LAGr: Label Aligned Graphs for Better Compositional Generalization in Semantic Parsing

Dóra Jámbor

School of Computer Science

McGill University, Montréal

A thesis submitted to McGill University in partial fulfillment of the requirements

of the degree of Master of Computer Science.

Acknowledgements

I am deeply grateful for my wonderful supervisors Dzmitry Bahdanau and Joelle Pineau. I thank Dima being an invaluable mentor to me, for the key role he played in shaping and guiding me through my thesis project and to get this research published. I also thank him for keeping up my spirit and motivation through some difficult months during the pandemic.

I thank Joelle for being one of the most caring and dedicated mentors I've had and for being a role model to me in many aspects of my career. I also thank William Hamilton who supervised me throughout the first year of my masters, and who also tremendously contributed to my studies on graph neural networks and knowledge graphs and to my first publication. I am also deeply grateful to Joelle and Will for the support, flexibility and understanding they showed during my mother's illness.

Additionally, I thank Komal Teru for his contributions to my first paper during my studies. It was a delightful experience to have him as a research collaborator and to work closely on getting my first paper published. I also extend my thanks to Danny Tarlow who kindly offered his time and mentorship through my first projects in semantic parsing. Furthermore, I thank Nitarshan Rajkumar for his feedback on this project for his friendship and constant supply of jokes during the final months of wrapping up my thesis. I also thank my amazing family and friends for their encouragement, patience and support during my studies.

Last but not least, I thank La Dépendance, Café Névé, Orr, Myriad, Replika, Les Faiseurs among many other wonderful cafés of le Plateau-Mont-Royal and Petite Italie for providing an inspiring and vibrant work environment in Montréal.

Abstract

The goal of semantic parsing is to map natural language utterances or questions to structured meaning representations, such as executable programs or logical forms. The dominant approach for this task is sequence-to-sequence (seq2seq) methods that produce meaning representations sequentially, generating one word at a time. While these methods have led to great advances in semantic parsing, recent research in compositional generalization has pointed out that these methods struggle to generalize systematically, i.e. to handle examples that require recombining known knowledge in novel contexts.

In this work, we show that better compositional generalization can be achieved by producing the meaning representation (MR) directly as a graph instead of modeling it as a sequence. To this end we propose LAGr, the **L**abel **A**ligned **Gr**aphs algorithm that produces semantic parses by predicting node and edge labels for a complete multi-layer input-aligned graph. We present two variants of LAGr: The strongly-supervised LAGr algorithm which requires aligned graphs as inputs, and the weakly-supervised LAGr algorithm where alignments are inferred for originally unaligned target graphs using an approximate MAP inference procedure. Using two compositional generalization benchmarks, we demonstrate that both the strongly- and weakly-supervised LAGr algorithms achieve significant improvements upon the baseline seq2seq semantic parsers.

Abrégé

Le but de l'analyse sémantique est d'associer des énoncés ou des questions dans un langage naturel à une représentation de sens structurée, telle qu'un programme exécutable ou une proposition logique. Les stratégies les plus communément déployées pour réaliser cette tâche sont les méthodes dites "séquence-à-séquence" (seq2seq) qui produisent des représentations structurées de manière séquentielle, générant un mot à la fois. Bien que ces méthodes aient mené à de grandes avancées dans l'analyse sémantique, des recherches récentes en généralisation compositionnelle ont souligné le fait que ces méthodes ont de la difficulté à généraliser systématiquement: elles ne sont pas capables de recombiner de manière efficace des connaissances dans un nouveau contexte - un attribut essentiel pour qu'une méthode reste fiable quand elle traite des nouveaux exemples.

Dans ce travail, nous démontrons qu'une meilleure généralisation compositionnelle peut être obtenue en produisant une représentation de sens structurée directement sous forme de graphe au lieu de la modéliser en tant que séquence. À cette fin, nous proposons LAGr (Label Aligned Graphs), un algorithme qui produit une analyse sémantique en prédisant les étiquettes de nœuds et d'arêtes pour un graphe complet multicouche dont les noeuds sont alignés à l'énoncé d'entrée. Nous présentons deux variantes de LAGr : l'algorithme LAGr fortement supervisé qui requiert des graphes pré-alignés à leurs entrées, et l'algorithme LAGr faiblement supervisé où les alignements sont déduits pour les graphes-cibles originellement non alignés en utilisant une procédure d'inférence maximum a posteriori approximative. Grâce à deux jeux de données de référence mesurant la capacité de généralisation compositionnelle, nous démontrons que les algorithmes LAGr fortement et faiblement supervisés offrent une amélioration significative par rapport aux analyseurs sémantiques de référence seq2seq.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural languages such as English and Hungarian are fundamental for humans to be able to communicate with one another. They provide flexible means to organize and formulate ideas and knowledge in ways that can be understood, retrieved and built upon. While historically we used natural language to store and accumulate knowledge via analogue systems such as books and documents, much of today's knowledge is now stored digitally, for example in databases.

In contrast to analogue systems, we access, manipulate and interact with digital information through machines that require highly structured formal languages (e.g. programming languages) for communication. These systems consequently place a significant barrier to interaction for most people who lack the requisite technical background.

Even for those comfortable with such languages, expressing questions and objectives in a machine readable language requires significant time and mental effort.

Semantic parsing is a paradigm of natural language processing (NLP) that aims to bridge this gap by translating natural language sentences to machine-understandable meaning representations (MR). These meaning representations take many forms, including executable programs such as Python code or SQL database queries, and logical forms such as lambda calculus. Consequently, semantic parsing holds the promise of enabling more convenient and accessible human-computer interaction.

1.1 Prior Work on Semantic Parsing

Most of the early works on semantic parsing concentrated on syntactic parsing, where the goal is to manually create (Woods et al., 1972; Woods, 1973) or to learn *grammars*. These grammars provided syntactic annotations to a given natural language sentence by grouping the words of the sentence into a hierarchical constituent structure, i.e., a *parse tree* (Berwick et al., 1985; Wirth, 1989). The resulting parse tree was then translated directly to the meaning representation such as a database query, usually by using a set of predefined rules.

Other works used so-called *semantic grammars* that performed both syntactic and semantic processing of sentences to yield meaning representations (Hendrix et al., 1978). These systems still parsed the natural language sentences to parse trees. However, in contrast to syntax-based grammars, these parse trees contained information about semantic concepts about the knowledge domain that further constrained the generation of the final MR. These semantic constraints were closely tied to the specific context of the semantic parsing task (e.g. database schemas), which prevented them from being reused in other applications (Androutsopoulos et al., 1995). For this reason, semantic grammars gradually lost popularity.

Subsequent works aimed to learn grammars. A notable example by Zelle & Mooney (1993; 1994) introduced the CHILL system to learn semantic grammars using a training corpus of sentences paired with their meaning representations. Specifically, CHILL made use of a deterministic shift–reduce parser and developed a learning algorithm based on Inductive Logic Programming to learn control rules for parsing.

Zelle & Mooney (1996) adopted CHILL to develop one of the earliest examples of a learning system for a natural language interface to databases that was able to directly output executable database queries. The major limitation of this work is that it required access to pre-built lexicons that mapped words from the input sentence to parts of the meaning representation. Later, Thompson & Mooney (2003) showed that lexicons can be incorporated into the learning problem and be acquired automatically. While these works still relied on syntactic or semantic annotations between the input-output pairs, Zettlemoyer & Collins (2005) proposed an approach that required minimal

level of annotations. Their algorithm learned to induce a grammar that maps sentences to logical form, along with a probabilistic model that assigns a distribution over syntactic and semantic analyses conditioned on the input sentence.

While many ideas from these pioneering works reappear in subsequent literature (Herzig & Berant, 2021; Akyürek et al., 2021; Wang et al., 2021), the dominant approach to semantic parsing shifted to deep learning models that use neural networks at their core. These approaches offer more generic and flexible semantic parsers that eliminate the need for hand-crafted and domain-specific lexicons, grammars, or other linguistic features. In particular, sequence-to-sequence methods (seq2seq Sutskever et al. (2014)) emerged as the main paradigm in deep learning to effectively model sequential data such as natural language. This was first adopted by Dong & Lapata (2016) who showed that seq2seq methods can be used for semantic parsing to replace previous approaches that relied on domain-specific feature engineering. They were also the first to study attention (Bahdanau et al., 2015) in seq2seq-based semantic parsing that allowed modeling the alignment between natural language sentences and their respective logical forms. Additionally, they also explored hierarchical decoders as a way to better capture the hierarchical nature of meaning representations. Similarly, a large body of work focuses on constraining the output space of neural seq2seq semantic parsers as a way to better adhere to the explicit structure of the corresponding logical form (Chen et al., 2018; Krishnamurthy et al., 2017; Wang et al., 2018a;b; Scholak et al., 2021).

While most of these works leveraged seq2seq methods based on recurrent neural networks, with the introduction of Transformers (Vaswani et al., 2017), many works have begun studying Transformer-based seq2seq models for semantic parsing. SQLova by Hwang et al. (2019) was the first to show how Transformers can be used to jointly encode the natural language sentence and database schema (i.e., the environment in which the question is interpreted) in semantic parsing tasks. Some works have explored the use of pretraining in semantic parsing via the use of pretrained language models (Hwang et al., 2019; Wang et al., 2020; Scholak et al., 2020; Herzig & Berant, 2018), while others proposed novel pretraining objectives specifically designed for semantic parsing (Yin et al., 2020; Herzig et al., 2020; Deng et al., 2020).

1.2 Challenges with Compositionality

A key challenge that remains with neural language models is that they lack a systematic understanding of language, often relying on exploitation of irrelevant statistical artifacts in training data (Jia & Liang, 2017; Weissenborn et al., 2017; McCoy et al., 2019; Gontier et al., 2020). This makes it challenging for models to differentiate between irrelevant contextual changes and relevant information that matters for performing a given task accurately, especially when datapoints fall outside the training distribution.

The lack of systematicity in reasoning about entities and the rules that relate them to one another has been widely studied over the past years by research on *compositional generalization* (also referred to as *systematic generalization*) (Lake & Baroni, 2018; sin; Keysers et al., 2019; Bahdanau et al., 2019; Gontier et al., 2020). We define compositional generalization as the model's ability to combine known entities in novel ways and thus generalize to data points that do not follow the training distribution.

Compositional generalization is especially important for the task of semantic parsing where models need to produce outputs that adhere to strict structural constraints. To generalize compositionally in this task, the model must be capable of producing MRs for examples that feature new combinations of meaning construction rules, such as the rule that maps a noun like "*hedgehog*" in Figure 1.1 to its respective predicate $hedgehog(.)$, and the rule that defines which semantic role (e.g. *agent* or *theme*) the resulting predicate takes with respect to the verb.

Using synthetic (Bahdanau et al., 2019; Kim & Linzen, 2020; Keysers et al., 2019) and natural benchmarks (Finegan-Dollak et al., 2018; Shaw et al., 2021), researchers have been studying compositional generalization of existing semantic parsing methods as well as proposing new approaches such as using meta-learning (Conklin et al., 2021), pretrained models (Furrer et al., 2020), or intermediate meaning representations (Herzig et al., 2021). A unifying theme across the majority of these works is that they rely on seq2seq methods that produce serialized MRs in an autoregressive fashion, predicting one token at a time while conditioning on all previously generated tokens.

Example from the training set
A ***hedgehog*** ate the cake
$*hedgehog(x_1) \wedge cake(x_4) \wedge eat.agent(x_2, x_1) \wedge eat.theme(x_2, x_4)$
Example from the generalization set
The baby liked the ***hedgehog***
$*baby(x_1) \wedge hedgehog(x_4) \wedge like.agent(x_2, x_1) \wedge like.theme(x_2, x_4))$

Figure 1.1: Examples from the training and the generalization sets from COGS, a compositional generalization benchmark by Kim & Linzen (2020). The examples show how compositional generalization requires models to understand familiar concepts seen in the training set, e.g. "*hedgehog*", in unseen combinations such as featuring "*hedgehog*" in the *theme* role instead of the *agent* role in the generalization set.

1.3 LAGr: Label Aligned Graphs

The main contribution of this thesis follows from the hypothesis that for semantic parsing, constructing the MR by combining independent predictions that are not conditioned on each other can generalize more compositionally than a classic seq2seq approach. For example, consider the sentence "*The dog liked that the hippo danced*". The predictions that "*dog*" is the agent of "*like*" and that "*hippo*" is the agent of "*danced*" can arguably be made independently of each other. Our intuition is that a model that predicts such aspects of meaning independently of each other can be better at learning context-insensitive rules because the overall context for each individual prediction is reduced. Accordingly, we propose LAGr (**L**abel **A**ligned **Gr**aphs), a framework to produce semantic parses by independently labelling the nodes and edges of a fully-connected multi-layer output graph that is aligned with the input utterance. While the general idea of predicting semantic parses as graphs is not new (Lyu & Titov, 2018), the compositional generalization benefits of doing so have not been investigated prior to this work. Importantly, LAGr retains most of the flexibility that seq2seq models have, without the complexity and rigidity that comes with other alternatives to seq2seq, such as grammar-based methods (Herzig & Berant, 2021).

We first introduce LAGr in the strongly-supervised setting where output graphs are aligned to the input sequences, thus allowing for standard supervised training. For the weakly-supervised case when the alignment is not available, we treat it as a latent variable. We infer the latent alingment with

a simple and novel approximate maximum-a-posteriori (MAP) inference approach which involves solving several minimum cost bipartite matching problems with the Hungarian algorithm (Kuhn, 1955). We then use the resulting aligned graphs to train the model. Our experiments demonstrate that in both strongly- and weakly-supervised settings LAGr significantly improves upon comparable seq2seq semantic parsers on the COGS and CFQ compositional generalization benchmarks (Kim & Linzen, 2020; Keysers et al., 2019).

The research we present in this thesis is also available as a long paper published in the 60th Annual Meeting of the Association for Computational Linguistics conference (ACL, 2022).

1.4 Thesis Outline

This thesis is organized as follows. In Chapter 2 we provide an extensive literature review covering classical rules-based approaches through to recent deep learning methods, and also review fundamental concepts such as grammars needed to understand these prior works.We also provide a technical background on deep learning methods for semantic parsing, such as sequence-to-sequence learning.

Chapter 3 expands on the summary presented in this section. We define what is meant by compositional generalization in the context of neural semantic parses, and review benchmarks and metrics proposed to measure this capability quantitatively. The remainder of the chapter is focused on reviewing prior research on compositional generalization for semantic parsing.

Chapter 4 introduces our main contribution: the LAGr algorithm. We describe the model and discuss the two training regimes in the strongly- and weakly-supervised settings. We then compare LAGr to related recent work on semantic parsing.

In Chapter 5, we present our extensive experimental results to demonstrate the effectiveness of LAGr on the COGS and CFQ compositional benchmarks. In particular, we describe how to convert sequential meaning representations to graphs, focusing on the semantic formalisms of COGS and CFQ (Neo-Davidson lambda calculus and SPARQL respectively). We then describe the training

details in our strongly- and weakly-supervised experiments. Finally, we conclude the chapter by a discussion of our results.

Chapter 6 concludes our findings with a brief discussion of the limitations of LAGr and proposals for future work to address these.

Chapter 2

Semantic Parsing: An Overview

In this chapter, we provide an overview on *semantic parsing*. Section 2.1 first discusses the task and the key components of semantic parsing. Then section 2.2 reviews various linguistic formalisms commonly encountered in the context of semantic parsing. Here we also introduce *Neo-Davidsonian lambda calculus*, the formalism we later use in our experiments in chapter 5. In the remaining part of this chapter, we review some classical semantic parsing algorithms, and finally, we focus our discussion on more recent literature using deep learning-based approaches for semantic parsing.

2.1 The Task

2.1.1 Objective

The objective of semantic parsing is to map a *natural language* (NL) sentence to a *logical form*: a representation of its meaning that can be interpreted by machines (Jia & Liang, 2016). Logical forms are also referred to as *meaning representations* (MRs), and are expressed in some formal language such as *lambda calculus* (section 2.2.1) or programming languages like SQL or SPARQL.

In semantic parsing literature, we often encounter three key notions: the use of a *grammar*, the *environment* in which a given utterance is interpreted, and finally the *formalism* used to represent logical forms. In the following sections, we discuss each of these concepts in greater detail.

2.1.2 Grammar

Many of the earlier semantic parsing systems are based on grammars (Woods et al., 1972; Waltz, 1978; Hendrix et al., 1978). In formal language theory, a grammar describes a set of rules that take words from the alphabet of a given language and generate syntactically valid sentences (Chomsky, 1957). Different grammars offer different computational complexities, while also determining the space of all possible valid expressions that can be generated (i.e., the language of the grammar). Throughout this thesis, we use this terminology to refer to systems that perform syntactic analyses of sentences, unless we explicitly state it otherwise. However, there also exist other types of grammars such as combinatory categorical grammars (Steedman, 1996) and so-called semantic grammars (Brown & Burton, 1975) that also provide semantic annotations. For now, we focus our discussion on syntax-based grammars, as these systems introduced a lot of the mechanisms that underlie how most grammars operate.

Syntax-based grammars do not concern the meaning of the generated sentence, but focus merely on their syntactic form. Following the seminal work of Chomsky (1957), we define a grammar as a tuple $G = (N, \Sigma, P, S)$ where

- N is a finite set of nonterminal symbols, often called variables or syntactic categories. These are used during the generation process, but they do not appear in the final string.
- Σ is a finite set of terminal symbols, i.e., the alphabet of the grammar that forms the content of a sentence, where Σ is disjoint from N.
- P is a finite set of production rules, where each rule takes a given string on the left-hand side and replaces it with another string specified on the right. The repeated application of these rules is what determines the expression we generate.
- $S \in N$ is a start symbol, a special nonterminal symbol, from which we begin the generation process.

