
THE UNIVERSITY *of* EDINBURGH

# Towards Human-like Compositional Generalization with Neural Models

*Hao Zheng*


Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
The University of Edinburgh

2023

# Abstract

The human language system exhibits systematic compositionality: the ability to produce and understand a potentially infinite number of novel linguistic expressions by systematically combining known atomic components. This type of systematic compositionality is central to the human ability to learn from limited data and make compositional generalizations. There has been a long-standing debate whether this systematicity can be captured by connectionist architectures. Recent years have witnessed a resurgence of interest in this problem with the revival of neural networks. In particular, neural sequence-to-sequence models, as a powerful workhorse of natural language processing (NLP), have been successfully applied to various NLP tasks. However, despite widespread adoption, there is mounting evidence that neural sequence-to-sequence models are deficient in compositional generalization.

In this thesis, we investigate the problem of how to improve compositional generalization of neural sequence-to-sequence models in pursuit of building systems with human-like systematic compositionality. First, assuming that connectionist architectures are fundamentally incapable of acquiring this systematic compositionality which is, in contrast, an inherent part of symbolic (e.g., grammar-based) systems, we attempt to marry symbolic structure with neural models to combine the best of both worlds. We present a two-stage decoding strategy to augment neural sequence-to-sequence models (connectionist architecture) with semantic tagging (symbolic structure), in which an input utterance is tagged with semantic symbols representing the meaning of individual words. Experimental results demonstrate that our framework improves compositional generation for semantic parsing across datasets and model architectures.

Secondly, despite superior compositional generalization, it has not yet been empirically established that symbolic models are appropriate for handling the noise and complexity of natural language, as evidenced by their sub-par performance in practical applications. Therefore, tackling compositional generalization via pure architectural modification has the potential to maintain the robustness and flexibility of neural models required to process real language. We thus attempt to devise a more competent neural model than standard sequence-to-sequence models for compositional generalization. To approach this problem, we design Dangle, a new neural network architecture for sequence-to-sequence modeling to learn more disentangled representations for better compositional generalization compared to the Transformer model. Empirical results on both semantic parsing and machine translation verify that our proposal leads to

more disentangled representations and better generalization, outperforming competitive baselines and more specialized techniques

Previously, we assess the proposed model on synthetic benchmarks to isolate compositional generalization. However, real-world settings involve both complex natural language and compositional generalization. We thus move on to apply disentangled sequence-to-sequence models to real-world compositional generalization challenges. Before doing so, we first propose a methodology for identifying compositional patterns in real-world data and create a new machine translation benchmark that better represents practical generalization requirements than existing artificial challenges.

Then we introduce two key modifications to Dangle which encourage learning more disentangled representations more efficiently. We evaluate the proposed model on existing real-world benchmarks and the benchmark created in this thesis. Experimental results demonstrate that our new architecture achieves better generalization performance across tasks and datasets and is adept at handling real-world challenges.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor Mirella Lapata. She has been extremely inspiring and supportive during my PhD research. She gave me the freedom to explore a variety of research topics, guided me in conducting interesting and solid research, provided detailed feedback on the drafts, and helped in writing beautiful papers. Outside research, I am greatly indebted to Mirella for the generous financial support that she offered during the hard times of my life.

Many thanks to Mark Steedman and Jacob Andreas for examining the thesis. Their thorough examination and insightful feedback helped shape the outcome of my PhD journey and will inspire me to continue exploring the frontier in the future.

I would like to thank the colleagues I interact with in ILCC: Bailin, Biao, Bowen, Chunchuan, Xinchi, Jiangming, Laura, Matthias, Nelly, Parag, Ratish, Reinald, Tom (Hosking), Tom (Sherbone), Xinchi, Xinnuo, Yanpeng, Yang, Yumo and Yao, for many insightful and helpful discussions at every stage of my PhD research. I have learned a lot from them.

I would also like to thank my friends with whom I share my spare time: Shucong, Tianyi, Muyang, Wenbin, Zhijiang, Haoran, Shangmin, Weihong, Sritay, Michael, Chao and some people already mentioned above. They made my PhD life a colorful experience.

I would also like to give special thanks to my girlfriend Xuefei for her heartwarming company and support. Staying with her is always the most comforting and relaxing time after a long day of hard work.

Finally, I wish to thank my parents for their unconditional love and support.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Hao Zheng)*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Human language exhibits systematic compositionality; human language speakers are able to produce and understand a potentially infinite number of novel linguistic expressions by systematically combining known atomic components (Chomsky, 2014; Montague, 1970). This type of compositionality is central to the ability of humans to learn from very limited data and make generalizations, which has been recently described as *systematic or compositional generalization* (Lake and Baroni, 2018; Kim and Linzen, 2020). For example, if a person knows the meaning of the utterance "A boy ate the cake on the table in a house" and the verb "like", it is natural for them to understand the utterance "A boy likes the cake on the table in a house" when they encounter it for the first time. There has been a long-standing debate about whether this systematicity can be captured by connectionist architectures (Fodor and Pylyshyn, 1988; Marcus, 2003). For instance, Fodor and Pylyshyn (1988) have argued that failure to capture systematicity is a major deficiency of neural architectures, contrasting human learners who can readily apply known grammatical rules to arbitrary novel word combinations to individually memorizing an exponential number of sentences.

Recent years have witnessed a resurgence of interest in this problem as neural models achieved tremendous success in natural language processing (Otter et al., 2018). In particular, neural sequence-to-sequence models (e.g., the Transformer architecture, Vaswani et al. (2017) ) have been successfully applied to various NLP tasks ranging from semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Scholak et al., 2021), to machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) and summarization (See et al., 2017; Liu et al., 2022). This general architecture treats different natural language tasks as sequence transduction where a sequence of source symbols is first encoded into continuous vectors, and then from these

source encodings, a decoder autoregressively generates a sequence of target symbols. However, despite widespread adoption, there is mounting evidence that neural sequence-to-sequence models are deficient in compositional generalization (Finegan-Dollak et al., 2018; Kim and Linzen, 2020; Keysers et al., 2020; Li et al., 2021). In the context of semantic parsing, Finegan-Dollak et al. (2018) showed that despite being effective at handling questions with different surface forms but the same meaning, modern neural semantic parsers fail to generalize compositionally to questions with unseen meaning representations. Kim and Linzen (2020) revealed that neural sequence-to-sequence models are more adept at lexical generalization (i.e., novel combination of familiar syntactic structure and words) than structural generalization (i.e., novel combination of familiar syntactic components). Li et al. (2021) demonstrated that neural machine translation models fail badly at translating synthesized novel phrases, although they perform remarkably well under traditional metrics.

How to enable human-like compositional generalization with neural networks has both scientific and practical value. Firstly, it might help us answer the long-standing question of whether connectionist architectures alone are capable of achieving human-level systematicity. Secondly, when deploying neural models in the wild, they often suffer from distribution shifts where the training distribution differs from the test distribution (Koh et al., 2021). Because of their ubiquity in real-world machine learning applications, neural models with better compositional generalization would be practically useful, especially for handling distribution shifts pertaining to novel or rare long-tail compositional patterns.

In this thesis, we investigate the problem of how to improve compositional generalization of neural sequence-to-sequence models from different perspectives. First, assuming that connectionist architectures are fundamentally incapable of systematic compositionality which is, in contrast, an inherent part of symbolic (e.g., grammar-based) systems, we attempt to address the question *how we can combine symbolic structure with neural models to obtain the best of both worlds.* In Chapter 3, We present a two-stage decoding strategy to augment neural sequence-to-sequence models (connectionist architecture) with semantic tagging (symbolic structure), in which an input utterance is tagged with semantic symbols representing the meaning of individual words. The proposed approach improves compositional generalization while preserving as much of the flexibility and generality of neural models as possible.

Secondly, despite superior compositional generalization, it has not yet been empirically established that symbolic models are appropriate for handling the noise and

complexity of natural language, as evidenced by their sub-par performance in practical applications. Therefore, tackling compositional generalization via pure architectural modifications possesses some special benefits. It has the potential to maintain the robustness and flexibility of neural models required to process real language. We thus assume that the inadequacy in compositional generalization is (at least partly) due to the architectural design of neural models. Then, a natural question is *how we can devise a more competent neural model for compositional generalization.* To approach this problem, we propose a new neural architecture for sequence-to-sequence modeling, which learns more disentangled representations for better compositional generalization.

We first assess the proposed model on synthetic benchmarks designed with compositional generalization in mind. However, real-world settings involve both complex natural language and compositional generalization. To investigate real-world compositional generalization, we propose a methodology for identifying compositional patterns in real-world data and create a new machine translation benchmark that better represents practical generalization requirements than existing artificial challenges. We hope that the introduced methodology and benchmark will advance the development of new modeling and learning solutions for real-world compositional generalization.

Finally, to apply the proposed sequence-to-sequence model to real-world compositional generalization challenges, we further improve the model by introducing two key modifications to encourage learning more disentangled representations more efficiently. Experimental results demonstrate that our new architecture achieves better generalization performance across real-world tasks and datasets and is particularly effective for handling long-tail compositional patterns.

We briefly describe our modeling solutions and benchmark below.

## 1.1 Thesis Overview

### 1.1.1 Semantic Tagging

In contrast to neural sequence-to-sequence models, compositional generalization poses no problem for traditional semantic parsers (Zettlemoyer and Collins, 2005, 2007; Wong and Mooney, 2006, 2007; Liang et al., 2013) which typically use a (probabilistic) grammar; the latter defines the meaning of individual words and phrases and how to best combine them in order to obtain meaning representations for entire utterances. Motivated by the superiority of classical grammar-based semantic parsers in composi-

tional generalization (Shaw et al., 2021), we propose to incorporate symbolic structure into neural models by adopting a two-stage decoding strategy with semantic tags as the intermediate layer. Specifically, given a natural language utterance, each word is first labeled with a semantic symbol representing its meaning via a tagger. Semantic symbols are atomic units like predicates (in λ-calculus) or columns (in SQL). The tagger explicitly aligns semantic symbols to tokens or token spans in the utterance. Moreover, the prediction of each semantic symbol is conditionally independent of other symbols in the logical form. This is reminiscent of lexicons in classical semantic parsers, but a major difference is that our tagger is a neural model which considers information based on the entire utterance and can generalize to new words. Then a sequence-to-sequence model takes the utterance and predicted tag sequence, which serves as a soft constraint on the output space, and generates the final meaning representation. Our proposal preserves much of the flexibility and generality of sequence-to-sequence models while inheriting some symbolic characteristics via lexicon-style alignments and two-stage information processing.

We evaluate the proposed approach on three semantic parsing benchmarks covering different semantic formalisms (λ-calculus and SQL) with LSTM- and Transformer-based sequence-to-sequence models (Dong and Lapata, 2016, 2018; Vaswani et al., 2017). Experimental results demonstrate that our framework improves compositional generation for semantic parsing across datasets and model architectures.

### 1.1.2 Disentangled Neural Sequence-to-Sequence Model

Ideally, neural networks that model the meaning of sentences should represent different semantic factors of a sentence (e.g., lexical meaning and semantic relations) in a disentangled way. Neural units modeling a particular semantic factor should be relatively invariant to changes in other factors. For example, the relation between "table" and "house" in the sentence "A boy ate the cake on the table in a house (beside the tree)." and its representation should not be affected by whether there is a PP (beside the tree) modifying "house". However, in a standard neural encoder (e.g., transformer-based), semantic factors tend to be entangled so that changes in one factor affect the representation of others. To remedy this, we proposed Dangle, a **D**isent**angle**d sequence-to-sequence model which allows us to learn disentangled representations for compositional generalization. Specifically, at each time step of the decoding, the proposed model adaptively re-encodes the source input by conditioning the source

representations on the newly decoded target context. We therefore build specialized representations which make it easier for the encoder to exploit relevant-only information for each prediction. In this way, we can effectively disentangle semantic factors involved in different prediction steps.

As a proof-of-concept, we perform experiments on two synthetic semantic parsing benchmarks COGS (Kim and Linzen, 2020), CFQ (Keysers et al., 2020), and one semi-synthetic machine translation benchmark CoGnition (Li et al., 2021) to isolate compositional generalization. Empirical results verify that Dangle leads to more disentangled representations and better generalization, outperforming competitive baselines and more specialized techniques.

### 1.1.3 A Real-world Compositional Generalization Challenge

As a general architecture innovation, Dangle has the potential of improving compositional generalization across a wide range of real-world tasks. However, it is still unclear whether Dangle is effective in real-world settings involving both complex natural language and compositional generalization. Before studying this problem, we first create ReaCT, a new **REA**l-world dataset for **C**ompositional generalization in machine **T**ranslation to better emulate a real-world setting.

We develop a new methodology for *detecting* examples representative of compositional generalization in naturally occurring text. Given a training and test set, our methodology consists of three steps: (a) we discard examples from the test set that contain out-of-vocabulary (OOV) or rare words (in relation to training) to exclude novel atoms which are out of scope for compositional generalization; (b) we then measure how compositional a certain test example is with respect to a training corpus; we introduce a metric which allows us to identify a candidate pool of highly compositional examples; (c) using uncertainty estimation (Malinin and Gales, 2021) , we further select examples from the pool that are both compositional in terms of surface form and challenging in terms of generalization difficulty. Based on this methodology, we create a machine translation benchmark using IWSLT 2014 German $\rightarrow$ English dataset as our training corpus and the WMT 2014 German $\rightarrow$ English shared task as our test corpus.

### 1.1.4 Real-world Disentangled Sequence-to-Sequence Learning

Finally, to apply disentangled sequence-to-sequence models to real-world settings, we devise R-Dangle, an extension of Dangle tailored for real-world languages tasks

Specifically, we present two key modifications to Dangle which encourage learning more disentangled representations more efficiently. The need to perform re-encoding at each time step substantially affects Dangle's training time and memory footprint. It becomes prohibitively expensive on datasets with long target sequences, e.g., programs with 400+ tokens in datasets like SMCalFlow (Andreas et al., 2020). To alleviate this problem, instead of adaptively re-encoding at each time step, we only re-encode periodically, at some interval. Our second modification concerns disentangling the representations of source keys and values, based on which the encoder in Dangle (and also Transformers) passes source information to the decoder. Instead of computing keys and values using shared source encodings, we disassociate their representations: we encode source *values once* and re-encode *keys periodically*.

We evaluate the proposed model on existing benchmarks (Andreas et al., 2020; Li et al., 2021) and the benchmark created in this thesis. Experimental results demonstrate that our new architecture achieves better generalization performance across tasks and datasets. In particular, performing translation with R-Dangle on a diverse corpus of 1.3M WMT examples indicates that it is particularly effective for handling long-tail compositional patterns.

The contributions of the thesis include:

- We proposed a two-stage decoding strategy to augment neural sequence-to-sequence models (connectionist architecture) with semantic tagging (symbolic structure) (Zheng and Lapata, 2021). Experimental results demonstrate that our framework improves compositional generation for semantic parsing across semantic formalisms and model architectures.

- We designed Dangle, a new neural network architecture for sequence-to-sequence modeling to learn more disentangled representations for better compositional generalization compared to the Transformer model (Zheng and Lapata, 2022).

- We proposed a new methodology for identifying compositional patterns in real-world data and created a new real-world machine translation benchmark that better represents practical compositional generalization requirements than existing artificial challenges (Zheng and Lapata, 2023).

- We extended Dangle and applied it to real-world benchmarks. Experimental results demonstrate that our new architecture achieves superior generalization performance across tasks and datasets and is adept at handling real-world challenges

(Zheng and Lapata, 2023).

## 1.2 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 first presents background knowledge on neural sequence-to-sequence models and the theoretical interpretation of compositional generalization. We then discuss the empirical evaluation of compositional generalization with neural sequence-to-sequence models and summarize the main challenges. Finally, we describe existing and concurrent attempts to overcome these challenges.

- Chapter 3 presents a new decoding framework that preserves much of the flexibility and generality of sequence-to-sequence models while featuring lexicon-style alignments and two-stage information processing. Experimental results on three semantic parsing datasets show that the proposed approach consistently improves compositional generalization across model architectures, domains, and semantic formalisms.

- Chapter 4 presents a new network architecture (Dangle) for sequence-to-sequence modeling. We first demonstrate via a toy experiment that one of the reasons hindering compositional generalization of neural models relates to representations being entangled. Then we propose an extension to sequence-to-sequence models which encourages disentanglement by adaptively re-encoding (at each time step) the source input. Experimental results on semantic parsing and machine translation empirically show that our proposal delivers more disentangled representations and better generalization.

- Chapter 5 presents a new real-world dataset for compositional generalization in machine translation. To better emulate a real-world setting, we propose a methodology for identifying compositional patterns in real data and create a new machine translation benchmark based on this methodology.

- Chapter 6 presents an extension of Dangle tailored for real-world compositional generalization challenges. We introduce two key modifications to Dangle to encourage learning more disentangled representations more efficiently. Experimental results on existing benchmarks and the newly created one demonstrate

that our new architecture achieves better generalization performance across tasks and datasets.

- Chapter 7 concludes the thesis and discusses directions for future work

# Chapter 2

# Background

As discussed in Chapter 1, neural sequence-to-sequence models are a powerful workhorse of natural language processing. In this chapter, we first present background knowledge of neural sequence-to-sequence models, which have evolved from pure RNN-based architectures (Cho et al., 2014; Sutskever et al., 2014) to attention-based architectures (Vaswani et al., 2017). Then we describe the theoretical foundation of compositional generalization and the empirical evaluation of compositional generalization for neural sequence-to-sequence models. We summarize the main challenges in building neural networks with human-like compositional generalization. Finally, we describe existing attempts to overcome these challenges.

## 2.1 Neural Sequence-to-Sequence Models

Neural sequence-to-sequence models have been successfully applied to various NLP tasks ranging from semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016; Scholak et al., 2021), to machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) and summarization (See et al., 2017; Liu et al., 2022). This general paradigm treats different natural language tasks as sequence transduction where a sequence of source symbols $X = [x_1, x_2, ..., x_n]$ is first encoded into continuous vectors via an encoder, and then a decoder takes these source encodings and autoregressively generates a sequence of target symbols $Y = [y_1, y_2, ..., y_m]$. Specifically, the input sequence $X$ is first embedded into a sequence of continuous vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, which are then fed to an encoder that fuses context information at each location, performs non-linear transformation, and outputs a sequence of contextualized vector representations $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]$ as source encodings. On the target side, the decoder

autoregressively takes the symbol generated at the preceding time step $y_{t-1}$, likewise embeds it into a vector $\mathbf{y}_{t-1}$, calculates the hidden state, and outputs the probability over the next word $p(y_t|X, Y_{<t})$.

Depending on how the encoder and decoder are instantiated, three popular architectures have been proposed over time.

### 2.1.1 Recurrent Neural Networks

Pioneering the application of neural networks to machine translation, the encoder-decoder paradigm based on recurrent neural networks (RNNs) was originally proposed to model the conditional probability $p(y_t|X, Y_{<t})$ (Cho et al., 2014; Sutskever et al., 2014). In this framework, an RNN encoder reads the sequence of vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ one by one, and outputs a fixed-length context vector $\mathbf{v}$:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{2.1}$$

and

$$\mathbf{v} = q(\{\mathbf{h}_1, \cdots, \mathbf{h}_n\}), \tag{2.2}$$

where $f$ and $q$ are some nonlinear functions.

Then another RNN decoder is used to compute the conditional probability with the initial hidden state set to the representation $\mathbf{v}$:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}) \tag{2.3}$$

and

$$p(y_t|X, Y_{<t}) = g(\mathbf{y}_{t-1}, \mathbf{s}_t, \mathbf{v}) \tag{2.4}$$

where $g$ is a nonlinear, potentially multi-layered, function that outputs the probability of $y_t$, and $\mathbf{s}_t$ is the hidden state of the decoder RNN.

Sutskever et al. (2014) used a long short-term memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) as $f$ and $q(\{h_1, \cdots, h_T\}) = h_T$. At each time step, the LSTM unit recurrently absorbs an input vector $\mathbf{x}_t$, updates its internal memory $\mathbf{c}_t$ and outputs a

hidden state $\mathbf{h}_t$:

$$\mathbf{f}_t = \sigma\left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1} + b_f\right) \tag{2.5}$$

$$\mathbf{i}_t = \sigma\left(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1} + b_i\right) \tag{2.6}$$

$$\tilde{\mathbf{c}}_t = \tanh\left(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + b_c\right) \tag{2.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tilde{\mathbf{c}}_\mathbf{t} \tag{2.8}$$

$$\mathbf{o}_t = \sigma\left(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t + b_o\right) \tag{2.9}$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh\left(\mathbf{c}_t\right) \tag{2.10}$$

where $V_f$, $V_i$ and $V_o$ are diagonal matrices; $W_f$, $W_i$, $W_c$, $W_o$ and $U_f$, $U_i$, $U_c$, $U_o$ are parameter matrices; $b_f$, $b_i$, $b_c$, $b_o$ are parameter vectors. The memory cell $\mathbf{c}_t$ is updated by partially forgetting the existing memory $\mathbf{c}_{t-1}$ and adding a new memory content $\tilde{\mathbf{c}}_t$. The extent to which the existing memory is forgotten is modulated by a forget gate $\mathbf{f}_t$, and the degree to which the new memory content is added to the memory cell is modulated by an *input gate* $\mathbf{i}_t$. The output $\mathbf{h}_t$ is obtained via an output gate $\mathbf{o}_t$ modulating the exposure of memory content.

### 2.1.2 Recurrent Neural Networks with Attention

Instead of computing a fixed context vector $\mathbf{v}$ for passing the source information to the target side, Bahdanau et al. (2015) proposed the attention mechanism to compute a distinct context vector $\mathbf{v}_i$ at each time step.

First, they used a bidirectional RNN to calculate a sequence of forward hidden states and a sequence of backward hidden states. The two hidden states at each location are concatenated to obtain the source encodings $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]$. Then, the context vector $\mathbf{v}_i$ is computed as a weighted sum of these vectors:

$$\mathbf{v}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{h}_j. \tag{2.11}$$

The weight $\alpha_{ij}$ of $h_j$ is computed by

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}, \tag{2.12}$$

where

$$e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$$

is an attention component that scores how much the output at position $i$ attends to the inputs around position $j$. The score is based on the RNN hidden state $\mathbf{s}_{i-1}$ and the

$j$-th hidden state $\mathbf{h}_j$. The attention component is parametrized as a feedforward neural network.

Finally, the RNN decoder uses both the previously generated input and the newly extracted context vector to update its hidden state:

$$s_i = f(\mathbf{s}_{i-1}, [\mathbf{y}_{i-1}\ ; \mathbf{v}_i]).$$

where $[\cdot\ ;\cdot]$ denotes vector concatenation, and each conditional probability is computed as:

$$p(y_t|X, Y_{<t}) = g(\mathbf{y}_{t-1}, \mathbf{s}_t, \mathbf{v}_i) \tag{2.13}$$

### 2.1.3 The Transformer Architecture

The Transformer model was proposed to enable higher parallelization and compute efficiency, as well as to facilitate learning long-range dependencies (Vaswani et al., 2017). It also adopts an encoder-decoder architecture where the encoder and decoder are both composed of a stack of multiple layers. In the encoder, each layer has two sub-layers. The first is a multi-head self-attention layer, and the second is a position-wise fully connected feed-forward network. Similarly, the decoder also uses the two sub-layers and an additional cross-attention layer that accesses the source information by performing attention over the output of the encoder stack.

