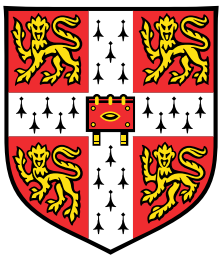
Relation Classification based on Deep Learning Approach



Yixuan Su

Department of Engineering
University of Cambridge

MPhil in Machine Learning, Speech and Language Technology
Master of Philosophy

Selwyn College August 2018

I would like to dedicate this thesis to my loving parents ...

Declaration

I, Yixuan Su of Selwyn College, being a candidate for the M.Phil in Machine Learning, Speech and Language Technology, hereby declare that this thesis and the work described in it are my own work, unaided except as may be specified below, and that the thesis does not contain material that has already been used to any substantial extent for a comparable purpose.

Word Count - 14843

Signed - Su, Yixuan 苏熠暄

Yixuan Su
August 2018

Acknowledgements

I would sincerely like to thank my supervisors, Prof. Anna Korhonen and Dr. Simon Baker for their expert advice and support, whilst also giving me the freedom to work on things of my interest. They have been tremendously supportive and have guided my work to the fullest. Without their ideas, support and patience, I would not have this project see this day.

I would also like to thank other members in the Language Technology Lab at University of Cambridge for their enormous support. And I would like to thank all lecturers in my course for teaching me so much valuable knowledge. And I am lucky to meet all other students in my course. I would like to express my special thanks to Marco, Tianyu, Yuanzhao, and Minglong for their help and advice through my MPhil Degree.

I am grateful to Selwyn College for providing me perfect environment to pursue my study in Cambridge. Only with your care and support, I could finish this course smoothly.

I would like to thank my family, without your support through all these years, I could not go this far. You always put my well-being and academic interests over everything else, I owe you all everything.

I am always thankful to my girlfriend, Jialu, for taking care of me from high school to master, from Beijing to Cambridge. And I am grateful to have this opportunity to spend this wonderful year with you in Cambridge. I am looking forward to spending more great years with you in my life.

Abstract

Relation classification and relation extraction are fundamental tasks in natural language processing (NLP) and information extraction with many downstream applications,such as knowledge base construction, and question answering. In this thesis, we propose a novel attention-based recurrent neural network model to tackle both of these tasks. For the relation classification task, we propose a composition model consisting of an attention-based classifier and a variational autoencoder acting as a regularizer. As for the relation extraction task, we first reformulate the task as a sequence tagging problem, and then propose a sequence model based on previous attention-based classifier. To evaluate our model comprehensively, for each task, we conduct experiments on three different established datasets from general and specialized domains. We use Bayesian Optimization to fine tune the model hyperparameters, and are able to achieve at least on par performance with the current state-of-the-art published models across all datasets, we significantly outperform the best state-of-the-art results on some of these datasets.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Relation extraction is an important Natural Language Processing (NLP) task with various downstream applications; for example, information extraction [74], question answering [81], and medical informatics [69].

The aim of relation extraction is to categorize the relation between two entities of interest into a set of predefined relation types. Traditional methods always solve this task in two steps: (1) entity recognition, (2) relation classification. In the first step, the system pinpoints the possible pairs of entities that may share a relation. In the second step, the system classifies the relation for every possible pair of entities. In this thesis, we focus on two goals: (1) solving relation classification task and (2) solving relation extraction task in general.

For relation classification, we focus on solving the task at sentence level. Given a sequence $X_L = \{x_1, ..., x_T\}$ with length T, and two marked nominals e1 and e2, the goal is to predict a relation $r \in R$ between e1 and e2, where R is a set of predefined relation labels. For instance, given a sequence: "A trillion gallons of $[\text{water}]_{e1}$ have been poured into an empty $[\text{region}]_{e2}$ of outer space", the entities *water* and *region* have the relation of Entity-Destination(e1, e2).

Conventional relation classification methods are mostly based on pattern matching, and an obvious disadvantage is that high-level features such as tags of part of speech (POS), name entities and dependency path [6] are often used. These requirements make the system rely on external NLP modules which not only increases computational cost, but also introduces external errors. Another important point is that manually designing patterns is prohibitively time-consuming and often results in low coverage.

Recent advancement in machine learning and deep neural networks have enabled new ways to reduce the dependency on manually designed features and patterns. For example, a general framework is proposed by Collobert et al. [11] to learn task-oriented features by using convolution neural networks (CNN) from raw text. Collobert et al. [11] evaluated

the learning approach on several NLP tasks such as part-of-speech tagging, name entity recognition (NER) and semantic role labelling (SRL). The results show that using the features learnt from raw text can achieve close to or even better performance than state-of-the-art systems that utilize manual feature engineering.

Deep learning methods have also been proposed for relation classification; for example, work by Zeng et al. [86] proposes a CNN-based approach that achieves competitive results without the help of external knowledge. Following the success of CNN, many other models such as multi-window CNN [45], CR-CNN [54] and NS-depLCNN [76] have been proposed.

Although CNN models can deliver good performance on this task, they have some notable drawbacks; for example, they do not learn temporal patterns very well. Since the semantic meaning of a relation is formed within the context of two target nominals (entities), this context includes the words between the nominals and words surrounding them.

Relations that are directional are sensteive to word order, for example, "London is in England". Therefore, different ordering of a word sequence may represent a different relation. As a result, semantic relation classification is similar to the task of temporal sequence modelling rather than feature extraction using CNN models [87].

A Recurrent neural network (RNN) is the most common choice to learn temporal knowledge from sequential data. Therefore, many models based on RNNs have been proposed for the task of semantic relation classification. For example, Zhang et al. [87] designed bidirectional RNNs to directly model temporal knowledge, and Vu et al. [68] proposed the Connectionist Bi-directional RNNs and combine it with a CNN-based model through a voting scheme to achieve competitive results. A well known drawback of traditional RNNs is that during learning process it equally focuses on each time-step. To help RNN-based models better select input features, the attention mechanism is widely used in many applications, i.e. machine translation [2]. With the attention mechanism, the RNN model can automatically focus more on inputs from important time-steps and ignore others. Zhang et al. [89] proposed attention-based model and achieve on a par with state-of-the-art performance, which further proves the effectiveness of non-static models over static ones.

A good relation classification model can be used as an intermediate component for solving relation extraction tasks in general. Conventional relation extraction systems handle the extraction process in a pipelined manner, i.e. entity recognition [43] and relation classification [52]. Improving each of these components can increase the overall system performance. One disadvantage of this is that pipelined systems ignore the internal connection of these two steps, and thus errors from entity recognition often propagate to the relation classification, leading to weaker overall results [31].

In contrast to pipelined extraction methods, end-to-end systems make use of joint learning to extract entities and corresponding relations together in one unified model. By leveraging both knowledge from entities and relations, it has been shown that joint learning framework achieves better performance in relation extraction task. However, most of existing joint models are feature-based systems [31], [42] and [84]. For these systems, complicated feature engineering is required. To reduce the manual effort of feature construction, Miwa and Bansal [41] proposed a neural network based method for end-to-end entities and relation extraction. Recently, Zheng et al. [90] proposed to transform the relation extraction task into sequence tagging task, and with the help of RNN-based end-to-end model they achieved state-of-the-art performance on one public dataset.

1.1 Contributions

In this thesis, we propose novel models to tackle both the relation classification and the relation extraction tasks. For relation classification, we propose a new framework which leverages RNNs and the attention mechanism to tackle relation classification. The model consists of two parts: an attention-based classifier and a regularizer based on a Variational Autoencoder (VAE). We conduct comprehensive experiments that utilize established relation classification tasks and datasets from both the general domain as well as specialized domain. We achieve at least on a par performance with the current state-of-the-art on these tasks, and by introducing a regularizer into our model, the performance surpasses the state-of-the-art results across all experimented tasks and datasets.

For the relation extraction task, we propose a novel sequence model based on our relation classification model. We conduct comprehensive experiments on datasets from general and specialized domains. By adopting the sequence tagging scheme for relation extraction, we achieve at least on a par with state-of-the-art performances across all experimented tasks and datasets.

For the **relation classification** task, the main contributions of this thesis are as follows:

- Propose a composition model which contains attention-based RNN classifier and VAE-based language model.
- Achieve state-of-the-art performance on different datasets both from the general domain and the biomedical domain.
- Demonstrate the usefulness of introducing a VAE language model as a regularizer to the classifier for relation classification and illustrate how the performance is improved.

For the **relation extraction** task, the main contributions of this thesis are as follows:

- Propose a novel RNN-based attention sequence model to tackle the general relation extraction task as sequence tagging problem.
- Conduct comprehensive experiments on datasets from various domains, and demonstrate the effectiveness of our proposed model.

1.2 Related Work

1.2.1 Relation Classification

Various learning paradigms have been proposed to relation classification problem, and supervised approaches have shown their advantages in this task. Researchers have initially focused on engineering complex features for relation classification, i.e. feature-based or kernel-based. Suchanek et al. [63] transformed classification clues, i.e. sequences and parse trees, into feature vectors. Different kernels, such as convolutional tree kernel [50], subsequence kernel [7] and dependency tree kernel [7], have been proposed for relation classification task. However, the need of manual annotation is expensive thus the quality is always limited which encouraged the usage of distant-supervision [40].

With recent improvement of deep neural networks, many researchers have focused on using neural networks model to automatically learn features. For natural language processing (NLP), most of the existing methods are primarily based on learning distributed representation for each word, which is also called as word embedding [65]. Socher et al. [59] proposed a recurrent neural network (RNN) model along with parse tree information for sentiment analysis and similar model is also used to classify relations [58]. Hashimoto et al. [17] employed a neural relation extraction model allowing for explicit weighting of important phrases. Zeng et al. [86] proposed to use convolutional neural networks (CNN) to leverage both lexical and sentence level feature together to make final classification. Santos et al. [54] proposed a ranking loss function along with CNN architecture model to perform relation classification.

Another line of research relates to attention mechanism for deep learning. Bahdanau et al. [2] first proposed to use attention mechanism in machine learning task for NLP. This attention mechanism selects the most relevant reference words in original sequence for words in a foreign language before translation. Xu et al. [75] used attention mechanism to generate caption for images. This mechanism helps the model select relevant image regions when generating captions. Pei et al. [48] leveraged attention mechanism along with

modified recurrent unit for speech recognition and sentiment analysis. Further usage of attention mechanism includes image question answering [32], document classification [80], and paraphrase identification [82]. Shen et al. [55] explored word level attention mechanism to discover better patterns in heterogeneous context for relation classification task.

In this thesis, we propose a novel attention-based architecture to determine relevance of each input word to the relation classification output. We also incorporate a variational autoencoder (VAE) into the entire framework to act as a regularizer for our classifier.

To the best of our knowledge, this is the first work in published literature which uses VAE for relation classification.

1.2.2 Relation Extraction

Relation Extraction is an important task in NLP literature. It can be used as an end-user application or as an intermediate step to further downstream applications, such as biomedical text-mining [5] and textual entailment [13].

Two general frameworks are often used to extract entities and their corresponding relations. One is pipelined method and the other is joint learning method.

The pipelined method treats relation extraction task as two step process, i.e. (1) name entity recognition (NER) [43] and (2) relation classification (RC) [52]. Conventional NER models are linear statistical models, such as Hidden Markov Models (HMM) and Conditional Random Fields [47] [35]. With recent development in deep learning, neural networks architectures [29] have been applied to NER task. Current relation classification models can also be divided into two categories: (1) handcrafted feature based method [52] and neural network based methods [54].

The joint learning method extracts entities and relations with an unified model. Most of the joint models are feature-based system [31] [51]. To reduce manual effort, Miwa and Bansal [41] proposed to use LSTM-based model to extract entities and relations separately. Recently, Zheng et al. [90] proposed a unified framework to transform relation extraction to a sequence tagging problem.

1.3 Thesis Outline

In this section we summerize the thesis outline.

Chapter 2: We detail our methodology for the relation classification task. The basic components of our model architecture are introduced and training objectives are discussed.

We also discuss the optimization challenges for our proposed model, we then describe how to use Bayesian Optimization as our hyperparameter tuning approach.

Chapter 3: We introduce our methodology for relation extraction task. We detail how to transform relation extraction into sequence tagging problem and we also describe how to adapt our proposed model from Chapter 2 to fit this task.

Chapter 4: We apply our methodology from Chapter 2 for various relation classification datasets. To fully evaluate our model performance, we use datasets from both general domain and biomedical domain. To better illustrate how our proposed model works. We provide concrete visualization of our model's results; the generative perspective of our model is also discussed in detail. Comprehensive analysis of our experiments' results are discussed along with a comparison between our model and the currently published state-of-the-art systems.

Chapter 5: We adopt our method from Chapter 3 to conduct experiments for relation extraction on datasets from general, scientific and biomedical domains. To fully understand the importance of each part contained in our model, we also perform experiments with various configurations of our proposed system. A detailed analysis of our experiments' results are also presented.

Chapter 6: We give an overall conclusion and summary of our presented work, and a brief discussion of possible future work.

Chapter 2

Relation Classification Methodology

When it comes to sequence modeling, the most common neural network architecture is a Recurrent Neural Network (RNN). There are multiple variations of RNNs that are described in literature, and different techniques have been developed to improve the performance of RNNs. For the relation classification task, we design an end-to-end model that combines RNNs and Variational Autoencoder (VAE). In section 2.1, we introduce several variations of RNNs and their corresponding advantages and disadvantages. In section 2.2, we describe the theory behind VAEs and how it can be incorporated into relation classification tasks. In section 2.3, we propose a novel attention-based model architecture. In section 2.4, we demonstrate how all the components can be combined together to deliver optimal performance. In section 2.5, we describe how to use Bayesian Optimization to automatically tune the hyperparameters of a neural network.

2.1 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a neural sequence model that achieves state-of-the-art performance on important tasks that include language modeling [39], speech recognition [15] and machine translation [23]. RNNs are used to deal with data with sequential structure which makes it a natural choice for us to build an RNN-based model for relation classification tasks. In this section, we describe several variations of RNNs and how they can be used to construct our model.

2.1.1 Basic Recurrent Neural Network

The vanilla architecture of a Recurrent Neural Network can be formulated as Equation (2.1), where x_t and h_t are the input and hidden state at time-step t. And σ is nonlinear activation

function which is usually the *tanh* function described in Equation (2.2). The softmax function in Equation (2.3) is used to calculate the probability distribution over all possible outcomes (classes) of the next word (i.e. a language model) or the probability distribution over possible entity tags (e.g. name entity recognition).

$$\begin{aligned} h_t &= \sigma(W^{hh}h_{t-1} + W^{hx}x_t) \\ y &= softmax(W^{(s)}h_t) \end{aligned} \tag{2.1}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^T \mathbf{w}_k}} \tag{2.3}$$

In Figure 2.1, we show an illustration of an RNN unrolled across 3 time-steps, and we can see that the input information is passed through the hidden state propagation which makes the RNN useful for sequential data.

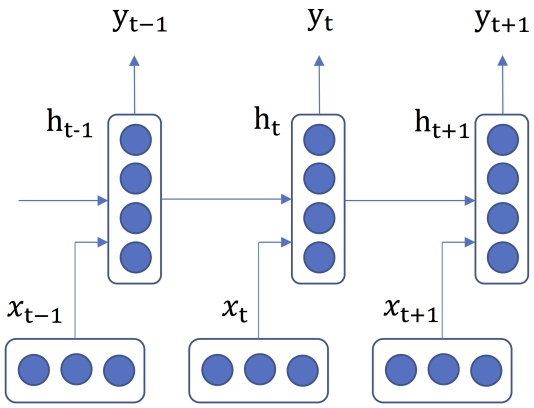


Fig. 2.1 Illustration of vanilla RNN architecture

Although vanilla RNN is able to capture sequential dependencies in the sequence, it tends to forget past information when the sequence gets too long. To address this problem, memory mechanism is introduced into different variations of the vanilla RNN, in the following subsections we focus on two variations of the vanilla RNN.

2.1.2 Long Short-Term Memory Network

Long Short-Term Memory Networks (LSTMs) [19] addresses the shortcomings of the vanilla RNN by introducing a memory mechanism that helps to retain information from early time-steps in the sequence. To update the information in memory unit, the LSTM cell sets three logical gates: an input gate i_t for writing, a forget gate f_t for removing and an output gate o_t for reading. Each gate provides the proportion of information that is involved in the corresponding operation which is determined by the input x_t and the previous hidden state h_{t-1}. An illustration of unrolled LSTM across three time-steps is shown in Figure 2.2[1].