Figure 2.1 shows an example of a grammar and corresponding *parse tree* for the sentence *"the hedgehog sleeps"*. We can derive this sentence by repeatedly applying the production rules, each

Figure 2.1: An example grammar and corresponding parse tree for the sentence "*The hedgehog dreams*". We use the following acronyms: NP for noun phrase, VP for verb phrase, DET for determiner, TV for transitive verb, IV for intransitive verb, and finally, N for noun.

S → NP VP

NP → DET N

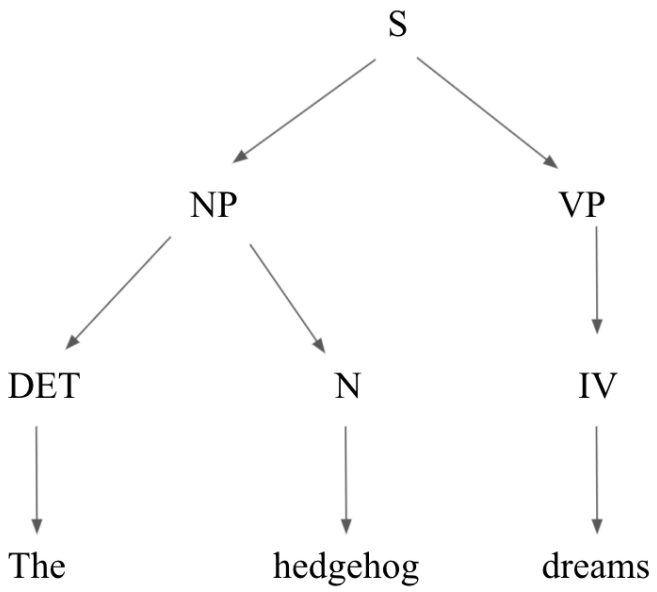
VP → TV NP
→ IV

DET → the
→ a
→ an

N → hedgehog
→ bird

IV → dreams

TV → eats
→ dreams



replacing a given syntactic category from the left-hand side (e.g. S, NP, VP etc.) with a string on the right-hand side (e.g. hedgehog, dreams etc.). For example, this grammar says that a sentence (S) consists of a noun phrase (NP) followed by a verb phrase (VP), and where a noun phrase has a determiner (DET) followed by a noun (N), where the determiner maps to either "the", "a", or "an", etc. As shown on the right, we can conveniently describe this parsing process by constructing a parse tree, a rooted tree whose nonterminal nodes are syntactic categories from N, and whose terminals are elements from Σ.

The example above uses a *context-free grammar* or CFG (Hopcroft et al., 1979) — an example of a highly structured grammar often encountered in semantic parsing literature. A CFG is called context-free as the application of any of the production rules happens regardless of the context of the nonterminals. This is in contrast with context-sensitive grammars where production rules are surrounded by a context of terminal and nonterminal symbols (Hopcroft et al., 1979). This effectively restricts the context in which certain words can appear in.

Synchronous context-free grammars (SCFG) are another formalism used in the semantic parsing literature (Jia & Liang, 2016; Yu et al., 2021). SCFGs are a generalization of CFGs that generates a pair of one-to-one aligned strings, instead of generating a single string. SCFGs were first used in machine translation tasks, where the pair of strings would correspond to the same expression in two languages (Chiang, 2007). In semantic parsing, SCFGs can be used to generate question-to-logical form templates that can help to create more examples to use in statistical learning techniques (Jia & Liang, 2016; Yu et al., 2021). We show an example in Figure 2.1.2 from Yu et al. (2021) that derives pairs of input utterances and corresponding tokens in the meaning representation (see Section 2.2).

Figure 2.2: Data augmentation via synchronous CFGs - example reproduced from Yu et al. (2021) with permission.

Non-terminals	Production rules
TABLE $\rightarrow t_i$ COLUMN $\rightarrow c_i$ VALUE $\rightarrow v_i$ AGG $\rightarrow \langle$ MAX, MIN, COUNT, AVG, SUM$\rangle$ OP $\rightarrow \langle =, \leq, \neq, \ldots$, LIKE, BETWEEN $\rangle$ SC $\rightarrow \langle$ ASC, DESC $\rangle$ MAX $\rightarrow \langle$"maximum", "the largest"...$\rangle$ $\leq \rightarrow \langle$"no more than", "no above"...$\rangle$ ...	1. ROOT $\rightarrow \langle$"For each COLUMN0 , return how many times TABLE0 with COLUMN1 OP0 VALUE0 ?", `SELECT COLUMN0 , COUNT ( * ) WHERE COLUMN1 OP0 VALUE0 GROUP BY COLUMN0` $\rangle$ 2. ROOT $\rightarrow \langle$"What are the COLUMN0 and COLUMN1 of the TABLE0 whose COLUMN2 is OP0 AGG0 COLUMN2 ?", `SELECT COLUMN0 , COLUMN1 WHERE COLUMN2 OP0 ( SELECT AGG0 ( COLUMN2 ) )` $\rangle$

A popular grammar that goes beyond syntactic analysis of sentences is combinatory categorical grammar (CCG) (Steedman, 1987; 1996; Abelson & Sussman, 1996). In addition to using production rules as in CFG, CCG also provides an elegant interface between syntax and semantics via the use of lexicons. An example lexicon for the sentence *"Flights to Boston"* may look as follows

$$\text{flights} := N : \lambda x.flight(x)$$
$$\text{to} := N/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x, y)$$
$$\text{Boston} := NP : Boston$$

where the left-hand side includes the lexicon items (words in the given sentence) and the right-hand size includes CCG *categories*, the main building block of CCG. Categories make up the nodes of the CCG parse tree, and their goal is to bridge the syntax of a sentence to its semantics. Let's consider the example of $N/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x, y)$. On the left, we find the syntactic component N/NP which contains simple part-of-speech tags (e.g. noun (N), noun phrase (NP), or verb phrase (VP)), and syntactic combination operators such as "/" or "\". The right hand-side represents the semantic part of the category, a lambda calculus function that defines how the lexicon items contributes to meaning via function composition (discussed more in depth in Section 2.2).

One limitation of the CCG formalism is that it may yield multiple valid parse trees. The ambiguity may arise from words having multiple entries in the lexicon. For example, the utterance "*New York*" may appear as NP : *new york state* or NP : *new york city* in the lexicon. Additionally, we can also have *spurious ambiguity*, i.e., different syntactic parses leading to identical semantics,

S → Specimen question | Spacecraft question
Specimen question → Specimen Emits info | Specimen Contains info
Specimen → “which rock ” | “which specimen ”
Contains info → “contains ” Substance
Substance → “magnesium ” | “calcium ”
Spacecraft question → Spacecraft Depart info | Spacecraft Arrive info
Depart info → “was launched on ” Date | “departed on ” Date

Table 2.1: Example Semantic Grammar, reproduced from Androutsopoulos et al. (1995).

thus the same logical form. In section 2.3.1, we discuss different approaches to overcome the non-uniqueness of CCG parse trees from previous literature.

So-called semantic grammars also commonly appeared in early semantic parsing literature (Brown & Burton, 1975; Hendrix et al., 1978; Waltz, 1978). Similarly to syntax-based techniques, these methods still parse the user's query to a parse tree. However, as shown in Table 2.1, in contrast to syntax-based grammars, categories in semantic grammars do not necessarily represent syntactic constituents (e.g. noun phrase, noun), but refer to semantic concepts (e.g. Substance, Contains etc.) related to a given application's domain.

Section 2.3 provides a more in-depth overview on semantic parsing approaches that leverage the grammars we reviewed here.

2.1.3 Environment

When performing semantic parsing tasks, we require a context in which we can interpret and resolve the full meaning of natural language sentences. We refer to this context as the *environment*. Environments can be unstructured, such as textual or visual documents, or structured such as databases or knowledge graphs. Figure 2.3 shows an example of a semantic parsing task of translating natural language questions to SQL queries grounded in a database environment from the Spider benchmark (Yu et al., 2018).

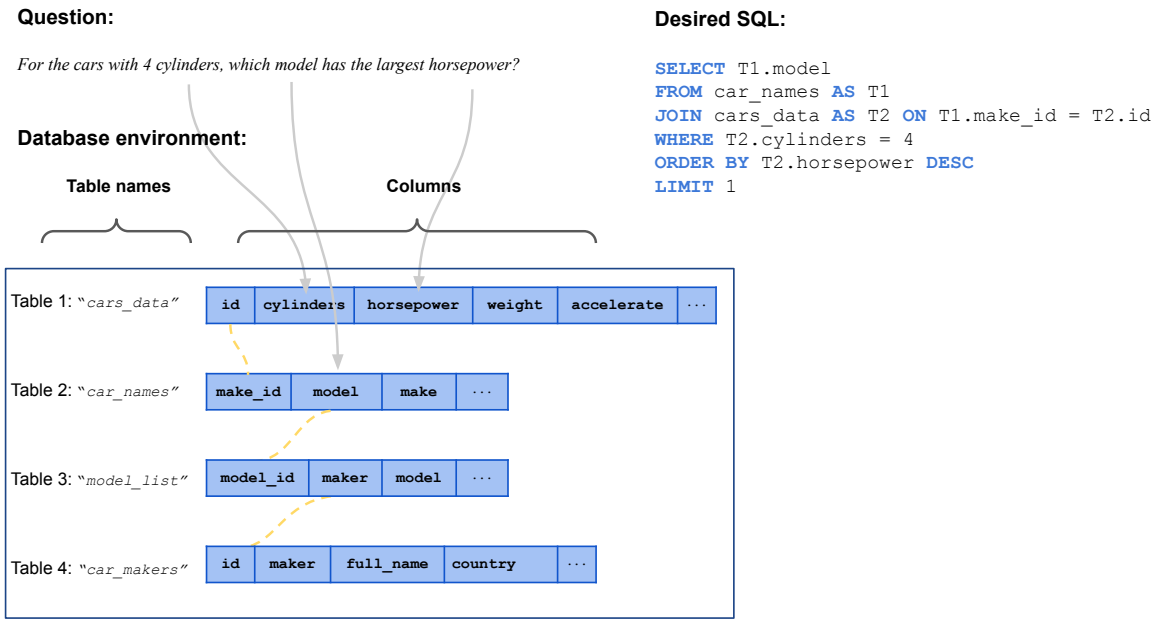


Figure 2.3: Example database environment taken from the Spider dataset (Yu et al., 2018) where natural language references get resolved in the schema of the database (shown with grey arrows). Yellow dotted lines show how foreign keys can be linked.

When working with databases, it is sometimes of interest to also assess whether the semantic parser is able to produce logical forms (i.e., database queries such as a SQL or SPARQL program) that can be executed against the database, and whether they can retrieve the correct answer to a given natural language question. In such scenarios, we also assume access to a database engine that can execute database queries. While execution engines are relevant in some of the literature we shortly review, they are less relevant for the remaining of this thesis, where we focus on producing the correct meaning representation, instead of retrieving answers.

2.2 Meaning representations

Formal semantics emerged with the goal of representing the meaning of natural language expressions. The field emerged in the 1970s and was greatly influenced by the seminal works of Montague (1970; 1973) and Montague (2019). Montague Grammar, as later referred to, was the

first to introduce the first formal system for English, arguing that natural languages could be treated and analysed as formal languages. He argued that it is possible to develop a single theory which explains the syntax and semantics of both natural and formal languages. Davidson (1967) and Kratzer & Heim (1998) argue that such theory of meaning should capture two characteristics of natural language: *novelty* and *compositionality* (Düsseldorf, 2017). Novelty concerns language users' ability to understand and produce an infinite number of sentences that they have never heard before. Closely linked to that, compositionality refers to the ability of language users to understand sentences they have never seen or heard before by composing the meaning of its constituents (words, expressions etc.). Montague semantics builds on *"the principle of compositionality"* (Pelletier, 1994), which states that *"the meaning of a compound expression is a function of the meanings of its parts and of the way they are syntactically combined"* (Landman & Veltman, 1984). While we do not go into more details on this from the perspective of formal semantics, compositionality is a principle that will guide our discussion on designing robust semantic parsers in the rest of this thesis, as we shortly see in chapter 3.

We now focus on the models of formal semantics that produce the meaning of natural language utterances by translating them to *first-order logic* formulas via *lambda calculus*.

2.2.1 First-order logic and lambda calculus

First-order logic (FOL) or predicate logic, is a formal system for logical reasoning. While propositional logic operates on propositions, FOL breaks propositions further down by introducing quantified variables over objects and individuals, as well as expressing properties and the relationships between them. This allows FOL to construct complex expressions and make general assertions basic building blocks. For example, using quantifiers and relations, FOC can express the proposition *"Marcus Aurelius is a stoic"* as

$$\exists x, \texttt{MarcusAurelius}(x) \wedge \texttt{stoic}(x),$$

i.e., that *"there exists x such that x is Marcus Aurelius and x is a stoic"*. We use $\exists$ to denote the existential quantifier, x as a variable, `MarcusAurelius` to express a constant, $\wedge$ for conjunction and `stoic` to denote a predicate.

Lambda calculus (Church, 1932) is another formal system in mathematical logic for expressing computation, such like propositional logic (PL) or first-order logic (FOL). It emerged as an extension to FOL to include the lambda (λ) operator. The λ operator acts differently from a quantifier in FOL - its role is to form new functions, which allows them to use existing functions to build more complex composite functions. This allows to express higher-order logic where the arguments are functions instead of simple entities (Briscoe, 2011). The elements of lambda calculus are lambda-expressions. These are functions that can be either typed to constrain the arguments they take and the values they return, or can be untyped with no such constraints. Lambda expressions are composed of the following components:

- **Variables**: Abstract constants whose exact values are not pre-determined until they are substituted for constants (a process called β reduction). These are usually denoted by lower-case letters, e.g. x.
- **Abstractions**: Function definition, e.g. $\lambda x.A$ where x is a variable, and A is a lambda term, where λ binds the variable x.
- **Applications**: Functions application, e.g. (AB) where A and B are lambda terms.

These components are used together to form a lambda calculus expression whose objective is to enumerate the set of constants that can be substituted for a variable x to obtain a well-formed expression in FOL.

2.2.2 Evolution of Neo-Davidsonian Semantics

Let us now turn our discussion to Neo-Davidsonian semantics (Kim & Linzen, 2020; Maienborn et al., 2011), the formalism we use in chapters 3 and 5. Neo-Davidsonian semantics refers to the combination of formal semantics with event semantics. The goal of event semantics is to formally

describe an event or an action, such as *"Brutus killed Caesar"* — the classical example often encountered in linguistics literature. Standard FOL represents this sentence as `stab(Brutus, Caesar)`, where `Brutus` and `Caesar` are entities that the `stab` verbal predicate takes as arguments. Although, this representation works well for simple sentences that do not have adverbial modifiers (i.e., words that modify a part of the sentence, such as verbs - e.g. quickly), they are problematic in more complex scenarios. For example, adding an instrument to the verbal predicate such as *"Brutus stabs Caesar with a knife."* would yield `stab(Brutus, Caesar, knife)`. Alternatively, we can also add a location to the verbal predicate such as *"Brutus stabs Caesar in the agora."*, which could be represented as `stab(Brutus, Caesar, agora)` in FOL. While we chose to represent both modifications by adding a new argument to the verbal predicate, it is clear that a *location* is not a participant in the same way as a *knife* is. Consequently, any addition of adverbial modifiers to the predicate would require changing the arity and type signiture of the predicate. Instead, verbal predicates should have a way to be directly refer to. This is necessary such that we can describe complex actions that may have variable number of arguments and adverbial modifiers.

In seminal work of Davidson (1967), Davidson argues that verbal arguments should show up in verbal predicates through event variables, where event variables could be related to their adverbial modifiers. Building on Davidson's idea, Neo-Davidsonian semantics — often attributed to subsequent work by Parsons (1990) — introduces *thematic roles* to posit a relationship between the event predicates and their non-event syntactic arguments. Thematic roles describe the participants of events. Commonly used examples are thematic roles such as *theme* denoting the object of the event, *agent* denoting the subject or *recipients* denoting the recipient of something. With the introduction of thematic roles, we can rewrite the sentence *Brutus stabs Caesar with a knife in the agora* as follows:

$$\exists e\ \texttt{stab}(e) \wedge \texttt{stab.theme}(e, \text{Caesar}) \wedge \texttt{stab.agent}(e, \text{Brutus}) \wedge \texttt{stab.location}(e, \text{Brutus}) \wedge \texttt{stab.instrument}(e, \text{knife}).$$

Chapter 3 and 5 showcase further examples of sentences and their respective logical forms using the Neo-Davidsonian formalism.

2.3 Approaches to Semantic Parsing

In the last section, we saw the key components of semantic parsing: grammars that can be used to generate natural language sentences, environments that provide the context in which a sentence is interpreted, and finally, we discussed various formalisms to represent the meaning of a sentence. Given this background, we now review various approaches to performing semantic parsing. We begin our discussion on classical approaches using rule-based and statistical learning techniques, then we dive into more recent deep learning-based methods.

2.3.1 Classical Approaches

Rule Based Semantic Parsing

Semantic parsing first emerged through applications for natural language interfaces to databases (NLIDBs) as early as the late sixties and seventies (Woods et al., 1972; Hendrix et al., 1978; Waltz, 1978; Codd, 1975). These systems allow users to access information from databases by expressing questions in natural language.

