with all key information rather than just a few relevant documents.
An automated system known as an information extraction system takes a sentence and extracts the key information then expresses it in a structured form that is readable for computers. The majority of information extraction systems extract binary relation tuples, which are made up of two entities and their relationships. Some IE systems also store some extra information like direction, context, and time. They are also known as n-ary tuples which an example of them is shown in table 1.1.

| Binary Relation | | N-ary Relation | |
|---|---|---|---|
| Tuple | Sentence | Tuple | Sentence |
| arg 1 | Omar Khayyam | arg 1 | Omar Khayyam |
| rel | was born | rel | was born |
| arg 2 | in Nishapur | arg 2 | in Nishapur |
| | | arg 3 | on May 18, 1048 |

TABLE 1.1: Binary VS N-ary Relation Tuple.

A great sample of a large database that contains binary relation tuples regarding actual entities is a **knowledge base (KB)**. Some of the most famous KBs are: Freebase [10], DBpedia [11] and Wikidata [12]. Many downstream natural language processing tasks, like question answering, benefit greatly

[1] https://en.wikipedia.org/wiki/Carnegie_Group
[2] https://en.wikipedia.org/wiki/DARPA
[3] https://en.wikipedia.org/wiki/Message_Understanding_Conference

from structured KBs because using knowledge bases will make it easy to answer the questions. In order to automatically fill up relevant information like infoboxes in search results, search engines employ knowledge bases. By checking the information from the infobox instead of reading the content, users will save a lot of time as a result. A graph is the most typical data structure for storing the data of a knowledge base. A graph's nodes stand in for the entities, and its directed edges connecting nodes stand in for the relationships.

Entity relation extraction was one of the key evaluation tasks in 1999's Automatic Content Extraction (ACE) evaluation, which was organized by the National Institute of Standards and Technology (NIST). The ACE entity relation corpus defines seven sorts of entities, including people, organizations, facilities, locations, geopolitical entities, vehicles, and weapons, each of which is also split into subcategories. ACE has been a part of the Text Analysis Conference (TAC) since 2009 and has grown to be an important part of the Knowledge Base Population (KBP).

## 1.2 Task Formulation

Relation extraction involves automatically extracting relation tuples from free texts. When given a sentence and a collection of relations, a relation extraction system produces a set of relation tuples that are contained in the sentence. Entity recognition and relation classification are the two sub-tasks that make up this task. In the first sub-task (entity recognition), entities are recognized and in the second sub-task (relation classification), we classify whether there is a relationship between each pair of entities or whether there is none at all. The method that we mentioned is the pipeline technique. A different strategy that is not a pipeline approach, looks for entities and relations jointly.

The relation extraction task can be classified into three groups based on the input: sentence level, document level, and corpus level.

**Sentence-level relation extraction** Identifying the relationships between entities in a sentence is the focus of sentence-level relation extraction. For instance, in Figure 1.2 we need to understand that Chicago and the United States have a *belongs to Country* relation.

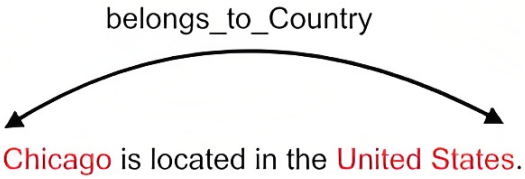


FIGURE 1.2: Sentence-level relation extraction. [13]

**Document-level relation extraction** Identifying the relationships between entities in a text or across multiple sentences is the objective of document-level relation extraction. In other words, we don't know where the targeted sentences are because the relational information about an entity pair is given in several continuous or split sentences.


FIGURE 1.3: Document-level relation extraction. [13]

**Corpus-level relation extraction** The goal of corpus-level relation extraction is to find relationships between two provided entities without taking their contexts into account.

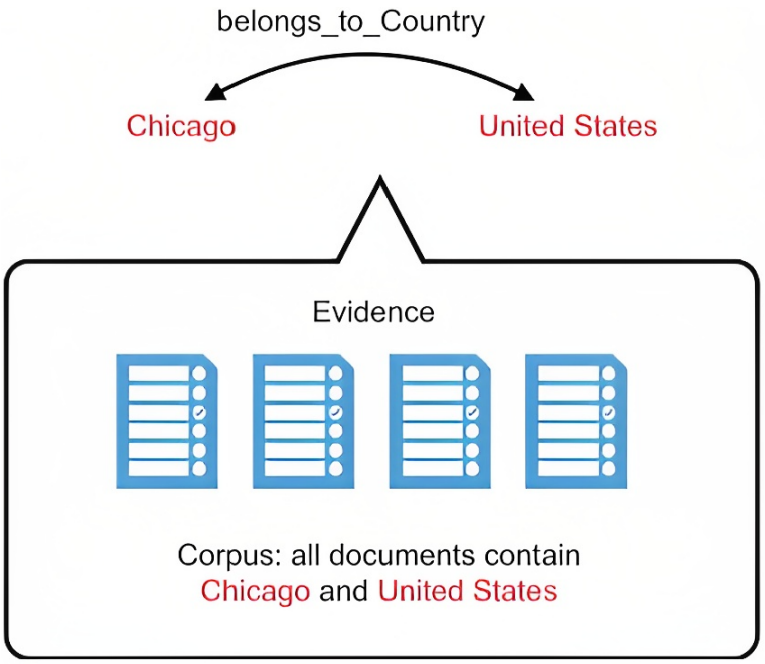


FIGURE 1.4: Corpus-level relation extraction. [13]

Furthermore, based on the definition of the relations, relation extraction might be categorized into pre-defined relation extraction and open relation extraction. As the name implies, all extracted relations in pre-defined relation extraction are pre-defined or known. So, it may be considered as a classification problem. The relationships in open relation extraction are unidentified. The objective of this task is to identify textual expressions that represent or suggest particular types of relationships.
In our work, we focus on sentence-level pre-defined relation extraction.

## 1.3 Scope of the Thesis

Our goal is to use deep neural network models to find the relation between entities in sentences. In our first work, we investigate a pipeline method in which we assume that two entities are provided, and we must determine their relationship or that no relationship exists between them. In our second work, we explore the impact of training data size on relation classification performance.

## 1.4 Contributions of the Thesis

The two parts of this thesis' contribution are as follows:
(1) We discover that sentences having an entity relationship can be long, and two entities can be positioned widely apart in a sentence. Furthermore, the evidence showing the existence of a relationship between two entities may not all contribute equally. To address this issue, we employ two different deep transformer models used for relation extraction introduced by Soares et al. [14] and Wu et al. [15]. By transferring knowledge from the pre-training step of the first model to the second model, we have enhanced the performance of the second model.
(2) Labeling and creating a training dataset in supervised learning is expensive. Thus, knowing how much labeled training data gives us acceptable performance is crucial. Furthermore, feeding more training data to the model does not affect the classifier's performance in a linear behavior. In other words, feeding more instances to achieve better performance may not be efficient since it may not increase the performance of the model enough. Therefore, we have investigated the impact of training data size on the performance of four different relation classification models to see how the results

of relation classification models change by increasing the size of the training dataset.

## 1.5 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, we briefly introduce neural networks, transfer learning, language model, knowledge bases, and existing relation extraction approaches. In Chapter 3, we cover related work on relation extraction. Chapter 4 describes how the deep transformer model can effectively represent relation statements and how we transferred this rich representation to classify relations. We describe our work on the impact of training data size on relation classification performance in Chapter 5. Experiments and results are covered in Chapter 6. Finally, we conclude the thesis in Chapter 7.

# Chapter 2

# Background

In this chapter, we gave some background details and definitions of the concepts related to this thesis's subject. These concepts include deep neural networks, transfer learning, language models, Knowledge bases, Named entity recognition, and Relation extraction approaches. Each of these concepts is explained comprehensively in the following.

## 2.1 Neural Networks

In this section, We go through the neural networks and algorithms used to model relation extraction task.

### 2.1.1 Feed-Forward Neural Networks

A subset of neural networks called **Feed-Forward Neural Networks (FFN)** only allows the flow of information in one way. These networks are often called **Multi-layer perceptron (MLP)**. FNNs are made up of three kinds of layers: an input layer, an output layer, and one or more hidden layers which are shown in Figure 2.1. Each hidden layer processes its input through a function before sending the results to the layer after it. A linear transformation and then a non-linear transformation is used to implement this function. The feed-forward network may approximate more complicated functions due to its nonlinearity. These non-linear functions, which are also applied in other neural nets that are detailed in the following sections, are commonly referred to as activation functions [16].

The widest and most important non-linear functions are sigmoid, ReLU, tanh, and softmax. Softmax is frequently utilized for output layer normalization. These non-linear functions with input x are defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

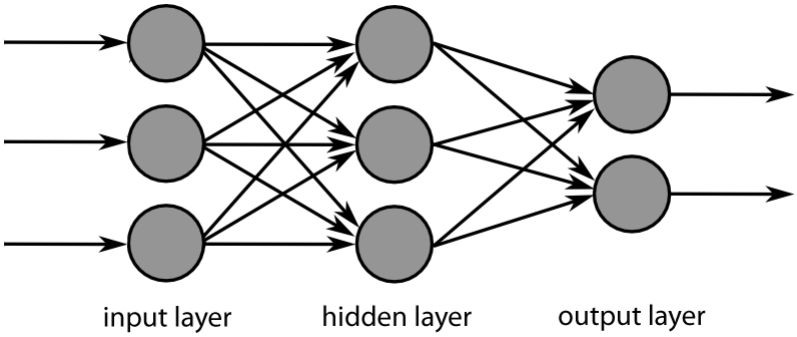


FIGURE 2.1: Structure of a feed-forward neural network. [17]

$$ReLU(x) = \max(0, x) \tag{2.2}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

$$softmax(x) = \frac{e^x}{\Sigma e^x} \tag{2.4}$$

With input x, an FFN with one hidden layer implements the following function:

$$FFN(x) = \rho(W_2(\rho(W_1 x + b_1)) + b_2) \tag{2.5}$$

W1 and b1 represent the trainable parameters for the hidden layer, while W2 and b2 represent the trainable parameters of the output layer. Any of the non-linear activation functions mentioned above is denoted by $\rho(.)$.

### 2.1.2 Convolutional Neural Networks

**Convolutional neural networks (CNNs)** [18] are a particular type of neural networks that operate on data with a grid-like structure, such as picture data which is a two-dimensional array of pixels shown in Figure 2.2 and text data which has a one-dimensional array of word vectors. CNN is used to automatically detect and extract essential features from data. For text data each token is represented by a vector, we have a sequence of vectors $\{x_1, x_2, x_3, ..., x_n\}$ where x $\in \mathbb{R}$ and n is the length of the sequence.

The convolution function for text data is defined by Eq. (2.6) and The concatenation operation is represented by $||$. **f** is a kd-length convolutional filter vector, where k is the filter width and superscript T denotes the transpose operation.

$$c_i = f^T(x_i||x_{i+1}||...||x_{i+k-1}) \tag{2.6}$$

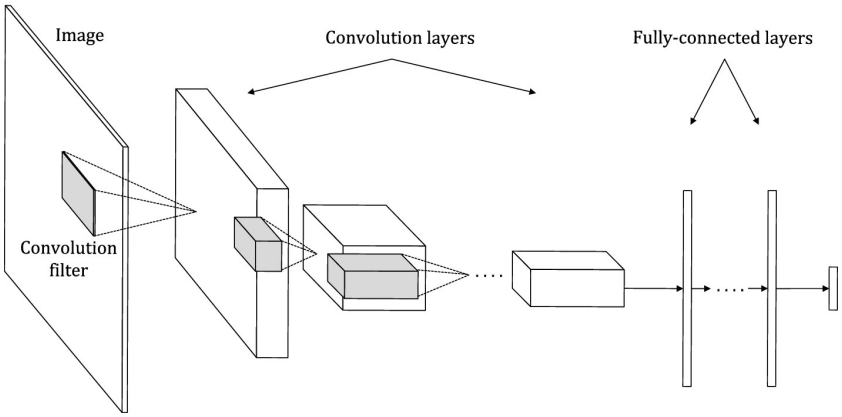


FIGURE 2.2: A typical architecture for CNN. [19]

Scalar values $\{c_1, c_2, ..., c_n\}$ are produced as the index $i$ shifts from 1 to n. The local aspects of the data are represented by these scalar values. A pooling operation, such as max-pooling (Eq. (2.7)) or average pooling (Eq. (2.8)) comes after convolutional operations.

$$c_{max} = \max(c_1, c_2, c_3, ..., c_n) \tag{2.7}$$

$$c_{avg} = \frac{c_1 + c_2 + c_3 + ... + c_n}{n} \tag{2.8}$$

The feature vector is created by concatenating the pooled results from various filters. $\mathbf{v}_{max}$ (Eq. (2.9)) and $\mathbf{v}_{avg}$ (Eq. (2.10)) are feature vectors with respect to $f_k$ as the number of filters.

$$\mathbf{v}_{max} = \max(c^1_{max}, c^2_{max}, c^3_{max}, ..., c^{f_k}_{max}) \tag{2.9}$$

$$\mathbf{v}_{avg} = \max(c^1_{avg}, c^2_{avg}, c^3_{avg}, ..., c^{f_k}_{avg}) \tag{2.10}$$

### 2.1.3 Recurrent Neural Networks

An exclusive class of neural networks called **recurrent neural networks (RNNs)** is used to analyze sequential input, like text. RNNs provide a feedback loop that aids in remembering previous information. The unwrapped form of an RNN is seen in Figure 2.3. With the exception of the fact that parameters in recurrent neural networks are shared across time steps, the unrolled form resembles an MLP network [16]. RNN comes in a variety of forms, but the following one is the most popular: [20].

$$h_t = \rho(W_h x_t + U_h h_{t-1} + b_h) \tag{2.11}$$

$$y_t = \rho(W_y h_t + b_y) \tag{2.12}$$

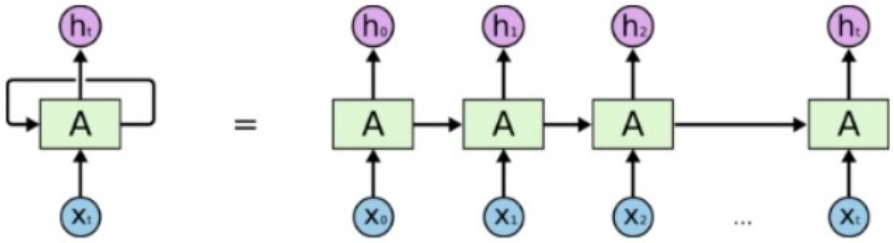


FIGURE 2.3: Structure of a recurrent neural network. [21]

where $x_t$ and $h_t$ are the input and hidden states at time step t, respectively, and $\rho(.)$ is any non-linear activation function. For this network, the hidden state $(h_t)$ is in charge of storing previous information. The network parameters $W_h$ , $U_h$ , $b_h$ , $W_y$ and $b_y$ are learned during training. The recurrent neural network faces the vanishing or exploding gradient problem over large sequences as a result of the repeated usage of the activation function on the hidden state of the RNN (Eq.(2.11)). By clipping the gradient up to a specific threshold, the exploding gradient problem could well be solved easily. The vanishing gradient issue, however, is more difficult to resolve. To solve the vanishing gradient problem for long sequences, Hochreiter and Schmidhuber [22] introduced **long short-term memory (LSTM)**, while Cho et al. [23] developed **gated recurrent unit (GRU)**.

### Long Short-Term Memory

By adding control gates to the network, Hochreiter, and Schmidhuber [22] were able to resolve the vanishing gradient problem in backpropagation. A gating mechanism is used by LSTMs to control the memorizing procedure. Through gates that open and close, information may be read, written, or stored in LSTMs. As a forget gate, input gate, and output gate, they made use of three control gates. The forget gate $(f_t)$ determines how much former information should be forgotten and the presence of the output is managed by the output gate $(o_t)$. For the RNN to successfully recall the previous information, they additionally included a cell state $(c_t)$. The Structure of LSTM is shown in Figure 2.4.
The forget gate determines what information must be remembered and what can be forgotten (Eq.(2.13)). The sigmoid function receives information from the current input $x_t$ and the hidden state $h_{t-1}$. The values that Sigmoid produces range from 0 to 1. It concludes whether the part of the old output is necessary by giving the output closer to 1. This value of $f_t$ will later be used

by the cell for point-by-point multiplication.
The amount of current information that must be held is managed by the input gate $(i_t)$. To update the cell state, the input gate does the following operations (Eq.(2.14)). First, the second sigmoid function receives two arguments: the current state $X_t$ and the previously hidden state $h_{t-1}$. Transformed values range from 0 (important) to 1 (not important). The tanh function will then get the same information from the hidden state and current state (Eq.(2.15)). The tanh operator will build a vector $(\tilde{c}_t)$ containing all possible values between -1 and 1 in order to regulate the network. The output values produced by the activation functions are prepared for point-by-point multiplication.
The input gate and forget gate have provided enough information to the network. Deciding and storing information from the new state in the cell state comes next (Eq.(2.16)). The forget vector $(f_t)$ get multiplied with the last cell state($c_{t-1}$). If the result is 0, then values will be removed from the cell state. Next, the network then updates the cell state by performing point-by-point addition on the output value of the input vector $i_t$, giving the network a new cell state $c_t$.
The operation of an LSTM network is described in the following equations.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.13}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.14}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.15}$$

$$c_t = i_t \circ \tilde{c}_t + f_t \circ c_{t-1} \tag{2.16}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.17}$$

$$h_t = o_t \circ tanh(c_t) \tag{2.18}$$

The information flow is controlled by the input gate $(i_t)$, forget gate $(f_t)$, and output gate $(o_t)$ using the sigmoid activation function. If this activation produces a 0 output, then no information will pass through, and a 1 output, then all information will pass through. Without any activation functions, the cell state $(c_t)$ is updated, which solves the issue of the gradient is very low.

### 2.1.4 Neural attention Networks

CNN and LSTM networks provide equal weight to each word in a text. However, for the majority of Natural language processing tasks, certain words

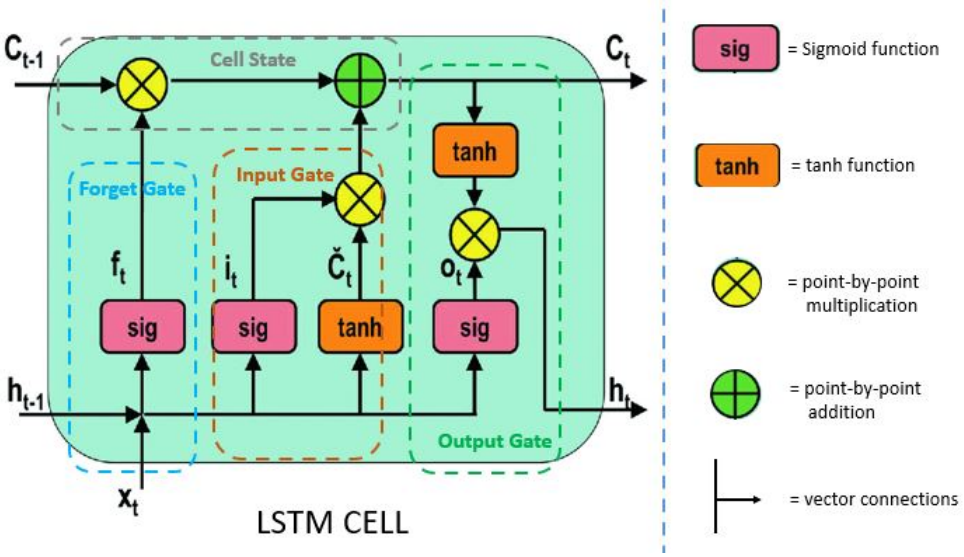


FIGURE 2.4: Structure of LSTM. [24]

contain more information than others. We want neural networks that can recognize which words are more essential than others on their own. These networks are known as attention networks. One or more attention layers, with trainable parameters in each layer, make up an attention network. Through the training process, the weights of parameters will be learned. All of the words are given normalized attention scores by an attention layer, some words gaining higher scores and others getting lower scores. So in the prediction process, keywords contribute more than other words. Such attention networks were utilized by Bahdanau et al.[25] for neural machine translation. Vaswani et al. [26] demonstrated that simple feed-forward neural networks with attention perform as well as LSTM networks.

### 2.1.5 Graph Convolutional Networks

Graph structure is a common format for real-world datasets. However, the CNNs and RNNs mainly perform on linear data. The generalized neural architectures known as **Graph Convolutional Networks (GCNs)** perform on any type of graph structure. A graph $\mathcal{G} = \{v, e\}$ is made up of a set of nodes V and a set of connecting edges E. An illustration of a multi-layer graph convolutional network is shown in Figure 2.5. Inputs for a GCN are as follows:

I A feature vector $x_i \in \mathbb{R}^d$ for every node i in graph $\mathcal{G}$. here d is the input feature vector's dimension. $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents the vectors of n nodes in graph $\mathcal{G}$.

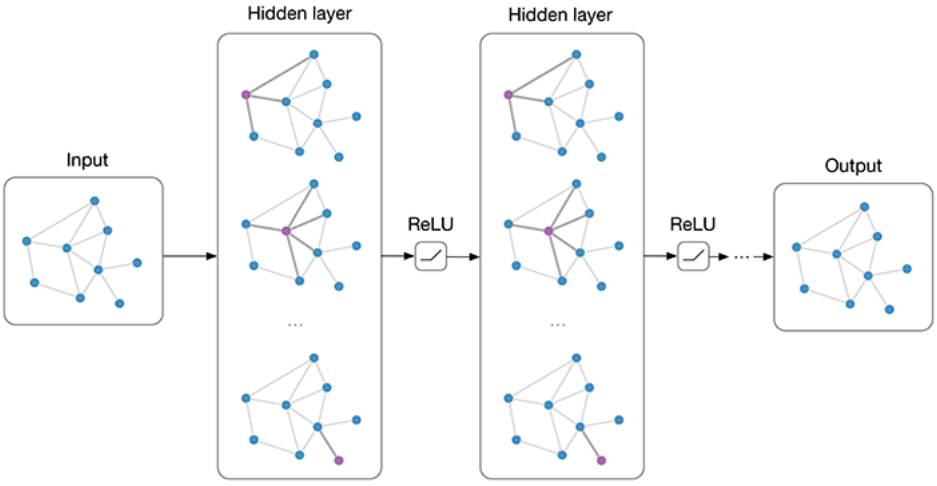


FIGURE 2.5: Structure of Multi-layer Graph Convolutional Network. [27]

II An n × n adjacency matrix **A** that defines the graph's structure. Self-loops are often added to **A**, and it is normalized.

The output of a GCN is node-level feature vector $\mathbf{Z} \in \mathbb{R}^{n \times f}$. Here $f$ is the output feature vector's dimension. Depending on the task, a pooling method can be employed to form the graph-level output from the node-level output. The function (E.q(2.19)) can be utilized to define each layer of a graph convolution network where $H^0 = X$ and $H^L = Z$. The number of GCN layers $i$ denoted by L. Parameterized function like any non-linear activation function is shown by $f()$. For example, in E.q(2.20) the activation function is sigmoid, and $W^l$ is the weight matrix of l-th layer in GCN.

$$H^{l+1} = f(H^l, A) \tag{2.19}$$

$$f(H^l, A) = p(AW^l H^l) \tag{2.20}$$

### 2.1.6 Neural Network Training

There are many parameters used by neural network models which initialized randomly, and during training, they change to approach their ideal values. Loss functions $C$ estimate the difference between the output of the network and the target or desired output. Minimizing this loss function is the aim of the training procedure. Using a gradient descent algorithm (E.q(2.21)), network parameters($\theta$) are tuned to reduce loss.

$$\theta_{t+1} = \theta_t - \eta \frac{\partial C}{\partial \theta_t} \tag{2.21}$$

Where $\eta$ is the step size, often known as the learning rate, which is a hyperparameter.
Batch training is the process of calculating the loss function C throughout the full training set. However, if the training data size is big, this is computationally quite costly. Alternatively, we can use another training algorithm which is called stochastic gradient descent (SGD). With this new algorithm, we can use each sample of the training set to calculate the loss function for each sample. Therefore, following each sample, we need to change the network parameters. But, for big training data sets, this will result in an extremely slow training process. Mini-batch training is a solution to address this problem in which the network parameters can be updated after each mini-batch of smaller sizes of training data. Each mini-batch chooses its instances at random and without replacement.
There are two fundamental issues with the stochastic gradient descent algorithm. First, the value of $\eta$ must be chosen by hand. The network may not converge if it is set very high, and if it is set too low, convergence will proceed extremely slowly. Second, the learning rate is constant across all parameters. However, to be adequately tuned, a neural network may require various learning rates for distinct parameters. Several adaptive optimization techniques, such as Adagrad [28] and Adam [29] are suggested to address these issues. During training, these optimization methods modify the learning rate for each parameter in a different way. Backpropagation is a crucial component of neural network training. Multiple layers make up neural networks, and each layer has certain parameters that will be updated according to the loss function. The chain rule of differentiation is used to accomplish this. The parameters of the output layer are changed first, followed by those of the layer under it, and so on. Up until all the layers' parameters are updated, this process (back-propagation) is repeated.

## 2.2 Language Model

Language models (LMs) are essential parts of NLP and applications that perform many tasks including sentiment analysis, audio-to-text, and speech recognition rely heavily on these models. LMs are designed to predict the probability of a pattern or sequence of words. Thus, NLP employs these models to understand the predictability of languages and phrases. Additionally, the development of transfer learning and effective language models removes the barriers that prevent machines from learning and interpreting

languages. Language models use word probability to analyze text. Additionally, they evaluate the data by processing it through an algorithm to include context-specific rules. Furthermore, using pre-trained language models, the rules are able to precisely predict and generate new phrases.

The field of natural language processing uses two different types of language models:

### 2.2.1 Statistical models

In order to make predictions for the following word in the sequence, statistical models construct probabilistic models. The model makes predictions based on words that came before. Examples of statistical models are the N-gram, Unigram, Bidirectional, and Exponential.