**Multi-Head Scaled Dot-Product Attention** Extending the original attention mechanism, Vaswani et al. (2017) proposed multi-head scaled dot-product attention. Given queries and keys of dimension $d_k$, and values of dimension $d_v$. The output corresponding to each query is computed as a weighted sum of the values where the weights are the softmax of dot products of the query with all keys, each divided by $\sqrt{d_k}$.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.14}$$

where $Q$, $K$ and $V$ are query, key and value matrices obtained by respectively packing together keys, values, and queries.

Instead of performing a single attention function, the model first linearly projects the queries, keys and values to multiple versions of queries, keys and values (multiple heads) with different weight matrices, and then performs attention for each head in parallel:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O \tag{2.15}$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.16}$$

where $W_i^Q$, $W_i^K$, and $W_i^V$ are the projection weight matrices for the $i$-th head. Note that the outputs of multiple attention heads are concatenated and projected to the final output with the weight matrix $W^O$. Compared to a single attention head, multi-head attention allows the model to attend to information from different representation subspaces at different positions.

The multi-head scaled dot-product attention is used in three different ways: the self-attention in the encoder, the self-attention in the decoder and the cross-attention in the decoder.

**Position-wise Feed-Forward Networks** In addition to the attention component, the Transformer model also adopts a position-wise feed-forward network to transform the input at each position separately. The transformation operation is nonlinear, consisting of two linear projections with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.17}$$

where $x$ is the input vector and $W_1$, $W_2$, $b_1$ and $b_2$ are parameter matrices or vectors.

## 2.2 Theoretical Foundations of Compositional Generalization

Before discussing how to build a system that mimics the compositional abilities of human language speakers, we first examine how humans acquire and process natural language utterances and describe the distinguishing properties of the human language system.

**Systematicity** Human language capacities are systematic:

> "The ability to produce/understand some sentences is intrinsically connected to the ability to produce/understand certain others ."(Fodor and Pylyshyn, 1988)

For example, the two sentences “John loves the girl.” and “The girl loves John.” are systematically related and we do not find native speakers in English who know how to utter and understand one and do not know how to utter and understand the other. This contrasts with learning a language by individually memorizing an exponential number of sentences in an atomic way. Since humans are able to understand and produce sentences they have never seen before, they must use some systematic process to recompose parts and rules that they know innately or have internalized before rather than memorizing an enormous phrase book.

**Productivity** Productivity is a concept closely related to systematicity, but with special emphasis on the unbounded nature of language. It is the ability to produce and understand a potentially infinite number of linguistic expressions by finite means (Chomsky, 2014). Chomsky argued that the knowledge underlying linguistic competence is generative — i.e., one’s knowledge of language supports an unbounded productive capacity. Nevertheless, it has not been empirically established that anyone does or could utter or understand more than a finite number of sentence types. As a result, it is unsurprisingly more controversial than systematicity. These de facto constraints on performance are often argued to be the consequence of the limited memory and lifespan of humans.

**Compositionality** How humans infer meanings from language symbols has been a long-standing research question, having received much attention in linguistics and philosophy. One of the most well-known theories is the principle of compositionality (Partee, 1995): “The meaning of a compound expression is a function of the meanings of the parts and of the way they are syntactically combined”. Depending on how local the composition operations are, there exist two interpretations of compositionality of language (Dankers et al., 2022). In the local (or strong) interpretation, the meaning of an expression depends only on the meanings of its parts, regardless of their internal structure and the external context (Jacobson, 2002). This implies that the meaning of any semantically coherent phrase is conditionally independent of the rest of the sentence. This is reminiscent of the type of compositionality observed in arithmetic: the meaning of (2 + 3) is always 5, independent of the context it occurs in. However, natural language is much more complex and nuanced than that. In some cases, the processing of meaning requires a more global approach. For example, polysemous words need to be disambiguated from broader context information (e.g., in the sentence “these dates

are perfect for our dish/wedding", "dates" could refer to the fruit or days of a month); in anaphora, a pronoun's interpretation depends on its antecedent (e.g., in the sentence "Jimmy has a child who influenced him", "him" refers to Jimmy ). The existence of these frequent linguistic phenomena that cannot be resolved locally leads to the global (or weak) form of compositionality: the meaning of an expression depends on its parts in a compositional way, but the composition function and the parts' meaning could take into account global syntactic structures and semantic information (Szabó, 2012; García-Ramírez, 2019; Dankers et al., 2022). In addition, language also contains various idiomatic expressions whose meanings are not derived from their parts at all (e.g., "to kick the bucket" has the informal meaning "to die"). An idiom's figurative meaning is established by convention and typically considered non-compositional (Swinney and Cutler, 1979).

Based on the various theoretical interpretations described above, we summarize three distinguishing features humans exhibit concerning their underlying compositional capability:

1. Humans can process language expressions that they have never seen before by systematically generalizing to novel compositions of known components. In classical symbolic systems, these known components are assumed to be grammar or lexical rules that are acquired or known innately.

2. The human language system is, to some extent, locally (strongly) compositional in that the meaning of a semantically coherent word/phrase is conditionally independent of the rest of the sentence.

3. There also exist many exceptions to the purely local compositionality property of language where the processing of meaning requires a more global approach.

## 2.3 Empirical Evaluation

There has been a long-standing debate about whether these features of compositionality can be captured by connectionist architectures (Fodor and Pylyshyn, 1988; Marcus, 2003; Lake and Baroni, 2018). Recent years have witnessed a resurgence of interest thanks to the tremendous success of neural networks at various natural language understanding and generation tasks (Sutskever et al., 2014; Vaswani et al., 2017; Dong and Lapata, 2016; Jia and Liang, 2016). To empirically evaluate compositional generalization in neural networks, previous work has translated theoretical concepts related to

compositionality of language into behavioral tests and proposed a variety of benchmarks that allow for measuring how models learn and generalize compositionally. In this thesis, we consider two classical NLP tasks: semantic parsing and machine translation.

**Compositional Type** In a broad sense, any test sentence that a model has not seen during training requires some sort of compositional generalization. However, different types of compositional generalization have been found to pose varying challenges to neural sequence-to-sequence models. On a high level, they can be divided into three categories: paraphrase generalization, lexical generalization, and structural generalization. Examples of each category are shown in Table 2.1. Given the set of all utterances $D_x$ and the set of their meaning representations $D_y$ in a training set:

- in *paraphrase generalization*, the test example involves an unseen $X (X \notin D_x)$ and a seen $Y (Y \in D_y)$. Some word/phrase in a sentence is replaced with its paraphrase with the same meaning, so the context that the original phrase occurs in is recomposed with its paraphrase. The overall meaning of the new sentence remains unchanged.

- in *lexical generalization*, the test example involves both an unseen $X (X \notin D_x)$ and an unseen $Y (Y \notin D_y)$. A word with a different meaning substitutes the original word in a familiar syntactic structure, but the resulting combination has not been seen before. The overall meaning of the new sentence is changed.

- in *structural generalization*, the test example likewise involves both a unseen $X (X \notin D_x)$ and a unseen $Y (Y \notin D_y)$. Different from lexical generalization, familiar syntactic components (such as nominal and prepositional phrases) give rise to novel combinations, leading to a new sentence with a novel syntactic structure. The overall meaning of the sentence is also changed.

Note that in reality, an example could be a complex hybrid of the above three types of generalization.

Traditional semantic parsing benchmarks such as GeoQuery (Zelle and Mooney, 1996) and ATIS (Price, 1990) commonly adopt *question-based* splits where many examples in the test set have the same query templates (induced by anonymizing named entities) as examples in the training (see Chapter 3 for more details). As a result, many of the queries in the test set are seen in training and parsers are being evaluated for their ability to generalize to questions with different surface forms but the same meaning.

| *Training Set* |
|---|
| A boy ate the cake on the table in a house. |
| *cake($x_4$); *table($x_7$); boy($x_1$) AND eat.agent($x_2$, $x_1$) AND eat.theme($x_2$, $x_4$) AND cake.nmod.on($x_4$, $x_7$) AND table.nmod.in($x_7$, $x_{10}$) AND house($x_{10}$) |
| *Test Set (* **Paraphrase Generalization** *)* |
| A boy had the cake on the table in a house. |
| *cake($x_4$); *table($x_7$); boy($x_1$) AND like.agent($x_2$, $x_1$) AND like.theme($x_2$, $x_4$) AND cake.nmod.on($x_4$, $x_7$) AND table.nmod.in($x_7$, $x_{10}$) AND house($x_{10}$) |
| *Test Set (***Lexical Generalization***)* |
| A boy likes the cake on the table in a house. |
| *cake($x_4$); *table($x_7$); boy($x_1$) AND like.agent($x_2$, $x_1$) AND like.theme($x_2$, $x_4$) AND cake.nmod.on($x_4$, $x_7$) AND table.nmod.in($x_7$, $x_{10}$) AND house($x_{10}$) |
| *Test Set (***Structural Generalization***)* |
| A boy ate the cake on the table in a house beside the tree. |
| *cake($x_4$); *table($x_7$); *tree($x_{13}$); boy($x_1$) AND eat.agent($x_2$, $x_1$) AND eat.theme($x_2$, $x_4$) AND cake.nmod.on($x_4$, $x_7$) AND table.nmod.in($x_7$, $x_{10}$) AND house($x_{10}$) AND house.nmod.beside($x_{10}$, $x_{13}$) |

Table 2.1: Examples in the style of COGS (Kim and Linzen, 2020) showcasing paraphrase, lexical and structural generalization. In paraphrase generalization, the word "ate" is replaced with its paraphrase the word "had". The overall meaning of the test example remains the same as the training example. In lexical generalization, a familiar word (e.g., *like*) is attested in a familiar syntactic structure but the resulting combination has not been seen before. In structural generalization, familiar syntactic components give rise to novel combinations (e.g., only prepositional phrases with nesting depth 2 have been previously seen whereas new combinations show nestings of depth 3 or 4).

| Dataset | Task | # examples | | | avg. length | |
|---|---|---|---|---|---|---|
| | | train | valid | test | source | target |
| **GeoQuery** (SQL) | Semantic Parsing | 536 | 159 | 182 | 7.4 | 33.0 |
| **GeoQuery** (λ-calculus) | Semantic Parsing | 598 | – | 282 | 7.4 | 19.0 |
| **ATIS** (SQL) | Semantic Parsing | 4,812 | 121 | 347 | 10.4 | 136.3 |
| **ATIS** (λ-calculus) | Semantic Parsing | 4,472 | 451 | 451 | 10.4 | 28.1 |
| **COGS** | Semantic Parsing | 24,155 | 3,000 | 21,000 | 8.5 | 54.0 |
| **CFQ** | Semantic Parsing | 95,743 | 11,968 | 11,968 | 13.8 | 29.0 |
| **CoGnition** | Machine Translation | 196,246 | 10,000 | 10,800 | 9.8 | 10.0 |
| **SMCalFlow-CS** | Semantic Parsing | 25,404 | 1,324 | 1,325 | 11.7 | 63.5 |
| **ReaCT** | Machine Translation | 162,239 | 7,283 | 3000 | 24.1 | 23.5 |

Table 2.2: Statistics on semantic parsing and machine translation benchmarks used in this thesis. For semantic parsing, the average length is computed based on tokens split by space; for machine translation, it is based on BPE tokens.

Therefore, they largely fall into paraphrase generalization. In contrast, when adopting a *query-based* split, the structure of the queries in the test set is unobserved at training time, and parsers therefore must generalize to questions with different meanings, thus demanding more lexical and structural generalization. Finegan-Dollak et al. (2018) showed that despite being very effective on the question-based split, the performance of neural sequence-to-sequence models drastically drops when shifting to query-based splits.

COGS is a synthetic semantic parsing dataset designed for testing compositional generalization (Kim and Linzen, 2020). The utterances and paired logical forms are generated using a probabilistic context-free grammar. In addition to the standard splits of Train/Dev/Test, COGS provides a generalization (Gen) set that covers five types of compositional generalization: interpreting novel combinations of primitives and grammatical roles, verb argument structure alternation, and sensitivity to verb class, interpreting novel combinations of modified phrases and grammatical roles, generalizing phrases nesting to unseen depths. The former three fall into lexical generalization while the latter two require structural generalization. They found that structural generalization is much more challenging to neural sequence-to-sequence models than lexical generalization.

CoGnition is a relatively realistic compositional generalization dataset targeting machine translation (Li et al., 2021). This benchmark includes 216K English-Chinese sentence pairs; source sentences were taken from the Story Cloze Test and ROCStories

Corpora (Mostafazadeh et al., 2016, 2017) and target sentences were constructed by post-editing the output of a machine translation engine. It also contains a synthetic test set to quantify and analyze compositional generalization of neural MT models. This test set includes 10,800 sentence pairs, which were constructed by embedding synthesized novel compounds into training sentence templates. These compounds are synthesized by composing atomic words with 12 syntactic patterns (corresponding to NP, VP, and PP). Depending on the chosen syntactic patterns, the constructed test examples could involve lexical or structural generalization. Li et al. (2021) demonstrated that neural machine translation models fail badly in translating synthesized novel phrases, although they perform remarkably well under traditional metrics.

**Compositional Level** The taxonomy described above concentrates on a coarse-grained and discrete categorization of compositional generalization. However, within each category, examples can vary greatly in terms of the level of composition and the difficulty of generalization. For example, substituting one word in a sentence is a far less drastic change than constructing a completely new sentence using the same syntactic structure but an entirely different set of words. Intuitively, the latter involves a more complex composition and raises a higher requirement for the compositional generalization ability of models. We review benchmarks that explicitly consider this dimension of compositional generalization below.

CFQ is a large-scale semantic parsing benchmark specifically designed to measure compositional generalization (Keysers et al., 2020). It contains 239,357 compositional Freebase questions paired with SPARQL queries. CFQ was automatically generated from a set of rules in a way that precisely tracks which rules (atoms) and rule combinations (compounds) were used to generate each example. Using this information, the authors generate three splits with *maximum compound divergence* (MCD) while guaranteeing a small atom divergence between train and test sets. In this dataset, atoms refer to entities and relations and compounds to combinations thereof. Large compound divergence indicates the test set is compositionally different than the train set to a large extent. Experiments on this dataset show that that neural sequence-to-sequence models fail to generalize compositionally. More interestingly, there is a surprisingly strong negative correlation between compound divergence and accuracy.

In the original MCD splits, the notion of atoms and compounds depends on the rule-based procedure that generates the source and target examples. It cannot be directly applied to existing non-synthetic datasets. (Shaw et al., 2021) introduced Target

Maximum Compound Divergence (TMCD) splits for GeoQuery and Spider (Yu et al., 2018b) in which they define the notion of atoms and compounds based only on the target representations and apply the same procedure to create splits with maximum compound divergence.

In Chapter 5, we present ReaCT, a *real-world* machine translation benchmark for compositional generalization. The generalization test set is obtained by detecting compositional patterns in relation to an existing training set from a large and diverse pool of candidates. Specifically, we use the IWSLT 2014 German → English dataset as our training corpus and the WMT 2014 German → English shared task as our test corpus and detect from the pool of WMT instances those that exemplify compositional generalization with respect to IWSLT. In the creation process, a metric based on n-gram matching is adopted to assess how compositional a certain example is with respect to a training corpus. This procedure allows us to identify naturally occurring compositional patterns in the hope of better representing practical generalization requirements than artificially constructed challenges. Likewise, the compositional test set constitutes a formidable challenge for neural sequence-to-sequence models, whose performance on it lags behind the in-distribution performance by a margin of about 20 BLEU points.

**Local vs Global Compositionality** Synthetic benchmarks almost exclusively consider the local interpretation of compositionality where the meaning of an expression is computed in a completely local manner, independent of its external context. Natural benchmarks like GeoQuery and ATIS only cover a small fraction of natural language and are likewise shown to largely adhere to this strong local principle (Jia and Liang, 2016). As discussed in the previous section, local compositionality cannot represent the full complexity of natural language.

Dankers et al. (2022) introduce an over-generalization machine translation test to investigate whether neural MT models can translate idioms properly. For instance, the idiom "raining cats and dogs" should be considered globally to arrive at its meaning "heavy rainfall". A local approach would yield an overly literal, non-sensical translation while a global process is required to produce the correct translation. Aside from the test, they call for developing benchmarks using real data to evaluate compositionality on natural language, where composing meaning is not as straightforward as doing the math.

There are a few benchmarks which, despite not explicitly targeting the concept of global compositionality, use noisy and complex real data and try to simulate real-

| Dataset | Split | Compositional Type | Highly Compositional | Real Language |
|---|---|---|---|---|
| **GeoQuery** | QUESTION | Paraphrase | ✗ | ✔ |
| **ATIS** | QUERY | Lexical \| Structural | ✗ | ✔ |
| **COGS** | – | Lexical \| Structural | ✗ | ✗ |
| **CFQ** | – | Lexical \| Structural | ✔ | ✗ |
| **CoGnition** | – | Lexical \| Structural | ✗ | ✗ |
| **SMCalFlow-CS** | – | Lexical \| Structural | ✗ | ✔ |
| **ReaCT** | – | Lexical \| Structural | ✔ | ✔ |

Table 2.3: A summary of compositional generalization attributes for semantic parsing and machine translation benchmarks. QUESTION means question-based splits; QUERY means query-based splits.

world generalization challenges. One example is ReaCT, the benchmark developed in this thesis. Another one is SMCalFlow-CS (Andreas et al., 2020; Yin et al., 2021), a large-scale semantic parsing dataset for task-oriented dialogue, featuring real-world human-generated utterances about calendar management. Yin et al. (2021) proposed a compositional skills split of SMCalFlow (SMCalFlow-CS) that contains single-turn sentences from one of two domains related to creating calendar events (e.g., *Set up a meeting with Adam*) or querying an org chart (e.g., *Who are in Adam's team?*), paired with LISP programs. The training set consists of samples from single domains while the test set contains compositions thereof (e.g., *create a meeting with Adam and his team*). Since zero-shot compositional generalization is highly non-trivial due to novel language patterns and program structures, they consider a few-shot learning scenario, where a small number of cross-domain examples are included in the training set. The benchmarks described above and their attributes are summarized in Table 2.3.

**Challenges** Now we distill the main challenges in building neural network models with human-like compositional generalization:

1. Structural generalization is typically harder than lexical generalization while both of them are generally much more difficult than paraphrase generalization.

2. Examples of increasing compositional levels regarding a training set become increasingly challenging to neural sequence-to-sequence models trained on the set.

3. The central part of the challenge lies in how to simultaneously capture local compositionality that allows for reliable generalization, and global compositionality that is essential for handling the full complexity of natural language.

| Dataset | | Example |
|---|---|---|
| **GeoQuery** | Utterance : | What state borders New York ? |
| | λ-calculus : | `( lambda $0 e ( and ( state:t $0 ) ( next_to:t $0 New_York ) ) )` |
| | SQL : | `SELECT border FROM border_info WHERE state_name = "New York"` |
| **ATIS** | Utterance : | Show me the flights arriving at DAL |
| | λ-calculus : | `( lambda $0 e ( and ( flight $0 ) ( to $0 DAL ) ) )` |
| | SQL : | `SELECT DISTINCT flight_alias.flight_id FROM airport AS airport_alias , flight AS flight_alias WHERE airport_alias.airport_code = "DAL" AND flight_alias.to_airport = airport_alias.airport_code` |
| **COGS** | Utterance : | A boy ate the cake on the table in a house. |
| | logical form : | `*cake(`$x_4$`); *table(`$x_7$`); boy(`$x_1$`) AND eat.agent(`$x_2$`, `$x_1$`) AND eat.theme(`$x_2$`, `$x_4$`) AND cake.nmod.on(`$x_4$`, `$x_7$`) AND table.nmod.in(`$x_7$`, `$x_{10}$`) AND house(`$x_{10}$`)` |
| **CFQ** | Utterance : | What sibling of M0 was M1' s parent? |
| | SPARQL query : | `SELECT DISTINCT ?x0 WHERE { ?x0 ns:people.person.child M1 . ?x0 ns:people.person.sibling M0 . FILTER ( ?x0 != M0 ) }` |
| **CoGnition** | Utterance (en) : | That winter, Taylor barely moved from the fire. |
| | Utterance (zh) : | 那年冬天，泰勒几乎没有从大火中挪动过。 |
| **SMCalFlow-CS** | Utterance : | Create an event called " Dentist Appointment " for December 18 th at 2 : 00 pm . |
| | Lispress : | `( Yield :output ( CreateCommitEventWrapper :event ( CreatePreflightEventWrapper :constraint ( Constraint[Event] :start ( ?= ( DateAtTimeWithDefaults :date ( MD :day # ( Number 18 ) :month # ( Month " DECEMBER " ) ) :time ( NumberPM :number # ( Number 2 ) ) ) ) :subject ( ?= # ( String " Dentist Appointment " ) ) ) ) ) )` |
| **ReaCT** | Utterance (en) : | both setting of tasks must successfully be mastered under supervision . |
| | Utterance (de) : | beide aufgabenstellungen müssen unter aufsicht erfolgreich bewältigt werden . |

Table 2.4: Examples for semantic parsing and machine translation datasets used in this thesis.

## 2.4 Existing Approaches

Existing approaches attempt to advance compositional generalization following three directions, namely devising new models, introducing novel objective functions, and resorting to data augmentation.

**Models** Traditional statistical semantic parsers (Zelle and Mooney, 1996; Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006, 2007; Zettlemoyer and Collins, 2007) typically consist of three key components: a grammar, a trainable scoring model, and a parsing algorithm. The grammar determines the space of derivations from utterances to logical forms, and the model together with the parsing algorithm finds the highest-scoring derivation. These components jointly define the meaning of individual words and phrases and how to best combine them in order to obtain meaning representations for entire utterances.

Some more recent work inherits this general paradigm. For example, Herzig and Berant (2021) make use of a neural network to instantiate the scoring model and develop a span-based parser which predicts a tree over an input utterance, explicitly encoding how partial programs compose over spans in the input. Shaw et al. (2021) propose a hybrid model that combines a high-precision grammar-based approach with a pre-trained sequence-to-sequence model, aiming to handle both compositional generalization and natural language variation. Akyurek and Andreas (2021) incorporate a lexicon into sequence models to disentangle lexical phenomena from syntactic ones. Guo et al. (2020) devise a hierarchical poset decoding architecture that consists of three components: sketch prediction, primitive prediction, and traversal path prediction. All these approaches try to explicitly instill compositional bias into neural models by combining them with task-specific grammars, rules, or procedures. In contrast, Bergen et al. (2021) propose a new neural model – Edge Transformers in the hope of implicitly inducing more compositional solutions. The two major innovations in this architecture are to associate vector states with every pair of input nodes and employ a novel triangular attention mechanism to simulate the process of unification in logic programming.

**Objective Functions** Oren et al. (2020) and Yin et al. (2021) augment the standard cross entropy training objective with attention supervision loss. They first exploit off-the-shelf alignment tools (such as the FastAlign word alignment package (Dyer et al., 2013)) to induce word- or span-level correspondences between the input and

output sequence. Then these alignments are used to guide the learning of the attention mechanism of neural sequence-to-sequence models. Conklin et al. (2021) apply meta-learning to directly optimize for out-of-distribution generalization. They sub-sample existing training data to construct pairs of tasks for meta-learning in an effort to inhibit memorization and encourage generalization.

