Grounded Graph Decoding Improves Compositional Generalization in Question Answering

Yu Gai* and **Paras Jain*** and **Wendi Zhang** and
Joseph Gonzalez and **Dawn Song** and **Ion Stoica**
University of California, Berkeley
{yu_gai, paras_jain, wendizhang,
jegonzal, dawnsong, istoica}@berkeley.edu

Abstract

Question answering models struggle to generalize to novel compositions of training patterns, such to longer sequences or more complex test structures. Current end-to-end models learn a flat input embedding which can lose input syntax context. Prior approaches improve generalization by learning permutation invariant models, but these methods do not scale to more complex train-test splits. We propose Grounded Graph Decoding, a method to improve compositional generalization of language representations by grounding structured predictions with an attention mechanism. Grounding enables the model to retain syntax information from the input in thereby significantly improving generalization over complex inputs. By predicting a structured graph containing conjunctions of query clauses, we learn a group invariant representation without making assumptions on the target domain. Our model significantly outperforms state-of-the-art baselines on the Compositional Freebase Questions (CFQ) dataset, a challenging benchmark for compositional generalization in question answering. Moreover, we effectively solve the MCD1 split with 98% accuracy. All source is available at https://github.com/gaiyu0/cfq.

1 Introduction

Can neural networks "*make infinite use of finite means*" with language (Chomsky and Lightfoot, 2002)? The ability of humans to reason compositionally enables us to form novel complex compound sentences by combining constituent concepts. Toki Pona (Lang, 2014) is an engineered language with only 120 words but can express a wide variety of concepts through composition.

Compositionality specifically refers to the phenomenon that the meaning of an expression is given by combining the meanings of its parts (Montague, 1970). For example, after understanding the questions "Who directed Inception?" and "Did Christopher Nolan produce Goldfinger?", one can understand "Who produced Inception?".

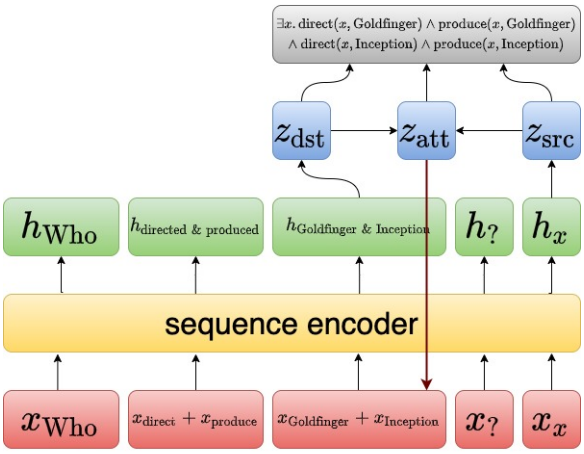

Figure 1: An illustration of Grounded Graph Decoding with the question "Who directed and produced Goldfinger and Inception?". The predicates "direct(ed)" and "produce(d)", and the entities "Goldfinger" and "Inception" are grouped together by the sequence encoder. Black lines represent linear transforms between latent spaces, while the red line represents the attention that grounds the prediction.

However, it's unclear whether neural networks truly reason compositionally; for example, Min et al. (2019) argue that compositional reasoning is not necessary to answer complex composite questions. To benchmark the performance of neural models under compositional generalization, Lake and Baroni (2018) propose the SCAN dataset containing sequences of instructions. SCAN uses length of input as a proxy for compositional complexity. However, SCAN's length-based splits are now considered solved (Chen et al., 2020).

As a more realistic benchmark, Keysers et al. (2020) introduced the Compositional Freebase Questions (CFQ) dataset. CFQ is a challenging knowledge graph question answering task with ad-

* equal contribution

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1829–1838
November 7–11, 2021.

versarial splits (MCD1, MCD2 and MCD3) that maximize compositional divergence between train and test sets. While a simple LSTM model achieves near-perfect accuracy on an i.i.d. split, large models such as T5-11B (Raffel et al., 2019) performs catastrophically poorly on the MCD splits.