### 2.2.2 Neural Language Models

Language models developed in neural networks are known as neural language models. Neural network approaches outperform conventional methods, both when used as standalone language models and when included in larger models. The method's capacity to generalize may be a fundamental cause for the significant improvements in performance.
Some of the drawbacks of classic language models are addressed by non-linear neural network models: With just a linear increase in the number of parameters, they enable conditioning on ever larger context sizes.

#### The Generative Pre-Training Transformer

Recurrent neural networks are a good option for NLP tasks because of their sequential nature, but they have difficulties processing long-term data. With the advent of gating mechanisms, which regulate the input flow into and out of the hidden state of the RNN, the issue was lessened. Examples of these techniques are long short-term memory [22] and gated recurrent unit neural networks [23]. Despite these advancements, RNNs still have a tendency to concentrate on the latest inputs, losing crucial information that was gathered over a number of tokens earlier. For applications like machine translation, where models must pay attention to context-specific word occurrences in the input in order to translate them, this behavior is especially detrimental.
The Attention Mechanism [25], a new method that enables recurrent neural networks to focus on words over longer input, has recently shown significant

achievements. In the past, machine translation models included two parts: an encoder and a decoder. The encoder in the case of RNNs encodes the data from the source text into one vector representation. The translated words are then outputted one by one by the decoder using this compressed form of the input. By using the attention mechanism, the decoder includes every hidden state seen so far by using a weighted sum computed across all the previous hidden states based on their similarity with the translated sequence up to this point, in addition to processing the final hidden state of the encoder to generate the next word.

Vaswani et al. [26] presented a novel machine translation model that is fully based on attention and does away with recurrence in order to take advantage of the effectiveness of the attention mechanism. This model, referred to as the Transformer, uses a type of attention known as self-attention by connecting words in the input sequence to other words of the same input. Along with decoder-encoder attention, self-attention is used in the encoder and the decoder. The scaled dot-product and multi-head attention, which are also covered in this chapter, are two more attention extensions that Vaswani et al. [26] proposed. In contrast to RNNs, which are considered to take more training time compared to their sequential versions and are challenging to parallelize over batches of data, they were able to establish a new state-of-the-art in machine translation at a reduced training cost.
In Part 2.3.1 we will focus on one of the most important deep-transfer learning architectures for NLP called BERT.

## 2.3 Transfer Learning

This section explains the concept of transfer learning. We describe this concept inside the AI and ML fields while we take a peak at this concept outside the mentioned fields. After that, we will focus on applying transfer learning in NLP, especially in relation the extraction task.

Despite its remarkable success in practical applications, traditional machine learning technology still has some limitations for specific real-world scenarios. One of these limitations is the scarcity of training instances with similar distribution to test data. We noted a similar distribution because the ideal scenario for applying machine learning is that there are abundant labeled training instances, which have an identical distribution as the test data.

However, in real scenarios, collecting sufficient training data is often expensive, time-consuming, or in some cases, unrealistic and impossible thus, collecting sufficient unlabeled data is difficult.

Even if we collect sufficient unlabeled data, annotating these instances is also costly and sometimes difficult or impossible. Hence, Most datasets are unlabeled, and labeled datasets are a minority in the entire available datasets in the real world. Also, In some applications, if we build a large-scale, high-quality annotated dataset, this training dataset may quickly get out of date and thus cannot be effectively applied in the new tasks.

The collecting and annotating problems are both concerned with the scarcity of labeled and unlabeled training instances. Machine learning researchers refer more generally to the issues mentioned above as the small-data problem (or challenge) [30].

Over the years, machine learning researchers tried to find a solution for the small-data problem. These solutions are discussed in machine learning paradigms other than supervised learning which is suitable for scenarios with abundant labeled training instances.

Semi-supervised and unsupervised learning are other paradigms that help to deal with this problem. Semi-supervised paradigm is used for scenarios in which collecting and annotating the training instances is difficult despite being possible but the annotating process is expensive. Unsupervised learning is ideal for scenarios where the collection of sufficient unlabeled data is challenging while annotating the instances is impossible.

Also, a few design methodologies like Active, Multi-task, multi-view, and transfer learning address the small-data issue. Among these, transfer learning is one of the most promising design methodologies that has attracted the attention of researchers in recent years because this methodology can effectively alleviate the small data problem even better than other mentioned methodologies.

### 2.3.1 Transfer Learning Techniques in NLP

Through practicing and collecting more information through time, humans gain a natural comprehension of language. The capacity to distinguish between words' meanings based on their context, knowledge of word similarity, and identification of semantic relationships between words are all examples of this.
On the other hand, computers are not designed with any innate understanding of certain words. They see words as just letter combinations with no underlying meaning. Unsupervised pre-training of word representations, which capture both syntactic and semantic aspects of words, may be the most important component of recent advancements in natural language processing. To tackle nearly any NLP problem, these word representations are employed as the input in a wide range of various machine learning models.
Modern methods go one step further and pre-train entire complicated models on huge text corpora in an unsupervised way, such as the BERT [31] that is used in this study for relation extraction. These models are then transferred to a different domain and fine-tuned for the specific tasks to improve their performance. They can learn more complicated relationships between words and differentiate the meaning of words based on the context in which they appear, as opposed to fixed word representations.
This section provides a quick review of improvements in transfer learning for NLP.

#### One-Hot Encoding

One common way to represent words is via the well-known one-hot encoding, which maps each word in a vocabulary to a vector with the same length as the vocabulary and sets the value at the correct index to 1 and all other values to 0. This causes issues since words are viewed as unique entities with no shared properties. In the one-hot encoding, every word in the vocabulary is equally distant from every other word, even if words might really be more similar to one another due to specific features. In the d-dimensional encoding space, for instance, an important feature of word encodings is that vector representations of words with similar meanings should be near together, and vector representations of words with meanings that are distinct from one another should be far away. In the end, this improves the generalizability of a relation extraction model. For example, the meaning of "Wright brothers invented the Wright Flyer in 1903" and "Wright brothers designed the Wright

Flyer in 1903" are the same, and the semantic relationship between the two entities (Wright brothers and Wright Flyer) remains unchanged by the use of "designed" rather than "invented."
Word representations should essentially capture prior knowledge of words, which is difficult to gather on a small dataset that is utilized to train a machine learning model for a specific domain and target task.

### Fixed Word Embeddings

Word embeddings are a method that was developed to address the drawbacks of one-hot representations. Mikolov et al.[32] suggested the widely used Word2vec implementation. The process of transferring words into a d-dimensional feature space is known as embedding and often refers to real-valued feature vectors that also store word meaning. In this case, the elements of the d-dimensional embedding can be seen as weights that represent the percentage of a certain word feature.
Word2Vec has two different subtypes, which are CBOW and Skip-gram. The key principle of both is that words have an association if they show up in the same context. The language modeling objective used by CBOW and Skip-gram is different. The skip-gram method predicts the context word for a given target word. The training objective of CBOW is the opposite of the Skip-gram network: Compute the probability that each word in the vocabulary will be the target word based on the words around it. Figure 2.6 illustrated the architecture of both models.

The ability to be trained unsupervised is a benefit of Word2Vec and similar models like GloVe [34]. As a result, they are not dependent on a costly annotation procedure and can instead be trained on an unlabeled corpus, which is available on the internet.

### Context-Aware Word Embeddings

Despite the fact that fixed word embeddings like Word2Vec or GloVe significantly improved performance in a variety of NLP tasks, they are fixed and hence remain the same regardless of the domain and particular sentence context. By adapting pre-trained models like the Transformer to a different domain, we can take advantage of more context-aware information. This is particularly helpful for de-ambiguating words, which might mean different in different contexts. As an example, the word "bat" can be a mammal or a wooden club used in the sport of baseball depending on the context. In fact,

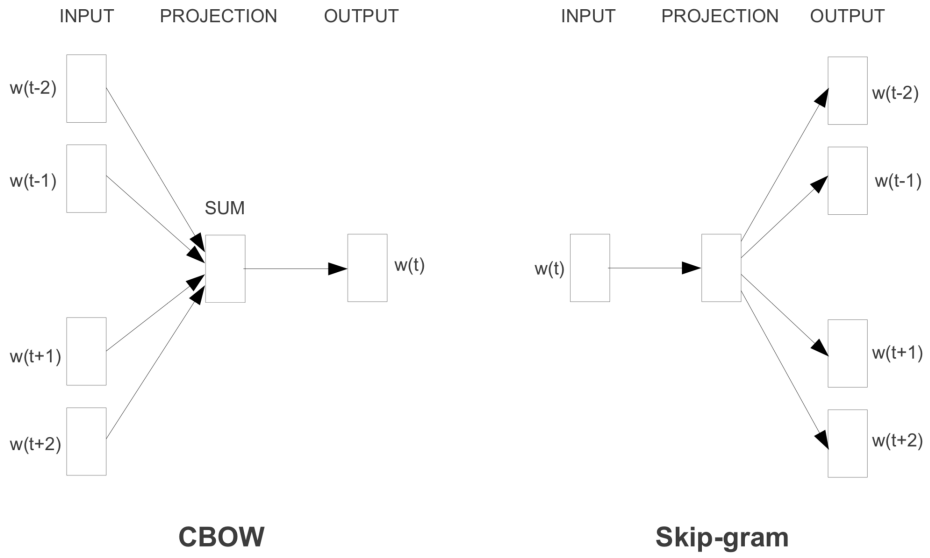


FIGURE 2.6: Network Architecture of CBOW and Skip-gram Word Embedding models. [33]

most of the words we use in our language have several meanings or senses depending on the situation. Furthermore, by capturing the information in the context, pre-trained models converge more easily, need fewer training samples, and eventually generalize better to new data [35]. To learn word embedding with respect to the context information, all models at first, are trained on a language model objective in an unsupervised way, then fine-tuned on a specific task.
Forward language modeling is an objective that is used in recent transfer learning models, which determines the probability of seeing a sequence of tokens $t_1, t_2, t_3, ..., t_l$ by computing the probability that each token $t_i$ will appear with regard to the tokens $t_1, t_2, t_3, ..., t_{i-1}$ that came before it:

$$P(t_1, t_2, t_3, ..., t_l) = \prod_{i=1}^{l} P(t_i | t_1, t_2, ..., t_{i-1}) \tag{2.22}$$

Where $l$ is the length of the sequence context.
In recent years, there has been increased interest in various language models that generate context-aware embeddings. Here is a quick explanation of two well-known models, ELMo [35] and BERT [31]:

**ELMo:** The term "Embeddings from Language Models" or ELMo refers to a bidirectional LSTM-based language model that has been pre-trained on a large text corpus. Due to the bidirectional nature of the model, token embeddings of the final layer of the model are context-aware based on the entire

input sequence. The sentence's forward and backward directions are processed by two LSTMs. The forward LSTM is trained to satisfy the forward language model objective as defined in Equation 2.22, whereas the backward LSTM is trained to satisfy the backward language model objective by calculating the probability of encountering token $t_i$ given the following token sequence $(t_{i+1}, t_{i+2}, ..., t_l)$.
Next, for each token in both layers, ELMo concatenates the hidden forward and backward states of the LSTMs. The

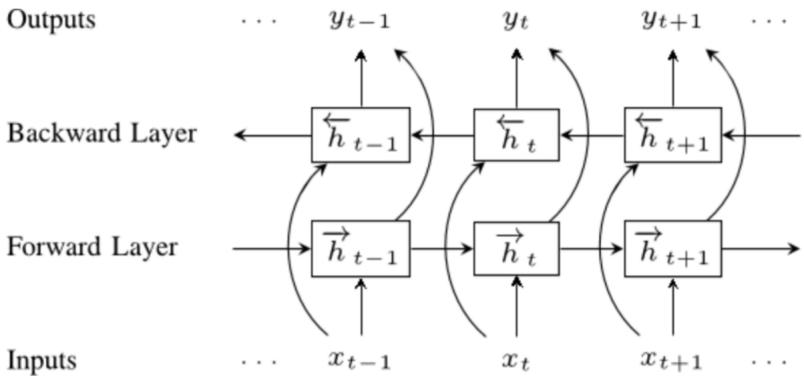


FIGURE 2.7: Network Architecture of Bidirectional Long Short-term Memory model. [35]

The last context-aware embedding of a token is created by learning a task-specific weighting of both the layers and the initial token representations, which are produced via character-based convolutional filters to map words that are unknown.

**BERT:** Bidirectional Encoder Representations from Transformers (BERT) [9] is similar in architecture to the Generative Pre-Training Transformer. It includes The transformer encoder part which is pre-trained on language modeling. Unlike the GP-Transformer, BERT is trained in a bidirectional way training the model on two unsupervised objectives: Masked language model (MLM) task and Next sentence prediction (NSP) task. In the masked language model task, a special Mask token is substituted for a set of randomly chosen context tokens, which the model must predict. Because the Mask token is not utilized during fine-tuning and the model must create embeddings for every position, selected tokens are randomly altered to other words or left untouched. The model learns context-aware embeddings for each input token with this change, according to the authors. For the sentence prediction task, The input context is separated into two parts. The second segment is substituted with a random segment from the corpus in 50% of the cases. The

objective of NSP is to determine whether the second segment will come after the first. Both tasks have been pre-trained using a mix of BooksCorpus [36] and English Wikipedia.
The authors demonstrate state-of-the-art performance in 11 natural language processing tasks, outperforming models like the ELMo and unidirectional GP-Transformer.

## 2.4 Knowledge Bases

**Knowledge bases (KBs)** that encompass entities in various domains have been developed recently by a variety of research teams. Here are a few examples of KBs such as YAGO [37], Freebase [10], Wikidata [12], DBpedia [11], and NELL [38].

**YAGO** The creation of Yet Another Great Ontology (YAGO)involves the extraction of information from the structured data found in WordNet and Wikipedia. In order to extract classes, entities, and relations between them, YAGO leverages Wikipedia category articles. Yet, Wikipedia category articles are ineffective for developing an ontology. contrastingly, WordNet offers a class hierarchy that YAGO uses to construct an ontology. To develop the ontology and retrieve facts, YAGO effectively integrates Wikipedia and WordNet.

**Freebase** Metaweb Technologies Inc. created Freebase, a structured knowledge base for relation tuples. When it was first released in 2007, Freebase contained 125 million entities and around 7,000 relations. Google bought Freebase in 2010 and shut it down in 2016 after preferring to move all of its data to Wikidata.

**Wikidata** Another large-scale, open-source, and collaborative knowledge base created by Wikimedia is Wikidata. Wikidata contains the relevant Wikipedia source page with the facts, so users may check its accuracy. Additionally, each relation's aliases and a short explanation are provided. It keeps records of facts as items and statements. A Wikidata item represents an entity, and each item contains a number of statements. Each statement has a claim that includes a property and its value. These properties reflect relations.

**DBpedia** Another project that pulls structured data from Wikidata's multilingual content is DBpedia. Raw infobox extraction and mapping-based infobox extraction are the two forms in which information is extracted from Wikidata. In raw infobox extraction, Wikidata infobox information is extracted by DBpedia in raw form without being mapped to an ontology. In mapping-based extraction, a community-generated ontology is constructed, and Wikidata infobox extractions are mapped to that ontology.

**NELL** Another project that automatically pulls facts from the Web is called Never-Ending Language Learner (NELL). NELL's creators developed it to be able to recognize a core set of semantic relations between a few hundred predefined types of data. The initial ontology for NELL, a semi-supervised system, had hundreds of categories and relations. Learning new examples of categories and relations is the aim of this system. This system is always picking up new facts and using them to enhance its learning algorithms for better quality extraction.

## 2.5 Named Entity Recognition

An essential NLP task is **named entity recognition (NER)**, which supports several other NLP applications including question-answering, information retrieval, and relation extraction. Named entity recognition is the initial step for pipeline relation extraction. Using NER in a text will give us name entities like person names and place names. With associated datasets like MUC [39], CoNLL 2002 [40], CoNLL 2003 [41] and ACE04 [42], many shared named entity recognition projects were developed.
For the NER problem, supervised learning models are frequently applied. NER on MUC-6 datasets was performed by Zhou and Su [43] using the Hidden Markov Model (HMM). For the same problem, Malouf [44] applied the HMM with maximum entropy. Carreras et al. [45] employed a binary AdaBoost classifier with features including capitalization, trigger words, and gazetteers. For the NER task, Takeuchi et al. [46] applied an SVM model. Chieu and Ng [47] suggested a maximum entropy solution for the NER challenge in which they combined sentence-level information with document-level information. They did their studies using the MUC-6 and MUC-7 datasets and also they applied the same method for the CoNLL 2003 NER shared task [48].

### 2.5.1 Neural NER Models

One of the earliest neural models for the NER problem was proposed by Collobert and Weston [49]. In their model, they treated the various NER features as vectors. Later, word vectors were used in place of these manually derived features [50]. For this challenge, Huang [51] used a Bidirectional LSTM-based neural network with word embeddings. For the task of sequence labeling, Ma and Hovy [52] and Chiu and Nichols [53] employed character-based embeddings in addition to word embeddings. Their investigations on the CoNLL 2003 dataset reveal that character-level embeddings aid in task performance. Lample et al. [54] employed an LSTM-CRF model for the NER task as well. For this task, deep contextualized word representations such as **Embeddings from Language Models (ELMo)** [35] and **Bidirectional Encoder Representations from Transformers (BERT)** [9] have proved to be particularly effective. Strakova [55] used models with these contextualized word representations on the CoNLL 2003 dataset and got state-of-the-art performance.

## 2.6 Relation Extraction

Methods for extracting relationships are generally categorized into traditional machine learning-based methods and neural network-based deep learning methods. There are several traditional methods that rely on machine learning, which can be categorized into rule matching, supervised learning, weakly supervised learning, and unsupervised learning. There are two types of deep learning methods based on neural networks: supervised learning and distant supervision. Supervised learning typically utilizes pipeline and joint learning approaches. The classifications and methods of relation extraction are explained in Figures 2.8 and 2.9.

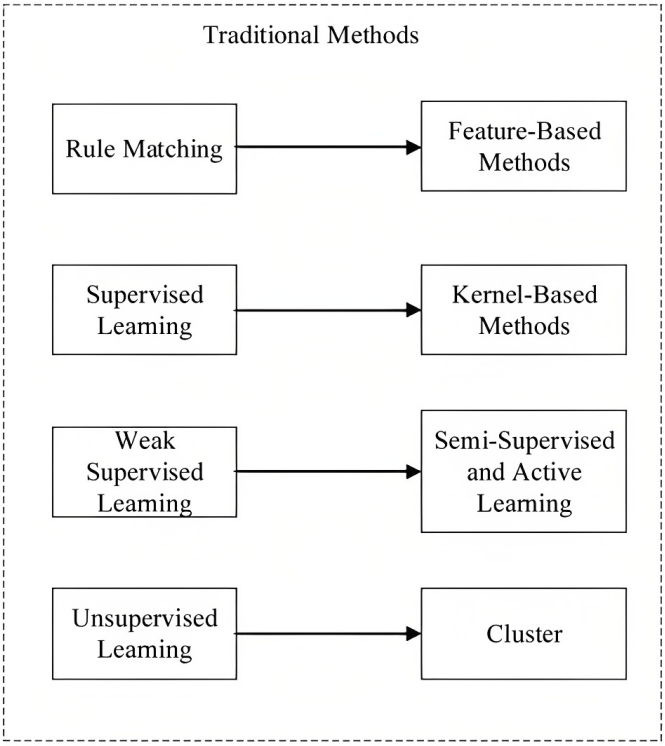


FIGURE 2.8: The Traditional methods used for extracting relationships. [56]

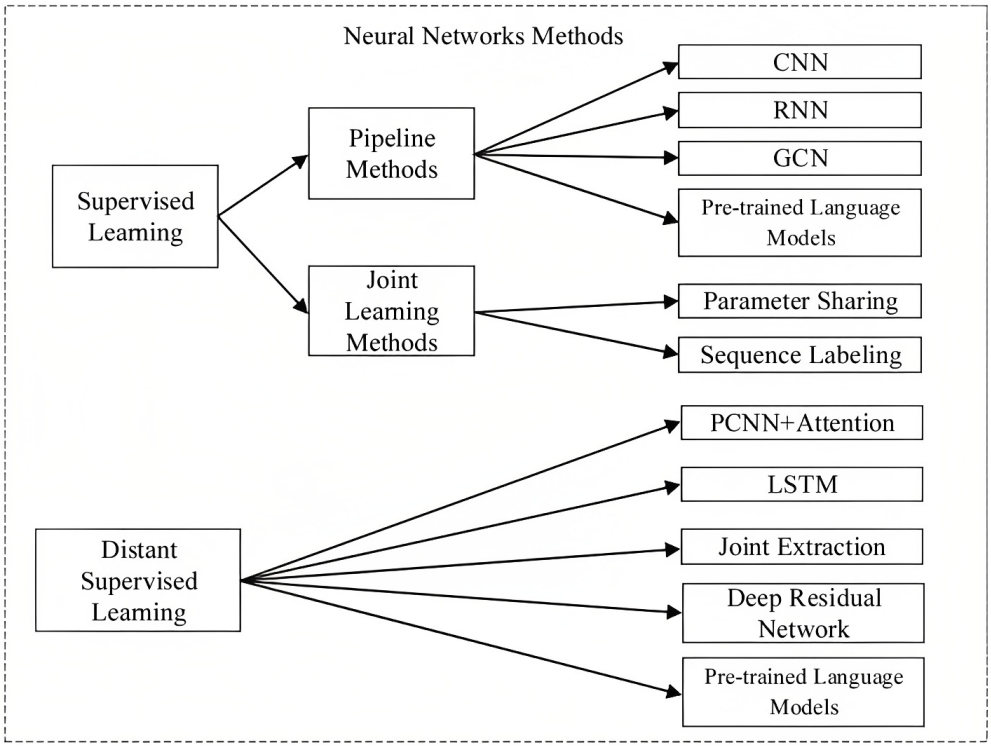


FIGURE 2.9: The Neural Network methods used for extracting relationships. [56]

### 2.6.1 Hand-built pattern methods

For hand-built pattern methods to be successful, domain specialists and linguists must work together to create a knowledge set of patterns based on words, part-of-speech (POSs), or semantics. By matching the preprocessed language fragment with the patterns, relation extraction can be achieved with the use of this linguistic knowledge and professional domain knowledge. If they match, it could be said that the statement has the relationship of the matching pattern. An example of Hand-built pattern methods considering the pattern "*such Y as X*" is illustrated in Table 2.1

| | |
|---|---|
| Pattern | such Y as X |
| Corpus | ... published by such authors as Miller, Johnson, and Shakespeare. |
| Relation | Hyponym (author, Miller), Hyponym (author, Johnson), Hyponym (author, Shakespeare) |

TABLE 2.1: Example of Hand-built pattern methods for Relation Extraction .

### 2.6.2 Semi-supervised methods

Semi-supervised methods are based on patterns, just like pattern-based methods, and use a bootstrapping algorithm additionally. The objective behind this approach is to first identify certain seed tuples with a high degree of confidence. Then, the bootstrapping algorithm extracts patterns from a large amount of unlabeled corpus using the tuples as its input. The procedure of extracting new triples using these patterns can be seen in Figure 2.10.

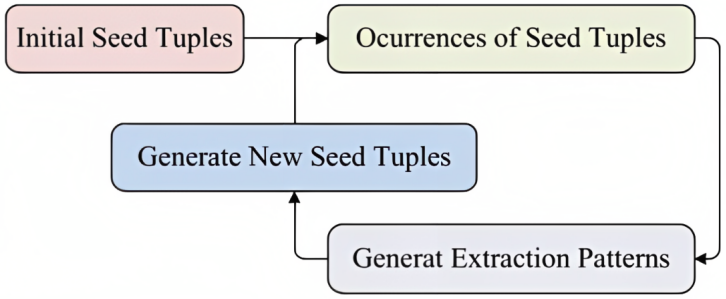


FIGURE 2.10: Semi-supervised relation extraction procedure. [57]

### 2.6.3 Supervised Methods

Relation extraction is regarded as a multi-class classification issue in supervised approaches. Feature-based and kernel-based methods are two types of

traditional supervised relation extraction approaches. In feature-based methods, a classifier is trained using each related instance in the labeled data and then fed subsequently new samples for classification. These features are often derived from valuable information (such as lexical, syntactic, and semantic information) collected from an instance context. The performance of a feature-based approach cannot be improved without appropriate feature selection.
The kernel-based methods are less dependent on explicit language preprocessing procedures than feature-based methods are. How to develop an efficient kernel becomes the crucial stage in this method.
Relation extraction has so far been done using supervised deep neural network-based methods. These DNN-based methods (such as CNN, RNN, or LSTM) can automatically learn features rather than features that are manually developed using different NLP toolkits. The majority of them have significantly outperformed the traditional methods.

**Pretrained Language Model for Supervised Relation Extraction** In comparison to earlier CNN-based or RNN-based supervised learning relation extraction approaches, pre-trained language models, such as BERT, have more hidden layers and can capture longer-distance information. By modifying BERT's coding strategy, input-output structure, merging it with other models, and employing knowledge base information, researchers have significantly increased the accuracy of the relation extraction task. Based on deep transformers' pre-trained language models, table 2.3 lists multiple supervised learning relation extraction methods.

| Model | The key aspects |
|---|---|
| R-BERT | In order to improve the BERT model's ability to capture entities, the model annotates the start and end of entities with $ and #. |
| $\text{BERT}_{EM}$ + MTB | MTB increases the RE's[1] accuracy. BERT performs even better when input and output are provided by *ENEITYMARKER* and *ENTITYSTART*, respectively. |
| EPGNN | In order to improve the BERT model's ability to capture entities, the model annotates the start and end of entities with $ and #. The entity pair graph's topological structure is extracted by the model using GCN [2], and it is merged with the sentence's semantic information to even further enhance the accuracy of RE. |
| Know-BertW+W | By the integration of KBs [3] into BERT through KAR, the Know-Bert model is able to integrate information from external KBs and hence increase the accuracy of RE. |
| Entity-Aware BERT | MER [4] extraction is handled by the model using One-Pass encoding. The accuracy of MER extraction is increased by the ability of Entity-Aware and Self-Attention methods to infuse relation information for multiple entities in every hidden state. |
| TRE | The input sequence is encoded using multiple layers of Transformer in the model. BPE[5] encodes the input text, enhancing the model's ability to recognize hidden semantic features. |
| ERNIE | In order to extract structured knowledge from the KG[6], the model uses KG information during the training step of the multi-layer Transformer model. This enhances the accuracy of RE. |
| BERT-LSTMbase | The model's robustness is increased when BERT and BiLSTM are used together. |
| SpERT | It is possible to extract overlapping entities relying on the Span-Based Joint Learning Model with BERT as its backbone. Following the BERT layer's encoding of the texts, the Span Classification and Span Filtering layers identify any overlapping entities and the Relation Classification layer determines the relationship between the entities. |
| DYGIE++ | By using spanning enumeration, span graph propagation, and BERT sentence encoding, the model can handle a wide range of tasks. |

TABLE 2.2: Comparisons of different supervised relation extraction models using BERT as Pre-trained Language Model.