**Data Augmentation** Another line of work resorts to various data augmentation strategies as a way of injecting a compositional inductive bias into neural models. In this paradigm, heuristics, grammars, or generative models are employed to synthesize parallel examples to augment the existing training data (Jia and Liang, 2016; Andreas, 2020; Akyürek et al., 2021; Wang et al., 2021; Qiu et al., 2022a). Jia and Liang (2016) induce a high-precision synchronous context-free grammar from training data, which allows for sampling new datapoints involving recombination of existing known patterns. Andreas (2020) do away with task-specific grammar engineering in favor of a simple rule-based data augmentation protocol. Under this protocol, synthetic training examples are constructed by taking real training examples and replacing fragments with other fragments that appear in at least one similar environment. Akyürek et al. (2021) develop a learning-based data augmentation approach. They first use a prototype-based generative model to generate new recombinations of training examples and then another sampling procedure to select more of useful examples. Wang et al. (2021) generate synthetic examples for semantic parsing by first generating new programs (e.g., SQL) via a PCFG, and then using a BART-based translation model to map programs to utterances. Qiu et al. (2022a) learn a quasi-synchronous context-free grammar and the parameters of the corresponding probabilistic model from training data. Synthetic examples are sampled from this probabilistic generative model to augment the training data for a sequence-to-sequence model.

**Hybrid Methods** Pre-trained large language models have demonstrated tremendous success in a wide variety of NLP tasks (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2020; Lewis et al., 2020; Brown et al., 2020). They have been proven effective in learning syntactic and semantic information as well as world knowledge from unannotated text corpora, which could be transferred to and benefit downstream tasks. It is thus no surprise that researchers have explored the potential of large language models for compositional generalization. Oren et al. (2020) found that using fixed ELMO and BERT embeddings improves compositional generalization, but performance on out-of-

distribution compositions is still substantially lower than in-distribution performance. Furrer et al. (2020) investigate the performance of finetuning varying sizes of T5 models on CFQ. While pretraining leads to significant improvements in performance compared to non-pre-trained counterparts, the biggest T5 model with 11B parameters still lags behind specialized architectures. Qiu et al. (2022b) evaluate the impact of the model scale of large language models on a suite of compositional generalization challenges in semantic parsing. They observe generally flat or negative scaling curves with full fine-tuning and more positive scaling curves with prompt tuning. In some cases, the latter can match or surpass the former's performance for larger models. In general, fine-tuning smaller models obtains optimal or near-optimal performance.

Most existing approaches rely on task and domain-specific grammars or rules to devise new models, training objectives, and data augmentation protocols. Therefore, improved compositional generalization on a particular dataset or task (most work focus on semantic parsing) often comes with compromised flexibility and generality and does not translate to gains in other datasets or tasks. This is in contrast with human language speakers who are competent generalists and can perform many language tasks. On the other hand, approaches based on pre-trained language models are general paradigms but show limited success in those compositional generalization challenges. In this thesis, we aim to advance compositional generalization while maximally maintaining the flexibility and generality of neural networks.

## 2.5 Summary

In this chapter, we introduced different variants of neural sequence-to-sequence models, one of the most widely adopted neural networks in natural language processing. We then described the theoretical foundations of compositional generalization and the empirical evaluation of compositional generalization for neural sequence-to-sequence models. Empirical evaluation reveals that the main deficiencies of current sequence-to-sequence models lie in highly compositional lexical and structural generalization. Existing attempts to overcome these challenges often come with compromised flexibility and generality and thus could not be appropriate for modeling noisy and complex natural language. It thus motivated our work to advance compositional generalization while maximally maintaining the flexibility and generality of neural networks.

# Chapter 3

# Semantic Tagging

In this chapter, we focus on compositional generalization in semantic parsing, as the majority of existing approaches and benchmarks concerning this problem are based on semantic parsing (Finegan-Dollak et al., 2018; Keysers et al., 2020; Shaw et al., 2021). Motivated by traditional semantic parsing where compositionality is explicitly taken into account by symbolic grammars (Zettlemoyer and Collins, 2005, 2007; Wong and Mooney, 2006, 2007; Liang et al., 2013), we investigate the problem of how to incorporate symbolic structure into neural semantic parsers for better compositional generalization. In this chapter, we propose a new decoding framework that preserves much of the flexibility and generality of sequence-to-sequence models while featuring lexicon-style alignments and two-stage information processing. Specifically, we decompose decoding into two phases where an input utterance is first tagged with semantic symbols representing the meaning of individual words, and then a sequence-to-sequence model is used to predict the final meaning representation conditioning on the utterance *and* the predicted tag sequence. Experimental results on three semantic parsing datasets show that the proposed approach consistently improves compositional generalization across model architectures, domains, and semantic formalisms.

## 3.1 Introduction

Semantic parsing aims at mapping natural language utterances to machine-interpretable meaning representations such as executable queries or logical forms. Sequence-to-sequence neural networks (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) have emerged as a general modeling framework for semantic parsing, achieving impressive results across different domains and semantic formalisms ((Dong and Lapata,

| *Training Set* |
| --- |
| What is the density of Texas? |
| `SELECT density FROM state WHERE state_name = "texas"` |
| *Test Set (Question split)* |
| What is the population density of Maine? |
| `SELECT density FROM state WHERE state_name = "maine"` |
| *Test Set (Query Split)* |
| How many people live in Washington? |
| `SELECT population FROM state WHERE state_name = "washington"` |

Table 3.1: Two test examples from the question- and query-based splits of GEOQUERY and a training example included in both splits. The example in the question-based split shares the same query pattern as the training example while the example in the query-based split has a query pattern different from the training example.

2016; Jia and Liang, 2016; Iyer et al., 2017; Wang et al., 2020), *inter alia*). Despite recent success, there has been mounting evidence (Finegan-Dollak et al., 2018; Keysers et al., 2020; Herzig and Berant, 2021; Lake and Baroni, 2018) that these models fail at *compositional generalization*, i.e, they are unable to systematically generalize to *unseen* compositions of *seen* components. For example, a model that observed at training time the questions "*How many people live in California?*" and "*How many people live in the capital of Georgia?*" fails to generalize to questions such as "*How many people live in the capital of California?*". This is in stark contrast with human language learners who are able to systematically generalize to such compositions (Fodor and Pylyshyn, 1988; Lake et al., 2019).

Previous work (Finegan-Dollak et al., 2018) has exposed the inability of semantic parsers to generalize compositionally simply by evaluating their performance on different dataset splits. Existing semantic parsing datasets commonly adopt *question-based* splits where many examples in the test set have the same query templates (induced by anonymizing named entities) as examples in the training. As a result, many of the queries in the test set are seen in training, and parsers are being evaluated for their ability to generalize to questions with different surface forms but the same meaning (i.e., paraphrase generalization introduced in Section 2.3). In contrast, when adopting a *query-based* split, the structure of the queries in the test set is unobserved at training time, and parsers therefore must generalize to questions with different meanings. Table 3.1 illustrates the difference between question- and query-based splits on

GEOQUERY (Zelle and Mooney, 1996).

On the contrary, compositional generalization poses no problem for traditional semantic parsers (Zettlemoyer and Collins, 2005, 2007; Wong and Mooney, 2006, 2007; Liang et al., 2013) which typically use a (probabilistic) grammar; the latter defines the meanings of individual words and phrases and how to best combine them in order to obtain meaning representations for entire utterances. Neural semantic parsers do away with representing symbolic structure explicitly in favor of a more general approach which directly transduces the utterance into a logical form, avoiding domain-specific assumptions and grammar learning. They usually deliver better performance in practical applications, especially when equipped with pre-trained embeddings or language models (Roy et al., 2022).

Given the superior compositional generalization ability of grammar-based semantic parsing, a natural question is: *how can we marry it with neural models to combine the best of both worlds*? In this chapter, we attempt to address this challenge. We believe that the symbolic paradigm provides two important insights that could guide our design of neural semantic parsers with better compositional generalization. Firstly, the probability of a logical form is decomposed into *local* factors under strong conditional independence assumptions while in neural semantic parsing the prediction of each symbol directly depends on *all* previously decoded symbols. This strong expressivity may hurt compositional generalization since different kinds of information are bundled together, rendering the model's predictions susceptible to irrelevant context changes. Secondly, there exist *hard* alignments between logical constructs and linguistic expressions but in neural parsers, the two are only loosely related via the *soft* attention mechanism. Explicit alignments can help distinguish which language segments are helpful for predicting certain components in the logical form, potentially improving compositional generalization.

Motivated by the above discussion, we devise a new decoding framework that preserves much of the flexibility and generality of sequence-to-sequence models while featuring lexicon-style alignments and separate information processing. Specifically, we decompose decoding into two phases. Given a natural language utterance, each word is first labeled with a semantic symbol representing its meaning via a tagger. Semantic symbols are atomic units like predicates (in $\lambda$-calculus) or columns (in SQL). The tagger *explicitly* aligns semantic symbols to tokens or token spans in the utterance. Moreover, the prediction of each semantic symbol is conditionally independent of other symbols in the logical form. This is reminiscent of lexicons in classical semantic parsers, but

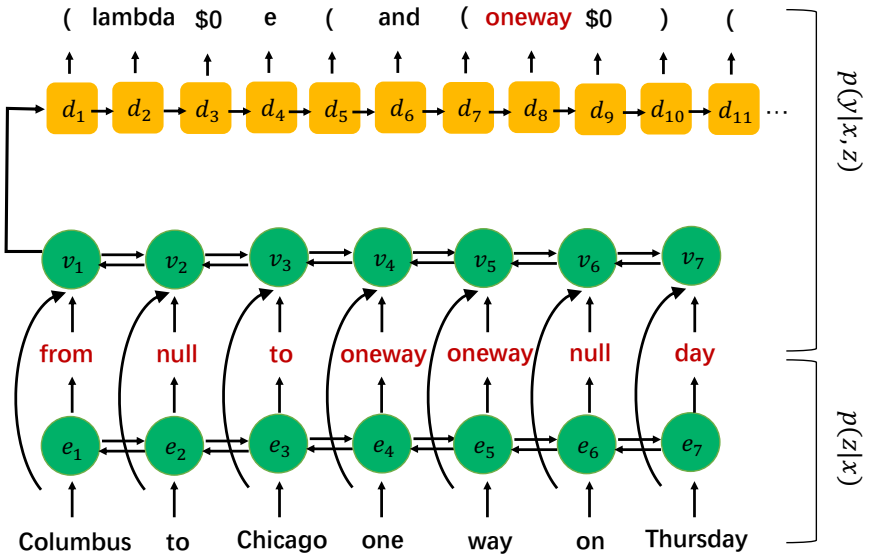


Figure 3.1: We first tag natural language input $X$ with semantic symbols (e.g., predicates shown in red) and predict tag sequence $Z$. We generate the final semantic representation $Y$, given $X$ and $Z$ as input.

a major difference is that our tagger is a neural model which considers information based on the entire utterance and can generalize to new words. A sequence-to-sequence model takes the utterance and predicted tag sequence which serves as a soft constraint on the output space, and generates the final meaning representation. Our framework is general in that it could incorporate any sequence-to-sequence model as the base model and augment it with semantic tagging.

We evaluate the proposed approach on query-based splits of three semantic parsing benchmarks: ATIS, GEOQUERY, and a subset of WIKISQL covering different semantic formalisms ($\lambda$-calculus and SQL). We report experiments with LSTM- and Transformer-based models (see Section 2.1 for more details) (Dong and Lapata, 2016, 2018; Vaswani et al., 2017) demonstrating that our framework improves compositional generation across datasets and model architectures. Our approach is also superior to a recent data augmentation proposal (Andreas, 2020), specifically designed to enhance compositional generalization.

## 3.2 Model Architecture

Our goal is to learn a semantic parser that takes as input a natural language utterance $X = [x_1, x_2, ..., x_n]$ and predicts a meaning representation $Y = [y_1, y_2, ..., y_m]$. We

decompose the parser $p(Y|X)$ into a two-stage generation process:

$$p(Y|X) = p(Y|X,Z)p(Z|X) \tag{3.1}$$

where $Z = [z_1, z_2, ..., z_n]$ is a tag sequence for $X$. Every tag $z_t$ is a symbol in $Y$ (roughly) representing the meaning of $x_t$. Therefore, the first-stage model $p(Z|X)$ is essentially a tagger that tries to predict the meanings of individual words. The second-stage model takes the word sequence $X$ and its accompanying tag sequence $Z$ as input, and generates the final semantic representation $Y$. Figure 3.1 shows this two-stage generation process. It is important to note that tags $Z$ are *latent* and must be induced from training data, i.e., pairs of natural language utterances and representations of their meaning. We discuss how the tagger is learned in Section 3.3.

### 3.2.1 Semantic Tagging

As shown in Figure 3.1, the tagging model $p(Z|X;\theta)$ contains an encoder which transforms input sequence $x_1, x_2, ..., x_n$ into a sequence of context-sensitive vector representations $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n$. Each word $x_i$ is mapped to embedding $\mathbf{x}_i$, and the sequence of word embeddings $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ is fed to a bi-directional recurrent neural network with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). A bi-LSTM recursively computes the hidden states at the $t$-th time step via:

$$\overrightarrow{\mathbf{h}}_i = f_{\text{LSTM}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{x}_i) \tag{3.2}$$

$$\overleftarrow{\mathbf{h}}_i = f_{\text{LSTM}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{x}_i) \tag{3.3}$$

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i ; \overleftarrow{\mathbf{h}}_i] \tag{3.4}$$

where $\mathbf{h}_i$ is the concatenation of vectors $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$, and $f_{\text{LSTM}}$ refers to the LSTM function. We feed both $\mathbf{h}_i$ and $\mathbf{x}_i$ to the final output layer in order to predict tags $z$:

$$p(Z|X;\theta) = \prod_{i=1}^{n} p(z_i|X;\theta) \tag{3.5}$$

$$= \prod_{i=1}^{n} \text{softmax}(W\mathbf{h}_i + U\mathbf{x}_i + b) \tag{3.6}$$

$W$, $U$, and $b$ are parameters in the output layer.

These semantic tags are automatically induced from logical forms without any extra annotation and vary depending on the meaning representation at hand (e.g., $\lambda$-calculus, SQL). They provide a task-specific middle layer, serving the goal of injecting inductive bias for compositional generalization. It is different from part-of-speech

tagging (Toutanova et al., 2003) or universal semantic tagging (Abzianidze and Bos, 2017) in the classical NLP pipeline. Those tagging modules typically express general syntactic or semantic information independent of downstream tasks and require (often costly ) additional annotation.

### 3.2.2 Meaning Representation Generation

LSTM-based encoder-decoder models with an attention mechanism have been successfully applied to a wide range of semantic parsing benchmarks (Dong and Lapata, 2016; Jia and Liang, 2016; Iyer et al., 2017), while Transformers have rapidly gained popularity for various NLP tasks including semantic parsing (Wang et al., 2020; Sherborne et al., 2020; Platanios et al., 2021; Scholak et al., 2021). Our approach is model-agnostic in that it could be combined with any type of sequence-to-sequence model; to highlight this versatility, we present experiments with both LSTM- and Transformer-based models. We first embed the predicted tag and word sequences, obtaining tag embeddings $\mathbf{e}^g_1, \mathbf{e}^g_2, .., \mathbf{e}^g_n$ and word embeddings $\mathbf{e}^w_1, \mathbf{e}^w_2, ..., \mathbf{e}^w_n$. Then, we concatenate the two types of embeddings at each time step and feed them to a sequence-to-sequence model:

$$\mathbf{u}_t = [\mathbf{e}^g_t ; \mathbf{e}^w_t] \tag{3.7}$$

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n] \tag{3.8}$$

$$Y = f_{\texttt{seq2seq}}(\mathbf{U}) \tag{3.9}$$

where $[\cdot\,;\cdot]$ denotes vector concatenation and $f_{\texttt{seq2seq}}$ denotes a sequence-to-sequence model variant (LSTM- or Transformer-based in our case) that takes a sequence of vector representations as input and ultimately generates a logical form. Tag embeddings are shared with the target symbol embeddings. Therefore, the only adaptation we make to the baseline model is to replace the original word embeddings with tag-augmented input.

## 3.3 Model Learning

Our proposed approach combines a semantic tagger with a sequence-to-sequence model. The tagger learning problem is challenging since $Z$ is unobserved. In this section, we explain how the tagger and the overall model are trained.

| λ-calculus | |
|---|---|
| $X$ : | Columbus to Chicago one way on Thursday |
| $Z$ : | Columbus/**from** to/**null** Chicago/**to** one/**oneway** way/**oneway** on/**null** Thursday/**day** |
| $S$ : | **oneway**, **from**, **to**, **day**, null |
| $Y$ : | `( lambda $0 e ( and ( oneway $0 ) ( from $0 columbus:ci ) ( to $0 chicago:ci ) ( day $0 thursday:da ) ) )` |

| SQL | |
|---|---|
| $X$ : | What is the area of Washington |
| $Z$ : | What/**null** is/**null** the/**null** area/**area** of/**null** Washington/**state_name** |
| $S$ : | **area**, **state_name**, null |
| $Y$ : | `SELECT area FROM state WHERE state_name = "washington"` |

Table 3.2: Utterances $X$, their meaning representations $Y$, symbol sets $S$, and predicted word/**tag** sequences $Z$.

### 3.3.1 Tagger Learning

We learn a tagger $p(Z|X;\theta)$ from training data consisting of pairs of natural language utterances $X = [x_1, x_2, ..., x_n]$ and symbol sets $S = \{s_1, s_2, ..., s_l\}$ (with $s_j \in Y$). The symbol set contains atomic semantic units such as λ-calculus predicates and SQL column names. Table 3.2 presents examples of symbol sets for these two formalisms. As can be seen, symbols have close ties to utterances, there is often a correspondence between them and individual words or phrases. It is therefore natural to predict this (basic) part of a meaning representation via a tagger. To bridge the gap between the tag sequence we intend to predict and the symbol set we have as supervision, we introduce latent variable $A = [a_1, a_2, ..., a_n]$ where $a_j$ denotes the index of a word aligned to $s_j$. We add $(n-l)$ null symbols to target set $S = \{s_1, s_2, ..., s_l, s_{l+1}, ..., s_n\}$ because $n$ is typically larger than $l$, and we allow the tagger to output null for some words.

**Entity Linking** For some symbols, it is rather straightforward to determine the corresponding alignments based on the results of entity linking, a critical subtask in semantic parsing which is generally treated as a preprocessing step (Dong and Lapata, 2016; Jia and Liang, 2016). We thus define the following two rules to automatically align symbols to words in an utterance based on entity linking: (1) for λ-calculus expressions, if a predicate takes only one entity as an argument (e.g., `day $0 thursday:da`) and

this entity can be linked to a word or phrase in the utterance, we assume there is an alignment between them (e.g., `day` aligns to *Thursday*); (2) for SQL expressions, if the entity in a filter clause (e.g., `state_name = "washington"`) can be linked to an expression in the utterance, again we align the column (e.g., `state_name`) to the linguistic expression (e.g., *Washington*). Both rules capture the intuition that some semantic symbols are implied by corresponding entities without being explicitly verbalized. As shown in Table 3.2, there is no linguistic expression in the utterance "*What is the area of Washington*" which corresponds to the logical expression of `state_name`, instead `state_name` is implied by the entity `washington`.

**Expectation-Maximization** Besides entity linking, there remain symbols without alignments, such as unary predicates (e.g., `oneway $0`). For these, we use an EM-style algorithm which iteratively infers latent alignments $A$ and uses them to update the tagger. A hard-EM algorithm that predicts the most probable $A$ seems reasonable as in most cases there is a single correct alignment. However, we find that hard-EM renders training unstable and prone to overfitting to incorrect alignments. We instead warm up the training with a soft-EM algorithm first and switch to hard-EM later on. Without loss of generality, we describe the algorithm for all symbols including those that could be aligned via (entity linking) rules. Specifically, we model the generation of $S$ as follows:

$$\begin{aligned} p(S|X;\theta) &= \sum_A p(A|X)p(S|X,A;\theta) \\ &= \sum_A p(A|X)\prod_{j=1}^{n} p(z_{a_j} = s_j|X;\theta) \end{aligned} \tag{3.10}$$

where $p(A|X)$ is a uniform prior over $A$ and $p(z_{a_j} = s_j|x;\theta)$ is the tagger model above. We could constrain the alignment from words to symbols to be injective (as this would more faithfully capture the complex dependencies between them). Unfortunately, this renders posterior inference on $A$ intractable. Instead, we model the alignment of each symbol independently as:

$$\begin{aligned} p(S|X;\theta) &= \sum_A \prod_{j=1}^{n} p(a_j|X) \prod_{j=1}^{n} p(z_{a_j} = s_j|X;\theta) \\ &= \prod_{j=1}^{n} \sum_{a_j} p(a_j|X)p(z_{a_j} = s_j|X;\theta) \end{aligned} \tag{3.11}$$

Under this assumption, we are able to exactly compute the posterior probability of each alignment $a_j$:

$$\begin{aligned}\pi_{ij}(\theta) &= p(a_j = i|X,S;\theta) \\ &= \frac{p(a_j = i|X)p(z_i = s_j|X;\theta)}{\sum_{k=1}^{n} p(a_j = k|X)p(z_k = s_j|X;\theta)}\end{aligned} \tag{3.12}$$

Note that we manually set the value of $\pi_{ij}(\theta)$ if $a_j$ can be induced in advance via entity linking. At the $t$-th iteration, we first use the present tagger $p(Z|X;\theta^t)$ to compute $\pi_{ij}(\theta^t)$, the likelihood of aligning symbol $s_j$ to word $x_i$. For soft-EM, these assignments are then directly used to train the tagger with the following objective:

$$\mathcal{J}_t(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} \pi_{ij}(\theta^t) \log p(z_i = s_j|X;\theta) \tag{3.13}$$

$$\theta^{t+1} = \arg\max_{\theta}\ \mathcal{J}_t(\theta) \tag{3.14}$$

For hard-EM, one could exploit $\pi_{ij}(\theta^t)$ to induce the most probable alignment for each symbol. However, there are cases where a symbol is aligned to multiple words, e.g., when the same word occurs multiple times in an utterance or when a symbol is aligned to a phrase. To deal with such cases, we induce a hard version of the posterior probability $\tilde{\pi}_{ij}(\theta^t)$ in the following way:

$$\tilde{\pi}_{ij}(\theta^t) = \begin{cases} 1 & \text{if } \pi_{ij}(\theta^t) > \beta \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

$$(1 \le j \le l)$$

$$\tilde{\pi}_{ij}(\theta^t) = \frac{1 - \sum_{k=1}^{l} \tilde{\pi}_{ik}}{n - l} \tag{3.16}$$

$$(l+1 \le j \le n)$$

where $\beta$ is a threshold used to discretize the soft alignment distributions. Reshaping the posteriors in this manner allows a symbol to be aligned to multiple words while removing noisy incorrect alignments. Equation (3.16) ensures that the sum of posteriors corresponding to a word is one, in the hope of encouraging the predicted tag sequence distribution to be as close to a normal tag sequence distribution as possible. We replace $\pi_{ij}(\theta^t)$ in $\mathcal{J}(\theta|\theta^t)$ with $\tilde{\pi}_{ij}(\theta^t)$ as the training objective to perform hard-EM updates:

$$\tilde{\mathcal{J}}_t(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{\pi}_{ij}(\theta^t) \log p(z_i = s_j|X;\theta) \tag{3.17}$$

$$\theta^{t+1} = \arg\max_{\theta}\ \tilde{\mathcal{J}}_t(\theta) \tag{3.18}$$

**Algorithm 1:** Training the tagger

**Input:** Dataset $\mathcal{D}$ where each example is a question $x$ paired with symbol set $s$.
Number of soft-EM updates $T_s$. Number of overall updates $T$.

