Compositional Generalization with Grounded Language Models

Sondre Wold, Étienne Simon, Lucas Georges Gabriel Charpentier, Egor V. Kostylev, Erik Velldal, Lilja Øvrelid
University of Oslo

Abstract

Grounded language models use external sources of information, such as knowledge graphs, to meet some of the general challenges associated with pre-training. By extending previous work on compositional generalization in semantic parsing, we allow for a controlled evaluation of the degree to which these models learn and generalize from patterns in knowledge graphs. We develop a procedure for generating natural language questions paired with knowledge graphs that targets different aspects of compositionality and further avoids grounding the language models in information already encoded implicitly in their weights. We evaluate existing methods for combining language models with knowledge graphs and find them to struggle with generalization to sequences of unseen lengths and to novel combinations of seen base components. While our experimental results provide some insight into the expressive power of these models, we hope our work and released datasets motivate future research on how to better combine language models with structured knowledge representations.

1 Introduction

Language models (LMs) acquire latent representations of text through their pre-training. One drawback of this paradigm is the implicit encoding of knowledge in a large parameter space, which can lead to factual hallucinations in model outputs. To reduce this effect, there has recently been a widespread interest in what is sometimes referred to as *grounded language models*, which are LMs grounded in external information sources, such as knowledge graphs (KGs). These approaches typically attempt to enable reasoning over KGs by combining the LM outputs with that of a separate graph encoder, such as a graph neural network (GNN; Lin et al., 2019; Zhang et al., 2022; Sun et al., 2022; Yasunaga et al., 2022; Yu et al., 2022). It can be difficult to determine to what extent grounded models exhibit any structured reasoning, or what precisely would constitute reasoning in this context, as the information contained in these KGs often targets the same domain as the corpora used for LM pre-training. One example of this is the overlap in information between Wikidata, often used as the KG in previous work on grounded LMs, and Wikipedia, which is a common corpus for pre-training.

Central to human reasoning is the ability to form novel combinations from seen components, often referred to as compositional generalization. Studies on compositionality have a long tradition within many fields of study, such as linguistics (Partee et al., 1995), philosophy (Pagin and Westerståhl, 2010), and cognitive science (Churchland, 1990), and determining to what degree neural networks capture compositionality is a long-standing research problem in machine learning, with debates going back several decades (Fodor and Pylyshyn, 1988; Smolensky, 1987). Recent studies in NLP have primarily used semantic parsing of synthetic utterances as a testbed for investigating to what extent neural networks combine syntactic elements to form new complex expressions (Hosseini et al., 2022). One of the most well-known examples of this is from Lake and Baroni (2018), who found *seq2seq* models to struggle both with systematic compositional skills and generalization to longer sequences than seen during training – abilities that are often taken to be central to compositional generalization. Some of these shortcomings were recently mitigated using a meta-learning framework (Lake and Baroni, 2023), but it is still not clear under what circumstances neural networks can consistently demonstrate compositional generalization, or if they can at all (Dziri et al., 2024).

In this work, we investigate to what extent grounded language models exhibit compositional generalization by combining two orthogonal research directions: compositionality in neural networks and the grounding of LMs in KGs.

Using question answering as a task, we train models to recognize the relation between certain patterns in a synthetic KG and natural language questions. We develop a data generation procedure that targets three aspects of compositionality from Hupkes et al. (2020): *substitutivity*, *productivity*, and *systematicity*. In our experiments, we evaluate LMs that interact with KGs through the use of a GNN, an approach previously proven effective for tasks such as question answering (Lin et al., 2019; Zhang et al., 2022; Sun et al., 2022) and pre-training (Yasunaga et al., 2022).

Our work targets two key limitations from the literature: Firstly, by extending existing work on compositional generalization to question answering over KGs, we can assess the capabilities of these models for structured reasoning over natural language questions of varying complexities; and secondly, by using a synthetic knowledge graph we do not run the risk of grounding a language model in a knowledge source containing information that already exists implicitly in the models' weights. The contribution of our work is three-fold: *i)* We provide the first experimental study on the expressive power of grounded LMs with respect to compositional generalization *ii)* We develop a procedure for generating dataset samples that target three theoretically motivated aspects of compositionality, and *iii)* We release a series of generated datasets using this procedure, together with all code and data related to our experiments, which can be used to benchmark grounded language models in a controlled environment.[1]

2 Compositionality

In what follows we discuss three aspects of compositionality that have been established as tests of compositionality in previous work (Hupkes et al., 2020): *substitutivity*, *systematicity*, and *productivity*. We also describe how we ground these notions to be applicable for our setting, where data consists of both text and KGs. We refer an interested reader to the overview by Szabó (2022) for a more extensive introduction to these terms.