[1] Relation Extraction
[2] Graph Convolution Network
[3] Knowledge Base
[4] Multiple Entity-Relations
[5] Byte Pair Encoding
[6] Knowledge Graph

| Model | Method | Use External Knowledge | Dataset | F1 Score |
|---|---|---|---|---|
| R-BERT | BERT | No | SemEval-2010 Task 8 | 89.25 |
| $BERT_{EM}$ + MTB | BERT + Matching The Blanks | English Wikipedia | SemEval-2010 Task 8 | 89.5 |
| | | | TACRED | 71.5 |
| | | | KBP37 | 69.3 |
| EPGNN | BERT + GCN | No | SemEval-2010 Task 8 | 90.2 |
| | | | ACE2005 | 77.1 |
| KnowBertW+W | BERT + KAR | Wikipedia and WordNet | SemEval-2010 Task 8 | 89.1 |
| | | | TACRED | 71.5 |
| Entity-Aware BERT | BERT + Entity-Aware + Self-Attention | No | SemEval-2018 Task 7 | 83.9 |
| | | | SemEval-2018 Task 8 | 89.0 |
| TRE | Pre-trained Transformer + Byte Pair Encoding | No | SemEval-2010 Task 8 | 87.1 |
| | | | TACRED | 67.4 |
| ERNIE | BERT + KG Embedding + TransE | Knowledge Graph | FewRel | 88.32 |
| | | | TACRED | 67.97 |
| BERT-LSTMbase | BERT + BiLSTM + WordPiece tokenizer | No | TACRED | 67.8 |
| SpERT | BERT + Span Classification+ Span filtering | No | ADE | 78.84 |
| | | No | CoNLL04 | 71.47 |
| | | SciBERT [] | SciERC | 50.84 |
| DYGIE++ | BERT + Span Enumeration + Span Graph Propagation | No | ACE2005 | 63.4 |
| | | | SciERC | 48.4 |
| | | | WLPC | 65.9 |

TABLE 2.3: Comparison of several relation extraction methods for supervised learning using Pre-trained Language Model.

### 2.6.4 Unsupervised Method

The unsupervised approach for information extraction is based on a bottom-up technique. It works on the concept that entity pairs with the same semantic relationship share similar context information. Hasegawa et al. [58] proposed an unsupervised approach earlier. The extraction process can be broken down into three steps: The first step is to extract the entity pair and its context. The second step is to cluster the entity pair based on the context. The third step is to annotate the semantic relation of classes or define the relation type.

#### Open Information Extraction

Open information extraction (OpenIE) is a general method for extracting relations from unstructured text, and it may extract any type of relation. Most systems apply hand-crafted rules and predefined structures to extract entities and relations from sentences using OpenIE. The benefit of this method is that it may be used with texts from any domain. Some OpenIE systems that can extract information from the free text include: KnowItAll [59], TEXTRUNNER [60], REVERB [61], SRL-IE [62], OLLIE [63], and RELNOUN [64].

**KnowItAll** This OpenIE rule-based system extracts information from the Web using hand-crafted patterns. Furthermore, it gives the extracted tuples a confidence score based on the point-wise mutual information between words connected to the recognized entity or triple and predefined phrases for each entity or relation.

**TEXTRUNNER** This system is based on the KnowItAll idea but does not need to be given hand-made patterns. It has a self-supervised learner that labels the training data as positive or negative instances by using dependency parse trees. It identifies the relations between the noun phrases in a dependency parse tree marked as arguments. It classifies a tuple as positive or negative based on specific grammatical features, such as the length of the dependency chain between two arguments and the path between two arguments. The next step is to train a naïve Bayes classifier on these samples to assess the reliability of upcoming tuples.

**REVERB** The TEXTRUNNER system has been upgraded by REVERB. In order to eliminate meaningless and nonsensical extractions, it applies additional syntactic and lexical restrictions.

**SRL-IE** A semantic role labeling (SRL) system developed at UIUC [65] is used for this open information extraction system. A typical NLP task is to identify the semantic arguments that are connected to a predicate in a sentence and then classify those arguments into various semantic roles, such as agent, patient, and instrument. An SRL system's extracted predicates and arguments can be called relation tuples.

**OLLIE** REVERB exclusively identifies relationships based on verbs. However, certain relations are founded on noun phrases instead of verbs. This problem of REVERB is fixed by Open Language Learning for Information Extraction (OLLIE). Starting with tuples retrieved by REVERB, OLLIE maps these tuples to the sentences after gathering sentences from the Web. OLLIE attempts to produce general patterns for each relation using the dependency parse tree of these sentences. In order to extract additional tuples from texts, OLLIE uses the learned patterns.

**RELNOUN** In order to extract relation tuples based on nouns rather than verbs, RELNOUN is an open relation extraction module for this task. This system uses titles and entity properties to extract tuples. To extract noun-based tuples, RELNOUN employs part-of-speech tags and chunk patterns.

#### Limitations of OpenIE

Despite being able to extract a significant number of tuples from unstructured texts, open information extraction algorithms have two serious drawbacks:

(i) They gather a sizable amount of useless tuples. Every verb is viewed as a possible relation; therefore, there will be a huge number of tuples that are not informative. It is difficult to get rid of useless tuples.

(ii) The relations are not made normal. Open IE systems view each verb as a special connection. However, The same relation may be expressed using a variety of verbs. In open IE systems, various verbs with the same meaning are not combined into a single connection.

### 2.6.5 Distant Supervision Method

Manual development is required to create existing knowledge bases like Freebase, Wikidata, and DBpedia, and it takes a lot of time and work. These KBs still miss a lot of relations. If we could be able to automatically extract relation tuples from raw text, we would be able to create a KB from the beginning or add new tuples to the existing knowledge bases without having to do any human work. To do this, we require a sizable collection of texts that have been annotated with relations, each linking two entities. However, manually building such a corpus is a difficult operation.
The concept of distant supervision was set out by Mintz et al. [2], Riedel et al. [66], and Hoffmann et al. [67] to automatically build such text-tuple mapping without the need for human intervention. In distant supervision, there is a mapping process from the tuple of a knowledge base to a raw or free text corpus like Wikipedia or news articles (for example, the New York Times). According to the principle of distant supervision, a sentence can be regarded as the source of a tuple from a KB if it includes two entities of a tuple from that knowledge base. The procedure of the distant supervision method is shown in Figure 2.11

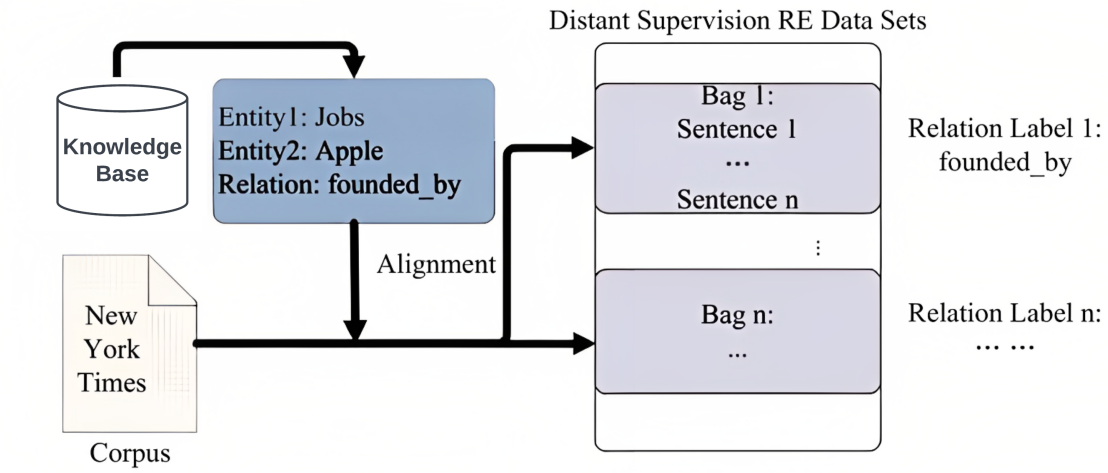


FIGURE 2.11: Distant supervision procedure. The knowledge base is in the top left corner of the picture, and the corpus source is in the lower left. The right side creates the packages matching the KB to represent various relationships after the text alignment procedure in the middle. These packages include many sentence occurrences, and each package is a relationship label. [57]

In contrast, a sentence is regarded as an instance of a *None* tuple between

two entities if it includes two entities from a KB but there is no relation between them in the KB. These *None* instances are important since distantly supervised models only take into account a small number of positive relationships. Any relation that is not a part of this set is regarded as a *None* relation. We can create supervised models for this task using a large number of the tuple-to-text mappings produced by this approach.
However, there may be a lot of noisy instances in the distantly supervised data. This could happen when the two entities of a positive tuple may occasionally appear in a sentence, but the sentence may not mention their relationship. Sentences and entity pairs of this type are referred to as noisy positive samples. Another set of noisy instances is produced using the method used to create instances for the *None* relation. Because in the distant supervision approach when two KB entities are present in a sentence but there is no relation between them, the sentence and entity pair are taken as an example of a *None* relation. However, knowledge bases are usually incomplete and lack legitimate relations between entities. Therefore, it's probable that the sentence provides details about a positive relationship between the two entities, but because that relation isn't included in the KB, this sentence and entity pair are mistakenly taken into account as an example of a *None* relation. Noisy negative samples are sentences and entity pairs of this type.
In spite of the existence of noisy instances, relation extraction models which are trained on distantly supervised data have shown to be effective for relation extraction task. By automatically extracting tuples from free texts, these models may be utilized to complete a knowledge base's missing information. In order to complete an existing KB, it can save a lot of manual work.

#### Noise Reduction for Distantly Supervised Data

The performance of models is negatively impacted by the existence of noisy samples in distantly supervised data. To reduce the impact of noisy data and strengthen their models, researchers have tried a variety of strategies. One of the famous strategies for noise reduction is multi-instance relation extraction. This multi-instance learning idea was applied in the relation extraction models suggested by Riedel et al. [66], Hoffmann et al. [67], Surdeanu et al. [68], Lin et al. [69], Yaghoobzadeh et al. [70], Vashishth et al. [71], Wu et al. [72], and Ye and Ling [73]. They utilized all the sentences that contained these two entities in order to determine the relation between each entity pair. They used this multi-instance setup, trying to minimize the effect of noisy samples. In order to prioritize the sentences that included relation-specific

keywords and disregard the noisy sentences, they applied several forms of sentence selection algorithms. The multi-task learning strategy was utilized by Ren et al. [74] and Yaghoobzadeh et al. [70] to reduce the impact of noisy samples. They added a second task to their model called fine-grained entity typing.
For the same goal, Wu et al. [72] employed an adversarial training strategy. To improve the model for distantly supervised training, noise was added to the word embeddings. A **generative adversarial network (GAN)** was employed by Qin et al. [75] to solve the problem of noisy sentences in relation extraction. For each class of positive relations, they utilized a separate binary classifier as a generator in their model to distinguish between the true positives and the noisy ones. Reinforcement learning was applied by Qin et al. [76] to find the noisy data for the positive relation classes. The noisy samples that were associated with the positive relations were found using reinforcement learning, and the identified noisy samples were later employed as unlabeled data in the model by He et al. [77]. To find the noisy samples, Shang et al. [78] employed a clustering strategy. These noisy samples were given the correct relation label, and they were included in their model's training data.

### 2.6.6 Zero-Shot or Few-Shot Relation Extraction

There are thousands of relations in existing knowledge bases like Freebase, Wikidata, and DBpedia, but only a limited portion of the relations from these KBs are covered by distantly supervised datasets. Distant supervision is unable to find sufficient training examples for the majority of relations in KBs because of the mismatch between the surface form of entities in KBs and texts. In other words, distant supervision methods are not able to cover the missing connections belonging to uncovered relations. This issue can be solved via zero-shot or few-shot relation extraction. These models have the ability to be trained on a set of relations and then used to infer a different set of relations.
Levy et al. [79] and Li et al. [80] applied the reading comprehension technique for zero-shot relation extraction while converting the relation extraction task to a question-answering task. According to this method, entity 1 and the relation serve as questions, while entity 2 serves as the answer. Answer: NIL if entity 2 doesn't exist. For this task, Levy et al. [79] used

the WikiReading [81] dataset with additional negative samples and the **Bi-directional Attention Flow (BiDAF)** model [82] with an additional NIL node in the output layer. During training, they utilized one set of relations, and during testing, they used another set of relations. Entity 1 and the relation were utilized as templates by Li et al. [80] to construct the question. To discover various answers to a query, they converted the machine-reading comprehension models to a sequence tagging model. The zero-shot relation extraction can also be done using this method, even if they did not test it out in that situation. FewRel 2.0 [83] is a dataset for few-shot relation extraction.

### 2.6.7 Datasets

In this part, we provide a quick overview of the relation extraction datasets that are currently accessible. In each chapter of this thesis, we go into detail about the datasets that we employed in our research.

In SemEval-2010 Task 8, Hendrickx et al. [84] offered a shared challenge on relation extraction and made available a dataset containing 8,000 training sentences and 2,717 test examples covering nine bi-directional relations and an *Other* relation. This dataset's relationships are not derived from any knowledge base. In the sentence, they express the relation between two nominals. Such relationships include Cause-Effect, Component-Whole, and so on.

ACE04 [42], CoNLL04 [85], and GDS [86] are three further datasets with 7, 5, and 4 relations, respectively. These datasets may not be appropriate for creating large-scale models since they include a limited number of relations and training samples.

The **TAC Relation Extraction Dataset (TACRED)** is recently developed as a large-scale relation extraction dataset with 106264 samples which was created by using corpus from newswire and web texts in the TAC Knowledge Base Population (TAC KBP) challenges [87]. Examples in TACRED include 41 relation classes that are utilized in TAC KBP challenges (for instance, *per:city-of-death* and *org:founded-by*), or they are tagged as *no relation* if there is no predefined relation between entities. The examples were produced using a combination of crowdsourcing and the human annotations that were available from the TAC KBP challenges.

All of these mentioned datasets are built at the sentence level. There are some document-level datasets for document-level relation extraction that we are not going to cover in this thesis.

# Chapter 3

# Related Work

We discuss the various works related to relation extraction in this chapter. Based on performing named entity recognition before extracting relations or extracting both name entities and relations simultaneously, we divided related work into pipeline extraction approaches and joint extraction approaches.

Closed domain relation extraction can be used to overcome the drawbacks of open IE systems which led to uninformative and unnormalized relations. For learning a model to perform the relation extraction task, we will apply supervised algorithms with a collection of pre-defined relations. Below, we go over a summary of previous studies on relation extraction as well as the datasets that are readily accessible.

## 3.1 Pipeline Extraction Approaches

The use of pipeline extraction approaches was very common in the early stages of the information extraction study. The two phases of a pipeline approach are:

(i) The named entities in a text are first recognized using a named entity recognizer.

(ii) Then, a classification model is applied to determine the relation between each pair of entities.

A significant number of text-tuple pairs may be produced via distant supervision, and these pairs can then be utilized to build supervised learning models for RE task.

### 3.1.1 Feature-Based Models

For RE task, a feature-based approach was presented by Mintz et al. [2]. Features they applied in their model were various, including: **lexical features** like the sequence of words between to entities and their part-of-speech tags, a label showing the first entity, k tokens of the left of the first entity, k tokens of the right of the second entity, **syntactic features** like named entity types and dependency path between two entities. Multi-instance learning was introduced by Riedel et al. [66] in order to address the issue of noisy sentences produced by the distant supervision technique. The idea to solve this problem was using a graph factor which models the decision of First, are two entities are related? and Second, is this relation expressed in the given sentence? Additionally, without knowing which sentences represent the relations, they trained their model using constraint-driven semi-supervision. In comparison to the Mintz et al. model [2], their multi-instance learning approach performs much better. To address the issue of overlapping relations, Hoffmann et al. [67] and Surdeanu et al. [68] developed the concept of multi-instance multi-relations. They employed probabilistic models, which look for all potential relations between pairs of entities in a bag of sentences.

### 3.1.2 CNN-Based Neural Models

Convolutional neural networks were employed by Zeng et al. [5] to deal with the RE task. For representing the tokens in sentences, they applied Turian et al.[88] pre-trained word embeddings, for positional embedding they applied two distance embedding vectors. To extract a sentence-level feature vector, they employed a convolutional neural network and max-pooling operation. Max-pooling operates through the entire sentence to extract the single most important feature from the whole sentence for a specific convolutional filter. To classify the relation, this sentence representation is fed into a feed-forward neural network with a softmax activation layer.

To make the relation extraction classifier performs better, Zeng et al. [6] developed the **piecewise convolutional neural network (PCNN)**. The max-pooling operation is not applied throughout the full sentence in PCNN. Instead, the sentence is broken into these three parts: from the beginning of the sentence to the first entity, from the first entity to the second entity, and from the second entity to the end of the sentence. For each of these three segments and each convolutional filter, max-pooling is used to produce three

feature values. Concatenating all of these feature values yields a sentence-level feature vector, which is then fed into a feed-forward neural network with a softmax activation layer to classify the relation.

Santos et al. [7] utilized a convolutional neural network to solve the relation classification task, which involves classification by ranking (CR-CNN). The network creates a distributed vector representation of the text using a convolutional layer and compares it to the class representations to provide a score for each class. In order to reduce the impact of artificial class (a class that is being used to group instances that do not belong to any of the actual classes), they created a pairwise ranking loss function. The main advantage of this approach is that the learning process concentrates exclusively on the "natural" classes by removing the embedding of artificial classes. So, CR-CNN does not produce a score for the artificial class and it will not have an impact on the prediction phase.

### 3.1.3 RNN-Based Neural Models

Since relevant information might exist anywhere in a sentence, which makes relation extraction difficult, Zhang et al. [8] suggested using bidirectional long short-term memory networks (BLSTM) to model the sentence with sequential information about all words. The authors also employed six kinds of features which are POS, NER, hypernyms, position feature, dependency feature, and relative-dependency feature as lexical features. Sentence level features which were derived from LSTM and lexical level features were combined into the final extracted feature vector using a multilayer perceptron. In order to predict the semantic relation between two entities, the final extracted feature vector was fed into a softmax classifier.

### 3.1.4 Attention-Based Neural Models

For various NLP tasks, attention networks have shown to be highly helpful. Word-level attention model was employed by Shen and Huang [89] and Jat et al. [86] for single-instance sentence-level relation extraction.

A convolutional neural network model and an attention network were combined, according to Shen and Huang's [89] idea. The global features of the sentence are first extracted using a convolution operation with max-pooling. Then, for each word in the sentence, attention mechanism is applied based on the two entities individually. The word embedding of each word

is concatenated with the word embedding of the final token of an entity. To get a scalar attention score for each word of that entity, this concatenated representation is fed into a feed-forward layer with *tanh* activation and then another feed-forward layer with soft-max. To obtain the attentive feature vectors, the word embeddings are averaged depending on the attention scores. To find the relation, a feed-forward layer with softmax receives the global feature vector as well as two attentive feature vectors for each of the two entities.

Wang et al. [90] proposed a convolutional neural network architecture, relying on two levels of attention. The first attention mechanism captures entity-specific attention with respect to the target entities, and the second attention mechanism is responsible for capturing relation-specific pooling attention with respect to the target relations. Because of the various structure of the input sentence, this multi-level attention mechanism enables the model to recognize more cues and effectively learn which sections of the input are important for classification. The authors also introduced a pair-wise margin-based loss function that uses a distance function to measure the distance between the output of the network and the candidate relation label.

Having access to both past and future context is useful for many sequence modeling tasks. Zhou et al. [91] added attention mechanism to previously developed bidirectional long short-term memory to propose attention-based bidirectional LSTM networks for relation classification task. Their model does not use NLP tools or lexical resources as input, it just takes raw text with position indicators.

A **bidirectional gated recurrent unit (Bi-GRU)** [92] was employed by Jat et al. [86] to identify the long-term dependency between the sentence's words. A Bi-GRU layer receives the tokens vectors $x_t$. To calculate a scalar attention score for each word, the hidden vectors from the Bi-GRU layer are fed into a bi-linear operator, which combines two feed-forward layers with softmax. For scaling up the hidden vectors, the Bi-GRU layer's hidden vectors are multiplied by the associated attention scores. The feature vector is produced by using a piecewise convolution neural network (Zeng et al. [6]) to the scaled hidden vectors. To find the relation, this feature vector is sent into a feed-forward layer with softmax.

Lee et al. [93] proposed an end-to-end Bidirectional Long Short-Term Memory network that combines an entity-aware attention mechanism with

a latent entity typing (LET) method which means using entities and their latent types as new features. They used self-attention mechanism to aggregate word representations and a Bi-LSTM network in order to capture the context of sentences. The authors also demonstrated that by considering relative position of words, entity pairs, and their types which are derived from LET, entity-aware attention concentrates on the most significant semantic information.

An Attention model was utilized by Lin et al. [69] to extract multi-instance relations. They used attention mechanism on a bag of distinct sentences which include two entities to extract the relation between all those sentences. The sentences in a bag are first encoded using CNN-based models. The importance of each sentence in the bag is then calculated using a bi-linear attention layer. To some extent, this attention aids in reducing the issue of noisy samples collected by distant supervision. More related sentences, as opposed to noisy ones, are expected to receive greater attention scores. Based on the attention scores assigned to each sentence vector in the bag, the sentences are combined into a weighted average. To find the relation, the weighted average vector of the sentences is fed into a feed-forward neural network with softmax. Only positive relations get this bag-level attention; None relations do not. It is challenging to determine appropriate weights for bags that represent no relations since their representations are constantly varied.

In a multi-instance scenario, Ye and Ling [73] employed intra-bag and inter-bag attention networks for relation extraction. They employ intra-bag attention, which is similar to Lin et al [69]. In order to solve the issue of noisy bags, they also employed inter-bag attention. They separate each bag into different categories according to the relation it belongs to. Based on how similar the bags in a group are to one another, the attention score for each bag in the group is calculated. As we do not know the relations during testing, this inter-bag attention is only employed during training. Nayak and Ng [94] developed a multi-factor attention model for relation extraction that focuses on the syntactic structure of a sentence. To determine the syntactic structure of a sentence, they employed a dependency parser. The authors employed multi-factor attention to assess the semantic similarity of words with the given entities and combine it with the dependency distance of words from the given entities to assess their effect in determining the relation.

An effective novel model developed by Zhang et al. [87] which combines an LSTM sequence model with an entity position-aware attention that is more effective for relation extraction. They also developed TACRED which was a better and larger supervised dataset for relation extraction. Due to new dataset and a more suitable high-capacity model, they improved the performance of relation extraction significantly.

Yu et al. [95] proposed a segment attention-based neural network model which incorporates segment information of a sentence based on the fact that information relevant to relation is generally held inside segments. In their method, the attention mechanism is seen as a set of linear-chain conditional random fields across a collection of latent variables, the edges of which encode the desired structure. The attention weight is seen as the marginal distribution of each word chosen as a component of the relational expression.

### 3.1.5 Dependency-Based Neural Models

In some earlier publications, the neural models for relation extraction included the dependency structure information from sentences. To determine the relation between two entities, Xu et al. [96] employed an LSTM network along the **shortest dependency path (SDP)**. Four embeddings are used to represent each token along the SDP: (i) pre-trained word vector, (ii) POS tag embedding, (iii) embedding of the token's and (iv) its child's dependency relation in the SDP and embedding for its WordNet hypernym ([97]). They separate the SDP into two different pathways: The left SDP and the right SDP. The left SDP connects the first entity to the common ancestor node and the right SDP connects the second entity to the common ancestor node. The definition of common ancestor node is the dependency tree's lowest common ancestor between the two entities. The token vectors along the left and right SDPs are given independently to an LSTM layer. To extract the feature vector from the left and right SDPs, a pooling layer is used for the hidden vectors. To determine the relation, these two vectors are concatenated and sent into a classifier.
For relation extraction, Liu et al. [98] used the SDP between two entities and the sub-trees associated with that path which is called the augmented dependency path. The pre-trained embedding and sub-tree representation of each token in the SDP serves as its representation. A token's sub-tree representation is derived from the dependency tree's sub-tree, where the token serves

as the root node. Trainable embeddings are used to express the dependency relations. Each node in a token's sub-tree receives data from its children, including the dependency relationships. By tracking the sub-tree rooted at the token from its leaf nodes to its root in a bottom-up manner, it can extract the subtree representation of the token. The vectors of the tokens sequence and dependency relations throughout the SDP are then fed into CNN with max-pooling. A classifier is used to identify the relation using the result of the max-pooling operation.
SDP between two entities was developed by Miwa and Bansal using a tree LSTM network [99]. In their model, they combined a top-down tree LSTM with a bottom-up tree LSTM. Each node in the bottom-up tree LSTM receives data from each of its children. The ultimate output of this bottom-up tree LSTM is the hidden representation of the root node. Each node in the top-down tree LSTM receives the data from its parent node. The output of this tree LSTM is the hidden representations of the head tokens of two entities. In order to identify the relation, the representations of the bottom-up tree LSTM and top-down tree LSTM are concatenated and sent to a classifier. They demonstrated how it is beneficial to use the SDP tree instead of the complete dependency tree since irrelevant tokens for the relation are disregarded during the process.

