공학박사학위논문

Towards mastering complex reasoning with Transformers: applications in visual, conversational and mathematical reasoning

트랜스포머를 통한 복잡한 추론 능력 정복을 위한 연구: 시각적, 대화적, 수학적 추론에의 적용

2021 년 2 월

서울대학교 대학원

산업공학과

안 진 원

Towards mastering complex reasoning with Transformers: applications in visual, conversational and mathematical reasoning

트랜스 포머를 통한 복잡한 추론 능력 정복을 위한 연구: 시각적, 대화적, 수학적 추론에의 적용

지도교수 조 성 준

이 논문을 공학박사 학위논문으로 제출함

2020 년 12 월

서울대학교 대학원

산업공학과

안 진 원

안진원의 공학박사 학위논문을 인준함

2021 년 1 월

위 원 장	이 재 욱	(인)
부위원장	조 성 준	(인)
위 원	박 종 헌	(인)
위 원	강 필 성	(인)
위 원	이 영 훈	(인)

Abstract

Towards mastering complex reasoning with Transformers: applications in visual, conversational and mathematical reasoning

Jinwon An
Department of Industrial Engineering
The Graduate School
Seoul National University

As deep learning models advanced, research is focusing on sophisticated tasks that require complex reasoning, rather than simple classification tasks. These complex tasks require multiple reasoning steps that resembles human intelligence. Architecture-wise, recurrent neural networks and convolutional neural networks have long been the main stream model for deep learning. However, both models suffer from shortcomings from their innate architecture. Nowadays, the attention-based Transformer is replacing them due to its superior architecture and performance. Particularly, the encoder of the Transformer has been extensively studied in the field of natural language processing. However, for the Transformer to be effective in data with distinct structures and characteristics, appropriate adjustments to its structure is required. In this dissertation, we propose novel architectures based on the Transformer encoder for various supervised learning tasks with different data types and characteristics. The tasks that we consider are visual IQ tests, dialogue state tracking and mathematical question answering. For the visual IQ test, the input is in a visual format with hierarchy. To deal with this, we propose using a hierarchical Transformer encoder with structured representation that employs a novel neural network architecture to improve both perception and reasoning. The hierarchical structure of the Transformer encoders and the architecture of each individual Transformer encoder all fit to the characteristics of the data of visual IQ tests. For dialogue state tracking, value prediction for multiple domain-slot pairs is required. To address this issue, we propose a dialogue state tracking model using a pre-trained language model, which is a pre-trained Transformer encoder, for domain-slot relationship modeling. We introduced special tokens for each domain-slot pair which enables effective dependency modeling among domain-slot pairs through the pre-trained language encoder. Finally, for mathematical question answering, we propose a method to pre-train a Transformer encoder on a mathematical question answering dataset for improved performance. Our pre-training method, Question-Answer Masked Language Modeling, utilizes both the question and answer text, which is suitable for the mathematical question answering dataset. Through experiments, we show that each of our proposed methods is effective in their corresponding task and data type.

Keywords: Deep learning, Transformer, Supervised learning, Structured representations, Pre-training, Visual IQ test, Dialogue state tracking, Mathematical question answering

Student Number: 2014-21811

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Early deep learning models focused on simple tasks such as image classification (Krizhevsky et al., 2017) and word relationship identification (Mikolov et al., 2013). As deep learning models advanced, research is focusing on more complex reasoning tasks that resemble human intelligence. Instead of merely recognizing images, comparing images and inferring rules and patterns requires much more sophisticated reasoning. Another example can be tracking the state of the conversation between two speakers with changing topics, instead of understanding text documents with a single topic and single speaker. Also, solving mathematical problems that require understanding axioms and deriving intermediate steps requires much more cognitive ability than understanding plain text. To solve these complex tasks, more sophisticated architectures are required.

Architecture-wise, the success of Deep learning models started with two major neural network architectures: recurrent neural networks (RNNs) (Rumelhart et al., 1986) and convolutional neural networks (CNNs) (LeCun et al., 1989).

The RNN, with its recursive structures, was able to model sequential data with variable length, allowing it to capture long-range dependencies. This allowed RNNs to excel in sequence modeling tasks such as translation (Sutskever et al., 2014), sequence labelling (Graves, 2012) and sequence generation (Graves, 2013). However, due to its sequential structure, RNNs could not fully enjoy the parallel computing power that modern Graphical Processing Units (GPUs) provide.

The CNN, on the other hand, was specialized for processing image data due to its built-in structure of invariance towards spatial translation and locality. After AlexNet (Krizhevsky et al., 2017), various CNN architectures went on to make new records on image classification tasks (Simonyan and Zisserman, 2014; He et al., 2016; Szegedy et al., 2015). Convolution operations were easy to implement in a parallel manner, which enabled CNNs to take advantage of the parallel computation of GPUs. However, their use in modeling sequential data was less effective.

The Transformer (Vaswani et al., 2017) proposed a new model architecture based on the attention mechanism for solving sequence-to-sequence tasks, which turned out to excel at sequential modeling as well as to be appropriate for parallel computation. Attention was previously used in recurrent neural networks in sequence-to-sequence tasks such as machine translation (Bahdanau et al., 2014). Attention for RNNs enabled focusing on targeted dependencies regardless of the distance between the input and the output. However, the Transformer relies entirely on the attention mechanism, rather than using it as a sub-component. The attention function used in the Transformer is a mapping from a query and a set of key-value pairs. The Transformer is composed of an encoder and a decoder. The encoder is based on self-attention, where the query and key-value pairs are from the same source, the encoder input. The decoder is based on encoder-decoder attention, where the query is based on the decoder output and the key-value pairs are from the encoder output. In practice, instead of a single attention, multi-head attention is used. Also, other components such as residual connection and normalization as well as position-wise feedforward networks are included in the Transformer. The technical details of the Transformer are explained thoroughly in Chapter 2.

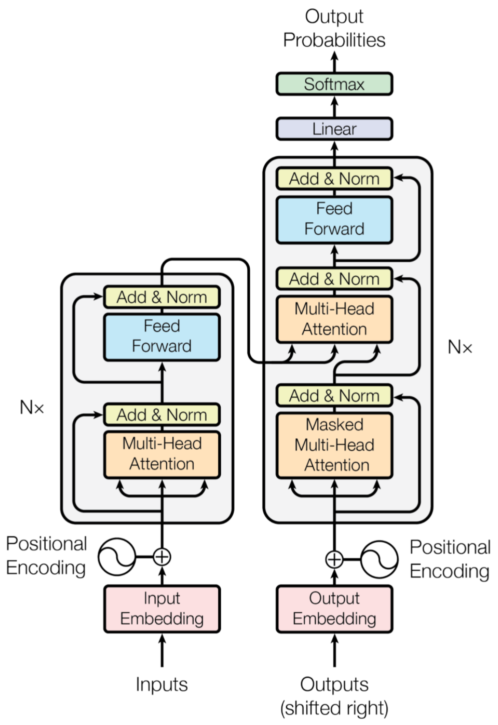


Figure 1.1: Overview of the Transformer. Figure from Vaswani et al. (2017).

The encoder of the Transformer is of particular interest because its structure can encode dependencies of sequential data effectively. Because of this characteristic, the Transformer encoder was used as a language model which is pre-train with unsupervised learning tasks such as masked language modeling and next sentence prediction (Devlin et al., 2019). Most of research that followed the Transformer was focused on natural language processing tasks (Dai et al., 2019; Yang et al., 2019b) or natural language with visual elements (Chen et al., 2019b; Tan and Bansal, 2019; Su et al., 2019). However, we believe that the Transformer can be used in various other tasks as well. In this dissertation, we propose models and methods to effectively to deal with different types of data using the Transformer encoder. Each task requires adjustments to the original Transformer architecture according to the characteristics of the dataset. We propose to solve the following three tasks using the Transformer encoder: Visual IQ tests, dialogue state tracking and mathematical question answering.

Visual IQ tests involve understanding the structure or pattern among given images or panel figures. In Raven's progressive matrices, a typical visual IQ test, the subjects are asked to complete the missing last entry of a 3×3 matrix of panel figures by selecting the most appropriate panel figure from a set of candidates. The images are usually given in abstract formats with particular shapes and sizes, which are generated according to a predefined pattern. This means that shapes inside each image has a structure. Also, the layout of the images as the panel figures also has a pattern. To solve this task, we propose to use the Transformer encoder in a hierarchical manner for both the shapes inside the image and image layout. This novel hierarchical use of the Transformer encoder enabled our model to achieve state-of-the-art performance on the RAVEN dataset (Zhang et al., 2019a), which is one of the benchmark dataset in Raven's progressive matrices.

Dialogue state tracking is a core component of a task-oriented dialogue system. It involves understanding the state of the dialogue between the user and the system, which is typically given in a triplet of $\{domain, slot, value\}$. If a user wants a cheap restaurant, the corresponding dialogue state is $\{restaurant, pricerange, cheap\}$. Pre-trained language encoders, which are pre-trained Transformer encoders, have been used for dialogue state tracking. However, we propose using the pre-trained language encoder for modeling dependencies among domain-slot pairs by introducing multiple special tokens that encodes information specific to their corresponding domain and slot. This novel introduction of the special tokens with the Transformer encoder enabled our model to achieve state-of-the-art performance on the MultiWOZ-2.1 dataset (Eric et al., 2019) among models without extra supervision, which is one of the benchmark datasets in dialogue state tracking.

Mathematical question answering is solving mathematical questions given in natural language. Pre-trained language models have achieved state-of-the-art results in many natural language processing tasks. Understanding mathematical problems has similarities with understanding natural language, in that it requires similar reasoning steps, such as recognizing words or symbols from a sequence of characters and identifying the meaning of those words and symbols in the context of the whole sequence. Thus, we propose a pre-training method for mathematical question answering. Our pre-training method, Question-Answer Masked Language Modeling, leverages both question and answer text by concatenating them into a single text. Models initialized with the pre-trained model not only showed improved results over the non-pretrained model, but also achieved the result with higher computational efficiency on the Mathematics dataset (Saxton et al., 2019).

The reminder of this dissertation is organized as follows. In Chapter 2, we review related works on the three types of tasks that we addressed using the Transformer encoder. In Chapter 3, we propose using a hierarchical Transformer encoder with structured representation that employs a novel neural network architecture to improve both perception and reasoning in a visual IQ test. In Chapter 4, we propose a dialogue state tracking model using a pre-trained language model, which is a pre-trained Transformer encoder, for domain-slot relationship modeling. In Chapter 5, we propose pre-training a Transformer encoder on a mathematical question answering dataset for improved performance. Finally, in Chapter 6 we discuss the contributions and future work of this dissertation.

Chapter 2

Literature Review

In this chapter, we start by reviewing the Transformer architecture in general, which is the main theme of this dissertation. Next, we presented previous studies on each of the three research topics we address.

2.1 Related Works on Transformer

The Transformer (Vaswani et al., 2017), which is based on multi-head attention, achieved state-of-the-art results on machine translation tasks. We focus on the Transformer encoder which is mainly based on self-attention. Self-attention allows every element to attend to all other elements, which enables each word to be considered in its full context. Self-attention is from a more general function of scaled dot attention which is formally defined as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.1}$$

where Q, K, V are a set of queries, keys and values respectively and d_k is the hidden dimension of K. Self-attention is a special case of the attention where the query, key and value are all the same: $Attention(X, X, X)$, where X is the input. Multi-head attention involves multiple linear projections of h times, which is intended to extract different representations or aspects of the data. h is the number of heads. This means that instead of performing a single attention with a dimension of d_{model} for the queries, keys and values, multi-head attention projects the queries, keys and values h times with different weights to dimensions of d_k, d_k and d_v, respectively. Multi-head attention is formally defined as follows:

$$MHA(Q, K, V) = concat(head_1, \cdots, head_h)W^O, \tag{2.2}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \tag{2.3}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$ and $d_{model}/h = d_k = d_v$.

The outputs of multi-head attention is followed by residual connection and layer normalization (Ba et al., 2016). This operation is abbreviated into *Add&Norm* in descriptions. Fig. 2.1 shows an overview of the attention.

$$Add\&Norm(X) = LayerNorm(X + MHA(X, X, X)). \tag{2.4}$$

Next, a position-wise feed-forward network is applied with a ReLU activation function.

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2, \tag{2.5}$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $b_1 \in \mathbb{R}^{d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$, $b_2 \in \mathbb{R}^{d_{model}}$. Lastly, this is followed by another *Add&Norm* operation. These operations form a single Transformer encoder block, which is stacked for L layers, where L is the number of

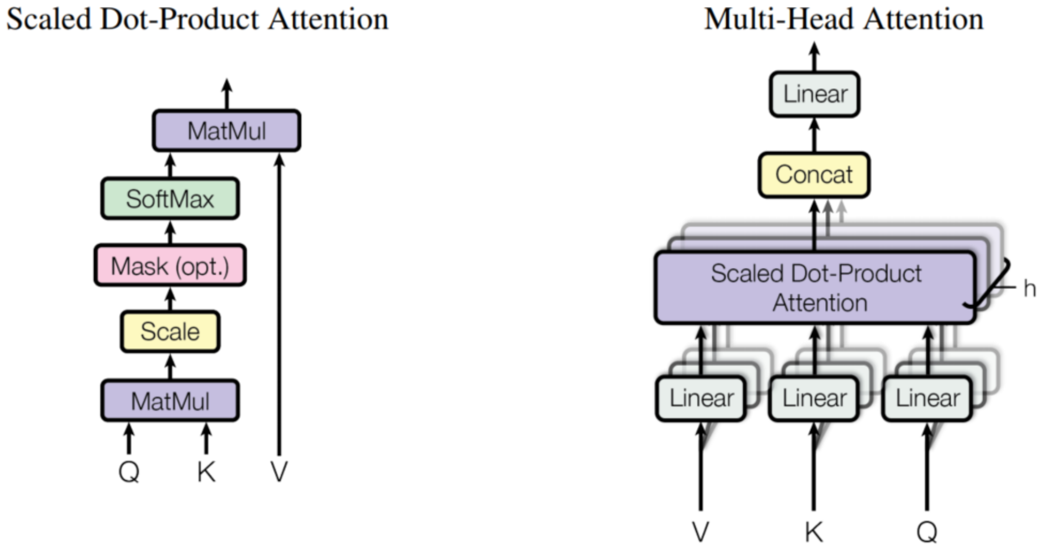


Figure 2.1: Overview of attention. Figure from Vaswani et al. (2017)

layers of the Transformer encoder.

After its development, other models based on the Transformer were introduced (Dai et al., 2019; Yang et al., 2019b). Other characteristics of the Transformer include position-wise feed-forward networks, positional encodings and residual connections. The Transformer has an encoder–decoder structure where an encoder embeds the source language and a decoder predicts the target language based on the encoder output. Transformer encoders, in particular, have been used in solving natural language problems such as question answering and language inference. BERT (Devlin et al., 2019) used the Transformer encoder with various pre-training tasks, such as masked language modeling and next sentence prediction. They also used segment embeddings when jointly embedding tokens from different sentences. BERT-style pre-training methods have also been used in visual question answering for multi-modal pre-training of images and text (Chen et al., 2019b; Tan and Bansal, 2019;

Su et al., 2019).

2.2 Related Works on Visual IQ Tests

In this section, we review the previous research related to neural network approaches to visual IQ tests such as Raven's Progressive Matrices as well as the object detection models.

2.2.1 RPM-related studies

After the introduction of large-scale datasets on RPM (Santoro et al., 2018; Zhang et al., 2019a), recent studies used deep learning models to solve RPM. Most of them use CNNs as the perception module (Santoro et al., 2018; Zhang et al., 2019b; Zheng et al., 2019). (Steenbrugge et al., 2018) takes a slightly different approach with unsupervised representational learning with β-variational autoencoders (β-VAE) (Higgins et al., 2017a). β-VAE is known to create disentangled features in latent space. The learned features from β-VAE were used instead of the ordinary CNN feature maps from the raw pixels. In terms of reasoning modules, each study proposes its own architecture. Santoro et al. (2018) proposed the Wild Relational Network, which was developed as a variant of the Relation network (Santoro et al., 2017) with adapted structures that enables explicit comparisons between different panels. Zhang et al. (2019b) proposed CoPINet which involves two core ideas: contrast effects and perceptual inference. In terms of contrast effects, model-level and objective-level contrasting were implemented in the model. Also, an additional perceptual head was used to infer the rules from the first two rows of the context panels. Zheng et al. (2019) presented a different approach using curriculum learning by means of reinforcement learning. A student-teacher architecture is proposed where the teacher adaptively optimizes the appropriate proportion of data that is to be learned by the student.

2.2.2 Object Detection related studies

Faster-RCNN (Ren et al., 2015) has become one of the standard models for object detection, which has been developing ever since (Wu et al., 2020a). Based on Fast R-CNN (Wang et al., 2017), this model adds a regional proposal network that informs the network on where to look for objects in images. It enables a more efficient discovery of region proposals for object detection models because the main bottleneck in Fast R-CNN has been computing region proposals. The advancement in object detection has been extended to other vision-related tasks, most notably in visual question answering (VQA) (Anderson et al., 2018; Kim et al., 2018; Jiang et al., 2018; Hu et al., 2019; Yi et al., 2018; Gao et al., 2020; Hong et al., 2020; Chen et al., 2020a; Yang et al., 2019a; Liu et al., 2018), referring expressions (Liu et al., 2019a; Zhang et al., 2018a) and caption generation (Anderson et al., 2018; Zeng et al., 2020).

One of the major studies that introduced using object detection models in VQA is Anderson et al. (2018). It uses object detectors and their corresponding object detection features for image features instead of pixel-level features from CNNs . After the introduction of this method, it became mandatory for VQA models to incorporate such techniques to achieve state-of-the-art results on various VQA benchmark datasets. While using object detection features as in Anderson et al. (2018), numerous studies have proceeded with different modes of fusion between the language and vision modalities (Kim et al., 2018; Jiang et al., 2018; Hu et al., 2019). A different path of research adopted by Yi et al. (2018) proposed to disentangle perception and reasoning by using object detection models for recognition and rule-based methods for reasoning.