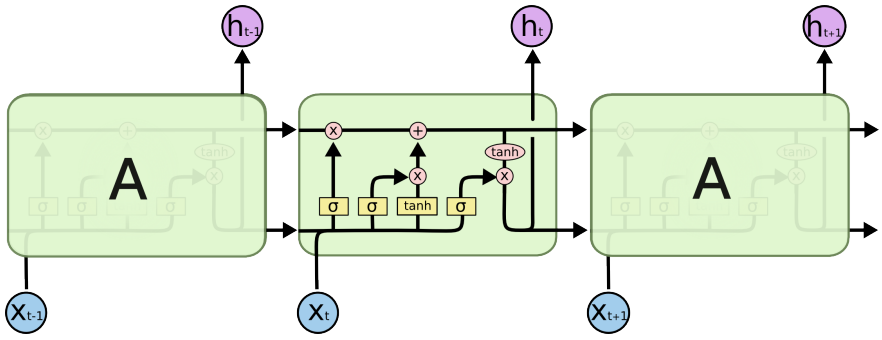


Fig. 2.2 Illustration of unrolled LSTM

The detalied calculations are shown in Equation (2.4). By introducing additional logical gates, LSTM is able to capture more knowledge from input and generally delivers better performance than vanilla RNN.

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1} + b_i) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1} + b_f) \\
g_t &= tanh(W^c x_t + U^c h_{t-1} + b_c) \\
c_t &= i_t * g_t + f_t * c_{t-1} \\
o_t &= \sigma(W^o x_t + U^o h_{t-1} + b_o) \\
h_t &= o_t * tanh(c_t)
\end{aligned}
\tag{2.4}
$$

2.1.3 Gated Recurrent Unit

The second variation of vanilla RNN unit is the Gated Recurrent Unit (GRU) [9]. A gated recurrent unit is designed to make each recurrent unit to adaptively capture dependencies of

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

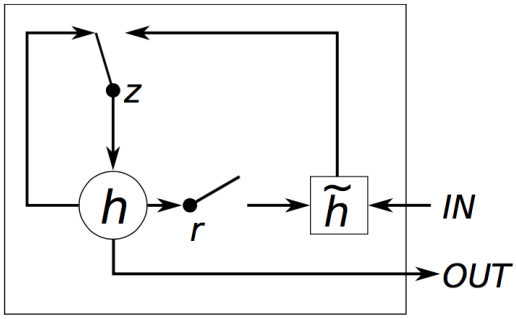


Fig. 2.3 Graphical illustration of GRU

different time-steps. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit.

The graphical illustration of the GRU is shown in Figure 2.3, and detailed calculations are shown in Equation (2.5). It can be seen that GRU parameters are very similar to LSTM parameters, however, without having a separate memory cells. For same parameters, the GRU units can deliver similar or better performance than LSTM unit on various sequence modeling tasks [10].

$$\begin{aligned} z_t &= \sigma(W^z x_t + U^z h_{t-1} + b^z) \\ r_t &= \sigma(W^r x_t + U^r h_{t-1} + b^r) \\ \tilde{h} &= tanh(W^h x_t + U^h h_{t-1} * r_t + b^h) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{2.5}$$

2.1.4 Challanges in training RNNs

Here we describe several challanges which are commonly encountered when training RNNs. The first one is the vanishing gradient problem [73] which is caused by the saturation of activation functions. To see this problem directly, we plot two popular activation functions *tanh* and *sigmoid* which are shown in Figure 2.4 and 2.5. From the plots it can be seen that for the *tanh*, once the function value approaches -1 or 1, the gradient is approximately zero, for sigmoid function, this happens when function approaches 0 or 1. Generally, the longer the sequence is the more likely vanishing gradients problem happens.

One solution to this problem is backpropagation through time (BPTT) [73]. This technique addresses the vanishing gradient problem by only unfolding a range of steps instead of entire sequence, as a result the training procedure of RNN can be more stable and faster.

The second problem we commonly encounter when training RNNs is overfitting. And LSTM and GRU are more likely to overfit than traditional RNN due to more parameters contained in recurrent unit [85]. Traditional regularization technique like dropout [60] cannot be directly imposed on weights of recurrent units. To this end, dropout on recurrent neural network [85] addresses the regularization problem by only using dropout on input and output of recurrent layers. As a result, the overfitting problem can be alleviated.

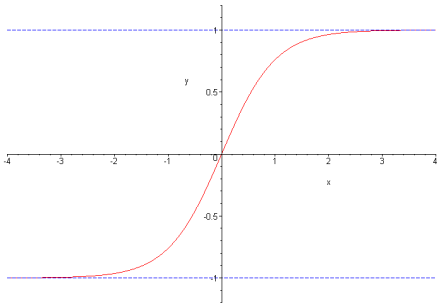


Fig. 2.4 Tanh function

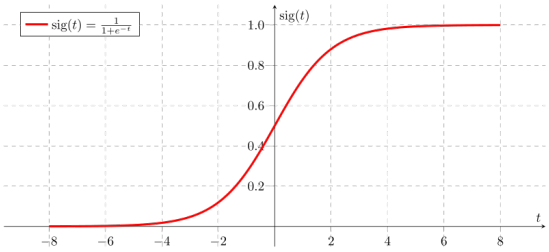


Fig. 2.5 Sigmoid function

2.2 Variational Autoencoder

In this subsection, the second main building block in our model is introduced which is the variational autoencoder (VAE) [24]. We first describe the VAE in the perspective of a probability model.

A variational autoencoder models the probability of data x and latent variables z. The joint probability can be formulated as $p(x,z) = p(x|z)p(z)$. For each datapoint i the generative process can be written as follows:

- Draw latent variables $z_i \sim p(z)$
- Draw datapoint $x_i \sim p(x|z_i)$

The directed graphical model is represented in Figure 2.6. The latent variables are drawn from a prior $p(z)$ and the data x has a likelihood $p(x|z)$ that is conditioned on latent variables z. The generative model defines a joint probability distribution over data and latent variables: $p(x,z)$ and this distribution can be further decomposed into $p(x,z) = p(x|z)p(z)$.

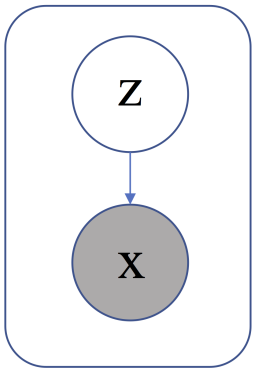


Fig. 2.6 Graphical illustration of VAE

VAE inference requires approximating good values of the latent variables given observed data by calculating the posterior $p(z|x)$ shown in Equation (2.6). The denominator $p(x)$ can be calculated by marginalizing out the latent variables as Equation (2.7). Unfortunately, this integral is always intractable to calculate, thus an approximation of the posterior distribution is required.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2.6}$$

$$p(x) = \int p(x|z)p(z)dz \tag{2.7}$$

Variational inference approximation of the posterior can be modelled as $q_\lambda(z|x)$ where λ indexes the family of distributions. For instance, if q were Gaussian, λ denotes the mean and variance of each datapoint $\lambda_{x_i} = (\mu_{x_i}, \sigma^2_{x_i})$.

To measure how well the variational posterior $q(z|x)$ approximates the true posterior $p(z|x)$, Kullback-Leibler divergence (KL-Divergence) [27] [26] shown in Equation (2.8) is used. To calculate better approximation, the goal is to minimize the KL-Divergence which means calculating parameters λ that satisfies Equation (2.9).

$$KL(q_\lambda(z|x)||p(z|x)) = \mathbb{E}_q[logq_\lambda(z|x)] - \mathbb{E}_q[logp(x,z)] + logp(x) \tag{2.8}$$

$$q^\star_\lambda(z|x) = \arg\min_\lambda KL(q_\lambda(z|x)||p(z|x)) \tag{2.9}$$

But the KL-Divergence is still impossible to compute directly due to the existence of evidence term $p(x)$. To address this problem, we consider the function in Equation (2.10) and combine it with KL-Divergence to rewrite the evidence as Equation (2.11).

$$ELBO(\lambda) = \mathbb{E}_q[logp(x,z)] - \mathbb{E}_q[logq_\lambda(z|x)] \tag{2.10}$$

$$logp(x) = ELBO(\lambda) + KL(q_\lambda(z|x)||p(z|x)) \tag{2.11}$$

According to Jensen's inequality [22], the KL-Divergence is always greater than or equal to zero. This means that minimizing KL-Divergence is equivalent to maximizing the ELBO (Evidence Lower Bound) which allows us to do approximate posterior inference. Because this helps us to bypass the need of computing and minimizing KL-Divergence, thus we only need to maximize the ELBO which is computationally tractable. Notice that in the variational autoencoder model, there are only local latent variables (no datapoint shares its latent z with the latent variable of another datapoint). As a result, we can decompose the ELBO into a sum where each term depends on a single datapoint and this allows us to use stochastic gradient descent with respect to the parameters λ. The ELBO for a single datapoint is shown in Equation (2.12).

$$ELBO_i(\lambda) = \mathbb{E}_{q_\lambda(z|x_i)}[logp(x_i|z)] - KL(q_\lambda(z|x_i)||p(z)) \tag{2.12}$$

Now we make connections to neural networks model. The final step is to parameterize the approximate posterior $q_\theta(z|x)$ with an inference network (or encoder) and likelihood $p_\phi(x|z)$ with a generative network (or decoder). The inference and generative networks have parameters θ and ϕ respectively which are typically the weights and biases of the neural nets. The revised version of ELBO with inference and generative network parameters is shown in Equation (2.13).

$$ELBO_i(\theta,\phi) = \mathbb{E}_{q_\theta(z|x_i)}[logp_\phi(x_i|z)] - KL(q_\theta(z|x_i)||p(z)) \tag{2.13}$$

The prior over latent variable $p(z)$ is always chosen as a normal distribution with zero mean and standard deviation of 1 which is shown as $p(\mathbf{z}) \sim N(\mathbf{0},\mathbf{I})$, because in this form we can easily reparameterize the parameters into gradient descent process. It is proved that with complex enough neural network we can regenerate any form of distribution from simple normal distribution [24].

2.3 Model Framework

In this section, we describe our neural network model for the task of relation classification. As a reminder, the relation classification task is about assigning sentence with two marked entities to a relation from a predefined set of relations. For example, the sentence "We poured the <e1> milk </e1> into the <e2> pumpkin mixture </e2>." expresses the relation Entity-Destination(e1,e2). Since all input data has sequential structure, our model is based on RNN architecture. An abstract illustration can be seen from Figure 2.7. We use a discriminative model for classification and introduce a generative model for regularization. The following parts will introduce the classifier (subsection 2.3.1) and introduce generative model (subsection 2.3.2). In subsection 2.3.3, several techniques which are used to improve training performance are described.

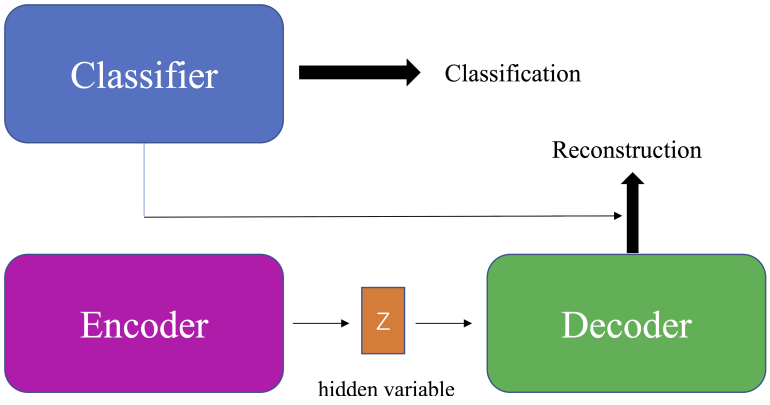


Fig. 2.7 Abstract illustration of model framework

2.3.1 Attention BiAGRU Model

Given a sequence of words as input, our goal is to: (1) calculate salience scores for each word in our input sequence, and (2) construct hidden representations based on the salience scores that is best suited for the sequence classification task. To this end, we propose the Attention BiAGRU Model (ABAGM) which contains two parts: temporal attention module and bidirectional attention-gated recurrent units. Our novel ABAGM model can be trained end-to-end efficiently. A graphical illustration of the model's architecture is shown in Figure 2.8.

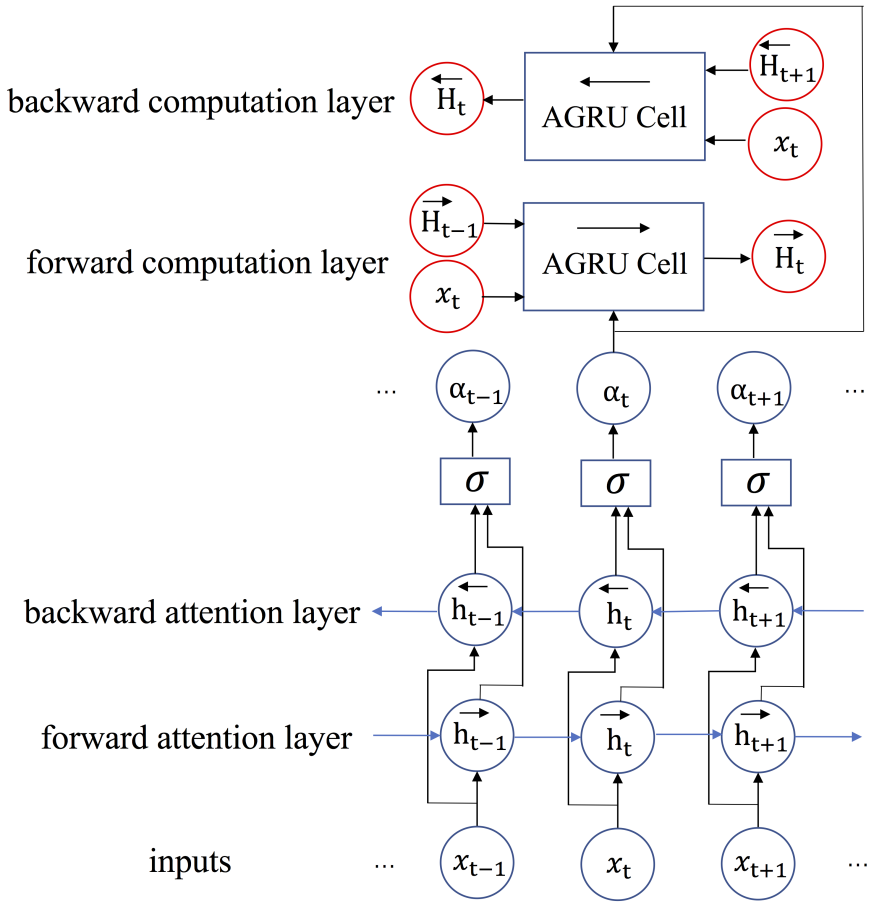


Fig. 2.8 The graphical illustration of our Attention BiAGRU Model (ABAGM). The top part of the figure is bidirectional recurrent attention-gated units and the bottom is the temporal attention module. Notice that α_t is scalar salience score which is calculated from both forward and backward attention layer hidden state. The acquired salience score is further propagated to bidirectional attention-GRU layer.

Temporal Attention Module

Attention is widely used in many machine learning tasks with sequential structure data, including object recognition [1], and machine translation [66] [2]. Incorporating the attention mechanism allows the model to pay various level of focus on different parts of the input data. Pei et al. [48] propose attention module specifically designed for sequence classification task. Inspired by this idea, we propose our novel attention model with new architecture.

The objective of this module is to estimate the relevance of each part (token) in the input sequence to the final classification. To fully capture the information carried by each token, the salience score should not only depend on current time-step but also depend on neighbouring tokens. To access more information, we consider the input sequence in both directions instead of only looking at traditional forward direction. To efficiently leverage information from both directions, we calculate the scalar salience score at time-step t using Equation (2.14).

$$\alpha_t = \sigma(\mathbf{m}^T(\overrightarrow{h}_t; \overleftarrow{h}_t) + b) \tag{2.14}$$

Where $\mathbf{m}$ is the weight vector of merging layer and b is bias term. We concatenate hidden states from both forward and backward directions, and feed it into the merging layer. To constrain the resulting salience score to the interval from zero to one, we use logistic function as nonlinear function σ. Unlike [48] which uses vanilla RNN (described in subsection 2.1.1) to calculate $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$, we use the bidirection LSTM (presented in Equation (2.4)) to model the attention hidden states. By doing so, we allow the temporal attention module to keep track of more information across the input sequence. Because the LSTM generally has larger capability than vanilla RNN to extract knowledge from long sequence data [19].