We hypothesize current models perform poorly on the CFQ task as they fail to take semantic and syntactic structures into account. We propose Grounded Graph Decoding to incorporate these structures with graph decoding and grounding. The encoder often fails to match common syntactic patterns from the input text question. We modify the decoder with attention over the input sequence to enable referencing specific input syntactic structures. Moreover, we find the semantics of knowledge graph queries (SPARQL) is a *conjunctive query graph*. Sequence decoders fail to represent this structure well. We integrate semantic structure by decoding output predictions as a graph.

Our work makes the following contributions:

- We ground the decoder using the input text in order to improve understanding of questions with novel syntactic structures.
- We leverage a conjunctive query graph decoder to enable generation of novel complex SPARQL queries.
- On the challenging Compositional Freebase Questions (CFQ) benchmark, Grounded Graph Decoding effectively solves the MCD1 split while achieving state-of-the art results on the remaining two splits.

2 Preliminaries

We formally present an overview of compositional generalization in deep learning to motivate Grounded Graph Decoding.

2.1 Compositional generalization and its benchmarks

Training models that achieve compositional generalization would be an important advancement for both practical systems and for furthering our understanding of intelligence more broadly as compositional intelligence is a key characteristic of the human mind. Moreover, compositional generalization would improve the sample efficiency of models, as argued by Lake and Baroni (2018). This would improve answers for rare questions while accelerating learning by decomposing the combinatorial nature of language.

Benchmarks for compositional generalization measure how well neural models trained on one set of structures generalize to an unseen test set with novel structures. Lake and Baroni (2018) experimented with several heuristics for splitting seq2seq datasets, such as splitting by sequence length, and found that some of them posed significant challenge for state-of-the-art sequence-to-sequence (seq2seq) models to generalize.

Keysers et al. (2020) take a systematic approach based on the distinction between atoms and compounds. For example, in the questions "Who directed Inception?" and "Did Christopher Nolan produce Goldfinger?", the atoms are the primitive elements that form these questions. These elements include the predicates "direct(ed)" and "produce(d)", the question patterns "Who [predicate] [entity]" and "Did [entity1] [predicate] [entity2]", and the entities "Inception", "Christopher Nolan", etc. Compounds are intuitively combinations of atoms, such as "Who directed [entity]?", which is the combination of the predicate "direct(ed)" and the question template "Who [predicate] [entity]?", etc. Keysers et al. (2020) proposes the following objective for partitioning the dataset: (a) minimum atom divergence, which ensures atoms that occur in test sets also occur in training sets and (b) maximum compound divergence, which maximizes the number of compounds in test sets that are not present in training sets.

Intuitively, in order to succeed on such splits, a model has to both learn the meaning of atoms, and learn the rules that combine the atoms. A reasonable question to appear in the test set of a MCD split, if the two questions above occur in the training set of the split, can be "Who produced Goldfinger?", which consists of only known atoms, but entirely different compositions.

2.2 Conjunctive queries and knowledge graph question answering (KG-QA)

Many factual questions can be answered by executing conjunctive queries against knowledge graphs (KG). Formally, given a KG $\mathcal{K} = (\mathcal{E}, \mathcal{R})$ that consists of a set of entities $\mathcal{E}$ and a set of relation triples $\mathcal{R}$, a conjunctive query

$$\exists x_1, \ldots, x_n . r_1(s_1, o_1) \wedge \ldots \wedge r_m(s_m, o_m)$$

against $\mathcal{K}$ is the conjunction of predicates $r_1(s_1, o_1), \ldots, r_m(s_m, o_m)$. The subject s_i and

object o_i of each predicate can be either an entity in $\mathcal{E}$, or one of the variables $x_1, \ldots, x_n$, and predicate $r_i(s_i, o_i)$ is true if and only if relation r_i holds between s_i and o_i in $\mathcal{K}$. For example, to answer the question "Who directed and produced Inception?", we can use the conjunctive query "$\exists x.\text{direct}(x, \text{Inception}) \wedge \text{produce}(x, \text{Inception})$". Although general queries, such as lambda dependency-based compositional semantics (Liang, 2013) are also popular in the KG-QA literature, both SCAN and CFQ find that parsing questions into conjunctive queries alone presents a nontrivial challenge to compositional generalization despite their simplicity.