Substitutivity One of the most well-known definitions of compositionality is the *principle of compositionality* from Partee et al. (1995): "The meaning of a compound expression is a function of the meanings of its parts and of the way they are syntactically combined". Closely related to this principle is the notion of *substitutivity*, which states that any change to a complex expression that maintains the meaning of individual parts also maintains the meaning of the whole expression. On the one hand, in the context of natural languages, this definition is not without controversy. Languages often have elements that do not adhere strictly to such a principle, for example through the use of idioms (Dankers et al., 2022a). On the other hand, in our setting, we can exploit the observation that if an atomic expression in natural language has a node in a graph as its reference, then we can substitute this reference with a compounded expression referencing the same node without changing the meaning of the whole expression. As the principle of compositionality is not committed to any specific theory of semantics, we can use *reference* as the qualification for semantic equivalence, a theory sometimes referred to as the *compositionality of reference* (Szabó, 2022). For example, given the KG in Figure 1, we can ask ourselves whether "Paper2 was authored by Professor2". We can then replace "Professor2" with a compounded reference, such as "the supervisor of GraduateStudent5", which gives rise to the question: "Was Paper2 authored by the supervisor of GraduateStudent5?" Following the compositionality of reference, we do not change the meaning of the overall question, since we still enquire about the same node in the graph, even though we have substituted a 1-hop reference with a 2-hop reference. Consequently, by constructing highly compositional questions in natural language, we can test whether or not neural networks can learn the mapping between constituents in text and paths in a graph.

Productivity Closely related to substitutivity, *productivity* refers to the ability to construct seemingly infinite constructions using a finite capacity. As Hupkes et al. (2020) point out, this view on the open-endedness of language is typically associated with generative linguistics, and in the context of neural networks, the most direct way of testing for this ability is to test models on generalization to sequences of lengths not seen during training. In our case, we can train models on questions that require reasoning over k-hop paths, for several k, in a graph, and then test on j-hop paths, for various j different from the k's used in training.

[1] https://github.com/ltgoslo/text-graph-generalization

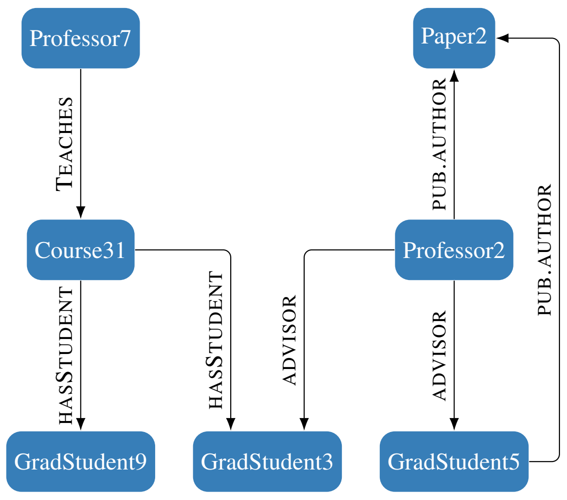

Figure 1: An example KG illustrating the relationship between entities from the LUBM inventory.

Systematicity One of the most tested properties of compositionality is *systematicity*, which was popularized as a term by Fodor and Pylyshyn (1988). Systematicity can loosely be described as the ability to combine seen parts and rules to form new combinations, an ability that has a long tradition within linguistics. In the context of neural networks, tests of systematicity typically involve exposing models to a finite set of items and combinations during training and then evaluating the same models on a test set containing the same set of items, but combined in ways not seen during training. Lake and Baroni (2018) use the example of a person that knows the meaning and usage of words such as "twice," "and," and "again," which upon learning the new verb "to dax" can generalize to new constructions such as "dax twice and then dax again," something they find neural networks to struggle with. Recently, there have also been some attempts to formalize and quantify systematic generalization in neural networks (Keysers et al., 2020; Ram et al., 2023). In our setting, we can use the typed relations of the edges in a graph, such as the one illustrated in Figure 1, to test for systematicity. If a model has seen the relations TEACHES, HASSTUDENT, and ADVISOR in numerous 3-hop paths, but never in this order, a display of systematicity would require generalization to this unseen order.

3 Dataset Generation

3.1 Overview

To target the compositional abilities specified above, we generate several datasets consisting of pairs of KGs and yes/no questions in natural language, all of which are about the information in these graphs. This section explains the properties of these datasets and how they are created.

A KG over a set $\mathcal{T}$ of *types* and a set $\mathcal{R}$ of *relations* is a directed labeled multi-graph $(\mathcal{V}, \mathcal{L}, \mathcal{E})$ with nodes $\mathcal{V}$, called *entities* in this context, set $\mathcal{L}$ of node labelings, each of which is of the form (v, t) where v is an entity in $\mathcal{V}$ and t a type in $\mathcal{T}$, and set $\mathcal{E}$ of edges, each of which is of the form (v, r, v'), where v and v' are entities from $\mathcal{V}$, and r is a relation from $\mathcal{R}$. In this work, we only concentrate on graphs from the Lehigh University Benchmark (LUBM; Guo et al., 2005), which is a synthetic KG benchmark popular for evaluating various aspects of knowledge graph repositories. In particular, we exploit LUBM's KG generator of customizable and repeatable synthetic data, which can produce KGs of a user-specified size describing a fictitious university domain and committing to a realistic ontology over sets $\mathcal{T}_{\mathsf{LUBM}}$ and $\mathcal{R}_{\mathsf{LUBM}}$ of university-related types and relations. We generate several graphs with an average number of 685 entities, each labeled by a single type, and 4 949 edges, each labeled by a single relation.[2]