2.3 Related works on Dialogue State Tracking

Recent work on dialogue state tracking can be largely divided into two groups according to how the slot-values are predicted: fixed-vocabulary and open-vocabulary. The fixed-vocabulary approach, also known as the picklisted-based approach, uses a classification module to predict the dialogue state for each slot from a pre-defined set of candidate values (Zhong et al., 2018; Nouri and Hosseini-Asl, 2018; Ramadan et al., 2018; Eric et al., 2019; Lee et al., 2019; Chen et al., 2020b). The open-vocabulary approach generates the dialogue state for each domain-slot pair either by using a generative decoder to generate text (Wu et al., 2019a; Hosseini-Asl et al., 2020) or by extracting text spans from the dialogue history (Gao et al., 2019; Goel et al., 2019; Heck et al., 2020). There is also an approach to use both picklist-based and span-based methods according to the slot type (Zhang et al., 2019c).

For models that deal with multi-domain dialogue, how they deal with different domain-slot pairs is another way to divide them. The first approach encodes the dialogue context independent of the domain-slot pairs and uses separate modules for each domain-slot pair (Eric et al., 2019; Gao et al., 2019; Goel et al., 2019; Heck et al., 2020). The second approach encodes the dialogue context using the domain-slot pair information as the prefix and run the encoder multiple times (Nouri and Hosseini-Asl, 2018; Wu et al., 2019a). Other approaches encode the dialogue context independently but merges it with domain-slot pair information later with a separate fusion module (Zhong et al., 2018; Ramadan et al., 2018; Lee et al., 2019). However, none of these models are able to model the relationship among different domain-slot pairs because there is no module that enables the interaction between them.

(Le et al., 2019) and (Chen et al., 2020b) directly models the relationship among different domain-slot pairs. (Le et al., 2019) uses a Fertility decoder to learn potential dependencies across domain-slot pairs, but without using a pre-trained language model. Also, their model requires additional data such as system action and delexicalized system responses for its performance. (Chen et al., 2020b) also explicitly models the relationship among different domain-slot pairs by using a Graph Attention Network (GAT) (Veličković et al., 2018). Schema graphs, which is the relation graph between domains and slots, are utilized for connecting edges in the GAT. Our work is different from these works in that we leverage the power of a pre-trained language encoder for directly modeling the dependencies among different domain-slot pairs.

(Hosseini-Asl et al., 2020) takes a different approach from the others by using multi-task learning that encompasses DST as well as action and response generation with a generative language model GPT-2 (Radford et al., 2019). However, since our work is focused on DST, we consider the model that is trained on DST only. In the decoding process, dialogue states for different domain-slot pairs are sequentially generated.

2.4 Related Works on Mathematical Question Answering

In this section, we review studies on pre-training neural networks in general and pre-trained models that are used in NLP. Next, we review previous studies on solving mathematical reasoning with neural networks. Last, we review the Transformer, which is the base model for our work.

2.4.1 Pre-training of Neural Networks

Pre-training of neural networks generally follow two phases. In the first phase, unsupervised pre-training, an unsupervised learning objective is used to initialize the weights of a neural network. This stage does not require any labeled data. In the next phase, supervised fine-tuning, a supervised learning objective is used to learn the weights of the neural network, which is pre-trained in the first phase. Deep Belief Networks (Hinton et al., 2006) and Stacked Denoising Auto-encoders (Vincent et al., 2010) are two models that represent this framework.

(Hinton et al., 2006) proposed Deep Belief Networks (DBN), a probabilistic generative model of multiple layers of latent variables, and its training strategy, a two-phase strategy of generative pre-training and discriminative fine-tuning. In the first phase, only the inputs are used to learn the weights using unsupervised learning. To be specific, the hidden layers are learned by greedy layer-wise pre-training using Restricted Boltzmann Machines (RBM). The weights of layers are learned in a top-down, layer-by-layer manner using the input, which works as a feature detector. After the hidden layer weights are pre-trained, discriminative fine-tuning follows by adding an additional layer for predicting labels given the input. Discriminative fine-tuning works much better if the hidden layers are initialized by the pre-training phase. DBN has been applied to precipitation forecast (Zhang et al., 2018b), future price prediction (Chen et al., 2019a), software defect prediction (Saifan and Al Smadi, 2019).

(Vincent et al., 2010) proposed Stacked Denoising Auto-Encoder (SDAE), which pre-trains the weights using a denoising auto-encoder, instead of using the RBM as in DBN. The denoising auto-encoder is trained to reconstruct a corrupted input, which is created by adding stochastic noise to the input. The layers are pre-trained layer-by-layer, which is 'stacking' pre-trained layers. Fine-tuning is proceeded in the same way as in DBN.

(Erhan et al., 2010) researched about the reason why unsupervised pre-training helps supervised learning tasks. Through experimentation, they show that pre-training mainly works as a regularizer, rather than an aid to optimization.

2.4.2 Language Model Pre-training

After the pioneering work of (Bengio et al., 2003), methods for representing words as vectors using neural networks has been continuously researched. The development of pre-trained language models can be largely divided into two phases (Qiu et al., 2020). The first phase being pre-trained word embeddings. (Mikolov et al., 2013) developed the popular Word2vec method for building word representations using shallow neural networks: Continuous Bag-of-Word (CBOW) and Skip-Gram models. They were able to demonstrate that the learned word embeddings exhibit semantic and syntactic regularities using vector arithmetic. A similar method for building pre-trained word embeddings is GloVe (Pennington et al., 2014), which incorporates the co-occurrence information of words.

In the first phase of pre-trained language models, representation of words is static. For example, the word “bank” is represented as the same vector in both “bank deposit” and “river bank”. In order to build word embeddings that depend on the context, in the second phase of language models, models are trained to produce contextual word embeddings. (Dai and Le, 2015) used a sequence autoencoder using LSTM (Hochreiter and Schmidhuber, 1997) in a sequence-to-sequence framework to pre-train the model to reconstruct the input sequence. (Ramachandran et al., 2017) used a pre-trained language models for translation tasks. The encoder and decoder are trained with monolingual data using the source and target language respectively, which showed improved performance compared to the models trained from scratch.

Modern Pre-trained language also produces contextualized embeddings of words. However, the size of the models became much bigger as deeper neural networks became more prevalent, which required a huge amount of data to be trained with. Also, as the self-attention based Transformer (Vaswani et al., 2017) set the new standard for NLP tasks, most pre-trained language models adopted the Transformer as their base model. BERT (Devlin et al., 2019) is one of first studies of modern pre-trained models. BERT involves two pre-training tasks: masked language modeling and next sentence prediction. In Masked language modeling (MLM), also known as cloze tasks, a portion of the input tokens are masked at random with a special mask token and the model is trained to predict the original id of the masked token. In Next sentence prediction, two sentences are either chosen from consecutive text sequences or not randomly. The model is trained to predict whether the two sentences are consecutive or not. It uses the encoder of the Transformer (Vaswani et al., 2017) as the base model.

It is known that this pre-training method gives a better initialization point for the Transformer encoder to better model contextualized representations of the input tokens. After the introduction of BERT, many follow-up research were conducted leading to various other models involving different pre-training tasks and Transformer architectures (Yang et al., 2019b; Dai et al., 2019; Lan et al., 2019; Liu et al., 2019b).

(Conneau and Lample, 2019) shows pre-training methods extended to multiple languages. Previous approaches focused on using monolingual text streams for pre-training. (Conneau and Lample, 2019) proposes a new pre-training method for translation data with paired sentences, which is called Translation Masked Language Modeling (TLM). In TLM, the source and target sentences from parallel translation data are concatenated into a single sentence. On this concatenated sentence, MLM is performed. This enables the masked token to either attend to its surrounding tokens or to the translated counterpart for prediction. This helps the model to leverage both languages for pre-training.

2.4.3 Mathematical Reasoning with Neural Networks

(Saxton et al., 2019) propose a dataset of various types of mathematics problems in free-form text called the Mathematics Dataset. The dataset is structured in a question answering format. Since the input and output are both sequences, a sequence-to-sequence framework, also known as an encoder-decoder model, can be applied. (Saxton et al., 2019) reported the performance of LSTM based models (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2014) and the Transformer (Vaswani et al., 2017) on the dataset. The Transformer model showed better results compared to the LSTM based models. Input and output sequences are given in character-level sequences.

(Lample and Charton, 2019) introduces another mathematics dataset which consists of two types of symbolic mathematics: function integration and ordinary differential equations. Compared to (Saxton et al., 2019), there is a limited variety of problem types. Also, instead of free-form text, each problem or expression is represented in a fixed form of trees. The Transformer model is trained for the evaluation of performance on the dataset.

Chapter 3

Hierarchical end-to-end architecture of Transformer encoders for solving visual IQ tests

3.1 Background

Abstract reasoning is one of the defining characteristics of human intelligence. To evaluate human capacity for abstract reasoning, visual IQ tests are proven to be surprisingly effective (Bilker et al., 2012; Carpenter et al., 1990; Hofstadter, 1995; Snow et al., 1984). Raven's Progressive Matrices (RPM) (Raven et al., 1998; Raven, 1936) is one such test. Given a 3×3 matrix of panel figures with the last figure missing, the subjects are asked to complete the missing entry by inferring a common rule from the first two rows and applying it to the third row. These panels are called context panels and are formulated as multiple-choice questions with a number of candidate panels to choose from. Within each panel, basic visual elements called entities are laid out in predefined configurations. Fig. 3.1 shows an example of an RPM question from the RAVEN dataset (Zhang et al., 2019a).

There have recently been a number of studies on RPM with neural networks (Santoro et al., 2018; Zhang et al., 2019b; Steenbrugge et al., 2018; Zhang et al., 2019a). The structure of models for solving RPM can be largely broken into two areas: perception and reasoning. The perception module reads the image input and

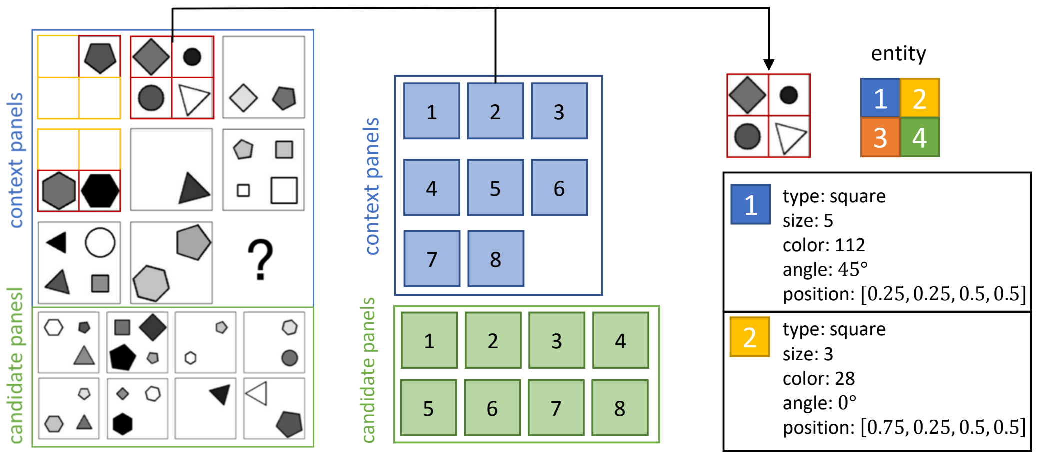


Figure 3.1: An example from the RAVEN dataset. The first column shows an example of an RPM question from RAVEN. The rule here is the distribution of the number of objects in each panel within each row, which is (1, 2, 4) objects. Since the first and second panel of the last row has (4, 2) objects, the answer panel should have 1 object. Thus, the appropriate answer would be the seventh candidate panel. Red boxes indicate positions in the layout grid where entities exist, whereas yellow boxes indicate the absence of entities in such positions. This applies to other panels as well. The second column shows panel numbering. The candidate panel numbers indicate the index order of the candidates, not class labels. The third column shows an example of entities in the second context panel. Each entity has its own attributes and position information. Should be viewed in color.

produces features. The reasoning module uses the features from the perception module to model the relationships among those elements to solve the problem. We next discuss the approaches undertaken by previous studies and how to improve them in terms of these two modules.

3.1.1 Perception

The perception module of current models (Santoro et al., 2018; Zhang et al., 2019b; Steenbrugge et al., 2018; Zhang et al., 2019a) are all based on convolutional neural networks (CNNs). Each panel image is passed through a CNN to obtain feature maps. However, it can be difficult for the CNN to extract patterns from raw pixels because the feature maps are produced in uniform grids. In RPM, however, there are primitive visual elements of salient objects called entities. A comparison between the CNN feature maps and salient object features is shown in Fig. 3.2. By separately extracting salient objects, we can acquire independent information for each entity. In addition, by predicting each attribute of the entity, such as color, size, angle, and type, we can acquire structured features for each entity. It is difficult for CNN feature maps to encode such attributes in a disentangled manner. The perception module can help the reasoning module work with quality features by providing these structured features rather than feature maps of uniform grids. To provide such features, we propose the use of object detection models (Ren et al., 2015; Wang et al., 2017; Girshick et al., 2015) that can extract salient objects from a given image and represent them as structured features by predicting the value of each attribute.

3.1.2 Reasoning

After the perception module extracts features from each panel, the reasoning module takes them as inputs to model the relationships or dependencies among the panels. Previous studies used a variety of methods to model the relationship among panels. Santoro et al. (2018) used Relational Networks (Santoro et al., 2017) as the basic structure of their model that supports direct comparison among elements. However, only pairs of two panels are modeled, even though the rules of RPM are based in terms of rows that consist of three panels. Zhang et al. (2019b) used contrast modules inspired by contrast effects (Bower, 1961) which encourages finding differences among candidates that help choose the correct answer. However, when aggregating

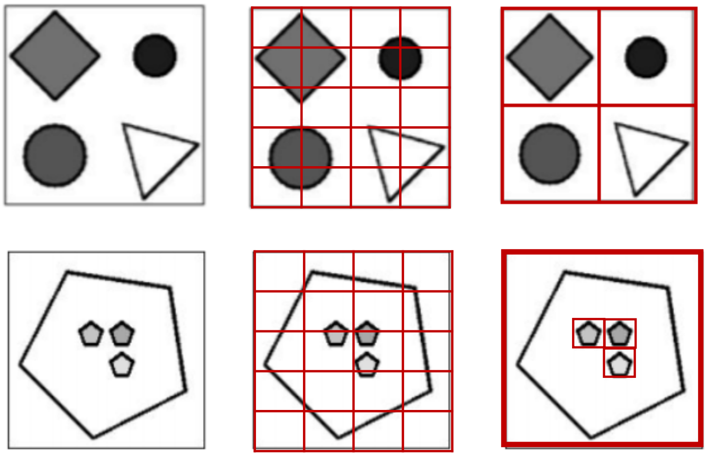


Figure 3.2: Grid features vs. salient object features. The first column shows the panel image. The second column shows the uniform grid features from a CNN. The third column shows salient object features from object detection models. Compared to the grid features, salient object features correctly find the entities in each layout.

the information on the three panels that make up rows, a simple summation was used without explicitly modeling the relationships between the panels within a row. The inductive bias that the RPM calls for are as follows: 1) involve processing all the panels at the same time 2) differentiate the panels in terms of row positions. To incorporate this inductive bias in our model, we propose using the Transformer encoder (Vaswani et al., 2017) that can model relationships among entities and among panels. The Transformer encoder is mainly based on self-attention which allows every input element to attend over all other elements at the same time. This structure, which allows each element to directly interact with the other, is the critical reason for selecting this structure. Within each panel, there exists one or more entities. For example, when we use recurrent neural networks to model the relationship among entities in each panel, the order in which the entity features of each panel are put into the model matters. However, if we use Transformer encoders because of their permutation invariance, this problem is overcome. This also applies to the panels. All panels should be able to interact with one another at once, rather than in pairs, as in Santoro et al. (2018). The Transformer encoder allows modeling all of the panels together according to their corresponding dependencies. Entities make up panels, and panels make up rows. To address this hierarchy, we first use the entity Transformer encoder to model the relationships among entities in each panel. After the panel embeddings are extracted by merging the information of entities, we model the relationships among panels using the panel Transformer encoder.

Our contribution is two-fold

- We introduce the use of object detection models for a perception module that can extract structured features of entities from panels that a CNN cannot provide
- We use a hierarchical Transformer encoder to obtain contextualized features of entities and panels that enables modeling dependencies among elements more effectively.

Our model extracts meaningful features by using the perception module and reasoning module. To further improve performance, we applied the idea of contrasting from Zhang et al. (2019b), which proposes methods for dealing with multiple-choice problems. We evaluate the performance of our model on the RAVEN dataset (Zhang et al., 2019a). Fig. 3.3 shows an overview of our model.

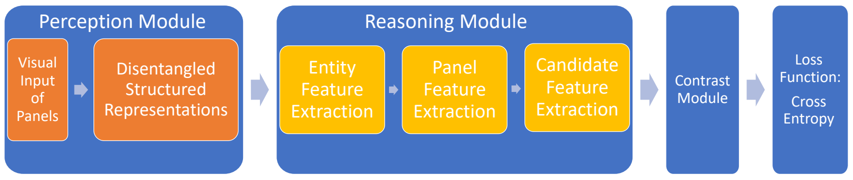

Perception Module
Visual Input of Panels
Disentangled Structured Representations
Reasoning Module
Entity Feature Extraction
Panel Feature Extraction
Candidate Feature Extraction
Contrast Module
Loss Function: Cross Entropy

Figure 3.3: Model overview

3.2 Proposed Model

The key idea of our model is two-fold. The first idea is to extract structured representations of entities by using object detection models as the perception module. The second idea is to use a hierarchical Transformer encoder as the reasoning module. We also incorporate the idea of contrasting introduced by Zhang et al. (2019b). We explain each model in detail in the following sections. The problem setting is described in Fig. 3.1.