Bidirectional Attention-Gated Recurrent Units

The objective of this module is to learn a hidden sequence representation that leverages the salience attention score (from the temporal attention module) and sequential input data. In order to achieve this, we design a novel bidirectional attention-gated recurrent units which takes scalar salience score and sequential data as input. The recurrent unit structure is devised based on the original GRU cell (Equation (2.5)). The model parameters are shown in Equation (2.15), where $*$ operation stands for scalar, vector multiplication. It can be seen that at time-step t, high salience score α_t will force the output of recurrent unit H_t to focus more on current hidden state $H_t^\star$ and input x_t. On the other hand, low salience score let the model tend to ignore current input and hidden state and inherit more information from previous time-steps. Unlike previous methods [2] [48] which only leverage unidirectional attention

information, we assume attention score denotes the relevance level at different tokens of input sequence from both forward and backward directions.

$$
\begin{aligned}
z_t &= \sigma(W^z x_t + U^z H_{t-1} + b^z) \\
r_t &= \sigma(W^r x_t + U^r H_{t-1} + b^r) \\
\tilde{H}_t &= tanh(W^H x_t + U^H H_{t-1} * r_t + b^H) \\
H_t^\star &= (1 - z_t) * H_{t-1} + z_t * \tilde{H}_t \\
H_t &= (1 - \alpha_t) * H_{t-1} + \alpha_t * H_t^\star
\end{aligned}
\tag{2.15}
$$

Formally, given an input sequence $X_{1,...,T} = \{x_1, ..., x_T\}$ of length T in which $x_t \in \mathbb{R}^D$ denotes input feature at t^{th} time-step. We first feed input sequence into temporal attention module to calculate salience score sequence $A_{1,...,T} = \{\alpha_1, ..., \alpha_T\}$. Then salience score sequence is propagated into the bidirectional attention-gated recurrent units along with the input sequence feature, the results are forward hidden state sequence $\{\overrightarrow{H}_1, ..., \overrightarrow{H}_T\}$ and backward hidden state sequence $\{\overleftarrow{H}_1, ..., \overleftarrow{H}_T\}$. Instead of directly using final state of $\overrightarrow{H}_T$ and $\overleftarrow{H}_T$, we use max-pooling over time to extract the final representation of the hidden state representaion $\overrightarrow{H}_{max}$, $\overleftarrow{H}_{max}$ from both directions. The values of each dimension contained in $\overrightarrow{H}_{max}$, $\overleftarrow{H}_{max}$ are the maximum value across all time-steps. By doing this, we can extract the most salient feature across all time-steps from both directions. For the last step, the final hidden representation of the sequence is acquired by using weighted sum of $\overrightarrow{H}_{max}$ and $\overleftarrow{H}_{max}$ shown in Equation (2.16). The sum of scalar weights w_1 and w_2 are normalized to one by using the softmax function, and both weights are updated during the training process.

$$
H_o = w_1 * \overrightarrow{H}_{max} + w_2 * \overleftarrow{H}_{max} \tag{2.16}
$$

Finally, the hidden representation H_o is fed into the classifier which has a weight matrix $W_{class} \in \mathbb{R}^{D \times N}$, where N is the number of classes, in order to calculate scores for each class.

2.3.2 Variational Autoencoder Language Model

The proposed model contains two parts: the discriminative classifier and the generative regularizer. In this subsection, we introduce the generative regularizer which is a language model (LM) based on the variational autoencoder (VAE).

Model Design

The graphical illustration of this part is shown in Figure 2.9. We adopt the same structure presented by Bowman et al. [4], where the inference network q_θ (encoder) and the generative network p_ϕ (decoder) are single-layer LSTMs. Same as standard VAE setting [24], Gaussin prior $N(\mathbf{0},\mathbf{I})$ acts as a constrain on the hidden variable $\mathbf{z}$. During the encoding stage, the encoder network maps the sequence to a latent variable $\mathbf{z}$. Notice that in order to backpropagate the error during the training process, reparameterization trick [24] is adopted where the mean and variance of hidden state distribution are parameterized as $\mu(z)$ and $\sigma(z)$. During the decoding stage, the decoder serves as a special RNN language model which is conditional on the hidden variable. In the degenerate setting where the distribution of z equals to the Gaussian prior and zero KL-Divergence is acquired, the hidden variable encodes no useful information. Then the VAE language model essentially equals to an RNN language model.

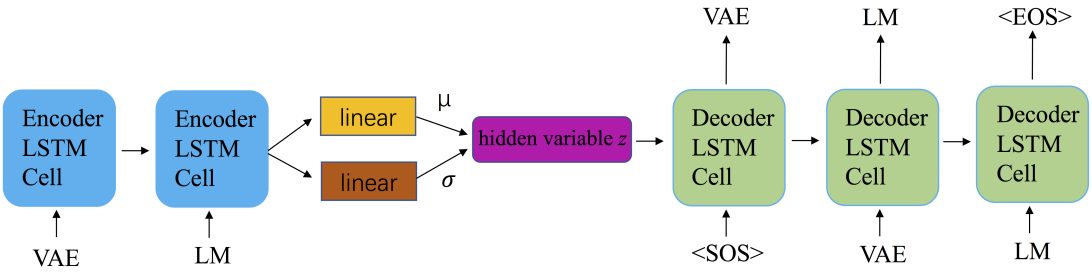


Fig. 2.9 The abstract architecture of the variational autoencoder based language model, and words are represented by feature vector.

When training VAE language model, we deem $ELBO_i(\lambda)$ in Equation (2.13) as negative of loss objective. As a result, the loss function is shown in Equation (2.17), where the first term is reconstruction cost and second term is KL-Divergence cost.

$$loss = -\mathbb{E}_{q_\theta(z|x_i)}[logp_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z)) \tag{2.17}$$

The reconstruction cost is the data likelihood under the variational posterior (expressed as cross-entropy) [4] and KL-Divergence cost of variational posterior can be calculated in closed-form given the Gaussian prior. To calculate the data likelihood, at each time-step t, the output of decoder LSTM cell $\mathbf{o}_t \in \mathbb{R}^M$ is passed to the softmax classifier with the weight matrix $\mathbf{W}_{softmax} \in \mathbb{R}^{M\times V}$, where M is the dimension of output size and V is the output vocabulary size. Then the probability distribution over all possible words at this time-step is inferred and based on this distribution the cross-entropy cost can be calculated.

Optimization Challenges

An inference network which encodes useful information using the latent variable **z** will have nonzero KL-Divergence term and a relatively small cross-entropy term. But when modelling a text sequence, straightforward implementation fails to learn this behaviour. Instead, it tends to learn a model that always sets $q_\theta(z|x)$ equal to prior $p(z)$, resulting in a zero cost for the KL-Divergence term. To address this, a KL cost annealing technique is introduced by Bowman et al. [4] which sets a variable weight on the KL-Divergence term of the cost function as shown in Equation (2.18). The weight term is set to a small value at the start of training, and it is gradually increased to one as the training process continues.

$$loss = -\mathbb{E}_{q_\theta(z|x_i)}[log p_\phi(x_i|z)] + \beta \times KL(q_\theta(z|x_i)||p(z)) \tag{2.18}$$

Therefore, training VAE language model using KL cost annealing forces the inference network to encode more useful information. As a result, the VAE based language model can produce comparable performance to the RNN language model but can encode variational information instead of just remembering word sequences like RNN language model [4].

2.3.3 Model Composition

In previous subsections, we described basic components of the proposed model framework. In this subsection, we introduce how to combine these two parts together to form our final configuration.

Suppose that we have the input sentence *"the <e1> car </e1> has an <e2> engine </e2>"*, the illustration of the proposed composition model is presented in Figure 2.10.

As shown in Figure 2.10, the entire framework combines the discriminative classifier and the generative regularizer together. There are two pipelines in the model:

- In the first pipeline, the input sequence is fed into the ABAGM model to produce the hidden representation of the sequence $H_o \in \mathbb{R}^D$ where D is the dimension of the hidden representation. The hidden representation H_o is further propagated into the classifier which has the weight matrix $\mathbf{W_{class}} \in \mathbb{R}^{D \times N}$ where N is the number of classes. Based on the scores calculated from classifier, the relation prediction can be made.

- In the second pipeline, the input sequence is fed into the VAE language model. For the inference network, the original input sequence is used. As for the generative network, two special symbols <SOS> and <EOS> are used to indicate the start and the end of input sentence.

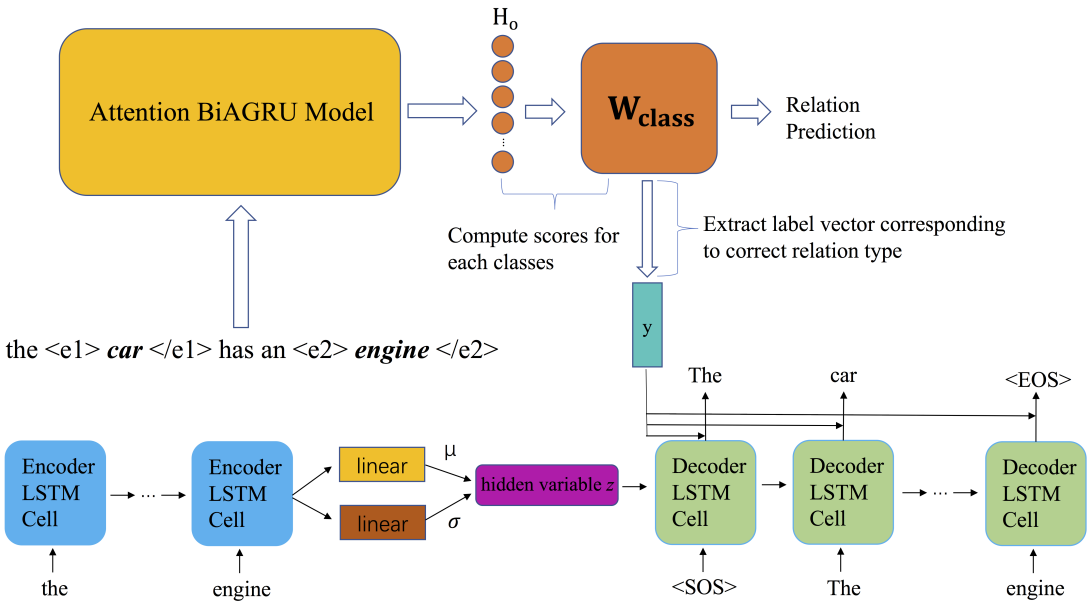


Fig. 2.10 Graphical illustration of compostion model

To connect these two pipelines together, at training time, each sentence s that belongs to a given class c, we extract the c^{th} column $\mathbf{y}$ of weight matrix $\mathbf{W_{class}}$ as the compact representation of the class c. The reason is that the class scores are always measured by the closeness of the sentence hidden representation H_o and each column of $\mathbf{W_{class}}$. So that the columns of $\mathbf{W_{class}}$ are compact vector representation of different classes [54]. The extracted label vector of the corresponding correct class type is further passed into the decoder of the VAE language model. At each time-step t, the label vector $\mathbf{y} \in \mathbb{R}^D$ is concatenated with the output of the decoder LSTM cell $\mathbf{o}_t \in \mathbb{R}^M$ and is fed into the softmax classifier with the weight matrix $\mathbf{W}_{softmax} \in \mathbb{R}^{(D+M)\times V}$ to produce a probability distribution of next word. Then the VAE language model can be trained using cross-entropy loss function.

After combining these two parts, the composition model can be trained jointly. We incorporate the VAE language model into the entire framework so it can act as a regularizer for the classifier. Because the generative network is conditioned on both the hidden variable $\mathbf{z}$ and the sentence label $\mathbf{y}$, the ELBO term in Equation (2.13) is modified to form Equation (2.19) where y_i is the class label for input x_i.

$$ELBO_i(\theta,\phi) = \mathbb{E}_{q_\theta(z|x_i)}[log p_\phi(x_i|z,y_i)] - KL(q_\theta(z|x_i)||p(z)) \tag{2.19}$$

In this case, the VAE language model forces the class matrix $\mathbf{W_{class}}$ to encode more information such that the decoder can better reconstruct the original input sentence. Apart

from regularizing the class matrix, we also include regularization for the input feature. In NLP tasks, the input text is always converted to compact feature representation by mapping from word embeddings. The detailed discussion is presented in Chapter 4. When training end-to-end model, the embeddings of words are always updated accordingly during the training process. Thus by introducing the VAE language model into the framework, we let the model update the word embeddings both for classification task and for language modelling task which is the second regularization imposed on input feature. In conclusion, the regularization from VAE language model for the classifier mainly contains two aspects:

- Forcing the class label matrix $\mathbf{W_{class}}$ to encode more information to help the VAE language model better reconstruct the original sentence.
- Updating the word embeddings during the training process for both the classification task along with the language modelling purpose.

2.4 Training Objective

In this subsection, we describe the objective function used for training the composition model. The model mainly contains three components and they are introduced in the following parts.

2.4.1 Cross-Entropy Loss for Relation Classification

First we use $\mathbf{W}_c$ to denote the class matrix $\mathbf{W_{class}}$ mentioned earlier. For a softmax classifier, given a hidden representation H_{so} of some input sequence s, the probability of the sequence s belonging to class k is calculated by Equation (2.20). The cross-entropy loss is further defined as Equation (2.21). By adopting this loss function, for each input, the model tends to maximize the probability of the correct label while minimize the probability of the incorrect ones.

$$P(y_k|s) = \frac{exp\{\mathbf{W}_{ck}^T H_{so} + b_k\}}{\sum_{i=1}^{K} exp\{\mathbf{W}_{ci}^T H_{so} + b_i\}} \tag{2.20}$$

$$L_{ce} = -\sum_i y_i log(P(y_i|s)) \tag{2.21}$$

2.4.2 Rank Loss

The pairwise rank loss for relation classification is proposed in [54] which is shown in Equation (2.22).

$$L = log(1 + exp(\gamma(m^+ - s_\theta(x)_{y^+}))) + log(1 + exp(\gamma(m^- + s_\theta(x)_{c^-}))) \tag{2.22}$$

In Equation (2.22), m^+ and m^- are margins and γ is scaling factor that magnifies the difference between the scores and the margin helps to penalize the prediction errors; the terms $s_\theta(x)_{y^+}$ and $s_\theta(x)_{c^-}$ stand for scores of the correct label class and the negative label class c^-, in [54] the class c^- is chosen as the incorrect class with largest score ($c^- = \text{argmax}_{c \in C; c \neq y^+} s_\theta(x)_c$). It can be seen that the first term in Equation (2.22) decreases as the correct label score $s_\theta(x)_{y^+}$ increases and the second term decreases as the incorrect label score c^- decreases.

$$s_\theta(x)_k = H_{so}^T [\mathbf{W}_c]_k \tag{2.23}$$

For the input sequence s, the scores are calculated using the class matrix $\mathbf{W}_c$ without the bias term shown in Equation (2.23). Santos et al. [54] propose a special treatment for artificial class *"Other"* where the label vector of *"Other"* is not learned, resulting in the class matrix $\mathbf{W}_c \in \mathbb{R}^{D \times (N-1)}$. At training stage, if one sentence does not belong to any of the actual classes, put it in another word it belongs to artificial class *"Other"*, the first term of Equation (2.22) is set to zero. At prediction stage, if all scores for actual classes are negative, then the predicted class will be the remaining artificial class *"Other"*.