Some of the early NLIDB systems retrieved answers by the use of pattern matching techniques (Johnson, 1984). Given the following simple example table from a database

Countries table

Capital	**Country**	**Language**
Budapest	Hungary	Hungarian
Lisbon	Portugal	Portuguese
...	...	...

a simple pattern-matching system could specify the following rule:

Pattern: ... “capital” ... <country>
Action : Retrieve **Capital** column of row where **Country** column has value <country>.

This rule captures a mention of a country name (i.e., <country>) and the word “capital” in the user’s sentence, and reports the respective capital of the mentioned country. While these systems

worked reasonably well for simple domains where the input sentences satisfied the rules the system covered, they were brittle for sentences the system was not designed to capture.

Another line of work focused on syntax-based grammars that translated natural language sentences to parse trees that contained syntactic annotations (see Section 2.1.2). While some systems only focused on the syntactic analysis of sentences, others also translated the parse trees to the final meaning representation such as the relevant database queries, using rules based on the syntactic information of the parse tree. One of the most well-known examples of the latter is LUNAR (Woods et al., 1972) that operated on a database containing information about moon rocks. A limitation of this latter approach is that it can be challenging to define appropriate rules between the parse tree and an MR such as a real-world database query language (e.g. SQL).

Other approaches (Brown & Burton, 1975; Hendrix et al., 1978; Waltz, 1978) leveraged semantic grammars that also contained semantic annotations to sentences. These systems were sometimes more accurate than previous approaches as semantic grammars introduced further constraints on how to generate a meaning representation from a given parse tree. However, since these semantic grammars were designed for a specific knowledge domain, naturally, they were difficult to adopt to new applications with different domains. For this reason, semantic grammars gradually lost popularity. For a more extensive overview on the early NLIDB systems we refer the reader to Androutsopoulos et al. (1995).

Statistical Learning Techniques

The problem with rule-based systems is that they often make portability to other databases laborious and difficult due to its domain-specific components. For this reason, by the end of the 1990s, there was growing interest to leverage corpus-based methods to automatically construct these NLIDB systems using more flexible statistical learning techniques. Early corpus-based approaches did not deal with producing meaning representations, but instead, they focused on syntactic parsing whose objective is to produce syntactic annotations to natural language sentences.

The pioneering work of Zelle & Mooney (1996) was the first to demonstrate that statistical learning techniques can be used to directly produce executable meaning representations using a

corpus of sentences paired with their respective database queries. To do so, they used the CHILL system, which treats parser acquisition as the learning of search-control rules within a logic program representing a shift-reduce parser. In particular, CHILL performed two tasks. First, the training instances were used to formulate an overly-general initial parser that produced many spurious analyses for a given input sentence. The parser was then constrained by inductively learning search-control heuristics that eliminate spurious parses.

One limitation of Zelle & Mooney (1996) was that it still relied on hand-crafted lexicons. Subsequent work by Thompson & Mooney (2003) showed that such semantic lexicons could in fact be acquired by training a model from a corpus of phrase-meaning pairs. The authors showed that the final learned lexicon performs nearly as well at answering questions than when the hand-built lexicon is used. Furthermore, they showed that when translating the original training corpus to Spanish, Japanese and Turkish, they were able use the same framework to build semantic parsers in a variety of languages. However, the automatic acquisition of semantic lexicons still involved a non-trivial effort for annotations. Thompson & Mooney (2003) addressed this by proposing an active learning approach that selects the most informative examples to use for annotation and for training. They were able to show that their active learning approach can significantly minimize the annotation cost as compared to a random sample of training examples.

While earlier works assume access to syntactic and semantic annotations between the input-output pairs, Zettlemoyer & Collins (2005) propose to directly learn to map natural language utterances to their corresponding lambda calculus (as discussed in section 2.2). The key challenge in this approach is to identify the entities in a natural language utterance that refer to entities in the logical form. Zettlemoyer & Collins (2005) address this by using a learning algorithm to induce a CCG (see section 2.1.2) which maps sentences to their logical forms, along with a probabilistic model that assigns a distribution over parses under the grammar. Their approach, Probabilistic CCG (PCCG) aims to resolve the ambiguity of having multiple valid parse trees with the CCG formalism by ranking the induced parses in order of probability conditional on the input sentence. Specifically, they do this by learning a pair (T, L), with T describing the parse tree containing a sequence of steps to derive the logical form L, given sentence S. First, they assume a function

f mapping the triple (T, L, S) to a feature vector in R^d, then they estimate the probability of a particular parse by the following log-linear model:

$$P(T, L|S) = \frac{e^{f(T,L,S)\theta}}{\sum_{(L,T)} e^{f(L,T,S)\theta}} \tag{2.1}$$

where θ stands for the model parameters and where the sum is over all valid parses for the input sentence S. This paper uses simple local features equivalent to the count of the number of times each lexical entry appears in a given parse T. The parsing (inference) involves computing the most likely logical form conditional on the input sentence and the estimated model parameters as follows:

$$\arg\max_{L} P(L|S;\theta) = \arg\max_{L} \sum_{L} P(L, T|S;\theta), \tag{2.2}$$

where the arg max is taken over all logical forms L, while marginalizing out the hidden parse tree T by summing out all possible parse trees that produce L. To do this, the authors used a dynamic programming algorithm, similar to those used in CKY-style algorithms often used to find the most likely parse in probabilistic context-free grammars.

The learning of the PCCG involves both learning a lexicon, which defines a set of parse trees for each training example, and the estimation of model parameters that specifies the distribution over the parse trees. This is achieved by the GENLEX function which takes a pair of natural language with its semantic representation, and outputs a large set of potential lexical entries — with vast majority being spurious — using a set of pre-specified rules. The second step of the algorithm then selects a subset of the entries and updates the model parameters using gradient ascent.

While this method works well, the use of the CCG formalism can be restrictive for raw natural language that may contain colloquial language involving informal expressions, omitted words, simplified grammar and flexible word order. To extend the work of Zettlemoyer & Collins (2005), while keeping the advantages of the CCG formalism, Zettlemoyer & Collins (2007) introduce new combinators in CCG. These combinators allow for a more flexible treatment of certain parts of the grammar, like word order, insertion of new lexical entries along with a learned cost for each new operator. In addition, they also introduce a more efficient hidden perceptron algorithm that

learns CCG in an online fashion, rather than in a batch-manner. In later work by Kwiatkowski et al. (2011), a more general framework is developed that is agnostic to the choice of natural language and of the corresponding meaning representation. The approach introduces a probabilistic CCG that defines how the meaning of indiviual words can be combined to analyze complete sentences. They define the hypothesis space of all grammars from the training data, and develop an online learning algorithm that efficiently searches this space while simultaneously estimating the parameters of a log-linear parsing model. Experiments demonstrate high accuracy on benchmark data sets in four languages with two different meaning representations.

2.3.2 Deep Learning Approaches

While many ideas from these classical methods are still relevant in recent literature, the field of semantic parsing has shifted towards more flexible deep learning-based approaches that often perform better without the need for domain-specific feature engineering.

The backbone of deep learning models is neural networks, or multi-layer perceptrons (MLPs) which define a parametrized mapping $f(\mathbf{x}, \theta) = y$ that approximates a function $f^*(\mathbf{x}) = \mathbf{y}$ that maps some input vector $\mathbf{x}$ to a target label y (Goodfellow et al., 2016). In order to best approximate f^*, neural networks are trained to estimate θ using a training corpus of input-output pairs. During training, the goal is to drive f to match f^* by minimizing an objective (*loss*) function, such as the negative log likelihood of the training data.

Recurrent neural networks such as *long short-term memory networks* (LSTM) (Hochreiter & Schmidhuber, 1997) are one of the dominant neural network architectures used in modeling sequential data. An example of working with sequential data is *language modeling* where the goal it to predict the next words that follow a given text. Language models do this by estimating the joint probability $p(\mathbf{x})$ of a sequence of words $\mathbf{x} = (x_1, \ldots, x_N)$ as a product of conditional probabilities:

$$p(\mathbf{x}) = \prod_{t=1}^{N} p(x_t | x_1, \ldots, x_{t-1}),$$

where the conditional probabilities can be approximated by neural networks.

Sequence-to-sequence Learning

In semantic parsing we consider both a source sequence $\mathbf{x} = (x_1, \ldots, x_N)$ (i.e., natural language utterance) and a target sequence $\mathbf{y} = (y_1, \ldots, y_T)$ (i.e., meaning representation) and estimate the conditional probability $p(\mathbf{y}|\mathbf{x})$. The dominant neural approach to this problem is *sequence-to-sequence* methods (seq2seq) in which two neural networks are jointly trained to maximize the conditional likelihood of input-output pairs from a given training corpus (Sutskever et al., 2014; Bahdanau et al., 2015; Cho et al., 2014). In particular, seq2seq methods use a neural network (as the *encoder*) to map a source sequence $\mathbf{x}$ to a fixed-dimensional vector $\mathbf{v}$. Another neural network (the *decoder*) takes $\mathbf{v}$ to predict the target sequence via estimating $p_{\theta_{dec}}(\mathbf{y}|\mathbf{v})$. By Bayes' rule, this distribution can be decomposed into a product of conditional probabilities of individual target tokens. Consequently, the target sequence can be generated in an auto-regressive fashion, omitting one token at a time, while conditioning on previously generated output tokens:

$$
\begin{aligned}
p_{\theta_{enc},\theta_{dec}}(\mathbf{y}|\mathbf{x}) &= p_{\theta_{enc},\theta_{dec}}(y_1, \ldots, y_T|x_1, \ldots x_N) \\
&= \prod_{t=1}^{T} p_{\theta_{dec}}(y_t|f_{\theta_{enc}}(\mathbf{x}), y_1, \ldots, y_{t-1})
\end{aligned}
$$

.

One limitation of using recurrent seq2seq neural networks is that they require compressing the entire content of the source sequence into a fixed-dimensional vector. This makes it difficult to model long-term dependencies between the source and target sequences. To alleviate this information bottleneck, Bahdanau et al. (2015) introduced *attention*, a mechanism that allows decoders to ingest an additional context vector by searching for parts of the source sequence that are relevant to predicting the next target word. Attention is done by considering a set of *values*, *queries* and *keys*. In its original form Bahdanau et al. (2015), we define both values and keys as the encoder's hidden states of the source sequence denoted by $(\mathbf{h_1}, \ldots, \mathbf{h_N})$, and queries as the hidden state of the decoder $\mathbf{s_j}$ at a given position j. Specifically, we define $\mathbf{c_i}$ as the context vector, and

estimate it as follows:

$$\mathbf{c_i} = \sum_{j}^{N} \alpha_{ij} \mathbf{h_j}, \tag{2.3}$$

where α_{ij} denote the normalized attention weights and are calculated as

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})} \tag{2.4}$$

$$e_{ij} = f_{att}(\mathbf{s_{i-1}}, \mathbf{h_j}) \tag{2.5}$$

where equation 2.5 is the alignment model that estimates the unnormalized alignment between a key at the j-th input position and a query at the $i-1$-th output position. While there exists a number of ways to define f_{att} (Luong et al., 2015), here we consider *the scaled dot-product attention model* where $f_{att}(\mathbf{s_{i-1}}, \mathbf{h_j}) = \frac{\mathbf{s_{i-1}} \cdot \mathbf{h_j}}{\sqrt{d}}$ with d denoting the dimension of $\mathbf{h_j}$.

Sequence-to-sequence with Transformers

While recurrent architectures are powerful for a number of tasks, they only scale linearly with the length of the source sequence, both during training and inference. Vaswani et al. (2017) introduced the Transformer architecture that demonstrated that attention alone can replace recurrent architectures. Thanks to modern GPU machines, Transformers allowed for effective parallelization, reducing the number of computation steps to a constant [1].

Specifically, using the scaled dot-product attention we saw above, Transformers consists of so-called *self-attention* layers that build increasingly-refined contextual representations where each word is represented in comparison to all words in a given context. The result of these comparisons yields the attention scores that determine how much a word's representation should contribute to another word's representation. In particular, the querying mechanism we saw earlier is now done in each of the L stacked encoder layers where in each layer, queries, keys and values are defined as

[1] While Transformers initially referred to the original encoder-decoder architecture of Vaswani et al. (2017), with the advent of encoder-only Transformer architectures such as BERT (Devlin et al., 2018), we no longer use this terminology to only refer to encoder-decoder architectures.

follows:

$$\begin{aligned} \mathbf{q_i} &= W_q \mathbf{h_i^l} \\ \mathbf{k_i} &= W_k \mathbf{h_i^l} \\ \mathbf{v_i} &= W_v \mathbf{h_i^l} \end{aligned} \tag{2.6}$$

where W_q, W_k, W_v define a linear projection of an input embedding h_i at the i-th input position and where the self-attention mechanism in each layer performs the following transformation on h_i:

$$\begin{aligned} e_{ij} &= \frac{\mathbf{q_i} \cdot \mathbf{k_j}}{\sqrt{d_k}} \\ a_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})} \\ h_{i^{l+1}} &= \sum_{k}^{N} a_{ik} \mathbf{v_k} \end{aligned} \tag{2.7}$$

where d_k denotes the dimension of keys. The repeated application of this self-attention mechanism allows to build increasingly-refined contextual representations, fusing information from other, relevant parts of a given text. Vaswani et al. (2017) also introduce the so-called *multi-head attention* that uses multiple self-attention mechanisms in parallel, each with its own linear projection matrices, where the output of each head — using information from different representational subspaces — is aggregated to yield an even more refined representation.

These innovations helped overcome the two key limitations of recurrent architectures: they allowed model training to be done in parallel, while helping language modeling tasks to capture dependencies from much wider context windows.

In follow-up work, Devlin et al. (2018) introduce Bidirectional Encoder Representations from Transformers (BERT). BERT is trained with the novel "masked language model" objective in which parts of the input sentence are randomly masked out which then have to be predicted by the model using the available context. This unsupervised training regime allowed for pretraining large-scale language models that can serve as powerful backbone models and can be easily finetuned with an

additional output layer to perform a variety of downstream tasks (see further discussion in Chapter 3).

With these foundations, we now provide an overview on prior work for seq2seq-based semantic parsers.

Recurrent Approaches for Semantic Parsing

Dong & Lapata (2016) were the first to adopt the formulation from Sutskever et al. (2014) for semantic parsing using recurrent neural networks with long short-term memory units (LSTM) (Hochreiter & Schmidhuber, 1997). They propose two model variants - a vanilla seq2seq model and seq2tree that uses a hierarchical tree decoder to explicitly capture compositionality in the logical form. In seq2tree, they introduce a special non-terminal token that represents a sub-tree in the logical form (e.g. representing content within parentheses). When this non-terminal token is predicted, the authors use an LSTM to condition decoding on the nonterminal's hidden vector. They demonstrate that seq2seq can perform semantic parsing similarly to traditional methods, while avoiding the need for domain-specific features. Furthermore, they show that a hierarchical decoder as well as the use of attention (Bahdanau et al., 2015) between input-output pairs provide a considerable improvement across the board.

Dong & Lapata (2016) also employ *entity anonymization*, a commonly used strategy in neural semantic parsers to help with the prediction of rare entities. This is performed by first replacing entities and numbers with placeholders for their respective types and unique identifiers, and then later substitute the predicted placeholders into the corresponding logical constants as a postprocessing step. As an example, the sentence *"employees with a salary of **50000**"* would be replaced by *"employees with a salary of **num0**"* and yield the logical form "`employees(ANS), salary_greater_than(ANS, num0)`".

Another important work aimed at tackling the occurrence of rare entities is Pointer Networks (Vinyals et al., 2015). Similarly to the original seq2seq framework, the authors propose to model the conditional probability of the target sequence given the input sequence. But instead of using soft attention to use relevant hidden states from the encoder as a context vector, Pointer Networks

use attention to select an entity from the input sentence as an output token, and thereby allow for variable-size output dictionaries.

While this method only allows to generate output tokens based on selecting from the input, Jia & Liang (2016) introduce *attention-based copying*, which combines attention-based seq2seq models with the copying mechanism from Pointer Networks. In particular, the decoder is augmented with a special `COPY` token in the output vocabulary, which, when predicted, generates the output by copying a word from the input sequence that has the highest attention score.

In addition, Jia & Liang (2016) also propose the "data recombination" framework to embed domain knowledge about logical regularities into neural semantic parsers. Specifically, they perform data augmentation by first building a generative model that generate new samples that adhere to certain logical regularities captured by a SCFG grammar. With this augmented domain-specific dataset, they can then train a domain-general sequence-to-sequence model with their proposed attention-based copying mechanism.

Although seq2seq methods achieved great success in semantic parsing tasks, they often struggle to generate logical forms that are both syntactically or semantically correct given the input utterance. A large body of work focuses on addressing this via constraining the output space of neural semantic parsers to adhere to the explicit structure of the corresponding logical form (Chen et al., 2018; Krishnamurthy et al., 2017; Wang et al., 2018a;b; Scholak et al., 2021). For example, when semantic parsing is done to produce executable database queries, a simple and commonly used strategy is *execution-guided decoding* (Chen et al., 2018; Wang et al., 2018a). This strategy consists of correction mechanisms that can guide models to adhere to certain type constraints in the logical forms, and make models disregard predictions that execute to undesirable outcomes such as runtime errors or empty outputs.