3.2.1 Perception Module: Object Detection Model

For the perception module, we train an object detection model to acquire a structured representation of each entity in a panel that is disentangled in terms of attribute types. Since most object detection models only predict the category of the object, we include additional region of interest (ROI) heads to predict different attributes. We use the type attribute as the category or class label for the object detection model. Other attributes were predicted using additional ROI heads. Normally, object detection models find only existing objects in an image. However, in terms of identifying patterns, the non-existence of an object in a given layout is often of the same importance as the existence of an object. To provide the model with such information, we added an ”empty” category for ROI heads to predict

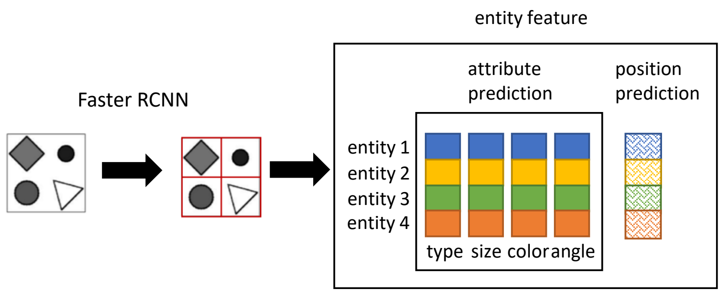


Figure 3.4: Perception module. For each entity, the object detection model (Faster R-CNN) outputs a discrete prediction value for each attribute. Position predictions are shared across different attributes. Every context and candidate panel runs through the perception module to obtain its entity feature.

regions of layouts without entities present. Fig. 3.1 shows an example of this, where yellow boxes indicate empty regions in a layout and red boxes indicate regions where entities exist.

We use discrete prediction results of the object detection for each attribute, not the intermediate features from the ROI pooling. Distinguishing each attribute has the effect of obtaining structured features. This is different from the approaches used in VQA (Anderson et al., 2018; Kim et al., 2018; Tan and Bansal, 2019; Su et al., 2019)), which also involve using object detection models for input features but in the form of ROI pooled features. For the positional information of each detected object, we match the regression output of bounding box coordinates with the closest ground truth bounding box position, which is defined according to the layout in each panel. Positions in predefined layouts are treated as discrete labels; this is because we use positional embedding vectors instead of bounding box coordinates to represent the positional information of entities to provide better performance. Fig. 3.4 shows how the object detection model works.

3.2.2 Reasoning Module: Hierarchical Transformer Encoder

The reasoning module takes the predictions of the perception module and learns the relationships among entities and panels using a variant of the Transformer encoder, which works in a hierarchical manner. First, the relationships among entities are modeled using the entity Transformer encoder. After the entity features are encoded as panel embeddings, panel Transformers encode the relationship among panels.

Entity Feature Extraction Module

The output of the object detection model contains the attribute and position prediction values as entity features for each panel. Because the attribute predictions from the object detection model are discrete variables, we embed each attribute into a continuous space. We embed the position value as well, which is shared for all attributes and added element-wise. The entity Transformer encoder learns the relationship among the entities in each panel. After the entity features are modified through the Transformer encoder, we aggregate the information on all the entities by summing them up, resulting in a panel embedding. To maintain the separation of attribute features from the object detection model, each attribute is separately passed through the same entity Transformer encoder. In other words, the entity Transformer encoder is shared for different attributes. Fig. 3.5 shows this process. The Entity feature extraction module for attribute a can be formally represented as follows:

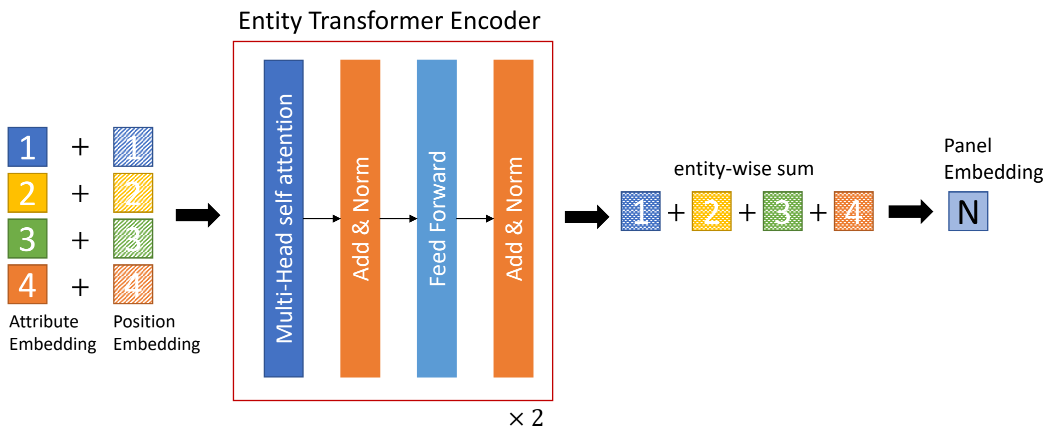


Figure 3.5: Entity feature extraction module. N is the panel number for context and candidate panels as shown in Fig. 3.1. This process is repeated for each attribute separately with the same entity Transformer encoder.

$$[\widehat{E_a^1}, \cdots, \widehat{E_a^n}] = Entity_{enc}([E_a^1 + E_p^1, \cdots, E_a^n + E_p^n]), \tag{3.1}$$

$$P_a^N = \widehat{E_a^1} + \cdots + \widehat{E_a^n}, \tag{3.2}$$

where E_a^k and E_p^k represents the attribute embedding and position embedding for k^{th} entity, respectively. n is the number of entities for the N^{th} panel. $\widehat{E_a^k}$ represents the the entity features for attribute a from the entity Transformer encoder. P_a^N is the panel embedding for N^{th} panel.

Panel Feature Extraction Module

The output of the entity feature extraction module for each attribute is panel embeddings for each panel, which is used as input for the panel Transformer encoder. The panel Transformer encoder is used to model the relationships among the panels.

As with the entity feature extraction module, each attribute is passed through the same panel Transformer encoder, but in a separate manner to maintain the disentanglement of attribute features. The input representation of the panel feature for each attribute is constructed by summing the panel embeddings with various segment embeddings for each panel. A visual representation of our input representation is shown in Fig. 3.6. The segment embeddings are used indicate row information, row item sequence information within a row and context/candidate panel type embedding. Row embedding is used to indicate which panels are in the same row. There are three row embedding vectors, as there are three rows. The row-item sequence embedding indicates the sequential information of panels in each row: first, second, and third panels for each row. This embedding vector is shared across all rows. Finally, the context/candidate panel type embedding is used to distinguish the context panels and candidate panels. A special [SEP] token embedding is used to differentiate between the different rows. To ensure permutation invariance, we randomly swap the first and second rows during training.

After the input representation is computed, it is passed through the panel Transformer encoder to model the dependencies among panels, as shown in Fig. 3.7. We use the candidate panels from the output of the panel Transformer encoder for each attribute. As each attribute is separately passed through the panel Transformer encoder, we merge the attribute information of each candidate panel by concatenating the candidate panels of each attribute. This concatenated feature for each candidate panel, which we call the candidate feature, is used as the input to the contrast module. Fig. 3.8 illustrates how the panel feature extraction module works. The panel feature extraction module can be formally represented as follows:

$$[\widehat{P_a^1}, \cdots, \widehat{P_a^8}] = Panel_{enc}([P^{Context}, P_a^1, \cdots, P_a^8]), \tag{3.3}$$
$$a \in \{type, size, color, angle, position\},$$

$$C^n = [\widehat{P_{type}^n}; \cdots; \widehat{P_{position}^n}], \quad n = 1, \cdots, 8, \tag{3.4}$$

where $P^{Context}$ are the panel embeddings of the context panels with the special $[SEP]$ token embedding between rows. $P_a^1, \cdots, P_a^8$ are the panel embeddings of the candidate panels. $\widehat{P_a^1}, \cdots, \widehat{P_a^8}$ are the panel features for attribute a from the panel Transformer encoder. ; indicates concatenation. $C^1, \cdots, C^8$ are the candidate features.

3.2.3 Contrasting Module and Loss function

We implement the idea of contrast effects as proposed in Zhang et al. (2019b). It essentially finds the differences among candidates by comparing each candidate against the aggregate. This contrast effect can be formally represented as:

$$Contrast(F_{context,c}) = F_{context,c} - h\big(\sum_{c \in C} F(context, c)\big), c \in C \tag{3.5}$$

where C represents all the candidates, and context refers to the context panels. F represents the model that learns the dependencies among context and candidate panels, which is the output of panel feature extraction module in our model.

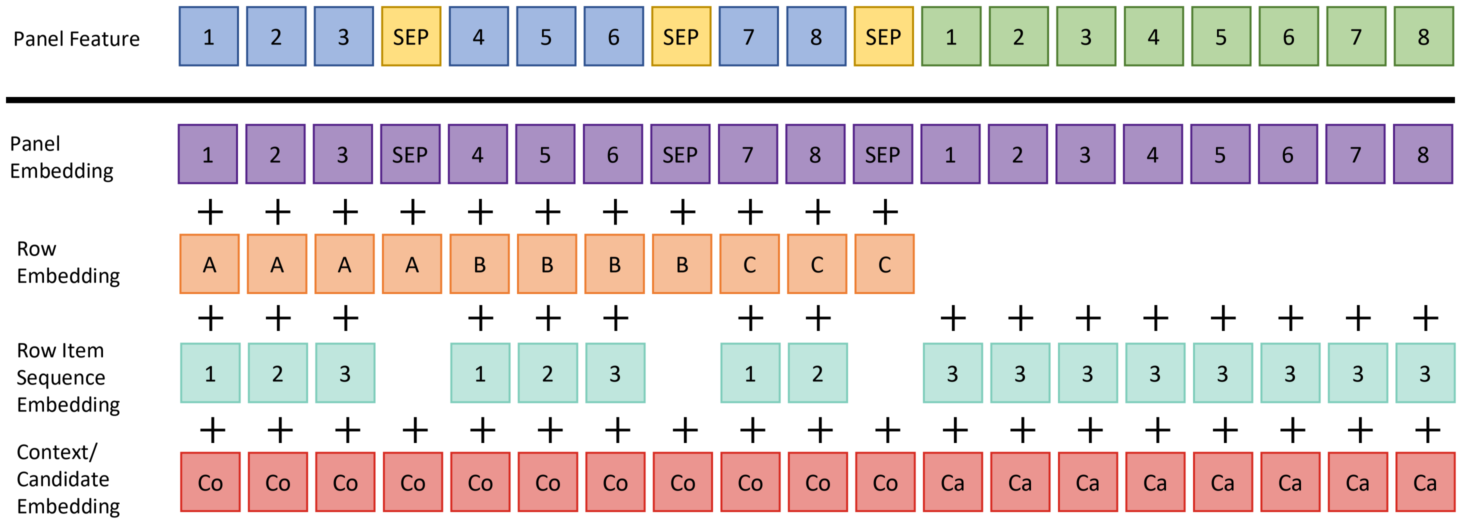


Figure 3.6: Panel input representation of a single attribute. Blue indicates context panels. Green indicates candidate panels. Co and Ca indicates context and candidate panel embeddings respectively. This panel input representation is repeated for each attribute separately. Should be viewed in color.

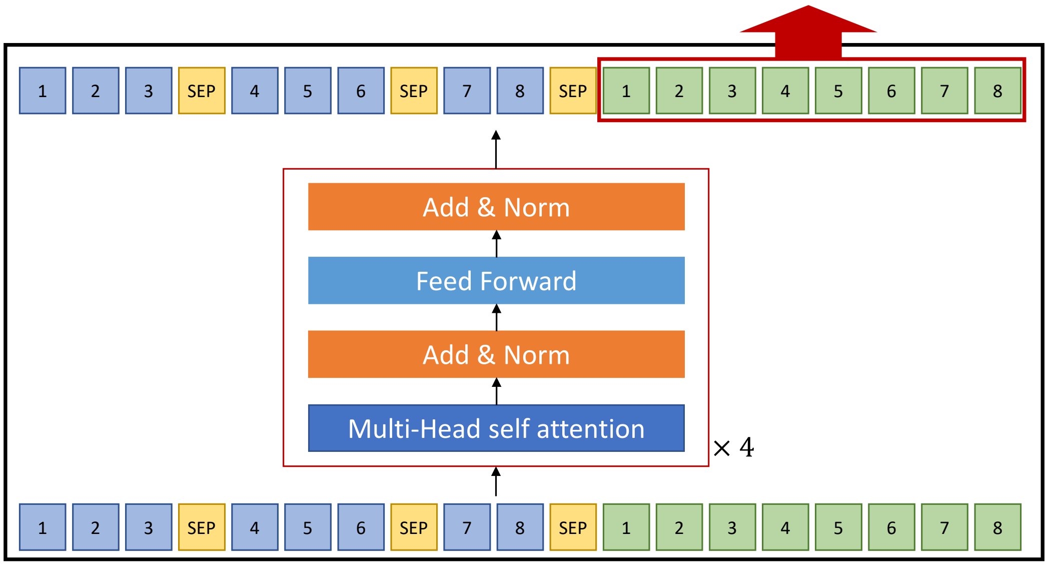


Figure 3.7: Panel Transformer encoder. $\times 4$ indicate that the Transformer encoder is stacked for 4 layers. Only the features for the candidate panels are used for the next step. Should be viewed in color.

$h(\sum_{c \in C} F(context, c)$ represents the aggregate information that summarizes the candidate features. Subtracting this information from each candidate helps the model to distinguish each candidate from each other. Since our the panel Transformer encoder involves self-attention, the candidate panel features already involves the context panel features. Fig. 3.9 illustrates the contrasting module. We use a 2-layer multi-layer perceptron (MLP) for the aggregate function h. To further enforce the contrast effects, Zhang et al. (2019b) proposed to use a variant noise contrastive estimation loss (Gutmann and Hyvärinen, 2010), rather than the cross-entropy loss. The proposed loss, in terms of implementation, is basically binary cross entropy loss for each candidate panel. Please refer to Zhang et al. (2019b) for further details about the contrasting effects in RPM. We apply a final output MLP and use it as

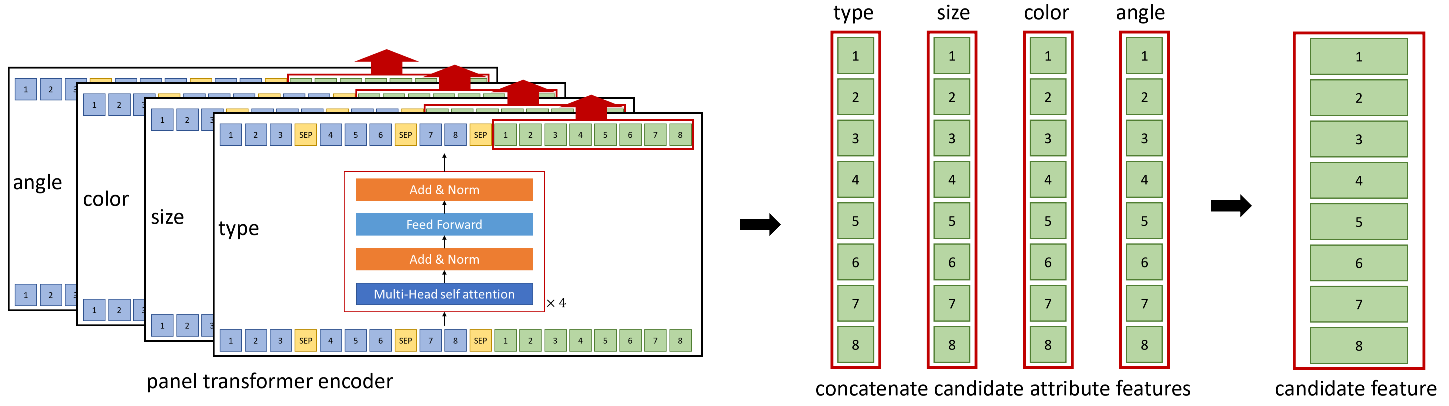


Figure 3.8: Panel feature extraction module. Each attribute is separately passed through the same panel Transformer encoder and the candidate panel features are concatenated.

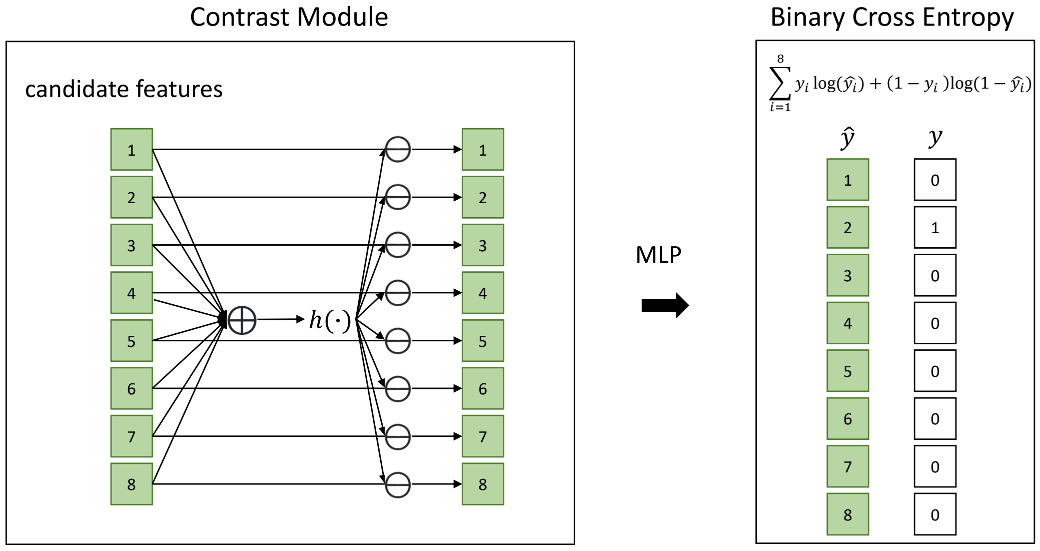


Figure 3.9: Contrast Module and Loss Function.

the logit value. Binary cross entropy is used for the loss function. For the prediction of each problem, we chose the candidate panel with the highest logit score.