For our case, we modify the original rank loss formula. First, in order to combine cross-entropy loss and rank loss, we do not impose special treatment on the artificial class, in which case, the class matrix $\mathbf{W}_c \in \mathbb{R}^{D \times N}$. Second, in our experiments, we found that only sampling one negative class is too unstable, thus instead of sampling negative class with largest score, we sample k negative classes with top k largest scores. We found that choosing k as 5 for Semeval-2010 task 8 dataset and KBP37 dataset is a suitable setting, and for CID task k is set to 1. As a result, the modified version of rank loss is shown in Equation (2.24), where C contains k negative class labels with top k negative class scores.

$$L_{rank} = log(1 + exp(\gamma(m^+ - s_\theta(x)_{y^+}))) + \sum_{i \in C} log(1 + exp(\gamma(m^- + s_\theta(x)_i))) \tag{2.24}$$

2.4.3 Language Model Loss

The third main component of entire loss function is acquired from the VAE language model which acts as a regularizer for the classifier. Since KL-annealing is adopted (subsection 2.3.2), the VAE loss for one input data x_i can be calculated as Equation (2.25). Notice that the Gaussian prior ($N(\mathbf{0}, \mathbf{I})$) is used over hidden variable, we assume $x_i = \{w_1, ..., w_L\}$ has length L and the dimension of hidden variable $\mathbf{z}$ is D. And KL-Divergence cost weight β is gradully increased to one as training proceeds, where the rate of increase is a hyperparameter which can be tuned on the validation dataset.

$$\begin{aligned} Lvae &= -\mathbb{E}_{q_\theta(z|x_i)}[log p_\phi(x_i|z)] + \beta \times KL(q_\theta(z|x_i)||p(z)) \\ &= -log\prod_{i=1}^{L} P(w_i|z) - \beta \times \frac{1}{2}\sum_{j=1}^{D}(1 + log(\sigma(z)_j^2) - \mu(z)_j^2 - \sigma(z)_j^2) \\ &= -\sum_{i=1}^{L} logP(w_i|z) - \beta \times \frac{1}{2}\sum_{j=1}^{D}(1 + log(\sigma(z)_j^2) - \mu(z)_j^2 - \sigma(z)_j^2) \end{aligned} \tag{2.25}$$

2.4.4 Final Composition Loss Function

The final composition loss function is built from all three components discussed earlier along with L2-Norm loss. The loss function over entire dataset which contains N samples is shown in Equation (2.26).

$$L_{final} = \frac{1}{N}\sum_{n=1}^{N}(L_{ce}^n + L_{rank}^n + \alpha \times L_{vae}^n + \gamma \times \|\lambda\|^2) \tag{2.26}$$

In Equation (2.26), λ stands for all the parameters contained in the composition model. We penalize the parameters to prevent large values, we also place equal importance on cross-entropy term and rank loss term since both terms are directly related to classification performance. We downgrade the importance of VAE language model loss and L2-Norm loss, since they act as a regularizer and do not hold equal importance as the previous two terms.

2.5 Tuning Hyperparameters using Bayesian Optimization

Neural network models are very susceptible to many factors like parameter initialization and hyperparameter setting [21], and different hyperparameter settings always have larger effect than different parameter initialization. To choose a relatively good set of hyperparameters, grid search method is usually used with cross validation or on held-out validation set. Grid search has its advantage that it is very easy to implement, but it also has obvious disadvantages like taking too much time to search all points on the grid and the search point value may not be optimal at the beginning. Therefore we use Bayesian Optimization (BO) to tune hyperparameters instead of using traditional grid search.

We treat the model as a complex function f, and our goal is to solve $x^\star = \arg\max_{x\in\mathbb{X}} f(x)$ where $f(x)$ stands for model performance on specific task. We make nonparametric treatment on function f and approximate f using a proxy function $\tilde{f}$ which we assume to be drawn from a Gaussian Process (GP) prior, i.e. $\tilde{f} \in GP(\mu, k)$ where we take mean function $\mu = 0$. The

choice of kernel has an impact on how well the proxy function represents the real objective function [57] and careful consideration must be taken when choosing kernel k. We follow the idea proposed in [57] to choose the ARD Matern $5/2$ kernel shown in Equation (2.27) as k.

$$K_{M52}(\mathbf{x},\mathbf{x}') = \theta_0(1+\sqrt{5r^2(\mathbf{x},\mathbf{x}')}+\frac{5}{3}r^2(\mathbf{x},\mathbf{x}'))exp\{-\sqrt{5r^2(\mathbf{x},\mathbf{x}')}\} \tag{2.27}$$

Suppose that f has been evaluated n times yielding $D = \{\mathbf{x}_n, y_n\}$, where $y_n \in N(f(\mathbf{x}_n), \sigma^2_{noise})$. The prior assumed on $\tilde{f}$ and D induce a posterior over functions, $\tilde{f}|D \in GP(\mu', k')$.

To choose where to evaluate f next, we rely on an acquisition function which depends on the previous observations D and on the GP hyperparameters θ. We denote the acquisition function as $a(\mathbf{x}|D,\theta)$ which computes salience scores for different choices of $\mathbf{x}$ to evaluate next. There are different acquisition functions which can be used [57] i.e. Probability of Improvement (Equation (2.28)), Expected Improvement (Equation (2.29)), and GP Upper Confidence Bound (Equation (2.30)).

$$\begin{aligned} &\alpha_{PI}(\mathbf{x};\mathbf{x}_n,y_n,\theta) = \Phi(\gamma(\mathbf{x})) \\ &\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{best}) - \mu(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta)}{\sigma(\mathbf{x}_n;\{\mathbf{x}_n,y_n\},\theta)} \end{aligned} \tag{2.28}$$

$$\alpha_{EI}(\mathbf{x};\{x_n,y_n\},\theta) = \sigma(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta)(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + N(\gamma(\mathbf{x});0,1)) \tag{2.29}$$

$$\alpha_{UCB}(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta) = \mu(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta) - k\gamma(\mathbf{x};\{\mathbf{x}_n,y_n\},\theta) \tag{2.30}$$

In our work, we use Expected Improvement (Equation (2.29)) as our acquisition function for it can take balance between exploration and exploitation. Specifically, when tuning hyperparameters, each set of hyperparameter denotes one sample of $\mathbf{x}$ and the corresponding model performance is y. For each experiment, if no official validation set exists, we then split the training set into two parts and use one as validation. By using Bayesion Optimization to tune hyperparameters on validation set, we aim at finding optimal setting of hyperparameters with less effort comparing with grid search approach.

Chapter 3

Relation Extraction Methodology

In previous chapter, we describe our methodology for the relation classification task. In this chapter, we move one step further to solve the relation extraction task in general. Relation classification is usually considered as a subtask of relation extraction which is defined as: given a sequence of raw text, the goal is to extract correct triplets that contain two entities and their corresponding relation r which belongs to predefined set of relations R. For instance, given input sentence **The United States President Trump will visit the Apple Inc and CEO Cook will entertain him in person.**, there should be two extracted triplets which are {United States, Country-President, Trump} and {Apple Inc, Company-CEO, Cook}. For these extracted triplets, the two entities are {United States, Trump} and {Apple Inc, Cook}, in which the corresponding relations are *Country-President* and *Company-CEO*.

The main differences between relation classification and relation extraction are:

- In relation classification, the target words are always given and the entity relation is classified given the target words and context. On the other hand, for relation extraction, only the raw text is given and the task is to determine the words of interest (target nominals) as well as the relation linking them.

- In relation classification, only one relation is classified at one time; so that for previous example, if we want to determine the relation between {United States, Trump} and {Apple Inc, Cook}, we have to classify twice and at each time we mark different entities as target nominals. For relation extraction, the goal is to extract all possible triplets at one run, thus it saves us the effort to run multiple classifications for different pair of target entities.

Traditional relation extraction solutions consist of two steps: (1) entity recognition [43], (2) relation classification [52]. In the first step, the system pinpoints the possible pairs of

entities that may share a relation. In the second step, the system classifies every possible pair of entities. This separated framework makes the task more straightforward to deal with and each component can be performed independently. However, one disadvantage is that by separating relation extraction in two steps, it neglects the internal relevance between these two components. Thus, the performance of entity recognition may affect the performance of relation classification and lead to error propagation [31]. To merge these two steps into one pipeline, Zheng et al. [90] proposed to transform relation extraction into sequence tagging problem and achieved state-of-the-art performance on one public dataset.

Inspired by this idea, we make modifications to our previously proposed model to fit sequence tagging task. In this chapter, we demonstrate how to leverage our model to solve relation extraction problem in general. In the first section, we discuss sequence-to-sequence modelling in NLP literature. In the second section, we demonstrate how to modify our proposed model for relation extraction.

3.1 Sequence-to-Sequence Modelling

Sequence-to-sequence modelling is a fundamental concept in NLP literature where many tasks can be seen as sequence-to-sequence prediction; for example, machine translation (MT) [64], part-of-speech (POS) tagging [71], and name entity recognition (NER) [29]. Sequence-to-sequence modelling can be generally formulated as given an input sequence $X_{1:T}$ with length T, the model is tasked to produce an output sequence $Y_{1:L}$ with length L. For different tasks the definition of input and output can differ. For instance, in machine translation, the input sequence is always the reference language sentence and the output sequence is the target language sentence. Since the input and output sequences belong to different languages, the input length T and output length L are not expected to be equal in general; whereas for name entity recognition, the input sequence is raw text and output sequence is the entity types corresponding to each input word; here, we expect the input sequence length to equal the output sequence length.

For the remainder of this chapter, we demonstrate how to transform the relation extraction task as a sequence tagging problem and how to modify our model to fit this task. Finally, at the end of this chapter, different training objectives which can be used in this scheme are described.

3.2 Relation Extraction as Sequence Tagging Task

3.2.1 The Tagging Scheme

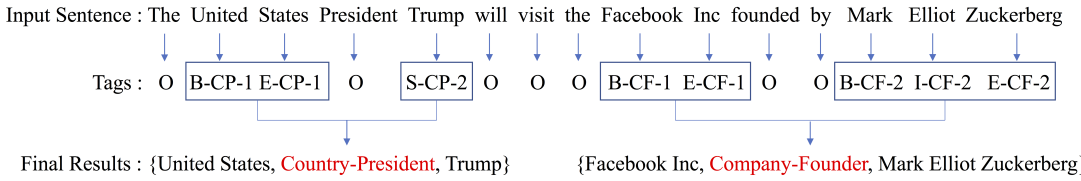


Fig. 3.1 Reference annotation for example sentence, where "CP" stands for "country-president" and "CF" stands for "Company-Founder".

An example of how tagging scheme works is shown in Figure 3.1. Each word of the input sentence is assigned to a label from the predefined label set. Tag "O" stands for the "Other" tag, where words assigned to this label do not contribute to the final extracted result. Apart from the "O" tag, the remaining tags consist of three parts: (1) the word position in the entity, (2) the relation type and (3) the relation role. To represent word position, "BIES" (Begin, Inside, End, Single) signs are used to denote the positional information of a word in the entity. The relation type belongs to a set of predefined relations and the relation role are denoted by number "1" and "2". An extracted result can be represented as a triplet: $(Entity_1, RelationType, Entity_2)$, where "1" means that the word belongs to the first entity in the triplet; likewise, the number "2" means that the word belongs to the second entity in the triplet. Thus, for a predefined relation set with size $|R|$, the total number of possible sequential tag labels is $N_t = 2 \times 4 \times |R| + 1$, where 1 accounts for the "Other" tag.

The example in Figure 3.1 illustrates how to transform the relation extraction task to a sequence tagging task. The first triplet is {United States, Country-President, Trump} in which the first entity is *United States* and the second entity is *Trump*. Note that for the first entity there is no inside word between *United* and *States*, thus the word *United* is mapped to tag *B-CP-1* and *States* is mapped to tag *E-CP-1*. Since there is only one word in the second entity, thus the word *Trump* is mapped to tag *S-CP-2*. The same rule applies to the second triplet {Facebook Inc, Company-Founder, Mark Elliot Zuckerberg}, in which case {Facebook Inc} is mapped to {B-CF-1 E-CF-1} and {Mark Elliot Zuckerberg} is mapped to {B-CF-2 I-CF-2 E-CF-2}.

By using this tagging scheme, the relation extraction problem can be reformulated as a sequence tagging problem instead of being solved in entity recognition and relation classification independently. This allows us to tackle this problem with unified end-to-end model.

3.2.2 Extract Results from Tag Sequence

After getting tag sequence, the triplet results can be formed from entities which have same relation type. For the example in Figure 3.1, *United States* and *Trump* share the same relation type, thus the extracted triplet is formed from these two entities and their corresponding relation. To determine the position of the entities in the extracted triplet, the 'relation role' component of the tag is used. Since *United States* has role "1" and *Trump* has role "2", thus the final extracted result is {United States, Country-President, Trump}.

An output tag sequence may contain several pairs of entities which all can have the same relation type. For example, if the sentence is {The United States President Trump will meet France President Emmanuel Macron}, then the corresponding tag sequence will be {O B-CP-1 E-CP-1 O S-CP-2 O O S-CP-1 O B-CP-2 E-CP-2}. When this situation happens, the triplet is formed from the entities that are closest to each other. So that *United States* forms a triplet with *Trump* and *France* forms a triplet with *Emmanuel Macron*.

In our work, we only consider entities that belong to only one triplet (As is the case with previous work by Zheng et al. [90]). Each entity can only be assigned to exactly one relation type, which means that different relations cannot be allocated to the same entity. The identification of overlapping relations is left for future work.

3.2.3 Att-BiAGRU Sequence Model

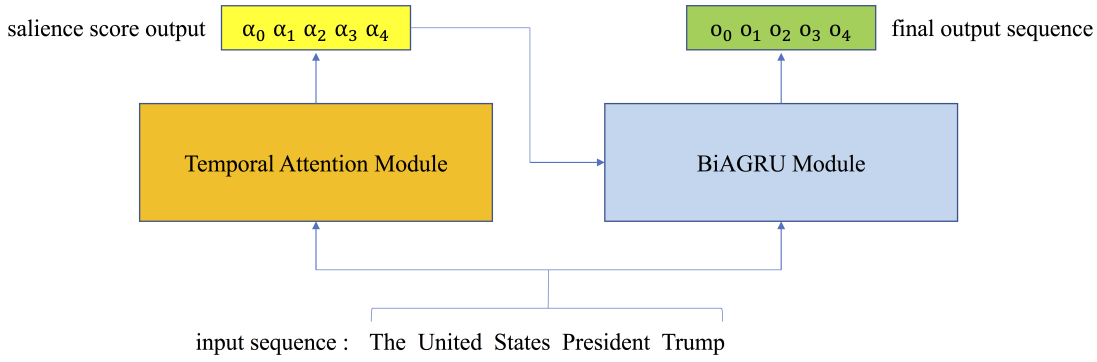


Fig. 3.2 General framework of our proposed Att-BiAGRU sequence model

The general paradigm of our proposed sequence model is shown in Figure 3.2. It can be seen that the sequence tagging model is mainly based on our relation classification model (discussed in Chapter 2), with one modification made. There are still two modules contained in the model (the temporal attention module and the BiAGRU module).

The temporal attention module has the same architecture as the one described in subsection 2.3.1. The input sequence feature $X_{1:T} = x_1, ..., x_T$ is first fed into a bidirectional LSTM layer and based on the bidirectional output the corresponding salience score sequence $A_{1:T} = \alpha_1, ..., \alpha_T$ can be calculated via Equation (2.14). The acquired salience score sequence is further propagated to bidirectional attention-gated recurrent units (AGRU) layer, then a forward hidden state sequence $\{\overrightarrow{H}_1, ..., \overrightarrow{H}_T\}$ and backward hidden state sequence $\{\overleftarrow{H}_1, ..., \overleftarrow{H}_T\}$ are calculated via Equation (2.15). The one modification compared with model in Chapter 2 happens after getting forward and backward hidden state sequences. Since, we are dealing with sequence modelling, we directly concatenate both hidden state sequences together to form the final sequence representation: $O_{1:T} = o_1, ..., o_T$, where at each time-step t the vector o_t is obtained by concatenating the forward state $\overrightarrow{H}_t$ and the backward state $\overleftarrow{H}_t$.

3.2.4 Calculate Sequence Output

For an input sequence $X_{1:T} = x_1, ..., x_T$, we first compute the sequence representation $O_{1:T} = o_1, ..., o_T \in \mathbb{R}^{T \times h}$, where h is hidden size of recurrent unit. Then, the sequence representation is passed through a weight matrix $W \in \mathbb{R}^{h \times N}$ and bias term $b \in \mathbb{R}^{1 \times N}$ to compute the final score matrix $P \in \mathbb{R}^{T \times N}$, where N is number of distinct tags and $P_{i,j}$ corresponds to the score of the j^{th} tag of i^{th} word in the sentence. The next step is to compute final sequence tag prediction from score matrix P, we consider two alternatives (i) using a Softmax, and (ii) using a Conditional Random Field. We discuss both alternatives in the remaining subsections.