Each of the natural language questions targets two entities in the corresponding generated LUBM graph $\mathcal{G}$ and a chain of relations from $\mathcal{R}_{\mathsf{LUBM}}$ and inverses of such relations that is to hold between these entities. In particular, for a number of hops (i.e., chain length) k, the questions are semi-manually created by first mapping each unique k-hop combination (i.e., chain of length k) of relations from a selected subset of $\mathcal{R}_{\mathsf{LUBM}}$ and their inverses to a natural language template with two parameters, and then substituting these parameters by two entities from $\mathcal{V}$ as arguments. As a result, the answer to the question is TRUE if there exists a path between the two entities in $\mathcal{G}$ as prompted by the k-hop combination of the question template, and FALSE otherwise. We create templates for 2, 3, and 4 hops, with 5 relations from $\mathcal{R}_{\mathsf{LUBM}}$ and their inverses as $\mathcal{R}'_{\mathsf{LUBM}}$. This makes for over 280 unique templates (while the number of questions depends on $\mathcal{G}$). To illustrate our dataset creation we concentrate on an example and a discussion of a key step, sampling of positive and negative question–graph pairs, in the next two subsections. We refer to Appendix A for further details.

[2]The ontology is specified in Web Ontology Language OWL (`https://www.w3.org/OWL/`).

3.2 Example

Given the KG in Figure 1, we can create the following templates, for an increasing number of hops, where the "−" superscript denotes the inverse of the relation:

2 hops: [TEACHES, TAKESCOURSE$^-$] $\rightarrow$ Does s teach a course that has a student named e?

3 hops: [TEACHES, TAKESCOURSE$^-$, ADVISOR] $\rightarrow$ Does s teach a course that has a student supervised by e?

4 hops: [TEACHES, TAKESCOURSE$^-$, ADVISOR, PUBLICATIONAUTHOR] $\rightarrow$ Does s teach a course that has a student supervised by an author of e?

To create questions from these templates that we label TRUE we can use PROFESSOR7 and GRADSTUDENT9 as arguments substituting parameters s and e, respectively, in the 2-hop case, PROFESSOR7 and PROFESSOR2 for the 3-hop case, and PROFESSOR7 and PAPER2 for the 4-hop case. Creating examples labeled as FALSE for these templates then involves sampling two entities to use as s and e that are *not* connected by the chains of relations, but are similarly connected in the graph as the entities in the positive examples. We describe this negative sampling in the next section.

3.3 Sampling Procedure

As mentioned, we create each question–graph pair by sampling pairs of entities and the relations on the edges that are to hold between them in the graph $\mathcal{G}$. The key requirement to this procedure is to guarantee that the k relations and their inverses in every sampled k-hop combination are as likely to be labeled TRUE as they are to be labeled FALSE; failing to do so introduces distributional bias. This means that our sampling procedure must ensure two important properties: *i)* it must be impossible for a text-only model to associate a question template (i.e., all questions produced from this template) to the TRUE or FALSE label, and *ii)* it must be impossible for a graph-only model to associate the presence of a specific set of relations in graph $\mathcal{G}$ with a specific label.

We ensure property *i)* by creating the same number of questions labeled TRUE as FALSE for every unique k-hop combination of relations. We ensure *ii)* by sampling k-hop relation paths from $\mathcal{G}$ symmetrically: We first sample i.i.d. two pairs of different entities, s_+, e_+ and s_-, e_-, such that s_+ and s_- are of the same entity type, and e_+ and e_- are also of the same entity type. We then identify a k-hop combination over $\mathcal{R}'_{\mathsf{LUBM}}$ that exists between s_+ and e_+ but not between s_- and e_-, as well as verify that there exists another (different) k-hop combination over $\mathcal{R}'_{\mathsf{LUBM}}$ between s_- and e_-. If all this succeeds, the pair s_+, e_+ with the question corresponding to the template becomes a positive example, and the pair s_-, e_- with the same template a negative one. Further details of the sampling procedure can be found in Appendix A.

4 Modeling

For all experiments, each data point includes a natural language question q and a graph $\mathcal{G}$. Following previous work on combining language models with KGs (Lin et al., 2019; Zhang et al., 2020; Yasunaga et al., 2021; Zhang et al., 2022; Kaur et al., 2022; Sun et al., 2022; Yasunaga et al., 2022), we use separate encoders for q and $\mathcal{G}$ to obtain representations of both modalities. Section 4.1 outlines these encoders and Section 4.2 how we combine them to produce the final output prediction $\hat{y} \in \{\text{TRUE}, \text{FALSE}\}$.

