Vietnamese-German University

COPYRIGHT WARNING

RELATION EXTRACTION FROM TEXT USING SUPERVISED MACHINE LEARNING

Duong Thanh Hung

Faculty of Engineering

Vietnamese - German University

Supervisor

Dr. Huynh Trung Hieu

Dr. Tran Thi Thu Huong

In partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

September 25, 2020

Declaration

I, Duong Thanh Hung, declare that this thesis titled, "Relation extraction from text using supervised machine learning" submitted to the Vietnamese-German University (VGU) and the Frankfurt University of Applied Science (FRA-UAS), is created by me and has not used any other sources than specified. This work is submitted in the partial fulfillment of the requirements for the degree Bachelor of Science in Computer Science at VGU and FRA-UAS. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Duong Thanh Hung | Date

Approved by

Dr. Huynh Trung Hieu
First supervisor

Dr. Tran Thi Thu Huong
Second supervisor

Acknowledgements

I would like to express my sincere gratitude to Dr. Huynh Trung Hieu and Dr Tran Thi Thu Huong, my thesis supervisors. They have spent their valuable time to give me guidance despite having a busy schedule.

I am also grateful to my parents and to my classmates for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

In the last decades, the volume of online resources, especially text, has increased at a magnificent speed. How to extract information efficiently from that data becomes an exciting research topic.

In this thesis, we set out to investigate the task of relation extraction. In particular, we only concern about the relation classification step for binary relations.

We conducted experiments with various settings, using BERT as the word representation. We also tested the idea of assigning "no relation" detection to a separate classifier.

Overall, the single-classifier architecture yields the best performance. On the SemEval 2010 Task 8 dataset, it achieved the F1 score of 90.12%, and on the GDS dataset, it reached the average precision score of 85.05%. The duo classifier architecture does not fall far behind (89.98% F1 on the SemEval dataset).

Keywords: *natural language processing, relation extraction, supervised machine learning, BERT, SemEval 2010 Task 8 dataset, GDS dataset*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

A massive amount of unstructured web-based electronic text exists, including news, forums, email messages, government records, chat logs, and many more. How could a human be helped in understanding all that data? One of the popular ideas, if not the most, is to convert unstructured text to structured data by annotating semantic relation in the text. This task is known as relation extraction. For example, from the following sentence:

> *In May 2002, Elon Musk founded SpaceX, an aerospace manufacturer and space transport services company.*

we can get the relation *FounderOf(Elon Musk, SpaceX)*

The sheer volume and diversity of data, however, make annotation impossible for humans. Rather, we would like to have a computer annotate all data with the structure of our interest, typically the relations between entities, such as organization, date time, location, and person.

The whole relation extraction process is no trivial task. Complex semantic

property needs to be understood by the computer in order to make a correct annotation. Therefore, extracting semantic relations between entities in natural language text is a crucial step towards natural language understanding applications. [8]

1.2 Context of the thesis

To extract relations from texts, first of all, the entities needed to be identified. With current state-of-the-art named entities recognizer (NER), data can be labeled automatically with high accuracy. In this thesis, we only concern about the next step: recognizing the relations between the entities.

Since the number of possible relations between entities is vast, this thesis's scope only covers several predefined common relations.

We would like to further narrow our interest down to *binary relations* only. A binary relation is a relation that involves precisely two entities. Consider the following example:

> *Markus Persson, the creator of Minecraft, left Mojang after Microsoft's acquisition of the company in 2014.*

Aquisition(Microsoft, Mojang) is a binary relation while *AquisitionYear(Microsoft, Minecraft, 2014)* is not.

In the above example, we can also see another relation: *Creator(Markus Persson, Minecraft)*. Extracting all relations in one pass is out of the thesis's scope as it will significantly complicate the problem.

To summarize, we expect the input sentences to have exactly two annotated entities. Sentences with more than one pair of entities have to be cloned; each entity pair is annotated in an independent duplication of the sentence. Our model will classify the pairs into predefined classes (including the "not related" relation).

1.3 Overview of the thesis

This thesis consists of five chapters:

- *Chapter 1 - Introduction,* which introduces the readers to the motivations and context of the thesis.
- *Chapter 2 - Literature review,* which gives an overview of relevant studies.
- *Chapter 3 - Methodology,* in which an approach to the problem is proposed.
- *Chapter 4 - Results and discussion,* which presents the model performance on two datasets and discusses the result.
- *Chapter 5 - Conclusion,* which summarizes the whole thesis.

Chapter 2

Literature review

2.1 Overview of relation extraction

Event extraction has been quite well studied. Generally, there are three approaches [9]:

- Data-driven approaches: aim to convert data to knowledge through the usage of statistics, machine learning, linear algebra, etc.
- Expert knowledge-driven methods: extract knowledge through representation and exploitation of expert knowledge, usually by means of pattern-based approaches.
- Hybrid approaches: combine the above methods.

Data-driven methods require many data and little domain knowledge and expertise while having low interpretability. Conversely, for knowledge-based relation extraction, little data is required, but domain knowledge and expertise are needed. These approaches generally offer higher traceability of the results. Finally, hybrid approaches seem to be a compromise between data and knowledge-driven

approaches, requiring a medium amount of data and domain knowledge and offering medium interpretability. However, it should be noted that the amount of expertise needed is high since multiple techniques are combined. [9]

2.2 Relation extraction with supervised machine learning

Generally, relation extraction with supervised machine learning fits into the category of data-driven methods. We can further divide them into three major approaches: RNN-based, CNN-based, and dependency-based.

2.2.1 RNN-based approaches

Recurrent Neural Network (RNN) is one of the most common choices, if not the most, for natural language processing (NLP). The reason is simple: to understand a word, a model needs to understand its surrounding context, and RNNs do just that. When processing a word, RNNs not only look at the target but also consider the network's previous state.

RNN-based approaches have been relatively well studied. Here, only some recent outstanding researches are highlighted.

Zhang et al. in their paper in 2015 proposed a common framework for relation extraction using RNN, which, despite its simplicity, yields promising results [10].

Miwa and Bansal enhanced the traditional *Long-Short Term Memory (LSTM)* network by stacking bidirectional tree-structured LSTMs on bidirectional sequential LSTMs. This enables their model to capture both word sequence and dependency tree substructure information. Both entities and relations can be jointly represented with shared parameters in a single model [11].

Other authors tried to incorporate *attention mechanism* to RNN networks. Some influential papers are "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification" (Zhou et al. 2016), "Semantic Relation Classification via Hierarchical Recurrent Neural Network with Attention" (Xiao and Liu 2016), and "Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing" (Lee et al. 2019).

We shall review one instance of RNN, Gated Recurrent Unit (GRU), in session 2.4 as it appears in our experiment.

2.2.2 CNN-based approaches

Convolutional Neural Network (CNN) is most commonly applied to analyzing visual imagery. However, it has found its way to Natural Language Processing in the last decade and has proved to be a robust method rather than just a novel idea. A basic use of CNN for the sentence classification task can be found in Figure 2.1.

We would like to highlight a few recent papers with this approach.

Zeng et al. (2014) proposed a model using CNN to extract lexical and sentence level features. These two-level features are concatenated and fed into a softmax classifier to predict the relationship between two marked nouns [12].

In their paper in 2016, Shen and Huang proposed a model that makes full use of word embedding, part-of-speech tag embedding, and position embedding information. Word level attention mechanism is applied to better determine which parts of the sentence are most influential with respect to the two entities of interest [13].

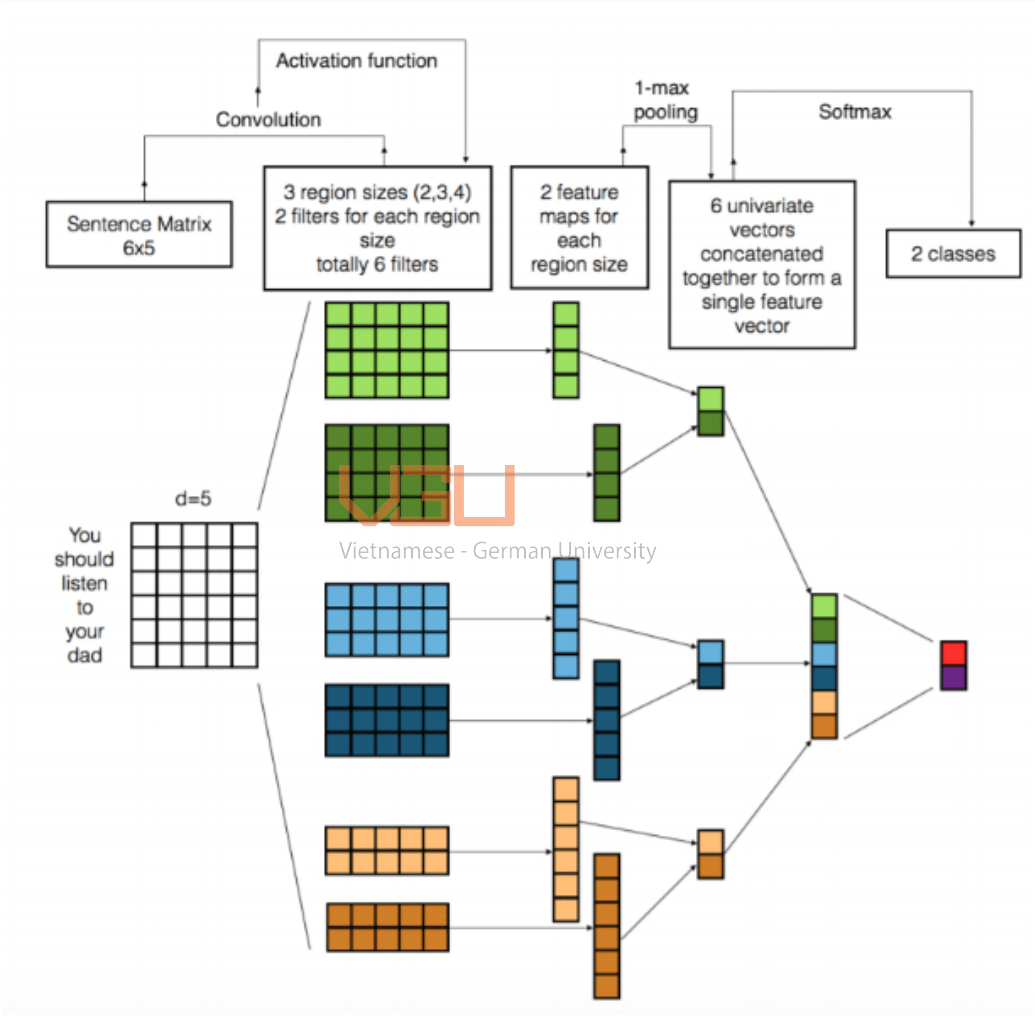

Figure 2.1: Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification. [1]

2.2.3 Dependency-based approach

Semantic dependency parsing has been frequently used to dissect sentence and to capture word semantic information close in context but far in sentence distance.

To extract the relationship between two entities, the most direct approach is to use Shortest Dependency Path (SDP). The motivation of using SDP is based on the observation that the SDP between entities usually contains the necessary information to identify their relationship.

With the paper "A shortest path dependency kernel for relation extraction" in 2005, Bunescu et al. laid a solid framework for this approach. An enhanced version of the shortest path dependency, called *Augmented Dependency Path*, was introduced by Liu et al (2015).

Besides that, there have been numerous attempts to integrate path dependency to RNNs and CNNs, notably are works by Yan et al. (2015) and Cai et al. (2016).