**Output:** Tagger model parameters $\theta^{T+1}$

Initialize tagger parameters $\theta^1$ randomly;

**for** $t = 1, ..., T$ **do**
  sample an example ($x$, $s$)
  **if** $t < T_s$ **then**
    /* do soft-EM update */
    Compute $\pi_{ij}(\theta^t)$
    $\theta^{t+1} \leftarrow \text{Optimizer}(\theta^t, \nabla_{\theta_t} \mathcal{J}_t(\theta_t))$
  **else**
    /* do hard-EM update */
    Compute $\tilde{\pi}_{ij}(\theta^t)$
    $\theta^{t+1} \leftarrow \text{Optimizer}(\theta^t, \nabla_{\theta_t} \tilde{\mathcal{J}}_t(\theta_t))$
  **end**
**end**
**return** $\theta^{T+1}$

Our training procedure is shown in Algorithm 1. Note that in each EM iteration, we use objective $\mathcal{J}_t(\theta)$ or $\tilde{\mathcal{J}}_t(\theta)$ to compute the gradient and update parameters once rather than maximizing the objective function.

### 3.3.2 Parser Learning

Learning a semantic parser in our setting is straightforward. After training the tagger, we run it over the examples in the training data and obtain tag sequence $\hat{Z}$ for each pair of utterance $X$ and meaning representation $Y$.

$$\hat{Z} = \arg\max_{Z} p(Z|X;\theta) \tag{3.19}$$

Then, we maximize the likelihood of generating $Y$ given $X$ and $\hat{Z}$:

$$\hat{\theta} = \arg\max_{\theta} \log p(Y|X,\hat{Z};\theta) \tag{3.20}$$

## 3.4 Experimental Setup

**Datasets** Our experiments evaluate the proposed framework on compositional generalization. We present results on query-based splits for three widely used semantic parsing benchmarks, namely ATIS (Dahl et al., 1994), GEOQUERY (Zelle and Mooney, 1996), and WIKISQL (Zhong et al., 2017). For GEOQUERY (880 language queries to a database of U.S. geography) and ATIS (5,410 queries to a flight booking system) meaning representations are in λ-calculus and SQL. We adopt the split released by Finegan-Dollak et al. (2018) for SQL. We create query-based splits for λ-calculus and we use the preprocessed versions provided in Dong and Lapata (2018), where natural language expressions are lowercased and stemmed with NLTK (Bird et al., 2009), and entity mentions are replaced by numbered markers.

WIKISQL is a large-scale semantic parsing dataset released more recently (Zhong et al., 2017). It is used as a testbed for generating an SQL query given a natural language question and table schema (i.e., table column names). Since SQL queries in most examples are simple and only contain one filtering condition (e.g., `SELECT city_of_license WHERE frequency = 89.9`), we use a subset (16,835 training examples, 2,602 validation examples, and 4,915 test examples) containing queries with more than one filtering condition (e.g., `SELECT year WHERE manufacturer = plymouth AND date = february 9`). These examples are more compositional and better suited to evaluating compositional generalization.

**Comparison Models** On ATIS and GEOQUERY we trained two baseline sequence-to-sequence models: LSTMs with attention and Transformers as the base units (see Section 2.1 for a detailed introduction). To examine whether our results carry over to pretrained contextual representations, we report experiments with an LSTM model enhanced with RoBERTa (Liu et al., 2019). We put an LSTM encoder-decoder on top of RoBERTa which first uses RoBERTa to encode input utterances and then the output encodings are fed to the LSTM-based sequence-to-sequence model to generate queries. We also compare against two related approaches. The first is GECA (Andreas, 2020), a recently proposed data augmentation method aimed at injecting a compositional inductive bias into sequence-to-sequence models via synthesizing new training examples. Synthetic examples are constructed by taking real training examples and replacing fragments with other fragments that appear in at least one similar environment. The second is Attention Supervision introduced in Oren et al. (2020). They encourage

generalization by supervising the decoder attention with pre-computed token alignments. The FastAlign word alignment package (Dyer et al., 2013) is used to induce the token-level alignments. We use the alignments induced by our tagger instead of an off-the-shelf word aligner adopted in their paper.

For WIKISQL, our baseline model follows the coarse-to-fine decoding approach (Coarse2fine) put forward in Dong and Lapata (2018) which is well suited to the formulaic nature of the queries, takes the table schema into account, and performs on par with some more sophisticated models (McCann et al., 2018; Yu et al., 2018a). They predict `SELECT` and `WHERE` SQL clauses separately (all queries in WIKISQL follow the same format, i.e., "`SELECT agg_op agg_col WHERE (cond_col cond_op cond AND)...`", which is a small subset of the SQL syntax). The `SELECT` clause is predicted via two independent classifiers, while the `WHERE` clause is generated via a sequence model with a sketch as an intermediate outcome. Their encoder augments question representations with table information by computing attention over table column vectors and deriving a context vector to summarize the relevant columns for each word.

Our tagger uses Coarse2fine's table-aware encoder to predict tags. Our parser diverges slightly from their model: while for each word the context vector is originally computed by the attention mechanism, we replace it with the column vector specified by the corresponding tag.

**Configuration** We implemented the base semantic parsers (the LSTM and Transformer models) with fairseq (Ott et al., 2019). As far as GECA is concerned, we have a different setting from Andreas (2020): we use the preprocessed versions provided by Dong and Lapata (2018) for ATIS and GEOQUERY, while they report experiments on GEOQUERY only, with different preprocessing. We used their open-sourced code to generate synthetic data for our setting in order to make experiments comparable. For Coarse2fine, we used the code released by the authors.

Hyperparameters for the semantic taggers were validated on the development split of ATIS and were directly copied for GEOQUERY because of its small size. Dimensions of hidden vectors and word embeddings were selected from {150, 200, 250, 300}. The number of layers was selected from {1, 2}. The batch size was set to 20 and the overall update step was set to 20,000. The number of steps for soft-EM updates was selected from {5,000, 7,000, 10,000, 13,000}. The threshold $\beta$ used in hard-EM was selected from {0.20, 0.23, 0.26, 0.29, 0.32, 0.35}. We used the Adam optimizer (Kingma and Ba, 2015) to train the models and the learning rate was selected from {0.0001, 0.0003,

| Method | λ-calculus | | SQL | |
|---|---|---|---|---|
| | GEO | ATIS | GEO | ATIS |
| GECA | 48.1 | 51.6 | 52.1 | 24.0 |
| Transformer | 39.8 | 51.2 | 53.9 | 23.0 |
| Transformer + ATTENTION SUPERVISION | 43.4 | 53.3 | 58.6 | 22.0 |
| Transformer + SEMANTIC TAGGING | 44.0 | 53.0 | 61.9 | 28.6 |
| LSTM | 49.8 | 56.2 | 48.5 | 28.0 |
| LSTM + ATTENTION SUPERVISION | 53.6 | 59.7 | 46.9 | 28.7 |
| LSTM + SEMANTIC TAGGING | 52.1 | 62.1 | 63.6 | 29.1 |
| RoBERTa | 54.4 | 57.5 | 58.8 | 28.6 |
| RoBERTa + ATTENTION SUPERVISION | 56.3 | 59.9 | 59.3 | 28.4 |
| RoBERTa + SEMANTIC TAGGING | 57.5 | 63.7 | 69.6 | 27.7 |

Table 3.3: Exact-match accuracy on GEOQUERY and ATIS; results are averaged over 5 random seeds.

| Method | Overall | Where |
|---|---|---|
| Coarse2fine | 58.0 | 71.3 |
| Coarse2fine + ATTENTION SUPERVISION | 58.8 | 72.8 |
| Coarse2fine + SEMANTIC TAGGING | 60.6 | 75.0 |

Table 3.4: Evaluation results on a WIKISQL subset. `Overall`: exact-match accuracy of whole SQL expressions; `Where`: accuracy of predicting `WHERE` clauses.

0.001}. Our semantic parsers used the same hyperparameters as the base models except for some necessary changes to incorporate tag inputs. For models using RoBERTa, we first freeze RoBERTa and train the model for some steps, and then resume fine-tuning.

**Evaluation** We use exact-match accuracy as our evaluation metric, namely the percentage of examples that are correctly parsed to their gold standard meaning representations.

| Method | λ-calculus | | SQL | |
|---|---|---|---|---|
| | GEO | ATIS | GEO | ATIS |
| LSTM | 16.1 / 10.9 / 23.2 | 13.7 / 9.8 / 20.1 | 14.1 / 5.6 / 31.8 | 21.3 / 26.5 / 23.7 |
| LSTM + ST | 16.5 / 9.7 / 21.7 | 13.0 / 8.9 / 15.9 | 19.0 / 6.7 / 10.5 | 22.1 / 24.9 / 23.9 |
| ROBERTA + ST | 13.0 / 9.6 / 19.7 | 12.9 / 6.9 / 16.4 | 12.7 / 5.7 / 11.8 | 22.6 / 16.6 / 32.8 |

Table 3.5: Breakdown of different types of error. In each cell, the left shows the proportion of predicting correct semantic symbols but incorrect queries; the middle is the proportion of predicting a subset of correct symbols (i.e., missing some semantic symbols); the right is the proportion of predicting symbols which do not exist in gold queries. ST stands for semantic tagging.

## 3.5 Results

**Does Tagging Help Parsing?** Table 4.6 summarizes our results on ATIS and GEOQUERY. On both datasets, we observe that the proposed tagger (+SEMANTIC TAGGING ) boosts the performance of the base model (i.e., Transformer, LSTM) for both λ-calculus and SQL. The LSTM model is generally superior to Transformer except on SQL GEOQUERY. Enhancing the LSTM model with pretrained contextual representations (see the last block in the table) generally increases accuracy, yet our semantic tagger brings improvements on top of ROBERTA (with the exception of SQL ATIS). This points to the generality of our approach which benefits neural parsers with different architectures trained on distinct semantic representations. Gains are particularly significant on ATIS with λ-calculus (we observe an absolute improvement of 6.2 points over ROBERTA) and GEOQUERY with SQL (with 10.8 points absolute improvement over ROBERTA).

In some settings, attention supervision also achieves improvements over baseline sequence models, but these are inconsistent and sometimes it even slightly hurts performance. We find that attention supervision is sensitive to the weight hyperparameter that controls the strength of attention loss and requires careful tuning to achieve good performance. We conjecture that the soft attention mechanism (even with proper supervision signals) is still sensitive to irrelevant context changes and prone to errors in cases requiring compositional generalization. The LSTM+SEMANTIC TAGGING model achieves better accuracy than GECA which adopts a data augmentation strategy to train an LSTM-based sequence-to-sequence model for compositional generalization. We incorporate a similar inductive bias into the parser but in an orthogonal way.

Results on WIKISQL are shown in Table 3.4. Semantic tagging boosts Coarse2fine

in terms of exact match. In particular, it improves the prediction of WHERE clauses, by a 4.3% absolute margin. We would not expect semantic tagging to benefit any other parts of the generation of the SQL query, since only WHERE clauses are decoded sequentially in the Coarse2fine model. Gains in the generation of WHERE clauses translate to improvements in overall accuracy. Attention supervision also improves generalization but falls behind our semantic tagger.

**Do Meaning Representations Matter?** Improvements of our semantic tagger on ATIS with SQL and GEOQUERY with λ-calculus are less dramatic compared to ATIS with λ-calculus and GEOQUERY with SQL. Upon closer inspection, we find that ATIS SQL queries typically include many bridging columns that are used to join two tables (e.g., the column airport_alias.airport_code in Table 2.4). This arises from the complex database structure in ATIS: there are 32 tables in total and each query involves 6.4 tables on average. These bridging columns are SQL-specific and generally do not align with any linguistic expressions, so we cannot improve their prediction via semantic tagging. A prerequisite for semantic tagging is that there exist alignments between language expressions and atomic semantic symbols. We could restrict the semantic tagger to only predicting symbols that align with linguistic expressions and leave the generation of other symbols to the second stage. However, how to automatically select appropriate symbols as semantic tags is an avenue for future work.

On GEOQUERY with λ-calculus, the semantic tagger performs extremely well, achieving 86.2% accuracy in predicting semantic symbols, but the final accuracy in predicting queries is only 52.1% (LSTM+SEMANTIC TAGGING). Although semantic tagging can help generalize to utterances where seen syntactic structure and concept words are combined in an unseen way (e.g., *Monkeys like bananas* generalizes to *Cats like fish*), it fails to generalize to utterances with unseen syntactic structure (e.g., *Monkeys like bananas* generalizes to *Cats like fish that like water*). Handling utterances with unseen composition of seen syntactic components is yet another generalization challenge for modern semantic parsers.

**Where do Gains Come from?** Our approach transfers much of the prediction of semantic symbols from the sequence-to-sequence model to the tagger; it does this by replacing the attention mechanism, which learns to attend to *specific* parts of an utterance, with per-word tagging which considers *all* parts of an utterance. We hypothesize that

this architecture can better exploit source information to predict individual semantic symbols. To test this hypothesis, we analyzed errors in the predictions of the LSTM model with and without the proposed semantic tagger, and classified them into three types. The first type predicts incorrect queries but correct semantic symbols. The second type predicts only a subset of correct semantic symbols, thus omitting some semantic symbols. The third type predicts wrong semantic symbols that do not exist in gold queries. As shown in Table 3.5, semantic tagging mainly reduces the errors of predicting wrong semantic symbols, while in some cases it can lead to a modest increase in the first type of errors. Overall, semantic tagging improves the prediction of individual semantic symbols even though this improvement does not always translate into more accurate queries.

## 3.6 Summary

In this chapter, we presented a two-stage decoding framework, aiming to improve compositional generalization in neural semantic parsing. Central to our approach is a semantic tagger which labels the input with semantic symbols representing the meanings of individual words. A neural sequence-to-sequence parsing model considers the input utterance *and* the predicted tag sequence to generate the final meaning representation. Our framework can be combined with different neural models and semantic formalisms. Experimental results on three semantic parsing datasets show that the proposed approach consistently improves compositional generalization across model architectures, domains, and semantic formalisms. It also demonstrates superior performance to related compositional generalization approaches (Andreas, 2020; Oren et al., 2020).

However, the success of semantic tagging requires well-defined local alignments between language expressions and logical expressions, which is not always present in real-world applications. Moreover, semantic tagging focuses on modeling lexical meanings and does not explicitly take into account syntactic relations. As a result, it is not expected to aid the generalization relating to novel syntactic structures. In the next chapter, we will propose a model attempting to tackle both problems.

# Chapter 4

# Disentangled Sequence-to-Sequence Learning

In the previous chapter, we propose a two-stage decoding framework without drastically compromising the flexibility of sequence-to-sequence models. However, there still exists one major obstacle that renders the framework less flexible and general compared to pure neural network systems. A prerequisite to semantic tagging is the presence of unambiguous local correspondences between language expressions and semantic symbols. However, this condition does not always hold in real-world applications. It can not handle cases where some fragments of the output do not have any alignment with the input (e.g., SQL-specific expressions in semantic parsing) or the alignment is more complex than a strict local one-to-one mapping (e.g., a many-to-many mapping in machine translation). As discussed in Section 2.2, more complex and global many-to-many mappings between words and meanings are ubiquitous in natural language. One word could participate in the shaping of multiple meaning units and likewise, one meaning unit could derive from multiple words (e.g., polysemy and anaphora).

As a result, tackling compositional generalization via pure architectural modifications without a symbolic component possesses some special benefits. It has the potential to fully maintain the robustness and flexibility of neural models required to process noisy and complex natural language. In this chapter, we thus assume that the incompetence in compositional generalization is (at least partly) due to the architecture design of neural models and investigate: *how can we devise a more competent neural model for compositional generalization?* To approach this problem, we design a new neural architecture for sequence-to-sequence modeling to learn more disentangled representations than standard sequence-to-sequence models (e.g., Transformer) for better

compositional generalization. The generality of the proposal allows us to apply the model to any sequence-to-sequence task. We therefore perform experiments on both semantic parsing and machine translation. Experimental results show that our proposal delivers more disentangled representations and better generalization.

## 4.1 Introduction

There is mounting evidence suggesting that the very popular sequence-to-sequence architectures struggle with compositional generalization (Finegan-Dollak et al., 2018; Lake and Baroni, 2018; Keysers et al., 2020; Herzig and Berant, 2021). In order to devise a more competent neural model for compositional generalization, we first look at the question: *what are possible factors that lead to this failure?*

One potential cause is spurious correlations that have been shown in different scenarios to hinder out-of-distribution generalization (Jia and Liang, 2017; Gururangan et al., 2018; Arjovsky et al., 2019; Sagawa et al., 2020). Models that rely on simple but predictive superficial clues (Weissenborn et al., 2017) can do well on in-distribution test sets, but lack robustness to distributional shifts. For example, Jia and Liang (2017) showed the existing well-performing neural systems for reading comprehension are brittle when evaluated on paragraphs that contain adversarially inserted sentences. Nevertheless, spurious correlation has been mostly exposed in the context of classification. It is unclear if the problem is the only obstacle to better generalization especially in more complex tasks like structured prediction. In this chapter, we identify an orthogonal *entanglement problem* with how different semantic factors (e.g., lexical meaning and semantic relations) are represented in neural sequence models that hurt generalization. Ideally, neural networks should represent semantic factors in a *disentangled* way by virtue of the principle of compositionality (Partee, 1995). As discussed in Section 2.2, the local (or strong) interpretation of compositionality suggests that semantic properties of syntactic constituents are, to a certain extent, conditionally independent of the context.

Disentanglement, i.e., the ability to uncover explanatory factors from data, is often cited as a key property of good representations (Bengio et al., 2013). For example, a model trained on 3D objects might learn factors such as object identity, position, scale, lighting, or color. Several types of variational autoencoders Kingma and Welling (2014) have been proposed for the unsupervised learning of disentangled representations in images (Higgins et al., 2017; Kim and Mnih, 2018; Chen et al., 2018). However,

some of the underlying assumptions of these models have come under scrutiny recently (Locatello et al., 2019).

Disentanglement for linguistic representations remains under-explored and has mostly focused on separating the style of text from its content (John et al., 2019; Cheng et al., 2020). In the context of sentence-level semantics, we argue that disentangled representations should be able to discriminate different semantic factors and neural units modeling a particular semantic factor should be relatively invariant to changes in other factors. Table 2.1 shows an example. The relation between "table" and "house" in the sentence "A boy likes the cake on the table in a house (beside the tree)" and its representation should not be affected by whether there is a PP modifying "house". However, we hypothesize that in a standard neural encoder (e.g., transformer-based) semantic factors tend to be entangled so that changes in one factor affect the representation of others. We further illustrate this problem in an artificial setting in Section 4.2 and find that a simple marking strategy enhances the learning of disentangled representations.

Motivated by this finding, we propose an extension to sequence-to-sequence models, which allows us to learn disentangled representations for compositional generalization. Specifically, at each time step of the decoding, we adaptively re-encode the source input by conditioning the source representations on the newly decoded target context. We therefore build specialized representations which make it easier for the encoder to exploit relevant-only information for each prediction and disentangle relevant source factors across different predictions. Experiments on three benchmarks across semantic parsing and machine translation, namely COGS (Kim and Linzen, 2020), CFQ (Keysers et al., 2020), and CoGnition (Li et al., 2021), empirically verify that our proposal leads to better generalization, outperforming competitive baselines and more specialized techniques.

## 4.2 Disentanglement in a Toy Experiment

We first shed light on the problem of entangled representations with a toy experiment and then move on to describe our modeling solution. For simplicity, we only focus on relations as the kind of semantic factors a model aims to represent, but the entanglement issue could also exist in representations of other factors, such as lexical meaning.

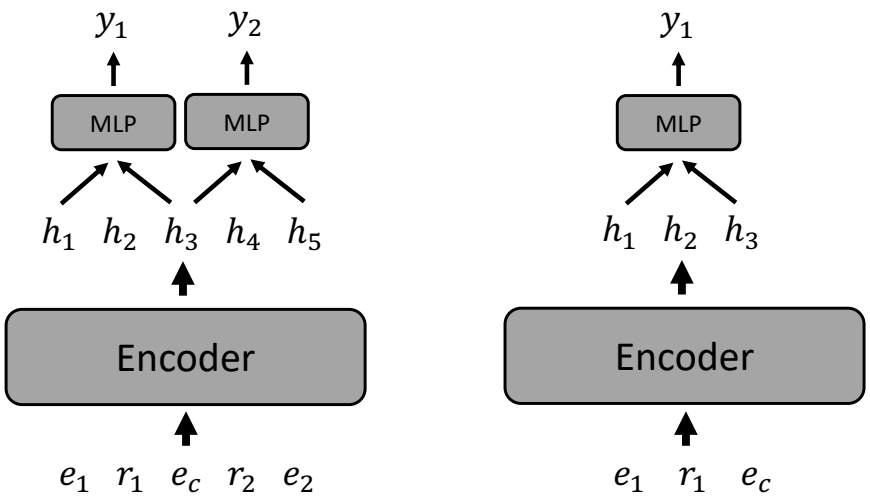


Figure 4.1: The model used in the toy experiment to predict the relation between $e_1$ and $e_c$ and the relation between $e_c$ and $e_2$.

**Data Creation** Let $x = [e_1, r_1, e_c, r_2, e_2]$ denote a sequence of symbols. We want to predict the relation between $e_1$ and $e_c$, and $e_c$ and $e_2$, which we denote by $y = (y_1, y_2)$, with $y_1 \in L_1$ and $y_2 \in L_2$ where $L_1$ are a set of relation labels for $y_1$ and $L_2$ are a set of relation labels for $y_2$. For simplicity, we set $e_1$, $e_c$, and $e_2$ to the same symbol $e$ (i.e., $e_1, e_c, e_2 \in \{e\}$) whereas $r_1 \in R_1$ and $r_2 \in R_2$ denote different relation symbols ($R_1$ and $R_2$ are the corresponding sets of relation symbols). In this toy setting, we will further assume that different relation symbols determine different relation labels (e.g., for the phrases "cat in house" and "cat with house", "in" and "with" represent two distinct relations between "cat" and "house"). In reality, relations between words could be dependent on broader context or not verbalized at all. We also assume that there is a one-to-one mapping between relation symbols and relation labels (i.e., between $L_1$ and $R_1$ and $L_2$ and $R_2$).