Wang et al. (2018b) propose another way to incorporate syntactic constraints into neural methods in which parts of the logical form are decoded as partial trees that will then guide the generation of the final logical form. Specifically, they use two RNNs: a "rule RNN" that generates the topology of the tree using a CFG (as discussed in section 2.1.2), and a "word RNN" that generates the output tokens for each leaf node. Krishnamurthy et al. (2017) develop a similar approach to constrain the

output space of seq2seq models, specifically for semantic parsing. Their decoder uses an LSTM with attention to select from a set of actions defined by a grammar that produces well-typed logical forms.

Transformer-based approaches for Semantic Parsing

With the advent of Transformers, a number of works began to incorporate pretrained language models in semantic parsing (Xu et al., 2017; He et al., 2019; Hwang et al., 2019; Wang et al., 2020; Scholak et al., 2020). While we provide a more extensive review of semantic parsing methods using pretrained models in section 3.2.1, here we discuss the general strategy on using Transformers for semantic parsing tasks.

SQLova by Hwang et al. (2019) is the first semantic parsing approach that uses a table-aware Transformer-based encoder. Specifically, they use a Transformer to jointly encode the question in the context of the database by ingesting the concatenation of the input question and table headers of the database as input. Specifically, they use BERT (Devlin et al., 2018), an encoder-only Transformer, to first obtain a contextualized representation of the question and the database, which is fed to two separate LSTMs that update the question and database representations independently from one another. As mentioned previously, in typical seq2seq models, the output is not explicitly constrained to adhere to syntactic regularities, often making it hard to respect the strict structural requirements of formal language generation tasks. SQLova overcomes this by a proposed NL2SQL layer that performs a syntax-guided sketch via the use of six task-specific modules, where each module predicts specific parts of the logical form. For example, the `select-column` module predicts the columns that should appear in the final `SELECT` clause of the database query. This is achieved by a special *column attention* mechanism that dynamically changes the representations of question words based on their relevance to a given column that is being considered for selection.

Instead of using an additional question and database LSTM to update the output of the pretrained BERT model, He et al. (2019) argue that BERT already produces good enough representations. Their suggestion is to use a global context vector that can be obtained from the special `[CLS]` token of BERT. Renamed as `[CTX]`, this token already contains enough information about the different

columns in the database schema, which makes it unnecessary to further update their representations. With this mechanism the authors are able to remove the use of the additional LSTM encoding layer and column attention from Hwang et al. (2019). Additionally, while Hwang et al. (2019) use separate binary classifiers (i.e., modules) to select relevant columns in the final SQL query, where each is optimized independently, He et al. (2019) suggest a different mechanism to model the relationship between columns. Their model X-SQL uses a global ranking approach by using the Kullback-Leibler divergence as its objective to bring all columns into the same representational space. Furthermore, they propose a learnable type embedding to differentiate between the question, special tokens and the various categorical and numerical columns of the database.

2.4 Summary

In this chapter, we reviewed the semantic parsing task, grammars, environments and various formalisms for representing the meaning of natural language sentences. We also covered rule-based approaches that rely on grammars, and methods that leverage corpus-based statistical learning techniques to perform semantic parsing. The encoded domain knowledge in these earlier techniques can endow models with a built-in awareness of the compositional structure of meaning representations, which can provide models that are both interpretable and can produce accurate meaning representations in specific domains. On the other hand, these models require grammars, high-quality lexicons or manually designed linguistic features which can be laborious to acquire. We discussed that neural networks can overcome this and offer more flexible and generic solutions across various semantic parsing tasks, domains and meaning representations that do not require such domain-specific resources. For the rest of this thesis, we focus on these deep-learning based approaches. Specifically, in the next chapter we zoom in on why compositionality is an important consideration in neural semantic parsers, and describe ways to incorporate ideas from the earlier works we saw in this chapter.

Chapter 3

Compositionality in Semantic Parsing

This chapter provides an overview on compositionality in machine learning for semantic parsing, and its role in effectively generalizing to distributions different from that of the training dataset. In section 3.1 we define compositional generalization, as a desideratum in natural language processing models. Next we discuss the state of compositional generalization in current machine learning literature, and review various benchmarks proposed to measure and evaluate it. Finally, in section 3.2, we review compositional generalization approaches specifically designed for semantic parsing.

3.1 Compositional Generalization in NLP

3.1.1 Challenges and Desiderata

Natural language understanding has seen significant progress over the past years across a wide spectrum of tasks from machine translation, reading comprehension to semantic parsing. Much of this progress has been fueled by the introduction of large-scale pretrained models such as BERT (Devlin et al., 2018) as we discussed in chapter 2. Though the progress is undeniable, these advancements do not always allow for reliably deploying models in real-world applications. In fact, over recent years there has been a large body of research demonstrating how language models systematically fail on certain tasks (Lake & Baroni, 2018; sin; Keysers et al., 2019; Bahdanau et al., 2019; Gontier et al., 2020). Specifically, these works show that neural methods often lack

a systematic understanding of language, and are unable to learn general rules on how to compose seen words in novel contexts they did not encounter in the training set. Instead, models tend to latch onto statistical artifacts in datasets (Jia & Liang, 2017), thus preventing them from systematically differentiating between irrelevant contextual changes and important information.

Such systematicity is required for models to generalize well to new examples where that exhibit contextual differences from what was observed in the training set — a key requirement for deploying them in real-world applications, where domain shifts are abundant. For example, as shown by Herzig & Berant (2021) a model that observes the questions *"What states border China?"* and *"What is the largest state?"* at training, is still unable to generalize to questions like *"What states border the largest state?"*. In fact, even simpler concepts such as negation is often not understood by models. For example, when the word "older" is negated in the question *"What are all the song names by singers who are older than average?"* to *"not older"*, even the recent state-of-the-art semantic parser by Wang et al. (2020) will ignore the negation and produce the same logical form `SELECT singer.Song_Name FROM singer WHERE singer.Age > (SELECT Avg( singer.Age) FROM singer)`. The lack of systematicity in reasoning about entities and their relations to another and to their surroundings is often attributed to the problem of *compositional generalization* in models. We define compositional generalization as the model's ability to combine known entities in novel ways and thus generalize to data points that do not follow the training distribution. Another term that is often interchangeably used in current literature is *systematic generalization*. In this thesis, we use the former terminology.

3.1.2 Evaluation

Although, research in compositional generalization has been quickly accelerating in recent years, there still has not been a unified way to define and measure compositional generalization in models. Instead, researchers have proposed a number of benchmarks with datasets (often referred to as *splits*) that split data into a training and test sets where both contain the same concepts, but where concepts appear in different contexts (Lake & Baroni, 2018; Kim & Linzen, 2020; Keysers et al., 2019; Bahdanau et al., 2019).

SCAN

Lake & Baroni (2018) propose the SCAN benchmark to test compositionality in sequence-to-sequence models. In particular, SCAN provides a grounded environment where the task is to translate a sequence of navigation commands expressed in a simple, artificial language into an sequence of actions. The input vocabulary consists of 13 commands, whereas the outputs are expressed in terms of possible actions the model can take, as shown in Table 3.1.

Instructions	**Action Sequence**
jump	→ JUMP
jump left	→ LTURN JUMP
turn left twice	→ LTURN LTURN
jump after walk twice	→ WALK WALK JUMP

Table 3.1: Examples from the SCAN dataset Lake & Baroni (2018)

SCAN measures compositionality by introducing a set of train-test splits. These splits divide the data into a training and a test set such that the primitive instructions are the same, while the combinations in which these instructions are observed are different. They provide a *random-split*, where examples are randomly sampled without replacement, a *length-split* where the training set contains action sequences up to 22 actions, whereas the test set contains longer sequences. They also provide two primitive splits that expose the model to primitives such as *jump* without any context in the training set. Then they test the model's ability to combine these primitives with other primitives such as *jump twice* at test time. These splits were directly designed to test the model's ability to understand familiar concepts seen in the training set in novel combinations.

CFQ

While SCAN pioneered compositional generalization research on sequence-to-sequence models, two limitations remain: (i) their data splits are based on heuristics, rather than a metric for measuring compositionality, and (ii) advances in compositionality as measured by synthetic benchmarks like SCAN may not translate to real-world language tasks.

Questions	**SPARKQL queries**

Who directed Elysium?	`SELECT DISTINCT ?x0`
	`WHERE {`
	`  ?x0 a ns:people.person .`
	`  ?x0 ns:film:director.film m.0gwm_wy}`

Was a film editor a star of M1?	`SELECT count(*)`
	`WHERE {`
	`  ?x0 a ns:film.editor`
	`  ?x0 ns:film.actor.film M1}`

Table 3.2: Examples of natural language questions paired with their respective meaning representations expressed as SPARQL queries from the CFQ dataset (Keysers et al., 2019).

In response to (i), Keysers et al. (2019) propose the *distribution-based compositionality assessment* (DBCA) metric to formalize and quantify the extent to which models need to use compositional generalization to solve tasks for a given train-test split. They adopt the terms *atoms* to refer to primitive units (words) and *compounds* (combinations of words, i.e., sentences) to refer to composed sequences of atoms. Using this terminology, compositional generalization is viewed as the ability to assemble familiar atoms in novel compounds. To measure the difficulty of a given train-test split is, the DBCA metric considers the divergence of atom and compound distributions. DBCA allows one to create data splits of varying difficulty by increasing the divergence between the compound distribution of a given candidate train and test split, while maintaining a low divergence in the distribution of atoms.

Using the DBCA metric, Keysers et al. (2019) also introduce a new dataset, *Compositional Freebase Questions (CFQ)*. CFQ consists of pairs of natural language questions and SPARQL queries, where the queries retrieve information from the Freebase knowledge base (Bollacker et al., 2008), as shown in Table 3.2.

Similarly to SCAN, CFQ also includes a set of train-test splits that require varying levels of compositional generalization. Notably, using the DBCA metric, Keysers et al. (2019) introduce the *Maximum Compound Divergence* splits that were generated by keeping the atom divergence steady, while increasing the compound divergence to different degrees. Table 3.3 shows the atom and compound divergence for all splits in the CFQ dataset.

	Split Method	$\mathcal{D}_A$ Atom Divergence	$\mathcal{D}_C$ Compound Divergence
CFQ	Random	0.000	0.000
	Output Length	0.033	0.176
	Input Length	0.047	0.062
	Output Pattern	0.000	0.008
	Input Pattern	0.000	0.005
	MCD$_1$	0.020	**0.694**
	MCD$_2$	0.020	**0.713**
	MCD$_3$	0.020	**0.704**

Table 3.3: Comparison of relevant measurements for different split methods on CFQ Keysers et al. (2019).

COGS

Kim & Linzen (2020) propose the *Compositional Generalization for Semantic Parsing* (COGS) dataset where examples include natural language sentences paired with their respective lambda calculus logical forms (see Table 3.4). COGS contains the standard data splits for training, development and testing, but it also includes a generalization set specifically designed to assess compositionality in language models. The generalization set in COGS is different from its predecessors in that it focuses on a more diverse set of linguistic abstractions that models are expected

Sentence	**Lambda calculus**
A monkey ran.	monkey(x_1) AND run.agent(x_2,x_1)
The pumpkin was shortened.	*pumpkin(x_1); shorten.theme(x_3,x_1)
The girl was lended the rose.	*girl(x_1); *rose(x_5); lend.recipient(x_3,x_1) AND lend.theme (x_3,x_5)
Michael rolled the donut in a house.	*donut(x_3); roll.agent(x_1, Michael) AND roll.theme(x_1,x_3) AND donut.nmod.in(x_3,x_6) AND house (x_6)

Table 3.4: Examples of natural language sentences paired with their respective meaning representations in lambda calculus from the COGS dataset (Kim & Linzen, 2020).

Case	Training	Generalization
Subject → Object	A **hedgehog** ate the cake.	The baby liked the **hedgehog**.
Object → Subject	Henry liked a **cockroach**.	The **cockroach** ate the bat.
Primitive → Object	**Paula**	The child helped **Paula**.
Depth generalization	Ava saw the ball **in the bottle on the table.**	Ava saw the ball **in the bottle on the table on the floor.**
Active → Passive	Emma **blessed** William.	A child was **blessed**.

Table 3.5: Examples from Kim & Linzen (2020) that show various linguistic phenomena included in the COGS generalization set.

to learn about. One such abstraction is the ability to interpret words and concepts (often referred to as *primitives*) in novel combinations and contexts. For example, some words in the training set may only appear in the input sentences as standalone words without any context (e.g. *Paula*), while they appear in a concrete context in the generalization set (e.g. "*The child helped Paula*"). Other examples require models to understand unseen combinations of modified phrases and grammatical roles, generalizing phrase nesting to unseen depths, verb argument structure alternation, and sensitivity to verb class. We refer the reader to Table 3.5 where we show examples for each linguistic abstraction from COGS for different cases of compositional generalization.

3.2 Compositional Semantic Parsing

As discussed in section 2.3.1, traditional grammars-based semantic parsers (Zettlemoyer & Collins, 2005; Liang et al., 2012) explicitly define the meaning of individual words and phrases and ways to combine them in order to obtain the meaning representations. This makes grammar-based approaches inherently compositional. While neural semantic parsing provide a more general approach to directly translate utterances to logical forms without having to make domain-specific assumptions and specify grammars, they do not explicitly capture compositionality.

For this reason, prior works on compositional generalization for semantic parsing often take inspiration from grammar-based approaches. In particular, a unifying theme for most works is to capture a mapping between logical constructs and linguistic expressions in neural models, and to disentangle the context-independent parts in the meaning representation.

In this section, we first review prior works that rely on sequence-to-sequence models. Within this category, we distinguish between approaches that use separate encoders to capture syntax and semantics, leverage lexicons, propose novel pretraining objectives, or use intermediate representations. Finally, in the remaining of this section, we discuss approaches that do not fall under the seq2seq paradigm.

3.2.1 Sequence-to-sequence Approaches

Separating the Learning of Syntax and Semantics

Inspired by work in neuroscience and cognitive sciences that suggests the existence of separate systems to process various language functions, there is a line of work that mimics this separation in neural networks via different encoders (Russin et al., 2019; Li et al., 2019). The key proposal in these works is that by separately encoding syntactic and semantic parts of the source text, language models can be equipped with better compositional understanding of language. Specifically, Russin et al. (2019) propose a mechanism called Syntactic Attention to separately model the processing of syntax and semantics. One encoder is responsible for finding the right alignment (loosely related to syntax), whereas the other is used to select the relevant output tokens (i.e., capturing the semantics). This separation discourages the model to entangle semantic information like word usage from syntactic information like constructing sentences in grammatically valid ways.

A similar approach focusing on the primitive splits of the SCAN task is CGPS (Li et al.,2019). In this work, the authors use one encoder to learn attention maps over the input sequence, while another encoder is used to align the attended input words to output words. Additionally, they also add entropy regularization to limit the representational capacity of each encoder and thus discourage the entanglement of information.

Both Syntactic Attention and CGPS achieve impressive results on the random, and primitive splits of SCAN, but they fall short on the length split of SCAN. Additionally, experiments show that these methods can barely perform any of the generalization tasks in the CFQ benchmark Furrer et al. (2020).

Semantic Labeling

Zheng & Lapata (2020) develop a sequence-to-sequence approach with a novel two-stage decoder that resembles the lexicon-style alignment and disentangled information processing from grammar-based techniques. Specifically, in the first stage, they label the input sequence with semantic symbols that represent the meaning of the individual words. Then in the second stage, they use standard sequence-to-sequence models such as an LSTM or a Transformer to predict the final meaning representation conditioned on the utterance and the predicted sequence of symbols.

Lexicon Learning

Akyürek et al. (2021) study compositional generalization in scenarios where we only have access to small datasets. They argue that often compositionality is difficult because neural models entangle lexical phenomena from syntax. To address this, similarly to the copy mechanism in seq2seq models, they augment standard sequential decoders with a special token that performs a lexical lookup to retrieve a corresponding output token in the logical form given an input token. This mechanism allows standard seq2seq models to incorporate lexicons when generating the logical form. Using various strategies to automatically learn lexicons, the authors demonstrate state-of-the-art performance on COGS.

Manual schema linking

A different approach to utilize lexicons is to map input references to components of a given environment, such as a database. As demonstrated by Wang et al. (2020) and Scholak et al. (2020), these types of lexicons can be very useful in Transformer-based semantic parsing models (e.g. SQLova as discussed in section 2.3.2), where the input sentence is jointly encoded with the database schema. In particular, as first shown by Wang et al. (2020), lexicons can be used to map input references to table and column names, or to values in a database - through a mechanism they call schema and value-based linking. The Transformer architecture can then be extended with *relation-aware self attention* to allow models to attend to these predefined relations between the input and environment. Effectively, this encourages models to not only soft-search for relational

structures between the input and the environment, but to inject an explicit bias towards these known relations. Wang et al. (2020) show impressive results on Spider, a dataset with question-SQL pairs that measures models' ability to generalize to new databases that models did not encounter during training. While Spider can effectively measure performance on question-SQL pairs that follow novel templates or concern unseen domains, most examples contains exact lexical matches between the input words and database components. However, real-world natural language questions about databases are hardly characterized by exact lexical references. This consequently makes manual schema linking less adoptable to more realistic applications (Lee et al., 2021). As Lee et al. (2021) point out a database documentation can be helpful to still make use of the linking mechanism proposed in Wang et al. (2020).