4.1 Encoders

Text encoder We obtain representations of q by using the special sentence representation token at the last layer of a pre-trained language model, as per Devlin et al. (2019): $z_{\text{text}} = \text{BERT}(q)$.

Graph encoder As a graph encoder, we use the relational graph convolution model, R-GCN, from Schlichtkrull et al. (2018):

$$h_i^{\ell+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{\mid\mathcal{N}_i^r\mid}W_r^\ell h_j^\ell + W_0^\ell h_i^\ell\right),$$

where h_i^ℓ is the hidden state of a node i in layer ℓ, σ is the ReLU function and $\mathcal{N}_i^r$ is the set of neighborhood vertices of node i under relation r.

4.2 Final models

Inspired by previous work on combining text and graph encoders, we experiment with different approaches to integrating the two modalities and apply them using the above-mentioned encoders and our experimental setup. Figure 2 provides an overview of the differences between these approaches.

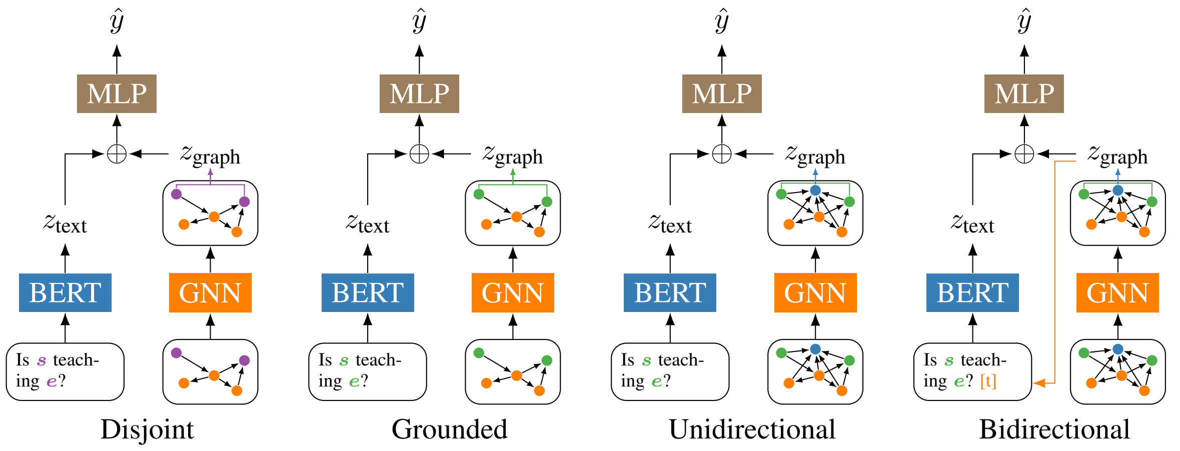

Figure 2: Overview of the different approaches to combining the two encoders. Purple nodes are initialized at random while green and blue nodes are initialized using a frozen BERT model.

Disjoint The simplest way of having the two modalities interact is to initialize and run both encoders separately before conjoining their outputs in a final classification layer. This keeps the encoders disjoint, essentially making a "two-tower architecture" – conceptually similar to approaches such as Wang et al. (2019).

To get a final representation of the graph, z_{graph}, we take the embedding of the head and tail node targeted by the question after L rounds of message passing: $z_{\text{graph}} = h_s^L \oplus h_e^L$, where s and e are the indices of the head and tail node and $\oplus$ is vector concatenation. We initialize these nodes by creating two distinct embeddings, E_s and E_e, with $E_s, E_e \sim \mathcal{N}(0, 1)$. We keep these static throughout training and testing and use them for all head and tail nodes targeted by the questions. The initialization of remaining nodes in each graph is discussed in more detail in Section 4.3.

For the question, we take the output of the text encoder without any modifications. The final output prediction is then obtained by the following: $f(z_{\text{text}} \oplus z_{\text{graph}})$, where f is a two-layered MLP with a ReLU activation function.

Grounded For this model, we concatenate the original question string from q with both the head and tail entity strings, respectively, using a separation token. This is then used as input to the text encoder, obtaining a representation of the head and tail nodes:

$$h_s^0 = \text{BERT}(q \texttt{ <SEP> } \text{head surface form}),$$
$$h_e^0 = \text{BERT}(q \texttt{ <SEP> } \text{tail surface form}).$$

These are then used to initialize the embeddings for the nodes pertaining to these entities in the graph. This is done in a pre-processing step before the text encoder is fine-tuned on our datasets. The final output prediction is calculated in the same way as with the disjoint model.