Compared to other question answering settings, such as reading comprehension question answering (RC-QA), KG-QA is exclusively concerned with question comprehension, making it an ideal unit test for QA models. We consider the *strongly supervised* setting of KG-QA, where each question for training is paired with a query that, when executed against a KG, yields the question's answer. Although this is less challenging than the *weakly supervised* setting, where only answers to questions are available, both (Keysers et al., 2020) and (Furrer et al., 2020) show that this setting already requires a level of compositional generalization not yet possessed by existing systems.

2.3 Compositionality and compositional language representation

Compositionality refers to the phenomenon that the meaning of an expression is given by combining the meanings of its parts (Montague, 1970). Specifically, for an expression z composed of subexpressions x and y, its semantics $\tau(z)$ is given by

$$\tau(z) = \tau(x \oplus y) = \tau(x) \odot \tau(y)$$

where $\oplus$ is a syntactic composition operator, $\odot$ is an semantic composition operator, and the semantics $\tau(x)$ and $\tau(y)$ of x and y are given recursively by the same rule.

For example, the meaning of the phrase "direct and produce" is given by the conjunction of the words "direct" and "produce". In this case the syntactic composition operator $\oplus$ simply takes two words and adds an "and" in between, while the semantic composition operator $\odot$ yields the conjunction of the two predicates. This phrase can be further composed with other phrases to express more complex meanings. Various formalisms exist for this process, such as the Compositional Categorial Grammar (Zettlemoyer and Collins, 2005).

The key question is how to model semantics in vector spaces. Motivated by Montague (1970), Andreas (2019) defines a neural model as compositional if it is a homomorphism from syntax trees to vector representations, that is, for an expression $z = x \oplus y$, its representation $\theta(z)$ is given by:

$$\theta(z) = \theta(x \oplus y) = \theta(x) \odot \theta(y)$$

This formulation should be simple to implement if both the syntactic and semantic structure of a language is known. However, a key challenge to building compositional natural language representation is the lack of both syntactic and semantic structure. First, natural languages only loosely follow the compositionality principle. Especially, natural languages are not always context-free, which is implicitly assumed in this formulation of compositionality. The semantics $\tau(x)$ can depend on only the phrase x only and not on phrase y. Second, the syntactic structure is not always known.

Despite the excellent performance of constituency parsers on benchmarks, they may generalize poorly to compositionally complex questions (see Section 4 for more discussion). Despite substantial progress in the NLP community to learn vector representations of semantics, how to learn compositionally generalizable vector representations of semantics without much knowledge of compositional structures remains challenging.

3 Methodology: Grounded Graph Decoding

In the following sections, we first describe how to decode conjunctive query graphs from natural language questions, then how to incorporate syntactic compositions in graph decoding, and finally introduce a grounding mechanism by explicitly conditioning graph decoding on syntactic compositions.

3.1 Graph decoding

A conjunctive query can be naturally represented as a directed graph by representing subjects and objects in the query as nodes, and relational predicates in the query as directed edges. For example, the conjunctive query

$$\exists x_1.\text{direct}(x_1, \text{Inception}) \wedge \text{produce}(x_1, \text{Inception})$$

can be represented as a directed graph with nodes x_1 and "Inception", and edges

$$x_1 \xrightarrow{\text{direct}} \text{Inception} \qquad x_1 \xrightarrow{\text{direct}} \text{Goldfinger}$$

where the type of an edge is the relation in the corresponding relational predicate. The key benefit of the directed graph representation is permutation invariance. Directed graphs are invariant to ordering of edges as conjunctive queries are invariant to ordering of relational predicates.