### 3.1.6 Graph-Based Neural Models

Due to their ability to handle nonlinear structures, graph-based models are useful for many NLP tasks. A graph-based approach for cross-sentence relation extraction was presented by Quirk and Poon [100]. From the sentences, they constructed a graph in which each word functioned as a node. Edges are generated based on discourse relationships, dependency tree relations, and word adjacency. Beginning with entity 1 and on to entity 2, they extract all the routes from the graph. Features like lexical tokens, the lemma of the tokens, POS tags, etc. are used to represent each route. They search for the relationship between the two entities using all of the route features.
For N-ary cross-sentence relation extraction, Peng et al.[101] and Song et al. [102] employed a similar graph. They employed an LSTM on a graph as an alternative to explicit paths. A general structure for a linear LSTM or tree LSTM is a graph LSTM. The graph's LSTM changes to a linear LSTM if it only has words adjacency edges and if the dependency tree's edges are present in the graph, it transforms into a tree LSTM. Cycles may be present in a general

graph structure. Thus, this graph is split into two **directed acyclic graphs (DAG)** by Peng et al. [101], where the forward DAG only includes forward edges among the tokens and the backward DAG only includes backward edges among the tokens. There is a unique forget gate for each of a node's neighbors. It uses LSTM equations ([22]) to update its hidden states after receiving input from its neighbors. This graph LSTM transforms into a bi-directional linear LSTM if we simply take the words adjacency edges into account. Instead of splitting the graph into two DAGs, Song et al. [102] updated the nodes' states directly using the graph structure. Each node collects information from its neighbors from the previous time step at time step t, updating its hidden states with the help of LSTM equations. A hyperparameter denoted as k determines the number of times this procedure is performed.

A **graph convolutional network (GCN)** model was suggested by Kipf and Welling [103] and Velickovi'c et al. [104] which employed linear transformations to update the node states. Velickovi'c et al. employed an attention method to assign various weights to the edges, Kipf and Welling provided equal weight to the edges.

GCN was utilized by Vashishth et al. [71], Zhang et al. [105], and Guo et al. [106] to extract sentence-level relations. By using each word in a sentence as a node, they employed the syntactic dependency tree to establish a graph structure between the nodes. The GCN was applied in a multi-instance situation by Vashishth et al. [71]. Over the whole dependency tree of the sentences, they employed a Bi-GRU layer and a GCN layer to encode them. To identify the relation, the sentence representations in a bag were combined and sent to a classifier. After Miwa and Bansal [99], Zhang et al. [105] constructed the graph's adjacency matrix using just the shortest dependency path (SDP) tree. The edges that are K distances away from the SDP tree, where K is a hyper-parameter, were also included. Guo et al. [106] suggested a new method called **AGGCNs (Attention Guided Graph Convolutional Networks)**. A soft pruning method in their GCN model as opposed to Zhang's hard pruning strategy. When constructing the adjacency matrix, they took into account the whole dependency tree, but by employing a multi-head self-attention-based soft pruning method, they were able to recognize between the essential and irrelevant edges in the graph.

Tian et al. [107] proposed a dependency-driven approach with attentive graph convolution networks (A-GCN). Their method leverages dependency information for RE task. The attention mechanism is applied to dependency

connections to impose weights on both connections and dependency types. This helps better identify the critical dependency information and leverage them appropriately. As a result, A-GCN learns from varied dependency connections, allowing less-informative dependencies to be pruned.

### 3.1.7 Contextualized Embedding-Based Neural Models

Relation extraction can benefit from contextualized word embeddings like ELMo [35], BERT [31] and SpanBERT [108]. These **language models (LMs)** can accurately represent the contextual meaning of words in their vector representations since they were trained on huge corpora. The word embedding layer of all described neural models for relation extraction applies word representation like Word2Vec [32] or GloVe[34]. To further enhance the performance of relation extraction models, contextualized embeddings can be joined to the embedding layer. On the TACRED dataset, the SpanBERT model performs significantly better. In order to train the model, Joshi et al. [108] changed the entity 1 token with its type and SUBJ, such as PER-SUBJ, and entity 2 token with its type and OBJ, such as LOC-OBJ in each sentence and then, they discovered the relation by adding a linear classifier on top of the CLS token vector.

Wang et al. [109] proposed a new method for extracting multiple entity-relation tuples in input sentences that only requires a single pass of encoding. They used BERT as the pre-trained encoder with two new modifications: a new prediction layer for predicting multiple relations for various entity pairs, and a self-attention mechanism in the self-attention layer that knows the position of all entity pairs. After fine-tuning, their model achieved acceptable results on single relation extraction task.

Relation extraction methods that compute explicit linguistic features (like lexical, syntactic, and semantic features) in the pre-processing step have some limitations. The need for extra annotated language resources when training feature extraction models limits the portability of relation extraction to new languages. On the other hand, pre-processing add more errors to the model. Alt et al. [110] proposed **TRE (Transformer for Relation Extraction)** to overcome these limitations. TRE represents long-range dependencies

across entity mentions by combining the self-attentive Transformer architecture with pre-trained deep language representations rather than explicit linguistic features. These linguistic features are learned implicitly by unsupervised pre-training from plain text. Then the fine-tuning process is applied to the learned language representation to do relation extraction task.

Shi and Lin [111] were the first researchers who used BERT pre-trained language model for the relation extraction task. They employed a simple BERT-based model without any external features, and their model outperformed all previous best-known models [105] and [112] which used GCNs [103] and external features to encode syntactic tree information.

Chen et al. [113] proposed DG-SpanBERT (SpanBERT-based graph convolutional network) to improve the performance of the relation extraction task on the TACRED dataset. Their model obtains the benefit of successfully extracting semantic features from a raw sentence using the pre-trained language model SpanBERT and GCN to combine latent features. DG-SpanBERT model uses SpanBERT's ability to learn complex lexical features from corpora. Because of the use of GCN on the dependency tree, their model also can capture long-distance relationships between entities.

By adding entity information to the pre-trained BERT model, Wu and He [15] developed a method for relation extraction and named it R-BERT. In order to be able to use target entities' location information, they added special separate tokens for each entity pair and used this information in sentence vector representation.

To improve performance on relation extraction, Tao et al. [114] provided an indicator-aware approach that uses both the syntactic indicator and the textual context. At first, they utilized syntactic knowledge to extract syntactic indicators. After that, a new neural network model based on BERT pre-trained language model was developed to include sentences and their syntactic indicators. By doing so, the authors claimed that the suggested approach overcomes the limitation of text triggers and reduces the influence of noisy information from sentences.

**Pre-trained Models Using External Knowledge** Several recent research has investigated ways to improve Contextual word representations by feeding

them knowledge from external sources, such as extra training data, and knowledge bases.

Soares et al. [14] proposed a new BERT-based model for better representation of relations. Their model (BERT-EM) represents target entities in the input to BERT by using entity marker tokens and extracts a fixed length representation of a relation from BERT's output by using entity start state. They also proposed a new method for training the model without predefined ontology or relation-labeled training data. This method is based on calculating a similarity score between two different relation statements. If two relation statements are semantically related, the similarity score should be high and it should be low if two relation statement relations are semantically different. Using the observation: two relation statements are more semantically similar if they share the same entities, they developed the learning method **matching the blanks (MTB)**, which can be used without any tuning for relation extraction. By applying this new learning method on TACRED and SemEval 2010 Task 8, the authors showed how their method is effective even with low-resource training data.

Factual information from the text cannot be adequately captured by pre-trained language representation models. On the other hand, knowledge embedding approaches are capable of accurately representing relational facts in knowledge graphs with meaningful entity embeddings.

In order to improve contextual word representation using structured knowledge, ME. Peters et al. [115] provided a general method for inserting various knowledge bases into large-scale pre-trained models with a Knowledge Attention and Recontextualization (KAR) system. Their method for each knowledge base begins by extracting relevant embeddings for entities via an integrated entity linker, next changing contextual word representations using word-to-entity attention. The knowledge-enhanced BERT (KnowBERT), which integrates WordNet and Wikipedia into BERT, improved many NLP downstream tasks like relation extraction, words in context (WiC), and entity typing. Unlike previous approaches, their method employs self-supervision on unlabeled data to lean entity linkers.

X.Wang et al. [116] provided an integrated model for Knowledge Embedding and Pre-trained Language Representation (KEPLER), which is able to incorporate factual knowledge into pre-trained language models much better and also construct text-enhanced knowledge embedding with respect

to powerful pre-trained language models. Previous works incorporate constant entity embeddings into pre-trained language models to provide external knowledge. So, embeddings are learned by another knowledge embedding model and thus, an entity linker is required to align the text with the corresponding entities. KEPLER, on the other hand, encodes texts and entities into a single semantic space using the same pre-trained language model and optimizes the knowledge embedding and masked language modeling objectives jointly.

R.Wang et al. [117] stated that most previous methods that inject different kinds of knowledge into pre-trained language models suffer from forgetting the historically infused knowledge. This problem happens when the parameters of the pre-trained model are updated by injecting new knowledge. In order to deal with this issue, authors proposed a new framework called K-ADAPTER which keeps the original parameters of the pre-trained model unchanged and facilitates the construction of flexible knowledge-infused models. They used RoBERTa [118] as the main language model and K-ADAPTER as a set of neural adapters. For each kind of injected knowledge, there is a neural adaptor that acts as a plug-in attached outside of the RoBERTa.

Because of the importance of entity representations in entity-related tasks like relation extraction, Yamada et al. [119] proposed LUKE, a new pre-trained contextualized representation of words and entities based on the bidirectional transformer. Their model generates contextualized representations of the words and entities in a text by treating them as separate tokens. LUKE is trained to predict words and entities that have been randomly masked by using entity-annotated words and entities corpus of Wikipedia. Additionally, they suggested an extension of the transformer's self-attention mechanism (entity-aware self-attention mechanism) that takes the type of each token (word or entity) into account while calculating attention scores.

According to Cohen et al. [120] argument, the relation classification task can be treated as a span-prediction task, similar to a question-answering problem. They provided a system for relation classification based on span predictions and evaluated its performance in comparison to the embedding-based system. In their method, each relation classification sample is reduced to a sequence of binary span-prediction tasks. The authors showed that the

supervised span-prediction objective outperforms the traditional classification-based objective by a large margin.

## 3.2 Joint Extraction Approaches

All of the efforts on relation extraction that have been previously discussed rely on named entity recognition systems to identify entities. At the sentence or bag-of-sentences level, they classify the relationship between two specified entities. To identify the entities in a text, these models rely on a third-party named entity recognition system. Some studies such as Katiyar and Cardie [121], Miwa and Bansal [99], Bekoulis et al. [122] and Nguyen and Verspoor [123] have attempted to eliminate this dependence. By sharing the parameters and jointly optimizing, they attempted to close the gap between the entity recognition and relation extraction tasks. They start by identifying every entity in a sentence, and then they work to determine the relation between every pair of entities they have recognized. They identify the entities and relations in the same network, but they still identify the entities first, then decide the relation between any two possible pairings in the same network. Therefore, these models fail to consider the interactions between the relation tuples that involve in a sentence. In some ways, these strategies are similar to the pipeline strategy.

For this task, Zheng et al. [124] introduced the first joint extraction model. In order to jointly extract the entities and relations, they employed a sequence tagging approach. They developed a set of tags based on the Cartesian product of entity and relation tags. These new tags have the ability to encode both relational and entity information simultaneously. However, because only one tag may be applied to a token, this technique fails when entities are shared throughout many tuples. According to Zeng et al. [125], relation tuples with overlapping entities may be extracted using an encoder-decoder model with a copy mechanism. Their model consists of two networks: a copy network to copy the last token of two entities from the source sentence, and a classification network to classify the relation between copied tokens. The complete entity names of the tuples cannot be extracted by their model. Their top-performing model extracts each tuple using a different decoder. They must fix the maximum number of decoders during training, and their model can only extract as many tuples as that predetermined number during inference. Additionally, their approach fails to account for the interaction between the

tuples since it uses distinct decoders for each tuple.
For the joint extraction of entities and relations, Takanobu et al. [126] developed a deep neural model based on **Hierarchical Reinforcement Learning (HRL)**. Based on the sentences' relation-specific tokens, a high-level reinforcement learning is utilized to determine the relation. Once a relation has been found, low-level reinforcement learning is employed utilizing a sequence labeling technique to extract the two entities connected to the relation. To extract all the relation tuples that are contained in the sentences, this method is performed several times. To recognize no relation situations in sentences, a specific *None* relation is employed and entities derived from *None* relations are not taken into account. Using a **Graph Convolutional Network (GCN**), Fu et al. [127] regarded each word in a sentence as a node in a graph and edges as relations. Trisedya et al. [128] completed knowledge bases utilizing distantly supervised data using an N-gram attention mechanism with an encoder-decoder model. In order to do this task, Chen et al. [129] employed the encoder-decoder architecture. First, they utilized a CNN-based multi-label classifier to identify all the relations, and then they used multi-head attention (Vaswani et al. [26]) to extract the entities that corresponded to each relation.
The CopyR (Zeng et al. [125]) model, which employed a sequence tagging method to extract multi-token entities, has been enhanced by Zeng et al. [130]. The joint extraction problem was divided into two subtasks by Bowen et al. [131]: (1) **Head-Entity (HE)** extraction and (2) **Tail-Entity and Relation (TER)** extraction. To complete these two subtasks, they applied a sequence tagging strategy. They used a sequence tagging technique to first identify the head entities, after which they identified the tail entities for each head entity and each relation. Wei et al. [132] used a sequence tagging approach to work on this task too. For their model to perform better, they employed pre-trained BERT (Devlin et al. [9]).

# Chapter 4

# Relation Classification Using Pre-trained Language Models

It's been a long-standing objective of natural language processing to read texts to recognize and extract relationships between entities. Using a classification model that seeks to classify the types of entities in a sentence is one method for extracting relationships between entities from a text corpus. The classes of relations that may be predicted using this modeling approach are constrained to those that the model learned during training, which is one of its key drawbacks. The model must be retrained on the additional classes to expand the existing ones. To put it another way, one drawback of using a classification approach to extract relations between entities is that the model will only be able to predict those specific relationships that exist in the training set. Additionally, since humans often express identical relations in various ways, the model may not be able to generalize sufficiently to recognize all representations of a relation.

In order to address mentioned issues, the authors of matching the blanks (MTB) [14] looked for a solution to model relations between entities as numerical representations rather than by employing a classification algorithm. There are two main advantages to this numerical representation. First, we could use transfer learning to teach the model how to predict the relation classes of interest in a downstream train task. Second, group various ways of expressing the same or similar relations between entities in an embedding space by representing similar expressions of relations. In order to achieve these advantages, the paper's authors proposed a model that can generate relation representations by projecting the relationships between entities in an embedding space.

At a high level, the model gains the ability to classify relationships between marked entities according to their true meaning. The network incorporates relationships between blanked entities to generate representations similar to those relationships if they originate from the same entities. This incorporation is done during the training step. The model then continues to the fine-tuning step, where it learns to do the exact classification task and predicts the relation between entities.

## 4.1 Motivation

A key goal in information extraction is to create general-purpose relation extractors that can represent any kind of relation. There have been attempts to develop general-purpose relation extractors that express relations together with their surface forms or that simultaneously embed surface forms and relations from an existing knowledge graph. But the generalizability of each of these methods is limited. In order to create task-independent relation representations only from entity-linked text, Soares et al. [14] created an extension of Harris' distributional hypothesis to relations as well as recent developments in learning text representations (particularly, BERT).
On SemEval 2010 Task 8, we demonstrate that the R-BERT model [15], fine-tuned on supervised relation extraction datasets after being initialized with task-agnostic representations, outperforms the previous approaches using the BERT model.

## 4.2 Problem Definition

We divide the problem into three parts. In the first part, we will explain how we are going to represent relation statements and how using the deep transformers model helps us gain a more task-specific representation of relation statements.
In the second part, we will talk about how to classify relation statements represented by the deep transformer model. And, in the third part, we will explain our method in which how to utilize pre-trained knowledge achieved from one deep transformer language model to another, in order to perform the relation classification task.

## 4.3 Relation Representation

Assuming that we have access to a corpus of text in which entities have been assigned specific identifiers, we define a relation statement as a block of text that contains two entities that have been tagged.

As shown in Figure 4.1, we next use this information to generate training data that includes relation statements where the entities have been changed to the unique [BLANK] symbol.

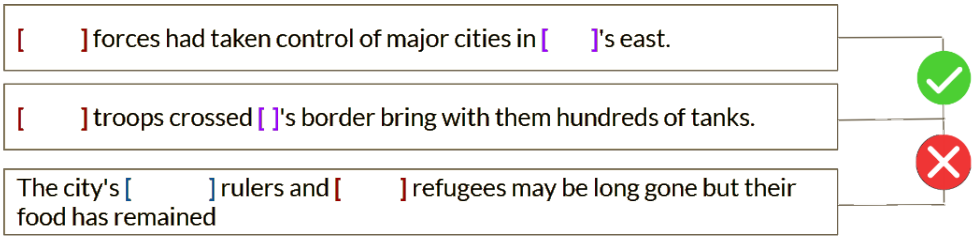


FIGURE 4.1: An illustration of "Matching the Blanks" example in which the first and second sentences, the same two entities are shared by both related relation. The semantic relation between the two entities in the third sentence is different.

The training approach receives pairs of relation statements with blanks in them, with the goal of encouraging relationship representations to be similar they represent the same pairs of entities.

The objective is to learn how to map relation statements in natural language to relation representations. Let $x = [x_0 ... x_n]$ be a formal representation of a series of tokens, $x_0 = [CLS]$ and $x_n = [SEP]$ operating as special start and end tokens. Consider the pairs of integers $s1 = (i, j)$ and $s2 = (k, l)$, where $0 < i < j - 1$, $j < k$, $k < l - 1$ and $l < n$. A relation statement is represented by the triplet $r = (x, s1, s2)$, where the indices in s1 and s2 define entity mentions in x. So, the sequences $[x_i ... x_{j-1}]$ and $[x_k ... x_{l-1}]$ both mention entities. The objective is to train the function $h_r = f_\theta(r)$ that converts the relation statement into a fixed-length vector $h_r \in \mathbb{R}^d$ which represents the relation between two entities denoted by s1 and s2 in sentence x.

### 4.3.1 Matching the blanks model architecture

The main objective of relation learning is to create models that generate relation representations straight from the text. The BERT model proposed by Devlin et al. [9] was used as the foundation for relation learning due to the

successful training of recent deep transformers on several language modeling versions. The deep transformer model is initialized with the pre-trained Masked language model BERT parameters. To start learning relation representation, relation statements containing two sentences annotated by entity markers special tokens are fed to the model as input.

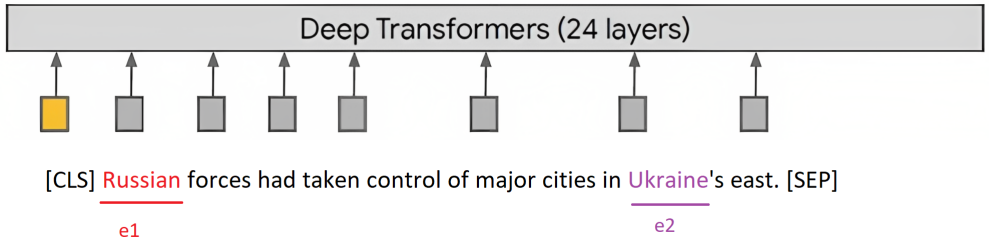


FIGURE 4.2: Input of deep transformer model used for matching the blanks pre-training task.

### 4.3.2 Relation Representations from Deep Transformers Model

For representing entities in the input to BERT and extracting a fixed-length representation of relationships as the last hidden layer of BERT, Soares et al. [14] provided three options. Six combinations are made for both the input encoding and the output representation, which is shown in Figure 4.3.

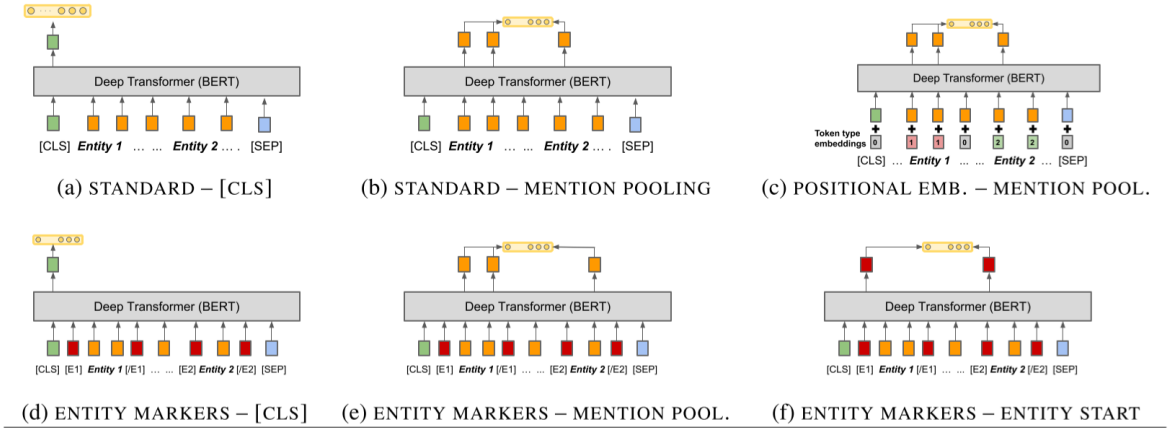


FIGURE 4.3: Different options of feeding relation statements to deep Transformers network and extracting relation representation from it. [14]

#### Entity span identification

Remember from Section 4.3 that the entity span identities s1 and s2 are contained in the relation statement r = (x, s1, s2). The following three options are

provided for feeding the BERT encoder with information about the spans s1 and s2:

**Standard input** When using the standard input option, the BERT model does not have any knowledge of entity spans.

**Positional embeddings** When employing the positional embedding option, BERT additionally adds a segmentation embedding for each of the tokens in its input. This embedding is generally used to contribute sentence segmentation information to the model.

**Entity marker tokens** Using the entity marker tokens option, each entity mention has four reserved tokens at the beginning and the end of its relation statement. This input representation is referred to as ENTITY MARKERS which is shown below:

$$x = \left[x_0 ... [E1_{start}] x_i ... x_j [E1_{end}] ... [E2_{start}] x_k ... x_{l-1} ... [E2_{end}] ... x_n\right] .$$

**Fixed length relation representation**

There are three different ways to get a fixed length relation representation, abbreviated *hr*, out of the BERT encoder. The last hidden states of the transformer, which we describe as $H = [h_0, ...h_n]$ for $n = |x|$, are what the three variations rely on.

**[CLS] token** Remember from Section 4.3 that each relation statement begins with a reserved [CLS] token. When using the [CLS] token option, the output of [CLS], $h_0$ is our relation representation.

**Entity mention pooling** When employing the entity mention pooling option, by max-pooling the last hidden layers that match the words in each entity mention, *hr* will be extracted.

**Entity start state** Using the entity start state option, The concatenation of the final hidden layers corresponding to the start tokens of two entities will be used to describe their relationship. This only works with the entity marker tokens option as input.

According to Soares et al. [14], employing the entity markers option as input and entity start as output representation achieves the best results after applying the BERT model to fine-tuning tasks. So, we only applied the mentioned setting as our input and output representation to the Bert model which is illustrated in Figure 4.4.

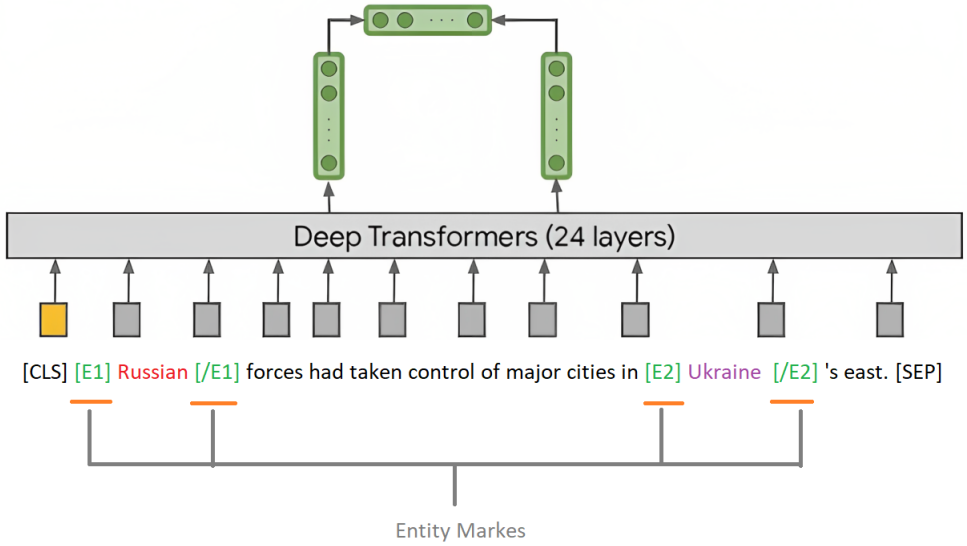


FIGURE 4.4: Input and output representation of deep transformer model used for matching the blanks pre-training task. The input is two statements annotated with the entity marker special token. The separator special token separates two statements. The output is a relation representation which is the concatenation of the last hidden layer of two entity start special tokens.

### 4.3.3 Training BERT model using Matching the Blanks

Based on a novel method introduced by Soares et al. [14], we are going to train $f_\theta$ without any relation-labeled data or ontology. The method is based on the assumption that "If two relation statements, $r$ and $r'$, express semantically similar relations, the inner product $f_\theta(r)^T f_\theta(r')$ for every pair of relation statements should be high. And If the two relation statements represent semantically different relations, then this inner product should be low."