Softmax Prediction

The first alternative to predict final tag sequence from score matrix is using softmax function, where at each time-step t, the vector $P_t \in \mathbb{R}^{1 \times N}$ is transformed into valid probability distribution over all distinct tags through Equation (3.1). The term $P(y_i|x_t)$ denotes the probability of word x_t has tag y_i.

$$P(y_i|x_t) = \frac{exp(P_{ti})}{\sum_{j=1}^{N} exp(P_{tj})} \tag{3.1}$$

For each time-step t, the model selects the tag prediction with the largest probability value, the tag sequence with length T can be acquired by repeating this calculation T times.

In the training process, the cross-entropy loss is computed over entire sequence. The formula is described in Equation (3.2), where T is sequence length and N is the distinct tag number.

$$L_{CE} = -\sum_{t=1}^{T}\sum_{j=1}^{N} y_j log(P(y_j|x_t)) \tag{3.2}$$

CRF Prediction

When using previous method to make final prediction, the tagging decisions are made locally. To leverage the intermediate connection between different time-steps, tagging model based on conditional random field (CRF) [28] is proposed in [29]. For an input sequence $X_{1:T} = x_1, ..., x_T$, the score matrix $P \in \mathbb{R}^{T \times N}$ is computed from sequence model. Similarly, T is sequence length and N is number of distinct tags, and $P_{i,j}$ denotes the score of the j^{th} tag of i^{th} word in the sentence. For a sequence of prediction $Y_{1:T} = y_1, ..., y_T$, the prediction score is defined as Equation (3.3), where A is a matrix of transition scores that $A_{i,j}$ denotes the score of transition from tag i to tag j. y_0 and y_n are always the *start* and *end* tags of a sequence which are manually added to the set of possible tags.

$$S(X_{1:T}, Y_{1:T}) = \sum_{i=0}^{T} A_{y_i, y_{i+1}} + \sum_{i=1}^{T} P_{i,y_i} \tag{3.3}$$

The probability of one possible output tag sequence y is computed as Equation (3.4), where Y_X denotes the set of all possible tag sequences for an input sequence $X_{1:T}$.

$$P(y|X_{1:T}) = \frac{exp(S(X_{1:T}, y))}{\sum_{\tilde{y} \in Y_X} exp(S(X_{1:T}, \tilde{y}))} \tag{3.4}$$

During training, the training objective is the negative log-probability of correct tag sequence which is defined in Equation (3.5). It is evident that this objective encourages the network to produce a valid sequence of tag labels.

$$L_{CRF} = -log(P(y|X_{1:T})) = -S(X_{1:T}, y) + log(\sum_{\tilde{y} \in Y_X} exp(S(X_{1:T}, \tilde{y}))) \tag{3.5}$$

While decoding, the output sequence is predicted as described in Equation (3.6), and it can be efficiently computed by dynamic programming, i.e. Viterbi Algorithm [67].

$$y^{\star} = \underset{\tilde{y} \in Y_X}{\operatorname{argmax}} S(X_{1:T}, \tilde{y}) \tag{3.6}$$

3.2.5 Hyperparameters Tuning

For simplicity, we keep all recurrent unit hidden size the same and we add L2-Norm regularization to the model parameters so that when using the softmax prediction the training objective is shown in Equation (3.7), where λ stands for model parameters and β is L2 regularization strength. When we use the CRF prediction, the training objective is defined as

Equation (3.8).

$$L = L_{CE} + \beta \times \|\lambda\|^2 \tag{3.7}$$

$$L = L_{CRF} + \beta \times \|\lambda\|^2 \tag{3.8}$$

As a result, to construct our sequence model, the hyperparameters set contains three parts: (1) recurrent unit hidden size, (2) L2-Norm regularization strength and (3) initial learning rate. All these hyperparameters are automatically tuned via Bayesian Optimization (BO) as described in Chapter 2.

Chapter 4

Relation Classification Experiments

In this chapter, we discuss about various aspects regarding experiments for relation classification task. In order to demonstrate the effectiveness of our work, we apply our proposed method to three relation classification datasets, two from general domain, and one from a specialised domain (biomedical). We detail the experiment setup along with the results for each experiment.

4.1 Experiment Setup

In this section, we discuss several key aspects for the experiment setup. First, we introduce how to construct feature representation from raw input data. Then, we demonstrate the regularization and learning technique for the model training process. Finally, we describe all hyperparameters that are involved in our model.

4.1.1 Word Embeddings Input Layer

In most NLP models and applications, words in raw text are represented as word embeddings [38] which is often a dense vector representation of words. Pretrained word embeddings can be acquired through unsupervised learning on large scale corpus.

Suppose we have a pretrained word embeddings $\mathbf{W}_{embedding} \in \mathbb{R}^{V \times d}$ where d is dimension of the word embeddings and V is the vocabulary size. A given raw text sequence $s = \{w_1, ..., w_L\}$ has the length L. Each word i is first mapped to a one-hot vector $\mathbf{s}_i \in \mathbb{R}^{1 \times V}$, where only one corresponding index has a value of one and all other indexes have value of zero. The one-hot representation of input sequence s of length L is $\mathbf{S} \in \mathbb{R}^{L \times V}$. Following Equation (4.1), the raw text sequence can be mapped into the corresponding compact input

representation $W_{input} \in \mathbb{R}^{L \times d}$.

$$W_{input} = \mathbf{SW}_{embedding} \tag{4.1}$$

One benefit of mapping the raw input to embeddings through input layer is that it allows us to jointly update the input layer weights (the word embeddings) through the training process. Generally, conducting task-oriented optimization on pretrain word embeddings can improve system performance on specific task.

In this work, we use pretrained word embeddings [1] trained on Wikipedia using Glove Algorithm [49], where the dimensionality of the embeddings is 300.

4.1.2 Position Indicator

Because in relation classification task, it is important to let the model know the position of target nominals. For RNN-based model, since it learns knowledge from entire word sequence, therefore the target nominal positions can be automatically recognized in the forward and backward recursive propagation. One simple and effective approach to help the model better recognize target nominals is proposed by Zhang et al. [87]. They propose position indicator technique, in which case, embeddings for four position indicators *<e1>, </e1>, <e2>* and *</e2>* are randomly initialized. And for each word sequence, these indicators are inserted around target nominals. The following is an example:"<e1> **people** </e1> have been moving back into <e2> **downtown** </e2>". In this example, the target nominals are **people** and **downtown**, and mapping this processed sequence through embedding layer can help the model better recognize the target word positions. It can further let the model better calculate the relative importance of the words surrounding the nominals.

4.1.3 Model Training

In this subsection, we detail the training process used in our experiments, as well as steps taken for the model's regularization, and hyperparameter selection.

Model Regularization and Training

Generally, in deep learning application, neural networks model has large number of parameters which give it high capacity to fit complex function but also let it suffer from overfitting problem. To address this, various approaches have been developed such as dropout [61], and L2 regularization. Work by Zaremba et al. [85] has shown that standard dropout does not work on Recurrent Neural Network, and they proposed a modified dropout. In our

[1] The word embeddings can be freely downloaded from https://github.com/chakki-works/chakin

training process, we use the approach proposed by Zaremba et al. [85] for both the Attention BiAGRU Model and the VAE language model. We apply this dropout in the input embedding layer (dropout=0.5). To reduce vanishing gradient or gradient explosion problem, we also adopt the back propagation through time (BPTT) [73] and gradient clipping (threshold is 10.0). To restrict parameter from getting too large, L2-Norm regularization is used on model parameters.

Optimizer and Learning Rate

There is no guarantee to find global optimal parameters while training a neural networks model, thus stochastic gradient descent approach is the standard approach for training. To improve training stability and model performance, various optimization algorithms have been devised. In this work, we use the Adagrad optimizer [12], because this optimizer can adapt the learning rate to model parameters. It performs smaller updates (low learning rates) for parameters associated with frequently occurring features, and larger updates (high learning rates) for parameters associated with infrequent features.

We follow Equation (4.2) to dynamically adjust the learning rate through the training process, where l_{init} is the initial learning rate and T is the training epoch. By doing so, at the start of training, we allow the model to update its parameters at a faster rate, and then update parameters at a slower rate towards the end of the training process. The value of l_{init} is a hyperparameter which is automatically tuned through the Bayesian Optimization process.

$$l_r = \frac{l_{init}}{T} \tag{4.2}$$

Another important parameter in the training process is the mini-batch size during the stochastic training process. In our work, we set batch size as 32.

4.1.4 Model Hyperparameters

In this part, we describe all model hyperparameters which are automatically tuned via Bayesian Optimization. The hyperparameters are listed as follows:

- **Recurrent unit hidden size**: For simplicity, we set same hidden size of all recurrent units contained in the entire framework, which includes temporal attention module unit, attention-gated recurrent unit, encoder and decoder units of the VAE.
- **L2-Norm regularization strength**: Another hyperparameter is the strength of L2-Norm regularization on model parameters which is γ in Equation (2.26). By setting large γ, we penalize more on weights that get too large, and vice versa.

- **Initial learning rate**: As described in subsection 4.1.3, learning rate is hyperparameter we have to tune in order to get robust system performance. So that in Bayesian Optimization process, we also tune the l_{init} shown in Equation (4.2).

- **Vocabulary size**: In the training process of language model, the number of words in the vocabulary can significantly affect the performance. In standard dataset like Penn Treebank [37], the vocabulary size is about 10,000. In our experiment, we found that when vocabulary gets too large (i.e. over 30,000), the VAE language model training process can become very unstable, thus we also set a hyperparameter of vocabulary size in our model framework. Suppose we have vocabulary size k, the words contained in specific dataset are sorted according to frequency and top k words are added to the vocabulary set while all other words are mapped to special token *"_unk_"*.

- **VAE language model loss weight**: In our model, we use the VAE language model as regularizer for the classifier. Since the VAE LM loss does not directly improve the classification performance, thus the regularization strength α (Equation (2.26)) should also be adjusted to achieve optimal performance.

- $\mathbf{m}^+$ and $\mathbf{m}^-$: The last two hyperparameters are m^+ and m^- contained in rank loss function shown in Equation (2.24). In our experiments, we found that fine tuning these two parameters has significant influence on model performance, and bad choice of m^+ and m^- can easily cause the training process diverse and model is therefore collapsed.

4.2 SemEval-2010 Task 8

Recall that relation classification task is about given a sentence and two marked entities, and we want to classify the relation between the marked entities given context. To demonstrate the effectiveness of our model, we perform experiments on two general domain datasets and one biomedical domain dataset.

First, we use the SemEval-2010 Task 8 dataset [18] to perform experiments. This dataset contains 10,717 examples annotated with 9 different relation types and an artificial relation type *"Other"*, which is used to indicate that the relation in the example does not belong to any of the nine main relation types. The nine relations are *Cause-Effect, Component-Whole, Content-Container, Entity-Destination, Entity-Origin, Instrument-Agency, Member-Collection, Message-Topic* and *Product-Producer*. Each example contains a sentence marked with two nominals e1 and e2, and the task is about predicting the relation between the two nominals and taking into account the directionality. That means that the relation Cause-

Effect(e1,e2) is different from the relation Cause-Effect(e2,e1), as shown in the examples below.

- When deliberations began , the <e1> crowd </e1> of people jamming the hallways radiated <e2> optimism </e2>. → Cause-Effect(e1,e2)
- Generally it appears that most of the <e1> damage </e1> was caused by the winds and the rough <e2> seas </e2>. → Cause-Effect(e2,e1)

The statistics of each relation type contained in training and test data of SemEval-2010 Task 8 dataset are shown in Table 4.1 and more information can be found in [18].

	Training Set		**Test Set**	
Relation Type	Number of Samples	Propotion (%)	Number of Samples	Propotion (%)
Cause-Effect(e1,e2)	344	4.30	134	4.93
Cause-Effect(e2,e1)	659	8.24	194	7.14
Component-Whole(e1,e2)	470	5.87	162	5.96
Component-Whole(e2,e1)	471	5.89	150	5.52
Content-Container(e1,e2)	374	4.68	153	5.63
Content-Container(e2,e1)	166	2.08	39	1.44
Entity-Destination(e1,e2)	844	10.55	291	10.71
Entity-Destination(e2,e1)	1	0.01	1	0.04
Entity-Origin(e1,e2)	568	7.10	211	7.76
Entity-Origin(e2,e1)	148	1.85	47	1.73
Instrument-Agency(e1,e2)	97	1.21	22	0.81
Instrument-Agency(e2,e1)	407	5.09	134	4.93
Member-Collection(e1,e2)	78	0.98	32	1.78
Member-Collection(e2,e1)	612	7.65	201	7.40
Message-Topic(e1,e2)	490	6.13	210	7.73
Message-Topic(e2,e1)	144	1.80	51	1.88
Product-Producer(e1,e2)	323	4.04	108	3.97
Product-Producer(e2,e1)	394	4.93	123	4.52
Other	1410	17.63	454	16.71
Total	**8000**	**100**	**2717**	**100**

Table 4.1 SemEval-2010 Task 8 dataset statistic

The SemEval-2010 Task 8 dataset is already partitioned into 8,000 training instances and 2,717 test instances. We evaluate our systems by using the SemEval-2010 Task 8 official scorer, which computes the macro-average F1 Score for the nine actual relations (excluding *"Other"*).

4.2.1 Experiment Results

In this part, we present the experiment results on SemEval-2010 Task 8 dataset. To fully analyze the design of our model, we conduct several experiments based on variations of our

proposed composition model, and we compare our results with the most recent state-of-the-art results in published literature.

Three different variations of our model are listed as below:

- **BiGRU-Attention + BiAGRU Model**: To construct temporal attention module, we use bidirectional recurrent layer with standard recurrent unit. We compare the performance of different recurrent units (GRU and LSTM cells) for constructing the attention module.

- **BiLSTM-Attention + UniAGRU Model**: To examine the effect of the bidirectionality attribute of the AGRU layer, the model with unidirectional AGRU layer on top of BiLSTM attention module is used to conduct experiments, so that we can see the performance difference.

- **BiLSTM-Attention + BiAGRU Model**: For this model, bidiretional LSTM cells are used to construct attention module and bidirectional AGRU layer is also used as described in Chapter 2. The results are compared with previous two variations.

The experiment results are shown in Table 4.2, all figures are F1 Score computed by the official task scorer [18] and {+ VAE LM} means adding VAE language model regularizer on the classifier.

Model Type	Classifier	+ VAE LM
BiAGRU-Attention + BiAGRU	85.7	86.0
BiLSTM-Attention + UniAGRU	85.4	85.6
BiLSTM-Attention + BiAGRU	85.9	**86.3**

Table 4.2 System performance of different model variations

From results in Table 4.2, we can see that the best model setting uses the BiLSTM for the temporal attention module and uses bidirectional ARGU layer to model the input sequence along with computed salience scores. The performance improvement acquired from the VAE language model is significant, the two reasons for this improvement is due to the regularizations provided by the language model (described in subsection 2.3.3).

The optimal hyperparameters for {BiLSTM-Attention + BiAGRU} model acquired from Bayesian Optimisation are shown in Table 4.3, note that we do not tune vocabulary size (21819) in the experiments because the vocabulary set is relatively small for this dataset.

Hyperparameter	Value
Recurrent unit hidden size	126
L2-Norm regularization strength	7.61e-4
Initial learning rate	7.92e-2
VAE language model loss weight	0.729
$\mathbf{m}^+$	2.089
$\mathbf{m}^-$	0.407

Table 4.3 Optimal model hyperparameters values for SemEval-2010 Task 8 dataset

We now compare our model performance with existing state-of-the-art results published in literature. The results are shown in Table 4.4. We can see that our model achieves on a par with state-of-the-art performance without using external knowledge (in paper [89], Stanford POS Tagger toolkit is used to acquire POS features).

4.2.2 Model Analysis

In this subsection, we present analysis of the temporal attention module and VAE language model and their effect on the overall architecture.

We select samples from six different relation types to visualize the salience scores computed from these samples. The results are shown in Figure 4.1, where the color indicates the value of salience score. As the right legend shows, the deeper the color is the larger the salience score is, then we can know which word matters more given different context.

From the results we observe that using the position indicators can help the model pinpoint the location of the two marked nominals. In the last example where two nominals are relatively far apart from each other, the model can still assign relatively high salience score for these two nominals, by doing so, the model improves its prediction compared to a model that does not know the exact nominal locations.

Secondly, we can see that the model is able to automatically learn which words are more important under different relation types. For example, the first instance belongs to relation type Cause-Effect(e1,e2), and the words with highest salience scores apart from target nominals are the phrase *caused a*. This phrase clearly indicates the relationship between two nominals is Cause-Effect(e1,e2) even without the help of other words. We can also see from last instance that if we only look at words with high salience score, the sentence is {*chimps make their rods*} which clearly belongs to the relation type Product-Producer(e2,e1).