2.3 Word representation

For machine learning models to interpret words, they need some form of numeric representation that models may use in their computation. This section will give an overview of this task.

2.3.1 Traditional context free approach

2.3.1.1 One Hot Encoding

Also known as Bag of words, in this approach, each element in the vector corresponds to a unique word or n-gram (token) in the corpus vocabulary. If the token appears in the document, the element is marked as 1, otherwise a 0.

An apparent drawback to this method is that it does not represent any idea, meaning, or word similarity within the vectors.

2.3.1.2 TF-IDF

TF-IDF, or Term frequency-inverse document frequency, is a statistical measure to evaluate the importance of a word to a document in a corpus. This importance is directly proportional to the number of times a word appears in the document but is offset by the number of documents in the corpus that contain that word. [14]

In formal mathematical terms, the TF-IDF score for the word t in document d from the document set D is calculated as follows:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Where

$$tf(t, d) = log\,(1 + log(t, d))$$

$$idf(t, D) = log\left(\frac{N}{count(d \in D : t \in d)}\right)$$

In this approach, instead of filling the document vectors with the raw count (like in the Bag of words approach), we fill it with the TF-IDF score of the term for that document.

Even though TF-IDF representations provide weights to different words, they are unable to capture the word meaning.

2.3.1.3 Word Embeddings

Besides the inability to capture word semantics, another major drawback of both of the above approaches is that as the vocabulary size increases, so does the size of the vector representing the document. The result is a matrix with lots of zeros

(a parse matrix), demanding more memory and computational resources during modeling.

Neural Word embedding solves both the deficiencies — achieving a reduction in dimensional space using dense representations and a more expressive representation using semantic similarity.

A word embedding is a learned representation (real-valued vectors) for text where terms of the same meaning are expressed similarly.

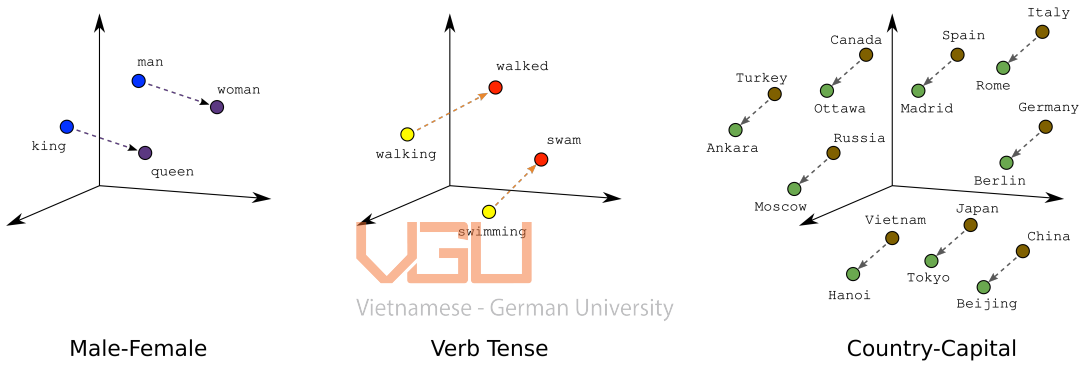

Figure 2.2: Embeddings can produce remarkable analogies [2]

The key to this method is the idea of using a densely distributed representation for each word. Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one-hot encoding. [14]

The two most popular word embeddings are *Word2Vec* and *GloVe*, which are both unsupervised approaches based on the distributional hypothesis (words that occur in the same contexts tend to have similar meanings) [15]. We shall not discuss these two in more detail here as they are not of our interest.

2.3.2 Pretrained Language model

2.3.2.1 The context free word representations limitations

The context-free word representations presume that the meaning of a word is relatively stable across sentences, which is not the case. We have to be aware of considerable variations in meaning for one single word. For example, *left* (as the opposite to right) and *left* (the past tense of leave); or *spring* (a season) and *spring* (coiled metal)

Traditional word vectors are shallow representations (a single layer of weights), and incorporate prior knowledge only into the model's first layer. The remainder of the network is yet to be trained on a specific objective task from scratch.

A computer vision model initialized with pre-trained representations can recognize only edges - they will be helpful for many tasks, but they fail to capture higher-level information that might be even more useful. Similarly, word embeddings are useful in only capturing semantic meanings of words, but we also need to understand higher-level concepts such as anaphora, long-term dependencies, agreement, negation, and many more. [14]

2.3.2.2 Pretrained language model

The philosophy behind pre-trained language models is to create word representations that understand the language (i.e., not only word meanings but also dependencies, anaphora). It can then be asked to do any specific task in that language. The idea is to create the machine equivalent of a "well-read" human being.

Since the introduction of pre-trained language models, practical applications of natural language processing have been substantially cheaper, quicker, and more straightforward due to their transfer learning capabilities.

Using pre-trained language models is one of today's most exciting directions for NLP and lots of papers recently explore transfer learning. Here we want to highlight three research papers [16] that are at the core of this latest NLP trend:

- *ULMFiT* - Universal Language Model Fine-Tuning method, is likely the first effective approach to fine-tuning the language model. The authors demonstrate the importance of several novel techniques, including discriminative fine-tuning, slanted triangular learning rate, and gradual unfreezing, for retaining previous knowledge and avoiding catastrophic forgetting during fine-tuning. [17]

- *ELMo* word representations, or Embeddings from Language Models, are generated in a way to take the entire context into consideration. In particular, they are created as a weighted sum of the internal states of a deep bi-directional language model (biLM), pre-trained on a large text corpus. Furthermore, ELMo representations are based on characters so that the network can understand even out-of-vocabulary tokens unseen in training. [18]

- *BERT*, or Bidirectional Encoder Representations from Transformers, is a new cutting-edge model that considers the context from both the left and the right sides of each word. [4]

We will discuss *BERT* further in the next section as it is used in our model.

2.3.2.3 BERT

Created and published in 2018 by engineers at Google, BERT has help achieved new state-of-the-art results in many common benchmarks like GLUE, SQuAD, SWAG [4]. So how is this model able to make such a breakthrough? A quote

from the team developed BERT would be a brief and precise description of this model: [3]:

> *BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.*

Architecture

BERT architecture builds on top of Transformer [4]. To be more precise, it consists of multiple Transformer encoders stack on top of each other. BERT's authors published two variants along with the paper:

- BERT Base: 12 layers, 12 attention heads, hidden size 768 and 110 million parameters
- BERT Large: 24 layers, 16 attention heads, hidden size 1024 and 340 million parameters

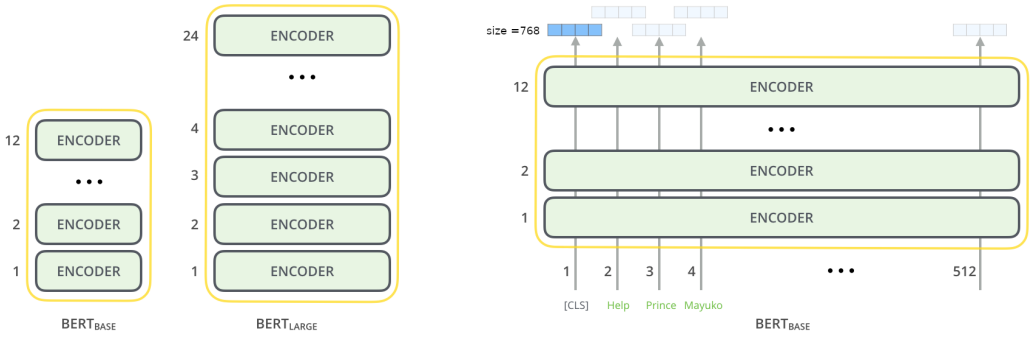

Figure 2.3: BERT architecture [3]

Pretraining tasks

BERT is trained with 2 pretraining task: masked language model and next sentence prediction.

Masked language model. The researchers randomly masked 15% of the words, in which:

- 80% of the time the words were replaced with the masked token [MASK]
- 10% of the time the words were replaced with random words
- 10% of the time the words were left unchanged

BERT is trained to predict the masked tokens.

Next sentence prediction. Additionally, BERT is also trained on the task of Next Sentence Prediction. Given two sentences – A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence? In this training, 50% sentence pairs are continuous sentences.

Input

First a sentence is tokenized. BERT uses WordPeice tokenizer, which not only breaks a sentence into words but also breaks a word into sub-words. For example, "unfortunately" can be split as "un" + "##fortun" + "##ate" + "##ly" (here the "##" prefix denotes a sub-word).

Three special tokens are used for markup:

- [CLS]: marks the beginning of the input
- [SEP]: marks the end of the input and separate the first and second sentence
- [PAD]: used for padding at the end if the number of tokens is less than the maximum input size

The actual input of BERT is the combination of the three following embeddings:

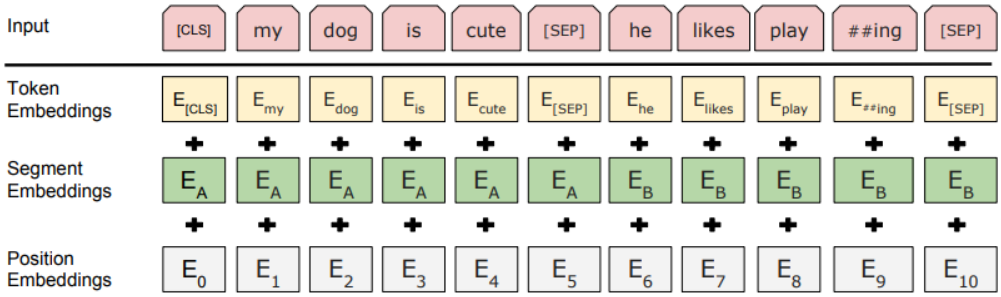

Figure 2.4: Bert embedding [4]

- Token Embeddings: These are the embeddings learned for the specific token from the WordPiece token vocabulary of more than 30000.
- Segment Embeddings: BERT can also take sentence pairs as inputs for tasks (Question-Answering). That's why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB).
- Position Embeddings: BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture "sequence" or "order" information.

Output

For each token, including [PAD], [SEP] and [CLS], BERT outputs a vector of a pre-configured hidden size. The output vector corresponds to the [CLS] special token is used for prediction tasks.

2.3.2.4 DistilBERT

While BERT and its variants perform really well in downstream tasks, their huge sizes pose significant restrictions in applications.

To tackle this problem, many approaches have been proposed, such as *quantization* (approximating the weights of a network with a smaller precision) and *weights pruning* (removing some connections in the network).

Researchers of the Transformer library decided to use distillation: a technique where a large model, called the teacher, can be compressed into a smaller model called the student.

In the teacher-student training, we train a student network to mimic the full output distribution of the teacher network (its knowledge).

Rather than training with a cross-entropy over the hard targets (one-hot encoding of the goal class), we transfer the knowledge from the teacher to the student with a cross-entropy over the soft targets (probabilities of the teacher).

Concretely, the loss function is calculated as follows [19]:

$$L = -\sum_{i} t_i \cdot log(s_i)$$

Where t_i and s_i are the logits from the teacher and student respectively.

The authors also used a few training tricks from the recent RoBERTa paper: using large batches (up to 4000) to leverage gradient accumulation, dynamic masking, and removed the next sentence prediction objective.

DistilBERT was trained on eight 16GB V100 GPUs for approximately three and a half days using the concatenation of Toronto Book Corpus and English Wikipedia (same data as original BERT).

Finally, the authors compared the performance of DistilBERT on the development sets of the GLUE benchmark against BERT base (DistilBERT's teacher).