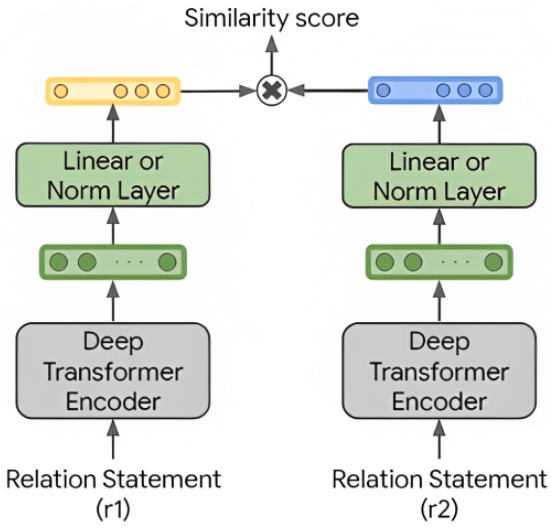


FIGURE 4.5: Pre-training procedure of MTB model.

### Training Objective

The objective is to develop a relation statement encoder $f_\theta$ that will allow us to identify whether two relation statements represent the same semantic relationship or not. The following binary classifier 4.1 is defined to determine the probability that r and r′ express the same relation (l = 1) or that they do not (l = 0).

$$p(l = 1|r, r') = \frac{1}{1 + \exp f_\theta(r)^T f_\theta(r')} \tag{4.1}$$

Then, we will perform learning step to parameterize $f_\theta$ to minimize the 4.2 loss.

$$L(D) = -\frac{1}{|D|^2} \sum_{(r,e1,e2 \in D)} \sum_{(r',e1',e2' \in D)} \delta_{e1,e1'}\delta_{e2,e2'} \cdot log p(l = 1|r, r') + (1 - \delta_{e1,e1'}\delta_{e2,e2'}) \cdot log(1 - p(l = 1|r, r')) \tag{4.2}$$

$D$ is the corpus containing relation statements labeled with two entities (D = $[(r^0, e_1^0, e_2^0)$, ..., $(r^N, e_1^N, e_2^N)]$). $\delta_{e,e'}$ is the Kronecker delta, which has a value of 1 if $e$ equals $e'$ and a value of 0 otherwise.

### Blanking Entity Mentions

The entity linking system utilized to generate D can precisely reduce the loss in Equation 4.2. Furthermore, it is implausible to expect that $f_\theta$ would construct meaningful relation representations because this linking system lacks any idea of relations. We develop a modified corpus to prevent only learning the entity linking mechanism.

$$\tilde{D} = \left[ (\tilde{r}^0, e_1^0, e_2^0), ..., (\tilde{r}^N, e_1^N, e_2^N) \right] \tag{4.3}$$

Each $\tilde{r}^i = (\tilde{x}^i, s_1^i, s_2^i)$ in corpus includes a relation statement, where one or both entity mentions may have been replaced with a special [BLANK] token. In particular, $\tilde{x}$ has the chance of containing the span specified by $s_1$ with probability $\alpha = 0.7$. For $s_2$, the same holds true. Only $\alpha^2$ of all relation statements in $D$ mention both of the entities involved in the relation.
Therefore, reducing $L(\tilde{D})$ needs $f_\theta$ to perform more than just detecting and matching named entities in r. Specifically, performing training on $\tilde{D}$ will provide an $f_\theta$ that represents the semantic relationship between the two probably omitted entity spans.

### Training of Matching the Blanks

We build a BERT-like training setup where the masked language model loss and the matching the blanks loss are applied simultaneously in order to train a model with the matching the blank task.

**Pre-training Data** We utilize CNN corpus [133] to create the training corpus. It should be noted that the research utilizes wiki dumps data for MTB pre-training, which is significantly larger than the CNN corpus. We extract paired entities from the text using Spacy NLP (within a window size of 40 tokens length) to create relation statements for pre-training. Entities are recognized using Named Entity recognition and the dependency tree parsing of objects and subjects.

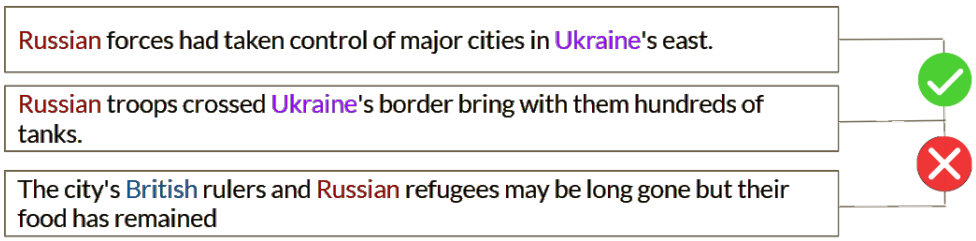


FIGURE 4.6: An example of three generated samples from CNN corpus used for Matching the Blanks pre-training task. In the first and second sentences, the same two entities are shared by both related relations. The semantic relation between the two entities in the third sentence is different.

### 4.3.4 Fine-tuning Task

For the supervised relation extraction task, Soares et al. [14] have created a new classification layer, $W \in R^{KxH}$, where K is the number of relation types and H is the size of the relation representation. The standard cross-entropy of the softmax of $h_r W^T$ with regard to the true relation type is the classification loss.

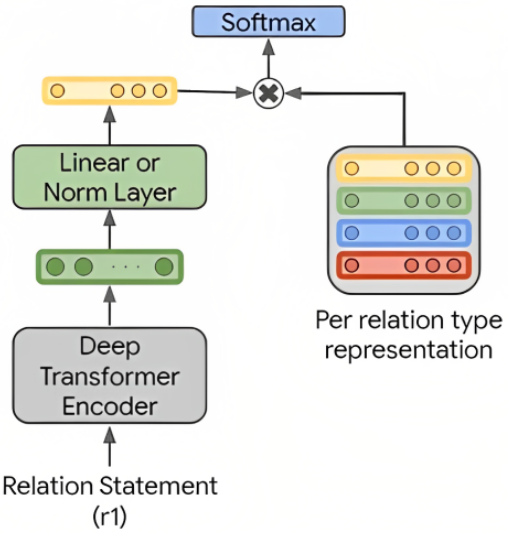


FIGURE 4.7: Fine-tuning procedure of MTB model.

## 4.4 Relation Classification

The objective is to learn how to classify relation statements in a supervised way. In order to fine-tune the BERT model for relation classification, Wu and He [15] offered a model that uses both the pre-trained BERT language model and information from the target entities. By applying BERT as the pre-trained

language model, using special tokens to locate the position of entities in the fine-tuning phase, and locating the embedding of entities in the output, the model is able to incorporate the semantic embedding of the sentence and the two target entities in order to fit better for the relation classification task.

### 4.4.1 Pre-trained BERT

The pre-trained BERT model [9] is a bidirectional transformer encoder [26] which is pre-trained on two massive English corpora: BooksCorpus [134] and English Wikipedia with respect to Masked Language Model (MLM) and Next Sentence Prediction (NSP) objectives. We explained the BERT model in more detail in 2.3.1.
In the following, we are going to fine-tune the pre-trained BERT to perform the relation classification task.

### 4.4.2 Model Description

The architecture of the model is illustrated in Figure 4.8. By adding a special token '[CLS]' at the beginning of the sentence, a special token '$' at the beginning and end of the first entity, and a special token '#' at the beginning and end of the second entity in a sentence, the desired input sequence is prepared which makes the BERT be able to first, embed the semantic representation of whole sequence in the '[CLS]' token, second, locate the position of the two entities.

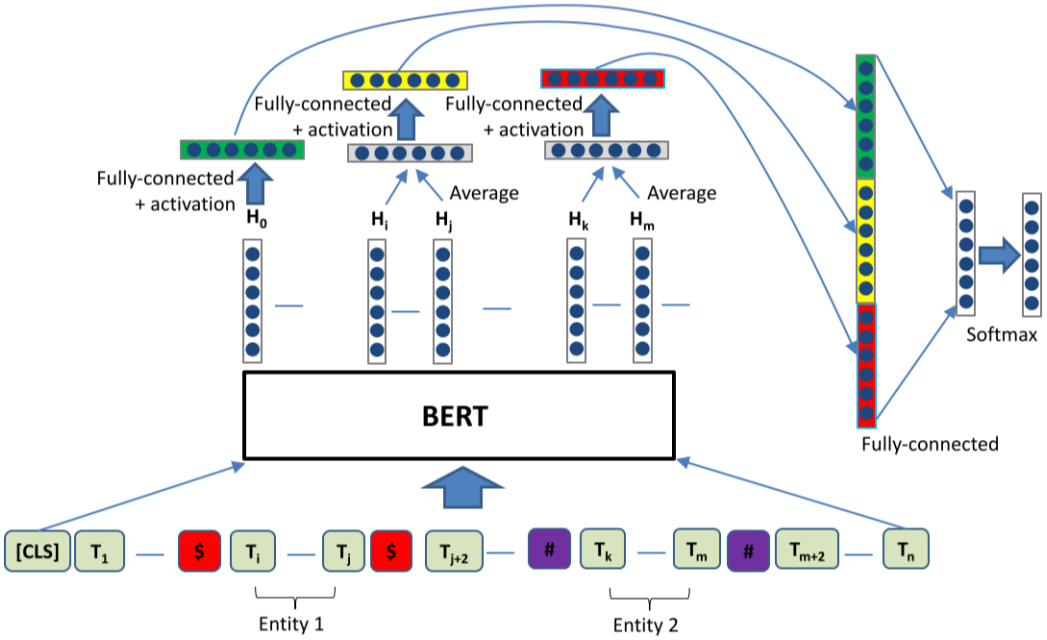


FIGURE 4.8: The model architecture to classify relation statements using BERT. [15]

$H_0$, $H_i$ to $H_j$, and $H_k$ to $H_m$ are the final hidden state outputs of the BERT module for the sentence S, the first entity, and the second entity, respectively. As described by Equations 4.4 and 4.5, by adding the average operation, the activation function, and a fully connected layer to the hidden representations of each entity, the final output of entities are obtained.

$$H_1' = W_1 \left[ tanh \left( \frac{1}{j-i+1} \sum_{t=i}^{j} H_t \right) \right] + b_1 \tag{4.4}$$

$$H_2' = W_2 \left[ tanh \left( \frac{1}{m-k+1} \sum_{t=k}^{m} H_t \right) \right] + b_2 \tag{4.5}$$

Also, the '[CLS]' token is passed through an activation operation and a fully connected layer which is described in Equation 4.6

$$H_0' = W_0 \left( tanh \left( H_0 \right) \right) + b_0 \tag{4.6}$$

After concatenating $H_1'$ and $H_2'$ to $H_0'$, a fully connected layer and a softmax layer will be applied as shown in Equations 4.7 and 4.8.

$$h'' = W_3 \left[ concat \left( H_0^{'}, H_1^{'}, H_2^{'} \right) \right] + b_0 \tag{4.7}$$

$$p = softmax(h'') \tag{4.8}$$

Cross entropy is the loss function used during training. The whole approach is called R-BERT by the authors.

## 4.5 Transferring Knowledge of The Relation Learning Task to The Relation Classification Task

In this section, we will combine the mentioned approaches as our contribution. Since the R-BERT classifies relation statements based on only pre-trained MLM BERT, which does not have any knowledge about relation representation, we believe if we could initialize the BERT with bert-base model parameters, then pre-train it using the MTB objective, the performance of the R-BERT would increase. The overall approach is illustrated in Figure 4.9

To elaborate, first, to train the MTB model to learn relation representation, we use the "Matching The Blanks" objective during the pre-training phase. The preprocessed CNN corpus is fed to the BERT model for this purpose. We rely on distributional similarity during the pre-training step to learn relation representation without using any labeled data. This process is known as distance supervision.

Second, instead of performing fine-tuning step to classify relation statements by the MTB model, we transfer the weight parameters of the pre-trained MTB model to the R-BERT initialized with the parameters of the bert-base-uncased. By doing so, when we perform the fine-tuning step to classify relation statements, the R-BERT model benefits from the pre-trained knowledge gained from both the Masked language model objective pre-trained on unlabeled, plain text corpus, the English Wikipedia, and the Brown Corpus besides the Matching the blanks objective pre-trained on the preprocessed CNN corpus.

Third, we perform the fine-tuning step of the R-BERT model to complete the relation classification task, using instances from SemEval-2010 Task 8 dataset.

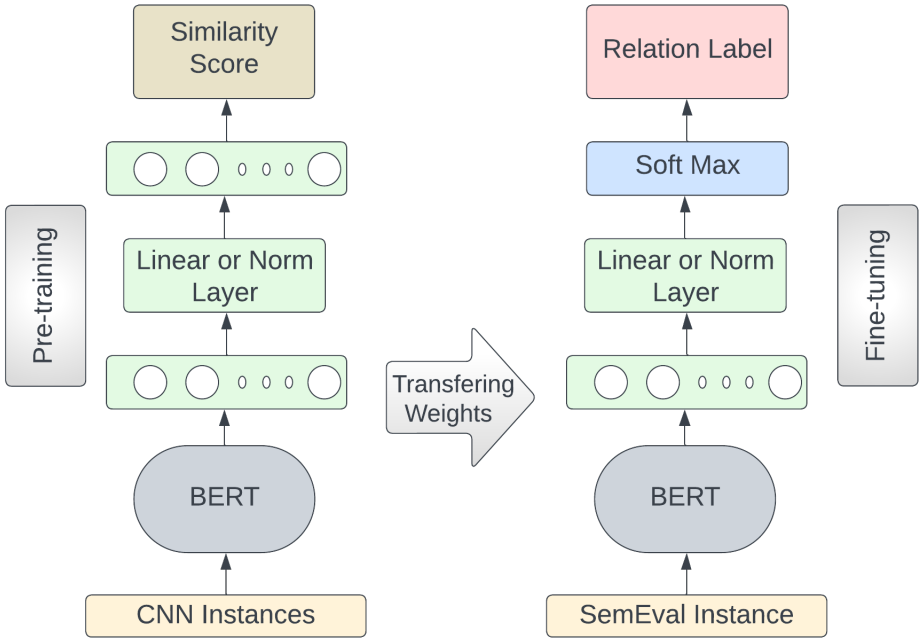


FIGURE 4.9: Transferring weights of pre-trained MTB model to R-BERT model in order to classify relation statements based on knowledge obtained from relation learning pre-training task.

# Chapter 5

# Impact of Training Data Size on Relation Classification

Feeding more instances to the classification models improves the model's performance in most cases. On the other hand, producing human-annotated datasets can be extremely costly. This cost might not be efficient regarding the amount of improvements that might be reached by increasing the number of instances. So, if we analyze the impact of dataset size on the model's performance, we can find the efficient amount of instances to perform our machine learning tasks.

## 5.1 Problem Definition

The accuracy of a machine learning model is highly dependent on how much training data has been fed to it. Most of the time, as the training dataset gets larger, the model's accuracy improves. The reason is that when we feed more data to the model, there is more information for the model to work with. Machine learning models are designed to learn the information in the data that is provided during training. If the dataset is sufficiently large, the model can identify patterns that a smaller dataset would have missed. Finally, as the amount of data goes up, the accuracy of the model improves. In addition to increasing accuracy, larger training datasets reduce the risk of overfitting the data, which can lead to improved machine learning accuracy. A model that is too complex may perfectly fit the training data, however, it will not generalize to new data. By limiting the model's complexity and reducing the likelihood of overfitting, more data might be used to train the model.
A larger dataset doesn't necessarily indicate higher accuracy. The model becomes less accurate as a result of the possibility of noise, outliers, and irrelevant information that a larger dataset with poor data quality could provide.

When building machine learning models, it is important to take extra care to establish a method for detecting trends in small datasets because it is both important practically and scientifically. Classification algorithms might perform worse if they are trained on small datasets. This is due to the fact that small datasets often contain fewer details, making it impossible for the classification model to generalize training data patterns. Furthermore, over-fitting becomes much more difficult to avoid because it often extends beyond training data to affect the validation set too.
When working with small datasets, classification task becomes more difficult. The small amount of training data, which results in an incorrect and biased classification model, is the main contributing factor to this issue. Since earlier research has focused on improving the classification algorithms' accuracy on datasets of a certain size, less attention has been paid to how the size of the dataset affects the performance of the algorithms, making this a field of study that needs to be investigated.

In machine learning, there are basically two types of errors:

- **Reducible errors:** The accuracy of the model can be increased by minimizing these errors. These errors be divided into bias and variance.
- **Irreducible errors:** No matter which approach is employed, these errors will always be included in the model. Unknown variables with nonreducible values are the fundamental cause of these errors.

Typically, a machine learning model analyzes the data, looks for patterns, and then makes predictions. These patterns are discovered by the model during training and are then used to make predictions using test data.

Bias errors are differences between the model's predicted values during prediction and the actual values. It can be described as a situation when machine learning algorithms fail to accurately identify the connections between the data points. Assumptions in the model produce bias, making the target function easy to learn, hence every algorithm comes with some bias. A model has one of the following characteristics:

- **Low Bias:** Basically, fewer assumptions will be made about the target function. The more complex the algorithm, the less bias is likely to be included.
- **High Bias:** The number of assumptions for a high bias model increases, which prevents it from capturing the key aspects of the dataset. These

kinds of models can not generalize well to unseen data points. The simpler the algorithm, the more bias is likely to be included.

High bias is caused mostly by a very naive model. Strategies like increasing the input feature, reducing the regularization term, and applying a more complex model can reduce the bias of a high-bias model.

The variance would describe the degree of variation in the prediction. In other words, variance indicates how much a random variable deviates from the value that is expected. A model should ideally not differ significantly from one training dataset to another, indicating that the method is effective at discovering the underlying relationship between the input and output variables. A model has one of the following characteristics:

- **Low Variance:** signifies that changes to the training data set cause a little variance in the target function's prediction.
- **High Variance:** demonstrates a significant variance in the target function's prediction as a result of changes to the training dataset.

A high variance model gains a lot of knowledge and performs well on the training dataset, but it fails to generalize well on the test dataset. Therefore, this model performs well on the training dataset but indicates significant error rates on the test dataset. This situation is known as overfitting. Strategies like decreasing the number of model parameters or input features, increasing the regularization term, and increasing the number of training data instances (Figure 5.1) can reduce the variance of a high-variance model.

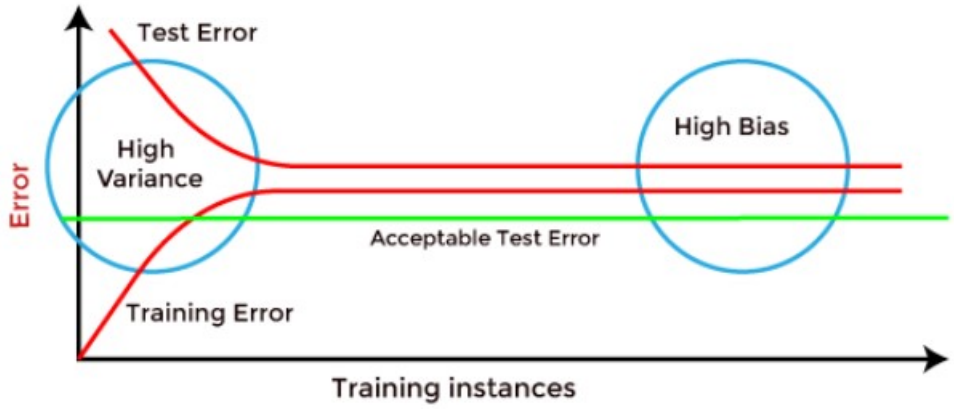


FIGURE 5.1: Impcat of the number of training data instances on bias and variance error. [135]

Little attention has been dedicated to studying the effect of dataset size on the performance of the classification algorithms; instead, research on the topic has mostly focused on improving the accuracy of the classification algorithms on small datasets.
The goal of this research is to determine how the size of the dataset affects the performance of four deep neural network models, including Attention-Based CNN, Attention-Based Bi-LSTM, and BERT with two different architecture and pretraining objectives performing in the given domain. Additional dataset size reduction scenarios were implemented, yielding a total of eight subsets. Next, we looked at how the models' accuracy, precision, recall, and f-score changed in response to the smaller datasets.

The process of partitioning the dataset is described in Algorithm 1. The objective is to investigate how the classification performance is affected by increasing the size of the same dataset. Four neural network models were trained on all subsets of the original dataset after the dataset had been preprocessed. We'll talk about the dataset and sub-dataset selection, classification models, and performance assessment metrics in the following subsections.

## 5.2 Subset Selection

Using the Python *random.sample()* method, we developed twelve subsets of the SemEval-2010 Task 8 dataset and sorted them by size, as shown in Table 5.1 to investigate how dataset size affects classifier performance.

**Algorithm 1** Dataset partitioning algorithm.

**Require:** $SE$ %Large Dataset
**Ensure:** $SE.1, SE.2, ..., SE.8$ %Eight Subsets of Original Dataset
  % Define seven subsets:
  $Subset - Sizes \leftarrow [1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1]$
  **for** i=1 to 8 **do**
    $X \leftarrow Subset - Sizes[i]$ % Get the size of the Subset
    $SE.i \leftarrow$ Randomly Select X Samples from SE
  **end for**

  **End**

| Dataset Notation | Portion | Number of Samples |
|---|---|---|
| 1 | 1/128 | 62 |
| 2 | 1/64 | 125 |
| 3 | 1/32 | 250 |
| 4 | 1/16 | 500 |
| 5 | 1/8 | 1000 |
| 6 | 1/4 | 2000 |
| 7 | 1/2 | 4000 |
| 8 | 1 | 8000 |

TABLE 5.1: Dividing the SemEval-2010 Task 8 Dataset to smaller subsets.

## 5.3 Classification Models

We applied four different deep neural classification models which are designed to deal with the relation extraction problem. The models include Attention-Based Neural Networks and Deep Transformer model.

### 5.3.1 Relation Classification from Attention-Based Neural Networks

In this part, we are going to explain two well-known attention-based neural networks which are created to extract semantic relations between two given entities. The main reason for using the attention mechanism in the relation extraction task is, not every word equally contributes to the semantic relation being represented. Therefore, learning to recognize important clues that define the primary semantic information is an essential task.
In the following, we will discuss two attention-based models. The models are Attention-Based Convolutional Neural Networks[89] followed by Attention-Based Bidirectional Long Short-Term Memory Networks [91]. We will explain these classification models in more detail below:

**Attention-Based Convolutional Neural Networks** The main idea of this architecture is to create an attention mechanism inside a convolution neural network to extract the words that are more related to the given entities and combine the representation of those relevant words into a sentence vector. The overall architecture of the model is illustrated in Figure 5.2.

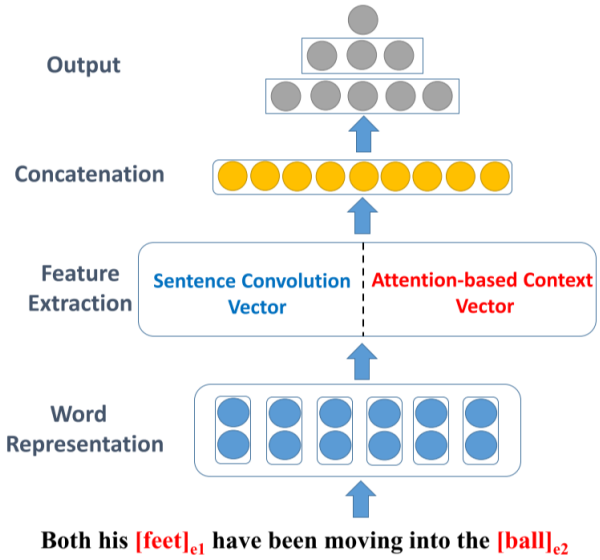


FIGURE 5.2: The model architecture to classify relation statements using Attention-Based Convolutional Neural Networks. [89]

The word representation is responsible for creating a numerical representation of a given sequence by the concatenation of word embedding, position embedding, and part-of-speech tag embedding. The feature extraction part of the model consists of two different modules.

In the sentence convolution vector module, First, a convolution layer is used to combine all local features and conduct global relation prediction. Second, a non-linear function is utilized to produce the feature map. After all, a max-pooling operation is applied on the feature map to capture the most important feature.

In the attention-based context vector module, an attention mechanism is developed to directly quantify the contextual importance of words with regard to target entities. To determine the weight of each word in the sentence, each word and entity in the sequence is fed to a multi-layer perceptron which is shown in Figure 5.3.

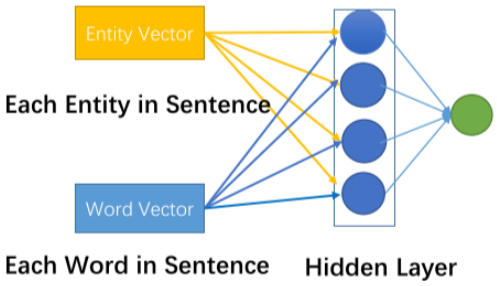


FIGURE 5.3: Attention weights computation unit. [89]

A new representation of a word is created by concatenating the representations of each entity and word. Figure 5.4 shows the attention layer architecture. So far, for each entity, we have a new sentence representation based on the computed attention score which tells us what are the informative words over each entity.

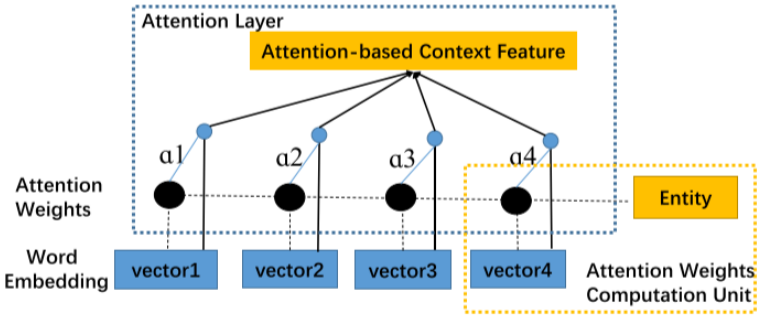


FIGURE 5.4: Attention layer network. [89]

After all, the sentence convolution vector and two attention-based context vectors of the two entities are concatenated into a fixed-length feature vector and fed to the output layer. The results of classifying relation statements on Attention-Based Convolutional Neural Networks with respect to different training subsets are reported in Table A.4.