Unidirectional In this model, we add a unidirectional interaction between the encoders, inspired by works such as Yasunaga et al. (2021). For each question–graph pair, $(q, \mathcal{G})$, we introduce a new special context node, u, and connect u to each node in $\mathcal{G}$. This special node is connected using two new relations, one for each direction, and its embedding is initialized using the text encoder representation of the question, $\text{BERT}(q)$. In Figure 2, u is marked as a blue node. For each $\mathcal{G}$, we perform global pooling over the hidden states:

$$z_{\text{graph}} = \frac{1}{|\mathcal{V}'|} \sum_{v \in \mathcal{V}'} h_v^L,$$

where $\mathcal{V}'$ is the set containing the head, tail, and special context nodes. We do not consider the set of all nodes due to the zero-initialization of the remaining nodes in $\mathcal{G}$ (see Section 4.3). The head and tail nodes are initialized as in the grounded model, using the text encoder representations. We also take the embedding of the special node at the last layer, h_u^L, as a representation of $\mathcal{G}$. The final output prediction is then given by: $f(z_{\text{text}} \oplus z_{\text{graph}} \oplus h_u^L)$, where f is a two-layered MLP with a ReLU activation function.

Bidirectional We also develop a model inspired by the interaction approach used in works such as

Zhang et al. (2022) and Yasunaga et al. (2022) to allow for a bidirectional interaction between the two encoders. We keep the special context node u as described in the unidirectional model, but we add a special interaction token, t, to the question q. t is set to be the representation of the special context node u after L layers of message passing: $t = h_u^L$. This is done after the embedding of q in the text encoder. We produce the final output prediction using the same method as in the unidirectional model.

Baseline We use a sole graph encoder as a baseline. Theoretically, it should not be possible to solve the tasks without having access to both the text and the graph. By using a graph-only system as a baseline we therefore get both a sanity check and an upper bound on the noise produced by the sampling. As the natural language questions are not sampled, but rather constructed deterministically from the paths sampled from our graphs, one positive and one negative, the performance of a text-only encoder will by design be equal to a random classifier. The output of the baseline is calculated the same way as for the disjoint model, taking the representations of the head and tail entity.

4.3 Node initialization

Except for the head and tail nodes targeted by each question, and the special context node u, all nodes are initialized as zero-vectors. This means that we leak the position of the head and tail entity in $\mathcal{G}$ for each sample in the dataset, making any pruning of $\mathcal{G}$ redundant. It is often necessary to reduce the computational complexity of combining language models and large knowledge graphs using mechanisms such as subgraph retrieval and entity linking, but these steps are also known to be error-prone (Wold et al., 2023). This initialization ensures that we further isolate the application of compositional generalization rules, as opposed to evaluating the performance of a pipeline model with numerous error-prone components.

5 Experiments and results

Hops	**#relation-paths**	**N**
2	23	$1043^{\pm 1941}$
3	73	$328^{\pm 449}$
4	188	$127^{\pm 260}$

Table 1: The mean number of occurrences of the relation paths across 24 000 generated graphs per hop k.

In this section, we test our suite of models on data specifically targeting substitutivity (Section 5.1), productivity (Section 5.2), and systematicity (Section 5.3), going test by test. For all experiments, the distribution between the two classes is perfectly balanced. Statistics on the diversity of the generated graphs with respect to relation paths can be found in Table 1. Details on the selection of hyperparameters can be found in Appendix B.

5.1 Substitutivity

In our first test, we evaluate to what extent the five model configurations can answer questions targeting substitutivity, as defined in Section 2. Success at this task requires recognizing the mapping between the presence of certain relations in the graph and phrases in the natural language questions. Substitutivity is the most basic setting of our experiments and the results on this task demonstrate the general difficulty of the task.

Test design We generate 12 000 question–graph pairs using our generation procedure. We do this for 2, 3 and 4-hops. Each experiment targets one of the hop lengths, i.e., both training and evaluating on the same number of hops. For all experiments, we use 10 000 samples for training and 2 000 for testing. All results are reported using mean accuracy and standard deviation from five randomly seeded runs.

Results The results of the substitutivity tests can be seen in Table 2. There is a strong generalization in the 2-hop case for all models. In the 3-hop case, performance drops significantly and the variance also increases for most models. There is a further loss in generalization when moving to the 4-hop case, where there is little difference between the models. Figure 3 shows that there is a positive correlation between test accuracy on a specific question type and the frequency of the corresponding relation path in the training data for the 3-hop case. For the 2-hop and 4-hop cases, where the performance is closer to perfect and random classification, respectively, the relationship flattens. This aligns with the increase in unique combinations of relation paths as the number of hops increases, as shown in Table 1. For 2-hops there is either a high number of examples per relation path combination, which enables learning, or very few, which makes the performance on these relation paths negligible on the overall accuracy. For 4-hops, on the other hand, there is too much of a long tail on the dis-

Model	2-hop	3-hop	4-hop
Baseline	$48.88^{\pm1.00}$	$49.8^{\pm0.64}$	$49.36^{\pm0.44}$
Disjoint	$98.17^{\pm0.39}$	$83.08^{\pm0.96}$	$67.86^{\pm1.89}$
Grounded	$92.46^{\pm0.49}$	$84.91^{\pm0.38}$	$69.32^{\pm1.96}$
Unidirectional	$91.78^{\pm0.72}$	$70.88^{\pm3.53}$	$62.0^{\pm3.97}$
Bidirectional	$92.44^{\pm0.23}$	$73.66^{\pm1.04}$	$63.24^{\pm3.36}$

Table 2: Substitutivity results.

tribution of number of samples per combination, making learning difficult. For 3-hops, however, there is a balance, which is reflected in the results. See Figure 4 in Appendix A for an illustration of the relative percentage of each unique combination to the total for each hop.