Similar to the graph decoder architecture proposed by Kipf and Welling (2016), our graph decoder first generates embeddings for nodes, and then predicts edges in graphs using the node embeddings. For simplicity, we assume that questions are tokenized into words, and entities in questions have been found and anonymized (Finegan-Dollak et al., 2018). The embedding of an entity node is simply the contextual embedding of its mentions in the question. If an entity is mentioned multiple times in the question, its node embedding is the sum of the contextual embeddings of all its mentions. Given a question q consisting of tokens $q_1, \ldots, q_l$, we concatenate the question with the list of variables $x_1, \ldots, x_n$ into a new sequence

$$q_1, \ldots, q_l, [\text{SEP}], x_1, \ldots, x_n$$

and input this sequence to a sequence encoder, such as an LSTM (Hochreiter and Schmidhuber, 1997). The embedding of the variable node x_i is the contextual embedding of the token x_i. Intuitively, these contextual node embeddings capture relations between both entity and variable nodes. We denote the embedding for node v as h_v.

Given the node embeddings, the probability that an edge $s \xrightarrow{r} o$ exists between subject s and object o is modeled as

$$P(s \xrightarrow{r} o|q) = \sigma\left(w_r^T [h_s, h_o]\right) \tag{1}$$

where σ denotes the sigmoid function, $w_r^T \in \mathbb{R}^{2d}$ is the weight vector specific to relation r, h_s and h_o are the node embeddings of subject s and object o respectively, and $[\cdot, \cdot]$ denotes vector concatenation. Multiple relations may hold between an entity pair.

The model is trained to maximize the conditional log-likelihood of all conjunctive query graphs in training set $\mathcal{Q}$:

$$\begin{aligned}\mathcal{L} &= \sum_{q \in \mathcal{Q}} \log P(\tau(q)|q) \\ &= \sum_{q \in \mathcal{Q}} \sum_{s \xrightarrow{r} o \in \mathcal{E}(\tau(q))} \log P(s \xrightarrow{r} o|q) \\ &\quad + \sum_{s \xrightarrow{r} o \notin \mathcal{E}(\tau(q))} \log(1 - P(s \xrightarrow{r} o|q)) \end{aligned} \tag{2}$$

where $\tau(q)$ denotes the conjunctive query graph for question q, and $\mathcal{E}(\tau(q))$ denotes the edges in $\tau(q)$.

3.2 Incorporating syntactic compositions

Although our graph decoder incorporates the compositional structure of conjunctive queries, it still cannot capture syntactic compositions in natural language questions due to the lack of compositionality in the sequence encoder. A solution to the problem is to embed syntactic structures in the sequence encoder. In general, however, these structures can be hard to identify.

We manually evaluated parses returned by the Stanford constituency parser (Bauer, 2014) given questions in the CFQ dataset, and observed a high error rate, possibly due to complex compositional structures in the questions. As a pilot study, instead of incorporating syntactic compositions exhaustively, we only consider a simple syntactic composition "A and B", where "A" and "B" share the same part-of-speech. This composition can be reliably identified using part-of-speech (PoS) tagging. Following the suggestion of Andreas (2019), the embeddings θ that model semantic compositions should satisfy

$$\theta(\text{"A and B"}) = \theta(\text{"A"}) \odot \theta(\text{"B"})$$

A key criterion for choosing the semantic composition operator $\odot$ is *permutation invariance*, which is evident from the fact that "A and B" has the same meaning as "B and A". We model the semantics of both "A" and "B" using learnable vector embeddings, and model the semantic composition using vector addition. This model can be easily extended to the case where both "A" and "B" are phrases, in which case both θ("A") and θ("B") can be embeddings generated by sequence encoders, and $\odot$ can be any learnable composition operator. However, for simplicity, we only consider the case that "A" and "B" are single words. To implement this idea, instead of inputting

$$q_1, \ldots, q_l, [\text{SEP}], x_1, \ldots, x_n$$

to the sequence encoder, we now merge the phrases like "A and B" into a single token, which we call a group, and input the groups to a sequence encoder

$$g_1, \ldots, g_k, [\text{SEP}], x_1, \ldots, x_n$$

The embedding x_g of a group g is given by

$$\theta(g) = \sum_{w \in g} \theta(w)$$

where $\theta(w)$ is the embedding of word w. For example, the embedding of "A and B" is $\theta(\text{A}) + \theta(\text{B})$. The sum can be replaced by learnable composition operators to model more complex semantic compositions.