DistilBERT performs surprisingly well to BERT: they can retain more than 95% of the performance while having 40% fewer parameters. In terms of inference time, DistilBERT is more than 60% faster than BERT. [19]

2.4 GRU networks

In this section, we shall review GRU - one of the building block of our model.

GRU, or Gated Recurrent Unit, was introduced by Cho et al. in 2014. GRU network overview is as follows:

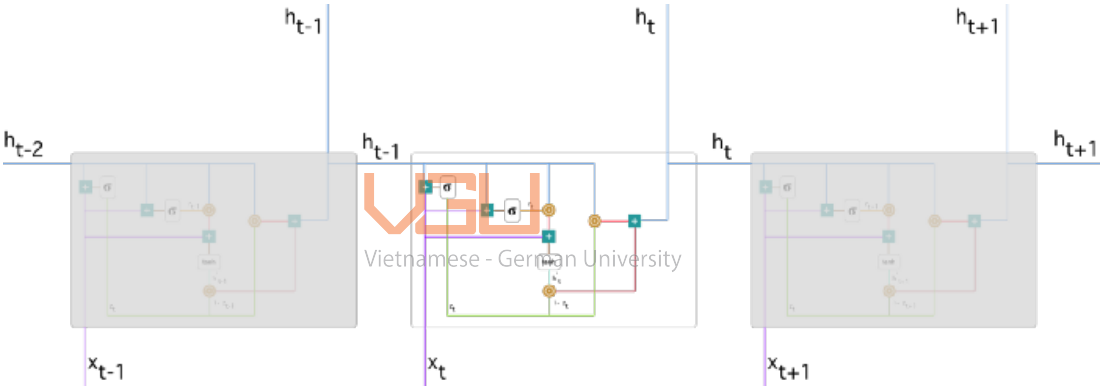

Figure 2.5: Overview of GRU [5]

As can be seen from the diagram, the output when processing $t-1^{th}$ token, h_{t-1}, is used as an input when processing the next word x_t.

Figure 2.6 provides a closer look at the GRU cell.

The red box is called the *update gate*. The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future.

On the opposite, there is *forget gate* (the blue box). Essentially, this gate is used from the model to decide how much of the past information to forget.

Bidirectional GRU

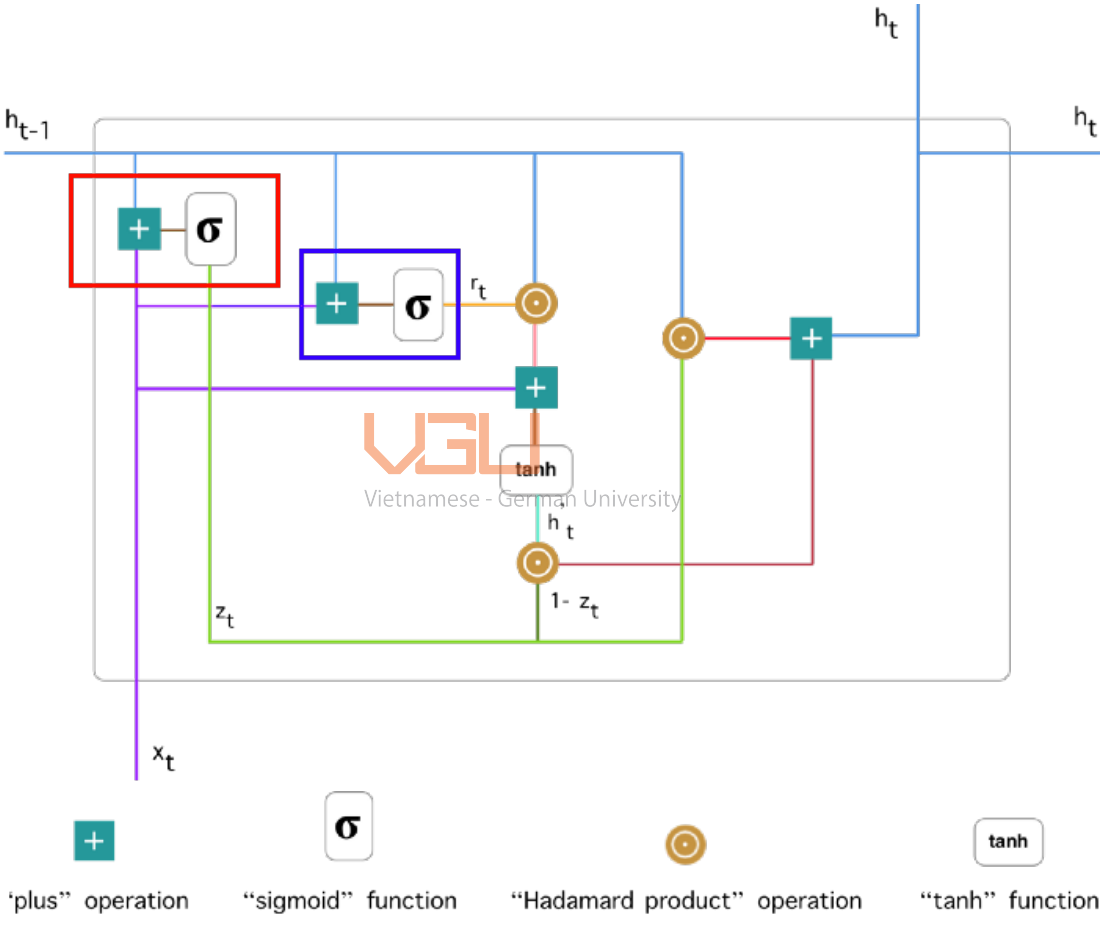

Figure 2.6: Detail of GRU cell [5]

One of the largest limitations of GRU is that it fails to capture the context of the words following the currently processed target. This drawback is built into GRU architecture itself: only the previous words output are used, not the upcoming words'.

To address this problem, a variant of GRU is introduced: Bidirectional GRU. Bidirectional GRU consists of two independent GRU networks; one processes the sentence in the usual order and one processes in reverse order. Their result is then concatenated word-wise.

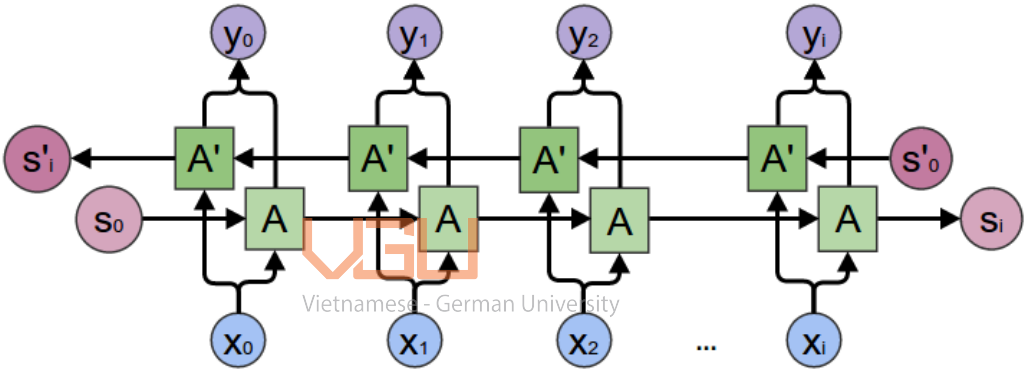

Figure 2.7: Bidirectional GRU [6]

Chapter 3

Methodology

Summary

This session gives a high-level description of the proposed model.

For this thesis, a variety of architectures have been tested. They fit into two categories: ones with a single classifier and ones with duo classifiers.

The first architecture is pretty straight forward. It consists of only one classifier, which can recognize both the "not-related" and other relations.

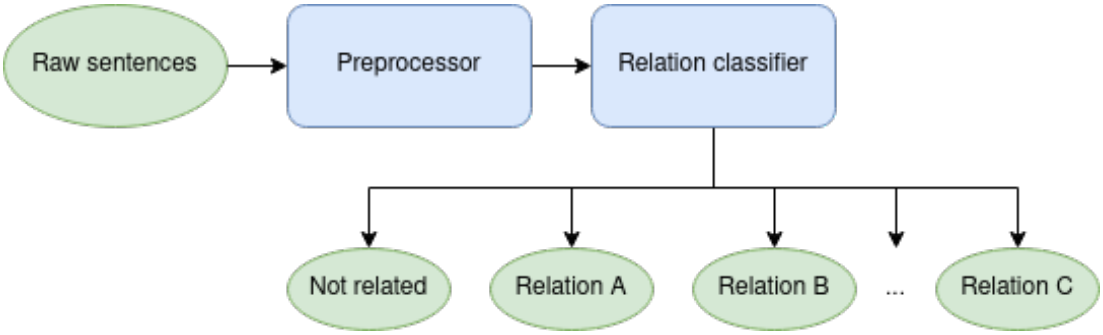

Figure 3.1: Single classifier architecture

The second architecture comprises two sub-modules: a binary classifier, which picks out sentences whose subject and object have no relations, and a multi-class

classifier that further groups the remaining sentences. The two sub-modules are trained and fine-tuned separately. The best variant of each are then combined and evaluated on the official test set.

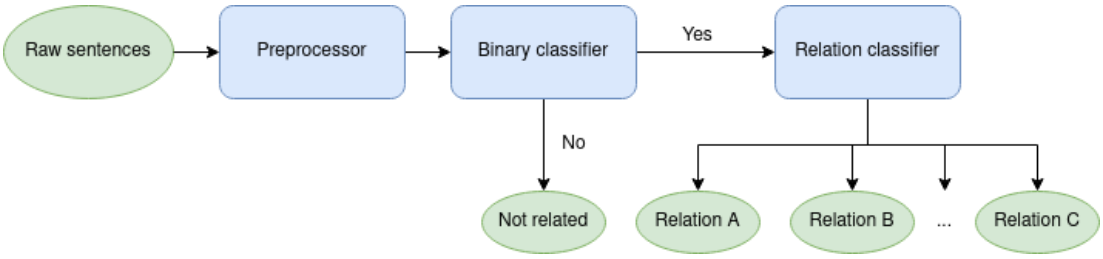

Figure 3.2: Duo classifiers architecture

3.1 Preprocessing

To mark the entities for relation extraction, four special characters are introduced. Subjects are wrapped in square brackets [ ], while objects are wrapped in curly brackets { }. Next, the sentence is broken into words and sub-words by the Word Pierce tokenizer. Special tokens ([CLS], [SEP], [PAD]) are then added. Note that as we do not train for the Next sentence prediction task, we do not need to provide the segment embeddings, and we only add [SEP] at the end of the sentence, right before the paddings. Finally, the whole token sequence is converted to vocabulary ids.

Sentences longer than model maximum input size (512, including special tokens) are truncated. If either the subject or the object is in the truncated parts, the sentence is skipped.