Pretraining

Although pretrained language models have led to great advances in semantic parsing, they often make mistakes when it comes to reasoning, recognizing and resolving domain-specific references against their respective environments like databases. To fill this gap, several recent works have tried to build more general-purpose representations for semantic parsing tasks via pretraining (Yin et al., 2020; Herzig et al., 2020; Deng et al., 2020). A common theme in these works is to use a parallel text-table corpus together with a set of pretraining objectives to better ground language in the respective database's schema. Grappa Yu et al. (2021) induces a synchronous context-free grammar (SCFG) from commonly used question-SQL templates from existing datasets to create a synthetic dataset. They then introduce a pretraining objective where the model has to predict whether a column appears in a given SQL query, and what operation it appears in, just given the question and table names. This objective encourages the model to identify relevant parts of the database based on their syntactic roles. Finally, they pretrain a model from the synthetic question-SQL pairs and respective table names using their semantic parsing objective in addition to the MLM objective. Although the semantic parsing task is often domain-specific, these papers argue that these general purpose pretrained encoders can provide good representations for tasks that require

jointly reasoning over unstructured natural language utterances and structured tables (Yin et al., 2020).

Intermediate representations

In Herzig et al. (2021), the authors posit that pretrained language models struggle with compositional generalization due to the lack of structural correspondence between the natural language input and its meaning representation. In order to fill this gap, they propose the use of intermediate representations (IRs) that have higher structural correspondence with the natural language input. Examples of such representations include omitting elements of the meaning representation that cannot easily align to the natural language, and adding structural cues like brackets around conjuncts to indicate nested scopes. Since these modified representations are lossy, in order to obtain executable meaning representations, they convert intermediate representations by either using a deterministic transformation, or by using a second seq2seq model that conditions on both the intermediate representation and the original natural language input. Herzig et al. (2021) show that such IRs can lead to impressive gains in compositional generalization benchmarks like CFQ and SCAN, as well as in text-to-SQL tasks.

The use of intermediate representations is closely related to the coarse-to-fine method of Dong & Lapata (2018) and IRNET from Guo et al. (2019). In the former, the authors propose a two-stage method, where given the input sentence they first generate a coarse structure, i.e., a sketch for its meaning — excluding low-level information like variable names and arguments — and then by conditioning on the input sentence and the sketch, they refine this by generating the missing details. In IRNET (Guo et al., 2019), semantic parsing is decomposed into three phases. First, they perform schema linking over a question and a database schema. Then they use a grammar-based neural model to predict an intermediate representation. Finally, they use a deterministic transformation to infer the final meaning representation. Herzig et al. (2021)'s approach is different from these works as they study IRs in the context of pretrained models, specifically focusing on improving compositional generalization.

Similar to the work of Guo et al. (2019), Furrer et al. (2020) propose to simplify meaning representations by applying a lossless compression to CFQ's SPARQL queries as a preprocessing step. For example, the question *"Did M0 and M1 direct M2 and M3?"* (where `M0, M1, M2` and `M3` refer to anonymized entities in the question) translates to the following triples in the respective SPARQL query:

```
M0 directed M2 .  M1 directed M2 .
M0 directed M3 .  M1 directed M3 .
```

Instead, the query can be compressed by merging triples if they share the same subject (e.g. `M0`), object (e.g. `M2` or `M3`) or predicate (e.g. `directed`):

```
{M0, M1} directed {M2, M3}.
```

This simplification allows SPARQL queries to better align to the question, thus improving the performance of neural semantic parsers. A similar representation is also explored by another work in Guo et al. (2020). While the authors report even more impressive gains than Furrer et al. (2020), it is unclear how their approach is different, and how one can reproduce their results [1].

3.2.2 Other Approaches

Sequence-to-Partially-Ordered-Sets

Inspired by separately encoding syntax and semantics, Guo et al. (2020) argue that semantics should be modeled to capture the partial permutation invariance in word order in logical forms. For example, the logical form of the sentence "Who influences Maxim Gorky and marries Siri von Essen" should be "$\exists \mathbf{x}$: INFLUENCE($\mathbf{x}$, *Maxim Gorky*) $\wedge$ MARRY($\mathbf{x}$, *Siri von Essen*)" where the logical form is equivalent irrespective of the order of the two predicates `INFLUENCE` and `MARRY`. Their goal is to prevent decoders from being biased to unnecessary information about word order among permutation invariant components in logical forms.

[1] In particular, it is unclear whether Guo et al. (2020) report results using the best run of their approach, or whether results are consistent across many runs.

To achieve this, they convert each logical form, as a directed acyclic graph (DAG) that represents a partially-ordered set (poset). They then propose a hierarchical mechanism to decode posets by predicting the topological traversal paths of a poset. To do so, similarly to Dong & Lapata (2018) and Guo et al. (2019), they first predict a sketch for the target poset, where the poset is represented by a directed acyclic graph (DAG). This sketch contains abstract placeholders for variables, entities, predicates. Second, they predict a set of candidate primitives to be used in the sketch. Last, they enumerate all possible posets by combining the sketch with all available primitives, and predict the most probable target poset. This poset can then be serialized into the final logical form. While this approach still uses a left-to-right decoder to predict the target DAGs, Guo et al. (2020) report impressive results, achieving state-of-the-art performance in the CFQ benchmark.

Sequence-to-Tree

Herzig & Berant (2021) propose a span-based parser for better compositional generalization. Their model, SpanBasedSP, first enumerates all possible spans, where each span is tasked to independently predict a partial program. The possible partial programs are to predict *a category*, e.g., entities and predicates from an underlying knowledge-base like a database, or *a composition category*, operations that combine categories, such as a `JOIN` clause in SQL, or *a null category*, where no program is generated. Because predictions are done independently, their training process is fast and simple. Herzig & Berant (2021) construct a tree over the input sequence using the predicted partial programs, where the tree can deterministically be serialized into the output program (i.e., the final logical form). To do this, they treat the span trees as a latent variable, use the hard-EM algorithm (Bishop, 2006). First, they find the most probable and semantically valid span tree under the current model using CKY, a grammar-based dynamic programming approach (John & Schwartz Jacob, 1970), then using the most probable tree, they update the current model. The authors also study the more practical, weakly-supervised setting in which one does not have access to gold trees to perform standard supervised semantic parsing, but only has the output programs. In this setting, the correct span tree is treated as a discrete latent variable, and is inferred using the hard EM approach. In order to help training with weak supervision, they also rely on a lexicon to

map input utterances to categories where we know the language description of constant or entities. Similar work by Yin et al. (2021) focuses on loss-based regularization of attention for span-level alignments between programs and utterances.

Chapter 4

LAGr

As discussed in section 3.2, compositionality has been a popular topic in recent research on semantic parsing. We reviewed approaches that rely on separately encoding syntax and semantics, lexicons, pretraining and intermediate representations. A unifying theme across these recent semantic parsers is sequence-to-sequence learning (seq2seq, Sutskever et al., 2014; Bahdanau et al., 2015). As we saw in section 2.3.2, a seq2seq neural semantic parser sequentially emits the serialized meaning representation token-by-token. This approach has two potential drawbacks in the context of semantic parsing. The first drawback is that seq2seq requires one to serialize the meaning representation (MR) that could often be more parsimoniously represented as a graph or a tree. The arbitrary serialization choices can adversely impact generalization. The second drawback is the autoregressive nature of seq2seq decoders. In particular, the sequential decoders learn to predict the next token based on all other previous tokens, whereas in semantic parsing many aspects of meaning can be predicted independently, which can facilitate generalization.

In section 3.2, we also discussed other approaches where instead of outputting sequential MRs, semantic parses are generated as span-trees or partially-ordered sets. While these approaches can produce impressive gains for compositional generalization, they are both overly complex, require workarounds around non-trivial corner cases like tenary rules (Herzig & Berant, 2021) or require hierarchical decoding mechanisms and the use of lexicons (Guo et al., 2020).

In this work, we take a simpler approach, and propose a sequence-to-graph decoder where semantic parses are produced directly as graphs, instead of sequences. Specifically, we introduce LAGr (Labeling Aligned Graphs), an approach to generate meaning representations graphs (MR graphs) by labeling nodes and edges of a fully-connected multi-layer output graph that is aligned with the input utterance. Our approach is inspired by Lyu & Titov (2018) where a similar method is used to construct Abstract Meaning Representation (AMR) graphs. However, to the best of our knowledge such a sequence-to-graph method has not been considered in compositional generalization research. Importantly, LAGr retains most of the flexibility of seq2seq models, while allowing for parallel computation that reduces the number of computation steps to a constant. Furthermore, LAGr demonstrates significant gains for compositional generalization, without the complexity and rigidity that comes with the grammar-based approaches.

In this chapter, we first describe how LAGr produces MR graphs by labeling nodes and edges of a fully-connected output graph. Then we discuss two training regimes for LAGr. First, in section 4.1.1 we consider the strongly-supervised version of LAGr in which training examples contain MR graphs that are aligned to input sequences. Then in section 4.1.2, we describe how LAGr can be trained in the weakly-supervised setting, where the alignment between MR graphs and the input sequence is latent. Lastly, in section 4.2, we compare LAGr to other relevant literature.

4.1 Model Description

We present LAGr (**L**abel **A**ligned **G**raphs), a framework for constructing meaning representations directly as graphs (so-called *MR graphs*). When LAGr is used to output logical forms, the graph nodes can be variables, entities, categories and predicates, and graph edges can be the Neo-Davidsonian style semantic role relations that the nodes appear in, e.g. *"is-agent-of"* or *"is-theme-of"* (Parsons, 1990). While this work focuses on predicting logical forms, LAGr can, in principle, also be used to output other kinds of graphs, such as abstract syntax tree parses of SQL queries. As illustrated in Figure 4.1, LAGr predicts the output by labeling the nodes and edges of a fully-connected multi-layer output graph that is aligned with the input utterance. We label a

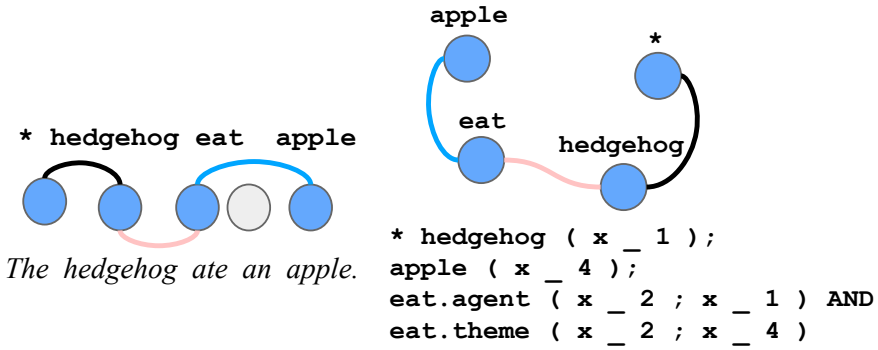
multi-layer as opposed to a single-layer graph because some MR graphs have more nodes than the number of input tokens. Using multiple graph layers also allows models to predict multiple nodes, one per each layer, where appropriate. For example, the sentence "*Which female French costume designer influenced M1 and M2*", where for the input token *French*, intuitively, we may want to align two output nodes: the `nationality` predicate and the `French` literal. Since the CFQ benchmark (Keysers et al., 2019) contains many such examples, we elaborate on this point further in Section 5.2.

Notation and Terminology Formally, let $x = x_1, x_2, ..., x_N$ denote a natural language utterance of N tokens. LAGr produces an MR graph G by labeling the nodes and edges of a complete graph Γ_a with $M = L \cdot N$ nodes that are arranged in L layers. The layers are aligned with the input sequence x in a way that for each input position i there is a unique corresponding output node in each layer. We say that nodes from different layers that are aligned with the position i form a column (an example column in Figure 4.1b contains the nodes labeled as `actor` and `?x0` for the word *star* at the position $i = 3$).

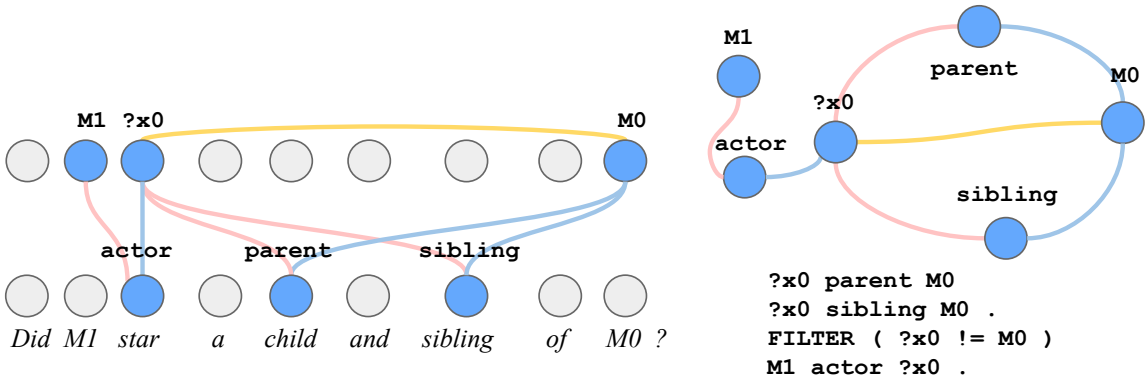
We write $\Gamma_a = (z, \xi)$ to indicate that a complete labeled graph Γ_a is characterized by its node labels $z \in V_n^M$ and edge labels $\xi \in V_e^{M \times M}$, where V_n and V_e are node and edge label vocabularies, respectively. Both vocabularies also include additional `null` labels that we use as padding. While we omit `null` edges in Figure 4.1, we show `null` nodes in grey. To produce the output MR graph G from Γ_a, we remove all `null` nodes and `null` edges. Lastly, we use z_j and ξ_{jk} notations to refer to the labels of node j and of the edge (j, k) where $j = (l-1)N + i$ is a one-dimensional index that corresponds to the i-th node in the l-th layer.

4.1.1 Labeling Aligned Graphs

To label the nodes of Γ_a we encode the input utterance x as a matrix of N d-dimensional vectors $H = f_{enc}(x) \in \mathbb{R}^{N \times d}$, where f_{enc} can be an arbitrary encoder model such as LSTM (Hochreiter & Schmidhuber, 1997) or a Transformer (Vaswani et al., 2017). LAGr then defines a factorized



(a) COGS



(b) CFQ

Figure 4.1: Aligned and unaligned graphs for COGS (a) and CFQ (b). For COGS, pink, blue and grey denote `agent`, `theme` and **`article`** edges, respectively. For CFQ, yellow, pink and blue mark **`FILTER`**, `agent`, `theme` edges. Grey nodes mark `null` nodes, and $\star$ denotes the definite article. The aligned graphs here are only provided for illustration purposes, and were not used for training. For the learned aligned graphs see Section 5.

distribution $p(z|x)$ over the node labels z as follows:

$$O = \mathop{||}_{l=1}^{L} HW^l, \tag{4.1}$$

$$\pi = \texttt{softmax}(O), \tag{4.2}$$

$$p(z|x) = \prod_{j=1}^{M} p(z_j|x) = \pi_{j,z_j}, \tag{4.3}$$

where $O \in \mathbb{R}^{M \times |V_n|}$ contains logits for $M = N \times L$ nodes from all the L graph layers, $||$ denotes the concatenation operation along the node axis, W_l denotes the weight matrix for layer l. Here and in following equations $\texttt{softmax}(.)$ is applied to the last dimension of the input tensor and every multiplication by a weight matrix is followed by the addition of a bias vector which we omit to enhance clarity. Our edge labelling computation is reminiscent of the multi-head self-attention by Vaswani et al. (2017) explained in Section 2.3.2. The key difference here is that we apply softmax across the edge labels and not across positions:

$$H_q^\alpha = \mathop{||}_{l=1}^{L} HU^{\alpha,l}, \tag{4.4}$$

$$H_k^\alpha = \mathop{||}_{l=1}^{L} HV^{\alpha,l}, \tag{4.5}$$

$$\rho = \texttt{softmax}\left[\mathop{\texttt{stack}}_{\alpha \in V_e}\left[H_q^\alpha H_k^{\alpha T}\right]\right], \tag{4.6}$$

where H_q^α and H_k^α contain concatenated key and query vectors for the label $\alpha \in V_e$ across all L graph layers, $U^{\alpha,l}, V^{\alpha,l} \in \mathbb{R}^{\frac{d}{|V_e|}, \frac{d}{|V_e|}}$ are two weight matrices for the edge label α, and the $\texttt{stack}$ operator stacks the matrices into a 3D tensor to which softmax is subsequently applied in equation 4.6. Similarly to $p(z|x)$, we obtain $p(\xi|x)$ as follows:

$$p(\xi|x) = \prod_{j=1}^{M}\prod_{k=1}^{M} p(\xi_{jk}|x) = \prod_{j=1}^{M}\prod_{k=1}^{M} \rho_{jk\xi_{jk}}. \tag{4.7}$$

The factorized nature of Equations 4.3 and 4.7 makes the argmax inference $\hat{z}, \hat{\xi} = \arg\max p(z, \xi|x)$ trivial to perform. When the groundtruth aligned graph $\Gamma_a^* = (z^*, \xi^*)$ for the MR graph G is

available, LAGr can be trained by directly optimizing $\log p(z = z^*, \xi = \xi^* | x)$. We refer to this training setting as *strongly-supervised LAGr*.