**Attention-Based Bidirectional Long Short-Term Memory Networks** This architecture's core concept is to use an attention mechanism to identify the words that have the closest semantic relationship to the provided entities from the output of a bidirectional long short-term memory network without using any external knowledge.
It is useful to have access to both past and future context for many sequence modeling tasks. Standard LSTM networks, on the other hand, analyze sequences in order of time and disregard future context. By adding a second layer to the unidirectional LSTM networks, bidirectional LSTM networks improve by allowing the hidden-to-hidden links to flow in the opposite time order too. As a result, the model makes use of both previous and upcoming information.
The overall architecture of the model is shown in Figure 5.5.

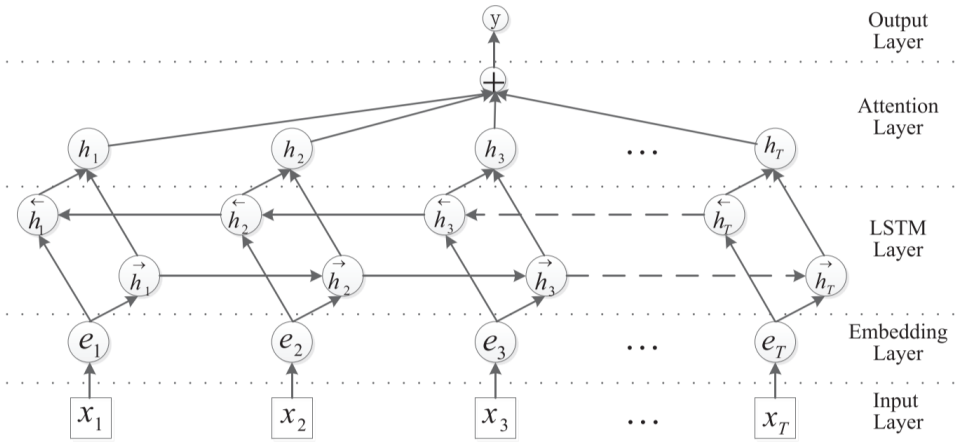


FIGURE 5.5: The model architecture to classify relation statements using Attention-Based Bidirectional Long Short-Term Memory Networks. [91]

The input of the model is a given sentence provided in the input layer. The embedding layer produces a low-dimensional vector of each word in the sentence. Then, as a real-valued vector, the sentence is fed into the following layer. The LSTM layer uses BLSTM to produce high-level features. The left and right sequence contexts, which are forward and backward pass respectively, are represented by two LSTM networks in the BLSTM network. Then, the element-wise sum operation merges the forward pass and backward pass outputs. After that, the Attention layer creates a weight vector and multiplies it to combine word-level features from each time step into a feature vector at the sentence level. Finally, in the output layer, a softmax classifier takes the hidden representation of the sentence and predicts the label based on the log-likelihood loss function. The results of classifying relation statements on the Attention-Based Bidirectional Long Short-Term Memory Network with respect to different training subsets are reported in Table A.5.

### 5.3.2 Relation Classification from Deep Transformer model

The objective is to learn how to classify relation statements in a supervised way using pre-trained language models. In order to utilize the pre-trained language models for relation extraction, we choose the R-BERT model [15] that addresses the relation classification task by using the pre-trained BERT language model as well as including information from the target entities and MTB model [14] which uses a novel objective (matching the blanks) to

learn more sophisticated relation representation than well-known BERT pre-training objectives.

**Enriching Pre-trained Language Model with Entity Information (R-BERT)** We have used the R-BERT model explained In section 4.4.2 to investigate the impact of training dataset size on this relation classification model. The results of fine-tuning the R-BERT model with respect to different training subsets are reported in Table A.6.

**Matching the Blanks: Distributional Similarity for Relation Learning (MTB)** We have used the MTB model explained In section 4.3.3 to investigate the impact of training dataset size on this relation classification model. The results of fine-tuning the MTB model with respect to different training subsets are reported in Table A.7.

# Chapter 6

# Experiments

In this section, the experimental results for classification models using the SemEval-2010 Task 8 dataset will be discussed. The experiments were performed on a Linux Ubuntu 22.04 personal computer with GPU NVIDIA GeForce RTX 2060 SUPER and 16GB memory (RAM).

## 6.1 Datasets

In this section, we will introduce the CNN corpus and the SemEval-2010 Task 8 dataset, which are employed in the pre-training and fine-tuning tasks, respectively.

### 6.1.1 CNN Corpus

We utilized CNN corpus [133] to pre-train the MTB model which is raw text and consists of 48487 samples.

### 6.1.2 The SemEval Dataset

The manually annotated SemEval-2010 Task 8 database [84] is the most used dataset for relation classification. This dataset contains entity pairs in sentences that were collected from the web and classified into nine distinct relation types. Hendrickx et al. [84] claim that the types were chosen to have little semantic overlap while yet covering a large number of relation mentions. As a result, categories like *Product-Producer* and *Content-Container* were created, which are relatively abstract semantic relations.

| Relaion | Freq | Pos | IAA |
|---|---|---|---|
| Cause-Effect (CE) | 1331 (12.4%) | 91.2% | 79.0% |
| Component-Whole (CW) | 1253 (11.7%) | 84.3% | 70.0% |
| Entity-Destination (ED) | 1137 (10.6%) | 80.1% | 75.2% |
| Entity-Origin (EO) | 974 (9.1%) | 69.2% | 58.2% |
| Product-Producer (PP) | 948 (8.8%) | 66.3% | 84.8% |
| Member-Collection (MC) | 923 (8.6%) | 74.7% | 68.2% |
| Message-Topic (MT) | 895 (8.4%) | 74.4% | 72.4% |
| Content-Container (CC) | 732 (6.8%) | 59.3% | 95.8% |
| Instrument-Agency (IA) | 660 (6.2%) | 60.8% | 65.0% |
| Other (O) | 1864 (17.4%) | N/A | N/A |

TABLE 6.1: Relations of SemEval-2010 Task 8 dataset. **Freq:** Absolute and relative frequency, **Pos:** percentage of positive samples in the candidate set, **IAA:** inter-annotator agreement.

Any entity pairs that don't fit into one of the nine relation types are included in the undirected "Other". This leads to a total of 19 classes since the order of the entities must be implied along with the relation type.

| Relaion | Entties |
|---|---|
| Cause-Effect(e1,e2) | That **machine** makes a lot of **noise**. |
| Instrument-Agency(e2,e1) | All **plants** absorb **carbon dioxide** for food. |
| Product-Producer(e2,e1) | The **machine** makes **bottle tops**. |
| Content-Container(e1, e2) | How do I recognize a **room** that contains **radioactive materials**? |

TABLE 6.2: A few examples from the SemEval-2010 Task 8 dataset.

Three annotation phases were performed: In the first phase, a pattern-based web search was manually crawled to find 1200 sentence candidates for each relation. In the second phase, these sentences were annotated by two separate annotators. Conflicts between the two annotators' annotations were resolved in the third round, or related sentences were eliminated if no agreement was reached. This method produced a total of 10717 annotated relation mentions, which were distributed unequally among the 10 relation types. As shown in table 6.2, the highest inter-annotator agreement was achieved for *Cause-Effect* (79.0%), and the lowest for *Instrument-Agency* (65%), implying that relation classification is a difficult and confusing task even for humans.

SemEval-2010 Task 8 has a total of 10717 annotated relation samples. The dataset is divided into two parts: training which includes 8000 sentences and testing which includes 2717 sentences The dataset has been utilized in several studies on the topic of standard relation classification since SemEval was released in 2010.

## 6.2 Evaluation Metrics

Relation extraction is described as a classification task in the domain of supervised methods, hence evaluation metrics like Accuracy, Precision, Recall, and F-Measure are used to assess performance. Precision and Recall are evaluated based on True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). If True Positive represents the total number of correctly predicted relation instances, True Positive + False Positive represents all predicted relation instances and True Positive + False Positive represents all relation instances, then the evaluation metrics define as:

$$Accuracy(Acc) = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.1}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{6.2}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{6.3}$$

$$F - Measure(F1) = \frac{2PR}{P + R} \tag{6.4}$$

## 6.3 Transferring Weights of The Relation Learning Task to The Relation Classification Task

In this part, we will report and analyze the results of fine-tuning MTB and R-BERT models using different pre-training models in three different scenarios. First, we will report the results of fine-tuning the R-BERT model using the pre-trained BERT.
Second, the results of fine-tuning the MTB model utilizing the learning weights transferred from the pre-training step with respect to the "Matching the blanks" and "Masked language model" objectives will be reported. The accuracy and

the loss of pre-training The MTB over the CNN corpus are illustrated in Figures A.3 and A.2.
Third, we will report the results of fine-tuning the R-BERT model utilizing the learning weights transferred from the pre-training step with respect to the "Matching the blanks" and "Masked language model" objectives. The results are recorded in table 6.4.

### 6.3.1 Parameters Settings

Table 6.3 includes the hyper-parameters of R-BERT and MTB models for performing both pre-training and fine-tuning tasks. We have set all hyper-parameters based on official implementations except the batch size of R-BERT in fine-tuning task which is changed from 32 to 8 to be able to deal with "CUDA out of memory" error. We have also limited the maximum number of epochs in the fine-tuning step for both the R-BERT and the MTB to 10 to be able to compare the results after specified epochs of training.

| Model | Task | Parameter value |
|---|---|---|
| R-BERT | Pre-training | bert-base-uncased |
| | Fine-tuning | Batch size = 8<br>Number of epochs = 1-10<br>Adam learning rate = 0.00002<br>Model = bert-base-uncased<br>Dropout rate=0.1<br>Gradient Accumulation steps = 1<br>Optimization steps = 9000 |
| MTB | Pre-training | Batch size = 32<br>Number of epochs = 18<br>Adam learning rate = 0.0001<br>Model = bert-base-uncased<br>Max norm = 1<br>Gradient Accumulation steps = 2 |
| | Fine-tuning | Batch size = 32<br>Number of epochs = 1-10<br>Adam learning rate = 0.00007<br>Model = bert-base-uncased<br>Max norm = 1<br>Gradient Accumulation steps = 2 |

TABLE 6.3: Parameter settings of the R-BERT and MTB models for performing pre-training and fine-tuning tasks.

### 6.3.2 Analysis and Discussion

Table 6.4 contains the results of R-BERT and MTB models after performing the relation classification task for ten epochs over the SemEval-2010 Task 8 dataset using different pre-training objectives.
The first row reports the results of fine-tuning the R-BERT model using pre-trained weights transferred from pre-trained BERT. The pre-training objective in this scenario is the well-known "Masked language model," which is performed over millions of web pages. We have utilized this rich knowledge only by downloading the pre-trained weights of the BERT language model.
The second row holds the results of fine-tuning the MTB model utilizing pre-trained weights transferred from pre-trained MTB. "Matching The Blanks," which is executed across the CNN corpus, is used as the pre-training target in this scenario.

The third row contains the results of fine-tuning the R-BERT model using pre-trained weights transferred from pre-trained MTB performed across the CNN corpus.

| **Pre-training Model** | **Fine-Tuning Model** | **Accuracy** | **Precision** | **Recall** | **F score** |
|---|---|---|---|---|---|
| Pre-trained BERT | R-BERT | 0.8381 | 0.7927 | 0.8316 | 0.8103 |
| Pre-training MTB on CNN corpus | MTB | **0.8445** | 0.7895 | 0.8069 | 0.7981 |
| Pre-training MTB on CNN corpus | R-BERT | 0.8425 | **0.8106** | **0.8353** | **0.8217** |

TABLE 6.4: Comparing the results of Fine-tuning R-BERT and MTB models over SemEval-2010 Task 8 after 10 epochs, using different training objectives with respect to Accuracy, Precision, Recall, and F-score.

Figures 6.1-A.6 illustrate the performance of three defined scenarios performing the relation classification task for one to ten epochs, concerning accuracy, precision, recall, and f-score, respectively.
Based on the f-score results, we claim that transferring pre-trained knowledge achieved from the "Matching The blanks" objective can enhance the performance of relation classification performed on the R-BERT model.

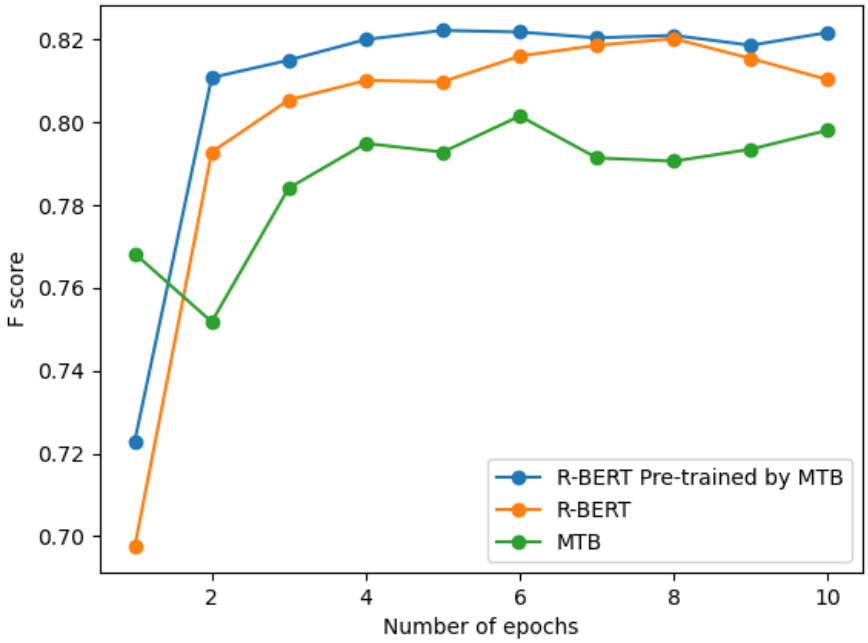


FIGURE 6.1: F-score results of the R-BERT model pre-trained by MTB, the R-BERT model, and the MTB model after 10 epochs of fine-tuning over the SemEval-2010 Task 8 dataset.

### 6.3.3 MTB vs. MTB + MLM

Since we used the bert-base-uncase to initialize the parameters of the Bert in the pre-training step of the MTB model, we decided to see how well the MTB classifies relations by only matching the blanks pre-training objective. As shown in Figure 6.2, the F-score of fine-tuning the MTB model using only matching the blanks is much lower than when we classify relation statements using both matching the blanks and masked language model objectives in the pre-training step. Notably, the masked language model is pre-trained over the CNN corpus, which is much smaller than the two enormous corpora the masked language model objective is pre-trained on. We believe that if we use a larger corpus than the CNN, the performance of fine-tuning the MTB model using only matching the blanks objective will improve.

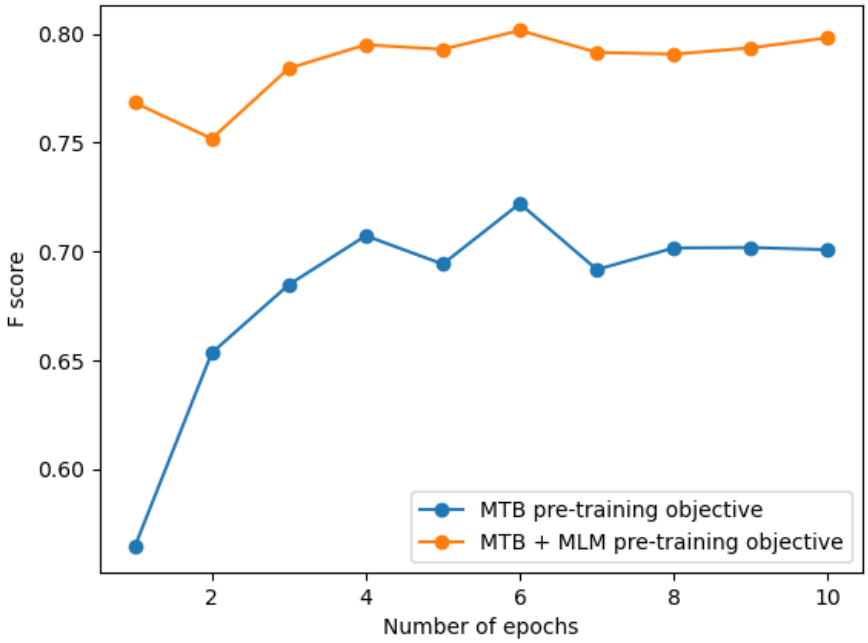


FIGURE 6.2: Comparison of MTB model's F-score when performing relation classification task using only Matching the blanks vs. using Matching the blanks + Masked language model pre-training objective.

## 6.4 Impact of Training Data Size on Relation Classification

In this part, we report and analyze the results of training four different architectures explained in 5.3 using different training data sizes.

### 6.4.1 Parameters Settings

For all four classification models, we have used default parameter values except the batch size of the R-BERT model which was 16 in the official implementation, but we changed it to 8 in order to be able to take care of the "CUDA out of memory" error. Parameter Values are listed in Table 6.5.

| Classification Model | Parameter Values |
|---|---|
| R-BERT | Batch size = 8<br>Number of epochs = 10<br>Adam learning rate = 0.00002<br>Model = bert-base-uncased<br>Dropout rate=0.1<br>Gradient Accumulation steps = 1<br>Optimization steps = 9000 |
| MTB | Batch size = 32<br>Number of epochs = 11<br>Learning rate = 0.00007<br>Model = bert-base-uncased |
| Attention-<br>Based<br>CNN | Batch size = 32<br>Number of epochs = 30<br>Word embedding size = 300<br>Word Position Embedding size = 5<br>Learning rate = 0.02<br>POS tag Embedding = 10<br>Word Window size = 100 |
| Attention-<br>Based<br>Bi-LSTM | Batch size = 10<br>Number of epochs = 30<br>Word embedding size = 100<br>Learning rate = 1.0<br>Word Window size = 100 |

TABLE 6.5: Classification models parameter values.

### 6.4.2 Analysis and Discussion

Table A.4-A.7 illustrate the performance of the Attention-Based CNN, Attention-Based Bi-LSTM, R-BERT, and MTB models, respectively in terms of accuracy, precision, recall, and f-score.

To investigate the effect of training data size on our models, we select different amounts of training data as shown in 5.1 and feed them to our models.

Table 6.6 provides some observations. First, we note that models trained on the subsets have an average F-score that varies from 0.1255% on SE.1 to 0.778% on SE.8.

Second, the table demonstrates that the models' average F-score across all subsets ranges from 0.3299% for MTB to 0.5643% for attention Bi-LSTM.

Third, it is also clear that classifier-specific standard deviations are higher than dataset-specific standard deviations, which can be interpreted as no matter what classification model is used, the model's performance is low for small subsets, and the performance gets higher for larger subsets. Also, the high classifier-specific standard deviation of all classification models means that changing the size of subsets affects all models' performance.

| Datasets | **Att-CNN** | **Att-BiLSTM** | **R-BERT** | **MTB** | **Avg.** | **Std. Dev.** |
|---|---|---|---|---|---|---|
| **SE.1** | 0.1608 | 0.2540 | 0.0758 | 0.0324 | 0.1307 | 0.0979 |
| **SE.2** | 0.1894 | 0.3769 | 0.1091 | 0.0915 | 0.1917 | 0.1305 |
| **SE.3** | 0.2697 | 0.3841 | 0.2774 | 0.2004 | 0.2829 | 0.0758 |
| **SE.4** | 0.4008 | 0.5741 | 0.5473 | 0.5098 | 0.508 | 0.0761 |
| **SE.5** | 0.5657 | 0.6531 | 0.6922 | 0.6768 | 0.6469 | 0.0565 |
| **SE.6** | 0.6606 | 0.7090 | 0.7622 | 0.7456 | 0.7193 | 0.045 |
| **SE.7** | 0.7433 | 0.7569 | 0.7916 | 0.7802 | 0.768 | 0.0219 |
| **SE.8** | 0.7907 | 0.8063 | 0.8103 | 0.7981 | 0.8013 | 0.0087 |
| **Avg.** | 0.4726 | 0.5643 | 0.5082 | 0.4793 | - | - |
| **Std. Dev.** | 0.2513 | 0.2029 | 0.3096 | 0.3231 | - | - |

TABLE 6.6: F-Score Table

Similar patterns in the performance of models can be seen in Tables A.8–A.10 regarding accuracy, precision, and recall. As an example, the average accuracy of classifiers varies from 23.05% on SE.1 to 79.62% on SE.8 (see Table A.8). The average precision of classifiers in Table A.9 ranges from 21.12% on SE.1 to 77.62% on SE.8, and the average recall of classifiers increases from 17.04% on SE.1 to 78.22% on SE.8, as shown in Table A.10.
Additionally, the classifiers' average accuracy across all of the subsets ranges from 47.87% reached by MTB to 59.20% accuracy by the attention Bi-LSTM. The average performance of classifiers for precision varies from 36.82% by MTB to 58.79% by attention Bi-LSTM all over the subsets and the average recall of classifiers across all subsets ranges from 31.35% by MTB to 60.45% by attention Bi-LSTM.
Moreover, the standard deviations between the classifiers are also higher than the standard deviations across all subsets, as shown in Tables A.8-A.10 which means that the size of the input dataset affects the performance of the

model more than the classification model. In other words, all mentioned classification models have similar behavior to the size of the training dataset.

Figures A.7-6.3 illustrate how well the classification models perform in terms of accuracy, precision, recall, and f-score, as the size of the training set is increased. The x-axis in the figures represents the size of the dataset, which is divided into two segments. Each line chart in all figures contains two segments that represent the result in two increment scenarios of dataset size. The first segment extends from SE.1 to 50% of the dataset size and illustrates the change in performance of the classifiers when trained on the first size increase case. The second segment of the line charts spans from 50% to 100% of the dataset size and shows how the classifier's performance changes in the second size increment scenario.

These figures provide several observations. First, for all four evaluation metrics, when the training set size is increased, all the classifiers show a performance trend that is very consistent. Second, the first size expansion scenario indicates a clear overall pattern of improving the performance of all classifiers with respect to all metrics, although figures show the performance of the classifiers almost doesn't change in the second segment, considering the point that half of the whole dataset is added into the training data.

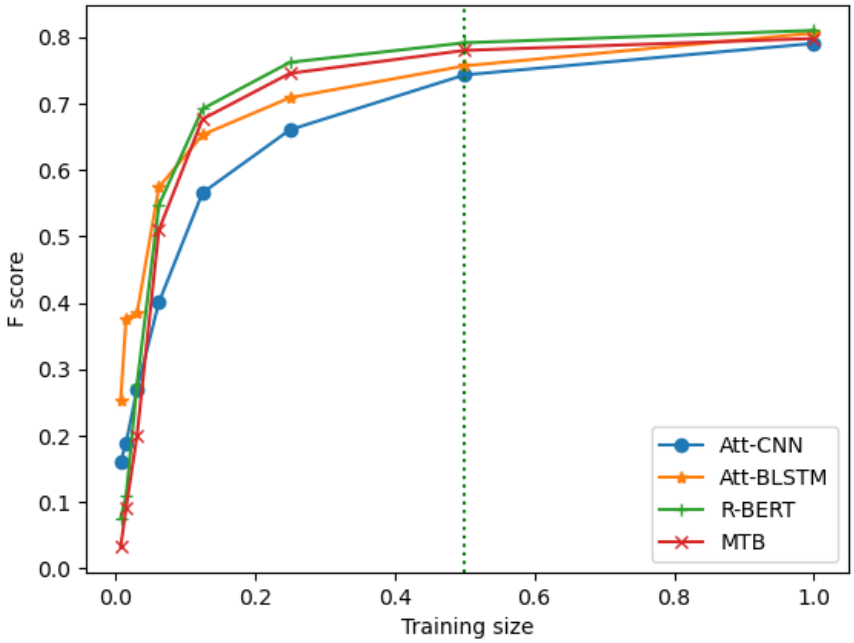


FIGURE 6.3: Impact of training data size on F-score of relation classification based on different architectures.

The findings shown in the figures are highly informative in several ways. They first demonstrate that a small dataset is not always a barrier to reaching high-performance results for a model, as the average F-score performance of classifiers was more than 70% on just 50% of the dataset. Second, the results show that, given a subset, classifiers perform similarly, although each classifier has changing performance across the small datasets. This is because the standard deviations among subsets are lower than the standard deviations across classifiers.

Figures A.10 and A.11 show the impact of training data size on training loss and test loss of att-BiLSTM and R-BERT model, respectively.

# Chapter 7

# Conclusion

This thesis begins by providing a brief overview of extracting useful information from text. Then we explain the importance of extracting semantic relationships from huge available raw text like web pages. After that, we give a short definition of some concepts related to relation extraction task such as neural networks, transfer learning, language model, knowledge bases, and name entity recognition. Next, we introduce relation extraction different methods like hand-built pattern methods, semi-supervised methods, supervised methods, unsupervised methods, and distant supervision. Afterward, we will go through related works for further review.

Subsequently, we focus on two deep transformer models utilized for classifying relations given entity annotated sentences. The first deep transformer model known as MTB focus on learning a new relation representation of a given entity-annotated sentence. By introducing the "matching the blanks" objective in the pre-training step, the semantic similarity of two input sentences can be measured based on whether two sentences share the same pair of entity mentions or not. The second deep transformer model, R-BERT, is a BERT-based model utilized for relation classification by applying the pre-trained "Masked Language model" and fine-tuning over the supervised SemEval-2010 task 8 dataset.

We have proposed our first contribution - transferring the pre-trained knowledge gained by completing the "matching the blanks" task on the CNN corpus to improve the results of the relation classification task on the R-BERT model. To do this, first, we recognize the entities of the CNN corpus by performing the NER and dependency tree parsing of subjects and objects. Then, the pre-training begins with feeding pairs of entity-annotated sentences to the model to measure the similarity score. When pre-training is done, we transfer the weights to the R-BERT model in order to improve the knowledge

of the pre-trained BERT model by knowing about the similarity between sentences. As we mentioned in Table 6.4, the performance of the R-BERT model improved after this knowledge transfer. We believe using an enormous corpus could affect the performance of the R-BERT model more than what we have experienced.