Model	F1-score (%)
Non-Neural Networks Model	
SVM [52]	82.2
Shortest Dependency Tree (SDP) Model	
MVRNN [58]	82.4
FCM[83]	83.0
DepNN [33]	83.6
depLCNN+NS[76]	85.6
SDP-LSTM[78]	83.7
DRNNs [77]	86.1
End-To-End Model	
CNN [86]	82.7
ER-CNN + R-RNN [68]	84.2
Attention-based Model	
Attention-CNN [56]	85.9
AT-BiLSTM [89]	86.3
Multi-Att-CNN [70]	88.0*
Our Model	
Att-BiAGRU-VAE	**86.3**

Table 4.4 Result comparison with previous publications ($\star$: We fail to reimplement the result of Multi-Att-CNN, our result is about 84.8. The authors of the paper [89] were also not able to reproduce the result. There are also online reimplementations that can be found, the experiment on the Github https://github.com/FrankWork/acnn is not able to replicate the result as well. We think the reason might be that some details or tricks are missing in the paper.)

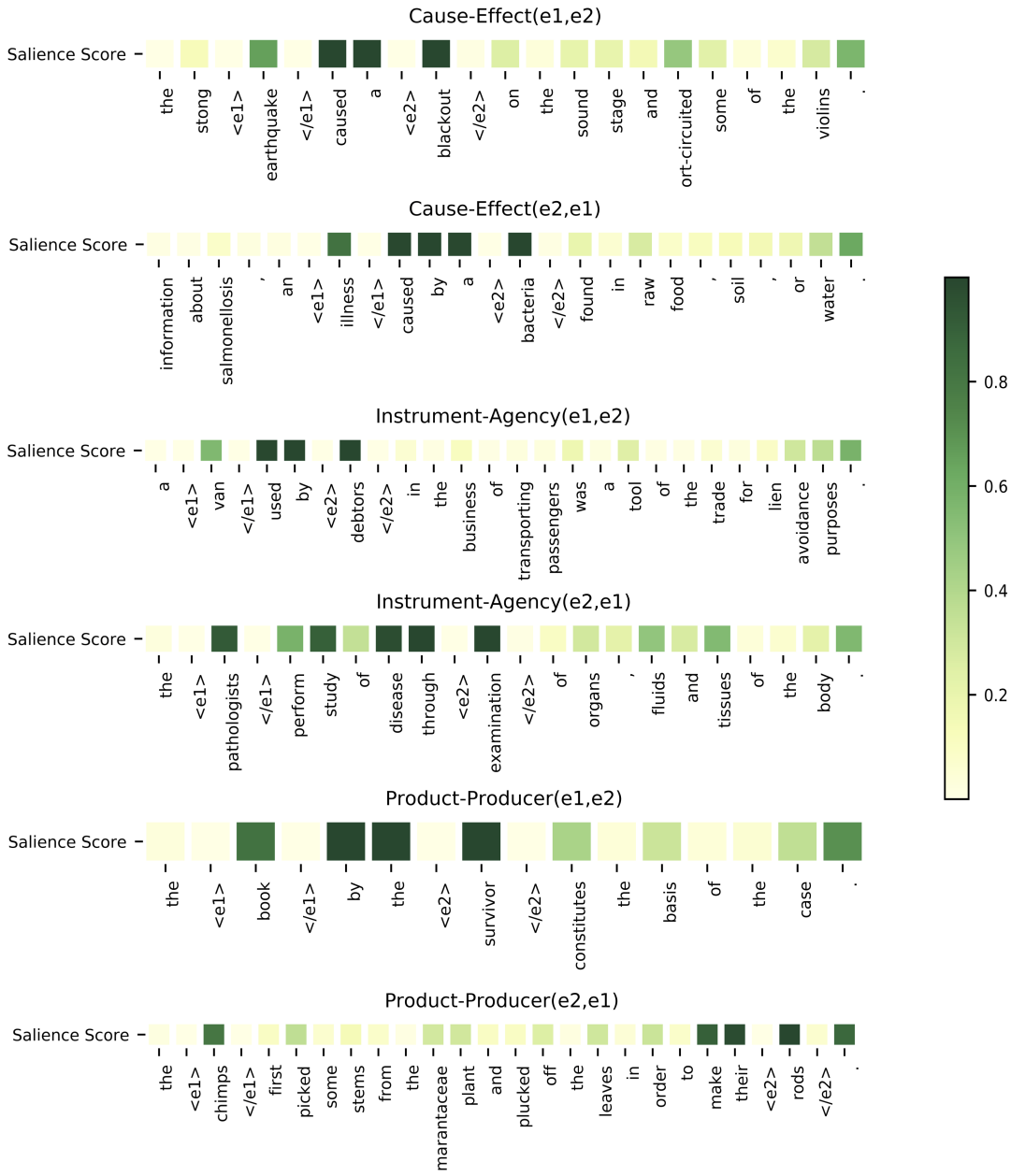


Fig. 4.1 Attention layer visualization

As a result, we can derive following conclusions from attention visualization:

- Using position indicator can help the model pinpoint the exact location of two marked nominals and therefore more accurate decision can be made by the model.
- The attention layer helps the model automatically learn which words are more important given specific relation type, even with long sequence where two nominals are relative far from each other, the model is still able to find important words.

Generating New Sentences

As mentioned earlier, our model consists of one discriminative classifier and one generative regularizer. To conduct classification, the discriminative component can be used; likewise, we can use the generative component to generate text, which is an advantage of generative model [24]. In this part, we use an interesting aspect of our model in which we can generate new instance of any relation type we want.

Recall that the generative network (decoder) in our model takes the form of $p_\phi(x_i|z, y_i)$, thus the generated sample x_i is both conditioned on the latent variable z and relation type y_i. As a result, it is expected that by varying label y_i we can generate sentences of different relations given same z and by varying z we can generate different sentences of same relation given same y_i. To justify our hypothesis, we randomly sample two initial z from Gaussian prior, and for each z we feed it into the generative network of the VAE along with ten different relation labels. For simplicity, to generate new samples, we use greedy decoding approach, in which case, at each time-step we choose the word with highest probability. The generated results are shown in Table 4.5, from which we can see that most of the generated instances are reasonable and belong to the specific type of relation. Note that by training the model with the special position indicators *<e1>, </e1>, <e2>* and *</e2>*, the generated network is able to produce sentence with corresponding position indicators. In which case, we can clearly see the two marked nominals.

From the above analysis, we can conclude that for each instance x_i, the composition model decomposes sentence knowledge into two parts. The first part is the internal topic of sentence which is the relation type y_i. The second part is abstract attributes which are contained in the hidden variable z. The attributes may decide how the instance is formed, i.e. the word choices, sentence length or the style of the sentence. To provide direct illustration, we use T-SNE [36] approach to visualize the hidden representation H_o (Equation (2.16)) and hidden variable z for both training and test data. The results are shown in Figure 4.2 to 4.5.

From above results, we can see that the hidden representation H_o of both training data and test data are clearly separable (the noisy class with colur of bright yellow is the artificial

Relation Type	Sample 1	Sample 2
Cause-Effect(e1,e2)	the <e1> transmitter </e1> emits a <e2> mistake </e2> .	the <e1> earthquake </e1> caused the resulting <e2> signal </e2> .
Cause-Effect(e2,e1)	the <e1> pain </e1> was caused by the <e2> sun </e2> .	the first <e1> accident </e1> was caused by the <e2> sun </e2> .
Instrument-Agency(e1,e2)	the <e1> electrician </e1> is a utensil for a <e2> programmer </e2> .	the best <e1> spikes </e1> are used by <e2> painters </e2> .
Instrument-Agency(e2,e1)	the <e1> defendant </e1> killed the victim with a <e2> spear </e2> .	the relying <e1> shooter </e1> uses a <e2> glove </e2> with a roller and a spear .
Content-Container(e1,e2)	the <e1> bomb </e1> was in a <e2> box </e2> .	the aerosmith <e1> bomb </e1> was in a <e2> box </e2> in the basement of the refrigerator .
Content-Container(e2,e1)	the <e1> stomach </e1> contains the <e2> contents </e2> .	the <e1> bottle </e1> contains the contents of a <e2> cigarettes </e2> .
Message-Topic(e1,e2)	the <e1> book </e1> is a depiction of the <e2> book </e2> .	the first <e1> chapter </e1> was in a <e2> letter </e2> of the first book .
Message-Topic(e2,e1)	the <e1> disparity </e1> has been reflected in the <e2> discussion </e2> .	the harrowing <e1> chapter </e1> has been reflected in a <e2> discussion </e2> of the inspection
Component-Whole(e1,e2)	the <e1> rudders </e1> of the <e2> dryer </e2> is a natinal .	the first primer <e1> system </e1> is the best part of the <e2> building </e2> .
Component-Whole(e2,e1)	the first <e1> train </e1> has a <e2> button </e2> with a single wavelength .	the <e1> kitchen </e1> comprises a <e2> trophy </e2> of the pendulum .

Table 4.5 Generated sentences of different relation types

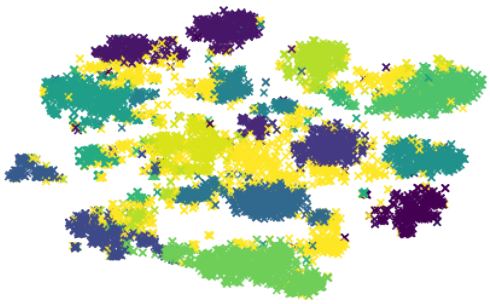


Fig. 4.2 T-SNE visualization of training H_o

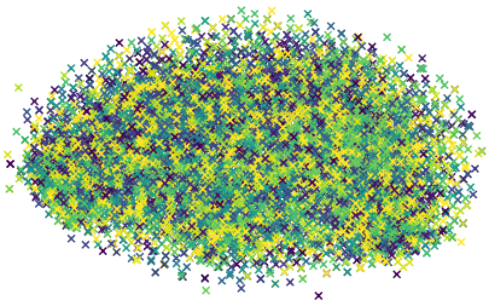


Fig. 4.3 T-SNE visualization of training z

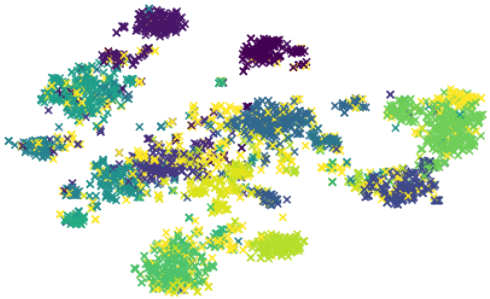


Fig. 4.4 T-SNE visualization of test H_o

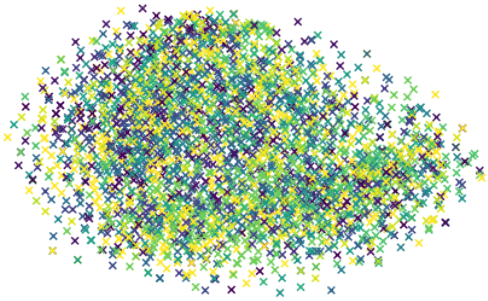


Fig. 4.5 T-SNE visualization of test z

class *"Other"*). On the other hand, the distributions of hidden variable z for both training data and test data across different relation types are very similar. And both of them are similar to predefined Gaussian prior. From above analysis, we can deem the hidden representation H_o as internal topic knowledge which is seperable across different sentences, and the hidden variable z as abstract attributes of sentences which is not separable.

4.3 KBP37

4.3.1 Dataset Demonstration

The second dataset we use is the KBP37 dataset proposed by Zhang et al. [87] which contains 18 *directional* relations and one artificial relation class *"no_relation"*. It is also a dataset from general domain but more difficult than previous SemEval-2010 Task 8 dataset. Several differences between these two datasets are listed as below:

Number of training data	15917
Number of development data	1724
Number of test data	3405
Number of relation types	37
per:alternate_names	org:alternate_names
per:origin	org:subsidiaries
per:spouse	org:top_members/employees
per:title	org:founded
per:employee of	org:founded_by
per:countries_of_residence	org:country_of_headquarters
per:stateorprovinces_of_residence	org:stateorprovince_of_headquarters
per:cities_of_residence	org:city_of_headquarters
per:country_of_birth	org:members
no_relation	

Table 4.6 Statistic of KBP37 dataset

- Pairs of nouns in KBP37 are always entity names which are more sparse than SemEval-2010 task 8. And there are more target nouns that own several words instead of one, which is rarely seen in the Semeval-2010 task 8 dataset.
- Average length of sentences in KBP37 is much longer than Semeval-2010 task 8 dataset.
- It is not guaranteed that there exists only one relation per data, though in test set only one relation is offered as the answer.

The statistic of KBP37 dataset is shown in Table 4.6 and several examples are shown as below:

- <e1> Thom Yorke </e1> of <e2> Radiohead </e2> has included the + for many of his signature distortion sounds using a variety of guitars to achieve various tonal options.→ per:employee_of(e1,e2)
- Hastings was unable to confirm news reports that the victim was the state's <e1> Democratic </e1> Party chairman <e2> Bill Gwatney </e2>. → per:employee_of(e2,e1)

4.3.2 Experiment Results

In this part, we present the experiment results on the KBP37 dataset. The results are also measured by macro-average F1 Score. For simplicity, we choose the best model design acquired from previous section to perform experiments and results are shown in Table 4.7. On comparison with baseline performance we show that our model delivers significantly better performance.

Model Type	F1-score
RNN+PI (baseline [87])	58.8
BiLSTM-Attention + BiAGRU	62.5
BiLSTM-Attention + BiAGRU + VAE LM	**63.4**

Table 4.7 Model performance on KBP37 dataset

The optimal hyperparameters for the model with best performance are shown in Table 4.8, note that we also fine tune the vocabulary size for the VAE languagae model.

Hyperparameter	Value
Recurrent unit hidden size	151
L2-norm regularization strength	8.12e-4
Initial learning rate	0.168
VAE language model loss weight	0.353
$\mathbf{m}^+$	1.841
$\mathbf{m}^-$	0.278
Vocabulary size	12663

Table 4.8 Optimal model hyperparameters values for KBP37 dataset

From the results we can see that adding the VAE language model can further increase the model performance by **0.9** F1 Score. On the other hand, for the SemEval-2010 Task 8 dataset, only **0.4** F1 Score improvement is gained. One explanation for this phenomenon is that the VAE language model is expected to produce more benefit as the number of relation types increases. This is because the more relation types exist, the more discriminative the columns of class matrix $\mathbf{W}_{\mathbf{class}}$ have to be in order to help the VAE language model better

reconstruct the original sentences. As a result, the VAE language model improves more for dataset with 37 relation types than a dataset with 19 relation types.

4.4 Chemical Disease Relation Task

4.4.1 Dataset Demonstration

In the previous two experiments, both datasets are from general domain. In order to test our model performance in a more specialized domain, we apply our model to the biomedical domain dataset. In this experiment, we use BioCreative V chemical disease relation (CDR) task [72] dataset, which aims to evaluate system performance on the task of chemical-disease relation extraction. To better fit the purpose of evaluating our model, we focus on the second subtask (Chemical-induced disease relation extraction) of [72]. The subtask description is summarized as follows:

Given with raw text from PubMed articles as input, the model is asked to return <chemicl, disease> pair which has drug-induced diseases relation.

The model is fed with PubMed articles and the task is to extract entity pairs that contain both a chemical entity and a disease entity. And the extracted chemical entity and disease entity must have chemical-induced disease relation. An example PubMed article is shown as follows:

Lidocaine*-induced* ***cardiac asystole****. Intravenous administration of a single 50-mg bolus of* ***lidocaine*** *in a 67-year-old man resulted in profound depression of the activity of the sinoatrial and atrioventricular nodal pacemakers. The patient had no apparent associated conditions which might have predisposed him to the development of bradyarrhythmias; and, thus, this probably represented a true idiosyncrasy to* ***lidocaine****.*

In the above example, chemical entity is marked as double underlines and disease entity is marked as single underline. Only the entity pair <lidocaine, cardiac asystole> has positive chemical-induced disease (CID) relation and all other possible chemical-disease entity pair combinations have negative CID relation.