3.3 Experimental results

In this section, we explain the dataset we used to evaluate our model and show the results.

3.3.1 Dataset

We train and evaluate our models on the RAVEN dataset (Zhang et al., 2019a), one of the two major benchmark datasets for computational models based on Raven's Progressive Matrices. There are 70,000 problems in the RAVEN with 8 context panels and 8 candidate panels. There are 7 different configurations of entities for the images. Examples of each configuration are shown in Fig. 3.10. There are train, validation and test splits designated in the dataset that consists of 60%, 20%, and 20% of the total data, respectively. RAVEN is accompanied by structural annotations, including information on the entities in panels. Each entity has its size, type, color and angle information as attributes and also position information. We would like to point out that solving RAVEN is different from ordinary classification tasks. This is because RAVEN involves selecting the answer from a given set of candidates, not from predefined set of classes. The candidate panel numbers indicate the index order of the candidates, not class labels. Procedurally Generated Matrices (PGM) (Santoro et al., 2018) is another major dataset created to measure model performance on RPM questions. However, PGM does not provide any structural annotations where object detection models can be trained. Therefore, we could not run experiments on this dataset.

There were studies that pointed out that the RAVEN dataset has a biased answer set that modifies only one attribute for the answer candidate (Hu et al., 2020; Wu et al., 2020b). This defect of the RAVEN dataset enables the model to find the answer from the candidate panels without looking into the context panel, exploiting a shortcut solution. For a fairer evaluation of our model, we note that our model can fully take advantage of this bias in the answer set, especially because we use the perception module for acquiring disentangled values for all the attributes of each entity.

3.3.2 Experimental Setup

We train a modified version of the Faster-RCNN (Ren et al., 2015) implemented in Wu et al. (2019b) for the object detection model. We used a ResNet 50 (He et al.,

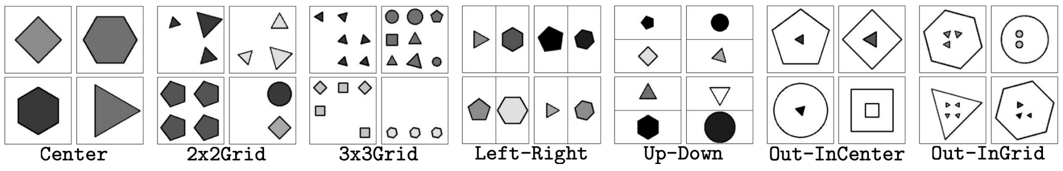


Figure 3.10: RAVEN Configurations. 4 panel examples shown for each configuration.

2016) backbone version, which is pre-trained on ImageNet (Deng et al., 2009). For training the reasoning module, we used an ADAM optimizer (Kingma and Ba, 2014) with default options. Learning rates were decayed by half when the validation loss did not decrease for more than 30 epochs. The embedding vector dimension is 32 for all attributes. The entity Transformer encoder has 2 heads and 2 layers and the panel Transformer encoder has 4 heads and 4 layers. The hidden dimensions for both Transformer encoders are 64. All models were run for 400 epochs.

3.3.3 Results for Perception Module

We used panels from the training split and did not use the test data for training the object detection model. Our object detection model detects objects by their types. Other attribute values are predicted based on those detected objects. Recall and precision for the entities was 100% for all type attributes with a confidence threshold set at 0.9. This can be expected since the entities in the figures are in a structured format, which makes it easy for object detection models to find their locations. For the additional prediction of attributes of those detected entities, Table 3.1 shows the attribute prediction accuracy of the object detection model on data that was not used for training. Additionally, the average L1-regression error of position regression is also very low at 0.13 with an image size of 160×160. It also shows perfect prediction

	Size	Color	Angle	Position
Accuracy	100%	100%	85.2%	100%

Table 3.1: Attribute prediction accuracy of the object detection model for detected entities

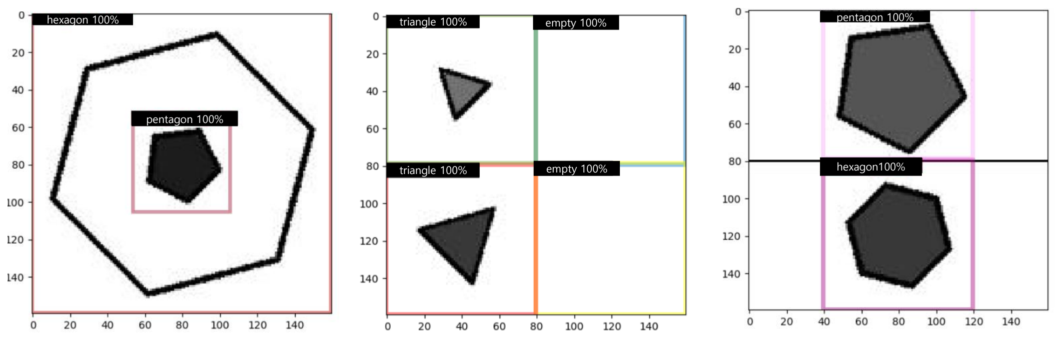


Figure 3.11: Object detection prediction examples. This is an example for the prediction of the type attribute.

for attributes of type, size and color. The accuracy of the angle attribute is not as high as that of the others, but this is expected because CNNs are not particularly good at recognizing angles. Fig. 3.11 shows the prediction examples of the object detection model.

3.3.4 Results for Reasoning Module

Table 3.2 shows the overall results of our model compared with previous studies. Our model shows the best performance compared to the previous best model by more than 8% points overall. In addition, our model's score beats the previous best model's for every configuration. Specifically, in the configurations of 2×2Grid and 3×3Grid, our model is better than the previous state of the art model by over 18% points. These two configurations have more complex structures than the other configurations. This suggests that structured features and a sophisticated reasoning module helped solve complex problems. In other simple structures such as the Center, LeftRight, Up-Down, and Out-InCenter, our model achieves near-perfect accuracy. We also tried cross entropy loss, which showed inferior performance. This is in line with Zhang et al. (2019b), where binary cross entropy loss showed better performance than cross entropy loss.

3.3.5 Ablation studies

We conduct ablation studies on our model to evaluate which structures of our model contribute to the enhancement in performance. Table 3.3 shows the results. First, we verify the performance of the perception module. We used the CNN instead of the object detection model, which showed a considerable loss in performance. This indicates that using a object detection model as the perception module is a major source of improvement in our model. We disabled either the entity or panel Transformer Encoder. Without the entity Transformer encoder, the accuracy is below the previous state-of-the art results although they are still above that of human-level. This emphasize that it is important to use a model architecture that has an inductive bias when treating disentangled structured representations. The panel Transformer encoder enhances the gain in performance although the increase is relatively small. For maximum performance we need to incorporate both the entity and panel Transformer encoder.

Method	Overall	Center	2x2Grid	3x3Grid	L-R	U-D	O-IC	O-IG
LSTM (Zhang et al., 2019a)	13.07	13.19	14.13	13.69	12.84	12.35	12.15	12.99
WReN (Santoro et al., 2018)	14.69	13.09	28.62	28.27	7.49	8.38	10.56	
WReN-NoTag-Aux (Zhang et al., 2019b)	17.62	17.66	29.02	34.67	7.69	7.89	12.3	13.94
CNN (Zhang et al., 2019a)	36.97	33.58	30.3	33.53	39.43	41.26	43.2	37.54
ResNet (Zhang et al., 2019a)	53.43	52.82	41.85	44.29	58.77	60.16	63.19	53.12
ResNet+DRT (Zhang et al., 2019a)	59.56	58.08	46.53	50.4	65.82	67.11	69.09	60.11
CoPINet (Zhang et al., 2019b)	91.42	95.05	77.45	78.85	99.1	99.65	98.5	91.35
Proposed Model	**99.62**	**100**	**97.9**	**99.9**	**99.95**	**99.9**	**99.8**	**100**
Human (Zhang et al., 2019b)	84.41	95.45	81.82	79.55	86.36	81.81	86.36	81.81

Table 3.2: Test accuracy of RAVEN models. L-R stands for Left-Right, U-D stands for Up-Down, O-IC stands for Out-InCenter, and O-IG stands for Out-InGrid, corresponding to Fig. 3.10.

Perception	Entity TE	Panel TE	Overall Accuracy
CNN	O	O	85.86
Object Detection	X	X	86.19
Object Detection	X	O	87.19
Object Detection	O	X	97.1
Object Detection	O	O	99.62

Table 3.3: Test accuracy of ablation studies. TE stands for Transformer Encoder. O means that the structure is used and X means that the structure is disabled.

3.4 Chapter Summary

In this chapter, we propose a hierarchical Transformer encoder with structured representation that achieves state of the art results on the RAVEN dataset that improves both perception and reasoning. For perception, we used object detection models to acquire structured representations. For reasoning, we used the Transformer encoder in a hierarchical way to effectively model the dependencies among the entities and panels. Our model outperforms the previous state-of-the-art model, CoPINet, by more than 8% points in terms of overall accuracy, achieving a near perfect score. However, it is reported that the RAVEN dataset has a bias answer set that can exploit a shortcut solution. We will investigate into this matter in future research.

Chapter 4

Domain-slot relationship modeling using Transformers for dialogue state tracking

4.1 Background

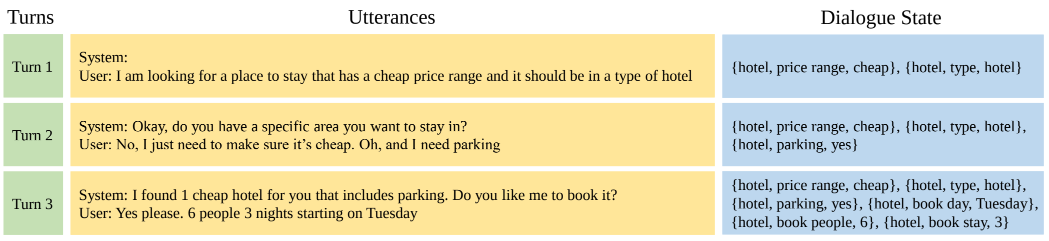
A task-oriented dialogue system is designed to help humans solve tasks by understanding their needs and providing relevant information accordingly. For example, such a system may assist its user with making a reservation at an appropriate restaurant by understanding the user's needs for having a nice dinner. It can also recommend an attraction site to a travelling user, accommodating the user's specific preferences. Dialogue State Tracking (DST) is a core component of these task-oriented dialogue systems, which aims to identify the state of the dialogue between the user and the system. DST represents the dialogue state with triplets of the following items: a domain, a slot, a value. A set of {*restaurant, price range, cheap*}, or of {*train, arrive-by, 7:00 pm*} are examples of such triplets. Fig. 4.1 illustrates an example case of the dialogue state during the course of the conversation between the user and the system. Since a dialogue continues for multiple turns of utterances, the DST model should successfully predict the dialogue state at each turn as the conversation proceeds. For multi-domain conversations, the DST model should be able to track dialogue states across different domains and slots.

Past research on multi-domain conversations used a placeholder in the model to represent domain-slot pairs. A domain-slot pair is inserted into the placeholder in each run, and the model runs repeatedly until it covers all types of the domain-slot pairs. (Wu et al., 2019a; Zhang et al., 2019c; Lee et al., 2019). A DST model generally uses an encoder to extract information from the dialogue context that is relevant to the dialogue state. A typical input for a multi-domain DST model comprises a sequence of the user's and the system's utterances up to the turn t, X_t, and the domain-slot information for domain i and slot j, D_iS_j. In each run, the model feeds the input for a given domain-slot pair through the encoder.

$$f_{encoder}(X_t, D_iS_j) \text{ for } i = 1, \cdots, n, \quad j = 1, \cdots, m, \tag{4.1}$$

where n and m is the number of domains and slots, respectively. However, because each domain-slot pair is modeled independently, the relationship among the domain-slot pairs can not be learned. For example, if the user first asked for a hotel in a certain place and later asked for a restaurant near that hotel, sharing the information between {*hotel, area*} and {*restaurant, area*} would help the model recognize that the restaurant should be in the same area as the hotel.

Recent approaches address these issues by modeling the dialogue state of every



Turns	Utterances	Dialogue State
Turn 1	System: User: I am looking for a place to stay that has a cheap price range and it should be in a type of hotel	{hotel, price range, cheap}, {hotel, type, hotel}
Turn 2	System: Okay, do you have a specific area you want to stay in? User: No, I just need to make sure it's cheap. Oh, and I need parking	{hotel, price range, cheap}, {hotel, type, hotel}, {hotel, parking, yes}
Turn 3	System: I found 1 cheap hotel for you that includes parking. Do you like me to book it? User: Yes please. 6 people 3 nights starting on Tuesday	{hotel, price range, cheap}, {hotel, type, hotel}, {hotel, parking, yes}, {hotel, book day, Tuesday}, {hotel, book people, 6}, {hotel, book stay, 3}

Figure 4.1: An example of a dialogue and its dialogue state.

domain-slot pair in a single run, given a dialogue context (Chen et al., 2020b; Le et al., 2019). This approach can be represented as follows:

$$f_{encoder}(X_t, D_1S_1, \cdots, D_nS_m). \tag{4.2}$$

Because the encoder receives all of the domain-slot pairs, the model can factor in the relationship among the domain-slot pairs through the encoding process. For the encoder, these studies used models that are trained from scratch, without pre-training. However, since DST involves natural language text for the dialogue context, using a pre-trained language model can help improve the encoding process. Several studies used BERT (Devlin et al., 2019), a pre-trained bidirectional language model, for encoding the dialogue context (Zhang et al., 2019c; Lee et al., 2019; Chao and Lane, 2019; Gao et al., 2019), but did not model the dependencies among different domain-slot pairs. Our approach fills the gap between these previous studies. In this work, we propose a model for multi-domain dialogue state tracking that effectively models the relationship among domain-slot pairs using a pre-trained language encoder. We modify the input structure of BERT, specifically the special token part of it, to adjust it for multi-domain DST.

The $[CLS]$ token of BERT (Devlin et al., 2019) is expected to encode the aggregate sequence representation as it runs through BERT, which is used for various downstream tasks such as sentence classification or question answering. This $[CLS]$ token can also be used as an aggregate representation for a given dialogue context. However, in a multi-domain dialogue, a single $[CLS]$ token has to store information for different domain-slot pairs at the same time. In this respect, we propose to use multiple special tokens, one for each domain-slot pair. Using a separate special token for each domain-slot pair is more effective in storing information for different domains and slots since each token can concentrate on its corresponding domain and slot.

We consider two different ways to represent such tokens: *DS-merge* and *DS-split*. *DS-merge* employs a single token to represent a single domain-slot pair. For example, to represent a domain-slot pair of {*restaurant, area*}, we use a special token $DS_{(restaurant,area)}$. *DS-split*, on the other hand, employs tokens separately for the domain and slot and then merges them into one to represent a domain-slot pair. For {*restaurant, area*}, the domain token $D_{restaurant}$ and the slot token S_{area}. is computed separately and then merged. We use $\{DS\}_{merge}$ and $\{DS\}_{split}$ to represent the special tokens for *DS-merge* or *DS-split*, respectively. Unless it is absolutely necessary to specify whether the tokens are from *DS-merge* or *DS-split*, we'll refer to the DS-produced tokens as $\{DS\}$ tokens, without special distinction, in our descriptions forward. The $\{DS\}$ tokens, after being encoded by the pre-trained language encoder along with the dialogue context, is used to predict its corresponding domain-slot value for a given dialogue context.

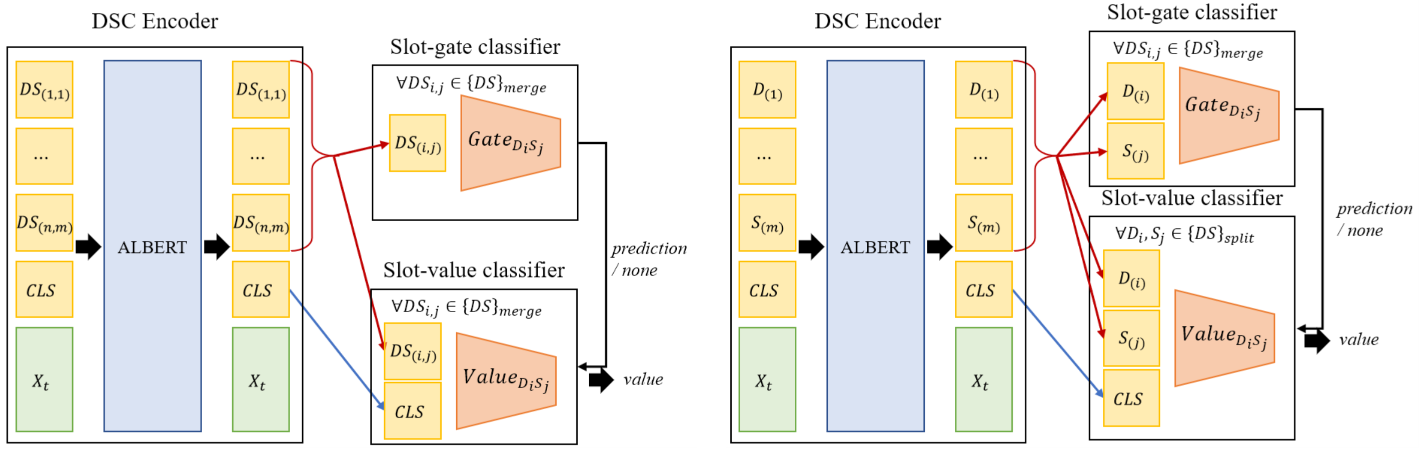
4.2 Proposed Method

Our model is composed of three parts. The first is the domain-slot-context (DSC) encoder, which encodes the dialogue context along with the special tokens representing domain-slot pairs. Next is slot-gate classifier, which is a preliminary classifier that predicts whether each domain-slot pair is relevant to the dialogue context. The adopted the concept of the slot-gate classifier from (Wu et al., 2019a) and made adjustments to apply to our model. The last is the slot value classifier for predicting the value for each domain-slot pair among the candidate values.