When an entity is mentioned in a group, we use the contextual embedding of the group as its node embedding. The probability for the conjunctive query graph to contain a predicate $s \xrightarrow{r} o$ is now modeled as

$$P(s \xrightarrow{r} o|q) = \sigma\left(w_r^T [g_s, g_o]\right) \quad (3)$$

and all parameters are still learned by maximizing the joint log-likelihood in Eqn. 2.

3.3 Grounded graph decoding

Curiously, we find in preliminary experiments that although incorporating syntactic compositions in the sequence encoder improves compositional generalization, the incorporated syntactic compositions are not always reflected in the model's outputs. For example, given the question "Who directed and produced Inception?", the model may output only the predicate $\text{direct}(x, \text{Inception})$ but not the other predicate $\text{produce}(x, \text{Inception})$, despite "produced" being grouped with "directed" by the sequence encoder.

The phenomenon indicates that syntactic compositions are insufficiently preserved, and the model actually fails to learn the correspondence between syntactic and semantic compositions. To encourage the model to learn the correspondence, we propose to augment the graph decoder with a mechanism that enables it to ground its outputs in syntactic compositions. Specifically, we add a grounded embedding $z_{s,o}$ to Eqn. 3:

$$P(s \xrightarrow{r} o|q) = \sigma(w_r^T [h_s, h_o, z_{s,o}])$$

The grounded embedding $z_{s,o}$ is given by an weighted average over syntax compositions. Mathematically,

$$z_{s,o} = \sum_{k=1}^{l} \alpha_{s,o}^{(k)} \nu_k = \sum_{k=1}^{l} \alpha_{s,o}^{(k)} \sum_{w \in g_k} x_w$$

The attention weight $\alpha_{s,o}^{(k)}$, which quantifies the relevance of the k-th group to the subject-object pair (s, o), is given by

$$\alpha_{s,o}^{(k)} = \frac{\exp(a_{s,o}^{(k)}/\sqrt{d})}{\sum_{j=1}^{l} \exp(a_{s,o}^{(j)}/\sqrt{d})}$$

where following Vaswani et al. (2017), we set the temperature of the softmax function to $\sqrt{d}$, square root of the latent space dimension. The unnormalized attention score $a_{s,o}^{(k)}$ is given by an inner product between the query $q_{s,o}$ and the key κ_k

$$a_{s,o}^{(k)} = q_{s,o}^T \kappa_k$$

Both the query $q_{s,o}$ and the key κ_k are given by linear transforms of their contextual embedding

$$q_{s,o} = Q[h_s, h_o] \qquad \kappa_k = K h_k$$

All parameters in the model are still learned by maximizing the joint log-likelihood in Eqn. 2. We append a special "NIL" token to the end of each question, so that the graph decoder can attend to this token when no relation exists between a pair of subject and object. See Fig. 1 for an illustration of the architecture. As this embedding is specific to node pairs, the model is able to ground different edges in different syntactic compositions.

4 Experiments

We evaluate Grounded Graph Decoding to understand how grounding as well as graph decoding improve semantic and syntactic understanding.

4.1 Dataset

We evaluate our model using the Compositional Freebase Queries (CFQ) dataset (Keysers et al., 2020), a semantic parsing dataset consisting of approximately 240k natural language questions paired with conjunctive queries. Compared to other semantic parsing datasets, the CFQ dataset features richer question patterns, making it an ideal benchmark for compositional generalization (see Keysers et al. (2020) for a quantitative comparison).

The CFQ dataset consists of three Maximum-Compound Divergence (MCD) splits (MCD1, MCD2, and MCD3), constructed by running a greedy algorithm with different initializations that maximize compound divergence between training and test sets, while keeping atom divergence between them minimum (see Section 2 for more details about atom and compound divergence).