These results show that the models struggle to generalize to the long tail of the training distribution, which is a known attribute of neural networks. However, single relation types are reused extensively throughout our training data, as there are only 10 unique relations of length 1. This means that the distribution only has a long tail if one considers each 4-hop combination to be a single unique type, as opposed to being composed of four individual relations that are individually high-frequent. We argue that these results indicate that these models do not create internal representations of each 1-hop path together with a function for composing them into compounded expressions, but rather learn individual representations for each compound. This also explains why the performance drops as the complexity increases and the number of samples per relation decreases, making such a strategy less viable. This explanation is further explored in the systematicity tests. We also note that there is little difference between the models, indicating that an increase in model complexity does not add any benefits to this task.

5.2 Productivity

In our next test, we evaluate to what extent models can answer k-hop questions when trained on j-hops, with $k \neq j$. We define experiments requiring *extrapolation* and *interpolation* for different values of k and j. Success at this task requires generalization to sequences with a different length than those seen during training.

Test design In our first experiment, we use an even amount of 2 and 3-hop questions for training while using 4-hop questions for testing, targeting extrapolation. In our second experiment, we target interpolation by training on an even amount of 2 and 4-hop questions while testing on 3-hops. For our last experiment, we target extrapolation to lower values, using a training split with an even amount of 3 and 4-hop questions while testing on 2-hops. For all three experiments, we generate 10 000 samples for training and 2 000 for testing.

Experiment	Acc.
TRAIN: 2 AND 3-HOP. TEST: 4-HOP	
Baseline	$49.92^{\pm0.82}$
Disjoint	$59.42^{\pm2.07}$
Grounded	$61.79^{\pm0.52}$
Unidirectional	$62.40^{\pm1.02}$
Bidirectional	$62.05^{\pm0.91}$
TRAIN: 2 AND 4-HOP. TEST: 3-HOP	
Baseline	$51.19^{\pm0.31}$
Disjoint	$63.87^{\pm2.24}$
Grounded	$66.10^{\pm2.02}$
Unidirectional	$66.65^{\pm2.24}$
Bidirectional	$66.29^{\pm1.25}$
TRAIN: 3 AND 4-HOP. TEST: 2-HOP	
Baseline	$50.98^{\pm0.25}$
Disjoint	$60.34^{\pm3.25}$
Grounded	$62.32^{\pm1.98}$
Unidirectional	$64.04^{\pm4.47}$
Bidirectional	$63.03^{\pm3.76}$

Table 3: Productivity results.

Results The results of the productivity tests can be seen in Table 3. In general, we observe that all models struggle to extrapolate to both shorter and longer hops. We observe some generalization for the interpolation test, where the model has seen sequences both shorter and longer during training, but the overall absolute performance is still poor. This aligns with previous work, such as Lake and Baroni (2018) and Hupkes et al. (2020). We observe a tiny increase in performance with the more complex models compared to the simpler, disjoint model, but given the overall low scores and the variance, we do not think these results are significant. This indicates that adding a more complex interaction between the encoders does not provide any added benefit for extrapolation or interpolation.

5.3 Systematicity

In our last test, we evaluate to what extent our models can generalize systematically. This means that all unique 1-hop relations are seen during training in some k-hop combination, but in the test data, we only include unseen combinations. For example, during training the models might see samples using the relations TAKESCOURSE and

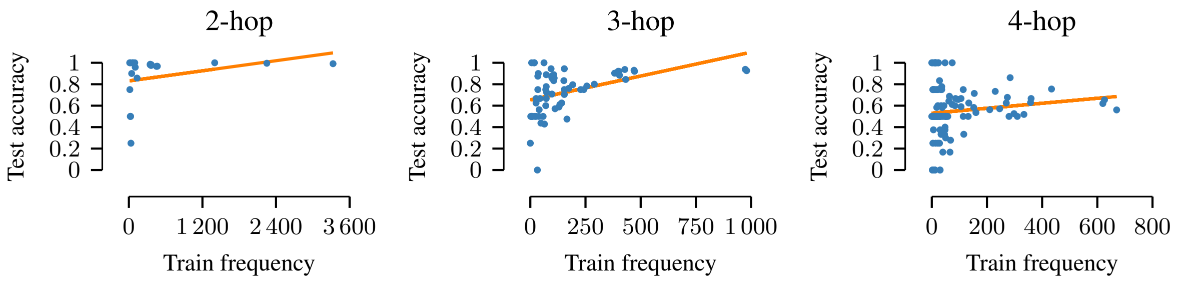

Figure 3: Relationship between frequency of a k-hop relation and the performance of the Disjoint model on these during testing. Each blue point corresponds to a tuple of k relations for $k = 1, 2, 3$. The orange lines are the linear regressions for each setting showing positive correlations between frequencies of k-hop relations and performances of the Disjoint model.