4.1.2 The Latent Alignment Model

In many practical settings, the alignment between the MR graph G and the question x is unavailable, making the aligned graph Γ_{a} unknown. To address this common scenario, we propose a *weakly-supervised LAGr* algorithm based on a latent alignment model. Similarly to the strongly-supervised case, we assume that the MR graph can be represented as a labeled complete, multi-layer graph $\Gamma_{\text{na}} = (s \in V_n^M, e \in V_e^{M \times M})$, with the difference that in this case the alignment between x and Γ_{na} is not known. We assume a generative process whereby Γ_{na} is obtained by permuting the columns of the latent aligned graph Γ_{a} with a random permutation a, where a_j is the number of the column in Γ_{a} that becomes the j-th column in Γ_{na}. For the rest of this section we focus on the single layer ($L = 1$) case to simplify the formulas. For this case our probabilistic model defines the following distribution over $\Gamma_{\text{na}} = (s, e)$:

$$p(e, s|x) = \sum_a \sum_z \sum_\xi p(e, s, a, z, \xi | x) \tag{4.8}$$

$$= \sum_a p(a) \prod_j p(z_{a_j} = s_j | x) \prod_j \prod_k p(\xi_{a_j a_k} = e_{jk} | x), \tag{4.9}$$

where $p(a) = 1/N!$. Computing $p(e, s|x)$ exactly is intractable. For this reason, we train LAGr by using an approximation of $p(e, s|x)$ in which instead of summing over all possible aligments a we only consider the Maximum A Posteriori (MAP) alignment $\hat{a} = \arg\max_a p(a|e, s, x)$. This approach is sometimes called the hard Expectation-Maximization algorithm in the literature on probabilistic models (Bishop, 2006). The training objective thus becomes

$$p(e, s|\hat{a}, x) = \prod_j p(z_{\hat{a}_j} = s_j | x) \prod_j \prod_k p(\xi_{\hat{a}_j, \hat{a}_k} = e_{jk} | x). \tag{4.10}$$

To infer the MAP alignment $\hat{a}$, we need to solve the following inference problem:

$$\begin{aligned}\hat{a} &= \arg\max_a p(a|e,s,x) \\ &= \arg\max_a \log p(s|a,x) + \log p(e|a,x) \\ &= \arg\max_a \Big[\sum_j \log p(z_{a_j} = s_j|x) + \sum_j \sum_k \log p(\xi_{a_j,a_k} = e_{j,k}|x) \Big]\end{aligned} \tag{4.11}$$

We are not aware of an exact algorithm for solving the above optimization problem, however if the edge log-likelihood term $\log p(e|a,x)$ is dropped in the equations above, maximizing the node label probability $p(s|a,x)$ is equivalent to a standard minimum cost bipartite matching problem. This optimization problem can be solved by a polynomial-time Hungarian algorithm (Kuhn, 1955). We can thus use an approximate MAP alignment $\hat{a}^1 = \arg\max_a \sum_j \log p(z_{a_j} = s_j|x)$. While dropping $p(e|a,x)$ from Equation 4.11 is a drastic simplification, in situations where node labels s are unique and the model is sufficiently trained to output sharp probabilities $p(z_j|x)$ we expect $\hat{a}_1$ to often match $\hat{a}$. To further improve the MAP alignment approximation and alleviate the reliance on the node label uniqueness, we generate a shortlist of K candidate alignments by solving K noisy matching problems of the form $\arg\max_a \sum_j \log p(z_{a_j} = s_j|x) + \epsilon_{ja_j}$, where $\epsilon_{ja_j} \sim N(0,\sigma)$. We then select the alignment candidate a that yields the highest full log-likelihood $\log p(s|a,x) + \log p(e|a,x)$. We summarize the above procedure of training LAGr with weak supervision in Algorithm 1.

Algorithm 1: Training LAGr with weak supervision.

Init: Let K be the number of alignment candidates, T be the number of training steps, θ_t be the model parameters after t steps.

for *t=1, ..., T* **do**
  Sample example (x, e, s).
  for κ*=1, ..., K* **do**
    $\epsilon_{ji} \sim N(0,\sigma)$
    $cost_{ji} = -\log p(z_i = s_j|x) + \epsilon_{ji}$
    $a^\kappa = \texttt{MinCostBipartiteMatching}(cost)$
    $J^\kappa = \sum_j \log p(z_{a^\kappa_j} = s_j|x) + \sum_j \sum_k \log p(\xi_{a^\kappa_j a^\kappa_k} = e_{jk}|x)$
  $\hat{\kappa} = \arg\max_\kappa J^\kappa$
  $\theta_{t+1} \leftarrow \texttt{Optimizer}(\theta_t, \nabla_\theta - J^{\hat{\kappa}})$
return θ_{T+1}

4.2 Related Work

The LAGr approach is heavily inspired by graph-based dependency parsing algorithms (Mcdonald, 2006). In neural graph-based dependency parsers (Kiperwasser & Goldberg, 2016; Dozat & Manning, 2017) the model is trained to predict the existence and the label of each of the possible edges between the input words. The Abstract Meaning Representation (AMR) parser by Lyu & Titov (2018) brings similar methodology to the realm of semantic parsing, although they do not consider the compositional generalization implications of using a graph-based parser instead of a seq2seq one. Lyu & Titov (2018) only output single layer graphs which requires aggresive graph compression; in LAGr we allow the model to output a multiple layer graph instead. Lastly, the amortized Gumbel-Sinkhorn alignment inference used by Lyu & Titov (2018) is much more complex than the Hungarian-algorithm-based approximate MAP inference that we employ here. Another important inspiration for LAGr is the UDepLambda method (Reddy et al., 2016) for converting dependency parses into graph-like logical forms. LAGr can be seen as an algorithm that produces the UDepLambda graphs directly with the neural model, side-stepping the intermediate dependency parsing step.

As seen in section 3.2.2, another alternative to seq2seq semantic parsers are span-based parsers that predict span-level actions for building MR expressions from sub-expressions (Herzig & Berant, 2021; Pasupat et al., 2019). A prerequisite for using a span-based parser is an MR that can be viewed as a recursive composition of MRs for subspans. While this strong compositionality assumption holds for the logical forms used in earlier semantic parsing research (e.g. Zettlemoyer & Collins (2005)), a span-based parser would require intermediate MR in order to produce other meaning representations, such as SPARQL or SQL queries. The designer for an intermediate MR for a span-based parser must think about MRs for spans and how they should be composed. This can sometimes lead to non-trivial corner cases, such as ternary grammar rules, as shown in Herzig & Berant (2021). On the contrary, a graph-based parser can in principle produce any graph.

Other related semantic parsing approaches include the semantic labeling approach by Zheng & Lapata (2020) and the structured reordering approach by Wang et al. (2021). As discussed

in section 3.2.1, Zheng & Lapata (2020) show that labelling the input sequence prior to feeding it to the seq2seq semantic parser improves compositional generalization. In contrast, our work goes one step further by adding edge labeling, which allows us to remove the sequential decoder entirely. Wang et al. (2021) model semantic parsing as structured permutation of the input sequence followed by monotonic segment-level transduction. This approach achieves impressive results, but is considerably more complex than LAGr. Meta-learning is also studied as a way to improve compositionality in neural semantic parsers (Conklin et al., 2021). Specifically, Conklin et al. (2021) propose the Tree-MAML algorithm for a meta-learning variation of the COGS tasks, in which they construct pairs of tasks set up in a way that allows to directly optimize for the performance of out-of-distribution examples. Since meta-learning requires a different task formulation for COGS, this approach is not directly comparable to LAGr.

Concurrently with this work, Ontañón et al. (2021) show that semantic parsing by sequence tagging improves compositional generalization on the CFQ benchmark. Their sequence tags are similar to 1-layer aligned graphs that we predict here. However, Ontañón et al. (2021) do not discuss how to infer sequence tags from logical forms when the former are not available.

Finally, the hierarchical poset decoding approach as described in 3.2.2 by Guo et al. (2020) achieves impressive performance on CFQ by combining the sketch prediction approach from Dong & Lapata (2018) with a poset decoding algorithm that outputs meaning representatins as directed acyclic graphs (DAGs). Unlike LAGr that predicts graphs in parallel, this algorithm produces DAGs in a sequential left-to-right fashion. We also note that without the sketch prediction stage, their poset decoding algorithm alone performs poorly.

Chapter 5

Experiments

In this chapter, we demonstrate the effectiveness of LAGr using two compositional generalization benchmarks: *Compositional Generalization for Semantic Parsing* (COGS) (Kim & Linzen, 2020) and *Compositional Freebase Questions* (CFQ) (Keysers et al., 2019). COGS serves as a convenient testbed to study both variants of LAGr. While aligned graphs are not immediately available, they can be easily reconstructed, and thus be used for training strongly-supervised LAGr. On the other hand, reconstructing alignments between the MR graphs and the input is prohibitively difficult for CFQ. For this reason, we use CFQ to measure the performance of LAGr in the weakly-supervised setting, where alignment is treated as a latent variable.

This chapter is organized as follows. Sections 5.1 and 5.2 focus on our experiments using the COGS and CFQ benchmarks, respectively. In each section, we start with a brief description of the benchmark dataset, followed by an explanation of how MR graphs can be constructed from the dataset's serialized meaning representations. We then describe our training procedure and hyperparameters and present our baselines. We conclude each section with a presentation of our results. Finally, Section 5.3 provides a further discussion on some commonly encountered errors and zooms in on the learned alignments from the weakly-supervised LAGr experiments.

In order to reproduce the experiments we discuss in this section, we refer the reader to our source code published under `https://github.com/ElementAI/lagr`.

5.1 Strongly- and weakly-supervised LAGr on COGS

5.1.1 Dataset

As we saw in chapter 3, COGS (Kim & Linzen, 2020) is a semantic parsing benchmark specifically designed to assess compositional generalization. It consists of pairs of English sentences and their respective lambda calculus logical forms, with examples we saw earlier such as:

"The hedgehog ate an apple."

$$\star\texttt{hedgehog}(x_1)\ ;\ \texttt{apple}(x_4);\ \texttt{eat.agent}(x_2,\ x_1)\ \texttt{AND}\ \texttt{eat.theme}(x_2,\ x_4))$$

where `agent` and `theme` mark the Neo-Davidsonian semantic roles between variables and entities — concepts we introduced in our earlier discussion in Section 2.2. As we also discussed previously, the out-of-distribution generalization set of COGS features novel combinations of words and syntactic structures from the training dataset. For example, while the word *hedgehog* is only observed as a subject (i.e., as the `agent` of the predicate) during training (e.g. *"the hedgehog sleeps"*), the generalization set requires language models to understand it in the object (i.e., `theme`) role (e.g. *"the hero painted the hedgehog"*). For a more extensive review on COGS, we refer the reader to Chapter 3.

5.1.2 Graph Construction

In order to study LAGr on COGS, we first convert the logical forms to UDepLambda-style (Reddy et al., 2016) MR graphs. First, we create graph nodes for the one- and two-place predicates and definite articles. Using the earlier example, Figure 5.1 shows below that this process yields the `hedgehog`, `apple`, `eat`, and the $\star$ nodes.

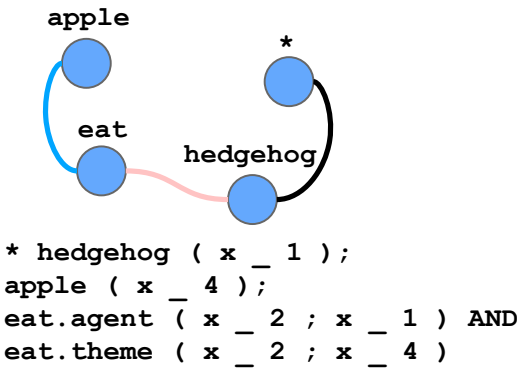


Figure 5.1: Graph construction from lambda calculus meaning representations for COGS.

We do not create dedicated nodes for variables, as every variable in COGS is either an argument to a unique one-place predicate (e.g. x_1 is for `hedgehog`(x_1)), or the first argument to a unique two-place predicate (e.g. x_2 for `eat` in `eat.agent`(x_2, x_1)). Instead, we let the respective predicate node represent the variable. To define the graph's labeled edges, we use the Neo-Davidsonian role predicates (i.e., `agent, theme, recipient, ccomp, nmod.on, nmod.in, xcomp, nmod.beside`). Consequently, as Figure 5.1 illustrates, the `eat.agent`(x_2, x_1) conjunct results in an `agent` edge between the `eat` and `hedgehog` nodes. We also add special article edges to connect definite article nodes (denoted by the $\star$ label) to their respective nouns (i.e., the `hedgehog` node in Figure 5.1).

We take advantage of the correspondence between variable names and input positions (e.g. variable x_i always corresponds to the i-th input token). This allows us to construct single-layer ($L = 1$) aligned graphs Γ_a for COGS that are suitable for training strongly-supervised LAGr, as described in Section 4.1.1. The node and edge vocabularies for the aligned graphs contain 645 and 10 labels respectively, each including a `null` label.

5.1.3 Training Details

Hyperparameter tuning on COGS is challenging since the performance on the in-distribution development set always saturates near 100%. This makes finding the best configuration extremely difficult and not reproducible. We adopt the hyperparameter tuning procedure discussed in Con-

klin et al. (2021) to find the best configuration for our baselines and strongly-supervised LAGr models. Specifically, we create a "Gen Dev" dataset by sampling 1000 random examples from the generalization set and use them to find the best hyperparameter configuration. Each configuration is then evaluated on 5 seeds. We report the best configurations for COGS in Table 5.1. To assess performance, we report the exact match accuracy, i.e., the percentage of examples for which the predicted graphs after serialization yielded the same logical form. We tune the hyperparameters for strongly-supervised LAGr first; we then use the same configuration for weakly-supervised LAGr and only tune the inference hyperparameters, i.e. the number of candidates K and the noise level σ. Once the best configuration is found, we retrain all models on at least 10 seeds. Specifically, the final number of seeds that are used to report our results in Table 5.2 are the following: 20 seeds for each of the weakly-supervised LAGr experiments with and without retraining, 80 and 20 seeds for strongly-supervised LAGr with a separate and shared encoder, respectively, and finally, 20 seeds for our baseline Transformer experiments. We vary the number of seeds in order to obtain more accurate estimates for the mean performance measures.

We find that both our Transformer-based seq2seq and LAGr models perform better when embeddings are initialized following He et al. (2015) and when positional embeddings are scaled down by $\frac{1}{\sqrt{dim}}$. We adopt latter techniques following the recent work of Csordás et al. (2021) under the PED (Positional Embedding Downscaling) name.

For strongly-supervised LAGr, we test 0.001, 0.004, 0.0001 and 0.0004 for learning rates, 64, 128 and 256 for batch sizes, and 0.1 versus 0.4 for dropout. We test an embedding size of 256 versus 512. Furthermore, we also experiment with 2 versus 4 Transformer layers, and 4 versus 8 attention heads. For weakly-supervised LAGr, we use the best configuration we find for strongly-supervised LAGr. We then investigate different values for K, the number of candidate alignments, with 1, 5 versus 10, and for the noise levels σ of 0, 0.001, 0.01, 0.1, 1, 10, 15 and 20. Since weakly-supervised LAGr does not always converge on the training set, we implement a restart mechanism that relaunches experiments with a new random seed where a training performance of at least 95% is not achieved. Setting $K = 10$ and $\sigma = 1.0$ allows us to achieve a convergence rate of around 50%.

Additionally, we observe that the training loss does not go to 0 in the weakly-supervised setting. We attribute this to a significant (2.7%) percentage of training examples in which there are three and more nodes with the same label (namely "*" for definite articles), which presents a challenge to our alignment inference mechanism. To remedy this, we cache and append the previously used alignment as the $K+1$st alignment candidate (see lines 3-8 in Algorithm 1). This allows the model to remember low-loss alignments and thereby helps achieve full convergence. Lastly, we also run weakly-supervised LAGr with retraining, in which we take the final learned alignments for all examples and retrain models with the learned alignments being used as strong supervision.

As for our seq2seq baseline, in order to reproduce the same Transformer performance as reported by Csordás et al. (2021), we reuse both their hyperparameters and their model implementation. Namely, we use a learning rate of 1e-4 with a linear scheduler and no warmup, a batch size of 128, an encoder dimension of 512 with dropout of 0.1. Lastly, we clip gradients larger than 1.0.

	Reproduced baselines		**Strongly-supervised LAGr with different encoders**			
.	LSTM	Transformer	LSTM$_{sh}$	LSTM$_{sep}$	Transformer$_{sh}$	Transformer$_{sep}$
batch_size	256	128	128	64	128	128
learning_rate	0.004	0.0001	0.0001	0.0004	0.0001	0.0001
scheduler	linear with warmup of 1000 steps	linear with no warmup	linear with warmup of 1000 steps	linear with warmup of 1000 steps	linear with no warmup	linear with no warmup
layers	2	4	2	2	4	4
enc_dim	256	256	256	256	512	512
train_steps	50000	50000	70000	70000	70000	70000
validate_every *(step)*	5000	5000	5000	5000	10000	10000
dropout	0.4	0.1	0.1	0.4	0.4	0.4
attention heads	-	8	-	-	4	4

Table 5.1: Best hyperparameters for our baselines and strongly-supervised LAGr experiments on COGS.