Labels are encoded into integers via a simple one-to-one mapping. In the duo classifiers architecture, some further processing is required. In particular, for the binary classifier, the "not-related" label is 0, and other labels are converted to 1. All the "not-related" sentences are removed from the training data set of the

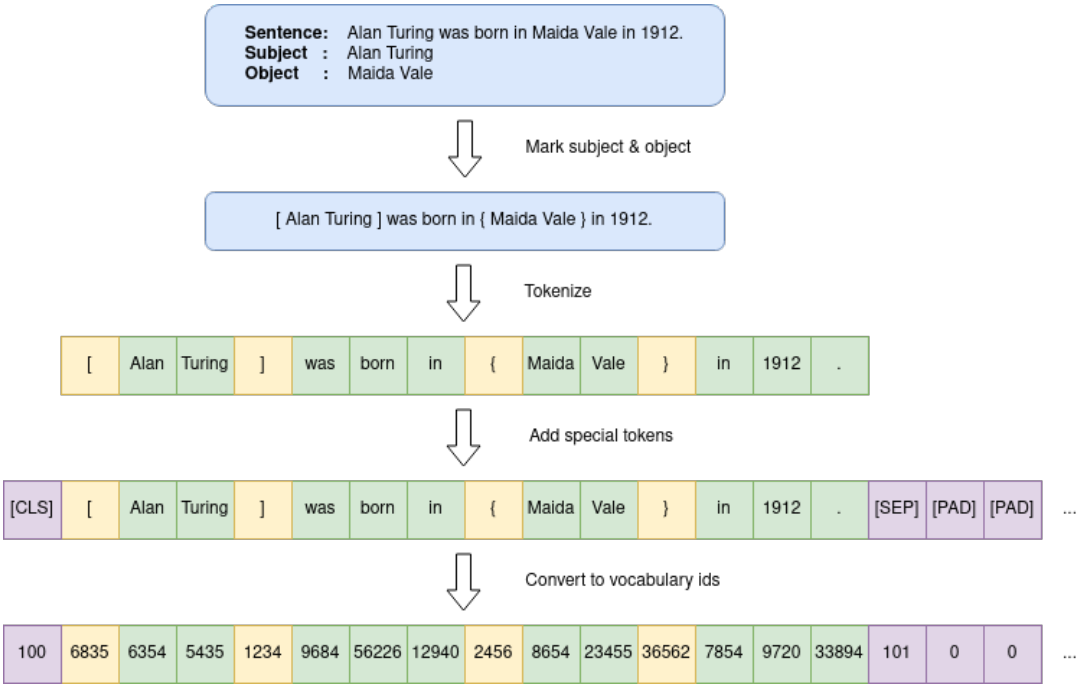

Figure 3.3: Example for pre-processing a sentence

relation classifier.

For BERT to work efficiently, for each sentence an *attention mask* should be provided. Each padding token in the sentences corresponds to a 0 in the attention mask and each remaining token corresponds to a 1.

We also created a training dataset with duplicated sentences. To be more specific, we cloned every sentence in the "not-related" class and swapped the object and subject annotations in one of the clones. For example, the sentence

> *In May 2002, [Elon Musk] founded { SpaceX }, an aerospace manufacturer and space transport services company.*

would have the clone

> *In May 2002, { Elon Musk } founded [SpaceX], an aerospace manufacturer and space transport services company.*

3.2 Smart batching

In our experiments, a technique calls "Smart batching" is applied to boost the model's speed and reduce the memory footprint.

This technique, proposed by Michaël Benesty in an online article [20], actually a combination of two well-known techniques: dynamic padding and uniform length batching.

Traditional fixed length padding

Neural networks only work with fixed length matrices, but not all sentences have the same length. A straight forward solution would be adding pad tokens to short sentences to have the same length as the longest one.

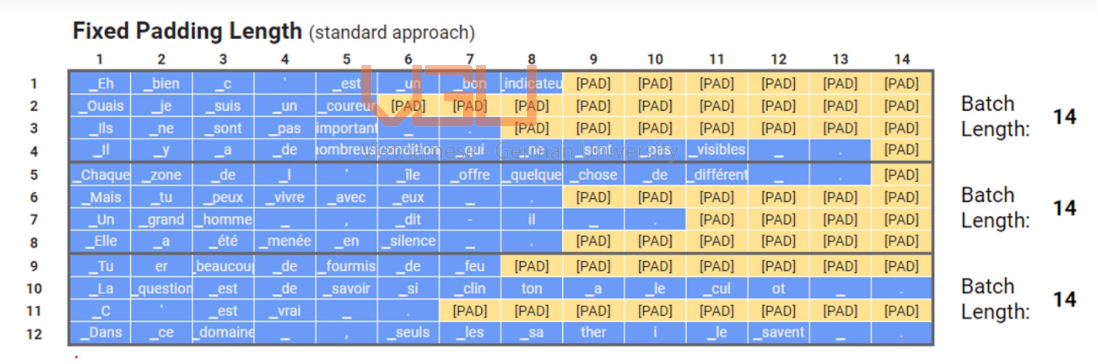

Figure 3.4: Example fix length padding [7]

The problem in this approach is that we would waste computational power on unnecessary padding tokens.

Dynamic padding

Instead of setting a fixed length for all sentences in the whole dataset, we can set a fixed length independently for each batch.

In a patch, every sentence is padded to the length of the longest sentence among them, not the global longest sentence.

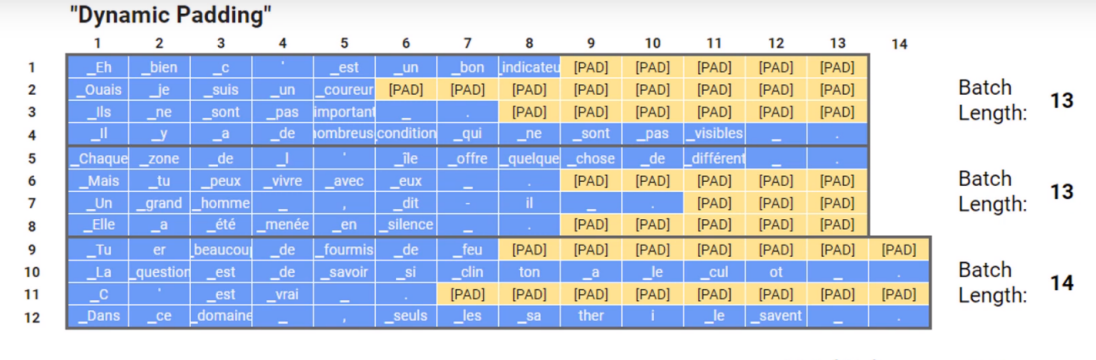

Figure 3.5: Example dynamic length padding [7]

Uniform length padding

We push the dynamic padding one step further: first, we will sort all sentences by their lengths, then pack consecutive sentences to the same batch. This will ensure that sentences with similar lengths are processed together, reducing the number of [PAD] tokens needed.

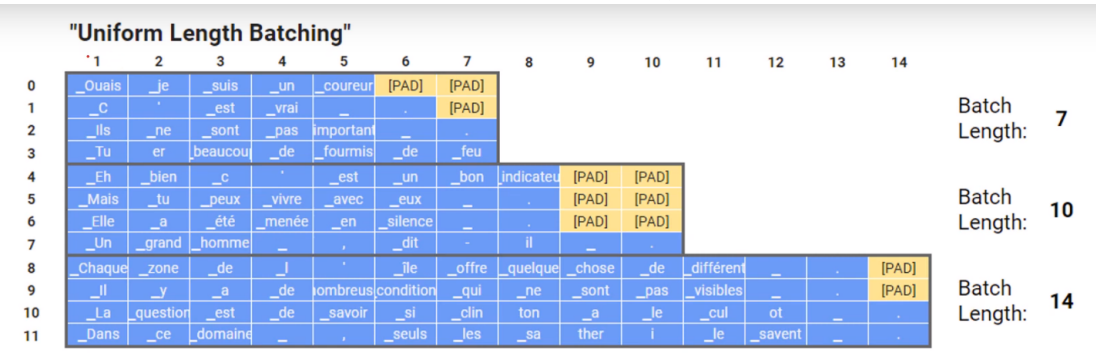

Figure 3.6: Example uniform length padding [7]

To ensure randomness when creating batches, we do not pack batches from the beginning but from random positions. The "smart batching" technique is described in *Algorithm 1.*

In our experiments, "Smart batching" actually sped up the training process three folds while reducing the memory footprint by half.

Algorithm 1 Smart batching

Require: *sentences*
Require: *batchSize*
  batches ← *Array*()
  while *sentences.length* > 0 **do**
    if *sentences.length* < *batchSize* **then**
      idx ← 0
    else
      idx ← *randomIntegerInRange*(0, *sentences.length* − *batchSize*)
    end if
    batch ← *sentences*[*idx* : *idx* + *batchSize*]
    maxLen ← *maxLength*(*batch*)
    for all *sentence* in *batch* **do**
      padSentenceToLength(*sentence*, *maxLen*)
    end for
    batches.append(*batch*)
  end while
  return *batches*

3.3 Classifiers

For each classifier, we experimented with five models. These five share two things: they all start with the BERT model and end with a linear layer (dense layer).

3.3.1 Using every token's embedding (ALL_TOKENS)

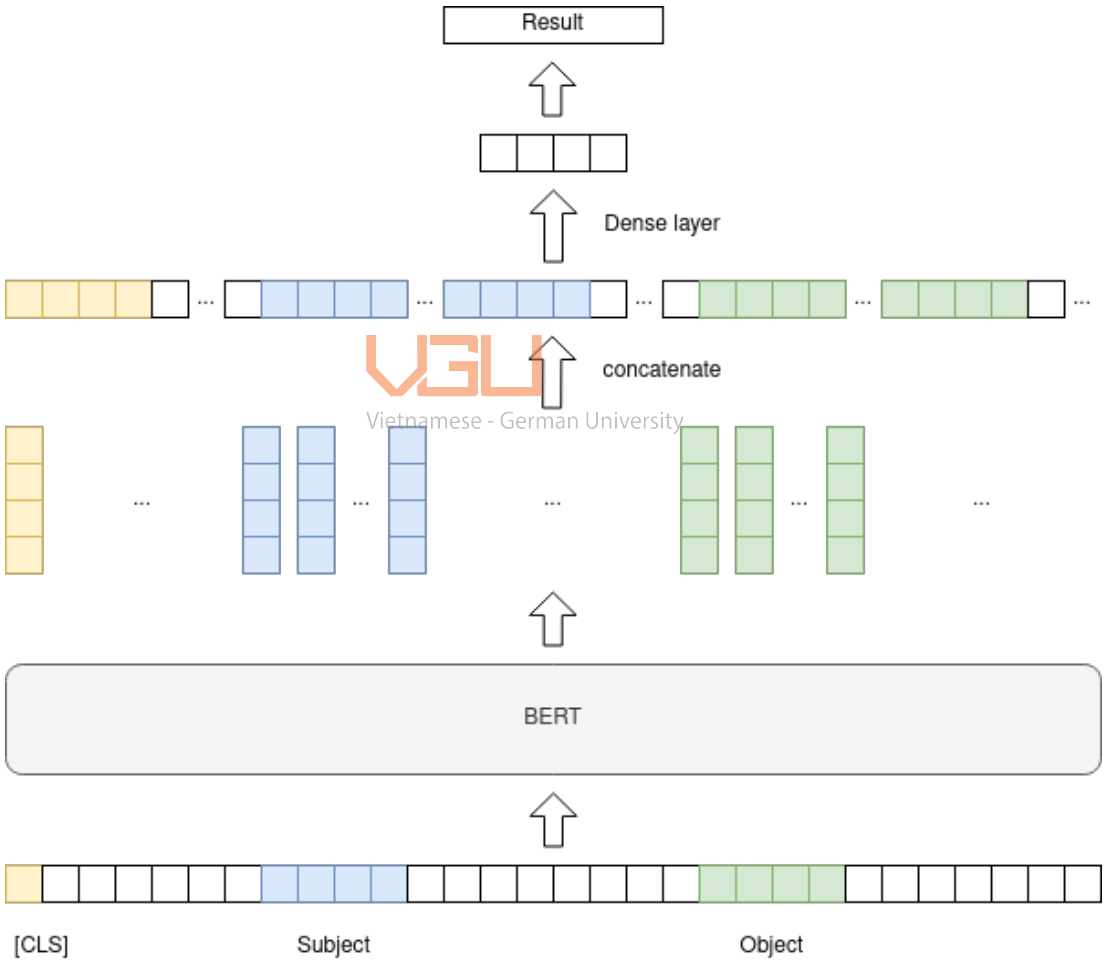

Figure 3.7: The model using every token's embedding

This probably is the most simple model of the five. After passing a sentence to BERT, every output vectors are concatenated into a single vector. This vector is

then connected to another dense layer before the final result is inferred.