In the following descriptions, we assume a dialogue context with a total of T turns. The task is to predict the dialogue state, which are $\{$*domain, slot, value*$\}$ triplets for all domain-slot pairs, for every turn $t = 1, \cdots, T$, using the dialogue context until each turn. Section 4.2 show the overview of our proposed model.

4.2.1 Domain-Slot-Context Encoder

The main structure of our model is the DSC encoder, which uses a pre-trained language to encode the dialogue context along with $\{DS\}$ tokens. For the pre-trained language encoder, we used ALBERT (Lan et al., 2019) due to its strong performance on numerous natural language understanding tasks while having fewer parameters compared to other BERT-style encoders. $\{DS\}$ tokens work like the $[CLS]$ token for BERT, encoding information corresponding to its domain-slot pair (*DS-merge*) or domain and slot (*DS-split*). The set of special tokens for each layout are shown in Eq. (4.3) and Eq. (4.4), respectively. In *DS-merge*, we used special tokens for each individual domain-slot pair. If there are many domain-slot pairs, using this layout can increase the number of special tokens as each domain-slot pair requires a separate special token. In *DS-split*, we used separate tokens for the domain and slot. To represent a domain-slot pair, we merged the corresponding tokens from each domain and slot by concatenating them. This promotes modeling compositionality, since the same slot token can be used for different domains. These $\{DS\}$ tokens and the dialogue context are processed through the DSC encoder, which results in each token in $\{DS\}$ being encoded with contextualized representations according to its domain and slot.



(a) *DS-merge* (b) *DS-split*

Figure 4.2: Model overview.

$$\{DS\}_{merge} = \{DS_{(domain_{(1)},slot_{(1)})}, \cdots, DS_{(domain_{(n)},slot_{(m)})}\} \tag{4.3}$$

$$\{DS\}_{split} = \{D_{domain_{(1)}}, \cdots, D_{domain_{(n)}}, S_{slot_{(1)}}, \cdots, S_{slot_{(m)}}\} \tag{4.4}$$

Fig. 4.3 shows the input representation of the DSC encoder. The sequence begins with $\{DS\}$ tokens. The special token $[CLS]$ follows, which encodes the overall information of the dialogue context. For the dialogue context, we added a special token $[SEP_u]$ to separate each user or system utterance, which is added at the end of each utterance from the user or system. The input ends with a special token $[SEP]$ as the end-of-sequence token.

4 types of embeddings are summed up to represent each token embedding. We used the pre-trained word embedding of ALBERT, except for the $\{DS\}$ tokens, which are randomly initialized. We introduced the token type embedding to differentiate the $\{DS\}$ tokens, user utterances tokens, and system utterances tokens. For *DS-merge*, we used a single token type embedding to represent a domain-slot pair, whereas for *DS-split*, we used two token type embeddings, one for the domain and the other for the slot. We did not apply this embedding for the $[CLS]$ token. Position embeddings are also employed from ALBERT, but the index of the positional embedding starts from the $[CLS]$ token. We did not use the positional embedding for the $\{DS\}$ tokens as the order within those tokens is meaningless. Lastly, the segment embedding from ALBERT was used to represent the whole sequence as a single segment, which is the default segment embedding of ALBERT.

DSC encoder encodes contextualized embeddings for every input token. However, for the slot-gate classifier and slot-value classifier, we only use the special token

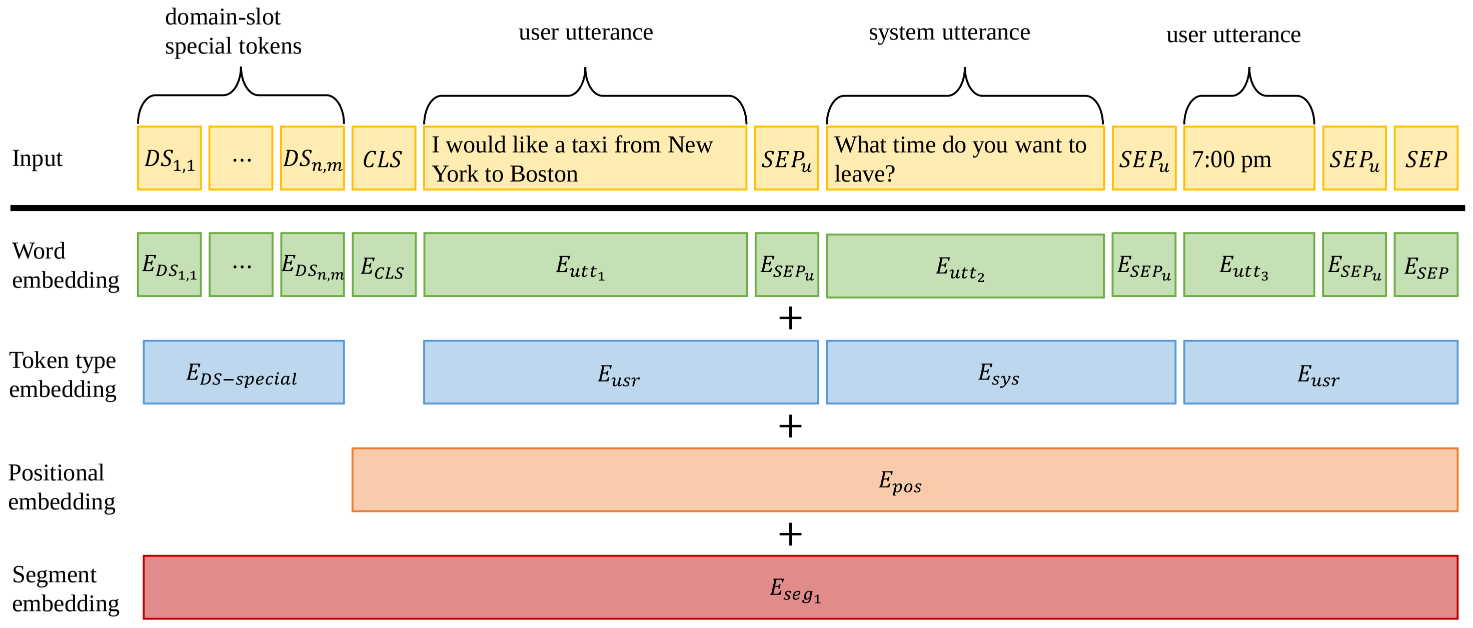


Figure 4.3: Input representation for the DSC encoder. The example here shows a dialogue context for 2 turns of (S^1, U^1, S^2, U^2). S^1 is omitted because the sequence starts with the user utterance and S^1 is just a placeholder: a blank sentence. In this figure, the special token layout for $\{DS\}$ tokens is represented in *DS-merge*. E_{utt_n} represents the sequence of word embeddings for each utterance after tokenization.

outputs of the DSC encoder ($[CLS]$ token and $\{DS\}$ tokens). This is formally defined as follows for *DS-merge* and *DS-split*, respectively, for turn t:

$$\widehat{DS_{(1,1)}}, \cdots, \widehat{DS_{(n,m)}}, \widehat{CLS} = DSCencoder([\{DS\}_{merge}, CLS, X^t, SEP]), \quad (4.5)$$

$$\widehat{D_1}, \cdots, \widehat{D_n}, \widehat{S_1} \cdots, \widehat{S_m}, \widehat{CLS} = DSCencoder([\{DS\}_{split}, CLS, X^t, SEP]), \quad (4.6)$$

where X^t represents the dialogue context of $(S^1, SEP_u U^1, SEP_u, \cdots, S^t, SEP_u, U^t, SEP_u)$. U^t and S^t represents the utterance for the t^{th} turn for the user and system respectively. The $\{DS\}$ tokens and $[CLS]$ token with the hat notation $\widehat{}$ represents the encoded output of the DSC encoder for those special tokens. They are vectors of $\mathbb{R}^d$, where d is the hidden dimension of ALBERT.

4.2.2 Slot-gate classifier

For the slot-gate classifier, we use the DSC encoder output of the $\{DS\}$ tokens for each domain-slot pair to predict whether it is relevant to the dialogue or not. In previous methods, gating used categories of $\{prediction, dontcare, none\}$, where *prediction* means a slot value is not *dontcare* or *none* and *dontcare* means that the predicted slot value is *dontcare* and *none* means that the domain-slot is non-relevant. The label for slot-gates are made from the slot-values. However, the performance for the *dontcare* category was far inferior to the other two categories, so we dismissed the *dontcare* category and only used $\{prediction, none\}$. In our preliminary models with ALBERT *large-v2*, the prediction and recall for *dontcare* was 48.87% and 17.21%, respectively. The precision and recall for *none* showed 98.91%, 99.45% and *prediction*

96.16%, 94.93%, respectively. In this setting, the *dontcare* category is included in *prediction*. For *DS-merge*, the slot-gate classifier predicts the value using the domain-slot pair special token. For the domain-slot pair of domain i and slot j, the slot-gate classifier output for *DS-merge* is

$$Gate_{D_iS_j} = sigmoid\big(W_{G_{DS_{(i,j)}}} \widehat{DS_{(i,j)}}\big), \tag{4.7}$$

where $W_{G_{D_iS_j}} \in \mathbb{R}^{1\times d}$. For *DS-split*, the slot-gate classifier uses the concatenated output of the corresponding domain and slot token. Similarly, for the same domain-slot pair, the slot-gate classifier output for *DS-split* is

$$Gate_{D_iS_j} = sigmoid\big(W_{G_{(D_i,S_j)}} \big[\widehat{D_i}|\widehat{S_j}\big]\big), \tag{4.8}$$

where $|$ represents concatenation of vectors and $W_{G_{(D_i,S_j)}} \in \mathbb{R}^{1\times 2d}$. The loss objective for the gate classification is as follows.set

$$\mathcal{L}_{gate} = \sum_{(i,j)\in DS} BinaryCrossEntropy\big(y^{gate}_{D_iS_j}, Gate_{D_iS_j}\big), \tag{4.9}$$

where DS refers to the set of all domain-slot pairs and $y^{gate}_{D_iS_j}$ is the binary slot-gate label for domain i and slot j. If the domain-slot is predicted to *none*, the corresponding output of the slot-value classifier is changed into *none* regardless of the prediction of the slot-value classifier.

4.2.3 Slot-value classifier

We employ the fixed-vocabulary based classification method for predicting slot values. As in (Zhang et al., 2019c), the candidate-value list for each domain-slot pair was constructed by using the values from the training dataset, rather than using the incomplete ontology from the dataset. The $[CLS]$ token is concatenated with each token from $\{DS\}$, and used as the input to the slot-value classifier for each domain-slot pair. The slot-value classifier output of domain i and slot j for *DS-merge* is as follows:

$$Value_{D_iS_j} = softmax\big(W_{V_{DS_{(i,j)}}}\,[\widehat{DS_{(i,j)}}|\widehat{CLS}]\big), \tag{4.10}$$

where $W_{V_{DS_{(i,j)}}} \in \mathbb{R}^{n_{D_iS_j}} \times 2d$ and $n_{D_iS_j}$ is the number of candidate values for domain i and slot j. Similarly, for *DS-split*, the slot-value classifier output is

$$Value_{D_iS_j} = softmax\big(W_{V_{(D_i,S_j)}}\,[\widehat{D_i}|\widehat{S_j}|\widehat{CLS}]\big), \tag{4.11}$$

where $W_{V_{(D_i,S_j)}} \in \mathbb{R}^{n_{D_iS_j}} \times 3d$. The loss objective for the slot-value classification is as follows:

$$\mathcal{L}_{value} = \sum_{(i,j)\in DS} CrossEntropy(y^{value}_{D_iS_j}, Value_{D_iS_j}), \tag{4.12}$$

where $y^{value}_{D_iS_j}$ is the label for domain i and slot j.

4.2.4 Total objective function

The DSC encoder, slot-gate classifier and slot-value classifier is jointly trained under the total objective function below.

$$\mathcal{L}_{total} = \mathcal{L}_{gate} + \mathcal{L}_{value} \tag{4.13}$$

4.3 Experimental Results

We evaluate our model using the joint goal accuracy, which considers a model prediction to be correct when the prediction jointly matches the ground truth values for all domain-slot pairs, given a dialogue context.

4.3.1 Dataset

We use the MultiWOZ-2.1 (Eric et al., 2019), which fixed noisy annotations and dialogue utterances of the MultiWOZ 2.0 dataset (Budzianowski et al., 2018). The dataset contains 7 domains and over 10,000 dialogues. We follow the previous studies and use 5 domains (train, restaurant, hotel, taxi, attraction) with 30 domain-slot pairs. The other two domains (police, hospital) have little data and do not appear in the test dataset. For MultiWOZ-2.1, we follow the label cleaning and pre-processing explained in (Wu et al., 2019a) with a few amendments. Wu et al. (2019a) delexicalized domain names but we disabled it. Also, apostrophes were erased from (Wu et al., 2019a) but used it again. We applied these adjustments so that the tokenizers of the pre-trained encoders should work as intended. We confirm that we did not alter the label cleaning of (Wu et al., 2019a).

4.3.2 Experimental Setup

For the pre-trained language encoder, we used ALBERT (Lan et al., 2019) and RoBERTa (Liu et al., 2019b) from HuggingFace (Wolf et al., 2019) in Pytorch (Paszke et al., 2019). We used the *xlarge-v2* version of ALBERT and *large* version of RoBERTa for the main experiment and compare other versions in the analysis section. The optimizer was AdamW (Loshchilov and Hutter, 2018) with a learning rate of $1e^{-5}$ for *ALBERT-xlarge-v2* and *RoBERTa-large* and $5e^{-5}$ for *ALBERT-base-v2*, *ALBERT-large-v2* and *RoBERTa-base*. We applied linear warm-up followed by linear decay for the learning rate. We trained all models with the effective batch size of 32, using gradient accumulation for bigger models. Models were selected based on their joint goal accuracy on the validation data split. Only the training data was used to build the labels for each domain-slot pair. The original ALBERT was pre-trained with a sequence length of up to 512 tokens. However, dialogues that are longer than 512 tokens exists in the data. Usually, the standard procedure for this situation is to truncate the sequence up to 512 tokens and discard the remaining tokens. However, to cover dialogues longer than 512 tokens that are in the dataset, we resized the positional embedding to cover a maximum length of the dialogue. We preserved the original pre-trained position embedding for positions indices up to 512 and randomly initialized the remaining position indices. This method showed better results than limiting the maximum sequence length to 512. We plan to release our code on Github.

4.3.3 Results for the MultiWOZ-2.1 dataset

Table 4.1 shows the joint goal accuracy of our model compared to previous methods. Our models show better performance among models without any additional supervision other than the dialogue context and domain-slot pair labels. Except for the *DS-split* version of ALBERT, the other 3 models surpass the previous best joint goal accuracy on the MultiWOZ-2.1 dataset without extra supervision. ConvBERT-DG + Multi (Mehri et al., 2020) achieves a higher joint goal accuracy, but it involves additional dialogue data. In terms of the layout of $\{DS\}$ tokens, mixed results can be seen. For ALBERT, *DS-merge* shows better results than *DS-split*. However, for RoBERTa, *DS-split* shows better results than *DS-merge*. This shows that in models without enough capacity, the slot-sharing of *DS-split* is not much effective.

4.3.4 Ablation Studies

In this section, we show that relationship modeling among different domain-slot pairs is indeed the key factor of our proposed model by running ablation studies. Also, we compare the effect of the size and type of the pre-trained language encoder in terms of performance.

Relationship modeling among different domain-slot pairs

First, we did not use any $\{DS\}$ tokens and only used the CLS token. Because there are no dedicated special tokens for each domain-slot pair, the performance is very poor as shown in 'None' row in Table 4.2. This shows that our approach to introduce $\{DS\}$ is effective.

Next, to evaluate the effect of relationship modeling among different domain-slot pairs, we blocked the attention among different $\{DS\}$ tokens during the encoding process, which restricts direct interaction among $\{DS\}$ tokens. Table 4.2 shows that without the relationship modeling, our model performance deteriorates. This validates our idea that relationship modeling is the important factor for our approach.

Table 4.1: Results for the test dataset of the Multi-WOZ 2.1. *: extra supervision information is from (Hosseini-Asl et al., 2020).

Model	**Extra Supervision**	**Joint Goal Accuracy**
SGD-baseline (Rastogi et al., 2020)	-	43.4
TRADE (Wu et al., 2019a)	-	46.0
NADST (Le et al., 2019)	sys. action, delex. sys. response	49.04
DSTQA (Zhou and Small, 2019)	knowledge graph*	51.17
DS-DST (Zhang et al., 2019c)	-	51.20
DS-Picklist (Zhang et al., 2019c)	-	53.30
SST (Chen et al., 2020b)	-	55.23
TripPy (Heck et al., 2020)	action decision*	55.3
SimpleTOD (Hosseini-Asl et al., 2020)	-	55.76
ConvBERT-DG + Multi (Mehri et al., 2020)	DialoGLUE (Mehri et al., 2020) data	58.7
Our work *(DS-merge, ALBERT-xlarge-v2)*	-	**56.43**
Our work *(DS-split, ALBERT-xlarge-v2)*	-	54.93
Our work *(DS-merge, RoBERTa-large)*	-	**55.94**
Our work *(DS-split, RoBERTa-large)*	-	**56.91**

Table 4.2: Results for ablation of domain-slot relationship modeling on the test dataset of MultiWOZ 2.1.

pre-trained Language Encoder	$\{DS\}$ **token layout**	**Joint Goal Accuracy**
ALBERT-large-v2	None	46.43
	DS-merge	55.85
	w/o relationship modeling	53.33
	DS-split	54.37
	w/o relationship modeling	53.56

Relationship modeling examples

Example 1

The dialogue below shows a dialog example and Table 4.3 shows dialogue state predictions. The value for {*taxi, departure*} and {*taxi, arrive by*} is not explicitly mentioned in the dialogue context. However, our proposed model correctly predicts the value of those domain-slots, which can be inferred from the dialogue context. The model without relationship modeling fails to predict the correct value for {*taxi, departure*} and {*taxi, arrive by*}.