5.1.4 Baselines

We compare LAGr to LSTM- and Transformer- based seq2seq semantic parsers that produce the COGS logical forms as sequences of tokens. In addition to training our own seq2seq baselines, we include the baseline results from the original COGS paper by Kim & Linzen (2020) and from follow-up works by Akyürek et al. (2021), and Conklin et al. (2021). The baseline seq2seq results for COGS vary highly from one implementation to another, hence we include results from several

	Exact match accuracy (%)		
	train	test	gen
Tree-MAML LSTM ♠	-	99.7	41.0 (±4.9)
Tree-MAML Transformer ♠	-	99.6	66.7 (±4.4)
LSTM+Attn ♢	-	99.	16. (±8.)
Transformer ♢	-	96.	35. (±6.)
LSTM+Attn ♡	-	-	51. (±5.)
Transformer ♣	-	-	**81.** (±1.)
LSTM + Lex: Simple ♡	-	-	**82.** (±1.)
LSTM + Lex: PMI ♡	-	-	**82.** (±0.)
LSTM + Lex: IBMM2 ♡	-	-	**82.** (±0.)
LSTM+Attn (ours)	100 (±0.0)	99.6 (±0.2)	26.1 (±6.8)
LSTM_{sh} strongly-supervised LAGr	100 (±0.0)	99.9 (±0.1)	39.0 (±9.1)
LSTM_{sep} strongly-supervised LAGr	100 (±0.0)	100 (±0.0)	71.4 (±2.9)
Transformer (ours)	100 (±0.0)	99.8 (±0.0)	80.6 (±1.4)
Transformer_{sh} strongly-supervised LAGr	100 (±0.0)	100 (±0.0)	80.2 (±1.4)
Transformer_{sep} strongly-supervised LAGr	100 (±0.0)	99.9 (±0.1)	**82.5** (±2.9)
Transformer_{sep} weakly-supervised LAGr	100 (±0.0)	99.9 (±0.0)	80.7 (±2.5)
Transformer_{sep} weakly-supervised LAGr + Retrain	100 (±0.0)	99.9 (±0.0)	**82.3** (±2.3)

Table 5.2: Average exact match accuracy and standard deviation on COGS. **Bottom**: reproduced seq2seq baselines and LAGr over 10 runs. **Middle:** Seq2seq baselines including the original results by Kim & Linzen (2020) ♢, best Transformer baseline by Csordás et al. (2021) ♣, and the best LSTM baseline by Akyürek et al. (2021) ♡. We also show a lexicon-based approach by Akyürek et al. (2021). **Top**: For reference, we also include results from Tree-MAML by Conklin et al. (2021) ♠.

studies for a more rigorous assessment. We also compare LAGr to the lexicon-based seq2seq model "LSTM+Lex" by Akyürek et al. (2021) from section 3.2.1. Lastly, we include results from the Tree-MAML algorithm (Conklin et al., 2021) that uses meta-learning to improve compositional generalization.

5.1.5 Results

Table 5.2 shows that our best Transformers trained with LAGr outperform the original (35% from Kim & Linzen (2020) and 81% from Csordás et al. (2021)) and our reproduced (80.6%) seq2seq Transformer baselines, obtaining 82.5% and 82.3% exact match accuracy in the strongly- and weakly-supervised settings, respectively.

We experiment with two variations of LAGr: using shared encoders and separating encoders for syntax (i.e., node predictions) and semantics (i.e., edge predictions) — reflected in Table 5.2 by the subindex "$\_sh$" versus "$\_sep$" in the model names respectively. We achieve the best result in the strongly-supervised setting using separate encoders. While this setting significantly improves the performance of LAGr in all cases, for the strongly-supervised LSTM-based LAGr models, separating encoders seems to be crucial (71.4% vs 39.0%).

Notably, we observe that the use of retraining in weakly-supervised LAGr is helpful. It allows us to increase the accuracy of weakly-supervised LAGr to match our strongly-supervised result.

Though it is difficult to directly compare LAGr with approaches that use meta-learning objectives, our best LAGr model still significantly outperforms Tree-MAML with 82.3% vs 66.7%. Finally, LAGr is able to match the performance of the LSTM+Lex approach by Akyürek et al. (2021) without relying on the use of lexicons — a result we further discuss in Section 6.1.

5.2 Weakly-supervised LAGr on CFQ

5.2.1 Dataset

As introduced in chapter 3, CFQ (Keysers et al., 2019) is another compositional generalization benchmark for semantic parsing that requires models to translate English questions to SPARQL database queries, with examples such as follows [1]:

Did M1 star a child and sibling of M0?

```
SELECT COUNT(*) WHERE {
?x0 parent M0 .  ?x0 sibling M0 .  FILTER ( ?x0 != M0 ) .  M1
actor ?x0.
```

To assess compositional generalization in our experiments, we use CFQ's *Maximum Compound Divergence* (MCD) splits. These splits were generated by making the distribution of compositional structures in the train and test sets as divergent as possible.

[1] To enhance clarity and readability, we abbreviated the original CFQ predicate names in this example

SPARQL queries contain two components: a `SELECT` and a `WHERE` clause. The `SELECT` clause is either of the form `SELECT count(*)` for yes/no questions or `SELECT DISTINCT ?x0` for wh- questions (those starting with "which", "what", "who", etc.). The `WHERE` clause can take on three forms: filter constraints ensuring two variables or entities are distinct (e.g. `FILTER ?x0 != M0`), two-place predicates expressing a relation between two entities (e.g. `?x0 parent ?x1`), and one-place predicates expressing if an entity belongs to a category (e.g. `?x0 a ns:film.actor`).

5.2.2 Graph Construction

Before constructing the graphs, similarly to prior work by (Furrer et al., 2020; Guo et al., 2020) as discussed in Section 3.2.1, we compress the SPARQL queries by merging triples in the `WHERE` clauses that have the same predicate and have either the same objects or the same subjects. As an example, consider the question "*Were M2 and M3 directed by a screenwriter that executive produced M1?*", where the original MR contains both `[M2 directed_by ?x0, M3 directed_by ?x0]` conjuncts. To make it easier to align SPARQL queries to the input question, we merge triples by concatenating their subjects and objects, yielding `[[M2, M3] directed_by ?x0]` for the above example. Consequently, the SPARQL queries can now contain an arbitrary number of entities in the triples. As Figure 5.2 shows, this procedure allows us to build more compact graphs by introducing only one instead of two predicate nodes with the `directed_by` label.

Figure 5.2: Compressed SPARQL queries for CFQ.

In order to convert the compressed SPARQL queries to graphs, we first remove the `SELECT` clauses. To preserve the question type information, for wh- questions we replace the `?x0` variable in the `WHERE` clause with a special `select_?x0` variable. For example, for the question *"Who directed Elysium?"* where the original MR includes the `SELECT DISTINCT ?x0` clause, we simplify the SPARQL query as follows:

```
[select_?x0 a ns:people.person],
[select_x0 ns:film:director.film m.0gwm_wy] ,
```

where we replaced `?x0` with `select_?x0` and only kept the modified constraints (i.e., triples) inside the `WHERE` clause.

Reusing our example from Figure 5.3 where we explained the LAGr methodology, we define the graph nodes by taking the entities (including variables, e.g. `?x0, M1`) and all predicates (`parent, sibling, actor`) from the triples.

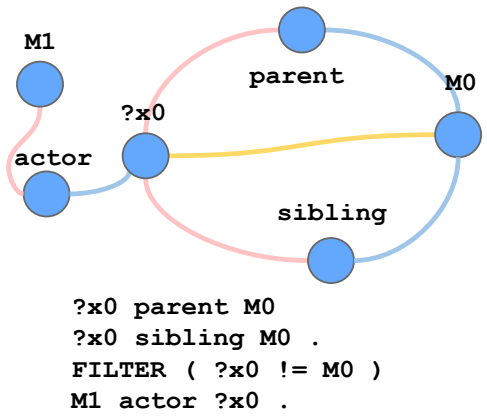


Figure 5.3: Graph construction from our preprocessed SPARQL queries for CFQ.

For one-place predicate triples, we connect the entity nodes to the predicate node with an `agent` edge label. For two-place predicates, we connect the predicate to the left-hand side and right-hand side entities with the `agent` and `theme` edge respectively. Figure 5.3 illustrates this by adding a pink and blue edge between the respective entities for the `(?x0 parent M0)`, `(?x0 sibling M0)`, and `(M1 actor ?x0)` triples. Indicated by a yellow edge in Figure 5.3), we also add a

`FILTER` edge between the variables or entities that participate in a filter constraint. The node and the edge vocabularies for our MR graphs contain 84 and 4 labels respectively, including the null labels in both cases.

5.2.3 Training Details

To better accommodate the large MR graphs of CFQ we use L=2 graph layers. Using a single layer, as done in the COGS experiments, would not be possible because of examples such as "*Who married M1's female German executive producer?*" that contains 8 tokens, but induces the following 10 nodes: `?x1, executive_produced, M1, gender, ns:m.02zsn, nationality, ns:m.0345h, select_?x0, spouse, person`. This is not only necessary for the model to be able to represent the MR graphs, it is also in line with our intuition for how the model may map input tokens to nodes (an example of a possible aligned graph is shown in Figure 4.1b).

In all our experiments on CFQ we use a shared Transformer encoder for both node and edge predictions. To assess performance, we use exact graph accuracy, which we define as the percentage of examples where the predicted and true graphs are isomorphic. The predicted graphs contain enough information to exactly reconstruct the SPARQL query. For this reason, our exact graph accuracy can be compared to exact match accuracy as used by our seq2seq baselines that measures exact matches between the predicted SPARQL queries and the groundtruth logical forms.

For hyperparameter tuning, we follow Keysers et al. (2019) and use CFQ's in-distribution *random* split to find the best model configuration. We do this by first fixing the number of candidate alignments at $K = 1$ and $\sigma = 0$ to search for the best hyperparameters, then fixing the best configuration and varying K and σ. We test learning rates of 0.0001, 0.0004, 0.0006, 0.0008 and 0.001, with a linear warmup of 0, 1000 versus 5000 steps, with dropout of 0.1 and 0.4, batch sizes of 64, 128 and 256, and 2 versus 4 Transformer layers. We also experiment with varying the number of training steps from 150000, 200000 to 750000 with appropriate batch sizes to ensure that training terminates within a day.

	CFQ	
	Weakly-supervised LAGr	
	LSTM_{sep}	Transformer_{sh}
batch_size	64	256
learning_rate	0.001	0.0004
scheduler	linear with warmup of 1000 steps	linear with warmup of 1000 steps
layers	2	4
enc_dim	512	256
train_steps	200000	750000
validate_every (step)	10000	10000
dropout	0.4	0.1
attention heads	-	8

Table 5.3: Best configuration for CFQ weakly-supervised LAGr.

For the best inference hyperparameters of $K = 5$ and $\sigma = 10$, we report the average graph accuracy and standard deviation for 8-11 runs of weakly-supervised LAGr on the out-of-distribution splits MCD1, MCD2, and MCD3 as well as on the random split. Similarly to COGS, we use the PED initialization technique from Csordás et al. (2021), and discard runs where weakly-supervised LAGr does not reach at least 99.5% graph accuracy on the training set (around 12% of all runs). In contrast to COGS, we are able to drive the training loss to 0 without caching and appending previously learned alignments as the K + 1st alignment candidates. For this reason, we do not use this caching technique. We report the best configuration used for CFQ in Table 5.3.

5.2.4 Results

We compare LAGr to seq2seq semantic parsing results reported in prior work (Keysers et al., 2019; Furrer et al., 2020), including results obtained with compressed SPARQL queries (Guo et al., 2020; Herzig et al., 2021). As shown in Table 5.4, weakly-supervised LAGr outperforms all comparable baselines on all of CFQ's out-of-distribution MCD splits. While both $K = 1$ and $K = 5$ with $\sigma = 10$ yield impressive performance gains compared to the baselines, we obtain mixed results about the impact of a higher K and the use of noise. Specifically, the best result on MCD1 is achieved with

	Graph Accuracy					
	Random		Mean MCD	MCD1	MCD2	MCD3
	train	test	test	test	test	test
HPD ♠	-	-	67.3 (∓4.1)	72.0 (∓7.5)	66.1 (∓6.4)	63.9 (∓5.7)
HPD w/o Hierarchical Mechanism ♠	-	-	-	21.3	6.4	10.1
T5-small + IR ♢	-	-	47.9	-	-	-
LSTM + Attn ♡	-	97.4 (∓0.3)	14.9 (∓1.1)	28.9 (∓1.8)	5.0 (∓0.8)	10.8 (∓0.6)
Transformer ♡	-	98.5 (∓0.2)	17.9 (∓0.9)	34.9 (∓1.1)	8.2 (∓0.3)	10.6 (∓1.1)
Universal Transformer ♡	-	98.0 (∓0.3)	18.9 (∓1.4)	37.4 (∓2.2)	8.1 (∓1.6)	11.3 (∓0.3)
Evol. Transformer ♣	-	-	20.8 (∓0.7)	42.4 (∓1.0)	9.3 (∓0.8)	10.8 (∓0.2)
LSTM + Simplified SPARQL ♠	-	-	26.1	42.2	14.5	**21.5**
Transformer + Simplified SPARQL ♠	-	-	31.4	53.0	19.5	**21.6**
T5-small from scratch ♢	-	-	20.8	-	-	-
T5-small from scratch + IR ♢	-	-	22.6	-	-	-
Transformer$_{sh}$ weakly sup. LAGr, $K = 1$	100 (∓0.0)	99.5 (∓0.2)	**38.2** (∓2.7)	**65.2** (∓2.6)	26.4 (∓3.2)	23.0 (∓2.0)
Transformer$_{sh}$ weakly sup. LAGr, $K = 5, \sigma = 10$	100 (∓0.0)	99.7 (∓0.0)	**39.5** (∓3.2)	62.8 (∓4.0)	**30.3** (∓2.7)	**25.4** (∓2.7)

Table 5.4: Average graph accuracy and standard deviation of weakly-supervised LAGr on CFQ (**bottom**). **Middle:** results by several seq2seq baselines from prior work (Keysers et al. (2019) ♡, Furrer et al. (2020) ♣). **Top:** results not directly comparable to LAGr: Hierarchical Poset Decoding Guo et al. (2020) ♠, and pretrained T5-small seq2seq model with intermediate representations (IR) Herzig et al. (2021) ♢. Approaches other than LAGr report the average exact match accuracy with 95% confidence intervals.

$K = 1$ in contrast to MCD2 and MCD3 where $K = 5$ with $\sigma = 10$ performs significantly better than when using $K = 1$.

For reference, Table 5.4 also includes the state-of-the-art Hierarchical Poset Decoding (HPD, Guo et al., 2020) method (see Section 4.2), which arguably is not a fair baseline to LAGr because of its use of sketch prediction and lexicons. Notably, when these techniques are not used, LAGr performs much better than their base HPD algorithm.

To further zoom into the impact of the weakly-supervised LAGr's hyperparameters, we report results of preliminary experiments[2] in which we tuned the number of alignment candidates K and the noise level σ. One can see that choosing the best alignment out of $K > 1$ candidates is indeed helpful, and that noise of high magnitude ($\sigma = 10$) brings the best improvement on the random split. These improvements also translate into systematic generalization gains for MCD2 and MCD3, as shown in Table 5.4 where we see that $K = 5$ achieves better performance than $K = 1$. The positive effect of a larger K on these splits is in line with our expectation since 3.7 - 5.7% of examples in

[2]These experiments were carried out using an earlier preliminary implementation. Results in Table 5.5 are thus not directly comparable to those reported in Table 5.4.

		Graph Accuracy	
K	σ	train	test
1	0.0	99.79 (∓0.4)	98.75 (∓0.5)
5	0.01	99.92 (∓0.1)	99.01 (∓0.2)
	0.1	99.88 (∓0.1)	99.10 (∓0.3)
	1.0	99.85 (∓0.2)	99.10 (∓0.3)
	10.0	**99.97** (∓0.1)	**99.69** (∓0.1)
	15.0	83.78 (∓1.6)	83.73 (∓1.7)
	20.0	2.18 (∓0.17)	2.28 (∓0.19)
10	0.01	99.77 (∓0.3)	98.85 (∓0.6)
	0.1	99.92 (∓0.1)	99.10 (∓0.2)
	1.0	99.70 (∓0.3)	98.68 (∓0.7)
	10.0	99.96 (∓0.1)	99.58 (∓0.2)
	15.0	99.77 (∓0.4)	99.42 (∓0.5)
	20.0	69.69 (∓3.9)	68.91 (∓4.0)

Table 5.5: The effect of the number of alignment candidates K and noise level σ on the performance of weakly-supervised LAGr using CFQ's random split. We report the average graph accuracy and the standard deviation over 5 runs. We show the best configuration in bold.

each CFQ split have at least two predicates with identical node labels, which can make it hard to align the MR graph to the input by looking at node labels only. Interestingly, in contrast to our intuition, when using ten candidate alignments, the random split test performance is slightly worse than when using five.

In Section 5.3, we present examples of the node labels that weakly-supervised LAGr predicts in the learned aligned CFQ graphs as well as the corresponding SPARQL queries.

5.3 Error Analysis

Table 5.6 shows some commonly encountered errors on COGS with strongly-supervised LAGr. In all examples, the model predicted the correct set of nodes. However, even when all nodes are correctly predicted, some may not show up in the final logical form, if it has no connecting edges to other nodes (see the node labeled as "*dog*" in example 4). Table 5.7 shows the predicted nodes of aligned graphs and resulting queries produced by the best weakly-supervised LAGr model on CFQ. The top two rows show common errors where some edge labels do not get predicted, and where some nodes are missing due to the model not having predicted any connecting edges for the

nodes, thus omitting the nodes from the final output graph. The bottom two rows show the inferred aligned graphs for examples that result in the correct output graph.