We construct a training set by including examples $[e_1, r_1, e_c, r_2, e_2]$ where $r_1$ is the same relation symbol throughout while $r_2$ can be any relation symbol in $R_2$ ($r_1 \in \{r_{train}\}$, $r_2 \in R_2$). We also include examples $[e_1, r_1, e_c]$ with all relation symbols from $R_1$ occurring in isolation ($r_1 \in R_1$). This way, the training set covers all primitive relations, but contains only a particular type of relation composition (i.e., $\{r_{train}\} \times R_2$). In contrast, the test set contains all unseen compositions $[e_1, r_1, e_c, r_2, e_2]$ (i.e., $r_1 \in R_1 \backslash \{r_{train}\}, r_2 \in R_2$) which will allow us to evaluate a model's ability to generalize. We set each relation set to include 10 relation symbols ($|R_1| = |R_2| = 10$).

Finally, we simplistically only consider the relations of the target word $e_c$ with its left and right words $e_1$ and $e_2$. In reality, a model would be expected to capture sentence-

level semantics, i.e., a word's relation to *all* context words in a sentence (including no relation).

**Modeling** For each input symbol, we sample a vector from a Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{0.2^2 I})$ and freeze it during training. As shown in Figure 4.1, we first embed each example $x$ into a sequence of vectors $[w_1, w_2, ..., w_n]$ (where $n = 3$ or $n = 5$) and transform them into contextualized representations $[h_1, h_2, .., h_n]$ using a Transformer encoder (Vaswani et al., 2017). To predict the relation between two symbols, we then concatenate their corresponding representations and feed the resulting vector to an MLP for classification.

To study how changes in relation $y_1$ affect the prediction of $y_2$ at test time, we explore two training methods. One is joint training where a model learns to predict both $y_1$ and $y_2$ (i.e., $h_1$ and $h_3$ are concatenated to predict $y_1$ or $h_3$ and $h_5$ are concatenated to predict $y_2$). The other method is separate training where a model is trained to only predict $y_2$ (i.e., only $h_3$ and $h_5$ are concatenated to predict $y_2$). For separate training, we basically ignore examples $[e_1, r_1, e_c]$ which only include $r_1$, as they have no bearing on the prediction of $y_2$.

**Observation** With separate training, the model learns to ignore $r_1$, the accuracy of predicting $y_2$ on the test set is 100%, regardless of which value $r_1$ takes. This indicates that random perturbation of $r_1$ alone does not lead to generalization failure. It also follows that there is no spurious correlation between $r_1$ and $y_2$. However, when the model is trained to predict both relations (which is what happens in realistic settings since we need to capture all possible relations) $r_1$ has a huge impact on the prediction of $y_2$ whose accuracy drops to approximately 55%. Taken together, these results suggest that the model fails to generalize to new relation compositions due to its internal representations being entangled and as a result, changes in one relation affect the representation of others.

Why is there a wide performance gap between joint and separate training? At test time the model processes the same utterance (no matter whether it is trained jointly or separately), and could in theory be susceptible to both $r_1$ and $r_2$. However, the induced representations show fundamentally different behaviors and remain invariant to $r_1$ with separate training. A possible explanation is that modern neural networks trained with stochastic gradient descent have a learning bias towards *simple* functions (Shah et al., 2020). When $r_1$ is not predictive of $y_2$, relying only on $r_2$ whilst remaining

invariant to $r_1$ constitutes a simpler function than making use of both $r_1$ and $r_2$. As a result, in separate training the model learns to ignore extraneous information, focusing exclusively on $r_2$. On the contrary, in joint training the target of predicting both $y_1$ and $y_2$ forces the hidden states (e.g., $h_3$) to capture information about both relations, leading to the entanglement problem discussed above.

**A Simple Solution** Although separate training presents a solution to entanglement, it is unrealistic for real-world data as it would be extremely inefficient to train separate models for each relation (the number of relations is quadratic with respect to sentence length). Instead, we explore a simple but effective approach where a single model takes as input an utterance enriched with different indicator features for different targets. Specifically, given utterance $[e_1, r_1, e_c, r_2, e_2]$, and assuming we wish to predict relation $y_1$, we add indicator feature 1 for symbols $e_1$, $r_1$, and $e_c$ (marking the relation and its immediate context), and 0 for all other symbols. The model then takes as input the utterance *and* relation indicators, i.e., $[1,1,1,0,0]$ for $y_1$ and $[0,0,1,1,1]$ for $y_2$, and learns embeddings for indicators during training. It thus learns specialized representations for *each* prediction rather than shared representations for *all* predictions. Based on the simplicity bias, the two representations will guide the model towards exclusively relying on $r_1$ and $r_2$, naturally disentangling different relations by encoding them separately. Such a model predicts $y_1$ with 100% test accuracy and $y_2$ with 97%.

## 4.3 Learning to Disentangle

While the marking strategy offers substantial benefits in learning disentangled relation representations, we typically do not have access to explicit labels indicating which words are helpful for predicting a specific relation. Nevertheless, the idea of learning representations specialized for different predictions (albeit with shared parameters) is general and could potentially alleviate the entanglement problem for compositional generalization.

Let $X = [x_1, x_2, ..., x_n]$ denote a source sequence. As described in Section 2.1, canonical sequence-to-sequence models like the Transformer model (Vaswani et al., 2017) first encode it into a sequence of contextualized representations which are then used to decode target symbols $Y = [y_1, y_2, ..., y_m]$ one by one. The same source encodings are used to predict all target symbols, and are therefore expected to capture all semantic factors in the input. However, these could be entangled as demonstrated in our analysis

above. To alleviate this issue, we propose to learn specialized source representations for different predictions by adaptively re-encoding the source input at every step of the decoding.

Specifically, at the $t$-th time step, we concatenate the source input with the previously decoded target and obtain the context for the current prediction $C_t = [x_1, x_2, ..., x_n, y_1, y_2, ..., y_{t-1}, [\text{PH}]]$ where [PH] is a placeholder (e.g., a mask token when using a pretrained encoder). $C_t$ is then fed to a standard encoder (e.g., the Transformer encoder) to obtain the contextualized representations $\mathbf{H}_t = [\mathbf{h}_{t,1}, \mathbf{h}_{t,2}, ..., \mathbf{h}_{t,n}]$:

$$\mathbf{H}_t = f_{\texttt{Adaptive\_Encoder}}(C_t) \tag{4.1}$$

The adaptive encoder could optionally consist of two components. $C_t$ is first fed to $k_1$ Transformer encoder layers to fuse the target information:

$$\bar{\mathbf{H}}_t = f_{\texttt{Adaptive\_Encoder}_1}(C_t) \tag{4.2}$$

where $\bar{\mathbf{H}}_t$ is a sequence of contextualized representations $[\bar{\mathbf{h}}_{t,1}, \bar{\mathbf{h}}_{t,2}, ..., \bar{\mathbf{h}}_{t,n}, \bar{\mathbf{h}}_{t,n+1}, ..., \bar{\mathbf{h}}_{t,n+t}]$. Then, optionally the first $n$ vectors corresponding to source tokens are extracted and fed to another $k_2$ Transformer encoder layers for further processing:

$$\mathbf{H}_t = f_{\texttt{Adaptive\_Encoder}_2}(\bar{\mathbf{H}}_t[:n]) \tag{4.3}$$

The key difference from the encoder in standard sequence-to-sequence models is that at each time step we adaptively *re-compute* source representations that condition on the newly decoded target $[y_1, y_2, ..., y_{t-1}]$. This way, the target context informs the encoder of predictions of interest at each time step. This simple modification unburdens the model from capturing all source information through a forward pass of encoding. Instead, based on the simplicity bias, the model tends to zero in on information relevant to the current prediction, remaining invariant to irrelevant details, thereby improving disentanglement. One might argue that the decoder in standard sequence-to-sequence models could also extract specialized information for each prediction (through the cross-attention mechanism). However, it would fail to do so when working with an entangled encoder that produces problematic representations for out-of-distribution examples and breaks down the decoding process.

We propose two strategies for exploiting the target-informed encoder. Firstly, we use a multilayer perceptron (MLP) to predict $y_t$ based on $\bar{\mathbf{h}}_{t,n+t}$:

$$p(y_t|X, Y_{<t}) = f_{\text{MLP}}(\bar{\mathbf{h}}_{t,n+t}) \tag{4.4}$$

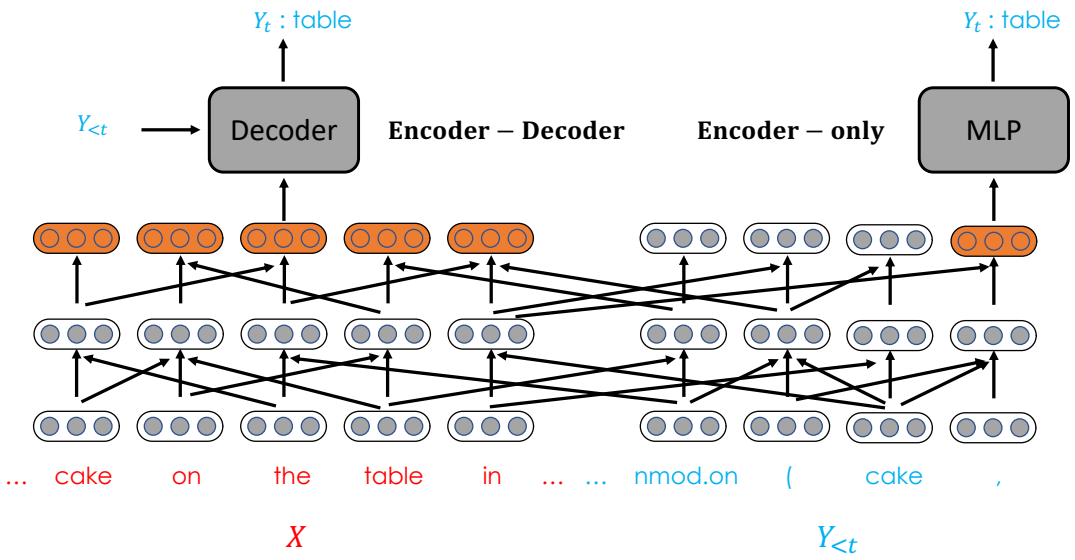


Figure 4.2: The proposed model at the $t$-th time step. It concatenates the source input $X$ with the previously decoded output $Y_{<t}$ and feeds it into a Transformer encoder to induce the target-informed adaptive source encodings. Then we either use an MLP or a Transformer decoder to predict the next token.

Secondly, we incorporate the proposed encoder into the standard encoder-decoder architecture: we take source encodings $H_t$ and feed them together with the previous target $[y_1, ..., y_{t-1}]$ to a standard decoder (e.g., Transformer-based) to predict $y_t$:

$$p(y_t|X, Y_{<t}) = f_{\texttt{Decoder}}(Y_{<t}, \mathbf{H}_t) \tag{4.5}$$

For complex tasks like machine translation, we empirically find that preserving the encoder-decoder architecture and adopting the extra second component for encoding is essential to achieving good performance.

We adopt the Transformer architecture to instantiate the encoder and decoder, however, the proposed method is generally applicable to any sequence-to-sequence model. The model is shown in Figure 4.2. We maintain separate position encodings for source and target symbols (e.g., $x_1$ and $y_1$ correspond to the same position). To differentiate between source and target content, we also add a source (target) type embedding to all source (target) token embeddings. Compared to the classical Transformer, our proposal increases the running time from $O(n^2 + m^2)$ to $O(m(n^2 + m^2))$ where $n$ is input length and $m$ is output length. In the next chapter, we will attempt to improve the efficiency of our approach.

## 4.4 Experiments: Semantic Parsing

In this section, we present our experiments for evaluating the proposed **D**isent**angle**d sequence-to-sequence model which we call Dangle. We refer to the two variants of Dangle as Dangle-enc and Dangle-encdec. In this chapter, as a proof of concept, we adopt synthetic benchmarks to isolate compositional generalization. We first focus on two fully synthetic semantic parsing benchmarks which target compositional generalization. Our second suite of experiments reports results on a partly synthetic machine translation benchmark for compositional generalization.

### 4.4.1 Datasets

Our semantic parsing experiments focus on two benchmarks. The first one is COGS (Kim and Linzen, 2020) which contains natural language sentences paired with logical forms based on lambda calculus (see an example in Table 2.4). In addition to the standard splits of Train/Dev/Test, COGS provides a generalization (Gen) set that covers five types of compositional generalization: interpreting novel combinations of primitives and grammatical roles, verb argument structure alternation, and sensitivity to verb class, interpreting novel combinations of modified phrases and grammatical roles, generalizing phrase nesting to unseen depths.

The former three fall into lexical generalization while the latter two require structural generalization. Interpreting novel combinations of modified phrases and grammatical roles involves generalizing from examples with PP modifiers within object NPs to PP modifiers within subject NPs. The generalization of phrase nesting to unseen depths is concerned with two types of nesting: nested CPs (e.g., [*Mary knows that* [*John knows* [*that Emma cooks*]$_{\text{CP}}$ ]$_{\text{CP}}$ ]$_{\text{CP}}$) and nested PPs (e.g., *Ava saw the ball* [*in the bottle* [*on the table*]$_{\text{PP}}$]$_{\text{PP}}$). The training set only contains nestings of depth 0–2, where depth 0 is a phrase without nesting. The generalization set contains nestings of strictly greater depths (3–12). The Train set includes 24,155 examples and the Gen set includes 21,000 examples.

Our second benchmark is CFQ (Keysers et al., 2020), a large-scale dataset specifically designed to measure compositional generalization. It contains 239,357 compositional Freebase questions paired with SPARQL queries (see an example in Table 2.4). CFQ was automatically generated from a set of rules in a way that precisely tracks which rules (atoms) and rule combinations (compounds) were used to generate each example. Using this information, the authors generate three splits with *maximum com-*

*pound divergence* (MCD) while guaranteeing a small atom divergence between train and test sets. In this dataset, atoms refer to entities and relations and compounds to combinations thereof. Large compound divergence indicates the test set contains many examples with unseen syntactic structures. We evaluate our model on all three splits. Each split consists of 95,743/11,968/11,968 train/dev/test examples.

### 4.4.2 Comparison Models

On COGS, we trained a baseline Transformer (Vaswani et al., 2017) with sinusoidal (absolute) and relative position embeddings (Shaw et al., 2018; Huang et al., 2020). We assessed the effect of pretraining on compositional generalization, by also fine-tuning BART-base (Lewis et al., 2020) on the same dataset. We created disentangled versions of Transformer adopting both the Dangle-enc and Dangle-encdec architectures. The pretrained version of our model only adopts the Dangle-encdec architecture to be compatible with BART-base.

We also compared with two models specifically designed for compositional generalization on COGS. The first one is Tree-MAML (Conklin et al., 2021), a meta-learning approach whose objective directly optimizes for out-of-distribution generalization. Their best-performing model uses tree kernel similarity to construct meta-train and meta-test task pairs. The second approach is LexLSTM (Akyurek and Andreas, 2021), an LSTM-based sequence-to-sequence model whose decoder is augmented with a lexical translation mechanism that generalizes existing copy mechanisms to incorporate learned, decontextualized, token-level translation rules. The lexical translation module is intended to disentangle lexical phenomena from syntactic ones.

Furrer et al. (2020) showed that pretrained sequence-to-sequence models are key to achieving good performance on CFQ. We compared against their T5-11B-mod model which obtained the best results among various pre-trained models. This is essentially a T5 model with 11B parameters fine-tuned on CFQ with intermediate representations (i.e., SPARQL queries are simplified to be structurally more aligned to the input for training and then post-processed to obtain the original valid SPARQL at inference time).

We built both variants of our model on top of RoBERTa (Liu et al., 2019) due to the effectiveness of pre-training on this dataset. The encoder-decoder variant (i.e., Dangle-encdec) uses RoBERTa as the encoder and a randomly initialized Transformer decoder. We do not build Dangle-encdec based on pretrained sequence-to-sequence models like BART, as the complex predicates in CFQ cause an extremely long target

| Model | OSM | CP | PP | Overall |
|---|---|---|---|---|
| Tree-MAML (Conklin et al., 2021) | 0.0 | 0.0 | 0.0 | 66.7 |
| LexLSTM (Akyurek and Andreas, 2021) | 0.0 | 0.0 | 1.3 | 82.1 |
| Transformer (ABS) | 0.0 | 3.4 | 8.9 | 85.5 |
| + Dangle-enc | 0.0 | 11.4 | 5.7 | 85.9 |
| + Dangle-encdec | 0.0 | 10.8 | 6.9 | 86.0 |
| Transformer (REL) | 0.0 | 0.0 | 0.0 | 83.3 |
| + Dangle-enc | 0.0 | 13.8 | 13.5 | 85.4 |
| + Dangle-encdec | 0.0 | 14.4 | 13.2 | 85.6 |
| BART-base | 0.0 | 11.5 | 9.0 | 85.1 |
| + Dangle-encdec | 0.0 | **26.5** | **60.6** | **88.3** |

Table 4.1: Exact-match accuracy on **COGS** by type of structural generalization and overall. OSM refers to generalizing from object modifier PPs to subject modifier PPs; CP and PP are recursion depth generalization for sentential complements and prepositional phrases. ABS denotes absolute position embeddings; REL denotes relative position embeddings

sequence when tokenized with the BART tokenizer. We instead use a custom decoder with RoBERTa, which allows for taking those predicates as atomic units and attaining a short average target length. To tease apart the effect of pretraining and the proposed approach, we implement another baseline, which uses RoBERTa as the encoder and likewise a Transformer decoder. Finally, we compared against HPD (Guo et al., 2020), a hierarchical poset decoding architecture which consists of three components: sketch prediction, primitive prediction, and traversal path prediction. This model is highly optimized for the CFQ dataset and achieves competitive performance.

We implemented all comparison models and Dangle with fairseq (Ott et al., 2019).

### 4.4.3 Model Configuration

On COGS, the small in-distribution development (Dev) set makes model selection extremely difficult and non-reproducible. We follow Conklin et al. (2021) and sample a small subset from the generalization (Gen) set denoted as 'Gen-Dev' for tuning hyper-parameters. Best hyper-parameters were used to rerun the model with 5 different random seeds for reporting final results on the Gen set. For the baseline Transformer, the layer number of encoder and decoders are both 2. The embedding dimension is 300.

| Model | 2 | | 3 | | 4 | | 5 | |
|---|---|---|---|---|---|---|---|---|
| | CP | PP | CP | PP | CP | PP | CP | PP |
| Transformer (ABS) | 3.4 | 8.9 | 1.2 | 6.6 | 0.8 | 5.5 | 3.1 | 8.2 |
| + Dangle-enc | 11.4 | 5.7 | 10.3 | 8.8 | 14.3 | 8.6 | 12.7 | 13.4 |
| Transformer (REL) | 0.0 | 0.0 | 0.0 | 0.6 | 0.1 | 2.5 | 1.4 | 4.6 |
| + Dangle-enc | 13.8 | 13.5 | 18.2 | 19.4 | 24.7 | 31.9 | 27.2 | 44.3 |

Table 4.2: Exact-match accuracy for CP and PP recursion on **different splits of COGS** (recursion depth with $[2-5]$ range). ABS denotes absolute position embeddings; REL denotes relative position embeddings.

The feedforward embedding dimension is 512. We use the same configuration for Transformer + Dangle-encdec. For Transformer + Dangle-enc, to maintain approximately identical model size with the baseline, we used the same embedding dimension and set the number of the encoding layers to 4. For all models, we initialized embeddings (on the both source and target side) with Glove (Pennington et al., 2014).

For the RoBERTa + Dangle model on CFQ, we use a separate target vocabulary; the target embedding matrix is randomly initialized and learned from scratch. RoBERTa-base is combined with a Transformer decoder that has 2 decoder layers with embedding dimension 256 and feedforward embedding dimension 512. All hyper-parameters are chosen based on validation performance. On CFQ, for both RoBERTa-base and RoBERTa+Dangle, results are averaged over 3 random seeds.

### 4.4.4 Results

Table 4.1 shows our results on COGS broken down by type of structural generalization and overall. All models achieve 0 accuracy on generalizing from PP object modifiers to PP subject modifiers. We find this is due to a predicate order bias. In all training examples, "agent" or "theme" come before preposition predicates like "in", so the models learn this spurious correlation and cannot generalize to cases where the preposition precedes the predicate.

Interestingly, a vanilla Transformer outperforms more complex approaches like Tree-MAML and LexLSTM. We conjecture the large discrepancy is mostly due to our use of Glove embeddings, which comparison systems do not use. Pretraining in general substantially benefits lexical generalization, our Transformer and BART-base models achieve nearly perfect accuracy on all such cases in COGS. An intuitive explanation is

| Model | MCD1 | MCD2 | MCD3 | Mean |
|---|---|---|---|---|
| T5-11B-mod (Furrer et al., 2020) | 61.6 | 31.3 | 33.3 | 42.1 |
| HPD (Guo et al., 2020) | 72.0 | 66.1 | 63.9 | 67.3 |
| RoBERTa-base | 60.6 | 33.6 | 36.0 | 43.4 |
| + Dangle-enc | 78.3 | 59.5 | 60.4 | 66.1 |
| + Dangle-encdec | 78.4 | 59.9 | 61.3 | 66.5 |

Table 4.3: Exact-match accuracy on **CFQ**, Maximum Compound divergence (MCD) splits. RoBERTa-base denotes a baseline model that uses RoBERTa as the encoder and a randomly initialized Transformer decoder. We do not build Dangle based on pre-trained sequence-to-sequence models like BART, as the complex predicates in CFQ cause an extremely long target sequence when tokenized with the BART tokenizer (90+ tokens on average). With RoBERTa, we use a custom decoder, which allows for taking those predicates as atomic units and obtaining a short average target length (about 30).

that pretrained embeddings effectively capture common syntactic roles for tokens of the same type (e.g., "cat" and "dog") and facilitate the generalization of the same decoding strategy to all of them. Dangle significantly improves generalization performance on CP and PP recursion when combined with our base Transformer and BART-base. Dangle-enc and Dangle-encdec perform similarly on top of the base Transformer.

To further show the potential of our proposal, we evaluated Transformer + Dangle-enc on additional COGS splits. Table 4.2 shows how model performance changes with exposure to progressively larger recursion depths. Given recursion depth $n$, we created a split by moving all examples with depth $\leq n$ from Gen to Train set. As can be seen, Transformer + Dangle-enc, especially the variant with relative embeddings, is continuously improving with exposure to additional training examples. In contrast, the vanilla Transformer does not seem to benefit from additional examples, even when relative position encodings are used. We can also explain why adding more recursion in training boosts generalization performance. In the original split, many nouns never occur in examples with recursion depth 2, which could tempt the model to exploit this kind of dataset bias for predictions. In contrast, seeing words in different contexts (e.g., different nesting depth) effectively reduces the possibility of learning these spurious correlations and therefore improves compositional generalization.