Our second contribution involves examining how the size of a training dataset affects the performance of a relation classification task. To achieve this, we utilized the SemEval-2010 task 8 labeled dataset and tested it on four models: attention-based CNN, attention-based BiLSTM, R-BERT, and MTB, which are all deep neural network models. Based on the proposed results, we see that the impact of training dataset size on the performance of models reduce as much as the size of the training set increase regardless of what kind of classification model is used. So, it is a model-agnostic behavior.
Finally, we provide a report and analysis of our experiments, followed by a discussion of the results.

# Appendix A

# More Results

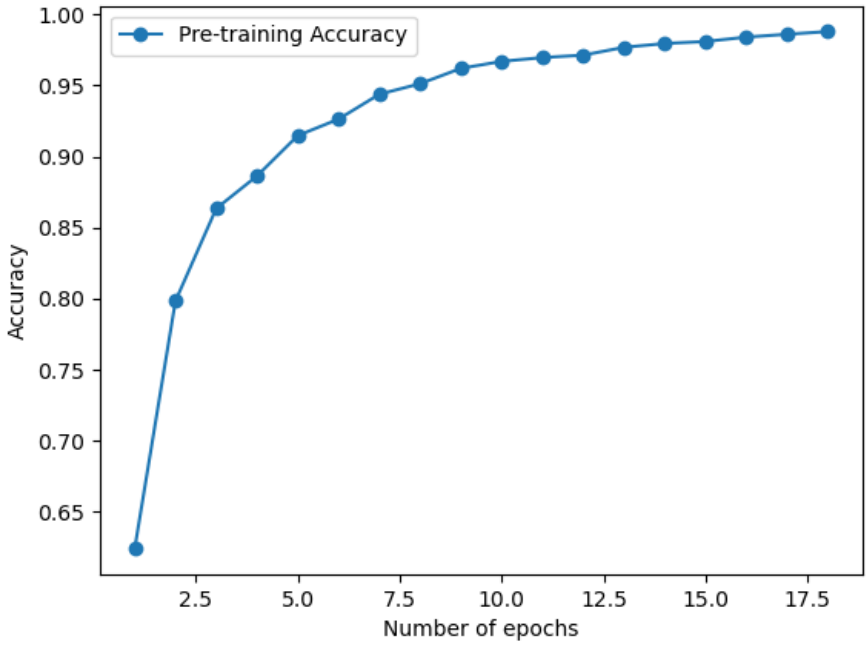


FIGURE A.1: The accuracy of pre-training CNN corpus by MTB model.

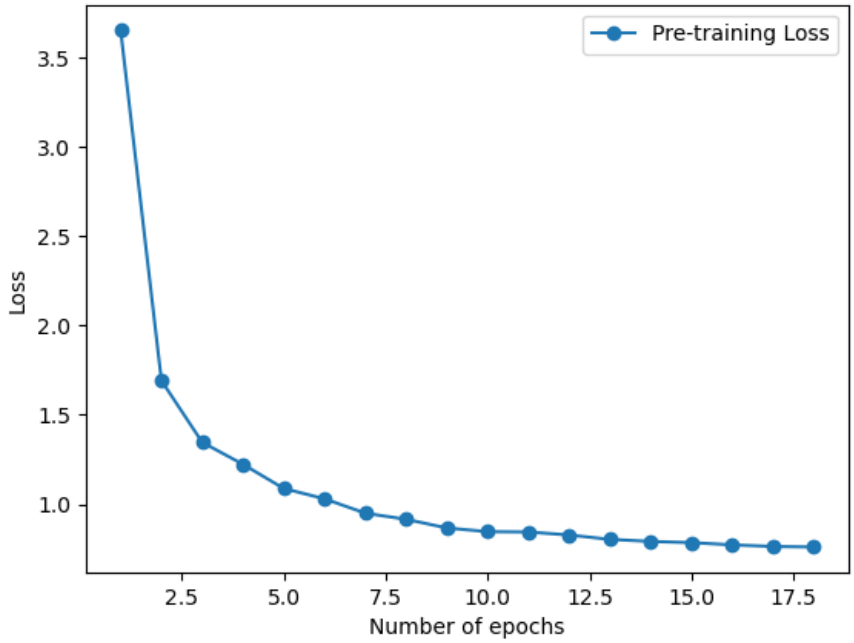


FIGURE A.2: The MLM + MTB loss of pre-training CNN corpus by MTB model.

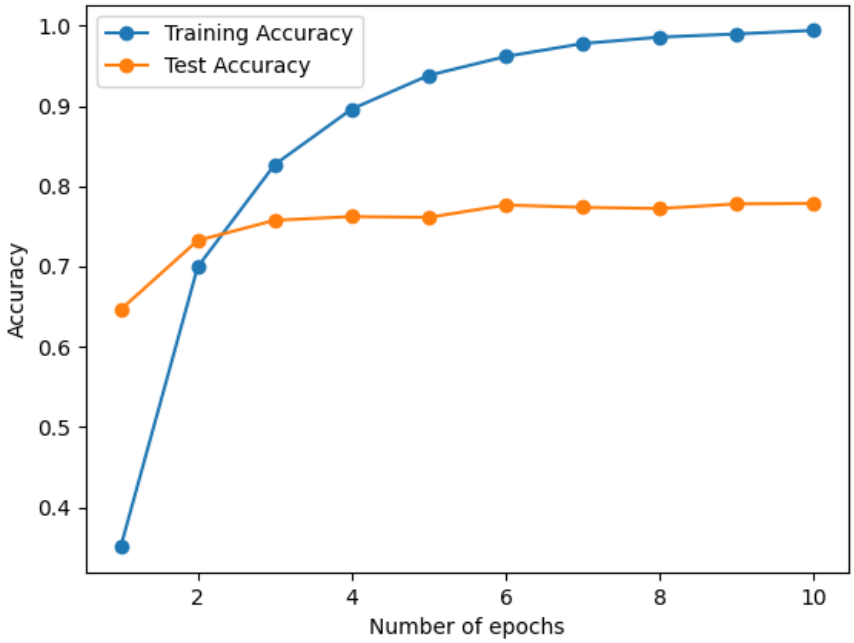


FIGURE A.3: The accuracy of Fine-tuning SemEval-2010 task 8 by MTB model.

| **Number of Epochs** | **Evaluation Metric** | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F score |
| 1 | 0.7825 | 0.7045 | 0.7203 | 0.6974 |
| 2 | 0.8233 | 0.7713 | 0.8177 | 0.7928 |
| 3 | 0.8344 | 0.7872 | 0.8262 | 0.8054 |
| 4 | 0.8395 | 0.7918 | 0.8309 | 0.8101 |
| 5 | 0.8392 | 0.7922 | 0.8307 | 0.8098 |
| 6 | 0.8436 | 0.7977 | 0.8381 | 0.8160 |
| 7 | 0.8420 | 0.7999 | 0.8402 | 0.8186 |
| 8 | 0.8458 | 0.8069 | 0.8414 | **0.8202** |
| 9 | 0.8410 | 0.7948 | 0.8387 | 0.8154 |
| 10 | 0.8381 | 0.7927 | 0.8316 | 0.8103 |

TABLE A.1: The results of the R-BERT model. Using the pre-trained BERT model, Fine-tuned by the SemEval-2010 Task 8 with respect to Accuracy, Precision, Recall, and F-score.

| **Number of Epochs** | **Evaluation Metric** | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F score |
| 1 | 0.8130 | 0.776 | 0.7607 | 0.7683 |
| 2 | 0.8178 | 0.7326 | 0.772 | 0.7518 |
| 3 | 0.8343 | 0.7952 | 0.7733 | 0.7841 |
| 4 | 0.8366 | 0.7933 | 0.7965 | 0.7949 |
| 5 | 0.8332 | 0.7748 | 0.8117 | 0.7928 |
| 6 | 0.8414 | 0.7938 | 0.8093 | 0.8015 |
| 7 | 0.8414 | 0.7886 | 0.7942 | 0.7914 |
| 8 | 0.8431 | 0.7893 | 0.7918 | 0.7906 |
| 9 | 0.8460 | 0.7879 | 0.7992 | 0.7935 |
| 10 | 0.8445 | 0.7895 | 0.8069 | 0.7981 |

TABLE A.2: The results of the MTB model. Pre-trained by the CNN corpus and Fine-tuned by the SemEval-2010 Task 8 with respect to Accuracy, Precision, Recall, and F-score.

| **Number of Epochs** | **Evaluation Metric** | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F score |
| 1 | 0.7965 | 0.7234 | 0.7404 | 0.7229 |
| 2 | 0.8373 | 0.7960 | 0.8269 | 0.8108 |
| 3 | 0.8417 | 0.7999 | 0.8323 | 0.8150 |
| 4 | 0.8447 | 0.8107 | 0.8306 | 0.8200 |
| 5 | 0.8465 | 0.8096 | 0.8366 | **0.8222** |
| 6 | 0.8480 | 0.8108 | 0.8359 | 0.8218 |
| 7 | 0.8454 | 0.8095 | 0.8339 | 0.8204 |
| 8 | 0.8473 | 0.8031 | 0.8435 | 0.8210 |
| 9 | 0.8447 | 0.8083 | 0.8315 | 0.8186 |
| 10 | 0.8425 | 0.8106 | 0.8353 | 0.8217 |

TABLE A.3: The results of the R-BERT model, pre-trained on the MTB by CNN corpus and Fine-tuned by the SemEval-2010 Task 8 with respect to Accuracy, Precision, Recall, and F-score.

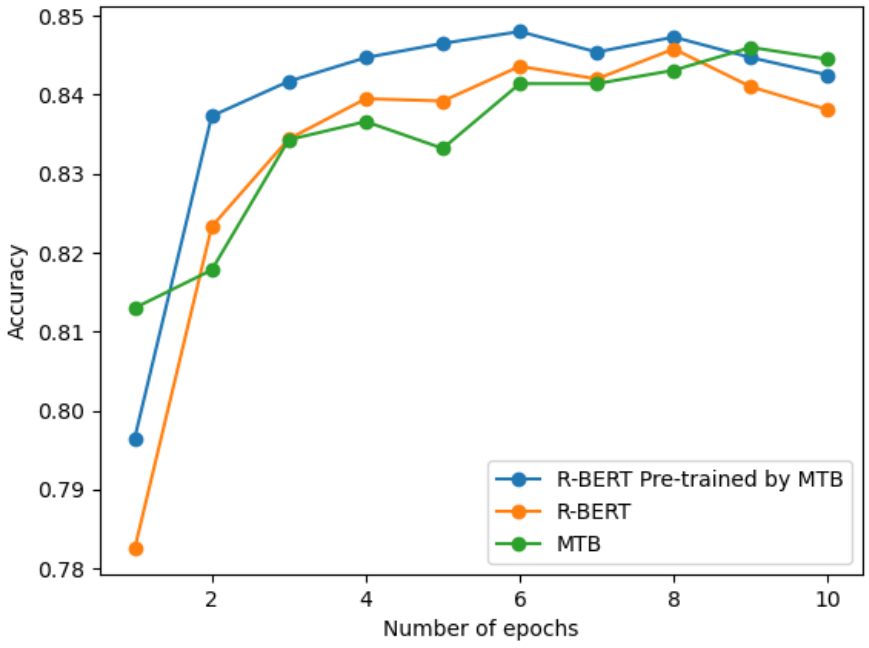


FIGURE A.4: Accuracy results of the R-BERT model pre-trained by MTB, the R-BERT model, and the MTB model after 10 epochs of fine-tuning over the SemEval-2010 Task 8 dataset.

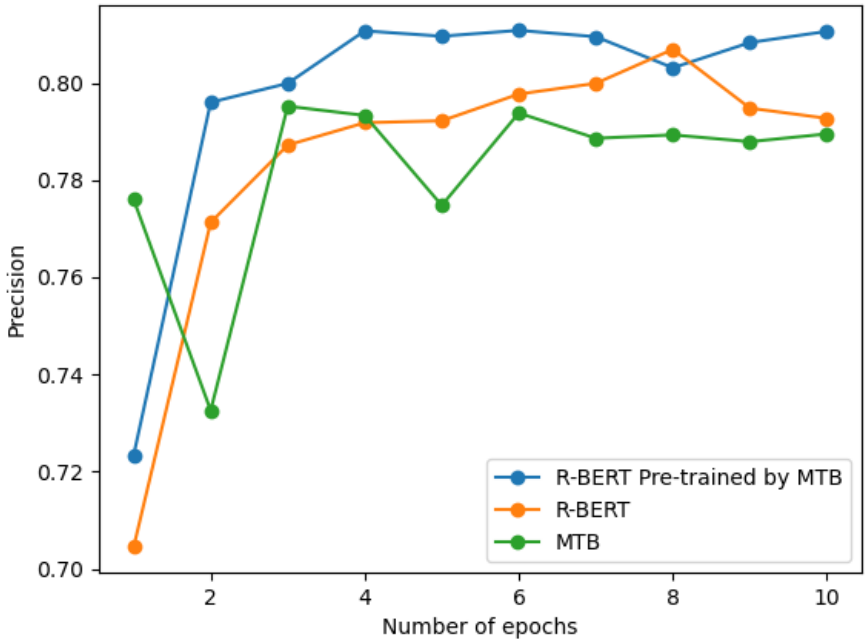


FIGURE A.5: Precision results of the R-BERT model pre-trained by MTB, the R-BERT model, and the MTB model after 10 epochs of fine-tuning over the SemEval-2010 Task 8 dataset.

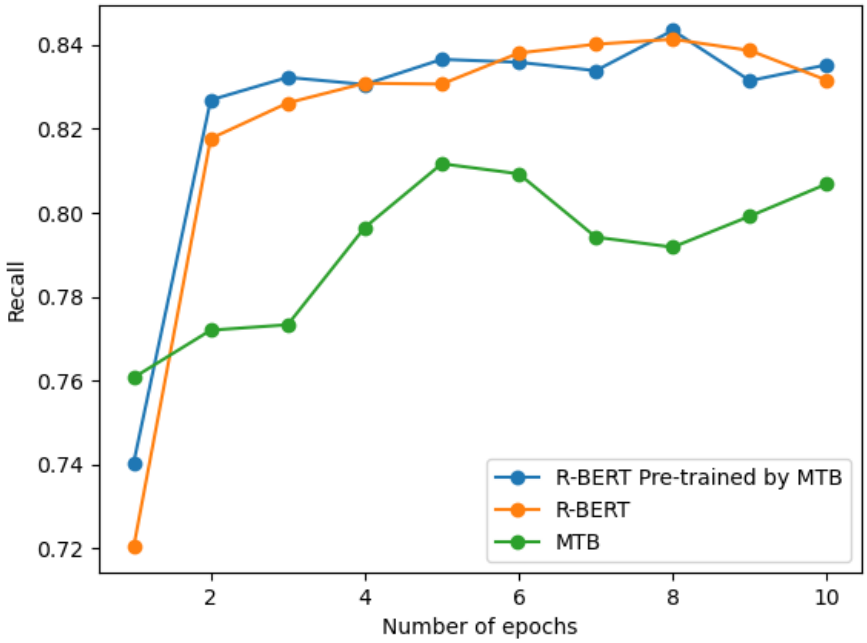


FIGURE A.6: Recall results of the R-BERT model pre-trained by MTB, the R-BERT model, and the MTB model after 10 epochs of fine-tuning over the SemEval-2010 Task 8 dataset.

| Portion | Accuracy | Precision | Recall | F score |
|---|---|---|---|---|
| 1/128 | 0.2267 | 0.2123 | 0.2013 | 0.1608 |
| 1/64 | 0.2591 | 0.1617 | 0.2517 | 0.1894 |
| 1/32 | 0.3710 | 0.2656 | 0.3050 | 0.2697 |
| 1/16 | 0.4844 | 0.4094 | 0.4087 | 0.4008 |
| 1/8 | 0.5841 | 0.5705 | 0.5901 | 0.5657 |
| 1/4 | 0.6621 | 0.6435 | 0.6907 | 0.6606 |
| 1/2 | 0.7346 | 0.7431 | 0.7461 | 0.7433 |
| 1 | 0.7770 | 0.7936 | 0.7900 | 0.7907 |

TABLE A.4: Impact of dataset size on the performance of the Attention-Based CNN model with respect to Accuracy, Precision, Recall, and F-score.

| Portion | Accuracy | Precision | Recall | F score |
|---|---|---|---|---|
| 1/128 | 0.3033 | 0.3424 | 0.3371 | 0.2540 |
| 1/64 | 0.4520 | 0.4203 | 0.3928 | 0.3769 |
| 1/32 | 0.4556 | 0.4192 | 0.4211 | 0.3841 |
| 1/16 | 0.5999 | 0.5904 | 0.6059 | 0.5741 |
| 1/8 | 0.6632 | 0.6528 | 0.7361 | 0.6531 |
| 1/4 | 0.7229 | 0.7065 | 0.7607 | 0.7090 |
| 1/2 | 0.7508 | 0.7607 | 0.7780 | 0.7569 |
| 1 | 0.7887 | 0.8115 | 0.8050 | 0.8063 |

TABLE A.5: Impact of dataset size on the performance of the Attention-Based Bi-LSTM model with respect to Accuracy, Precision, Recall, and F-score.

| Portion | Accuracy | Precision | Recall | F score |
|---|---|---|---|---|
| 1/128 | 0.1987 | 0.2471 | 0.1334 | 0.0758 |
| 1/64 | 0.2602 | 0.2628 | 0.1230 | 0.1091 |
| 1/32 | 0.4133 | 0.3412 | 0.2873 | 0.2774 |
| 1/16 | 0.6496 | 0.6491 | 0.5657 | 0.5473 |
| 1/8 | 0.7298 | 0.7144 | 0.6889 | 0.6922 |
| 1/4 | 0.7957 | 0.7512 | 0.7769 | 0.7622 |
| 1/2 | 0.8241 | 0.7741 | 0.8129 | 0.7916 |
| 1 | 0.8381 | 0.7927 | 0.8316 | 0.8103 |

TABLE A.6: Impact of dataset size on the performance of the R-BERT model with respect to Accuracy, precision, recall, and f-score.

| **Portion** | **Accuracy** | **Precision** | **Recall** | **F score** |
|---|---|---|---|---|
| 1/128 | 0.1947 | 0.1934 | 0.1105 | 0.0324 |
| 1/64 | 0.2202 | 0.2328 | 0.1205 | 0.0915 |
| 1/32 | 0.3833 | 0.3212 | 0.2573 | 0.2004 |
| 1/16 | 0.5946 | 0.5784 | 0.5122 | 0.5098 |
| 1/8 | 0.7102 | 0.6915 | 0.6599 | 0.6768 |
| 1/4 | 0.7838 | 0.7457 | 0.7349 | 0.7456 |
| 1/2 | 0.8311 | 0.7681 | 0.7829 | 0.7802 |
| 1 | 0.8445 | 0.7895 | 0.8069 | 0.7981 |

TABLE A.7: Impact of dataset size on the performance of the MTB model with respect to Accuracy, Precision, Recall, and F-score.

| Datasets | **Att-CNN** | **Att-BiLSTM** | **R-BERT** | **MTB** | **Avg.** | **Std. Dev.** |
|---|---|---|---|---|---|---|
| **SE.1** | 0.2267 | 0.3033 | 0.1987 | 0.1947 | 0.2308 | 0.0503 |
| **SE.2** | 0.2591 | 0.4520 | 0.2602 | 0.2202 | 0.2978 | 0.1044 |
| **SE.3** | 0.3710 | 0.4556 | 0.4133 | 0.3833 | 0.4058 | 0.0376 |
| **SE.4** | 0.4844 | 0.5999 | 0.6496 | 0.5946 | 0.5821 | 0.0697 |
| **SE.5** | 0.5841 | 0.6632 | 0.7298 | 0.7102 | 0.6718 | 0.0648 |
| **SE.6** | 0.6621 | 0.7229 | 0.7957 | 0.7838 | 0.7411 | 0.0615 |
| **SE.7** | 0.7346 | 0.7508 | 0.8241 | 0.8311 | 0.7851 | 0.0495 |
| **SE.8** | 0.7770 | 0.7887 | 0.8381 | 0.8445 | 0.8120 | 0.0341 |
| **Avg.** | 0.5123 | 0.5920 | 0.5886 | 0.5703 | - | - |
| **Std. Dev.** | 0.2116 | 0.1722 | 0.2604 | 0.2692 | - | - |

TABLE A.8: Accuracy Table of performing Att-CNN, Att-BiLSTM, R-BERT, and MTB models on different portions of the training dataset.

| Datasets | **Att-CNN** | **Att-BiLSTM** | **R-BERT** | **MTB** | **Avg.** | **Std. Dev.** |
|---|---|---|---|---|---|---|
| **SE.1** | 0.2123 | 0.3424 | 0.2471 | 0.1934 | 0.2488 | 0.0662 |
| **SE.2** | 0.1617 | 0.4203 | 0.2628 | 0.2328 | 0.2694 | 0.1091 |
| **SE.3** | 0.2656 | 0.4192 | 0.3412 | 0.3212 | 0.3368 | 0.0635 |
| **SE.4** | 0.4094 | 0.5904 | 0.6491 | 0.5784 | 0.5568 | 0.103 |
| **SE.5** | 0.5705 | 0.6528 | 0.7144 | 0.6915 | 0.6573 | 0.0632 |
| **SE.6** | 0.6435 | 0.7065 | 0.7512 | 0.7457 | 0.7117 | 0.0496 |
| **SE.7** | 0.7431 | 0.7607 | 0.7741 | 0.7681 | 0.7615 | 0.0134 |
| **SE.8** | 0.7936 | 0.8115 | 0.7927 | 0.7895 | 0.7968 | 0.0099 |
| **Avg.** | 0.4749 | 0.5879 | 0.5665 | 0.54 | - | - |
| **Std. Dev.** | 0.2467 | 0.1752 | 0.2396 | 0.2516 | - | - |

TABLE A.9: Precision Table of performing Att-CNN, Att-BiLSTM, R-BERT, and MTB models on different portions of the training dataset.

| Datasets | **Att-CNN** | **Att-BiLSTM** | **R-BERT** | **MTB** | **Avg.** | **Std. Dev.** |
|---|---|---|---|---|---|---|
| **SE.1** | 0.2013 | 0.3371 | 0.1334 | 0.1105 | 0.1955 | 0.1019 |
| **SE.2** | 0.2517 | 0.3928 | 0.1230 | 0.1205 | 0.2219 | 0.1293 |
| **SE.3** | 0.3050 | 0.4211 | 0.2873 | 0.2573 | 0.3176 | 0.0717 |
| **SE.4** | 0.4087 | 0.6059 | 0.5657 | 0.5122 | 0.5231 | 0.0853 |
| **SE.5** | 0.5901 | 0.7361 | 0.6889 | 0.6599 | 0.6687 | 0.0611 |
| **SE.6** | 0.6907 | 0.7607 | 0.7769 | 0.7349 | 0.7408 | 0.0376 |
| **SE.7** | 0.7461 | 0.7780 | 0.8129 | 0.7829 | 0.7799 | 0.0273 |
| **SE.8** | 0.7900 | 0.8050 | 0.8316 | 0.8069 | 0.8083 | 0.0172 |
| **Avg.** | 0.4979 | 0.6045 | 0.5274 | 0.4981 | - | - |
| **Std. Dev.** | 0.2349 | 0.1934 | 0.3024 | 0.2951 | - | - |

TABLE A.10: Recall Table of performing Att-CNN, Att-BiLSTM, R-BERT, and MTB models on different portions of the training dataset.

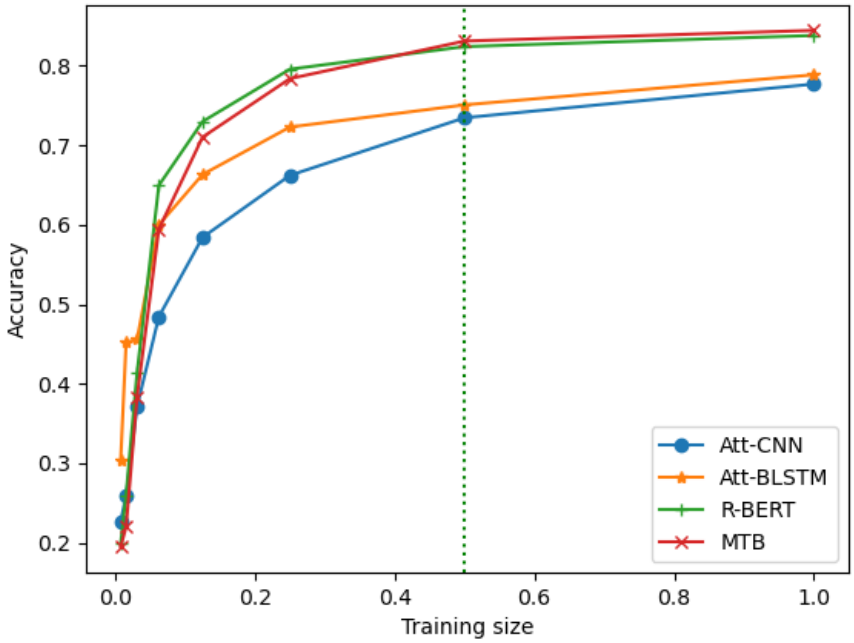


FIGURE A.7: Impact of training data size on Accuracy of relation classification based on different architectures.

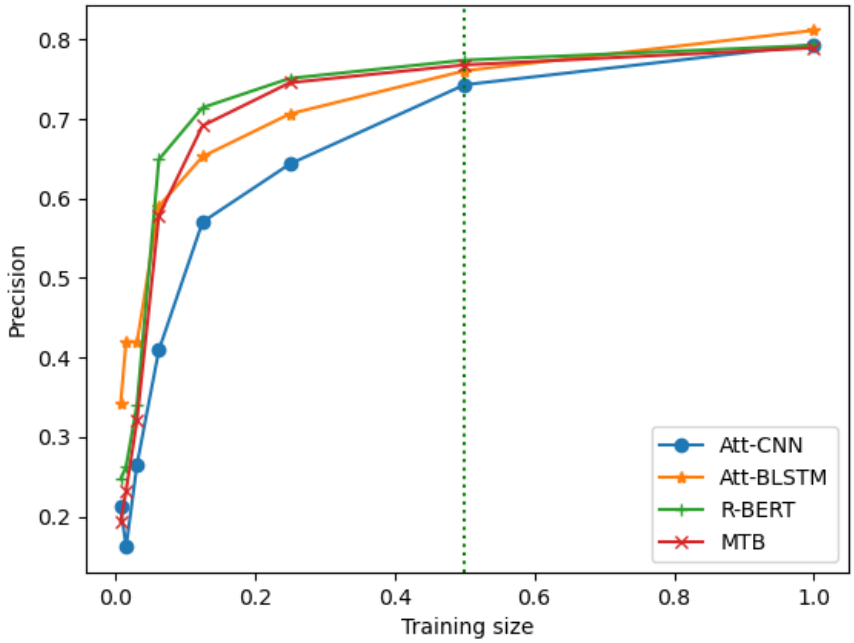


FIGURE A.8: Impact of training data size on Precision of relation classification based on different architectures.