3.3.2 Using the mean of every token's embedding ($MEAN$)

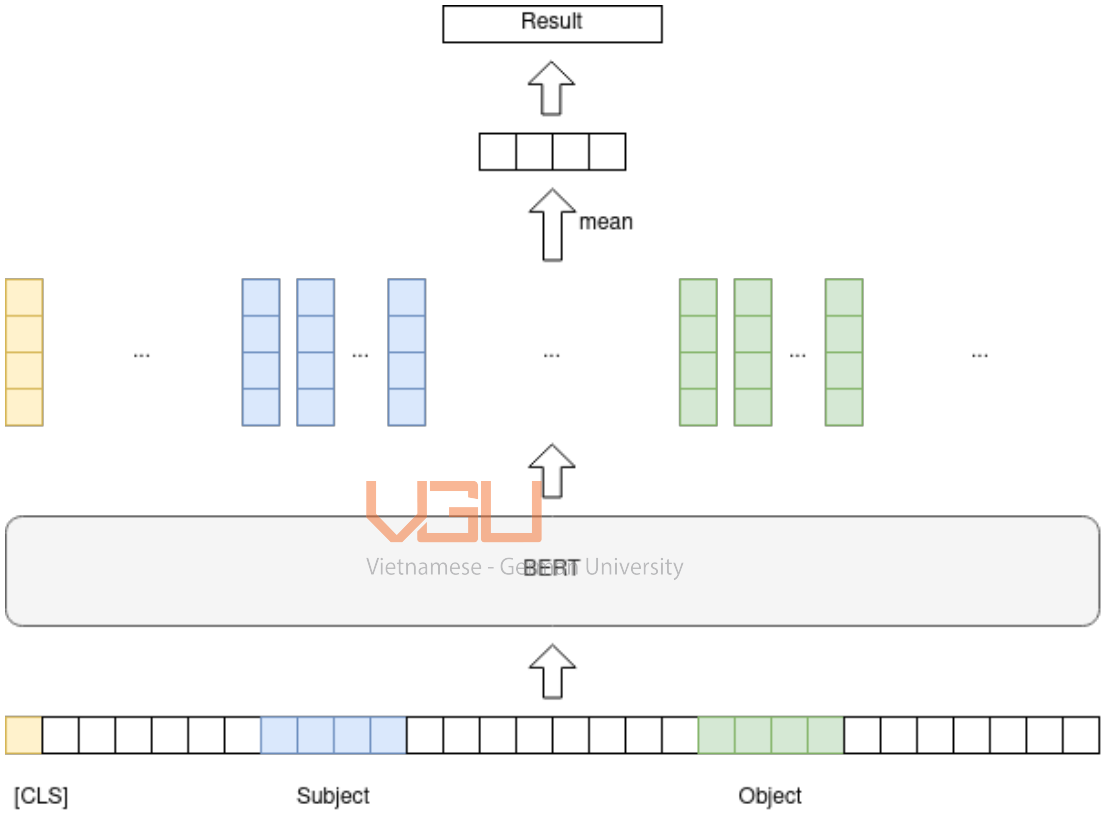

Figure 3.8: The model using the mean of every token's embedding

This model is quite similar to the one above. The difference is that instead of BERT output's concatenation, their mean vector is used.

The motivation for this approach is to use all of BERT's output as in the last approach while keeping the model to a reasonable size.

3.3.3 Using only [CLS] token's embedding (CLS)

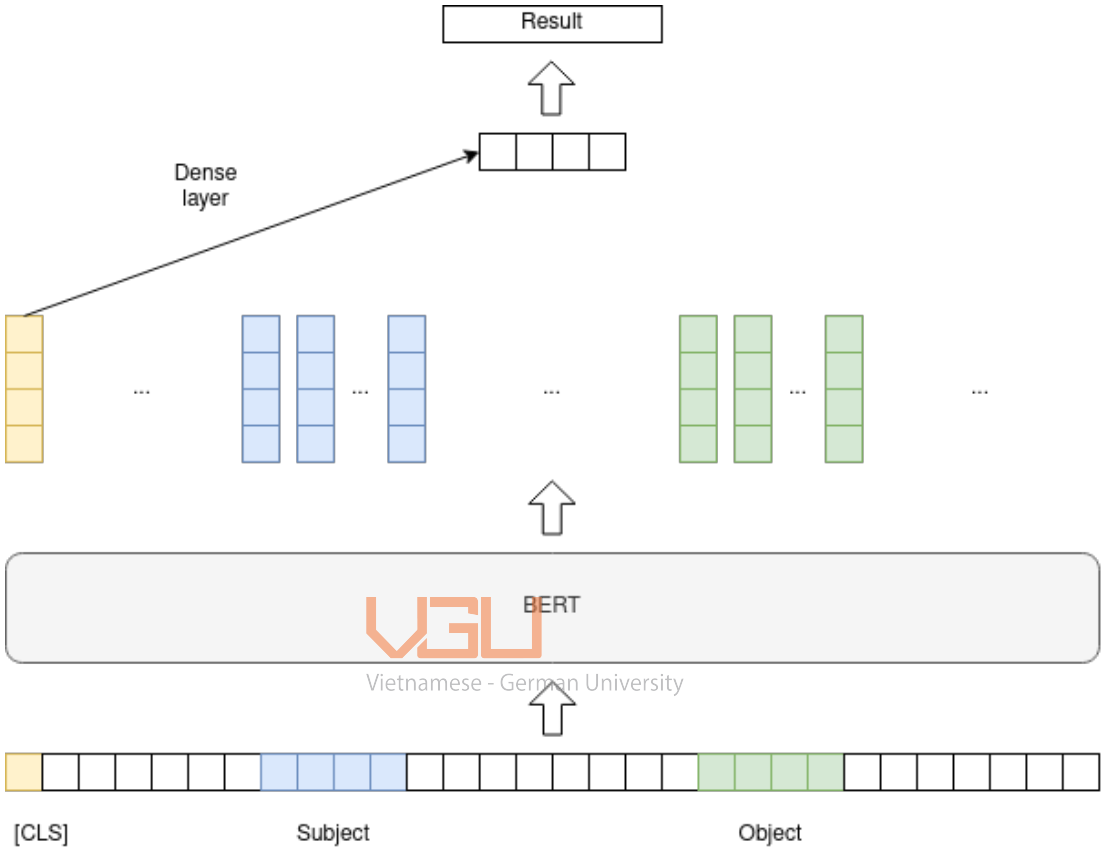

Figure 3.9: The model using only [CLS] token's embedding

In this model, after passing the whole sentence to BERT, only the output of the [CLS] token (the first token) is kept and passed to a dense layer.

The reason why only the [CLS] token is used is pretty straight forwards. In the original BERT's training task, the [CLS] token output is used for Next sentence prediction. Therefore, BERT's output of this token can, to some extent, capture the meaning of the whole sentence.

3.3.4 Using the embeddings of [CLS] token, subject and object (CLS_ENT)

A more sophisticated version of the last presented model. Not only [CLS]'s corresponding vector is used but also the subject's and object's. By adding these vectors, the model can work in a richer context.

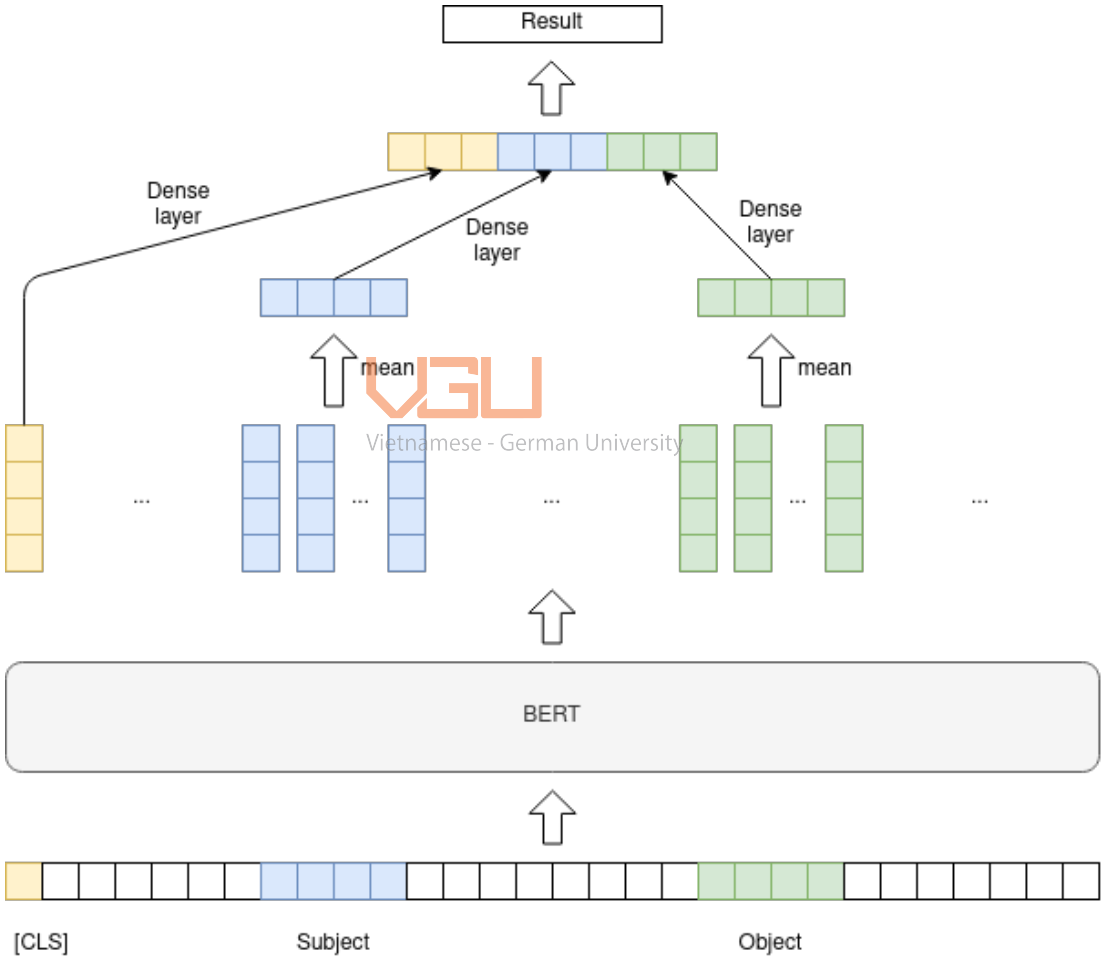

Figure 3.10: The model using every token's embedding

To be more concrete, for every token in the subject, their BERT's output is taken. We then calculate their mean and pass to a dense layer. A similar procedure is carried out on the object and the [CLS] token. The three output

vectors are concatenated and used to determine the final result.