User: hi, can you please give me information on a restaurant called midsummer house restaurant located in cambridge?

System: this is a british restaurant found in the center of town that is in the expensive price range. what other information do you need?

User: can you book this restaurant for 3 people on wednesday at 15:45?

System: i booked it for you the reference number is 5cwu2u98. is there anything else?

User: i would like to do a little site seeing. do you have any good attractions in the east?

System: there are many in the east,what type are you most interested in?

User: are there museums?

System: may i suggest cambridge artworks? it sounds like just what you are looking for and is free to visit.

User: okay, that sounds great. can you also help me with a taxi in order to get to the restaurant?

Table 4.3: Dialogue state prediction for example 1. The red bold text indicates that the prediction is wrong.

Domain-slot	Proposed method	w/o relationship modeling
attraction-area	east	east
attraction-type	museum	museum
restaurant-book day	wednesday	wednesday
restaurant-book people	3	3
restaurant-book time	15:45	15:45
restaurant-name	midsummer house restaurant	midsummer house restaurant
taxi-arrive by	15:45	**none**
taxi-departure	cambridge artworks	**none**
taxi-destination	midsummer house restaurant	midsummer house restaurant

Example 2

The dialogue below shows another dialog example and Table 4.4 shows dialogue state predictions. The value for $\{$*taxi, destiantion*$\}$ has to be inferred from the dialogue context that it is the same as the $\{$*hotel, name*$\}$. Our proposed model correctly predicts the value while the model without relationship modeling fails.

User: i need a hotel for tonight please.

System: i can help you with that. do you have a particular price point you are looking for?

User: it does not matter but i do need free parking.

System: i have several hotels matching your request available. what part of town would you prefer?

User: i have no preference on the part of town, but i do need free wifi as well.

System: ok, that narrows it down to 8 options for you. any other preferences?

User: i would like it to be a guesthouse.

System: well there are 21 of those spread across town so i recommend the arbury lodge guesthouse on the north side. would you like me to book it?

User: yes can you please?

System: what day would you like me to book the hotel for?

User: i apologize for not mentioning it before, please book it starting sunday for 1 person for 5 days?

System: you are all set for sunday. your reference number is nivmcdk1. is there anything else i can help with?

User: thanks. can you give me some information on all saints church?

System: sure. what sort of information do you require?

User: i just need the entrance fee and postcode, please.

System: admission is free and their post code is cb5 8bs. what else can i help you with today?

User: i also need a taxi that will leave all saint's church by 08:45

System: and where will the cab need to take you?

User: i need it to take me back to the hotel room i just reserved

System: no problem. i have got a taxi booked for you.

User: i am not psychic, so i will need to know what kind of car to look for, also the phone number please.

System: i apologize. the annual psychics convention is next week. they are very particular about being allowed to sense the information. it is a yellow skoda with contact number 07621484267.

User: thank you.

System: you are welcome. is there anything else we can help you with today?

User: thank you for helping me.

Table 4.4: Dialogue state prediction for example 2. The red bold text indicates that the prediction is wrong.

Domain-slot	Proposed method	w/o relationship modeling
hotel-book day	sunday	sunday
hotel-book people	1	1
hotel-book stay	5	5
hotel-internet	yes	yes
hotel-name	arbury lodge guesthouse	arbury lodge guesthouse
hotel-parking	yes	yes
taxi-departure	all saints church	all saints church
taxi-destination	arbury lodge guesthouse	**express by holiday inn cambridge**
taxi-leave at	08:45	08:45

Size and type of the pre-trained language encoder

We compared ALBERT and RoBERTa (Liu et al., 2019b) and various model sizes within those pre-trained language encoders. Table 4.5 shows the result for different versions of the pre-trained language encoders. For ALBERT, a bigger language model

Table 4.5: Results for different ALBERT configurations on the evaluation test dataset of MultiWOZ 2.1.

$\{DS\}$ **token layout**	**pre-trained Language Encoder**	**Joint Goal Accuracy**
DS-merge	*ALBERT-base-v2*	54.38
	ALBERT-large-v2	55.85
	ALBERT-xlarge-v2	56.43
	RoBERTa-base	54.82
	RoBERTa-large	55.94
DS-split	*ALBERT-base-v2*	53.28
	ALBERT-large-v2	54.37
	ALBERT-xlarge-v2	54.93
	RoBERTa-base	55.86
	RoBERTa-large	56.91

shows better results as is shown in various downstream tasks that ALBERT was evaluated on (Lan et al., 2019). Except for *ALBERT-xx-large*, all other configurations show that *DS-merge* shows better performance than *DS-split*. Based on the drastic increase in performance with *xx-large*, we presume that the high model complexity of *ALBERT-xx-large* enabled $\{DS\}_{split}$ tokens to effectively encode information and make slot-sharing to work. In smaller models, this slot-sharing might not have been as effective due to their smaller encoding capacity. Also, concatenation, which was used for merging domain and slot embeddings in *DS-split*, might not have been enough for fully representing the information for the domain-slot pair in smaller models. RoBERTa also shows similar results with bigger models showing stronger performance.

4.4 Chapter Summary

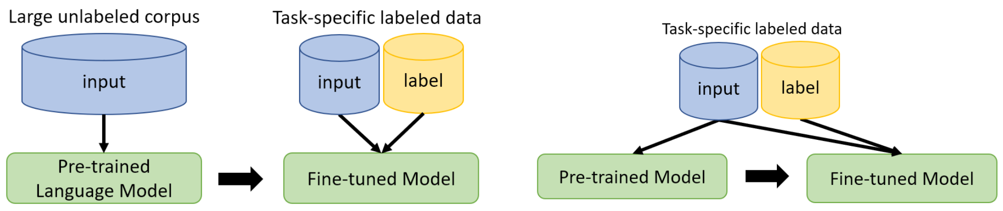
In this chapter, we propose a model for multi-domain dialogue state tracking that effectively models the relationship among domain-slot pairs using a pre-trained language encoder. We introduced two methods to represent special tokens for each domain-slot pair: *DS-merge* and *DS-split*. These tokens work like the [CLS] token for BERT, encoding information corresponding to its domain-slot pair (*DS-merge*) or domain and slot (*DS-split*). These special tokens are run together with the dialogue context through the pre-trained language encoder, which enables modeling the relationship among different domain-slot pairs. Experimental results show that our model achieves state-of-the-art performance on the MultiWOZ-2.1 dataset among models without extra supervision. The ablation experiments show that the relationship modeling among different domain-slot pairs is the key element of our model. Also, we showed that larger pre-trained language encoders improves performance.

Chapter 5

Pre-training of Transformers with Question-Answer Masked Language Modeling for Mathematical Question Answering

5.1 Background

Pre-trained language models (Devlin et al., 2019; Radford et al., 2018) based on Transformers (Vaswani et al., 2017) has achieved state of the art results in various natural language processing (NLP) tasks. Instead of training the model from scratch on a handful of labeled data for downstream tasks, these models are trained in two phases. First, the models are pre-trained with various self-supervised learning objectives on a large corpus of natural language text. Next, the model is initialized with the pre-trained weights and fine-tuned on the labeled data. This process is shown in Fig. 5.1a. It is important that the pre-training phase does not require labeled data, which makes it possible to use a huge amount of raw text data collected from various corpora. In the case of BERT (Devlin et al., 2019), masked language modeling (MLM), also referred to as the cloze task (Taylor, 1953), is the main self-supervised learning objective. MLM randomly masks words in a sentence and trains the model to predict those masked tokens in the context of other words in the sentence. Because masks can be made on the fly, data for MLM can be generated from any text available, which can be acquired by web-scraping documents. Models with improvements on BERT has been studied extensively on various NLP tasks (Liu et al., 2019b; Yang et al., 2019b; Conneau and Lample, 2019; Lan et al., 2019; Clark et al., 2019).



(a) Natural Language (b) Mathematical question answering

Figure 5.1: Pre-train and Fine-tune Framework.

Understanding natural language requires a series of reasoning steps. To understand the meaning of a sentence, one has to begin with recognizing words from a sequence of characters. Next, the meaning of each words has to be identified. Last, the meaning of the words should be understood in the context of the whole sentence. Understanding and solving mathematical problems also follow a similar reasoning process.

For example, consider the following mathematical problem.

Q: What is $g(f(a))$, where $f(a) = a + 3, g(a) = 2a - 4$?

Answer: $2a + 2$

To understand and solve such a problem, one has to parse the sequence of characters into tokens of words, numbers, operators and variables. Next, the meaning of each token has to be identified. Knowing that the symbol "$-$" means subtraction would be an example. After that, the meaning of the tokens has to be understood in context. That is, the relationship among the tokens has to be put into consideration to know the meaning of each token. For example, even with the same token a, one needs to understand that a of $f(a)$ corresponds to the a in $a + 3$, and that a of $g(a)$ corresponds to the a in $2a - 4$. Also, a could be used as a variable, as in the case above, or be used as an indefinite article. After such context is expanded to the whole question, one can understand the meaning of the problem and solve it.

Because of these similarities, we propose to apply the pre-training methods of natural language to mathematical question answering. In this work, we assume the questions and answers for mathematical question answering are given in free-form texts, which are recognized as a sequence of characters.

MLM pre-training improves mere supervised learning because it helps the model to learn the initial representations of words, which is the building block of natural language sentences. Because the size of the vocabulary is so huge, using only supervised learning makes it easy for models to overfit. The vocabulary of math is much smaller than that of natural language. However, we assumed that pre-training and can still help solve mathematical question answering because it can help the model recognize the syntax of how mathematical equations are formed. For example, by pre-training, the model can learn that parentheses are always used in pairs, so that an open parenthesis should have its corresponding close parenthesis. Another structure that could be learned is that numbers or variables should be placed between math operators for the equation to form a legitimate mathematical equation.

However, the pre-training method used in natural language tasks cannot be directly applied to mathematical question answering. Pre-training of natural language are based on a continuous stream of text from large documents. On the other hand, data in mathematical question answering are given in pairs of questions and answers. Pre-training natural language requires an abundant amount of unlabeled text data that is much bigger in size than the labeled data of the downstream task. It is much harder to acquire mathematical question answering data than acquiring random texts from the web. Because of this, we need to use the same labeled data for pre-training as also, as shown in Fig. 5.1b. These points call for adjustments of the pre-training methods of NLP if it is to be applied to mathematical question answering.

5.2 Proposed Method

In this section, we explain the proposed pre-training method for mathematical question answering. Next, we explain the fine-tuning process. Note that different from most NLP tasks, where word-level tokenization is used for text, we use character-level tokenization, to compare our work with previous studies on the Mathematics dataset.

5.2.1 Pre-training: Question-Answer Masked Language Modeling

In BERT (Devlin et al., 2019), masked language modeling was used on consecutive text streams from documents. However, our mathematical problem consists of a question and answer pair. To leverage both question and answer texts, we concatenate the question and answer as a single text. Masked language modeling is performed on this concatenated text. The concatenated text is parsed using character-level tokenization. A portion of the question-answer tokens is masked with a special mask token. The Transformer encoder is trained to predict the original id of the masked token based on the context of other tokens. The Transformer encoder can refer to neighboring tokens in the same segment to find syntactic rules such as a missing letter in a word. If a masked token cannot find what the original token is by looking at the question tokens, it can also refer to the answer tokens. The opposite is also possible. We call this pre-training method Question-Answer Masked Language Modeling (QA-MLM). Before concatenating the question and answer tokens, a special Start-of-Sequence token ($[SOS]$) and End-of-Sequence ($[EOS]$) are attached to the beginning and end of the answer tokens respectively. Fig. 5.2 shows the framework of QA-MLM.

Following BERT (Devlin et al., 2019), we mask 15% of the question-answer concatenated tokens for each question-answer pair. The chosen tokens are replaced with (1) the [MASK] token 80$ of the time (2) a random token 10% of the time (3) the unchanged original token 10% of the time.

Next sentence prediction (NSP) is another self-supervised objective of BERT. However, follow-up studies of BERT (Yang et al., 2019b; Liu et al., 2019b; Lan et al., 2019; Clark et al., 2019) removed NSP as it is shown to be ineffective. Following their result, we drop NSP as our self-supervised objective.

The input representation for pre-training is given in Fig. 5.3. The input representation of each token consists of token embedding, segment embedding and positional embedding. Segment embedding is used to differentiate the question tokens from the answer tokens. Notice that the positional embeddings are reset when the answer segment begins. All embeddings are summed up to form the input representation for each token, which is given as the input to the Transformer.

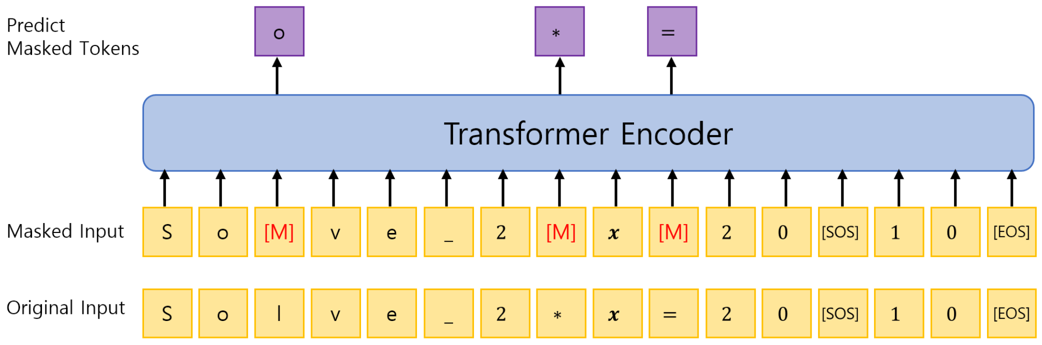

Predict
Masked Tokens
o
*
=
Transformer Encoder
Masked Input
S o [M] v e _ 2 [M] x [M] 2 0 [SOS] 1 0 [EOS]
Original Input
S o l v e _ 2 * x = 2 0 [SOS] 1 0 [EOS]

Figure 5.2: Framework of Question-Answer Masked Language Modeling. The token "[M]" represents the [MASK] token. The token "_" represents a white space.

5.2.2 Fine-tuning: Mathematical Question Answering

For the end task of mathematical question answering, the encoder and decoder of the Transformer are both needed. However, the pre-training procedure only trains the weight of the encoder part of the Transformer. For the initialization of the decoder, we try xavier uniform initialization or copying the weights from the pre-trained encoder. For the latter case, the encoder-decoder attention weights are randomly initialized while other weights are copied from the pre-trained encoder. However, this does not mean that the weights are tied between the encoder and decoder. Only the initialization values for the weights are the same.

The same input representation of token, segment and position embeddings from pre-training are used for fine-tuning. The different part is that the encoder now only receives question tokens as input and that the decoder only receives answer tokens as input. Simple greedy decoding was used for prediction. Fig. 5.4 shows the Transformer used for fine-tuning.

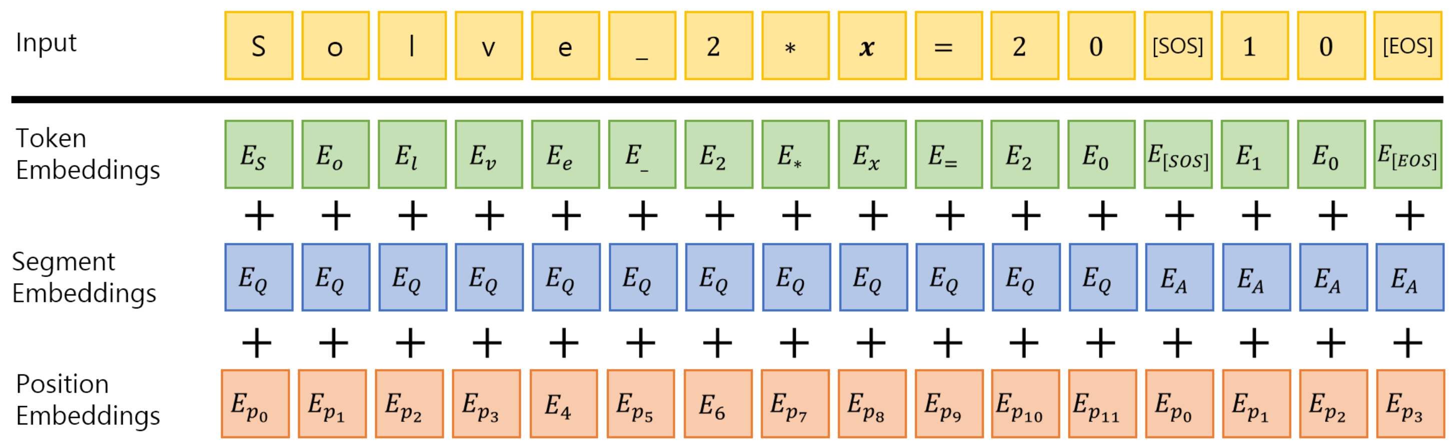

Input
S o l v e _ 2 * x = 2 0 [SOS] 1 0 [EOS]
Token Embeddings
E_S E_o E_l E_v E_e $E_\_$ E_2 E_* E_x $E_=$ E_2 E_0 $E_{[SOS]}$ E_1 E_0 $E_{[EOS]}$
Segment Embeddings
E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_Q E_A E_A E_A E_A
Position Embeddings
E_{p_0} E_{p_1} E_{p_2} E_{p_3} E_4 E_{p_5} E_6 E_{p_7} E_{p_8} E_{p_9} $E_{p_{10}}$ $E_{p_{11}}$ E_{p_0} E_{p_1} E_{p_2} E_{p_3}

Figure 5.3: Input Representation for Question-Answer Masked Language Modeling. Tokenization is done in word-level for simplicity. E_Q and E_A represent Question and Answer segment respectively. E_{p_n} represents the positional embedding for index n.