HASTEACHINGASSISTANT, and the 2-hop combination [TAKESCOURSE, HASTEACHINGASSISTANT], but in the test we target [HASTEACHINGASSISTANT, TAKESCOURSE]. Success at this task requires the model to learn that a 3-hop question, for example, consists of three individual parts that are combined, as opposed to encoding the entire sequence as one entity.

Test design With k being 2, 3, or 4, we generate question–graph pairs with length k. Then, for each k, we randomly select a fraction of the available k-length combinations of relations as our test combinations. We then separate the generated pairs into two parts: those who use the test combinations and those who do not. The former is saved as the test set. We note that the use of test combinations is not limited to the relations on the path between the head and tail entities targeted by the question, but *any* k-hop path that exists between these entities. For the training set, we split the remaining possible combinations into five chunks and collect all samples that use these combinations. This leaves us with five different folds of the training set. Models are then trained on each fold independently, but evaluated on the same test set. We do this to control for the fact that some combinations of relations might have a similar distribution of samples as the relations picked out for the test set, making them better fits for that particular split.

As this tests out-of-distribution generalization, we argue that folding is better suited for finding the true performance and variance of the models than the more straightforward evaluation method used for our substitutivity and productivity experiments. The results are reported as the mean accuracy over all five folds.

Results The results of the systematicity tests can be seen in Table 4. As with the productivity tests, we see that there is no consistent display of generalization for any of the models across hops. We see, however, that for the 3-hops setting the two simpler models achieve some success on the task. As these models had the highest performance on the substitutivity test, we take this to indicate that additional architectural complexity does not provide any performance benefit for this type of generalization. We also note that it is possible that these experiments are sensible to the hyperparameters, and that the two simpler models happened to have a better fit for the 3-hop setting. We did not do a specific hyperparameter search per model for the systematicity experiments but relied on those from the initial hyperparameter search.

Overall, these results might come from undersampling, or it might be the case, as is reported in Hupkes et al. (2020), that the models are not able to construct useful representations of smaller syntactic units. We argue that our results further support the intuition from the substitutivity experiments: the models do not learn to create representations that correspond to each 1-hop relation, together with a function for combining them, but rather rely on memorization of whole combinations of relation types. For short-tailed distributions and some settings of hyperparameters this works, but it is not a robust solution to the general problem of matching the text questions to certain patterns in the graphs, illustrated by the significant drop in performance in these results.

We also see adjacent work on assessing compositional generalizations in LLMs to provide possible explanations for our results. Dziri et al. (2024) hypothesize that transformer-based models try to solve a highly compositional problem, like ours, by

Model	**2-hop**	**3-hop**	**4-hop**
Baseline	$49.50^{\pm 0.24}$	$49.40^{\pm 0.33}$	$49.92^{\pm 1.01}$
Disjoint	$53.62^{\pm 8.39}$	$74.05^{\pm 3.42}$	$62.18^{\pm 1.20}$
Grounded	$60.02^{\pm 3.22}$	$75.68^{\pm 3.27}$	$64.96^{\pm 1.39}$
Unidir.	$64.24^{\pm 0.75}$	$57.12^{\pm 1.81}$	$64.10^{\pm 3.62}$
Bidir.	$64.15^{\pm 1.10}$	$60.37^{\pm 2.31}$	$63.72^{\pm 3.63}$

Table 4: Systematicity results.

reducing it into linearized path matching. If this is the case, models do not learn the compositional rule needed to solve the task. If the samples from the test distribution only resemble the training distribution *superficially*, this pattern matching approach is not a viable solution for out-of-distribution generalization in both our productivity and systematicity experiments. In these experiments, the mapping from the input samples to output labels cannot be based on surface-form frequency, due to our balanced sampling, which could then explain the collapse in performance we are observing for these experiments.

6 Previous work

There have been numerous works on compositional generalization in neural networks (Andreas, 2019; Lake and Baroni, 2018; Kim and Linzen, 2020; Hupkes et al., 2020; Dankers et al., 2022b; Lake and Baroni, 2023; Dziri et al., 2024). Most similar to our work is Gu et al. (2021), which develops a large-scale dataset targeting aspects of compositional generalization using question answering over Wikidata. Both this work and similar efforts reformulate the problem of integrating the KG to a seq2seq task where the questions are first translated into an intermediate logical form, such as SPARQL, which can be executed over the KG (Shu et al., 2022). Under the same framework, Ravishankar et al. (2022) explicitly tests generalization to unseen combinations of relations in a similar fashion to our systematicity test. They too find that the performance tends to drop significantly for this setting.