3.3.5 Using GRU network (*GRU*)

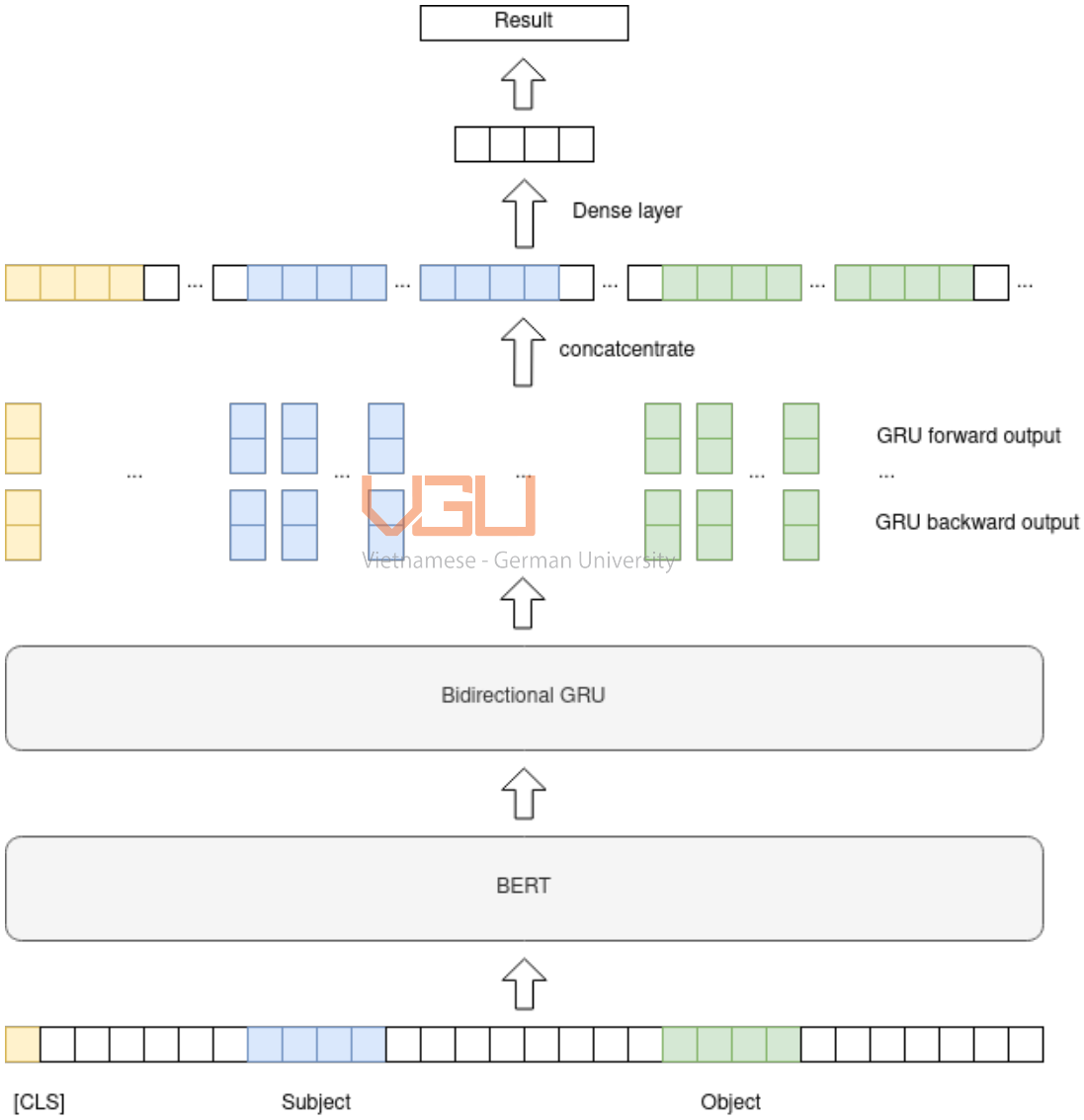

Figure 3.11: The model using GRU network

One of the most common approaches to sentence classification is to use RNNs.

In this model, the BERT's output is passed into a bidirectional GRU network.

The output is then concatenated and fed into a dense layer.

3.4 Training process

AdamW, a variant of the Adam algorithm, is chosen as the optimizer for the model.

The initial learning rate is 0.00002 and is reduced every epoch.

We performed hyper-parameter tuning on the following variables:

- BERT variant (BERT-base, BERT-large, distilBERT, RoBERTa)
- Batch size
- Dense layers sizes
- Dropout for dense layers
- Activation function
- Learning rate reduction speed
- Number of training epochs

In the binary classifier, the discrimination thresholds that yield the highest accuracy and F1 score are recorded. Testing is performed on both of these thresholds.

For each configuration, we trained and tested four times and averaged the results. This is quite a small sample, but we had to compromise due to the time and resources limitation. Finally, we re-ran the best configuration 20 times to ensure that the outcomes are reproducible.

Chapter 4

Results and discussion

4.1 Datasets

4.1.1 SemEval 2010 Task 8 dataset

SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems; it evolved from the Senseval word sense evaluation series. The evaluations are intended to explore the nature of meaning in language.

Started in 1998, as of 2020, SemEval has organized ten workshops. The fifth one took place in Uppsala (Sweden) in 2010. This workshop had 18 tasks in total, covering a wide range of natural language processing fields. In task 8, the participants were asked to classify pairs of words into semantic relations classes.

The dataset consists of 8000 sentences for training and 2717 sentences for testing. In each sentence, the two target words are marked with <e1> </e1> and <e2> </e2> notations. These 10717 sentences belong to nine actual semantic relation classes and one "Other" class.

Note that except for the "Other" relation, these semantic relations are di-

rectional, i.e., $Relation(A, B)$ is not the same as $Relation(B, A)$. Therefore, we have 19 classes in total.

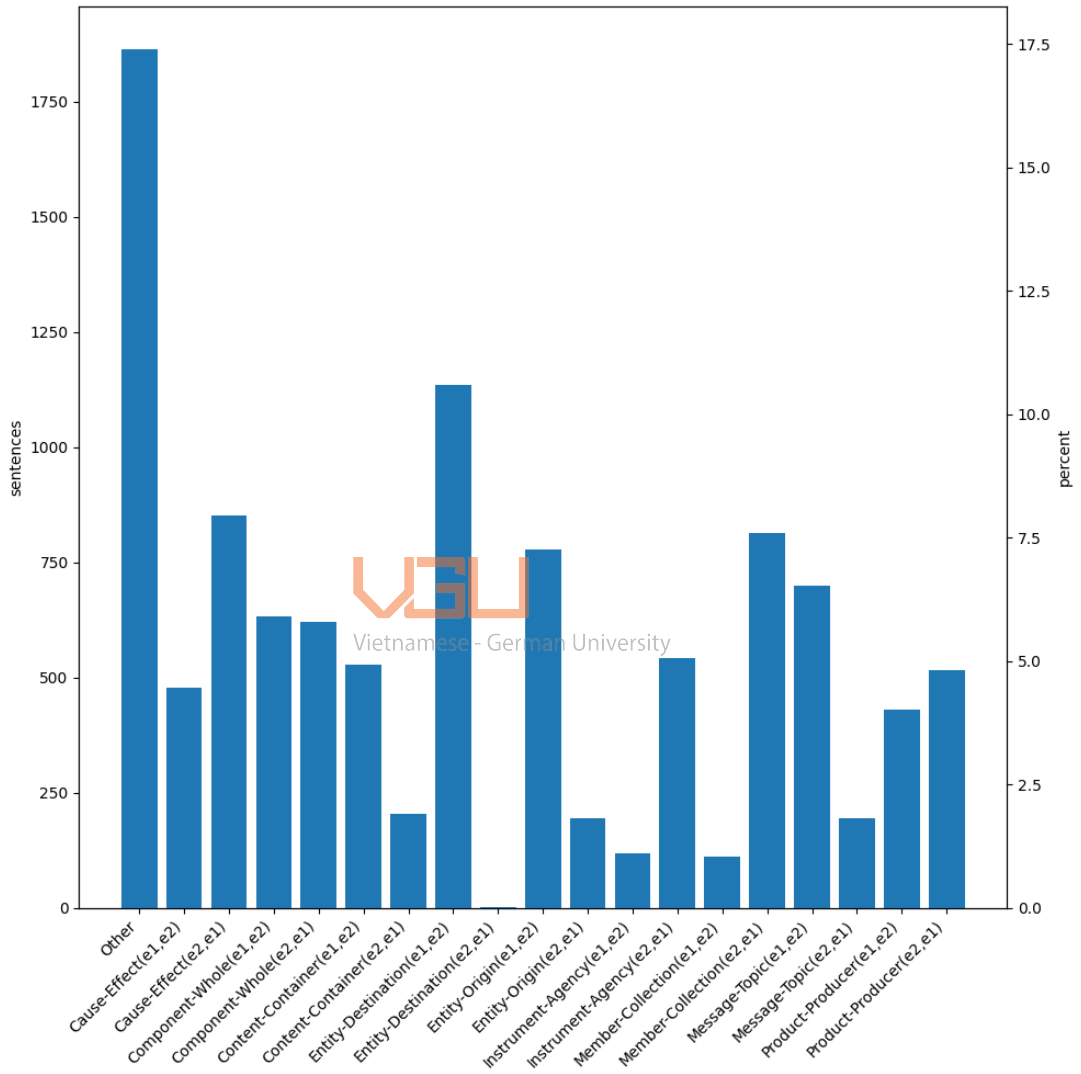

Figure 4.1: SemEval 2010 task 8 dataset relation distribution

The SemEval 2010 organizers also provide a scorer for this dataset. The model's official score is the macro F1 score, excluding the "Other" class, with relation direction taken into account. Besides that, we also measured this metric with the "Other" class included.

4.1.2 GDS dataset

The Google Distant Supervision dataset was created by Jat et al. in 2018, aimed specifically for the relation extraction task. Similar to the SemEval 2010 Task 8 dataset, an example includes a sentence, two entities whose relation needed to be classified, and the actual relation between them.

There are 11297 sentences for training, 1864 for validation, and 5663 for testing. There are only five relations, including one "N/A," and unlike the previous dataset, relation direction is *not* of our concern.

One point worth mentioning is that the GDS dataset's average sentence length is tripled SemEval's (85.95 vs. 27.14), which makes the task more challenging.