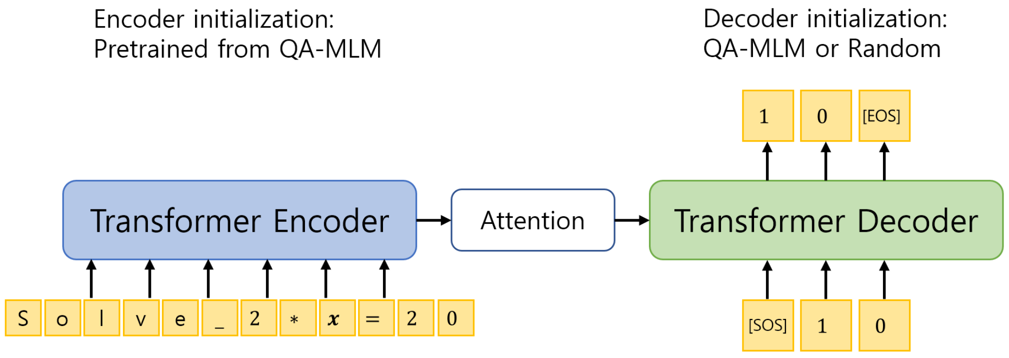


Figure 5.4: Transformer for Mathematical Question Answering

5.3 Experimental Results

In this section, we explain the dataset that is used to evaluate the effect of pre-training and show the results on the dataset.

5.3.1 Dataset

The Mathematics Dataset (Saxton et al., 2019) is a dataset that consists of 56 different types of mathematics problems such as algebra, calculus, arithmetic, probability, and measurement. Each problem is structured as a free-form text of question and answer as shown in Fig. 5.5. For the training data split, there are 2×10^6 problems given for each problem type, which is split into equal amount of data according to their difficulty of easy, medium and hard. The test data split also has the same number of problem types with 5×10^3 problems for each question type. The test data is split into interpolation set and extrapolation set. The interpolation set has similar characteristics with the training dataset. The extrapolation set contains larger numbers and more complex compositions than the training data, which is used to test generalization.

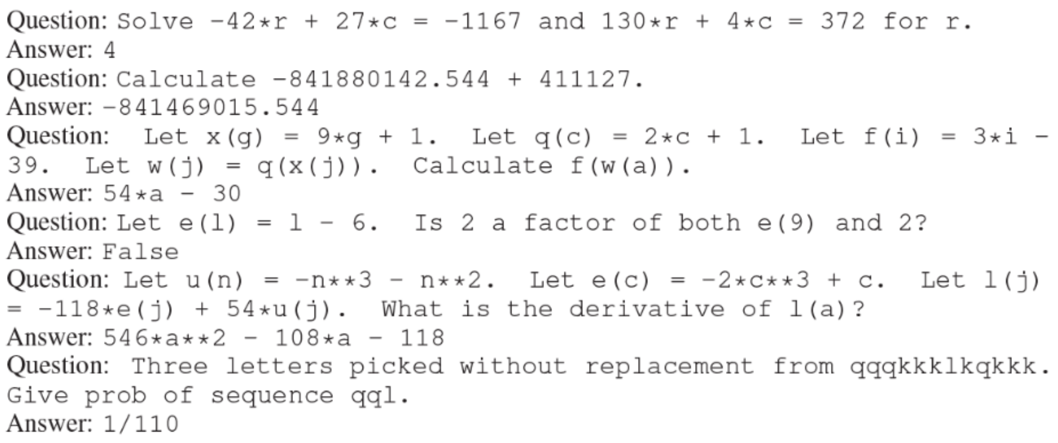

Question: Solve -42*r + 27*c = -1167 and 130*r + 4*c = 372 for r.
Answer: 4
Question: Calculate -841880142.544 + 411127.
Answer: -841469015.544
Question: Let x(g) = 9*g + 1. Let q(c) = 2*c + 1. Let f(i) = 3*i - 39. Let w(j) = q(x(j)). Calculate f(w(a)).
Answer: 54*a - 30
Question: Let e(l) = l - 6. Is 2 a factor of both e(9) and 2?
Answer: False
Question: Let u(n) = -n**3 - n**2. Let e(c) = -2*c**3 + c. Let l(j) = -118*e(j) + 54*u(j). What is the derivative of l(a)?
Answer: 546*a**2 - 108*a - 118
Question: Three letters picked without replacement from qqqkkklkqkkk. Give prob of sequence qql.
Answer: 1/110

Figure 5.5: Example of problems from the Mathematics Dataset. Figure from (Saxton et al., 2019)

5.3.2 Experimental Setup

For the input, character-level tokenization was used for the question and answer text. For the model, a 4-layer, 8-head Transformer with an input dimension of 512 and feedforward dimension of 2048 was used. The Adam optimizer (Kingma and Ba, 2014) with a learning rate of $1e-4$ and batch size of 1024 was used. We chose this configuration to follow the experimental setup of (Saxton et al., 2019) to compare and assess the impact of pre-training with question-answer MLM. All other configurations are the same of (Saxton et al., 2019) except for the learning rate and the way positional information is encoded. As for the learning rate, we could not replicate the results of (Saxton et al., 2019) with the learning rate of $6e-4$, which was used for their work. We were able to train the model with a learning rate of $1e-4$, which was used for all training procedures. As for the positional information, since the original Transformer (Vaswani et al., 2017) uses positional encodings over positional embeddings, it is assumed that (Saxton et al., 2019) used positional encodings. However, our implementation showed that models with positional encodings show slightly worse performance compared to their counterparts with positional embeddings. Thus, we use positional embeddings for our experiments. Pre-training was run for 500k steps before being reused for fine-tuning. For the models trained from scratch, we used xavier uniform initialization.

5.3.3 Experimental Results on the Mathematics dataset

In this section, we first show the results for pre-training and then show the results for fine-tuning.

Pre-training results

MLM accuracy for each token type is shown in table Section 5.3.3. Alphabets are mainly used in words or variables. Symbols refer to non-alphanumeric tokens. Alphabets and symbols are predicted with a very high accuracy. This indicates that during pre-training, the model seems to be learning the semantic structure of mathematical equations. Number tokens show the least accuracy which is understandable since it requires numerical reasoning. Also, if other tokens are masked as well, the masked number token could be replaced with another token without harming the plausibility of the question and answer. For example, in the case where the question is SOLVE $x + 3 = 2$ and answer is -1, if the token 3 and 2 are all masked, any number pairs that satisfy the answer can be replaced with the original tokens, like $4, 3$ or $5, 4$. Because of this ambiguity that masked numbers generate, we also trained the QA-MLM with only masking non-numeric tokens. However, this model showed worse results for fine-tuning. Based on this finding, we argue that even though the prediction of masked number tokens raises some ambiguity in its effectiveness, there exists some syntax information that it can provide for the model to learn from.

Dataset	Number	Alphabet	Symbols
Interpolate	59.88	98.14	98.04
Extrapolate	55.22	94.62	97.96

Table 5.1: Masked language model accuracy.

Finetuning results

In this section, we assess the effect of pre-training on the Mathematics Dataset. We report average exact-match accuracy following (Saxton et al., 2019). Exact-match accuracy is scored by checking if the predicted answer matches the correct answer character-for-character. Section 5.3.3 shows the result of fine-tuning on the Mathematics Dataset.

We could not reproduce the results of (Saxton et al., 2019), using the training details that they provided. We could not run the exact code as (Saxton et al., 2019) since they did not provide the code. As an alternative, we implemented the Transformer using the configuration from (Saxton et al., 2019).

It is clear that pre-training the Transformer with QA-MLM shows better results than the model trained from scratch. In the same number of steps, pre-trained models achieve higher performance in both Interpolation and Extrapolation test datasets. Running QA-MLM takes roughly 60% of the computation cost of fine-tuning for the same number of steps. Even putting this into account, the model trained from scratch for 1M steps shows worse performance than the pre-trained models trained for 500k steps, which is roughly the same computation cost of fine-tuning for 850k steps. Also, to train the model from scratch until achieving the accuracy of the pre-trained model, it needs to be trained for 1.6M steps, which is computationally less efficient. Thus, it can be said that pre-training with QA-MLM is not only better in performance, but computationally efficient than training the model from scratch.

For decoder initialization, a randomly initialized decoder performed almost the same as the the decoder initialized from QA-MLM. This may be due to the fact that discrepancy between pre-training of the decoder and the fine-tuning of the decoder. The pre-trained weights of the encoder that are used to initialize the decoder was trained in a bidirectional manner, while the decoder works in an auto-regressive manner.

	Train steps	Interpolation	Extrapolation
Transformer (Reported from (Saxton et al., 2019))	500k	76.00	50.00
Transformer (Our implementation of (Saxton et al., 2019))	500k	69.16	39.43
	1M	73.26	46.25
	1.6M	77.32	49.42
Transformer w/ QA-MLM ENC	500k	**73.28**	**46.32**
	1M	**77.21**	**50.60**
	1.6M	**78.28**	**51.23**
Transformer w/ QA-MLM ENC, DEC	500k	**73.98**	**46.49**
	1M	**77.34**	**50.66**
	1.6M	**78.31**	**51.26**

Table 5.2: Average Exact-Match accuracy for the Mathematics Dataset. ENC represents that only the encoder of the Transformer is initialized with pre-trained weights of QA-MLM. ENC, DEC represents that the both components of the Transformer are initialized with pre-trained weights of QA-MLM.

Effect of Pre-training in terms of Problem Types

To further investigate the effect of pre-training, we show the learning curves for the course of training for different question types in Fig. 5.6. The pre-trained model that we compare with the model trained from scratch is the model with both the encoder and decoder initialized from the pre-trained weights. The first row of Fig. 5.6 shows problem types, which are simple compared to other problem types, where both models learn fast. In the second row, examples of problem types where the pre-trained model shows better accuracy in the early stage than the model trained from scratch, but is caught up during the course of learning. These question types have more structure than the simple ones mentioned above. Lastly, the third row show examples of problem types where the pre-trained model consistently shows but However, the pre-trained model learns faster than the non-pretrained model for more complex arithmetic problems with different types of operations. For harder questions involving algebra, the pre-trained model shows better performance throughout the course of learning. Further examples of each question types are show in the Appendix.

Examples of the Mathematics dataset

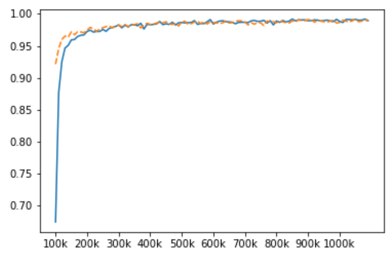
We show examples for the problem types of the Mathematics dataset that are mentioned above. For each question type, three examples of question-answer pairs are shown.

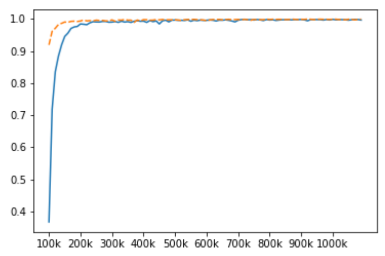
comparison__closest

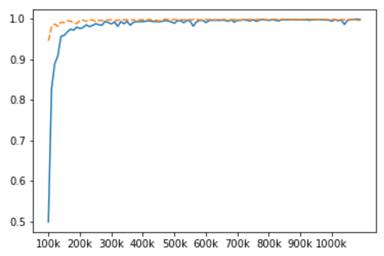
```
Q: Which is the closest to -1/3? (a) -8/7 (b) 5 (c) -1.3
A: a
Q: Which is the nearest to 27/5? (a) 0.4 (b) 0.2 (c) -0.5 (d) -0.1
A: a
Q: Which is the closest to -4/17? (a) 4/7 (b) -42/11 (c) 2
A: a
```

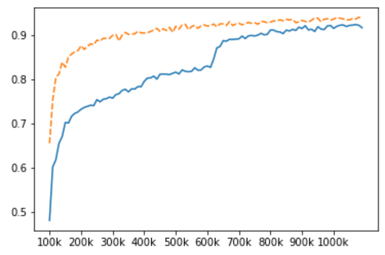
arithmetic__add_or_sub

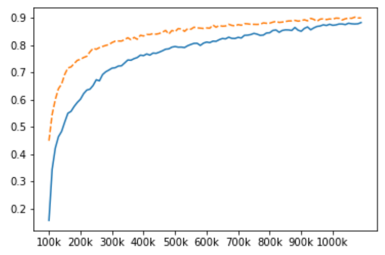
```
Q: What is -5 - 110911?
A: -110916
Q: What is -0.188 + -0.814?
A: -1.002
```

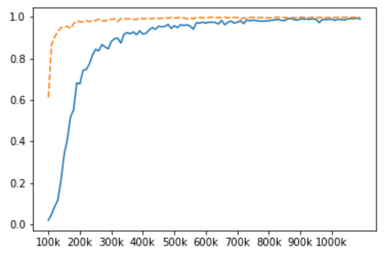

(a) compare_closest

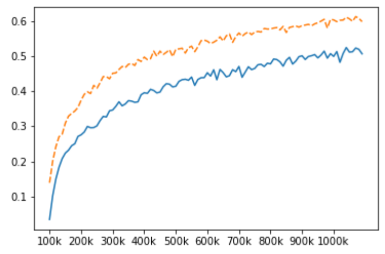

(b) arithmetic__add_or_sub

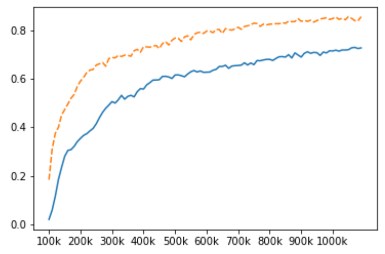

(c) calculus__differentiate


(d) compare_closest_composed


(e) calculus_diff_composed


(f) arithmetic_add_sub_multiple


(g) algebra__polynomial_roots


(h) algebra__sequence_nth_term

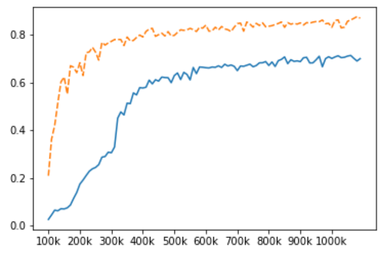

(i) probability__swr_p_sequence

Figure 5.6: Learning curve for fine-tuning for selected question types. The orange, dashed line indicates the pre-trained model with encoder and decoder initialization. The blue, straight line indicates the non-pretrained model. The x axis indicates training steps. The y axis indicates exact-match accuracy.

Q: Sum 259 and -46.
A: 213

calculus__differentiate

Q: Find the first derivative of 2*d**4 - 35*d**2 - 695 wrt d.
A: 8*d**3 - 70*d
Q: Find the third derivative of -a**3*g**3*t**3 + 642*a**3*g*t**3 + 16*a**3*g*t**2 - 5*a**2*t**2 + a*g**3 wrt t.
A: -6*a**3*g**3 + 3852*a**3*g
Q: What is the second derivative of 12518*f**3 + 3760*f?
A: 75108*f

comparison__closest_composed

Q: Let q = -54.3 + 54. Suppose 0 = -5*z - 8 - 7. Which is the nearest to -1/5? (a) 5 (b) z (c) q
A: c
Q: Let d(j) = -j**3 - 5*j**2 - 4*j + 1. Let n be d(-4). Suppose -5*h = 2*i - 2*h + n, 0 = i + 5*h - 10. What is the nearest to 0 in 1/3, i, -2?
A: 1/3
Q: Let f = -2.31 + 0.31. What is the nearest to f in 0.3, -2, 0.2?
A: -2

calculus__differentiate_composed

Q: Let h(t) = t**3 + t**2 + 1. Let v(d) = 6*d**3 + 24*d**2 + 4. Let w(j) = 4*h(j) - v(j). What is the third derivative of w(x) wrt x?
A: -12
Q: Let v = -7 - -12. Suppose 0 = 2*h - 3*x - 16 - 5, 0 = -v*h + 3*x + 30. What is the first derivative of 5*t - h - t + 0 - 2*t wrt t?
A: 2
Q: Let b(y) be the second derivative of -3*y**8/56 - y**4/6 - y. What is the third derivative of b(o) wrt o?
A: -360*o**3

arithmetic__add_sub_multiple

Q: What is 1 + -9 - -5 - -1?
A: -2
Q: -2 + 0 + (3 - 1)
A: 0
Q: Calculate 8 - (0 + 7 + -4).
A: 5

algebra__polynomial_roots

Q: Solve -3*h**2/2 - 24*h - 45/2 = 0 for h.
A: -15, -1

```
Q: Factor -n**2/3 - 25*n - 536/3.
A: -(n + 8)*(n + 67)/3
Q: Let c**3/9 - 11*c**2/3 + 35*c - 75 = 0. What is c?
A: 3, 15
```

algebra__sequence_nth_term

```
Q: What is the k’th term of 485, 472, 459, 446?
A: -13*k + 498
Q: What is the c’th term of 362, 746, 1144, 1562, 2006, 2482, 2996?
A: c**3 + c**2 + 374*c - 14
Q: What is the b’th term of 178, 367, 566, 775, 994?
A: 5*b**2 + 174*b - 1
```

probability__swr_p_sequence

```
Q: What is prob of sequence ccbc when four letters picked without
replacement from nnscspb?
A: 0
Q: Three letters picked without replacement from
{g: 3, w: 1, t: 7, u: 3}. Give prob of sequence tuw.
A: 1/104
Q: Three letters picked without replacement from dxaxxaaxxxaax.
What is prob of sequence aad?
A: 5/429
```

5.4 Chapter Summary

In this chapter, we proposed a pre-training method for mathematical question answering that shows better results and efficiency than mere fine-tuning. We believe that this is the first approach to use pre-training methods of Transformers for mathematical reasoning. Our pre-training method, Question-Answer Masked Language Modeling (QA-MLM) uses both question and answer text for masked language modeling. In QA-MLM, masked tokens in the question can be inferred by referring to the neighboring tokens in the same segment or to the tokens in the answer. The opposite is also possible.

Models initialized with the pre-trained model not only showed improved results on the Mathematics dataset (Saxton et al., 2019) over the non-pretrained model, but achieved the result with higher computational efficiency. The improvement mainly came from pre-training the encoder.