CFQ results are shown in Table 4.3. RoBERTa + Dangle substantially boosts the performance of RoBERTa, and is in fact superior to T5-11B-mod. This result highlights

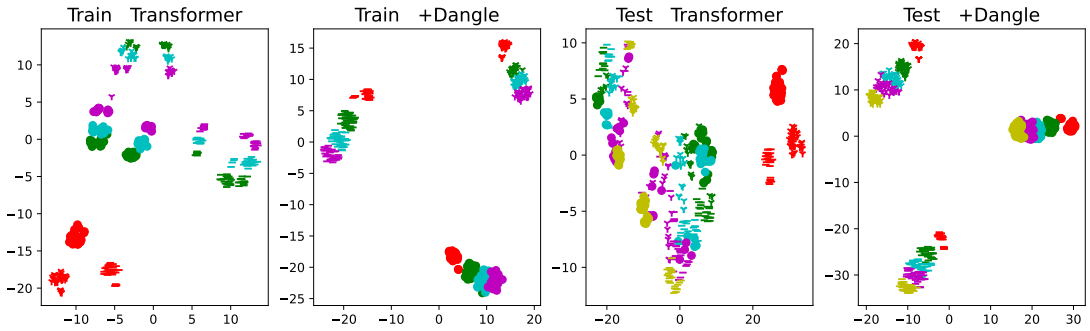


Figure 4.3: t-SNE visualization of hidden states corresponding to predicates “in”, “on”, and “beside” on training examples with PP recursion depth 4 and test examples with PP recursion depth 5. Different colors denote different recursion contexts and different shapes of markers correspond to different predicates.

the limitations of pretraining as a solution to compositional generalization underscoring the benefits of our approach. RoBERTa + Dangle is comparable to HPD which is a special-purpose architecture highly optimized for the CFQ dataset. On the contrary, Dangle is generally applicable to any sequence-to-sequence task including machine translation, as we will show in Section 4.5.

### 4.4.5 Analysis

As discussed in Section 4.2, we hypothesize that a neural model’s inability to perform compositional generalization partly arises from its internal representations being entangled. To verify this, we visualize the hidden representations for a Transformer model with and without Dangle. Specifically, we train both models on the 4th split of COGS (i.e., data with maximum PP recursion depth 4) and test on examples with PP recursion depth 5. Then, we extract the hidden states before the softmax layer used to predict the preposition predicates “in”, “beside”, and “on” and use t-SNE (van der Maaten and Hinton, 2008) to visualize them. Ideally, the representations of these prepositions should be invariant to the contexts accompanying them so that their prediction is not influenced by distribution shifts (e.g., contextual changes from PP recursion 4 to PP recursion 5).

The visualization is shown in Figure 4.3. Different colors correspond to different recursion depths while different shapes of markers denote different prepositions (e.g., for a training example like “NP in NP in NP in NP in NP”, the hidden states corresponding to the four “in” prepositions have the same marker but different colors). In training,

| Model | COGS | | | CFQ | | |
|---|---|---|---|---|---|---|
| | IntraV | InterV | ↓ R | IntraV | InterV | ↓ R |
| Transformer | 0.24 | 0.64 | 0.37 | 0.25 | 1.13 | 0.22 |
| + Dangle-enc | 0.19 | 0.73 | 0.26 | 0.01 | 0.52 | 0.01 |
| Transformer | 0.28 | 0.44 | 0.63 | 0.32 | 1.06 | 0.30 |
| + Dangle-enc | 0.23 | 0.54 | 0.42 | 0.04 | 0.48 | 0.08 |

Table 4.4: Entanglement for Transformer and our approach (+Dangle-enc) on COGS and CFQ (for which both models employ a RoBERTa encoder). Results for training/test set in first/second block. Intra/InterV denotes intra/inter-class variance and R is their ratio.

Transformer's hidden states within the same preposition scatter more widely compared to those of Dangle, which implies that its internal representations conflate information about a preposition's context with itself. In other words, Transformer's hidden states capture more context variations *in addition to* variations corresponding to the predicate of interest. This in turn causes catastrophic breakdown on the test examples, where Transformer's hidden states cannot discriminate context from predicate information at all. This is in stark contrast with Dangle, where information about predicates is preserved even in the presence of unseen contexts.

We further design a metric to quantify entanglement in neural representations drawing inspiration from Kim and Mnih (2018). Their metric assumes the ground-truth factors of a dataset are given, and it is applied to images with one factor fixed and all other factors varying randomly; if the representation is perfectly disentangled, the dimension with the lowest variance should correspond to the fixed factor. Since in our setting we do not have access to ground-truth factors, we assume the variable-length target token sequence is the factor of interest. We also do not need to perform a mapping between neurons and factors, because their correspondence is hard-coded in sequence-to-sequence models (e.g., a predicate and the hidden units used to predict it).

For each predicate $y$ occurring in different examples $e$, we extract all corresponding representations $\{\mathbf{v}_{e,y}\}$, i.e., the last layer of the hidden states used to predict $y$, and compute the empirical variance $\mathrm{Var}_e(\mathbf{v}^i_{e,y})$ for each $y$; we compute *intra-class* variance as the average of all predicates' variance weighted by their respective frequency:

$$V_{intra} = \frac{1}{d}\sum_{i=1}^{d} \mathbb{E}_y \mathrm{Var}_e(\mathbf{v}^i_{e,y}) \tag{4.6}$$

where $d$ is the dimension of hidden states and $\mathbb{E}$ is the weighted average of their

| *Training Set* |
|---|
| en: That winter, Taylor barely moved from the fire. |
| zh: 那年冬天，泰勒几乎没有从大火中挪动过。 |
| *Test Set* |
| en: That winter, the dog he liked barely moved from the fire. |
| zh: 那年冬天，他喜欢的狗狗几乎没有从火堆里挪动过。 |

Table 4.5: A training and test example from the CoGnition dataset. The test example is constructed by embedding the synthesized novel compound "the dog he liked" into the template extracted from the training example "That winter, [NP] barely moved from the fire.".

variances. Intuitively, if the representations are perfectly disentangled, they should remain invariant to context changes and intra-class variance should be zero.

We also measure *inter-class* variance by taking the mean of $\mathbf{v}_{e,y}$ for each predicate $y$ and then computing the variance of the means:

$$V_{inter} = \frac{1}{d}\sum_{i=1}^{d} \mathrm{Var}_y \, \mathbb{E}_e(\mathbf{v}^i_{e,y}) \tag{4.7}$$

Inter-class variance, on the contrary, should be relatively large for these hidden states, because they are intended to capture class variations. The ratio of intra- and inter-class variance collectively measures entanglement.

As shown in Table 4.4, representations in Dangle consistently obtain lower intra- to inter-class ratios than baseline models on both COGS and CFQ on both training and test sets.

## 4.5 Experiments: Machine Translation

### 4.5.1 Dataset

We also applied our approach to CoGnition (Li et al., 2021), a recently released semi-natural compositional generalization dataset targeting machine translation. This benchmark includes 216K English-Chinese sentence pairs; source sentences were taken from the Story Cloze Test and ROCStories Corpora (Mostafazadeh et al., 2016, 2017) and target sentences were constructed by post-editing the output of a machine translation engine. It also contains a synthetic test set to quantify and analyze compositional

generalization of neural MT models. This test set includes 10,800 sentence pairs, which were constructed by embedding synthesized novel compounds into training sentence templates. Table 4.5 shows an example. Each newly constructed compound is combined with 5 different sentence templates, so that every compound can be evaluated under 5 different contexts.

### 4.5.2 Comparison Models

We compared our model to a Transformer translation model following the same setting and configuration of Li et al. (2021). We adopted the encoder-decoder architecture variant of our approach (i.e., Dangle-encdec), as the encoder-only architecture performed poorly possibly due to the complexity of the machine translation task. The number of parameters was kept approximately identical to the Transformer baseline for a fair comparison. All models were implemented using fairseq (Ott et al., 2019). More details are provided below.

We adopted a Transformer translation model consisting of a 6-layer encoder and a 6-layer decoder with hidden size 512. Each training batch includes 8,191 tokens at maximum. This model was trained for 100,000 steps and we chose the best checkpoint on the validation set for evaluation. Again, we experimented with sinusoidal (absolute) and relative position embeddings.

We used the same hyperparameters as the baseline model except for the number of layers which we tuned on the validation set; for relative position embeddings, the encoder has 4 vanilla source-only Transformer encoder layers on top of 4 target-informed Transformer encoder layers (i.e., $k_1 = 4$ and $k_2 = 4$) and the decoder has 4 Transformer decoder layers; for absolute position embeddings, the encoder has 4 vanilla source-only Transformer encoder layers on top of 2 target-informed Transformer encoder layers (i.e., $k_1 = 2$ and $k_2 = 4$) and the decoder has 6 Transformer decoder layers. For a fair comparison, we also experimented with 8 encoder layers and 4 decoder layers for the Transformer baseline and found that it performs similarly to the standard 6-layer architecture.

### 4.5.3 Results

As shown in Table 4.6, + Dangle-encdec improves over the base Transformer model by 1.2 BLEU points when relative position embeddings are taken into account. In addition to BLUE, Li et al. (2021) evaluate compositional generalization using novel

| Model | ↓ $\text{ErrR}_{\text{Inst}}$ | ↓ $\text{ErrR}_{\text{Aggr}}$ | ↑ BLEU |
|---|---|---|---|
| Transformer (ABS) | 29.4 | 63.8 | 59.4 |
| + Dangle-encdec | 24.4 | 55.5 | 59.7 |
| Transformer (REL) | 30.5 | 63.8 | 59.4 |
| + Dangle-encdec | **22.8** | **50.6** | **60.6** |

Table 4.6: BLEU and compound translation error rates (ErrR) on the compositional generalization test set. Subscript Inst denotes instance-wise error rate while Aggr denotes aggregate error over 5 contexts. All results are averaged over 3 random seeds. ABS denotes absolute position embeddings; REL denotes relative position embeddings.

compound translation error rate which is computed over instances and aggregated over contexts. + Dangle-encdec variants significantly reduce novel compound translation errors both across instances and on aggregate by as much as 10 absolute accuracy points (see first two columns in Table 4.6). Across metrics, our results show that + Dangle-encdec variants handle compositional generalization better than the vanilla Transformer model.

### 4.5.4 Analysis

Two natural questions emerge given the substantial gain achieved by Dangle on the compositional generalization (CG) test set: (a) Is this gain related to our treatment of the entanglement problem? and (b) How does entanglement manifest itself in machine translation? We attempt to answer these questions with an example.

In the CG test set, five new utterances are constructed by embedding the novel compound ”behind the small doctor on the floor” into five sentence templates. In the training set, the phrases “behind the [ADJ] [NOUN]” and “the [ADJ] [NOUN] on the floor” appear frequently, but the phrase “behind the [ADJ] [NOUN] the [ADJ] [NOUN]” is very rare. This poses a serious challenge for the baseline encoder-decoder model, which mistakenly translates the compound phrase into 地板后面的小医生 (the small doctor behind the floor), or 地板上的小医生 (the small doctor on the floor), or altogether ignores the translation of some content words like 地板后面 (behind the floor). It seems the baseline model cannot simultaneously represent the relation between “behind” and “the small doctor” and the relation between “the small doctor” and “the floor”, even though the two are conditionally independent. In contrast, Dangle generates the correct translation 地板上的小医生后面 in all five contexts. We believe

this is due to the proposed adaptive encoding mechanism and its ability to decompose the representation problem of an unfamiliar compound phrase into sub-problems of familiar phrases (i.e, “behind the small doctor” and “the small doctor on the floor”).

## 4.6 Summary

In this chapter, we first identified an entanglement problem with how different semantic factors (e.g., lexical meaning and semantic relations) are represented in neural sequence models that hurts generalization. Then we proposed an extension to sequence-to-sequence models which allows us to learn more disentangled representations for compositional generalization. We have argued that taking into account the target context makes it easier for the encoder to exploit specialized information for improving its predictions. Experiments on semantic parsing and machine translation have shown that our proposal improves compositional generalization without any model-, dataset-, or task-specific modification. Despite the promising performance, we exclusively evaluate the proposal on synthetic benchmarks to isolate compositional generalization. In the later chapters, we will extend the proposed model and apply it to real-world language tasks.

# Chapter 5

# A Real-world Compositional Generalization Challenge

In the previous chapter, we evaluated Dangle on two synthetic semantic parsing benchmarks, namely COGS (Kim and Linzen, 2020) and CFQ (Keysers et al., 2020), and one semi-natural machine translation benchmark, namely CoGnition (Li et al., 2021). Synthetic datasets are typically generated via context-free grammars with a small lexicon and can not represent the full complexity and noise of natural language. As a general architectural innovation, Dangle has the potential of improving compositional generalization while fully maintaining the robustness and flexibility of neural models required to process real language. Thus, we would like to extend and apply Dangle to real-world settings involving both complex and noisy natural language and compositional generalization.

Before doing so, we need to first address the evaluation problem: what is an appropriate benchmark for evaluating real-world compositional generalization? In this chapter, we develop a new methodology for detecting examples representative of compositional generalization in naturally occurring text to better emulate a real-world setting. Based on the proposed methodology, we create a real-world machine translation challenge to complement the existing benchmarks.

## 5.1 Introduction

Models of compositional generalization are as good as the benchmarks they are evaluated on. Several existing benchmarks are made of artificially synthesized examples using a grammar or rules to systematically control for different types of generalization

(Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020; Li et al., 2021). Unfortunately, synthetic datasets lack the complexity of real natural language and may lead to simplistic modeling solutions that do not generalize to real-world settings (Dankers et al., 2022). Other benchmarks focus on naturally occurring examples but create train-test splits based on the properties of formal meaning representations of examples (e.g., logical forms, Finegan-Dollak et al. (2018); Shaw et al. (2021)). Unfortunately, formal annotations of meaning are not readily available for tasks beyond semantic parsing. Since compositional generalization is a general problem, it is desirable to define it based on natural language without being limited to semantic parsing and the availability of formal annotations.

In this chapter, we develop a new methodology for *detecting* examples representative of compositional generalization in naturally occurring text. Given a training and test corpora: (a) we discard examples from the test set that contain out-of-vocabulary (OOV) or rare words (in relation to training) to exclude novel atoms which are out of scope for compositional generalization; (b) we then measure how compositional a certain test example is *with respect to* the training corpus; we introduce a metric which allows us to identify a candidate pool of highly compositional examples; (c) using uncertainty estimation, we further select examples from the pool that are both compositional in terms of surface form and challenging in terms of generalization difficulty. Following these three steps, we create a *machine translation* benchmark using the IWSLT 2014 German-English dataset as our training corpus and the WMT 2014 German-English shared task as our test corpus. Analysis on the created benchmark shows that it contains much more diverse patterns in terms of lexical and syntactic composition than existing artificial benchmarks.

## 5.2 The ReaCT Dataset

It is fair to assume that a SOTA model deployed in the wild (e.g., a Transformer-based translation system) will be constantly presented with new test examples. Many of them could be similar to seen training instances or compositionally different but in a way that does not pose serious generalization challenges. An ideal benchmark for evaluating compositional generalization should therefore consist of phenomena that are of practical interest while challenging for SOTA models. To this end, we create ReaCT, a new **REA**l-world dataset for **C**ompositional generalization in machine **T**ranslation. Our key idea is to obtain a generalization test set by *detecting* compositional patterns in relation

| Selected | Examples | Compositional Degree | Uncertainty |
|---|---|---|---|
| ✗ | but what can we do about this ? | 2 / 8 = 0.25 | — |
| ✗ | please report all changes here . | 5 / 6 = 0.83 | 0.054 |
| ✔ | you have disabled your javascript ! | 5 / 6 = 0.83 | 0.274 |

Table 5.1: Candidate examples from the WMT corpus. Different $n$-grams previously seen in the IWSTL training corpus are highlighted in color. The first example is composed of two $n$-grams (*but* and *what can we do about this?*) with a compositional degree 0.25, and is discarded in the second stage. The second example has a high compositional degree but receives a low uncertainty score, and is thus filtered in the third stage. The third example is high in terms of both compositional degree and uncertainty, and is included in the compositional test set.

to an existing training set from a large and diverse pool of candidates. Specifically, we use the IWSLT 2014 German → English dataset as our training corpus. The IWSLT 2014 De→En dataset consists of approximately 170K sequence pairs. We used the fairseq script `prepare-iwslt14.sh` to randomly sample approximately 4% instances of this dataset as validation set and kept the rest as training set. Following standard practice, we created an in-domain test set, the concatenation of files dev2010, dev2012, tst2010, tst2011, and tst2012. We use the WMT 2014 German → English shared task as our test corpus and detect from the pool of WMT instances those that exemplify compositional generalization with respect to IWSLT. This procedure identifies naturally occurring compositional patterns which we hope better represent practical generalization requirements than artificially constructed challenges.

### 5.2.1 Compositional Test Set

In the following, we describe how we identify examples that demand compositional generalization. While we create our new benchmark with machine translation in mind, our methodology is general and applicable to other settings such as semantic parsing. For instance, we could take a relatively small set of annotated user queries as our training set and create a generalization challenge from a large pool of unlabeled user queries.

**Filtering Out-of-Vocabulary Atoms** Compositional generalization involves generalizing to *new* compositions of *known* atoms. The WMT corpus includes many new

semantic and syntactic atoms that are not attested in IWSLT. A large number of these are out-of-vocabulary (OOV) words which are by definition unknown and out of scope for compositional generalization. We thus discard WMT examples with words occurring less than 3 times in the IWSLT training set which gives us approximately a pool of 1.3M examples. For simplicity, we do not consider any other types of new atoms such as unseen word senses or syntactic patterns.

**Measuring Compositionality** How to define the notion of compositional generalization is a central question in creating a benchmark. Previous definitions have mostly centered around linguistic notions such as constituents or context-free grammars (Kim and Linzen, 2020; Keysers et al., 2020; Li et al., 2021; Shaw et al., 2021). These notions are appropriate for synthetic examples or logical forms as their underlying hierarchical structures are well-defined and can be obtained with ease.

Since we do not wish to synthesize artificial examples or be limited to formal language but rather detect real-world utterances, relying on the notion of constituent might be problematic. Sentences in the wild are often noisy and ungrammatical and it is far from trivial to analyze their syntactic structure so as to reliably identify new compositions of known constituents. We overcome this problem by devising a metric based on n-gram matching which assesses how compositional a certain example is with respect to a training corpus.

Specifically, we first create a lookup dictionary of atomic units by extracting all $n$-grams that occur more than 3 times in the training corpus. Given a candidate sentence, we search the dictionary for the minimum number of $n$-grams that can be composed to form the sentence. For example, for sentence "$x_1x_2x_3x_4x_5$" and dictionary $(x_1, x_2, x_3x_4, x_5, x_1x_2, x_3x_4x_5, )$, the minimum set of such $n$-grams is $(x_1x_2, x_3x_4x_5)$. A sentence's *compositional degree* with respect to the training corpus is defined as the ratio of the minimum number of $n$-grams to its length (e.g., 2/5 = 0.4 for the above example). We select the top 60,000 non-overlapping examples with the highest compositional degree as our *candidate pool*. As we will discuss in the next chapter ( Section 6.3.3), compositional degree further allows us to examine at a finer level of granularity how model performance changes as test examples become increasingly compositional.

**Estimating Uncertainty** Examples with the same compositional degree could pose more or less difficulty to neural sequence models (see last two utterances in Table 5.1). Ideally, we would like to identify instances that are compositional in terms of the surface

| Dataset | # examples | Comp Degree | Word $n$-gram 2 | Word $n$-gram 3 | POS $n$-gram 2 | POS $n$-gram 3 |
|---|---|---|---|---|---|---|
| COGS | 21,000 | 0.392 | 6,097 | 24,275 | 12 | 27 |
| CoGnition | 10,800 | 0.502 | 1,865 | 13,344 | 1 | 38 |
| CFQ | 11,968 | 0.268 | 168 | 2,736 | 8 | 30 |
| ReaCT | 3,000 | 0.811 | 19,315 | 33,652 | 76 | 638 |

Table 5.2: Dataset Statistics: unique novel $n$-grams computed over words and parts of speech (POS) in ReaCT, and test partitions of COGS, CoGnition, and CFQ benchmarks. ReaCT shows a much higher compositional degree and more diverse n-gram patterns compared to other benchmarks.

form *and* hard in terms of the underlying generalization (see the third example in Table 5.1). We detect such examples using a metric based on *uncertainty estimation* and orthogonal to compositional degree. We quantify predictive uncertainty based on model ensembles, a method that has been successfully applied to detecting misclassifications and out-of-distribution examples (Lakshminarayanan et al., 2017; Malinin and Gales, 2021).

We follow the uncertainty estimation framework introduced in Malinin and Gales (2021) for sequence prediction tasks. Specifically, we train $M = 10$ Transformer models with different random initializations on IWSLT (our training corpus). Give the ensemble of models $\{P(Y|X;\theta^{(m)})\}_{m=1}^{M}$ whose parameters are assumed to be sampled from an approximate posterior $q(\theta|\mathcal{D})$, the *predictive posterior* is obtained by taking the expectation over the ensemble:

$$P(Y|X,\mathcal{D}) = \quad \mathbb{E}_{q(\theta|\mathcal{D})}\left[P(Y|X,\theta)\right] \approx \frac{1}{M}\sum_{m=1}^{M} P(Y|X,\theta^{(m)}) \tag{5.1}$$

Malinin and Gales (2021) introduced *reverse mutual information* (RMI) between each model and the predictive posterior to measure diversity or disagreement between models in the ensemble :

$$\mathcal{M}\left[Y,\theta|X,\mathcal{D}\right] = \mathbb{E}_{q(\theta|\mathcal{D})}\left[\mathbb{E}_{P(Y|X,\mathcal{D})}\left[\ln\frac{P(Y|X,\mathcal{D})}{P(Y|X,\theta)}\right]\right] \tag{5.2}$$

The assumption is that model disagreement on examples reflects *knowledge uncertainty* which is the models' uncertainty in predictions due to lack of understanding of the task rather than *data uncertainty*, the intrinsic uncertainty associated with the task (e.g., a sentence could have multiple correct translations).

Since it is intractable to evaluate the expectation over all $Y$s due to the combinatorial explosion of the hypothesis space, in practice, we approximate it using 10 samples from the predictive posterior for the ensemble via a *product-of-expectations*:

$$P(Y|X,\mathcal{D}) = \prod_{l=1}^{m} \mathbb{E}_{q(\theta|\mathcal{D})}\Big[P(y_l|Y_{<l},X,\theta)\Big] \tag{5.3}$$

which is implemented in fairseq (Ott et al., 2019) to ensemble models for generation.

We run inference over the candidate pool created in the previous stage; for each example in this pool, we compute the knowledge uncertainty. Note that the computation does not require access to gold targets. Thus, our approach could be used to select examples from a large pool of unannotated candidates.

We empirically found that the most uncertain examples are extremely noisy and barely legible (e.g., they include abbreviations, typos, and non-standard spelling). We thus discard the top 2,000 uncertain examples and randomly sample 3,000 instances from the next 18,000 most uncertain examples in an attempt to create a generalization test set with diverse language patterns and different levels of uncertainty.

### 5.2.2 Analysis

In this section, we analyze the compositional nature of ReaCT by comparing it to several popular benchmarks. Specifically, for all datasets, we count the number of novel test set $n$-grams that have not been seen in the training. We extract $n$-grams over words and parts of speech (POS); word-based $n$-grams represent more superficial lexical composition while $n$-grams based on POS tags reflect more of syntactic composition.