Method	# Params.	Accuracy per-split		
		MCD1	*MCD2*	*MCD3*
LSTM w/ attention (Keysers et al., 2020)		28.9 ± 1.8%	5.0 ± 0.8%	10.8 ± 0.6%
Transformer (Keysers et al., 2020)		34.9 ± 1.1%	8.2 ± 0.3%	10.6 ± 1.1%
Universal Transformer (Keysers et al., 2020)		37.4 ± 2.2%	8.1 ± 1.6%	11.3 ± 0.3%
Evolved Transformer (Keysers et al., 2020)		42.4 ± 1.0%	9.3 ± 0.8%	10.8 ± 0.2%
T5-base (Furrer et al., 2020)	220M	57.6 ± 1.4%	19.5 ± 1.0%	16.6 ± 1.5%
T5-large (Furrer et al., 2020)	770M	63.3 ± 0.6%	22.2 ± 1.5%	18.8 ± 2.6%
T5-11B (Furrer et al., 2020)	11000M	61.4 ± 4.8%	30.1 ± 2.2%	31.2 ± 5.7%
T5-11B (modified) (Furrer et al., 2020)	11000M	61.6 ± 12.4%	31.3 ± 12.8%	33.3 ± 2.3%
Grounded Graph Decoding	0.3M	**97.9 ± 0.2%**	**47.1 ± 10.4%**	**50.8 ± 17.2%**

Table 1: **CFQ evaluation without tuning on development sets** Grounded Graph Decoding achieves significantly higher performance than state-of-the-art seq2seq baselines across all MCD splits. As the MCD1 accuracy of Grounded Graph Decoding is withing the range of baselines trained on the random split ($\sim$ 98%), we consider MCD1 to be solved.

Method	# Params.	Accuracy per-split		
		MCD1	*MCD2*	*MCD3*
Hierarchical Poset Decoding (Guo et al., 2020)		79.6%	59.6%	67.8%
CBR-KGQA (Das et al., 2021)		87.9%	61.3%	60.6%
T5-3B (Herzig et al., 2021)	3000M	65.0%	41.0%	42.6%
LIR + RIR (T5-3B) (Herzig et al., 2021)	3000M	88.4%	**85.3%**	**77.9%**
Grounded Graph Decoding	0.3M	**98.6%**	67.9%	77.4%

Table 2: **CFQ evaluation with tuning on development sets** Keysers et al. (2020) discourage tuning models with development sets in MCD splits as it compromises the divergence split. We therefore report separately in Table 2 results obtained with tuning on development sets. Results without tuning on development sets are not reported in Guo et al. (2020), Herzig et al. (2021), and Das et al. (2021).

4.2 Baselines

We compare Grounded Graph Decoding with various baselines established by Keysers et al. (2020) and Furrer et al. (2020), as well as HPD (Guo et al., 2020), CBR-KBQA (Das et al., 2021), as well as the fine-tuning scheme by Herzig et al. (2021).

The baseline seq2seq models include LSTM (Hochreiter and Schmidhuber, 1997) with attention (Bahdanau et al., 2015), Transformer (Vaswani et al., 2017), Universal Transformer (Dehghani et al., 2019), Evolved Transformer (So et al., 2019), and T5 (Raffel et al., 2019), all of which, except "T5-11B (modified)", output conjunctive queries as sequences of tokens. "T5-11B (modified)" refers to a scheme proposed by Furrer et al. (2020) in which T5 models only need to output conjunctive queries that are aligned with syntactic structures in questions. All seq2seq baselines are reported in Keysers et al. (2020) and Furrer et al. (2020).

Hierarchical Poset Decoding (HPD) follows the encoder-decoder architecture of seq2seq models, but instead of outputting conjunctive queries as sequences of tokens, it outputs them as posets. Compared to seq2seq models, this has the advantage that posets are partially invariant to permutation of relational predicates, although decoding actually can be made fully permutation invariant using the graph representation described in Section 3.

Concurrent work, CBR-KBQA (Das et al., 2021), is a semi-parametric scheme in which a parametric model parses questions into queries at test by combining relevant queries in training sets.