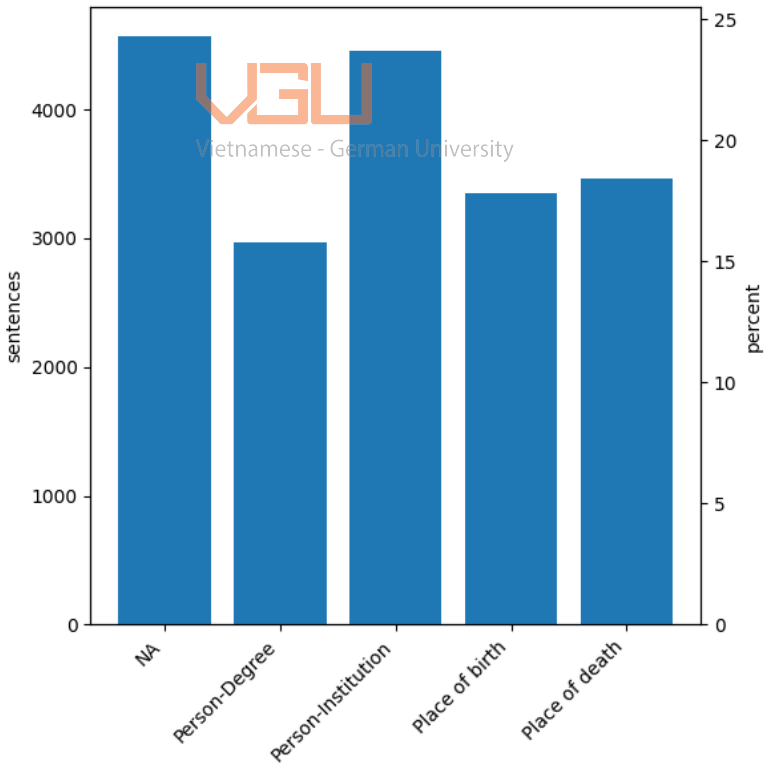

Figure 4.2: GDS dataset relation distribution

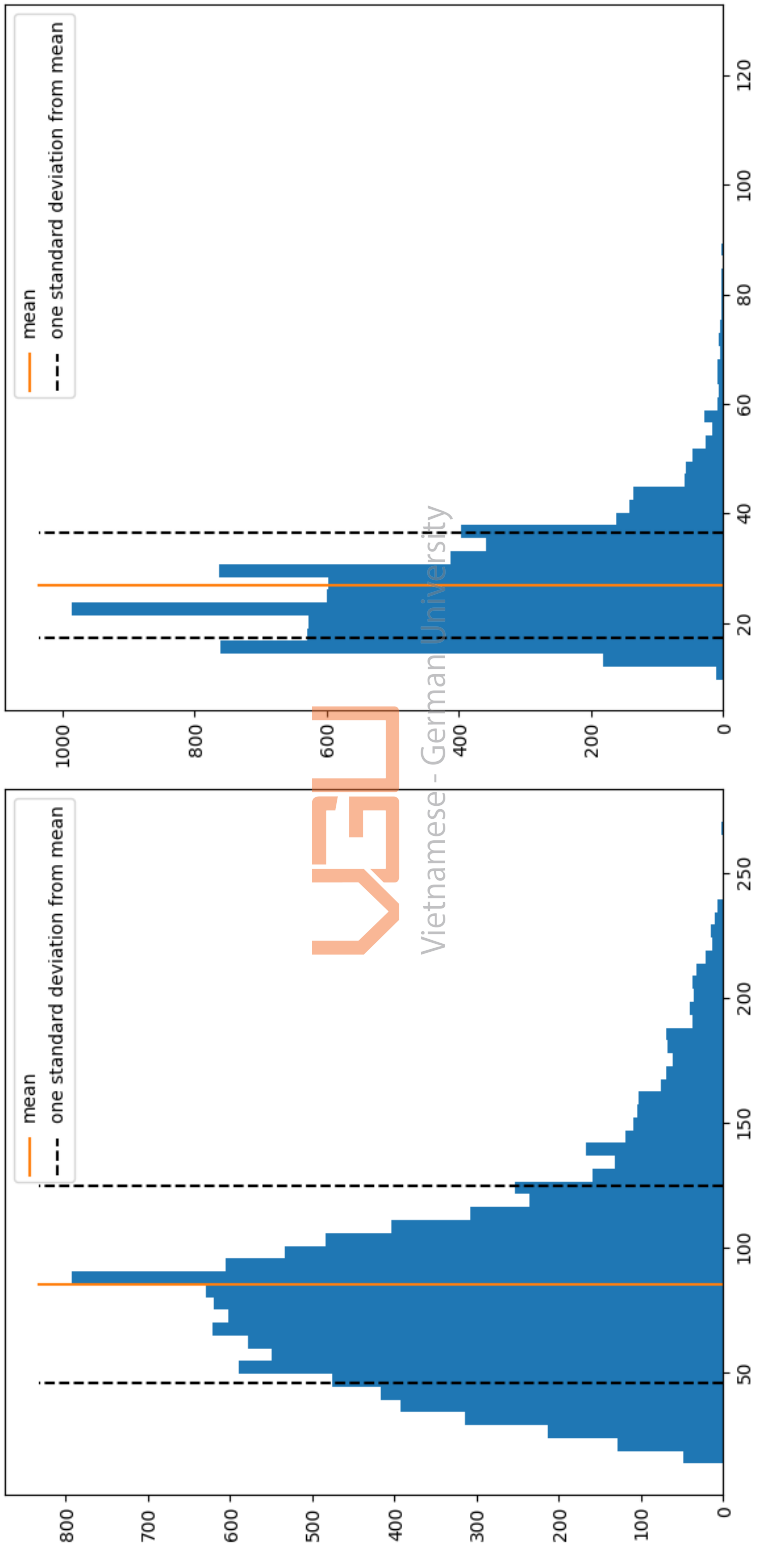

Figure 4.3: GDS dataset sentence length distribution (left) vs. SemEval's (right)

4.2 Experiment result

4.2.1 SemEval 2010 Task8 dataset

The results can be found in Table 4.1 to 4.4.

In the single classifier architecture, the $MEAN$ model yields the best result: marco F1 is 84.45% and the official score is 90.12%.

In the duo classifiers architecture, its sub-classifiers perform best when the CLS_ENT model is used. We then combine these two and evaluate the test dataset. The final marco F1 is 82.4% and the official score is 89.98%, slightly worse than the best single-classifier architecture.

In comparison with state-of-the-art models, our model does not fall far behind.

Model	Model params	Dropout	Activation	Batch	#epoch	MacroF1	Official score
ALL	linear_size = 512	0.2	Tanh	32	5	81.95%	89.82%
MEAN	linear_size = 256	0.2	Tanh	8	6	**84.45%**	**90.12%**
CLS	linear_size = 256	0	PReLU	16	6	84.07%	89.82%
CLS_ENT	cls_size = 64 entity_size = 32	0	PReLU	16	6	84%	89.76%
GRU	GRU_hidden_size = 64 GRU_layer = 2 linear_size = 1024	0.2	Tanh	32	5	82.19%	88.47%

Table 4.1: Single classifier architecture. Best result of each model for SemEval2010 Task 8 dataset.

Model	Model params	Dropout	Activation	Batch	#epoch	Accuracy	F1
ALL	linear_size = 512	0.2	PReLU	32	5	90.67%	94.47%
MEAN	linear_size = 256	0.2	PReLU	16	3	91.87%	95.16%
CLS	linear_size = 256	0.2	PReLU	8	4	91.68%	95.08%
CLS_ENT	cls_size = 64 entity_size = 32	0.2	PReLU	8	4	**92.21%**	**95.37%**
GRU	GRU_hidden_size = 32 GRU_layer = 1 linear_size = 256	0.2	Tanh	8	4	91.76%	95.15%

Table 4.2: Two classifiers architecture. Best result of binary classifier for SemEval2010 Task 8 dataset.

Model	Model params	Dropout	Activation	Batch	#epoch	Accuracy	Macro F1
ALL	linear_size = 512	0.2	PReLU	32	5	95.17%	88.09%
MEAN	linear_size = 256	0.2	PReLU	8	5	95.16%	88.48%
CLS	linear_size = 512	0.2	PReLU	32	6	94.8%	87.98%
CLS_ENT	cls_size = 64 entity_size = 32	0.2	PReLU	8	5	**95.23%**	**88.39%**
GRU	GRU_hidden_size = 64 GRU_layer = 1 linear_size = 1024	0.2	PReLU	8	5	95.2%	88.31%

Table 4.3: Two classifiers architecture. Best result of multiclass classifier for SemEval2010 Task 8 dataset.

Model	Official score F1	Authors	Year
EPGNN	90.2	Zhao et al. [21]	2019
Our one-classifier model	90.12		
Our two-classifier model	89.98		
BERTEM+MTB	89.5	Soares et al. [22]	2019
R-BERT	89.25	Zhao et al. [23]	2020
KnowBert-W+W	89.1	Wu et al. [24]	2019
Entity-Aware BERT	89.0	Zhao et al. [25]	2019
Att-Pooling-CNN	88.0	Wang et al. [26]	2016
SpanRel	87.4	Jiang et al. [27]	2019
TRE	87.1	Alt et al. [28]	2019
Entity Attention Bi-LSTM	85.2	Lee et al. [29]	2019

Table 4.4: Our model performance in comparison with other models on SemEval2010 Task 8 dataset.

4.2.2 GDS dataset

We conducted similar experiments on this dataset as on SemEval. However, due to the time limitation, we only tested the one-classifier architecture, which from our experience, demonstrates better outcomes.

Follows Vashishth et al., we evaluate our models using the precision-recall curve and the average precision score. The best results from the five models are summarized in table 4.5.

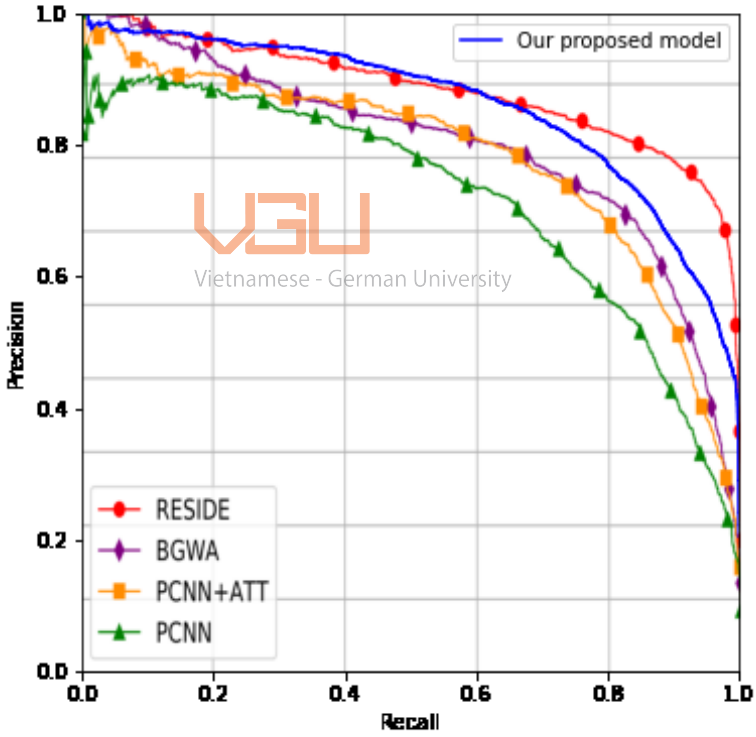

Figure 4.4: Our model precision - recall curve in comparison with other models.

Model	Model params	Dropout	Activation	Batch	#epoch	Average precision score
ALL	linear_size = 1024	0.2	PReLU	16	3	80.35%
MEAN	linear_size = 1024	0.2	Tanh	8	3	**85.05%**
CLS	linear_size = 1024	0.2	PReLU	16	3	85.04%
CLS_ENT	cls_size = 64 entity_size = 32	0.2	PReLU	32	6	73.85%
GRU	GRU_hidden_size = 32 GRU_layer = 1	0.2	PReLU	16	3	79.61%

Table 4.5: Single classifier architecture. Best result for GDS dataset.

Again, the *MEAN* model outperforms others with an average precision of 85.05%.

We want to compare our model with others. Vashishth et al., in their paper in 2018 visualized the performance of four models, including theirs - RESIDE, on the GDS dataset. We used this graph as the basis for our comparison.

As shown in Figure 4.4, our proposed model yields a better result than three others but is still unable to take the state-of-the-art position.

4.3 Discussion

Single classifier architecture vs. Duo classifiers architecture

The most unexpected result in our experiments is the out-performance, although only by a small margin, of the single classifier architecture over the duo-classifier architecture.

The motivation behind assigning "no relation" detection to a separate sub-module based on two observations:

1. The "no relation" constitutes the largest portion in both datasets (and very likely in other corpora as well)

2. It is easily misclassified the most due to its semantic and lexical diversity. (See Figure 4.5)

We cannot give a complete explanation for the unexpected performance of the duo-classifier. It can be likely that the "no relation" class needs to make up to a larger portion for the duo-classifier to be helpful. Or perhaps we had not fine-tuned the model properly. Or it could be that the two classifiers would yield better results when trained simultaneously. In either way, more experiments must be conducted before any meaningful conclusion can be drawn.