4.4.2 Task Reformulation

In order to evaluate the Chemical-induced disease (CID) relation extraction on our model, we first need to restructure the data. Since our model performs classification not extraction, we deem the task as a binary classification task. We consider all possible combinations of chemical entity and disease entity, and for an entity pair which has a positive CID relation, we mark it as positive example; likewise with an entity pair which has a negative CID

relation, we mark it as negative example. We then process the dataset on sentence-level and article-level. For sentence-level, we only consider entity pairs that co-occur in one single sentence. As for article-level, all entity pairs that co-occur in same article are considered and it is very similar as original task. After postprocessing the dataset, we perform experiment on both levels and use F1 Score to measure system performance as [72] did.

4.4.3 Biomedical Word Embeddings

Unlike previous two datasets which are from general domain, chemical disease relation task is from biomedical domain. So that we expect word embeddings pretrained in biomedical corpus can produce better performance than word embeddings pretrained on general domain corpus. To this end, we use pretrained BioMedical word embeddings from [8] to initialize our embedding layer. This word embeddings is pretrained on two BioMedical corpora which are PubMed Central Open Access subset (PMC) and PubMed. The embedding size is 200 instead of 300.

4.4.4 Experiment Results

In the following part, we present the experiment results on both sentence-level and article-level. And on comparison with baseline method, we show that our approach can significantly improve system performance on sentence-level, and on article-level we can still outperform baseline method. The results are shown in Table 4.9.

Model Type	F1-score (%)
Baseline [8]	57.03
Sentence-level	
BiLSTM-Attention + BiAGRU	67.27
BiLSTM-Attention + BiAGRU + VAE LM	67.41
Article-level	
BiLSTM-Attention + BiAGRU	58.31
BiLSTM-Attention + BiAGRU + VAE LM	58.49

Table 4.9 Model performance on CID task

From the results, we can see that the performance on both levels outperform the baseline result, and sentence-level performance is significantly higher than the article-level. Also, the

VAE language model is also able to slightly improve the performance on both levels. More specifically:

- **1.** On sentence-level, the model achieves much better performance than article-level because the average length of input sequence is much shorter. So that it is easier for the model to fully capture the temporal information on the sentence-level which leads to a better performance.
- **2.** On this task, the VAE language model is still able to slightly improve the system performance but not as much as it did in the previous two experiments. Because the relation type number is only two (positive and negative CID relation), and for both cases, the sentences may present similar structure and word choices. As a result, we assume the improvement of VAE language model only comes from the regularization of word embeddings through training process and not from regularization of relation label matrix $\mathbf{W}_{\mathbf{class}}$.
- **3.** By reformulating the task, we are able to use our classification model on the extraction task in biomedical domain. The experiment results show that we can outperform the baseline approach on both levels which further proves that our model is effective in both general and specialized domains.

Chapter 5

Relation Extraction Experiments

In this chapter, we detail the experiment set up and results for our relation extraction model. In order to get a more comprehensive understanding of the model's performance, we conduct experiments on datasets from three different domains: the general domain (newswire), the scientific domain and the biomedical domain.

5.1 Experiment Setup

We follow the same experiment setup as described in Chapter 4. In summary, to get an input feature representation, the raw text is first mapped through a word embeddings layer that is initialized by pretrained domain-specific embeddings. Since in relation extraction, there is no marked entities, thus the position indicator technique is not used to preprocess the data. To update model parameters, we use the same update rule for learning rate as shown in Equation (4.2), and the optimizer is also chosen as Adagrad optimizer [12]. We also use Bayesian Optimization (BO) to fine tune the hyperparameters, and there are only three hyperparameters for the proposed sequence model which are: (1) recurrent unit hidden size, (2) initial learning rate and (3) L2-Norm regularization strength.

5.2 SemEval-2018 Task 7 Subtask 2

5.2.1 Task Demonstration

SemEval-2018 Task 7 [16] is a recently published task concerning semantic relation extraction and classification of text from scientific papers.

To evaluate our relation extraction model, we focus on subtask 2 of the overall shared task, which is about relation extraction and classification. Given an abstract from a scientific

paper with annotated named entities, the two components for this subtask are: (1) identifying instances of semantic relations between the named entities occurring in the same sentence and (2) assigning class labels, i.e. one of six predefined relation types, to the identified relation instances. There are six general predefined relations and five of them are directional and their detailed explanations are presented in Table 5.1. In all of these six general relations, only the COMPARISON relation is symmetric; all other relations are directional. In general, the fine-grained relation type number is $5 \times 2 + 1 = 11$.

Relation Type	**Explanation**
USAGE	Methods, tasks, and data are linked by usage relations
RESULT	An entity affects or yields a result
MODEL	An entity is a analytic characteristic or abstract model of another entity
PART_WHOLE	Entities are in a part-whole relationship
TOPIC	This category relates a scientific work with its topic
COMPARISON	An entity is compared to another entity

Table 5.1 Relation types and corresponding explanations

Recall that the task data is abstract text from scientific papers with annotated entities, in the following we present an example from test set:

This paper introduces a simple <entity id="H94-1014.1"> mixture language model </entity> that attempts to capture <entity id="H94-1014.2"> long distance constraints </entity> in a <entity id="H94-1014.3"> sentence </entity> or <entity id="H94-1014.4"> paragraph </entity>. The model is an <entity id="H94-1014.5"> m-component mixture </entity> of <entity id="H94-1014.6"> digram models </entity>. The models were constructed using a <entity id="H94-1014.7"> SK vocabulary </entity> and trained using a 76 million word <entity id="H94-1014.8"> Wall Street Journal text corpus </entity>. Using the <entity id="H94-1014.9"> BU recognition system </entity>, experiments show a 7% improvement in <entity id="H94-1014.10"> recognition accuracy </entity> with the <entity id="H94-1014.11"> mixture digram models </entity> as compared to using a <entity id="H94-1014.12"> Digram model </entity>.

From above example, we can see that all entities and their corresponding entity IDs are provided and we will discuss how to leverage this information in following subsections.

5.2.2 Data Preparation

Domain Specific Word Embedding

Since we are dealing with scientific domain dataset, thus we use domain-specific word embeddings to initialize embedding layer instead of using word embedding pretrained on general domain. In our experiment we use embedding pretrained by Lauscher et al. [30]. The embedding[1] is 50-dimensional trained by Glove [49] algorithm on the CORE corpus of scientific publications aggregated from Open Access repositories and journals [25].

Data Augmentation

Because the SemEval-2018 Task 7 dataset provides limited training data, we enlarge the training data by combining both training data from subtask 2 and subtask 1.2 [16]. The statistic of relation instances appear in both training and test sets are shown in Table 5.2.

	Training Set		**Test Set**	
Relation Type	Number of Samples	Propotion (%)	Number of Samples	Propotion (%)
USAGE	619	25.00	111	30.25
USAGE_REVERSE	334	13.49	63	17.17
MODEL-FEATURE	349	14.10	57	15.53
MODEL-FEATURE_REVERSE	152	6.14	16	4.36
PART_WHOLE	275	11.10	60	16.35
PART_WHOLE_REVERSE	155	6.26	22	5.99
RESULT	137	5.53	14	3.81
RESULT_REVERSE	58	2.34	2	0.54
TOPIC	238	9.61	3	0.82
TOPIC_REVERSE	23	0.93	0	0.00
COMPARE	136	5.92	19	5.18
Total	**2476**	**100**	**367**	**100**

Table 5.2 SemEval-2018 Task 7 Subtask 2 Statistic

Leverage Entity ID Information

As presented in above example, all entities contained in the abstract are provided with corresponding IDs, i.e. the entity *long distance constraints* has its ID *H94-1014.2*. Suppose this entity belongs to relation PART_WHOLE and has relation role "1", then the tag reference according to subsection 3.2.1 should be *{B-PART_WHOLE-1 I-PART_WHOLE-1 E-PART_WHOLE-1}*. But we can use entity ID to represent the entire entity text to further simplify the tag reference. By doing so, the entity ID *H94-1014.2* itself is used in the raw text

[1]The pretrained embedding can be freely downloaded via this link: https://github.com/anlausch/scientific-domain-embeddings

as input and its corresponding tag is *{S-PART_WHOLE-1}*. And to calculate the embedding of this entity ID, we use the average embedding of these three words *long*, *distance*, and *constraints*. The reason we propose this simplification is that in this dataset only the entities which have their entity IDs are considered for relation extraction, so that all non-entity words will be mapped to tag "O". And using entity ID to replace entity text can help us reduce the size of possible tag set. Now for each relation, there are only two possible tags, i.e. for relation *PART_WHOLE*, the possible tags are *{S-PART_WHOLE-1}* and *{S-PART_WHOLE-2}*. As a result, the size of possible tags N_t are now $2 \times |R| + 1$ instead of $2 \times 4 \times |R| + 1$, where $|R|$ is relation set size and 1 is for "O" tag. Therefore, the model parameters can be reduced since prediction class number is reduced.

5.2.3 Result Evaluation

Recall that the goal of this subtask consists of two parts, which are:

- identifying instances of semantic relations between entities in the same sentence
- assigning class labels, i.e. one of six predefined relation types (see Table 5.1), to the relation instances

For the first part, the aim is to extract entity pairs that have internal relation, and the relation label and directionality are not considered at this step. In the second part, the aim is to make relation classification on extracted entity pairs from previous step. The detailed evaluation metrics for each step are listed as below:

- **Evaluation of relation extraction**: Extraction evaluation assesses the quality of identified relation instances. Relation labels and directionality are ignored in this step. *Precision* is calculated as the percentage of correctly connected entity pairs. *Recall* is calculated as the percentage of gold entity pairs found by the system. The official F1 score is calculated as the harmonic mean of precision and recall.
- **Evaluation of relation classification**: Classification evaluation considers only correctly identified relation instances as per step 1. For these instances, the official score evaluation metrics is **macro-average F1**.

From above definitions we can see that the first evaluation considers how many correct entity pairs are extracted and the second evaluation considers how well the system assigns correct relations to these correct extracted entity pairs.

5.2.4 Experiment Results

To get a more comprehensive understanding on our model performance, we choose three variations of our proposed model in subsection 3.2.3 to conduct experiments. The variations are described in the following:

- **BiLSTM**: Only the bidirectional LSTM layer is used to directly model the input sequence, and no attention layer or AGRU layer is used.
- **BiLSTM-Attention + UniAGRU**: Bidirectional LSTM layer is used to construct temporal attention module and unidirectional AGRU layer is used on top of BiLSTM-Attention module to perform sequence modelling.
- **BiLSTM-Attention + BiAGRU**: The exact model architecture described in subsection 3.2.3. Bidirectional LSTM layer is used to construct temporal attention module and bidirectional AGRU layer is used on top of BiLSTM-Attention module to perform sequence modelling

The experiment results on two separate steps of subtask 2 along with the state-of-the-art results on published literature are shown in Table 5.3, and both CRF and Softmax are used for each variation.

	Extraction (**F1 Score**) (%)	Extraction + Classification (**F1 Score**) (%)
UWNLP [34]	**50.0**	39.1
ETH-DS3Lab [53]	48.8	49.3
SIRIUS-LTG-UiO [46]	37.4	33.6
UC3M-NII [62]	35.4	18.5
NTNU [3]	33.9	17.0
Bf3R [44]	33.4	20.3
Our Experiment		
BiLSTM	38.3	73.8
BiLSTM + CRF	37.9	74.5
BiLSTM-Attention + UniAGRU	25.9	55.8
BiLSTM-Attention + UniAGRU + CRF	27.1	60.1
BiLSTM-Attention + BiAGRU	40.3	**92.5**
BiLSTM-Attention + BiAGRU + CRF	39.9	88.4

Table 5.3 Experiment Results on SemEval-2018 Task 7 Subtask 2

5.2.5 Result Analysis

From results in Table 5.3, it can be seen that our models do not outperform state-of-the-art systems on extraction task. Which means that these state-of-the-art systems can extract more

correct entity pairs which have potential internal relation. On the other hand, for extraction + classification task, all of our system variations deliver much better performance than current state-of-the-art systems. For our systems, the best model variation is **BiLSTM-Attention + BiAGRU**, and it performs 40.3 F1 Score on extraction task and 92.5 F1 Score on extraction + classification task. Comparing with other published results, although our system takes a 10% hit on extraction step, the high performance on classification step outperforms the published best performance by more than 40% F1 Score. This means that our model assigns correct relation to almost all of the correct extracted entity pairs, and by combining these two steps together we can say that our model gives overall state-of-the-art performance on SemEval-2018 Task 7 Subtask 2.

Then we give analysis about different variations of our proposed model. Firstly, from all of the aforementioned configurations, we see that the CRF layer and the Softmax layer deliver similar performance on both steps. Generally, the Softmax layer performs better, and the best model setting **BiLSTM-Attention + BiAGRU** is constructed with the Softmax layer. Secondly, comparing the **BiLSTM** with the **BiLSTM-Attention + UniAGRU** configuration, we can note that the **BiLSTM** achieves better performance. We conclude that the bidirectionality attribute of our model matters more than the attention attribute when solving relation extraction as a sequence tagging problem; by leveraging both bidirectionality and attention attributes, the model can generally produce best performance.

5.3 New York Times

5.3.1 Dataset Demonstration

For the second experiment, we conduct relation extraction on New York Times[2] (NYT) dataset [51]. The training set contains large amount of data (353k triplets) which is acquired by means of distant supervision without manually labelling. As for test set, to precisely evaluate model performance, it is manually labelled to ensure its quality (3,880 triplets). Since our model does not deal with overlapping relations (described in subsection 3.2.2), we preprocess the dataset to make it suitable for our model. The total number of possible relations is 25 (24 actual relations and one "None" relation). Because there is no entity ID information provided, we cannot conduct the same simplification as we did in subsection 5.2.2. As a result, the total number of possible tags is $2 \times 4 \times 24 + 1 = 193$, where 1 is for the "None" relation.

[2]The dataset can freely be downloaded at: https://github.com/shanzhenren/CoType

5.3.2 Data Preparation

In this experiment, we use same general domain embeddings that was used for the SemEval-2010 Task 8 and the KBP37 task in Chapter 4. The embedding is pretrained on Wikipedia with Glove [49] algorithm with embedding dimensionality of 300.

To tune model hyperparameters, 10% data of test set are randomly selected to form validation set. Bayesian Optimization (BO) is used on validation set to evaluate model performance and automatically tune hyperparameters.

5.3.3 Result Evaluation

To evaluate system performance, standard Precision, Recall and F1 Score are used. Same as [90], an extracted triplet is correct when its relation type and the head offsets of two corresponding entities are both correct.

5.3.4 Experiment Results

We use the best model configuration acquired previously, which is **BiLSTM-Attention + BiAGRU**, to conduct this experiment. Both the Softmax and CRF layers are also used to compare the models' performance. The experiment results and published state-of-the-art results are shown in Table 5.4.

	Precision (%)	Recall (%)	F1 Score (%)
FCM [14]	55.3	15.4	24.0
DS + logistic [40]	25.8	39.3	31.1
LINE [88]	33.5	32.9	33.2
MultiR [20]	33.8	32.7	33.3
DS-Joint [31]	57.4	25.6	35.4
CoType [51]	42.3	51.1	46.3
LSTM-CRF [90]	**69.3**	31.0	42.8
LSTM-LSTM [90]	68.2	32.0	43.6
LSTM-LSTM-Bias [90]	61.5	41.4	49.5
Our Experiment			
BiLSTM-Attention + BiAGRU	58.77	**49.32**	**53.63**
BiLSTM-Attention + BiAGRU + CRF	56.11	48.77	52.19

Table 5.4 Experiment results on NYT dataset

5.3.5 Result Analysis

In the above experiment results, we can see that our sequence model achieves state-of-the-art performance on the NYT dataset. This is done without constructing a sophisticated loss function. Our model is able to deliver around 50% recall while still maintaining a good level of precision. As a result, our model achieves the best F1 Score on this task. This result further suggests that our proposed model is a good sequence model for relation extraction.

We can also observe that the Softmax layer outperforms CRF layer. The reason from our analysis is that the possible tag number for this task is quite large (193), which makes it difficult for CRF layer to learn a good transition matrix $A \in \mathbb{R}^{193\times193}$. As a result, the CRF layer does not produce better results in contrast with the Softmax layer.

5.4 Chemical Disease Relation Task

5.4.1 Dataset Demonstration

For the third experiment, we utilize a dataset from the biomedical domain. We use the CDR task dataset [72] that was also used previously in section 4.4 for relation classification; however, this time, we perform direct relation extraction on this task.

5.4.2 Data Preparation

We use the biomedical domain specific word embeddings as we did in subsection 4.4.3. The embeddings are pretrained on PubMed with embedding dimensionality of 200.