Concurrent work from (Herzig et al., 2021) proposes to fine-tune pre-trained models with not only questions and queries but also intermediate query representations that align with question structures. (The reported results are obtained using a T5 model with 3 billion parameters). Although this scheme is model-agnostic, designing intermediate query representations that align with question structures requires nontrivial insights into correspondence be-

Method	MCD1	MCD2	MCD3
Poset Decoding (Guo et al., 2020)	21.3%	6.4%	10.1%
Grounded Graph Decoding	**98.6%**	**67.9%**	**77.4%**
Syntax-aware graph decoding	76.0%	29.0%	32.7%
Graph decoding only	59.1%	25.7%	20.4%

Table 3: Ablations of Grounded Graph Decoding. Both grounding and grouping significantly improve accuracy on all splits. We also compare our bare graph decoder with poset decoding to show the advantage of the directed graph representation.

tween question and query structures. Grounded Graph Decoding learns the correspondence end-to-end thanks to the grounding mechanism.

4.3 Evaluation methodology

(Keysers et al., 2020) note that the structure of the development set is similar to the test set. Therefore, hyperparameter tuning could leak information from the test set. Unfortunately, not all baselines follow this evaluation procedure. Therefore, we report results obtained via development set tuning separately in Table 2. Results without tuning on development sets are not reported in Guo et al. (2020), Herzig et al. (2021), and Das et al. (2021). All results in Table 1 are obtained without tuning on development sets.

4.4 Ablations

We benchmarked the following ablations of Grounded Graph Decoding:

- *Graph decoding only* This is the bare graph decoder architecture described in Section 3.1, which uses a sequence encoder without awareness of syntactic composition and does not have the grounding mechanism.

- *Syntax-aware graph decoding* This is the architecture described in Section 3.2, with a graph decoder on top of a sequence encoder aware of the “A and B" syntax.

The purpose of the ablation study is to validate the effectiveness of both (1) syntax-aware sequence encoders and (2) the grounding mechanism. We also compare graph decoding with poset decoding to show the benefit of graph decoding.

4.5 Results and discussion

We report performance of models tuned with and without development sets in Table 2 and Table 1 respectively. Remarkably, in both settings, on the MCD1 split Grounded Graph Decoding attains accuracies attained previously only with random splits, effectively solving the split.

On the MCD2 and MCD3 splits of the CFQ dataset, Grounded Graph Decoding consistently outperforms the pre-trained T5 models regardless of their size (without the fine-tuning trick proposed by Herzig et al. (2021)), as well as HPD and CBR-KGQA. T5 performs remarkably when fine-tuned with lossy and reversible intermediate representations (LIR + RIR), though the results are not very surprising given that the LIRs and RIRs used for fine-tuning are tailored specifically to the CFQ dataset and can thus drastically ease the task.

Our error analysis shows that Grounded Graph Decoding is not able to completely solve the remaining two splits (MCD2 and MCD3) primarily as it only incorporates one type of syntactic composition, namely conjunctive queries of the form of “A and B". Pre-trained approaches such as T5 support a wider range of syntactic structures due to a more general training objective. However, rich syntactic structures acquired from pre-training contribute little to the compositional generalization.

We find that performance of T5 models are close to our graph decoder without any added compositionality (see Table 1). This suggests that pre-trained language models understand the syntax of conjunctive queries, possibly because their pre-training corpus contains conjunctive queries. However, their generally low performance indicates that they do not utilize the information. Domain specific IRs could mitigate these challenges but are complex to apply to a real-world dataset like CFQ.

The ablation study verifies that graph decoder, syntax-aware sequence encoder, and the grounding mechanism are all important to compositional generalization. Grounding with graph decoding alone (no permutation invariance) results in a state-of-the-art model on the MCD1 split. However, this

model struggles with MCD2 and MCD3. We also find that a directed graph is a better representation of conjunctive queries than poset decoding. This is due partially to the richer structure of a graph which is a natural fit for common query patterns.

4.6 Future work

There are several clear areas of future work beyond Grounded Graph Decoding. First, our method is highly complementary with pre-training based approaches to improving compositional generalization. Second, incorporating more complex syntactic structures which will likely further boost our results. Error analysis revealed conjunctive structures predominantly help improve accuracy in the MCD1 split but have less impact on the other splits.

5 Related work

Various approaches to the compositional generalization challenge posted by the CFQ dataset has been explored in prior or contemporary works, including Guo et al. (2020), Das et al. (2021), and Herzig et al. (2021). These approaches are discussed in more detail in Section 4.