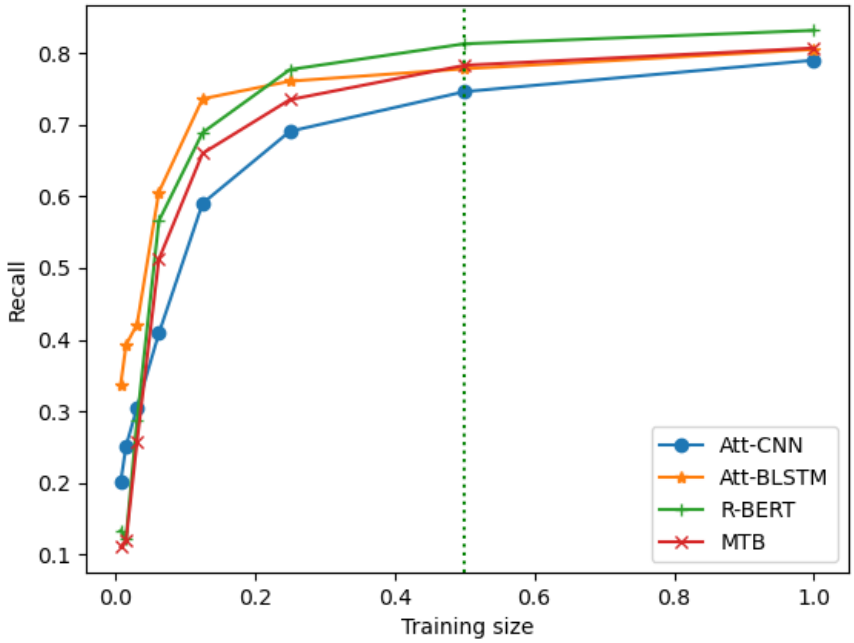


FIGURE A.9: Impact of training data size on Recall of relation classification based on different architectures.

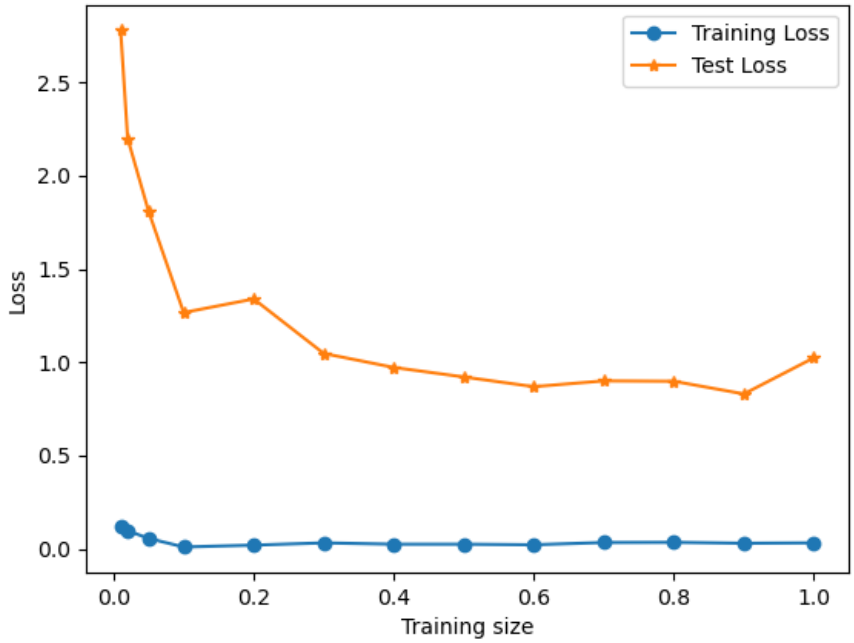


FIGURE A.10: Impact of training data size on training loss and test loss of Attention-Based Bidirectional Long Short-Term Memory Network for relation classification.

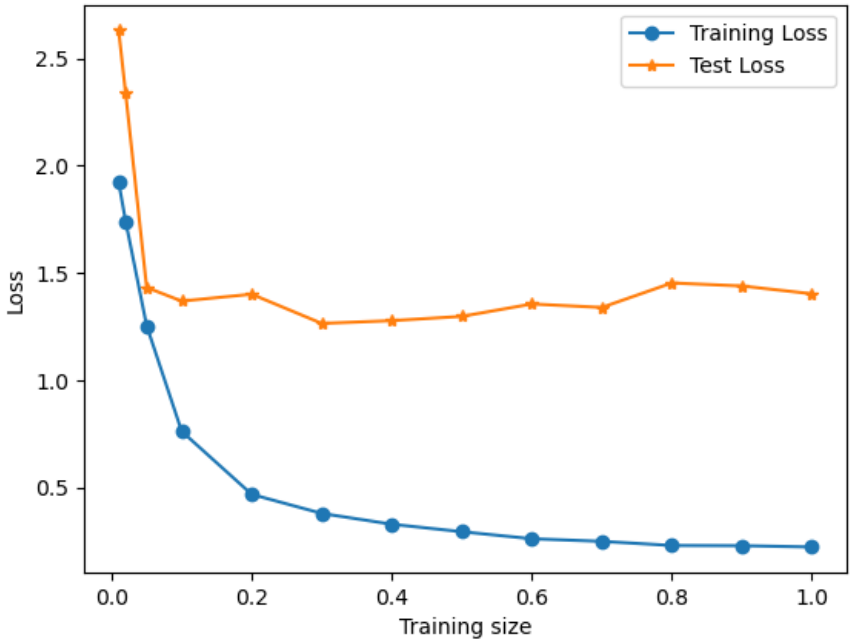


FIGURE A.11: Impact of training data size on training loss and test loss of R-BERT model for relation classification.

# References


[1] Christoph Benedikt Alt. *Neural sequential transfer learning for relation extraction*. Technische Universitaet Berlin (Germany), 2021.

[2] Mike Mintz, Steven Bills, Rion Snow, and Daniel 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*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011. URL: https://aclanthology.org/P09-1113.

[3] Nguyen Bach and Sameer Badaskar. "A Review of Relation Extraction". In: (May 2011).

[4] Natalia Konstantinova. "Review of Relation Extraction Methods: What Is New Out There?" In: vol. 436. Apr. 2014, pp. 15–28. ISBN: 978-3-319-12579-4. DOI: 10.1007/978-3-319-12580-0_2.

[5] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. "Relation classification via convolutional deep neural network". In: *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*. 2014, pp. 2335–2344.

[6] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. "Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1753–1762. DOI: 10.18653/v1/D15-1203. URL: https://aclanthology.org/D15-1203.

[7] Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. "Classifying relations by ranking with convolutional neural networks". In: *arXiv preprint arXiv:1504.06580* (2015).

[8] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. "Bidirectional long short-term memory networks for relation classification".

In: *Proceedings of the 29th Pacific Asia conference on language, information and computation*. 2015, pp. 73–78.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 2008, pp. 1247–1250.

[11] Georgi Kobilarov, Chris Bizer, Sören Auer, and Jens Lehmann. "Dbpedia–a linked data hub and data source for web applications and enterprises". In: (2009).

[12] Denny Vrandečić and Markus Krötzsch. *Wikidata: a free collaborative knowledgebase*. 2014.

[13] Kang Liu. "A survey on neural relation extraction". In: *Science China Technological Sciences* 63 (Oct. 2020), pp. 1971–1989. DOI: 10.1007/s11431-020-1673-6.

[14] Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. *Matching the Blanks: Distributional Similarity for Relation Learning*. 2019. DOI: 10.48550/ARXIV.1906.03158. URL: https://arxiv.org/abs/1906.03158.

[15] Shanchan Wu and Yifan He. "Enriching pre-trained language model with entity information for relation classification". In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 2361–2364.

[16] Tapas Nayak. "Deep neural networks for relation extraction". PhD thesis. National University of Singapore (Singapore), 2021.

[17] *Structure of a feed-forward neural network.* URL: https://upload.wikimedia.org/wikipedia/commons/c/c2/MultiLayerNeuralNetworkBigger_english.png.

[18] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[19] Mahdi Hashemi. "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation". In: *Journal of Big Data* 6 (Nov. 2019). DOI: 10.1186/s40537-019-0263-7.

[20] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[21] *Structure of a recurrent neural network.* URL: https://databricks.com/wp-content/uploads/2019/02/neural2.jpg.

[22] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[23] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[24] *Structure of LSTM.* URL: https://pluralsight2.imgix.net/guides/8a8ac7c1-8bac-4e89-ace8-9e28813ab635_3.JPG.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[27] *Structure of GCN.* URL: https://tkipf.github.io/graph-convolutional-networks/.

[28] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[29] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[30] Guo-Jun Qi and Jiebo Luo. "Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2020), pp. 2168–2187.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[33] *Network Architecture of CBOW and Skip-gram Word Embedding models.* URL: https://www.sallys.space/image/word2vec/1.jpg.

[34] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[35] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[36] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books*. 2015. arXiv: 1506.06724 [cs.CV].

[37] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 697–706.

[38] Tom Mitchell and Edward Fredkin. "Never-ending language learning". In: *2014 IEEE International Conference on Big Data (Big Data)*. IEEE. 2014, pp. 1–1.

[39] Ralph Grishman and Beth M Sundheim. "Message understanding conference-6: A brief history". In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. 1996.

[40] Erik F. Tjong Kim Sang. "Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition". In: *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. 2002. URL: https://aclanthology.org/W02-2024.

[41] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147. URL: https://aclanthology.org/W03-0419.

[42] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. "The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation". In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Lisbon, Portugal: European Language Resources Association (ELRA), May 2004. URL: http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

[43] GuoDong Zhou and Jian Su. "Named Entity Recognition using an HMM-based Chunk Tagger". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 473–480. DOI: 10.3115/1073083.1073163. URL: https://aclanthology.org/P02-1060.

[44] Robert Malouf. "Markov Models for Language-independent Named Entity Recognition". In: *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. 2002. URL: https://aclanthology.org/W02-2019.

[45] Xavier Carreras, Lluis Marquez, and Lluıs Padró. "Named entity extraction using adaboost". In: *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. 2002.

[46] Koichi Takeuchi and Nigel Collier. "Use of Support Vector Machines in Extended Named Entity Recognition". In: *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. 2002. URL: https://aclanthology.org/W02-2029.

[47] Hai Leong Chieu and Hwee Tou Ng. "Named Entity Recognition: A Maximum Entropy Approach Using Global Information". In: *COLING 2002: The 19th International Conference on Computational Linguistics*. 2002. URL: https://aclanthology.org/C02-1025.

[48] Hai Leong Chieu and Hwee Tou Ng. "Named Entity Recognition with a Maximum Entropy Approach". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 160–163. URL: https://aclanthology.org/W03-0423.

[49] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[50] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural language processing (almost) from scratch". In: *Journal of machine learning research* 12.ARTICLE (2011), pp. 2493–2537.

[51] Zhiheng Huang, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991* (2015).

[52] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074. DOI: 10.18653/v1/P16-1101. URL: https://aclanthology.org/P16-1101.

[53] Jason PC Chiu and Eric Nichols. "Named entity recognition with bidirectional LSTM-CNNs". In: *Transactions of the association for computational linguistics* 4 (2016), pp. 357–370.

[54] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. "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*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030. URL: https://aclanthology.org/N16-1030.

[55] Jana Straková, Milan Straka, and Jan Hajic. "Neural Architectures for Nested NER through Linearization". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5326–5331. DOI: 10.18653/v1/P19-1527. URL: https://aclanthology.org/P19-1527.

[56] Tiange Xu and Fu Zhang. "A Brief Review of Relation Extraction Based on Pre-Trained Language Models." In: *FSDM* (2020), pp. 775–789.

[57] Hailin Wang, Ke Qin, Rufai Yusuf, Guoming Lu, and Jin Yin. "Deep neural network-based relation extraction: an overview". In: *Neural Computing and Applications* 34 (Mar. 2022), pp. 1–21. DOI: 10.1007/s00521-021-06667-3.

[58] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. "Discovering relations among named entities from large corpora". In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (acl-04)*. 2004, pp. 415–422.

[59] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. "Web-scale information extraction in knowitall: (preliminary results)". In: *Proceedings of the 13th international conference on World Wide Web*. 2004, pp. 100–110.

[60] Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. "TextRunner: Open Information Extraction on the Web". In: *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. Rochester, New York, USA: Association for Computational Linguistics, Apr. 2007, pp. 25–26. URL: https://aclanthology.org/N07-4013.

[61] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. "Open Information Extraction: The Second Generation". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 3–10. ISBN: 9781577355137.

[62] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. "Semantic Role Labeling for Open Information Extraction". In: *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Los Angeles, California: Association for Computational Linguistics, June 2010, pp. 52–60. URL: https://aclanthology.org/W10-0907.

[63] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. "Open Language Learning for Information Extraction". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 523–534. URL: https://aclanthology.org/D12-1048.

[64] Harinder Pal and Mausam. "Demonyms and Compound Relational Nouns in Nominal Open IE". In: *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. San Diego, CA: Association for Computational Linguistics, June 2016, pp. 35–39. DOI: 10.18653/v1/W16-1307. URL: https://aclanthology.org/W16-1307.

[65] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. "The Importance of Syntactic Parsing and Inference in Semantic Role Labeling". In: *Computational Linguistics* 34.2 (2008), pp. 257–287. DOI: 10.1162/coli.2008.34.2.257. URL: https://aclanthology.org/J08-2005.

[66] Sebastian Riedel, Limin Yao, and Andrew McCallum. "Modeling Relations and Their Mentions without Labeled Text". In: *ECML/PKDD*. 2010.

[67] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. "Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 541–550. URL: https://aclanthology.org/P11-1055.

[68] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. "Multi-instance Multi-label Learning for Relation Extraction". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*

*Learning*. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 455–465. URL: https://aclanthology.org/D12-1042.

[69] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. "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)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2124–2133. DOI: 10.18653/v1/P16-1200. URL: https://aclanthology.org/P16-1200.

[70] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. "Noise Mitigation for Neural Entity Typing and Relation Extraction". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1183–1194. URL: https://aclanthology.org/E17-1111.

[71] Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. "RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1257–1266. DOI: 10.18653/v1/D18-1157. URL: https://aclanthology.org/D18-1157.

[72] Shanchan Wu, Kai Fan, and Qiong Zhang. "Improving Distantly Supervised Relation Extraction with Neural Noise Converter and Conditional Optimal Selector". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 7273–7280. DOI: 10.1609/aaai.v33i01.33017273. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4713.

[73] Zhi-Xiu Ye and Zhen-Hua 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2810–2819. DOI: 10.18653/v1/N19-1288. URL: https://aclanthology.org/N19-1288.

[74] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. "Cotype: Joint extraction of typed entities and relations with knowledge bases". In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1015–1024.

[75] Pengda Qin, Weiran Xu, and William Yang Wang. "DSGAN: Generative Adversarial Training for Distant Supervision Relation Extraction". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 496–505. DOI: 10.18653/v1/P18-1046. URL: https://aclanthology.org/P18-1046.

[76] Pengda Qin, Weiran Xu, and William Yang Wang. "Robust distant supervision relation extraction via deep reinforcement learning". In: *arXiv preprint arXiv:1805.09927* (2018).

[77] Zhengqiu He, Wenliang Chen, Yuyi Wang, Wei Zhang, Guanchun Wang, and Min Zhang. "Improving Neural Relation Extraction with Positive and Unlabeled Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 7927–7934. DOI: 10.1609/aaai.v34i05.6300. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6300.

[78] Yuming Shang, He-Yan Huang, Xian-Ling Mao, Xin Sun, and Wei Wei. "Are Noisy Sentences Useless for Distant Supervised Relation Extraction?" In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 8799–8806. DOI: 10.1609/aaai.v34i05.6407. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6407.

[79] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. "Zero-Shot Relation Extraction via Reading Comprehension". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 333–342. DOI: 10.18653/v1/K17-1034. URL: https://aclanthology.org/K17-1034.

[80] Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. "Entity-Relation Extraction as Multi-Turn Question Answering". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1340–1350. DOI: 10.18653/v1/P19-1129. URL: https://aclanthology.org/P19-1129.

[81] Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. "WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1535–1545. DOI: 10.18653/v1/P16-1145. URL: https://aclanthology.org/P16-1145.

[82] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. "Bidirectional attention flow for machine comprehension". In: *arXiv preprint arXiv:1611.01603* (2016).

[83] Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. "FewRel 2.0: Towards More Challenging Few-Shot Relation Classification". 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6250–6255. DOI: 10.18653/v1/D19-1649. URL: https://aclanthology.org/D19-1649.

[84] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. "SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 33–38. URL: https://aclanthology.org/S10-1006.

[85] Dan Roth and Wen-tau Yih. "A Linear Programming Formulation for Global Inference in Natural Language Tasks". In: *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*. Boston, Massachusetts, USA: Association for Computational Linguistics, May 2004, pp. 1–8. URL: https://aclanthology.org/W04-2401.

[86] Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. "Improving distantly supervised relation extraction using word and entity based attention". In: *arXiv preprint arXiv:1804.06987* (2018).

[87] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. "Position-aware Attention and Supervised Data Improve Slot Filling". In: *Proceedings of the 2017 Conference on Empirical*

*Methods in Natural Language Processing (EMNLP 2017)*. 2017, pp. 35–45. URL: https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

[88] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. "Word Representations: A Simple and General Method for Semi-Supervised Learning". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 384–394. URL: https://aclanthology.org/P10-1040.

[89] Yatian Shen and Xuanjing Huang. "Attention-Based Convolutional Neural Network for Semantic Relation Extraction". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 2526–2536. URL: https://aclanthology.org/C16-1238.

[90] Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. "Relation classification via multi-level attention cnns". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1298–1307.

[91] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 207–212. DOI: 10.18653/v1/P16-2034. URL: https://aclanthology.org/P16-2034.

[92] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

[93] Joohong Lee, Sangwoo Seo, and Yong Suk Choi. "Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing". In: *Symmetry* 11.6 (2019), p. 785.

[94] Tapas Nayak and Hwee Tou Ng. *Effective Attention Modeling for Neural Relation Extraction*. 2019. DOI: 10.48550/ARXIV.1912.03832. URL: https://arxiv.org/abs/1912.03832.

[95] Bowen Yu, Zhenyu Zhang, Tingwen Liu, Bin Wang, Sujian Li, and Quangang Li. "Beyond Word Attention: Using Segment Attention in Neural Relation Extraction." In: *IJCAI*. 2019, pp. 5401–5407.

[96] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi 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*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1785–1794. DOI: 10.18653/v1/D15-1206. URL: https://aclanthology.org/D15-1206.

[97] Adam Kilgarriff. *Wordnet: An electronic lexical database*. 2000.

[98] Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. "A dependency-based neural network for relation classification". In: *arXiv preprint arXiv:1507.04646* (2015).

[99] Makoto Miwa and Mohit 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)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1105–1116. DOI: 10.18653/v1/P16-1105. URL: https://aclanthology.org/P16-1105.

[100] Chris Quirk and Hoifung Poon. "Distant Supervision for Relation Extraction beyond the Sentence Boundary". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1171–1182. URL: https://aclanthology.org/E17-1110.

[101] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. "Cross-Sentence N-ary Relation Extraction with Graph LSTMs". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 101–115. DOI: 10.1162/tacl_a_00049. URL: https://aclanthology.org/Q17-1008.

[102] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. "N-ary Relation Extraction using Graph-State LSTM". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2226–2235. DOI: 10.18653/v1/D18-1246. URL: https://aclanthology.org/D18-1246.

[103] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[104] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. "Graph Attention Networks". In: *ArXiv* abs/1710.10903 (2018).

[105] Yuhao Zhang, Peng Qi, and Christopher D. Manning. "Graph Convolution over Pruned Dependency Trees Improves Relation Extraction". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2205–2215. DOI: 10.18653/v1/D18-1244. URL: https://aclanthology.org/D18-1244.

[106] Zhijiang Guo, Yan Zhang, and Wei Lu. "Attention guided graph convolutional networks for relation extraction". In: *arXiv preprint arXiv:1906.07510* (2019).

[107] Yuanhe Tian, Guimin Chen, Yan Song, and Xiang Wan. "Dependency-driven relation extraction with attentive graph convolutional networks". 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)*. 2021, pp. 4458–4471.

[108] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. "SpanBERT: Improving Pre-training by Representing and Predicting Spans". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 64–77. DOI: 10.1162/tacl_a_00300. URL: https://aclanthology.org/2020.tacl-1.5.

[109] Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. "Extracting multiple-relations in one-pass with pre-trained transformers". In: *arXiv preprint arXiv:1902.01030* (2019).

[110] Christoph Alt, Marc Hübner, and Leonhard Hennig. "Improving relation extraction by pre-trained language representations". In: *arXiv preprint arXiv:1906.03088* (2019).

[111] Peng Shi and Jimmy Lin. "Simple bert models for relation extraction and semantic role labeling". In: *arXiv preprint arXiv:1904.05255* (2019).

[112] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. *Simplifying Graph Convolutional Networks*. 2019. DOI: 10.48550/ARXIV.1902.07153. URL: https://arxiv.org/abs/1902.07153.

[113] Jun Chen, Robert Hoehndorf, Mohamed Elhoseiny, and Xiangliang Zhang. *Efficient long-distance relation extraction with DG-SpanBERT*. 2020. DOI: 10.48550/ARXIV.2004.03636. URL: https://arxiv.org/abs/2004.03636.

[114] Qiongxing Tao, Xiangfeng Luo, Hao Wang, and Richard Xu. "Enhancing relation extraction using syntactic indicators and sentential contexts". In: *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*. IEEE. 2019, pp. 1574–1580.

[115] Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. "Knowledge enhanced contextual word representations". In: *arXiv preprint arXiv:1909.04164* (2019).

[116] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. *KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation*. 2019. DOI: 10.48550/ARXIV.1911.06136. URL: https://arxiv.org/abs/1911.06136.

[117] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. "K-adapter: Infusing knowledge into pre-trained models with adapters". In: *arXiv preprint arXiv:2002.01808* (2020).

[118] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

[119] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. "LUKE: deep contextualized entity representations with entity-aware self-attention". In: *arXiv preprint arXiv:2010.01057* (2020).

[120] Amir DN Cohen, Shachar Rosenman, and Yoav Goldberg. "Relation Classification as Two-way Span-Prediction". In: *arXiv preprint arXiv:2010.04829* (2020).

[121] Arzoo Katiyar and Claire Cardie. "Investigating LSTMs for Joint Extraction of Opinion Entities and Relations". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 919–929. DOI: 10.18653/v1/P16-1087. URL: https://aclanthology.org/P16-1087.

[122] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. "Joint entity recognition and relation extraction as a multi-head selection problem". In: *CoRR* abs/1804.07847 (2018). arXiv: 1804.07847. URL: http://arxiv.org/abs/1804.07847.

[123] Dat Quoc Nguyen and Karin Verspoor. "End-to-end neural relation extraction using deep biaffine attention". In: *CoRR* abs/1812.11275 (2018). arXiv: 1812.11275. URL: http://arxiv.org/abs/1812.11275.

[124] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. "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)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1227–1236. DOI: 10.18653/v1/P17-1113. URL: https://aclanthology.org/P17-1113.

[125] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. "Extracting Relational Facts by an End-to-End Neural Model with Copy Mechanism". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 506–514. DOI: 10.18653/v1/P18-1047. URL: https://aclanthology.org/P18-1047.

[126] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. "A hierarchical framework for relation extraction with reinforcement learning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7072–7079.

[127] Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun 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*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1409–1418. DOI: 10.18653/v1/P19-1136. URL: https://aclanthology.org/P19-1136.

[128] Bayu Distiawan Trisedya, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. "Neural Relation Extraction for Knowledge Base Enrichment". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 229–240. DOI: 10.18653/v1/P19-1023. URL: https://aclanthology.org/P19-1023.

[129] Jiayu Chen, Caixia Yuan, Xiaojie Wang, and Ziwei Bai. "MrMep: Joint Extraction of Multiple Relations and Multiple Entity Pairs Based on Triplet Attention". In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 593–602. DOI: 10.18653/v1/K19-1055. URL: https://aclanthology.org/K19-1055.

[130] Daojian Zeng, Haoran Zhang, and Qianying Liu. "CopyMTL: Copy Mechanism for Joint Extraction of Entities and Relations with Multi-Task Learning". In: *CoRR* abs/1911.10438 (2019). arXiv: 1911.10438. URL: http://arxiv.org/abs/1911.10438.

[131] Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Yubin Wang, Tingwen Liu, Bin Wang, and Sujian Li. *Joint Extraction of Entities and Relations Based on a Novel Decomposition Strategy*. 2019. DOI: 10.48550/ARXIV.1909.04273. URL: https://arxiv.org/abs/1909.04273.

[132] Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. *A Novel Cascade Binary Tagging Framework for Relational Triple Extraction*. 2019. DOI: 10.48550/ARXIV.1909.03227. URL: https://arxiv.org/abs/1909.03227.

[133] *CNN corpus*. URL: https://drive.google.com/file/d/1aMiIZXLpO7JF-z_Zte3uH7OCo4Uk_Odo/view.

[134] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 19–27. DOI: 10.1109/ICCV.2015.11.

[135] *Bias and Variance in Machine Learning*. URL: https://static.javatpoint.com/tutorial/machine-learning/images/bias-and-variance-in-machine-learning5.png.