As shown in Table 5.2, despite being considerably smaller compared to other benchmarks (see # examples column), ReaCT presents substantially more diverse patterns in terms of lexical and syntactic composition. It displays a much bigger number of novel word $n$-grams, which is perhaps not surprising. Being a real-world dataset, it has a larger vocabulary and more linguistic variation. While our dataset creation process does not explicitly target novel syntactic patterns (approximated by POS $n$-grams), ReaCT still includes substantially more compared to other benchmarks. This suggests that it captures the complexity of real-world compositional generalization to a greater extent than what is achieved when examples are synthesized artificially. We show ReaCT examples with novel POS $n$-gram compositions in Table 5.3. These are novel syntactic patterns approximated by POS $n$-grams.

| Train | Test |
|---|---|
| • and i can 't believe you 're here and that i 'm meeting you here at ted . ( PRP RB IN NN . )<br>• you see , this is what india is today . the ground reality is based on ( DT NN NN VBZ VBN IN ) a cyclical world view . | the account data is provided to you directly via e-mail . |
| • a couple of hours ( DT NN IN NNS ) later , the sun will shine on the next magnifying glass .<br>• but this could also be used for good . ( MD RB VB VBN IN NN . ) | both setting of tasks must successfully be mastered under supervision . |
| • the national science foundation , other countries ( JJ NN NN , JJ NN ) are very interested in doing this<br>• no , they are full of misery . ( VBP JJ IN NN .) | its warm water temperature , small depth are convenient for bathing . |

Table 5.3: Novel syntactic compositions in ReaCT test set (syntactic atoms of the same type are color coded). POS-tag sequences for these atoms are shown in parentheses (PRP: pronoun, RB: adverb, IN: preposition, NN/S: noun singular/plural, DT: determiner, JJ: adjective, MD: modal, VBZ/P: non-/3rd person singular present, VBN: verb past participle.)

## 5.3 Summary

In this chapter, we proposed a methodology for identifying compositional patterns in real-world data to represent practical generalization requirements. Given a training and test corpora: (a) we discard examples from the test set that contain out-of-vocabulary (OOV) or rare words (in relation to training) to exclude novel atoms; (b) we then measure how compositional a certain test example is *with respect to* the training corpus; (c) using uncertainty estimation, we further select examples from the pool that are both compositional in terms of surface form and challenging in terms of generalization difficulty. Based on our methodology, we create a machine translation benchmark using the IWSLT 2014 German-English dataset as our training corpus and detecting 3,000 highly compositional examples from the WMT 2014 German-English training corpus as our test set. We show that the constructed test set contains much more diverse compositional patterns than existing artificial benchmarks. In the next chapter, we will use the created benchmark to assess models' performance on real-world compositional generalization.

# Chapter 6

# Real-world Disentangled Sequence-to-Sequence Learning

The main motivation for Dangle (see Chapter 4) was to advance compositional generalization while maintaining the robustness and flexibility of neural models required to process real language. However, in Chapter 4 we only evaluated Dangle on synthetic benchmarks in order to isolate compositional generalization. Therefore, it is still unclear whether Dangle is effective in real-world settings involving both complex and noisy natural language and compositional generalization. In this chapter, we aim to extend Dangle and apply it to real-world language tasks, including ReaCT developed in the previous chapter.

To this end, we propose a new variant of Dangle, which adaptively re-encodes keys periodically, at some interval. Our modifications encourage learning more disentangled representations more efficiently. We benchmark the proposed model on existing datasets (Andreas et al., 2020; Li et al., 2021) and the ReaCT benchmark. Experimental results demonstrate that our new architecture achieves better generalization performance across tasks and datasets and is adept at handling real-world challenges.

## 6.1 Introduction

In this chapter. we aim to advance real-world compositional generalization via general neural architectural modifications which are applicable to a wide range of real tasks. Our starting point is Dangle introduced in Chapter 4, a sequence-to-sequence model that learns more **D**isent**angle**d representations by adaptively re-encoding (at each time step) the source input. For each decoding step, Dangle learns specialized source encodings

by conditioning on the newly decoded target which leads to better compositional generalization compared to vanilla Transformers where source encodings are shared throughout decoding. Although promising, the results presented in Chapter 4 are based on synthetic datasets, leaving open the question of whether Dangle is effective in real-world settings involving both complex and noisy natural language and compositional generalization.

We present two key modifications to Dangle which encourage learning *more disentangled* representations *more efficiently*. The need to perform re-encoding at each time step substantially affects Dangle's training time and memory footprint. It becomes prohibitively expensive on datasets with long target sequences, e.g., programs with 400+ tokens in datasets like SMCalFlow (Andreas et al., 2020). To alleviate this problem, instead of adaptively re-encoding at each time step, we only re-encode periodically, at some interval. Our decoder is no different from a vanilla Transformer decoder except that it just re-encodes once in a while in order to update its history information. Our second modification concerns disentangling the representations of source keys and values, based on which the encoder in Dangle (and also in Transformers) passes source information to the decoder. Instead of computing keys and values using shared source encodings, we disassociate their representations: we encode source *values once* and re-encode *keys periodically*.

We evaluate the proposed model on existing benchmarks (Andreas et al., 2020; Li et al., 2021) and the ReaCT benchmark introduced in the previous chapter. Experimental results demonstrate that our new architecture achieves better generalization performance across tasks and datasets and is adept at handling real-world challenges. Machine translation experiments on a diverse corpus of 1.3M WMT examples show it is particularly effective for long-tail compositional patterns.

## 6.2 The R-Dangle Model

In this section, we describe the proposed model, which we call R-Dangle as a shorthand for **R**eal-world **D**isent**angle**d Transformer.

### 6.2.1 Background: The Dangle Model

We first briefly recap Dangle introduced in Chapter 4, focusing on the encoder-decoder architecture which delivers better performance on complex tasks like machine transla-

tion.

Let $X = [x_1, x_2, ..., x_n]$ denote a source sequence; let $f_{\texttt{Encoder}}$ and $f_{\texttt{Decoder}}$ denote a Transformer encoder and decoder, respectively. $X$ is first encoded into a sequence of contextualized representations $\mathbf{H}$:

$$\mathbf{H} = f_{\texttt{Encoder}}(X) \tag{6.1}$$

which are then used to decode target tokens $[y_1, y_2, ..., y_m]$ one by one. At the $t$-th decoding step, the Transformer takes $y_{t-1}$ as input, reusing the source encodings $\mathbf{H}$ and target memory $\mathbf{M}_{t-1}$ which contains the history hidden states of all decoder layers corresponding to past tokens $[y_1, y_2, ..., y_{t-2}]$:

$$y_t, \mathbf{M}_t = f_{\texttt{Decoder}}(y_{t-1}, \mathbf{H}, \mathbf{M}_{t-1}) \tag{6.2}$$

This step not only generates a new token $y_t$, but also updates the internal target memory $\mathbf{M}_t$ by concatenating $\mathbf{M}_{t-1}$ with the newly calculated hidden states corresponding to $y_{t-1}$.

Dangle differs from vanilla Transformers in that it concatenates the source input with the previously decoded target to construct target-dependent input for *adaptive* decoding:

$$C_t = [x_1, x_2, ..., x_n, y_1, y_2, ..., y_{t-1}, [\text{PH}]] \tag{6.3}$$

$$\mathbf{H}_t = f_{\texttt{Adaptive\_Encoder}}(C_t) \tag{6.4}$$

The adaptive encoder consists of two components. $C_t$ is first fed to $k_1$ Transformer encoder layers to fuse the target information:

$$\bar{\mathbf{H}}_t = f_{\texttt{Adaptive\_Encoder}_1}(C_t) \tag{6.5}$$

where $\bar{\mathbf{H}}_t$ is a sequence of contextualized representations $[\bar{\mathbf{h}}_{t,1}, ..., \bar{\mathbf{h}}_{t,n}, \bar{\mathbf{h}}_{t,n+1}, ..., \bar{\mathbf{h}}_{t,n+t}]$. Then, the first $n$ vectors corresponding to source tokens are extracted and fed to another $k_2$ Transformer encoder layers for further processing:

$$\mathbf{H}_t = f_{\texttt{Adaptive\_Encoder}_2}(\bar{\mathbf{H}}_t[:n]) \tag{6.6}$$

Finally, the adaptive source encodings $\mathbf{H}_t$ together with the target context $Y_{<t} = [y_1, y_2, ..., y_{t-1}]$ are fed to a Transformer decoder to predict $y_t$:

$$y_t, \mathbf{M}_t = f_{\texttt{Decoder}}(Y_{<t}, \mathbf{H}_t, \{\}) \tag{6.7}$$

In a departure from vanilla Transformers, Dangle does not reuse the target memory from previous steps, but instead re-computes *all* target-side hidden states based on new source encodings $\mathbf{H}_t$.

Similarly to Transformers, Dangle accesses source information at each decoding step via encoder-decoder attention layers where the same encodings $\mathbf{H}_t$ are used to compute both keys $\mathbf{K}_t$ and values $\mathbf{V}_t$:

$$\mathbf{K}_t = \mathbf{H}_t W^K \tag{6.8}$$

$$\mathbf{V}_t = \mathbf{H}_t W^V \tag{6.9}$$

$$\mathbf{O}_t = \text{Attention}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t) \tag{6.10}$$

where key and value projections $W^K$ and $W^V$ are parameter matrices; and $\mathbf{Q}_t$, $\mathbf{K}_t$, $\mathbf{V}_t$, and $\mathbf{O}_t$ are respectively query, key, value, and output matrices, at time step $t$.

### 6.2.2 Re-encoding at Intervals

The need to perform re-encoding (and also re-decoding) at each time step substantially increases Dangle's training cost and memory footprint, so that it becomes computationally infeasible for real-world language tasks with very long target sequences (e.g., in the region of hundreds of tokens). Adaptively re-encoding at every time step essentially means separating out relevant source concepts for each prediction. However, the Transformer is largely capable of encoding source phrases and decoding corresponding target phrases (or logical form fragments in semantic parsing), as evidenced by its remarkable success in many machine translation and semantic parsing benchmarks (Vaswani et al., 2017; Wang et al., 2020; Zheng and Lapata, 2021). This entails that the entanglement problem (i.e., not being able to disassociate the representations of different concepts for a sequence of predictions) does not occur very frequently. We therefore relax the strict constraint of re-encoding at every step in favor of the more flexible strategy of re-encoding at intervals.

Given source sequence $X = [x_1, x_2, ..., x_n]$, we specify $P = [t_1, t_2, ..., t_l](t_{i+1} - t_i = o)$ in advance, i.e., a sequence of re-encoding points with interval $o$. Then, during decoding, when reaching a re-encoding point $t(t = t_i)$, we update source encodings $\mathbf{H}_t$ and target memory $\mathbf{M}_t$:

$$\mathbf{H}_t = f_{\texttt{Adaptive\_Encoder}}(C_t) \tag{6.11}$$

$$y_t, \mathbf{M}_t = f_{\texttt{Decoder}}(Y_{<t}, \mathbf{H}_t, \{\}) \tag{6.12}$$

where $f_{\texttt{Adaptive\_Encoder}}$ denotes the adaptive encoder described in Section 6.2.1. For the next time step $t(t_i < t < t_{i+1})$, we fall back to the vanilla Transformer decoder using the source encodings $\mathbf{H}_{t_i}$ computed at time step $t_i$:

$$y_t, \mathbf{M}_t = f_{\texttt{Decoder}}(y_{t-1}, \mathbf{H}_{t_i}, \mathbf{M}_{t-1}) \tag{6.13}$$

Note that we always set $t_1$ to 1 to perform adaptive encoding at the first time step.

### 6.2.3 Disentangling Keys and Values

During decoding, Dangle accesses source information via cross-attention (also known as encoder-decoder attention) layers where the same source encodings are used to compute both keys *and* values. The core design principle underlying Dangle is that learning specialized representations for different purposes will encourage the model to zero in on relevant concepts, thereby disentangling their representations. Based on the same philosophy, we assume that the source keys and values encapsulate different aspects of source information, and that learning more specialized representations for them would further improve disentanglement, through the separation of the concepts involved.

A straightforward way to implement this idea is using two separately parameterized encoders to calculate two groups of source encodings (i.e., corresponding to keys and values, respectively) during re-encoding. However, in our preliminary experiments, we observed this leads to serious overfitting and performance degradation. Instead, we propose to encode values once and only update keys during adaptive encoding. We compute source *values* via the standard Transformer encoder:

$$\mathbf{H}^v = f_{\texttt{Encoder}}(X) \tag{6.14}$$

and adaptively re-encode source *keys* at an interval:

$$\mathbf{H}^k_t = f_{\texttt{Adaptive\_Encoder}}(C_t) \tag{6.15}$$

$$y_t, \mathbf{M}_t = f_{\texttt{KV\_Decoder}}(Y_{<t}, \mathbf{H}^v, \mathbf{H}^k_t, \{\}) \tag{6.16}$$

where $f_{\texttt{KV\_Decoder}}$ denotes a slightly modified Transformer decoder where source keys and values in each cross-attention layer are calculated based on different source encod-

ings:

$$\mathbf{K}_t = \mathbf{H}_t^k W^K \tag{6.17}$$

$$\mathbf{V} = \mathbf{H}^v W^V \tag{6.18}$$

$$\mathbf{O}_t = \text{Attention}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}) \tag{6.19}$$

At time step $t$ (where $t_i < t < t_{i+1}$), we perform vanilla Transformer decoding:

$$y_t, \mathbf{M}_t = f_{\texttt{KV\_Decoder}}(y_{t-1}, \mathbf{H}^v, \mathbf{H}_{t_i}^k, \mathbf{M}_{t-1}) \tag{6.20}$$

Note that fully sharing values could potentially cause some entanglement, however, we did not observe this in practice. We also experimented with a variant where keys are shared and values are repeatedly re-computed but empirically observed it obtains significantly worse generalization performance than the value-sharing architecture described above. This indicates that entanglement is more likely to occur when sharing keys.

## 6.3 Experimental Setup

### 6.3.1 Datasets

We evaluated R-Dangle on two machine translation datasets and one semantic parsing benchmark which we selected to maximally reflect natural language variations and real-world generalization challenges.

**ReaCT** is the machine translation benchmark developed in Chapter 5. The IWSLT 2014 De→En dataset consists of approximately 170K sequence pairs. We used the fairseq script `prepare-iwslt14.sh` to randomly sample approximately 4% of this dataset as validation set and kept the rest as training set. Following standard practice, we created an in-domain test set, the concatenation of files dev2010, dev2012, tst2010, tst2011, and tst2012. We created a compositional test set from the WMT'14 De→En training corpus (see Section 5.2 for more details).

**CoGnition** is another machine translation benchmark targeting compositional generalization (see an example in Table 2.4) (Li et al., 2021). It contains a synthetic test set to quantify and analyze compositional generalization of neural MT models. This test set was constructed by embedding synthesized novel compounds into training sentence tem-

plates. Each compound was combined with 5 different sentence templates, so that every compound can be evaluated under 5 different contexts. A major difference between REACT and CoGnition is the fact that test sentences for the latter are not naturally occurring. Despite being somewhat artificial, CoGnition overall constitutes a realistic benchmark which can help distinguish subtle model differences compared to purely synthetic benchmarks. For example, we showed in Chapter 4 that the encoder-only variant of Dangle performed badly on this dataset in spite of impressive performance on synthetic semantic parsing benchmarks (Kim and Linzen, 2020; Keysers et al., 2020).

**SMCalFlow-CS** is a large-scale semantic parsing dataset for task-oriented dialogue (Andreas et al., 2020), featuring real-world human-generated utterances about calendar management (see an example in Table 2.4). Yin et al. (2021) proposed a compositional skills split of SMCalFlow (SMCalFlow-CS) that contains single-turn sentences from one of two domains related to creating calendar events (e.g., *Set up a meeting with Adam*) or querying an org chart (e.g., *Who are in Adam's team?*), paired with LISP programs. The training set $\mathbb{S}$ consists of samples from single domains while the test set $\mathbb{C}$ contains compositions thereof (e.g., *create a meeting with Adam and his team*). Since zero-shot compositional generalization is highly non-trivial due to novel language patterns and program structures, we follow previous work (Yin et al., 2021; Qiu et al., 2022a) and consider a few-shot learning scenario, where a small number of cross-domain examples are included in the training set. We report experiments with 6, 16, and 32 examples.

### 6.3.2 Models

**Machine Translation** On machine translation, our experiments evaluated two variants of R-Dangle depending on whether keys and values are shared (R-Dangle$_{\text{shr}}$) or separate (R-Dangle$_{\text{sep}}$). We implemented all translation models with fairseq (Ott et al., 2019). Following previous work (Li et al., 2021; Zheng and Lapata, 2022), we compared with the machine translation models Dangle and vanilla Transformer using the popular fairseq configuration `transformer_iwslt_de_en`. We also implemented a bigger variant of these models using a new configuration, which empirically obtained better performance. We used 12 encoder layers and 12 decoder layers. We set the dropout to 0.3 for attention weights and 0.4 after activations in the feed-forward network. We also used pre-normalization (i.e., we added layer normalization before each block) to ease optimization. Following Zheng and Lapata (2022), we used relative position embeddings

| **CoGnition** | 1 | 2 | 4 | 8 |
|---|---|---|---|---|
| R-Dangle$_{\text{shr}}$ | 62.5 | 62.3 | 62.3 | 61.9 |
| R-Dangle$_{\text{sep}}$ | 63.4 | 63.1 | 62.3 | 62.1 |
| **ReaCT** | 1 | 2 | 4 | 8 |
| R-Dangle$_{\text{shr}}$ | 11.8 | 11.9 | 11.8 | 11.6 |
| R-Dangle$_{\text{sep}}$ | 12.3 | 12.2 | 11.9 | 11.7 |

Table 6.1: BLEU score for R-Dangle variants (with different re-encoding intervals) on CoGnition and ReaCT compositional generalization test sets. Note that R-Dangle$_{\text{shr}}$ with interval 1 is Dangle. Results are averaged over 5 random runs on CoGnition and 3 random runs on ReaCT.

(Shaw et al., 2018; Huang et al., 2020) which have demonstrated better generalization performance.

Hyperparameters for R-Dangle were tuned on the respective validation sets of CoGnition and ReaCT. Both R-Dangle$_{\text{shr}}$ and R-Dangle$_{\text{sep}}$ used a 12-layer decoder. For R-Dangle$_{\text{shr}}$, we tuned the number of layers of the two adaptive components $k_1$ and $k_2$, and set $k_1$ and $k_2$ to 2 and 10, respectively. For R-Dangle$_{\text{sep}}$, we shared some layers of parameters between the value encoder and the adaptive key decoder and experimented with different sharing strategies. Finally, we adopted a 10-layer value encoder and a 10-layer key encoder ($k_1 = 2$ and $k_2 = 8$). The top 8 layers in the two encoders were shared. This configuration produced 12 differently parametrized transformer encoder layers, thus maintaining an identical model size to the baseline.

**Semantic Parsing** Qiu et al. (2022a) showed the advantage of pre-trained sequence-to-sequence models on SMCalFlow-CS. We therefore built R-Dangle on top of BART-large (Lewis et al., 2020), which is well supported by fairseq. We used BART's encoder and decoder to instantiate the adaptive encoder and decoder in our model. For compatibility, we only employ the R-Dangle$_{\text{shr}}$ architecture. We also set $k_1$ and $k_2$ to 2 and 10, respectively.

### 6.3.3 Results

**Disentangling Keys and Values Improves Generalization** Table 6.1 reports the BLEU score (Papineni et al., 2002) achieved by the two R-Dangle variants on ReaCT and CoGnition, across different re-encoding intervals. R-Dangle$_{\text{sep}}$ is consistently

| Models | CoGnition | | | | ReaCT | |
|---|---|---|---|---|---|---|
| | ↓ $\text{ErrR}_{\text{Inst}}$ | ↓ $\text{ErrR}_{\text{Aggr}}$ | ↑ ind-test | ↑cg-test | ↑IWSLT14 | ↑cg-test |
| Transformer (small) | 30.5 | 63.8 | 69.2 | 59.4 | 34.4 | 9.5 |
| Dangle (small) | 22.8 | 50.6 | 69.1 | 60.6 | — | — |
| Transformer (large) | 23.4 | 53.7 | 70.8 | 61.9 | 36.0 | 11.4 |
| Dangle (large) | 19.7 | 47.0 | 70.6 | 62.5 | 36.1 | 11.8 |
| $\text{R-Dangle}_{\text{sep}}$ (interval = 1 ) | 16.0 | 42.1 | 70.7 | 63.4 | 36.0 | 12.3 |

Table 6.2: **Machine Translation Results:** we compare R-Dangle to baseline models on CoGnition and ReaCT. The small variant has a 6-layer encoder and decoder; the large variant has a 12-layer encoder and decoder. For CoGnition, we report instance-wise and aggregate compound translation error rates (ErrR) on the compositional generalization test set (cg-test) and BLEU on both in-domain test set (ind-test) and cg-test. For ReaCT, we report BLEU on the in-domain IWSLT 2014 De→En test set and the compositional generalization test set (cg-test) created in this paper. Results are averaged over 5 random runs on CoGnition and 3 random runs on ReaCT.

better than $\text{R-Dangle}_{\text{shr}}$ which confirms that representing keys and values separately is beneficial. We also observe that smaller intervals lead to better performance (we discuss this further later).

Table 6.2 compares $\text{R-Dangle}_{\text{sep}}$ (with interval 1) against baseline models. In addition to BLUE, we report novel compound translation error rate, a metric introduced in Li et al. (2021) to quantify the extent to which novel compounds are mistranslated. We compute the error rate over instances and an aggregate score over contexts. $\text{R-Dangle}_{\text{sep}}$ delivers compositional generalization gains over Dangle and vanilla Transformer models (both in terms of BLEU *and* compound translation error rate), even though their performance improves when adopting a larger 12-layer network. $\text{R-Dangle}_{\text{sep}}$ achieves a new state of the art on CoGnition (a gain of 0.9 BLEU points over Dangle and 1.5 BLEU points over the Transformer baseline). $\text{R-Dangle}_{\text{sep}}$ fares similarly on ReaCT; it is significantly superior to the Transformer model by 0.9 BLEU points, and Dangle by 0.5 BLEU points. Moreover, improvements on compositional generalization are not at the expense of in-domain performance (R-Dangle obtains similar performance to the Transformer and Dangle on the IWSLT2014 in-domain test set).

**R-Dangle Can Handle Long-tail Compositional Patterns Bettter** We next examine model performance on real-world examples with diverse language and different levels

| **Systems** | 8-ℂ | 16-ℂ | 32-ℂ |
|---|---|---|---|
| BERT2SEQ (Yin et al., 2021) | — | 33.6 | 53.5 |
| BERT2SEQ+SS (Yin et al., 2021) | — | 46.8 | 61.7 |
| C2F (Yin et al., 2021) | — | 40.6 | 54.6 |
| C2F+SS (Yin et al., 2021) | — | 47.4 | 61.9 |
| T5 (Qiu et al., 2022a) | 34.7 | 44.7 | 59.0 |
| T5+CSL (Qiu et al., 2022a) | 51.6 | 61.4 | 70.4 |
| BART-large | 32.1 | 47.2 | 61.9 |
| +R-Dangle$_{\text{shr}}$ (interval = 6) | 36.3 | 50.6 | 64.1 |

Table 6.3: **Semantic Parsing Results:** we compare R-Dangle to various systems on SMCalFlow-CS. *-ℂ denote different settings with 8, 16, and 32 cross-domain examples added to the training set. Results for BERT and C2F models are from Yin et al. (2021). Results for T5 models are from Qiu et al. (2022a). Results for BART and R-Dangle are averaged over 3 random runs.

of composition. Specifically, we train R-Dangle$_{\text{sep}}$ (interval=1) and a Transformer on the IWSTL14 corpus and test on the pool of 1.3M WMT examples obtained after filtering OOV words (see Section 5.2 for more details). Figure 6.2 plots the difference in BLEU between the two models against compositional degree. This fine-grained evaluation reveals that they perform similarly on the majority of less compositional examples (BLUE difference is around zero), however, the performance gap becomes larger with more compositional examples (higher difference means higher BLEU for R-Dangle$_{\text{sep}}$). This indicates that R-Dangle is particularly effective for handling long-tail compositional patterns.