Example 1: wrong edge label, between right nodes

In	A cockroach sent Sophia the sandwich beside the yacht .
Out	* sandwich (x _ 5) ; * yacht (x _ 8) ; cockroach (x _ 1) AND send . **theme** (x _ 2 , x _ 1) AND send . recipient (x _ 2 , Sophia) AND send . theme (x _ 2 , x _ 5) AND sandwich . nmod . beside (x _ 5 , x _ 8)
Pred	* sandwich (x _ 5) ; * yacht (x _ 8) ; cockroach (x _ 1) AND send . **agent** (x _ 2 , x _ 1) AND send . recipient (x _ 2 , Sophia) AND send . theme (x _ 2 , x _ 5) AND sandwich . nmod . beside (x _ 5 , x _ 8)

Example 2: Right edge label, but between wrong nodes

In	The girl beside the bed lended the manager the leaf .
Out	* girl (x _ 1) ; * bed (x _ 4) ; * manager (x _ 7) ; * leaf (x _ 9) ; **girl . nmod . beside (x _ 1 , x _ 4)** AND lend . agent (x _ 5 , x _ 1) AND lend . recipient (x _ 5 , x _ 7) AND lend . theme (x _ 5 , x _ 9)
Pred	* girl (x _ 1) ; * bed (x _ 4) ; * manager (x _ 7) ; * leaf (x _ 9) ; lend . agent (x _ 5 , x _ 1) AND lend . recipient (x _ 5 , x _ 7) AND lend . theme (x _ 5 , x _ 9) AND **leaf . nmod . beside (x _ 9 , x _ 4)**

Example 3: Mistaking edge labels

In	The dog noticed that a hippo juggled .
Out	* dog (x _ 1) ; notice . agent (x _ 2 , x _ 1) AND notice . ccomp (x _ 2 , x _ 6) AND hippo (x _ 5) AND juggle . **agent** (x _ 6 , x _ 5)
Pred	* dog (x _ 1) ; notice . agent (x _ 2 , x _ 1) AND notice . ccomp (x _ 2 , x _ 6) AND hippo (x _ 5) AND juggle . **theme** (x _ 6 , x _ 5)

Example 4: Correct nodes, but incorrect edges predicted

In	A dog beside a chair said that a melon on the bed was liked .
Out	* bed (x _ 11) ; dog (x _ 1) AND **dog . nmod . beside (x _ 1 , x _ 4)** AND chair (x _ 4) AND say . agent (x _ 5 , x _ 1) AND say . ccomp (x _ 5 , x _ 13) AND melon (x _ 8) AND **melon** . nmod . on (x _ 8 , x _ 11) AND like . theme (x _ 13 , x _ 8)
Pred	* bed (x _ 11) ; chair (x _ 4) AND say . agent (x _ 5 , x _ 4) AND melon (x _ 8) AND **bed** . nmod . in (x _ 11 , x _ 13) AND like . theme (x _ 13 , x _ 8)

Table 5.6: Incorrectly predicted logical forms for COGS with strongly-supervised LAGr. Errors are highlighted in bold.

Example 1: Wrong edge predictions

Layer 2	?x0	M3	influenced				director			spouse	M2	?x2	cinematographer				M4					?x1	actor
Layer 1																							
Input	Did	M3	influence	a	film		director	,		marry	M2	's	cinematographer	,	influence		M4	,	and	influence		a	actor
Target	?x1 actor . ?x0 director . ?x2 cinematographer M2 . FILTER M3 != ?x2 . M3 influenced [?x0 ?x1 M4] . M3 spouse ?x2																						
Predicted	?x0 actor . ?x0 director . ?x1 director . ?x2 cinematographer M2 . FILTER M3 != ?x2 . M3 influenced [?x0 ?x1 M4] . M3 spouse ?x2																						

Example 2: Missing node

Layer 2	select_?x0	ns:m.0f8l9c		editor		M1	influenced_ by		?x1	employer	?x2	organizations_founded						M2
Layer 1		nationality																
Input	What	French	film	editor	that	M1	influenced	influenced	a	company	s	founder	and	was	influenced	by	M2	
Target	?x1 actor . ?x0 director . ?x2 cinematographer M2 . FILTER M3 != ?x2 . M3 influenced [?x0 ?x1 M4] . M3 spouse ?x2																	
Predicted	?x0 actor . ?x0 director . ?x1 director . ?x2 cinematographer M2 . FILTER M3 != ?x2 . M3 influenced [?x0 ?x1 M4] . M3 spouse ?x2																	

Example 3: Correct prediction

Layer 2	select_?x0	ns:m.05zppz	ns:m.059j2		editor	director	M3
Layer 1			gender	nationality			
Input	Which	male	Dutch	film	editor	directed	M3
Predicted	select_?x0 director M3 . select_?x0 editor . select_?x0 gender ns:m.05zppz . select_?x0 nationality ns:m.059j2						

Example 4: Correct prediction

Layer 2	select_?x0		ns:m.06mkj	actor		influenced	M2		?x1	actor			
Layer 1	nationality	person											
Input	Who	was	a	Spanish	actor	that	influenced	M2	and	influenced	a	actor	
Predicted	?x1 actor . select_?x0 actor . select_?x0 influenced ?x1 . select_?x0 influenced M2 . select_?x0 person . select_?x0 nationality ns:m.06mkj												

Table 5.7: Predicted nodes of aligned graphs and resulting queries produced by the best weakly-supervised LAGr with $k = 5$, $\sigma = 10$ on the development set of CFQ. Top two rows show common errors with missing edge labels and missing nodes, and bottom rows show the inferred alignments for correct examples.

Chapter 6

Conclusion

6.1 Discussion

In this thesis, we hypothesized that generating meaning representations directly as graphs can lend itself to better compositional generalization in semantic parsing, i.e., help models generate meaning representations for examples that feature new combinations of meaning construction rules. To this end, we proposed the LAGr framework that outputs the meaning representation graphs by labeling the nodes and edges of a fully-connected, multi-layer output graph that is aligned with the input utterance. We presented two variants of LAGr based on whether alignment between the target graph and the input is available during training or not. First, when aligned graphs are available in the training dataset, we showed that LAGr can be trained with strong supervision, i.e. by minimizing standard classification loss functions such as the log likelihood. On the other hand, we proposed weakly-supervised LAGr for scenarios where alignments for originally unaligned target graphs are unavailable and thus, are treated as a latent variable. In this case, weakly-supervised LAGr infers latent alignments with a simple and novel approximate maximum-a-posteriori (MAP) inference approach that involves solving several minimum cost bipartite matching problems with the Hungarian algorithm (Kuhn, 1955). Finally, we use the resulting inferred aligned graphs for training LAGr.

We tested our hypothesis with strongly- and weakly-supervised LAGr through extensive experiments on the COGS (Kim & Linzen, 2020) and CFQ (Keysers et al., 2019) datasets - two benchmarks specifically designed to assess compositional generalization in neural semantic parsers. Our experiments showed that LAGr significantly improves upon sequence-to-sequence baselines in both strongly and weakly-supervised settings. Specifically on COGS, LAGr in both strongly- and weakly-supervised settings outperforms our carefully-tuned seq2seq baselines and performs similarly to LSTMs that leverage lexicons. While the use of lexicons can be integrated into LAGr, we do not expect this to improve LAGr performance on COGS, as both of our best performing strongly- and weakly-supervised LAGr models already predict node labels perfectly. Additionally, lexicons also bring their own challenges of dealing with context-dependency and ambiguity. For this reason, it is a notable result that LAGr matches the performance of lexicon-equipped models while making less assumptions about the nature of the input-to-output mapping.

Our experiments on CFQ showed that weakly-supervised LAGr outperforms all seq2seq baselines all out-of-distribution MCD splits. This result holds irrespective of using $K = 1$ versus $K > 1$ alignment candidates. Notably, however, on two out of three MCD splits we do observe an improvement from our proposed approximate inference procedure in which we use a shortlist of $(K > 1)$ candidate alignments by solving K noisy minimum cost bipartite matching problems.

6.2 Limitations

A limitation of LAGr is that node and edge predictions are done independently of one another. That is, the edge prediction model has no knowledge of the underlying node labels of the connecting nodes. This limitation is clearly illustrated in our error analysis in Section 5.3 that demonstrates that the majority of errors are made due to missing or wrong edge label predictions. For this reason, we believe that a modification of LAGr that conditions edge predictions on node labels could bring further improvements. Importantly, this modification would only change the model, and thus would still be compatible with our current alignment inference algorithm.

In our experiments on CFQ, LAGr has to predict named variables (`?x0, ?x1` etc.). This was done such that we can compare predicted graphs with the serialized SPARQL queries from prior work. We expect that further improvements in compositional generalziation could be achieved by removing variable names from LAGr's prediction task. Our hypothesis is that this simplification would debias models from learning the arbitrary order in variable names (denoted by variable indices).

Additionally, while the current alignment inference algorithm is effective, applying more advanced discrete optimization or amortized inference methods could be an interesting direction for future work.

Lastly, while this was outside the scope of this thesis, we also expect that using more powerful pretrained backbone models as encoders in LAGr could also lead to further gains in compositional generalization.

6.3 Future Directions

LAGr's strategy to perform semantic parsing by labeling aligned graphs unlocks many advantages compared to prior work. First, it opens the door for models to generate meaning representations as arbitrary graphs. This sets forth a promising direction for LAGr to benefit other types of meaning representations such as the abstract syntax trees of SQL database queries, Python programs or other more complex formal languages. Another advantage of LAGr is that it predicts meaning representations in constant time, instead of previous methods that perform decoding sequentially.

On the other hand, with concurrent advances in large-scale pretrained natural language to code models such as Codex (Chen et al., 2021), another fruitful avenue could be to incorporate LAGr into such models. Currently, pretrained models are extremely computation-intensive, making it difficult for other deep learning researchers and practitioners that do not have the necessary computation budget to access and partake in building pretrained models. We speculate that the improvements in compositional generalization can potentially lend themselves to more efficient ways to pretrain

large-scale language models that require less compute. Consequently, we hope that LAGr is a fruitful contribution in this direction.

Bibliography

Harold Abelson and Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. The MIT Press, 1996.

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. Learning to Recombine and Resample Data For Compositional Generalization. In *International Conference on Learning Representations, ICLR*, 2021.

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural Language Interfaces To Databases - An Introduction. *Natural Language Engineering*, 1(1):29–81, 1995.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations, ICLR*, 2015.

Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. CLOSURE: Assessing Systematic Generalization of CLEVR Models. *arXiv preprint:1912.05783*, 2019.

Robert C Berwick, Robert Cregar Berwick, and Robert S Berwick. *The Acquisition of Syntactic Knowledge*, volume 16. MIT press, 1985.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

C Ted Briscoe. Introduction To Formal Semantics For Natural Language. 2011. URL `www.cl.cam.ac.uk/teaching/1011/L107/semantics.pdf`.

John Seely Brown and Richard R Burton. Multiple Representations of Knowledge for Tutorial Reasoning. In *Representation and Understanding*, pp. 311–349. Elsevier, 1975.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint:2107.03374*, 2021.

Xinyun Chen, Chang Liu, and Dawn Song. Execution-guided neural program synthesis. In *International Conference on Learning Representations (ICLR)*, 2018.

David Chiang. Hierarchical Phrase-based Translation. *Computational Linguistics*, 33(2):201–228, 2007.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Noam Chomsky. *Syntactic Structures*. Universitas Negeri Malang, 1957.

Alonzo Church. A Set of Postulates for the Foundation of Logic. *Annals of Mathematics*, pp. 346–366, 1932.

EF Codd. Data Base Management. In *Proceedings of National Computer Conference and Exposition*, pp. 377–378, 1975.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. Meta-Learning to Compositionally Generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The Devil is in the Detail: Simple Tricks Improve Systematic Generalization of Transformers. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Donald Davidson. Truth and Meaning. In *Philosophy, Language, and Artificial Intelligence*, pp. 93–111. Springer, 1967.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-Grounded Pretraining for Text-to-SQL. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2018.

Li Dong and Mirella Lapata. Language to Logical Form with Neural Attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Li Dong and Mirella Lapata. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Timothy Dozat and Christopher D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. *arXiv preprint:1611.01734*, 2017.

Heinrich-Heine-Universität Düsseldorf. *Truth-conditional Theories of Meaning*, pp. 127–170. Düsseldorf University Press, 2017.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving Text-to-SQL Evaluation Methodology. *arXiv preprint:1806.09029*, 2018.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *arXiv preprint:2007.08970*, 2020.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Christopher Pal. Measuring Systematic Generalization in Neural Proof Generation with Transformers. *arXiv preprint:2009.14786*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. *arXiv preprint:1905.08205*, 2019.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. Hierarchical poset decoding for compositional generalization in language. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. X-sql: reinforce schema representation with context. *arXiv preprint:1908.08113*, 2019.

Gary G Hendrix, Earl D Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS)*, 1978.

Jonathan Herzig and Jonathan Berant. Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Jonathan Herzig and Jonathan Berant. Span-based semantic parsing for compositional generalization. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations. *arXiv preprint:2104.07478*, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8): 1735–1780, 1997.

John E. Hopcroft, Rajeev Motwani, Rotwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 1979.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. *arXiv preprint:1902.01069*, 2019.

Robin Jia and Percy Liang. Data Recombination for Neural Semantic Parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint:1707.07328*, 2017.

Cocke John and T Schwartz Jacob. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, 1970.

Tim Johnson. Natural Language Computing: The Commercial Applications. *The Knowledge Engineering Review*, 1984.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2019.

Najoung Kim and Tal Linzen. COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 2016.

Angelika Kratzer and Irene Heim. *Semantics in generative grammar*, volume 1185. Blackwell Oxford, 1998.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics quarterly*, 2(1-2):83–97, 1955. Publisher: Wiley Online Library.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1512–1523, 2011.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2018.

Fred Landman and Frank Veltman. *Varieties of formal semantics: proceedings of the fourth Amsterdam Colloquium, September 1982*. Number 3. Foris Publications, 1984.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. KaggleDBQA: Realistic Evaluation of Text-to-SQL Parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, 2021.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. Compositional Generalization for Primitive Substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning Dependency-Based Compositional Semantics. *Computational Linguistics*, 39(2):389–446, 2012.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Chunchuan Lyu and Ivan Titov. AMR Parsing as Graph Prediction with Latent Alignment. *arXiv preprint:1805.05286 [cs]*, May 2018. URL `http://arxiv.org/abs/1805.05286`. arXiv: 1805.05286.

Claudia Maienborn, Klaus von Heusinger, and Paul Portner. *Semantics: An International Handbook of Natural Language Meaning*, volume 1. Walter de Gruyter, 2011.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, (ACL)*, 2019.

Ryan Mcdonald. *Discriminative learning and spanning tree algorithms for dependency parsing*. phd, University of Pennsylvania, 2006.

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

Richard Montague. The Proper Treatment of Quantification in Ordinary English. 1973.

Richard Montague. *English as a Formal Language*. De Gruyter Mouton, 2019.

Santiago Ontañón, Joshua Ainslie, Vaclav Cvicek, and Zachary Fisher. Making Transformers Solve Compositional Tasks. *arXiv preprint:2108.04378*, 2021.

Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. 1990.

Panupong Pasupat, Sonal Gupta, Karishma Mandyam, Rushin Shah, Mike Lewis, and Luke Zettlemoyer. Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Francis Jeffry Pelletier. The Principle of Semantic Compositionality. *Topoi*, 13(1):11–24, 1994.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140, 2016.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint:1904.09708*, 2019.

Torsten Scholak, Raymond Li, Dzmitry Bahdanau, Harm de Vries, and Chris Pal. DuoRAT: Towards Simpler Text-to-SQL Models. October 2020. URL `https://arxiv.org/abs/2010.11119v1`.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 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*, 2021.

Mark Steedman. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 1987.

Mark Steedman. *Surface Structure and Interpretation*. MIT press, 1996.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NeurIPS)*, 2014.

Cynthia A. Thompson and Raymond J. Mooney. Acquiring word-meaning mappings for natural language interfaces. 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conferenceon Neural Information Processing Systems (NeurIPS)*, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

David L Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, 1978.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 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 (ACL)*, 2020.

Bailin Wang, Mirella Lapata, and Ivan Titov. Structured Reordering for Modeling Latent Alignments in Sequence Transduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust Text-to-SQL Generation with Execution-Guided Decoding. *arXiv preprint:1807.03100*, 2018a.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. A Tree-based Decoder for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018b.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making Neural QA as Simple as Possible but not Simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*, 2017.

Ruediger Wirth. Completing logic programs by inverse resolution. *EWSL-89*, 1989.

William Woods, Ronald Kaplan, and Bonnie Webber. The Lunar Science Natural Language Information System: Final Report. 1972.

William A. Woods. Progress in natural language understanding: an application to lunar geology. In *American Federation of Information Processing Societies: 1973 National Computer Conference*, volume 42 of *AFIPS Conference Proceedings*, pp. 441–450, 1973.

Xiaojun Xu, Chang Liu, and Dawn Song. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. *CoRR*, 2017.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8413–8426, 2020.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 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 (NAACL)*, 2021.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 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 (EMNLP)*, 2018.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *9th International Conference on Learning Representations, (ICLR)*, 2021.

John M. Zelle and Raymond J. Mooney. Learning semantic grammars with constructive inductive logic programming. In *AAAI'93 Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1993.

John M. Zelle and Raymond J. Mooney. Inducing Deterministic Prolog Parsers from Treebanks: A Machine Learning Approach. In *AAAI '94 Proceedings of the Twelfth National Conference on Artificial Intelligence - Volume 1*, 1994.

John M. Zelle and Raymond J. Mooney. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, 1996.

Luke Zettlemoyer and Michael Collins. 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)*, 2007.

Luke S. Zettlemoyer and Michael Collins. 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*, 2005.

Hao Zheng and Mirella Lapata. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2020.