Another promising approach that has received relatively less attention so far is grammar induction, which can potentially derive grammar rules directly from question-query pairs. Grammar induction methods, such as Zettlemoyer and Collins (2005), typically assumes a limited set of grammar rules to bootstrap the model, and then search some grammar spaces to find grammars that can lead to successful parsing of observed questions.

The idea of grammar induction has inspired various work that to different extent solved the SCAN dataset, such as Nye et al. (2020) or Chen et al. (2020). The advantage of grammar induction methods is that they can potentially identify the complete set of transformation rules and thus attain perfect compositional generalization. However, grammar induction methods are generally search-based, which limits their scalability to long sentences due to the size of search spaces.

Additionally, there has been considerable research in the semantic parsing literature to design neural network architectures that incorporate different query structures, including tree (Dong and Lapata, 2016), graph (Buys and Blunsom, 2017; Damonte et al., 2017; Lyu and Titov, 2018; Fancellu et al., 2019). However, these architectures only incorporating query structures without incorporating syntactic structures in questions. Our ablation study (Table 3) indicates that only incorporating query structure is insufficient for compositional generalization. Our graph decoder alone only attains performance on bar with T5 models.

Similar to our work, Russin et al. (2019) also proposes to improve the compositional generalization of seq2seq models using attention. However, their work only studies token-level attention without consideration of syntactic or semantic structures. Both Russin et al. (2019) and Gordon et al. (2020) use part-of-speech (PoS) tags to attain some level of invariance among words that share the same PoS.

Finally, in the domain of semantic parsing, prior to Keysers et al. (2020), Finegan-Dollak et al. (2018) proposed to split datasets such that training and test sets contain no common SQL patterns. Although this approach increases task difficulty, different SQL query patterns may still share similar substructures, which enables neural networks to solve the tasks relatively easily using the “mix-and-match" strategy (Lake and Baroni, 2018).

6 Conclusion

In this paper we propose Grounded Graph Decoding to make compositionally generalizable predictions of conjunctive query from natural language questions. Our model consists of a graph decoder that captures permutation invariance in conjunctive queries, a sequence encoder that is aware of syntactic composition, and an attention mechanism that enables strong association between syntactic and semantic compositions. The proposed method solves the MCD1 split of the challenging CFQ dataset, and improves the state-of-the-art of the other two splits. Notably, Grounded Graph Decoding significantly outperforms competitive baselines including large pre-trained models (such as T5) as well as domain-specific models. Careful ablations of our method demonstrate the importance of both graph decoding and grounding.

Acknowledgments

We thank Lisa Dunlap, Daniel Furrer, Ajay Jain, Daniel Rothchild, Nathan Scales, Rishabh Singh, Justin Wong and Marc van Zee for their help. In addition to NSF CISE Expeditions Award CCF-1730628, this research is supported by gifts from Amazon Web Services, Ant Financial, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, NVIDIA, Scotiabank, Splunk and VMware.

References

Jacob Andreas. 2019. Measuring compositionality in representation learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

John Bauer. 2014. The stanford shift-reduce constituency parser.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Noam Chomsky and David W Lightfoot. 2002. *Syntactic structures*. Walter de Gruyter.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Federico Fancellu, Sorcha Gilroy, Adam Lopez, and Mirella Lapata. 2019. Semantic graph parsing with recurrent neural network DAG grammars. 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)*, pages 2769–2778, Hong Kong, China. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *ArXiv preprint*, abs/2007.08970.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020. Hierarchical poset decoding for compositional generalization in language. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv*.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Sonja Lang. 2014. *Toki Pona : the language of good*. Sonja Lang, United States.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *ACL*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4249–4257, Florence, Italy. Association for Computational Linguistics.

Richard Montague. 1970. Universal grammar. *1974*.

Maxwell Nye, Armando Solar-Lezama, Joshua Tenenbaum, and Brenden Lake. 2020. Learning compositional rules via neural program synthesis. *ArXiv preprint*, abs/2003.05562.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

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

David R. So, Quoc V. Le, and Chen Liang. 2019. The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.