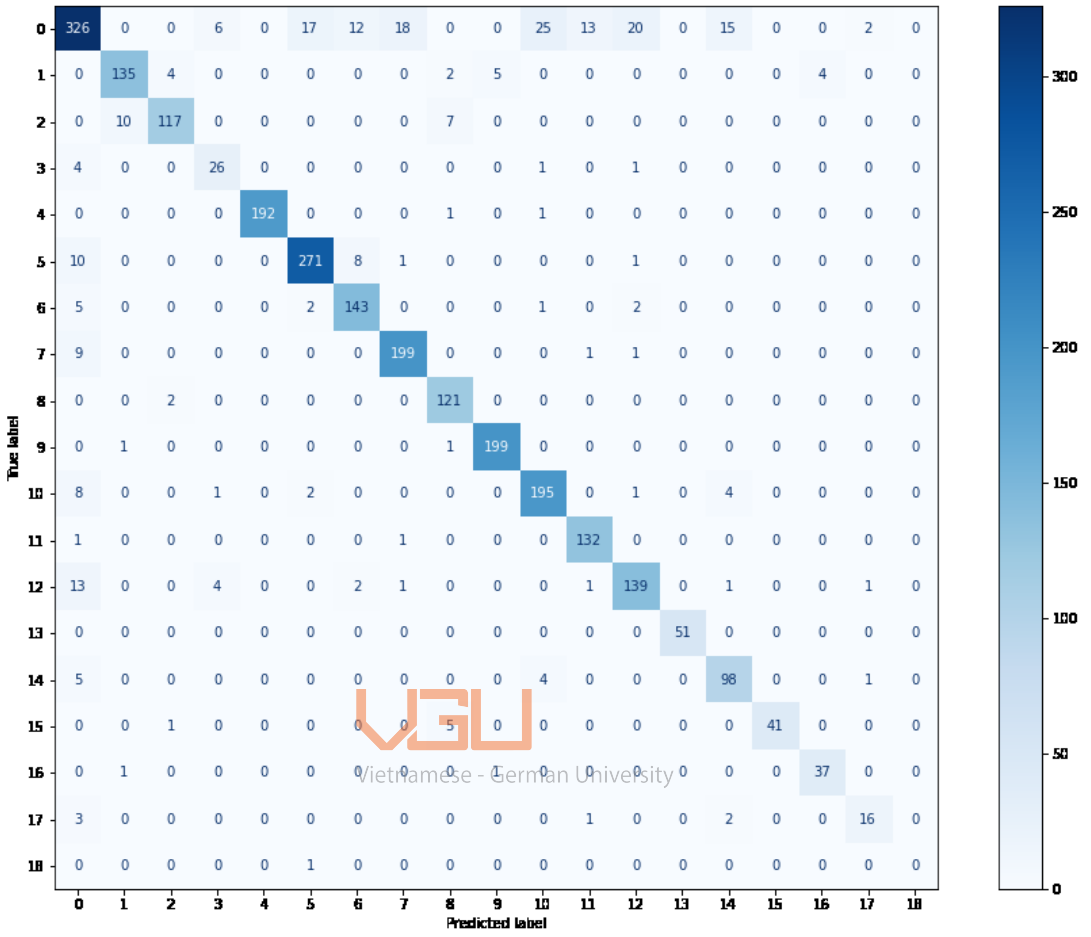

Figure 4.5: Confusion matrix of our best model on SemEval 2010 Task 8 test dataset. Here, the 0 label represents the "Other" class.

The impact of duplicated sentences in the training dataset

As mentioned in Session 3.1, for each sentence in the "not-related" class, we created a copy with the subject and object annotation swapped. The motivation for this setup also emerged from the fact that this class is the most misclassified one.

We deducted that more examples might help and our experiments showed that indeed, cloning does improve the models' performance by about 1% on both datasets.

BERT-based models dominance

Excluding the BERT word presentation, the rest of our models are relatively simple and straight forward. Nevertheless, they manage to perform quite well in comparison to much more complicated ones.

Looking at Table 4.4 closely, we can see a clear dominance of BERT-based models in the SemEval 2010 Task 8's top performers.

This indicates that BERT provides a very rich word representation, capturing lexical, grammatical, and semantic features.

Model's convergence

While experimenting with the GDS dataset, we observed a few times the models struggled to converge or got stuck in a "bad position". Also, the result's standard deviation is higher than SemEval2010 dataset's. This is as expected because the average sentence length in the GDS dataset is three times larger, making the task more challenging.

Chapter 5

Conclusions

In the last decades, the volume of online resources, especially text, has increased at a magnificent speed. How to extract information efficiently from that data becomes an exciting research topic. That is our motivation for investigating the task of relation extraction in this thesis.

In our proposed model, BERT is chosen to be the word embedding layer for its robustness and the ability to capture a word meaning in the sentence context.

We experimented with many different networks on top of it. In particular, we used:

- All of BERT's output (ALL)
- The mean of BERT's output ($MEAN$)
- BERT's output of the [CLS] token (CLS)
- BERT's output of the [CLS] token in conjunction with the subject and object (CLS_ENT)
- GRU network (GRU)

We also tested the idea of assigning "no relation" detection to a separate submodule.

Overall, the one-classifier $MEAN$ architecture yields the best performance. On the SemEval 2010 Task 8 dataset, it achieved the official F1 score of 90.12%, and on the GIDS dataset it reached the average precision score of 85.05%.

The duo classifier architecture does not fall far behind with the F1 score of 89.98% on the former dataset. However, in this setup, the CLS_ENT model yields the best outcomes.

These results, we firmly believe, can be further improved in future researches. One particular idea would be using other sentence features in addition to BERT's output for prediction. Those features can be part of speech tagging, sentence structure encoding, sematic dependencies, etc. In the duo classifier architecture, the two models can be trained simultaneously, and we can also try various settings with CNN network. The possible improvements are endless.

With this thesis and its modest results, we hope to contribute our effort to the advancement of the relation extraction task.

References

[1] M. M. Lopez and J. Kalita, "Deep learning applied to nlp," *arXiv preprint arXiv:1703.03091*, 2017. ix, 7

[2] "Embeddings: Translating to a lower-dimensional space." https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space, 2020. Accessed: 2020-08-07. ix, 10

[3] M. S. Z. Rizvi, "Demystifying bert: A comprehensive guide to the groundbreaking nlp framework." https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/, 2019. Accessed: 2020-08-08. ix, 13

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. ix, 12, 13, 15

[5] S. Kostadinov, "Understanding gru networks." https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be, 2020. Accessed: 2020-08-27. ix, 17, 18

[6] C. Olah, "Neural networks, types, and functional programming." https://colah.github.io/posts/2015-09-NN-Types-FP/, 2015. Accessed: 2020-08-27. ix, 19

[7] C. McCormic, "Smart batching tutorial - speed up bert training!." https://www.youtube.com/watch?v=ynOZUNnbEWU, 2020. Accessed: 2020-08-27. ix, 23, 24

[8] N. Bach and S. Badaskar, "A review of relation extraction," *Literature review for Language and Statistics II*, vol. 2, pp. 1–15, 2007. 2

[9] F. Hogenboom, F. Frasincar, U. Kaymak, and F. De Jong, "An overview of event extraction from text.," in *DeRiVE@ ISWC*, pp. 48–57, Citeseer, 2011. 4, 5

[10] D. Zhang and D. Wang, "Relation classification via recurrent neural network," *arXiv preprint arXiv:1508.01006*, 2015. 5

[11] M. Miwa and M. Bansal, "End-to-end relation extraction using lstms on sequences and tree structures," *arXiv preprint arXiv:1601.00770*, 2016. 5

[12] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 2335–2344, 2014. 6

[13] Y. Shen and X. 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), pp. 2526–2536, The COLING 2016 Organizing Committee, Dec. 2016. 6

[14] S. Ghelani, "From word embeddings to pretrained language models — a new age in nlp — part 1." https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models-a-new-age-in-nlp-part-1-7ed0c7f3dfc5, 2019. Accessed: 2020-08-07. 9, 10, 11

[15] R. Neskorozhenyi, "Word embeddings in 2020. review with code examples." https://towardsdatascience.com/word-embeddings-in-2020-review-with-code-examples-11eb39a1ee6d, 2020. Accessed: 2020-08-07. 10

[16] M. Yao, "What every nlp engineer needs to know about pre-trained language models." https://www.topbots.com/ai-nlp-research-pretrained-language-models/, 2019. Accessed: 2020-08-08. 12

[17] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018. 12

[18] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations. arxiv 2018," *arXiv preprint arXiv:1802.05365*, 1802. 12

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019. 16, 17

[20] M. Benesty, "Divide hugging face transformers training time by 2 or more with dynamic padding and uniform length batching." https://towardsdatascience.com/divide-hugging-face-transformers-training-time-by-2-or-more-21bf7129db9q-21bf7129db9e, 2020. Accessed: 2020-08-27. 23

[21] Y. Zhao, H. Wan, J. Gao, and Y. Lin, "Improving relation classification by entity pair graph," in *Asian Conference on Machine Learning*, pp. 1156–1171, 2019. 38

[22] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," *arXiv preprint arXiv:1906.03158*, 2019. 38

[23] S. Wu and Y. 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*, pp. 2361–2364, 2019. 38

[24] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," *arXiv preprint arXiv:1909.04164*, 2019. 38

[25] H. Wang, M. Tan, M. Yu, S. Chang, D. Wang, K. Xu, X. Guo, and S. Potdar, "Extracting multiple-relations in one-pass with pre-trained transformers," *arXiv preprint arXiv:1902.01030*, 2019. 38

[26] L. Wang, Z. Cao, G. De Melo, and Z. 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)*, pp. 1298–1307, 2016. 38

[27] Z. Jiang, W. Xu, J. Araki, and G. Neubig, "Generalizing natural language analysis through span-relation representations," *arXiv preprint arXiv:1911.03822*, 2019. 38

[28] C. Alt, M. Hübner, and L. Hennig, "Improving relation extraction by pre-trained language representations," *arXiv preprint arXiv:1906.03088*, 2019. 38

[29] J. Lee, S. Seo, and Y. S. Choi, "Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing," *Symmetry*, vol. 11, no. 6, p. 785, 2019. 38

[30] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers:

State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need. arxiv 2017," *arXiv preprint arXiv:1706.03762*, 2017.

[32] S. Jat, S. Khandelwal, and P. Talukdar, "Improving distantly supervised relation extraction using word and entity based attention," *arXiv preprint arXiv:1804.06987*, 2018.

[33] S. Vashishth, R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar, "Reside: Improving distantly-supervised neural relation extraction using side information," *arXiv preprint arXiv:1812.04361*, 2018.

[34] M. Xiao and C. Liu, "Semantic relation classification via hierarchical recurrent neural network with attention," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1254–1263, 2016.

[35] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. 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)*, pp. 207–212, 2016.

[36] R. Bunescu and R. Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 724–731, 2005.

[37] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang, "A dependency-based

neural network for relation classification," *arXiv preprint arXiv:1507.04646*, 2015.

[38] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin, "Improved relation classification by deep recurrent neural networks with data augmentation," *arXiv preprint arXiv:1601.03651*, 2016.

[39] R. Cai, X. Zhang, and H. Wang, "Bidirectional recurrent convolutional neural network for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 756–765, 2016.