The chemical and disease entity ID are provided for this task, we therefore leverage this entity ID information as we did in subsection 5.2.2, in which case we replace the entity text contained in raw data with its corresponding entity ID. Since only one actual relation **Chemical-Induced Disease** (CID) is considered for this task (demonstrated in subsection 4.4.1), the possible tag number is $2 \times 1 + 1$, where 1 stands for the "None" relation.

Since our model mainly focuses on relation extraction on sentence level, we therefore only consider entity pairs that co-occur in same sentence. The training and test data that do not satisfy this condition are ignored to better evaluate our model on sentence level.

5.4.3 Result Evaluation

As with previous experiments, the evaluation metrics are standard Precision, Recall and F1 Score. The extracted triplet is considered as correct when both entity IDs and corresponding relation type are correct.

5.4.4 Experiment Results

For this experiment, we also use the best model configuration acquired previously, which is **BiLSTM-Attention + BiAGRU**. And both Softmax layer and CRF layer are experimented with to compare model performance. The baseline performance [72] and our model performance are shown in Table 5.5.

	Precision (%)	Recall (%)	F1-score (%)
Baseline [8]	55.67	58.44	57.03
Our Experiment on Sentence-level			
BiLSTM-Attention + BiAGRU	53.85	**59.93**	56.73
BiLSTM-Attention + BiAGRU + CRF	**58.92**	57.03	**57.96**

Table 5.5 Experiment Results on CDR dataset

5.4.5 Result Analysis

Firstly, from above results, we see that our model delivers comparable performance on sentence-level relation extraction. One advantage of our model over baseline system [8] is that in our model we do not leverage any external knowledge. The only knowledge we use is domain-specific pretrained word embeddings, so that this result further proves the effectiveness of our model on domain-specific data.

Secondly, for this experiment, we see that the CRF layer performs better than Softmax layer. Our analysis is that the possible distinct tag number for this task is small (3), so that the model is able to learn a good transition matrix $A \in \mathbb{R}^{3\times3}$. As a result, the CRF layer is able to capture global sequence information with a good transition matrix, therefore a better performance is acquired with the CRF layer.

5.5 Conclusion

In this chapter, we perform relation extraction on three datasets from scientific, general and biomedical domains. All results show that our model is able to achieve at least on par results with current state-of-the-art performance on various datasets, in several configurations, our model significantly outperform the state-of-the-art. These results demonstrate that our

proposed model is able to perform well on sequence tagging task, and can be appropriately used when reformulating the task of relation extraction as a sequence tagging task.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Relation classification and relation extraction are fundamental tasks in information extraction with many downstream applications, such as question answering, text mining, textual entailment, and knowledge base construction. Relation classification task is defined as: given an input sequence with two named entities, the goal is to categorize a possible relation between these entities from a set of predefined relations. On the other hand, the relation extraction task can be formulated as: given a sequence of raw text, the goal is to extract correct triplets that contain two entities and a corresponding relation. Thus, we can decompose relation extraction as two steps: named entity recognition (NER) and relation classification. As a result, relation classification can be seen as a subtask of relation extraction.

For relation classification, we proposed an attention-based RNN classifier. To further improve model performance, a VAE-based regularizer is incorporated into the model. For relation extraction, we proposed a joint end-to-end neural model for NER and relation classification . In this model, we reformulated the relation extraction task as a sequence tagging problem, from which the triplets containing entities and corresponding relation can be directly extracted. Then, we construct our sequence model from our attention-based RNN classifier. To further improve system performance, softmax and CRF layers are used for decoding the output tag sequence.

We conduct comprehensive experiments to evaluate our proposed models for both tasks. For relation classification, we perform experiments on datasets from the general domain (SemEval-2010 Task 8, KBP37) and the biomedical domain (CDR). In SemEval-2010 Task 8 dataset, we achieve state-of-the-art performance using our proposed **BiLSTM-Attention + BiAGRU + VAE LM** model without leveraging external knowledge. For this dataset, we do not count the performance of Wang et al. [70] due to difficulty to replicate their stated results

by both of us and other published works, for example [89] and authors for this Github link https://github.com/FrankWork/acnn. We also give a detailed analysis of how the attention module works. To illustrate the generative perspective of our model, we generate various reasonable new samples given different relations. On KBP37 dataset, we achieve significantly better performance comparing with state-of-the-art result by using our **BiLSTM-Attention + BiAGRU + VAE LM** model. The improvement is 4.6% F1 Score. For CDR task, we first reformulate it into binary classification problem, and we conduct experiments on article-level and sentence-level. Results from both levels are better than baseline performance, which proves the usefulness of our model on specific domain dataset.

For relation extraction, we perform experiments on datasets from the general domain (NYT), the scientific domain (SemEval-2018 task 7 subtask 2) and the biomedical domain (CDR). For NYT dataset, by leveraging sequence tagging scheme along with our proposed sequence model, we are able to improve the state-of-the-art performance by 4.13% F1 Score. For the CDR task, we conduct direct relation extraction on sentence-level and we are able to achieve comparable state-of-the-art performance. For SemEval-2018 task 7 subtask 2, this subtask is divided into two steps: named entity recognition and relation classification. Our model performs 40.3 F1 Score for named entity recognition and 92.5 F1 Score for relation classification. Although we do not deliver best performance for the first step (compared to the current state-of-the-art F1 Score of 50.0), we are able to deliver 92.5 F1 Score on relation classification step (compared to the current state-of-the-art F1 Score of 49.3). By combining performances of these two steps together, we achieve overall state-of-the-art performance for the entire task.

6.2 Future Work

We plan to investigate our methods on other tasks in NLP. The classifier built for relation classification can be easily adapted for other sequence classification tasks, such as sentiment analysis, sentence paraphrasing, semantic role labeling (SRL), and coreference resolution.

As for general relation extraction task, VAE-based regularizer can still be incorporated into the model's framework. For example, we can use VAE regularizer to tackle name entity recognition of input sequence, at the same time as the relation extraction is performed. By predicting named entities information jointly, the model may achieve better regularization and produce better performance. The proposed sequence model can also be extended to other information extraction tasks, such as event extraction [79]. In this case, we can conduct name entity recognition and event extraction jointly.

References

[1] Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *Computer Science*.

[2] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *Computer Science*.

[3] Barik, B., Sikdar, U. K., and Gambäck, B. (2018). NTNU at semeval-2018 task 7: Classifier ensembling for semantic relation identification and classification in scientific papers. pages 858–862.

[4] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *Computer Science*.

[5] Bravo, À., González, J. P., Queralt-Rosinach, N., Rautschka, M., and Furlong, L. I. (2015). Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. pages 55:1–55:17.

[6] Bunescu, R. C. and Mooney, R. J. (2005a). A shortest path dependency kernel for relation extraction. pages 724–731.

[7] Bunescu, R. C. and Mooney, R. J. (2005b). A shortest path dependency kernel for relation extraction. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

[8] Chiu, B., Crichton, G., Korhonen, A., and Pyysalo, S. (2016). How to train good word embeddings for biomedical nlp. In *The Workshop on Biomedical Natural Language Processing*, pages 166–174.

[9] Cho, K., Merrienboer, B. V., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *Computer Science*.

[10] Chung, J., Gulcehre, C., Cho, K. H., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *Eprint Arxiv*.

[11] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[12] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):257–269.

[13] Eichler, K., Xu, F., Uszkoreit, H., Hennig, L., and Krause, S. (2016). TEG-REP: A corpus of textual entailment graphs based on relation extraction patterns.

[14] Gormley, M. R., Yu, M., and Dredze, M. (2015). Improved relation extraction with feature-rich compositional embedding models. *Computer Science*.

[15] Graves, A., Mohamed, A. R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[16] Gábor, K., Buscaldi, D., Schumann, A.-K., QasemiZadeh, B., Zargayouna, H., and Charnois, T. (2018). Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. pages 679–688.

[17] Hashimoto, K., Miwa, M., Tsuruoka, Y., and Chikayama, T. (2013). Simple customization of recursive neural networks for semantic relation classification. In *EMNLP*, pages 1372–1376.

[18] Hendrickx, I., Su, N. K., Kozareva, Z., Nakov, P., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2009). Semeval-2010 task 8: multi-way classification of semantic relations between pairs of nominals. In *The Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

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

[20] Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., and Weld, D. S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550.

[21] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., and Simonyan, K. (2017). Population based training of neural networks.

[22] Jensen, J. L. W. V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193.

[23] Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models.

[24] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

[25] Knoth, P. and Zdrahal, Z. (2012). Core: Three access levels to underpin open access. *D-Lib Magazine*, 18(11/12).

[26] Kullback, S. (1959). *Information Theory and Statistics*. WILEY.

[27] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

[28] Lafferty, J. D., Mccallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Eighteenth International Conference on Machine Learning*, pages 282–289.

[29] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. pages 260–270.

[30] Lauscher, A., Glavaš, G., Ponzetto, S. P., and Eckert, K. (2017). Investigating convolutional networks and domain-specific embeddings for semantic classification of citations. page in press. Association for Computing Machinery (ACM).

[31] Li, Q. and Ji, H. (2014). Incremental joint extraction of entity mentions and relations. In *Meeting of the Association for Computational Linguistics*, pages 402–412.

[32] Lin, R., Liu, S., Yang, M., Li, M., Zhou, M., and Li, S. (2015). Hierarchical recurrent neural network for document modeling. In *Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

[33] Liu, Y., Wei, F., Li, S., Ji, H., Zhou, M., and Wang, H. (2015). A dependency-based neural network for relation classification. *Computer Science*.

[34] Luan, Y., Ostendorf, M., and Hajishirzi, H. (2018). The UWNLP system at semeval-2018 task 7: Neural relation extraction model with selectively incorporated concept embeddings. pages 788–792.

[35] Luo, G., Huang, X., Lin, C. Y., and Nie, Z. (2015). Joint entity recognition and disambiguation. In *Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

[36] Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605.

[37] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). *Building a large annotated corpus of English: the penn treebank*. MIT Press.

[38] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

[39] Mikolov, T. A. (2012). Statistical language models based on neural networks.

[40] Mintz, Mike, Steven, Jurafsky, and Dan (2009). Distant supervision for relation extraction without labeled data. In *Joint Conference of the Meeting of the ACL and the International Joint Conference on Natural Language Processing of the Afnlp: Volume*, pages 1003–1011.

[41] Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. In *Meeting of the Association for Computational Linguistics*, pages 1105–1116.

[42] Miwa, M. and Sasaki, Y. (2014). Modeling joint entity and relation extraction with table representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 944–948.

[43] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):págs. 3–26.

[44] Neves, M., Butzke, D., Schönfelder, G., and Grune, B. (2018). Bf3r at semeval-2018 task 7: Evaluating two relation extraction tools for finding semantic relations in biomedical abstracts. pages 816–820.

[45] Nguyen, T. H. and Grishman, R. (2015). Relation extraction: Perspective from convolutional neural networks. In *The Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48.

[46] Nooralahzadeh, F., Øvrelid, L., and Lønning, J. T. (2018). Sirius-ltg-uio at semeval-2018 task 7: Convolutional neural networks with shortest dependency paths for semantic relation extraction and classification in scientific papers.

[47] Passos, A., Kumar, V., and Mccallum, A. (2014). Lexicon infused phrase embeddings for named entity resolution. *Computer Science*.

[48] Pei, W., Baltrusaitis, T., Tax, D. M. J., and Morency, L. P. (2016). Temporal attention-gated model for robust sequence classification. pages 820–829.

[49] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

[50] Qian, L., Zhou, G., Kong, F., Zhu, Q., and Qian, P. (2008). Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *COLING 2008, International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, Uk*, pages 697–704.

[51] Ren, X., Wu, Z., He, W., Qu, M., Voss, C. R., Ji, H., Abdelzaher, T. F., and Han, J. (2017). Cotype: Joint extraction of typed entities and relations with knowledge bases. pages 1015–1024.

[52] Rink, B. and Harabagiu, S. (2010). Utd: Classifying semantic relations by combining lexical and semantic resources. In *International Workshop on Semantic Evaluation*, pages 256–259.

[53] Rotsztejn, J., Hollenstein, N., and Zhang, C. (2018). Eth-ds3lab at semeval-2018 task 7: Effectively combining recurrent and convolutional neural networks for relation classification and extraction.

[54] Santos, C. N. D., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. *Computer Science*, 86(86):132–137.

[55] Shen, Y. and Huang, X. (2016a). Attention-based convolutional neural network for semantic relation extraction. In *COLING*.

[56] Shen, Y. and Huang, X. (2016b). Attention-based convolutional neural network for semantic relation extraction. In *COLING*.

[57] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. 4:2951–2959.

[58] Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

[59] Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 151–161.

[60] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014a). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[61] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014b). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[62] Suárez-Paniagua, V., Segura-Bedmar, I., and Aizawa, A. (2018). UC3M-NII team at semeval-2018 task 7: Semantic relation classification in scientific papers via convolutional neural network. pages 793–797.

[63] Suchanek, F. M., Ifrim, G., and Weikum, G. (2006). Combining linguistic and statistical analysis to extract relations from web documents. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 712–717.

[64] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. 4:3104–3112.

[65] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

[66] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[67] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. pages 260–269.

[68] Vu, N. T., Adel, H., Gupta, P., and Schütze, H. (2016). Combining recurrent and convolutional neural networks for relation classification. In *NAACL*.

[69] Wang, C. and Fan, J. (2014). Medical relation extraction with manifold models. In *Meeting of the Association for Computational Linguistics*, pages 828–838.

[70] Wang, L., Cao, Z., Melo, G. D., and Liu, Z. (2016). Relation classification via multi-level attention cnns. In *Meeting of the Association for Computational Linguistics*, pages 1298–1307.

[71] Wang, P., Qian, Y., Soong, F. K., He, L., and Zhao, H. (2015). Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *Computer Science*.

[72] Wei, C. H., Peng, Y., Leaman, R., Davis, A. P., Mattingly, C. J., Li, J., Wiegers, T. C., and Lu, Z. (2015). Overview of the biocreative v chemical disease relation (cdr) task. In *Biocreative Challenge Evaluation Workshop*, pages 154–166.

[73] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proc IEEE*, 78(10):1550–1560.

[74] Wu, F. and Weld, D. S. (2013). Open information extraction using wikipedia. *Proc Acl*, pages 118–127.

[75] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015a). Show, attend and tell: Neural image caption generation with visual attention. *Computer Science*, pages 2048–2057.

[76] Xu, K., Feng, Y., Huang, S., and Zhao, D. (2015b). Semantic relation classification via convolutional neural networks with simple negative sampling. *Computer Science*, 71(7):941–9.

[77] Xu, Y., Jia, R., Mou, L., Li, G., Chen, Y., Lu, Y., and Jin, Z. (2016). Improved relation classification by deep recurrent neural networks with data augmentation.

[78] Yan, X., Mou, L., Li, G., Chen, Y., Peng, H., and Jin, Z. (2015). Classifying relations via long short term memory networks along shortest dependency path. *Computer Science*, 42(1):56–61.

[79] Yang, B. and Mitchell, T. (2016). Joint extraction of events and entities within a document context. pages 289–299.

[80] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

[81] Yao, X. and Durme, B. V. (2014). Information extraction over structured data: Question answering with freebase. In *Meeting of the Association for Computational Linguistics*, pages 956–966.

[82] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Computer Science*.

[83] Yu, M., Gormley, M. R., and Dredze, M. (2014). Factor-based compositional embedding models. In *The NIPS 2014 Learning Semantics Workshop*.

[84] Yu, X. and Lam, W. (2010). Jointly identifying entities and extracting relations in encyclopedia text via A graphical model approach. pages 1399–1407.

[85] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *Eprint Arxiv*.

[86] Zeng, D., Liu, K., Lai, S., Zhou, G., and Zhao, J. (2014). Relation classification via convolutional deep neural network.

[87] Zhang, D. and Wang, D. (2015). Relation classification via recurrent neural network. *Computer Science*.

[88] Zhang, M., Tang, J., Qu, M., Yan, J., and Wang, M. (2015). Line: Large-scale information network embedding. 2(2):1067–1077.

[89] Zhang, R., Meng, F., Zhou, Y., and Liu, B. (2018). Relation classification via recurrent neural network with attention and tensor layers. *Big Data Mining and Analytics*.

[90] Zheng, S., Wang, F., Bao, H., Hao, Y., Zhou, P., and Xu, B. (2017). Joint extraction of entities and relations based on a novel tagging scheme. pages 1227–1236.