To analyze the type of problems in which pre-training impacted the most, we compared the learning curve of the pre-trained model with that of the model trained from scratch. For simple question types, both models learn fast, while the pre-trained model started with a higher starting point. For some harder question types, the pre-trained model showed higher performance in the early stage of learning, but both models later converged. For other harder question types, the pre-trained model consistently outperformed the non-pretrained model throughout the learning curve.

Chapter 6

Conclusion

6.1 Contributions

To achieve human-like intelligence, it is important to solve tasks that require complex reasoning. These complex reasoning tasks typically require multiple and hierarchical reasoning steps. In an attempt to solve complex reasoning tasks that require such capacities, we solved tasks in visual, conversational, and mathematical reasoning. The Transformer architecture, which is used mainly in the natural language processing field, was chosen as the main frame to be applied to these various tasks. In this dissertation, we proposed three types of novel architectures based on the Transformer encoder for visual IQ tests, dialogue state tracking and mathematical question answering. Each data type has its own characteristics, which demands appropriate adjustments to the original Transformer encoder structure.

First, we propose using a hierarchical Transformer encoder with structured representation that employs a novel neural network architecture to improve both perception and reasoning in a visual IQ test. For perception, we used object detection models to extract the structured features. For reasoning, we used the Transformer encoder in a hierarchical manner that fits the structure of Raven's Progressive Matrices. Experimental results on the RAVEN dataset, which is one of the major large-scale datasets on Raven's Progressive Matrices, show that our proposed architecture achieved state-of-the-art performance.

Next, we propose a dialogue state tracking model using a pre-trained language model, which is a pre-trained Transformer encoder, for domain-slot relationship modeling. Dialogue state tracking for multi-domain dialogues is challenging because the model should be able to track dialogue states across multiple domains and slots. Past studies had its limitations in that they did not factor in the relationship among different domain-slot pairs. Although recent approaches did support relationship modeling among the domain-slot pairs, they did not leverage a pre-trained language model, which has improved the performance of numerous natural language tasks, in the encoding process. Our approach fills the gap between these previous studies. We propose a model for multi-domain dialogue state tracking that effectively models the relationship among domain-slot pairs using a pre-trained language encoder. Inspired by the way the special $[CLS]$ token in BERT is used to aggregate the information of the whole sequence, we use multiple special tokens for each domain-slot pair that encodes information corresponding to its domain and slot. The special tokens are run together with the dialogue context through the pre-trained language encoder, which effectively models the relationship among different domain-slot pairs. Our experimental results show that our model achieves state-of-the-art performance on the MultiWOZ-2.1 dataset among models without extra supervision.

Finally, we propose a method to pre-train a Transformer encoder on a mathematical question answering dataset for improved performance. Pre-trained language models have achieved state-of-the-art results in many natural language processing tasks. Understanding mathematical problems has similarities with understanding natural language, in that it requires similar reasoning steps, such as recognizing words or symbols from a sequence of characters and identifying the meaning of those words and symbols in the context of the whole sequence. Thus, we propose a pre-training method for mathematical question answering. We believe that this is the first approach to use pre-training methods of Transformers for mathematical question answering. Our pre-training method, Question-Answer Masked Language Modeling, leverages both question and answer text by concatenating them into a single text. Models initialized with the pre-trained model not only showed improved results over the non-pretrained model, but also achieved the result with higher computational efficiency.

6.2 Future Work

Our work is focused on the Transformer encoder. However, the original Transformer is composed of two parts: encoder and decoder. As we have developed new architectures of the Transformer encoder, enhancements to the Transformer decoders can also be applied. As generative models are becoming more popular these days, research on the development of novel architectures for the Transformer decoder is a promising research topic.

Also, our work also involves pre-trained language encoders such as BERT, ALBERT and RoBERTa. It is reported that BERT shows social, racial and gender bias present in human-generated data (Tan and Celis, 2019; Bhardwaj et al., 2020; Kurita et al., 2019). This is an important in terms of legal issues when implementing services with BERT, which can be damaging if the model showed any kind of discrimination regarding social, racial and gender issues. It is also a humanitarian issue since advanced deep learning models are spreading and aggravating this bias. In our second work, we use pre-trained language models to detect dialogue states between a system and a user. In our future work, we will find methods to overcome this bias to improve quality of our model and ensure that our model is unbiased in terms of social, racial and gender issues. This will involve filtering the dataset as well as finding training methods that can avoid the model to learn such biases.

Now, we provide future work with respect to our main research of the dissertation.

For the visual IQ test, we can use the PGM dataset (Santoro et al., 2018) to test the generalizability of our model. Because PGM does not have annotations for training the object detection model, how to obtain structured representations is an issue to be addressed. In addition, we can adapt our reasoning module to cover other reasoning problems that do not involve visual elements but still deals with reasoning among different elements.

For dialogue state tracking, we hope to advance our current apporach by finding ways to effectively apply our model towards the open-vocabulary approach, which will enable better generalization for candidate values that are outside of the training data.

Lastly, for mathematical question answering, our model is limited in that only the masked language modeling task is used for the pre-training task. Training the model with a sophisticated pre-training task that are made to incorporate explicit mathematical reasoning would be a promising research subject for the future.

Bibliography

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

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

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013.

R. Bhardwaj, N. Majumder, and S. Poria. Investigating gender bias in bert. *arXiv preprint arXiv:2009.05021*, 2020.

W. B. Bilker, J. A. Hansen, C. M. Brensinger, J. Richard, R. E. Gur, and R. C. Gur. Development of abbreviated nine-item forms of the raven's standard progressive matrices test. *Assessment*, 19(3):354–369, 2012.

G. H. Bower. A contrast effect in differential conditioning. *Journal of Experimental Psychology*, 62(2):196, 1961.

P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, 2018.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

P. A. Carpenter, M. A. Just, and P. Shell. What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test. *Psychological review*, 97(3):404, 1990.

G.-L. Chao and I. Lane. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*, 2019.

C. Chen, D. Han, and J. Wang. Multimodal encoder-decoder attention networks for visual question answering. *IEEE Access*, 8:35662–35671, 2020a.

J.-H. Chen, Y.-H. Hao, H. Wang, T. Wang, and D.-W. Zheng. Futures price prediction modeling and decision-making based on dbn deep learning. *Intelligent Data Analysis*, 23(S1):53–65, 2019a.

L. Chen, B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *AAAI*, pages 7521–7528, 2020b.

Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019b.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

N. Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 2014.

S. Cirillo and V. Ström. An anthropomorphic solver for raven's progressive matrices. Master's thesis, 2010.

K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019.

A. Conneau and G. Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069, 2019.

B. C. Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

S. Edunov, A. Baevski, and M. Auli. Pre-trained language model representations for language generation. *arXiv preprint arXiv:1903.09722*, 2019.

D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 201–208, 2010.

M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, and D. Hakkani-Tur. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*, 2019.

L. Gao, L. Cao, X. Xu, J. Shao, and J. Song. Question-led object attention for visual question answering. *Neurocomputing*, 391:227–233, 2020.

S. Gao, A. Sethi, S. Agarwal, T. Chung, and D. Hakkani-Tur. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, 2019.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

R. Goel, S. Paul, and D. Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *Proc. Interspeech 2019*, pages 1458–1462, 2019.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

A. Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth*

International Conference on Artificial Intelligence and Statistics, pages 297–304, 2010.

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

M. Heck, C. van Niekerk, N. Lubis, C. Geishauser, H.-C. Lin, M. Moresi, and M. Gašić. Trippy: A triple copy strategy for value independent neural dialog state tracking. *arXiv preprint arXiv:2005.02877*, 2020.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017a.

I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Bosnjak, M. Shanahan, M. Botvinick, D. Hassabis, and A. Lerchner. Scan: Learning hierarchical compositional visual concepts. *arXiv preprint arXiv:1707.03389*, 2017b.

F. Hill, A. Santoro, D. G. Barrett, A. S. Morcos, and T. Lillicrap. Learning to make analogies by contrasting abstract relational structure. *arXiv preprint arXiv:1902.00120*, 2019.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

D. R. Hofstadter. *Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought.* Basic books, 1995.

J. Hong, S. Park, and H. Byun. Selective residual learning for visual question answering. *Neurocomputing*, 2020.

D. Hoshen and M. Werman. Iq of neural networks. *arXiv preprint arXiv:1710.01692*, 2017.

E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*, 2020.

R. Hu, A. Rohrbach, T. Darrell, and K. Saenko. Language-conditioned graph networks for relational reasoning. *arXiv preprint arXiv:1905.04405*, 2019.

S. Hu, Y. Ma, X. Liu, Y. Wei, and S. Bai. Hierarchical rule induction network for abstract visual reasoning. *arXiv preprint arXiv:2002.06838*, 2020.

Y. Jiang, V. Natarajan, X. Chen, M. Rohrbach, D. Batra, and D. Parikh. Pythia v0. 1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*, 2018.

J.-H. Kim, J. Jun, and B.-T. Zhang. Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

M. Kunda, K. McGreggor, and A. K. Goel. A computational model for solving problems from the raven's progressive matrices intelligence test using iconic visual representations. *Cognitive Systems Research*, 22:47–66, 2013.

K. Kurita, N. Vyas, A. Pareek, A. W. Black, and Y. Tsvetkov. Quantifying social biases in contextual word representations. In *1st ACL Workshop on Gender Bias for Natural Language Processing*, 2019.

G. Lample and F. Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

G. Lample and A. Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

H. Le, R. Socher, and S. C. Hoi. Non-autoregressive dialog state tracking. In *International Conference on Learning Representations*, 2019.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

H. Lee, J. Lee, and T.-Y. Kim. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, 2019.

D. R. Little, S. Lewandowsky, and T. L. Griffiths. A bayesian model of rule induction in raven's progressive matrices. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 34, 2012.

J. Liu, C. Wu, X. Wang, and X. Dong. Sequential visual reasoning for visual question answering. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 410–415. IEEE, 2018.

R. Liu, C. Liu, Y. Bai, and A. L. Yuille. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4185–4194, 2019a.

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

F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

A. Lovett, K. Forbus, and J. Usher. A structure-mapping model of raven's progressive matrices. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

J. Mańdziuk and A. Żychowski. Deepiq: A human-inspired ai system for solving iq test problems. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

K. McGreggor, M. Kunda, and A. Goel. Fractals and ravens. *Artificial Intelligence*, 215:1–23, 2014.

S. Mehri, M. Eric, and D. Hakkani-Tur. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*, 2020.

C. S. Mekik, R. Sun, and D. Y. Dai. Similarity-based reasoning, raven's matrices, and general intelligence. In *IJCAI*, pages 1576–1582, 2018.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

E. Nouri and E. Hosseini-Asl. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*, 2018.

S. Park and S.-C. Zhu. Attributed grammars for joint estimation of human attributes, part and pose. In *Proceedings of the IEEE International Conference on Computer vision*, pages 2372–2380, 2015.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms.* MIT press, 2017.

X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*, 2020.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

P. Ramachandran, P. J. Liu, and Q. Le. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, 2017.

O. Ramadan, P. Budzianowski, and M. Gašić. Large-scale multi-domain belief tracking with knowledge sharing. *arXiv preprint arXiv:1807.06517*, 2018.

A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

J. C. Raven. Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. *Unpublished master's thesis, University of London*, 1936.

J. C. Raven et al. *Raven's progressive matrices and vocabulary scales*. Oxford pyschologists Press, 1998.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

A. A. Saifan and N. Al Smadi. Source code-based defect prediction using deep learning and transfer learning. *Intelligent Data Analysis*, 23(6):1243–1269, 2019.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

A. Santoro, F. Hill, D. Barrett, A. Morcos, and T. Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pages 4477–4486, 2018.

D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

I. Schlag, P. Smolensky, R. Fernandez, N. Jojic, J. Schmidhuber, and J. Gao. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*, 2019.

B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

Y. Shan, Z. Li, J. Zhang, F. Meng, Y. Feng, C. Niu, and J. Zhou. A contextual hierarchical attention network with adaptive objective for dialogue state tracking. *arXiv preprint arXiv:2006.01554*, 2020.

S. Shegheva and A. Goel. The structural affinity method for solving the raven's progressive matrices test for intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

R. E. Snow, P. C. Kyllonen, B. Marshalek, et al. The topography of ability and learning correlations. *Advances in the psychology of human intelligence*, 2(S 47): 103, 1984.

K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936, 2019.

X. Steenbrugge, S. Leroux, T. Verbelen, and B. Dhoedt. Improving generalization for abstract reasoning tasks using disentangled feature representations. *arXiv preprint arXiv:1811.04784*, 2018.

C. Strannegård, S. Cirillo, and V. Ström. An anthropomorphic method for progressive matrix problems. *Cognitive Systems Research*, 22:35–46, 2013.

W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

Y. C. Tan and L. E. Celis. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems*, pages 13230–13241, 2019.

W. L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

W. Von Humboldt, W. F. von Humboldt, et al. *Humboldt:'On Language': On the Diversity of Human Language Construction and Its Influence on the Mental Development of the Human Species.* Cambridge University Press, 1999.

K. Wang and Z. Su. Automatic generation of raven's progressive matrices. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2606–2615, 2017.

M. Wertheimer. Gestalt theory. 1938.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush.

Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, 2019a.

T.-F. Wu, G.-S. Xia, and S.-C. Zhu. Compositional boosting for computing hierarchical image structures. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

X. Wu, D. Sahoo, and S. C. Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 2020a.

Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019b.

Y. Wu, H. Dong, R. Grosse, and J. Ba. The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning. *arXiv preprint arXiv:2007.04212*, 2020b.

P. Xu and Q. Hu. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457, 2018.

C. Yang, M. Jiang, B. Jiang, W. Zhou, and K. Li. Co-attention network with question type for visual question answering. *IEEE Access*, 7:40771–40781, 2019a.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019b.

K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, pages 1031–1042, 2018.

Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2019.

X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, and J. Chen. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *ACL 2020*, page 109, 2020.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

X. Zeng, L. Wen, B. Liu, and X. Qi. Deep learning for ultrasound image caption generation based on object detection. *Neurocomputing*, 392:132–141, 2020.

A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. 2020. `https://d2l.ai`.

C. Zhang, F. Gao, B. Jia, Y. Zhu, and S.-C. Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5317–5327, 2019a.

C. Zhang, B. Jia, F. Gao, Y. Zhu, H. Lu, and S.-C. Zhu. Learning perceptual inference by contrasting. In *Advances in Neural Information Processing Systems*, pages 1073–1085, 2019b.

H. Zhang, Y. Niu, and S.-F. Chang. Grounding referring expressions in images by variational context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4158–4166, 2018a.

J.-G. Zhang, K. Hashimoto, C.-S. Wu, Y. Wan, P. S. Yu, R. Socher, and C. Xiong. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*, 2019c.

P. Zhang, Y. Jia, L. Zhang, J. Gao, and H. Leung. A deep belief network based precipitation forecast approach using multiple environmental factors. *Intelligent Data Analysis*, 22(4):843–866, 2018b.

K. Zheng, Z.-J. Zha, and W. Wei. Abstract reasoning with distracting features. In *Advances in Neural Information Processing Systems*, pages 5834–5845, 2019.

V. Zhong, C. Xiong, and R. Socher. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467, 2018.

L. Zhou and K. Small. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*, 2019.

S.-C. Zhu, D. Mumford, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.

국문초록

순환 신경망과 합성곱 신경망은 오랫동안 딥러닝 분야에서 주요 모델로 쓰여왔다. 하지만 두 모델 모두 자체적인 구조에서 오는 한계를 가진다. 최근에는 어텐션(attention)에 기반한 트랜스포머(Transformer)가 더 나은 성능과 구조로 인해서 이들을 대체해 나가고 있다. 트랜스포머 인코더(Transformer encoder)는 자연어 처리 분야에서 특별히 더 많은 연구가 이루어지고 있다. 하지만 Transformer가 특별한 구조와 특징을 가진 데이터에 대해서도 제대로 작동하기 위해서는 그 구조에 적절한 변화가 요구된다. 본 논문에서는 다양한 데이터 종류와 특성에 대한 교사 학습에 적용할 수 있는 트랜스포머 인코더에 기반한 새로운 구조의 모델들을 제안한다. 이번 연구에서 다루는 과업은 시각 IQ 테스트, 대화 상태 트래킹 그리고 수학 질의 응답이다. 시각 IQ 테스트의 입력 변수는 위계를 가진 시각적인 형태이다. 이에 대응하기 위해서 우리는 인지와 사고 측면에서 성능을 향상 시킬 수 있는 새로운 뉴럴 네트워크 구조인, 구조화된 표현형을 처리할 수 있는 계층적인 트랜스포머 인코더 모델을 제안한다. 트랜스 포머 인코더의 계층적 구조와 각각의 트랜스포머 인코더의 구조 모두가 시각 IQ 테스트 데이터의 특징에 적합하다. 대화 상태 트래킹은 여러 개의 도메인-슬롯(domain-slot)쌍에 대한 값(value)이 요구된다. 이를 해결하기 위해서 우리는 사전 학습된 트랜스포머 인코더인, 사전 학습 언어 모델을 활용하여 도메인-슬롯의 관계를 모델링하는 것을 제안한다. 각 도메인-슬롯 쌍에 대한 특수 토큰을 도입함으로써 효과적으로 도메인-슬롯 쌍들 간의 관계를 모델링 할 수 있다. 마지막으로, 수학 질의 응답을 위해서는 수학 질의 응답 데이터에 대해서 사전 학습을 진행함으로써 수학 질의 응답 과업에 대해서 성능을 높히는 방법을 제안한다. 우리의 사전 학습 방법인 질의-응답 마스킹 언어 모델링은 질의와 응답 텍스트 모두를 활용 함으로써 수학 질의 응답 데이터에 적합한 형태이다. 실험을

통해서 각각의 제안된 방법론들이 해당하는 과업과 데이터 종류에 대해서 효과적인 것을 밝혔다.

주요어: 딥러닝, 트랜스포머, 교사 학습, 구조화된 표현형, 사전 학습, 시각 IQ 테스트, 대화 상태 트래킹, 수학 문제 풀이

학번: 2014-21811

감사의 글

서울대학교 산업공학과의 모든 식구들께 감사드립니다.