**R-Dangle Boosts the Performance of Pretrained Models** The "pre-train and fine-tune" paradigm (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2020; Lewis et al., 2020) has been widely adopted in NLP, and semantic parsing is no exception (Shin et al., 2021; Qiu et al., 2022a). We further investigate R-Dangle's performance when combined with a pre-trained model on the SMCalFlow-CS dataset (across the three cross-domain settings). Table 6.3 shows that R-Dangle$_{\text{shr}}$ boosts the performance of BART-large, which suggests that generalization improvements brought by R-Dangle are complementary to generalization benefits afforded by large-scale pre-training (see Chapter 4 for a similar conclusion). The proposed model effectively marries pre-training with disentangled representation learning to achieve better generalization.

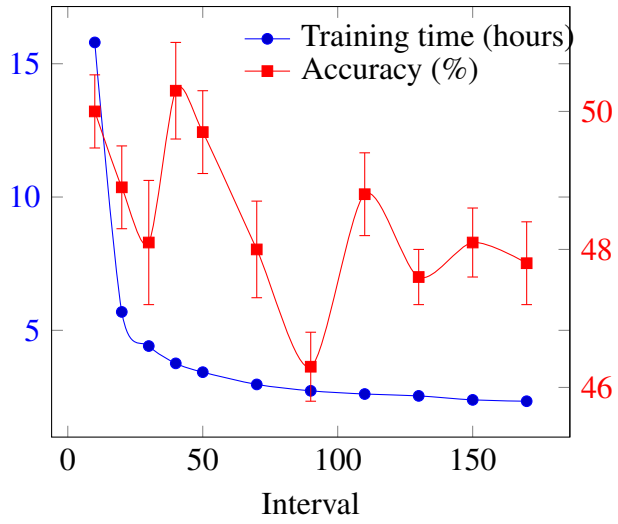


Figure 6.1: Training cost (hours) and test accuracy vs interval length. R-Dangle$_k$ was trained on SMCalFlow-CS using 4 A100 GPUs. For each interval, we perform 5 random runs. Error bars are 95% confidence intervals.

In Table 6.3, we also compare R-Dangle with other top-performing models on SMCalFlow-CS. These include: (a) a sequence-to-sequence model with a BERT encoder and an LSTM decoder using a copy mechanism (BERT2SEQ; Yin et al. 2021); (b) the coarse-to-fine model of Dong and Lapata (2018) which uses a BERT encoder and a structured decoder that factorizes the generation of a program into sketch and value predictions; (c) and combinations of these two models with span-supervised attention (+SS; Yin et al. 2021). We also include a T5 model and a variant thereof trained on additional data using a model called Compositional Structure Learner (CSL) to generate examples for data augmentation (T5+CSL; Qiu et al. 2022a). R-Dangle with BART performs best among models that do *not* use data augmentation across compositional settings. Despite its excellent performance, R-Dangle still lags behind T5+CSL, which indicates that in some scenarios explicitly injecting compositional inductive bias (through grammars) is still helpful for further improving the generalization of neural networks. At a high level, how to build neural systems that show human-like symbolic information processing is still an open problem.

**Larger Re-encoding Intervals Reduce Training Cost** The results in Table 6.1 indicate that re-encoding correlates with R-Dangle's generalization ability, at least for machine translation. Both model variants experience a drop in BLEU points when increasing the re-encoding interval to 8. We hypothesize that this sensitivity to interval length is task-related; target sequences in machine translation are relatively short and

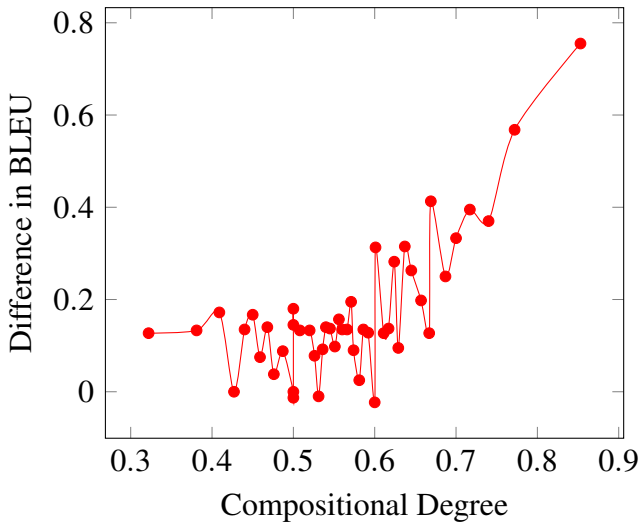


Figure 6.2: Difference in BLEU score between R-Dangle$_{\text{sep}}$ (interval = 1) and Transformer vs compositional degree. A positive score means R-Dangle$_{\text{sep}}$ is better than Transformer. Each data point is computed on 30K WMT examples. R-Dangle shows increasing performance improvements as test examples become more compositional.

representative of real language, whereas in SMCalFlow-CS, the average length of target sequences (in formal language) is 99.5 (when tokenized with the BART tokenizer) and the maximum length is 411. It is computationally infeasible to train R-Dangle with small intervals on this dataset, however, larger intervals still produce significant performance gains.

Figure 6.1 shows how accuracy and training time vary with interval length on SMCalFlow-CS with the 16-$\mathbb{C}$ setting. Larger intervals substantially reduce training costs with an optimal speed-accuracy trade-off in between 10 and 50. For instance, interval 40 yields a 4x speed-up compared to interval 10 while achieving similar accuracy (i.e., 50.3% vs 50%). Therefore, SMCalFlow-CS allows a much larger interval that is able to achieve both superior generalization to Transformer and better compute efficiency than Dangle. Interestingly, the performance is not changing monotonically. The random variance, shown in the error bars, could partly explain this. Besides, we conjecture that it could also be due to the effect of re-encoding points. A large interval could produce new re-encoding points that a small interval does not produce and these points could be critical to generalization performance. As a result, raising intervals sometimes leads to performance boosts. In general, finding the minimum set of *critical* re-encoding points to achieve an optimal trade-off between generalization and efficiency is an open research problem, which we leave to future work.

## 6.4 Summary

In this chapter, we have moved closer towards real-world compositional generalization. We adapted Dangle to real-world tasks by improving upon it with two key modifications. We showed that re-encoding keys periodically, at some interval, improves both efficiency and accuracy. Experimental results on real-world language tasks including both semantic parsing and machine translation showed that the disentangled sequence-to-to-sequence models deliver superior compositional generalization compared to the Transformer model across tasks, metrics, and datasets. Machine translation experiments on a diverse corpus of 1.3M WMT examples show the proposed R-Dangle model is particularly effective for long-tail compositional patterns. As a general neural network architecture, R-Dangle inherits the robustness and flexibility of neural models and is adept at handling real-world challenges.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

In this thesis, we investigated how to improve compositional generalization of neural sequence-to-sequence models, aiming at building systems with human-like systematic compositionality. We studied this problem from two different perspectives. Firstly, motivated by the inherent compositionality of grammar-based systems for semantic parsing, in Chapter 3 we attempted to incorporate symbolic structure into neural sequence-to-sequence models. The key challenge is how to combine the best of both worlds: the compositional nature of symbolic systems and the flexibility and robustness of neural systems. We proposed a two-stage decoding strategy, aiming to improve compositional generalization while having a robust and flexible system that is applicable to different datasets and semantic formalisms. Specifically, we augmented neural sequence-to-sequence models (connectionist architecture) with semantic tagging (symbolic structure). An input utterance is first tagged with semantic symbols representing the meaning of individual words, and then the final meaning representation is generated based on the tagged input. This two-stage decoding process leads to improved compositional generalization across different semantic parsing datasets and semantic formalisms.

The success of semantic tagging assumes that there is a well-defined one-to-one mapping between input symbols and output symbols. However, this condition does not always hold in real-world applications. For example, when the output logical form in semantic parsing includes many formalism-specific symbols that do not align with language expressions or the output utterance in machine translation bears a more complex and global many-to-many relationship to the input utterance. In Chapter 4,

we thus tackled compositional generalization through pure neural architectures without symbolic components. It has the potential to fully maintain the robustness and flexibility of neural models required to perform many real language tasks. We designed Dangle, a new neural network architecture for sequence-to-sequence modeling to learn more disentangled representations for better compositional generalization. Specifically, we first identified an entanglement problem with how different semantic factors (e.g., lexical meaning and semantic relations) are represented in neural sequence models that hurts generalization. Then we proposed an extension to sequence-to-sequence models which adaptively re-encodes (at each time step) the source input by conditioning the source representations on the newly decoded target context. This mechanism makes it easier for the encoder to exploit specialized information for each prediction and disentangle relevant source factors across different predictions. We conducted experiments on two semantic parsing benchmarks and one machine translation benchmark specifically designed for testing compositional generalization. We empirically verified that this new network architecture leads to better generalization than the Transformer architecture.

The main motivation of advancing compositional generalization via pure architectural modifications is to preserve the flexible and general ability of neural networks to perform real-world tasks. However, as a proof-of-concept, we evaluated Dangle exclusively on synthetic datasets to isolate compositional generalization. We would like to move towards more real-world compositional generalization. In Chapter 5, we first addressed the evaluation problem: we developed a new methodology for detecting examples representative of compositional generalization in naturally occurring text to better emulate a real-world setting. Based on the methodology, we created a real-world machine translation challenge to complement the existing benchmarks. In Chapter 6, we adapted Dangle to real-world language tasks with two key modifications. We showed that re-encoding keys periodically, at some interval, improves both efficiency and accuracy. Experimental results confirmed that our modifications improve generalization in real-world settings across tasks, metrics, and datasets and our new benchmark provides a challenging testbed for evaluating new modeling efforts.

The findings of this thesis include:

- We proposed a two-stage decoding strategy to augment neural sequence-to-sequence models (connectionist architecture) with semantic tagging (symbolic structure) for semantic parsing. We showed that for tasks that show clear local alignments between the source input and the target output, explicitly taking into account the alignments leads to improved compositional generalization.

- We proposed a new family of neural network architectures for compositional generalization, namely disentangled sequence-to-sequence models. We showed that learning specialized representation for different purposes ( through re-encoding and key value separation) gives rise to more disentangled representations and superior compositional generalization compared to the widely used Transformer architecture.

- We developed a new methodology for creating real-world compositional generalization challenges, in which we detect examples representative of compositional generalization with respect to a training set in naturally occurring text. Based on the proposed methodology, we created a real-world machine translation challenge to complement the existing compositional generalization benchmarks.

## 7.2 Future Work

Avenues for future research about compositional generalization are many and varied. We discuss several promising topics as follows:

**Evaluation of Compositional Generalization** Systems of compositional generalization are as good as the benchmarks they are evaluated on. While it is well known that compositionality is central to human language ability, there is no agreement as to what it means for a system to be compositional. Different benchmarks evaluating compositional generalization imply different interpretations of compositionality. Most existing synthetic benchmarks (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020) stick to a local and arithmetic-like interpretation: the meaning of expressions is computed from its parts independently of the external context as in arithmetic the meaning of (3 + 2) is always 5, independently of where it occurs. Dankers et al. (2022) highlight this limitation of existing benchmarks and call for developing benchmarks using real data to evaluate human-like compositionality. As discussed in Section 2.2, in natural language there exist many frequent linguistic phenomena that do not adhere to this local protocol such as polysemy, anaphora, idioms and so on. Therefore, an ideal benchmark would emphasize compositional aspects of language without ignoring weakly or non-compositional aspects of language.

**Efficient Disentangled Representation Learning** The disentangled sequence-to-sequence model proposed in this thesis has shown superior performance in composi-

tional generalization compared to the Transformer model. However, its cubic complexity hinders its application to scenarios with extremely long examples and limited compute resources. In this thesis, we only explore a simple and general periodic re-encoding strategy to speed up training and inference. On machine translation, the optimal generalization performance still requires using small interval values. R-Dangle with small intervals still runs much slower than an equivalent Transformer model. It would be extremely costly to run a large R-Dangle model with small intervals on large datasets. However, potentially more efficient re-encoding strategies based on different assumptions could be introduced to further narrow the gap. For example, we empirically find that the Transformer models fail more frequently at compositional generalization for rare words. A direct solution taking into account this factor is to only re-encode for the prediction of rare words. This is efficient because rare words, by definition, do not occur frequently, therefore unburdening the model of frequent re-encoding.

**Causal Representation Learning** A potential alternative to learning disentangled representations for better generalization is causal representation learning (Scholkopf et al., 2021). Causal representation learning aims to discover causal relations underlying the data generating process and model the effect of interventions and distribution changes. Causal relations are expected to hold more robustly across different environments and tasks than statistical associations. They have the potential to provide predictions for situations that are very far from the observed distribution and enable more robust out-of-distribution generalization. As an example, the notion of independent mechanisms has been introduced from the field of causal inference to motivate the design of neural networks with independent modular structures (Goyal et al., 2021; Lamb et al., 2021). If a neural model is able to learn the underlying causal relations between every meaning unit and language symbols that it originates from, it would generalize compositionally.

**Large Language Models** Large language models (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2020; Lewis et al., 2020; Brown et al., 2020) have achieved tremendous success in a variety of NLP tasks. However, they have shown limited success at improving compositional generalization, as discussed in Chapters 4 and 6. The limited gain partly arises from the standard "pre-train and fine-tune" paradigm where the pre-trained model is fully fine-tuned on downstream tasks. There is accumulating evidence suggesting that fine-tuning the full model tends to overfit to the training data and

hurts generalization in out-of-distribution settings whereas parameter-efficient tuning or prompting without any parameter updates has shown more robust generalization (Lester et al., 2021; Li and Liang, 2021; Qiu et al., 2022b). However, parameter-efficient tuning or prompting could be not expressive enough to learn complex tasks. Recently, He et al. (2022) performed comprehensive empirical studies on different parameter-efficient fine-tuning methods. They proposed a new fine-tuning method achieving comparable results to fine-tuning all parameters on complex sequence-to-sequence tasks such as machine translation and summarization. While they only perform experiments in standard settings, it could be beneficial for the more challenging compositional setting because parameter-efficient fine-tuning could potentially reduce overfitting and facilitate generalization.

# Bibliography


Abzianidze, L. and Bos, J. (2017). Towards universal semantic tagging. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Akyürek, E., Akyurek, A. F., and Andreas, J. (2021). Learning to recombine and resample data for compositional generalization. In *Proceedings of the 9th International Conference on Learning Representations*, Online.

Akyurek, E. and Andreas, J. (2021). Lexicon learning for few shot sequence modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4934–4946, Online. Association for Computational Linguistics.

Andreas, J. (2020). Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Andreas, J., Bufe, J., Burkett, D., Chen, C., Clausman, J., Crawford, J., Crim, K., DeLoach, J., Dorner, L., Eisner, J., Fang, H., Guo, A., Hall, D., Hayes, K., Hill, K., Ho, D., Iwaszuk, W., Jha, S., Klein, D., Krishnamurthy, J., Lanman, T., Liang, P., Lin, C. H., Lintsbakh, I., McGovern, A., Nisnevich, A., Pauls, A., Petters, D., Read, B., Roth, D., Roy, S., Rusak, J., Short, B., Slomin, D., Snyder, B., Striplin, S., Su, Y., Tellman, Z., Thomson, S., Vorobev, A., Witoszko, I., Wolfe, J., Wray, A., Zhang, Y., and Zotov, A. (2020). Task-Oriented Dialogue as Dataflow Synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International*

*Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Bergen, L., O' Donnell, T., and Bahdanau, D. (2021). Systematic generalization with edge transformers. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1390–1402. Curran Associates, Inc.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chen, R. T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, pages 2610–2620. Curran Associates, Inc.

Cheng, P., Min, M. R., Shen, D., Malon, C., Zhang, Y., Li, Y., and Carin, L. (2020). Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, Online. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference*

*on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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

Conklin, H., Wang, B., Smith, K., and Titov, I. (2021). Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, pages 43–48, Stroudsburg, PA, USA.

Dankers, V., Bruni, E., and Hupkes, D. (2022). The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Dong, L. and Lapata, M. (2016). Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Dong, L. and Lapata, M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of ibm model 2. In *North American Chapter of the Association for Computational Linguistics*.

Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D. (2018). Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Furrer, D., van Zee, M., Scales, N., and Schärli, N. (2020). Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

García-Ramírez, E. (2019). *Open Compositionality: Toward a New Methodology of Language*. Rowman & Littlefield.

Ge, R. and Mooney, R. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 9–16, Ann Arbor, Michigan. Association for Computational Linguistics.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2021). Recurrent independent mechanisms. In *International Conference on Learning Representations*.

Guo, Y., Lin, Z., Lou, J.-G., and Zhang, D. (2020). Hierarchical poset decoding for compositional generalization in language. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6913–6924. Curran Associates, Inc.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. (2022). Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Herzig, J. and Berant, J. (2021). Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017*, Toulon, France.

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

Huang, Z., Liang, D., Xu, P., and Xiang, B. (2020). Improve transformer models with better relative position embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335, Online. Association for Computational Linguistics.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., and Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Jacobson, P. (2002). The (dis)organization of the grammar: 25 years. *Linguistics and Philosophy*, 25:601–626.

Jia, R. and Liang, P. (2016). Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural*

*Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

John, V., Mou, L., Bahuleyan, H., and Vechtomova, O. (2019). Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. (2020). Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations, ICLR 2020*, Online.

Kim, H. and Mnih, A. (2018). Disentangling by factorising. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658, Stockholmsmässan, Stockholm Sweden. PMLR.

Kim, N. and Linzen, T. (2020). COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, California.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, Banff, AB, Canada.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2021). Wilds: A benchmark of in-the-wild distribution shifts. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR.

Lake, B., Linzen, T., and Baroni, M. (2019). Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Meeting of the Cognitive Science Society*, pages 611–617, Montréal, Canada.

Lake, B. M. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Lamb, A., He, D., Goyal, A., Ke, G., Liao, C.-F., Ravanelli, M., and Bengio, Y. (2021). Transformers with competitive ensembles of independent mechanisms. *ArXiv*, abs/2103.00336.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Li, Y., Yin, Y., Chen, Y., and Zhang, Y. (2021). On compositional generalization of neural machine translation. In *Proceedings of the 59th Annual Meeting of the Associ-*

*ation for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4767–4780, Online. Association for Computational Linguistics.

Liang, P., Jordan, M. I., and Klein, D. (2013). Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Liu, Y., Liu, P., Radev, D., and Neubig, G. (2022). BRIO: Bringing order to abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2890–2903, Dublin, Ireland. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4114–4124, Long Beach, California, USA. PMLR.

Malinin, A. and Gales, M. (2021). Uncertainty estimation in autoregressive structured prediction. In *Proceedings of the 9th International Conference on Learning Representations*, Online.

Marcus, G. F. (2003). *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT press.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

Montague, R. (1970). Universal grammar. *Theoria*, 36(3):373–398.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., and Allen, J. (2016). A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

*Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Mostafazadeh, N., Roth, M., Louis, A., Chambers, N., and Allen, J. (2017). LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain. Association for Computational Linguistics.

Oren, I., Herzig, J., Gupta, N., Gardner, M., and Berant, J. (2020). Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online. Association for Computational Linguistics.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Otter, D., Medina, J. R., and Kalita, J. K. (2018). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32:604–624.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Partee, B. (1995). Lexical semantics and compositionality. In Gleitman, L. and Liberman, M., editors, *Invitation to Cognitive Science Part I: Language*. MIT Press, Cambridge, MA.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018*

*Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Platanios, E. A., Pauls, A., Roy, S., Zhang, Y., Kyte, A., Guo, A., Thomson, S., Krishnamurthy, J., Wolfe, J., Andreas, J., and Klein, D. (2021). Value-agnostic conversational semantic parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3666–3681, Online. Association for Computational Linguistics.

Price, P. J. (1990). Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Qiu, L., Shaw, P., Pasupat, P., Nowak, P., Linzen, T., Sha, F., and Toutanova, K. (2022a). Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Qiu, L., Shaw, P., Pasupat, P., Shi, T., Herzig, J., Pitler, E., Sha, F., and Toutanova, K. (2022b). Evaluating the impact of model scale for compositional generalization in semantic parsing. *ArXiv*, abs/2205.12253.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Roy, S., Thomson, S., Chen, T., Shin, R., Pauls, A., Eisner, J., and Durme, B. V. (2022). Benchclamp: A benchmark for evaluating language models on semantic parsing. *ArXiv*, abs/2206.10668.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2020). Distributionally robust neural networks. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia.

Scholak, T., Schucher, N., and Bahdanau, D. (2021). PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings*

*of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Scholkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109:612–634.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. (2020). The pitfalls of simplicity bias in neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9573–9585. Curran Associates, Inc.

Shaw, P., Chang, M.-W., Pasupat, P., and Toutanova, K. (2021). Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Sherborne, T., Yumo, X., and Lapata, M. (2020). Bootstrapping a crosslingual semantic parser. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 499–517, Online. Association for Computational Linguistics.

Shin, R., Lin, C., Thomson, S., Chen, C., Roy, S., Platanios, E. A., Pauls, A., Klein, D., Eisner, J., and Van Durme, B. (2021). Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in*

*Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Swinney, D. A. and Cutler, A. (1979). The access and processing of idiomatic expressions. *Journal of Verbal Learning and Verbal Behavior*, 18(5):523–534.

Szabó, Z. G. (2012). 64 The case for compositionality. In *The Oxford Handbook of Compositionality*. Oxford University Press.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2020). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Wang, B., Yin, W., Lin, X. V., and Xiong, C. (2021). Learning to synthesize data for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2760–2766, Online. Association for Computational Linguistics.

Weissenborn, D., Wiese, G., and Seiffe, L. (2017). Making neural QA as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational*

*Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada. Association for Computational Linguistics.

Wong, Y. W. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA. Association for Computational Linguistics.

Wong, Y. W. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

Yin, P., Fang, H., Neubig, G., Pauls, A., Platanios, E. A., Su, Y., Thomson, S., and Andreas, J. (2021). Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

Yu, T., Li, Z., Zhang, Z., Zhang, R., and Radev, D. (2018a). TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. (2018b). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence - Volume 2*, page 1050–1055. AAAI Press.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical*

*Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic. Association for Computational Linguistics.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, page 658–666, Arlington, Virginia, USA. AUAI Press.

Zheng, H. and Lapata, M. (2021). Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zheng, H. and Lapata, M. (2022). Disentangled sequence to sequence learning for compositional generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4256–4268, Dublin, Ireland. Association for Computational Linguistics.

Zheng, H. and Lapata, M. (2023). Real-world compositional generalization with disentangled sequence-to-sequence learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1711–1725, Toronto, Canada. Association for Computational Linguistics.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.