The mapping from natural language to logical form can also use techniques from semantic parsing, a topic where compositionality is of special interest (Lindemann et al., 2023). Our work, however, is the first to target compositionality in a setup where the models learn representations of both text and graphs *directly*, without any logical form as an intermediate step. In contrast to previous work on grounded language models (Lin et al., 2019; Zhang et al., 2020; Yasunaga et al., 2021; Zhang et al., 2022; Kaur et al., 2022; Sun et al., 2022), our methods can also more precisely quantify the effect of combining LMs with GNNs, as our sampling procedure ensures that the encoders can not achieve above random performance on their own, a property which we believe is unique to our approach.

7 Conclusion

Research on compositional generalization in neural networks has a long history. To the best of our knowledge, we provide the first experimental evaluation of compositional generalization in grounded LMs, using question answering over KGs as a task. Our results indicate that these models *i)* require large sample sizes and quickly lose generalization power as the complexity of the compositional structure increases, *ii)* struggle with extrapolation and interpolation to sequences of lengths not seen during training, and *iii)* cannot systematically generalize from seen base components to novel compositions given the sample sizes used in our experiments. We show this across a set of model architectures inspired by the literature. Our results are in line with previous work on compositional generalization in other domains and for other types of neural architectures. Overall, we believe our results show that grounded LMs do not display signs of structured reasoning over KGs and that there is still much to be done on the improvement of models that try to exploit external information stored as symbolic objects. This is the case not only for knowledge graphs but also for other types of data representations, such as tabular data. We hope our generation procedure and released datasets can help motivate future efforts at overcoming these challenges.

Limitations

Models Our models are inspired by previous work. However, since our experimental setup differs from the original settings these models were implemented to work in, we had to make multiple simplifications. Consequently, we are not able to conclude anything about the abilities of the *specific* models from the literature, but rather about the general interaction approach they propose.

Question variety The questions in our dataset are formulated in English, and although they have more linguistic variety than similar studies from semantic parsing, such as Lake and Baroni (2018) and Hupkes et al. (2020), we still use a rather limited version of the language, which is by design devoid of any metaphors, idioms, analogies and other constructs that would make compositionality more difficult to measure. It is not given that our results transfer to real language data.

Acknowledgments

We acknowledge Sigma2, Norway, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium.

References

Jacob Andreas. 2019. Measuring compositionality in representation learning. In *International Conference on Learning Representations*.

Paul M. Churchland. 1990. Cognitive activity in artificial neural netowrks. *An invitation to cognitive science: Thinking*, 3:199–229.

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

Verna Dankers, Christopher Lucas, and Ivan Titov. 2022b. Can transformer be too compositional? analysing idiom processing in neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3608–3626, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

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

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. Lubm: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182.

Arian Hosseini, Ankit Vani, Dzmitry Bahdanau, Alessandro Sordoni, and Aaron Courville. 2022. On the compositional generalization gap of in-context learning. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 272–280, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Jivat Kaur, Sumit Bhatia, Milan Aggarwal, Rachit Bansal, and Balaji Krishnamurthy. 2022. LM-CORE: Language models with contextually relevant external knowledge. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 750–769, Seattle, United States. Association for Computational Linguistics.

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 *International Conference on Learning Representations*.

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

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks.

Brenden M. Lake and Marco Baroni. 2023. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. 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 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2023. Compositional generalisation with structured reordering and fertility layers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2172–2186, Dubrovnik, Croatia. Association for Computational Linguistics.

Peter Pagin and Dag Westerståhl. 2010. Compositionality i: Definitions and variants. *Philosophy Compass*, 5(3):250–264.

Barbara Partee et al. 1995. Lexical semantics and compositionality. *An invitation to cognitive science: Language*, 1:311–360.

Parikshit Ram, Tim Klinger, and Alexander G. Gray. 2023. How compositional is a model? In *International Joint Conference on Artificial Intelligence 2023 Workshop on Knowledge-Based Compositional Generalization*.

Srinivas Ravishankar, Dung Thai, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Tahira Naseem, Pavan Kapanipathi, Gaetano Rossiello, and Achille Fokoue. 2022. A two-stage approach towards generalization in knowledge base question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5571–5580, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Paul Smolensky. 1987. Connectionist ai, symbolic ai, and the brain. *Artificial Intelligence Review*, 1(2):95–109.

Yueqing Sun, Qi Shi, Le Qi, and Yu Zhang. 2022. JointLK: Joint reasoning with language models and knowledge graphs for commonsense question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5049–5060, Seattle, United States. Association for Computational Linguistics.

Zoltán Gendler Szabó. 2022. Compositionality. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Fall 2022 edition. Metaphysics Research Lab, Stanford University.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215.

Sondre Wold, Lilja Øvrelid, and Erik Velldal. 2023. Text-to-KG alignment: Comparing current methods on classification tasks. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 1–13, Toronto, ON, Canada. Association for Computational Linguistics.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *Advances in Neural Information Processing Systems*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2022. Jaket: Joint pre-training of knowledge graph and language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11630–11638.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2031–2043, Online. Association for Computational Linguistics.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations*.

A Dataset generation

A.1 LUBM

We use the LUBM UBA to generate knowledge graphs on the fly.[3] We limit our graphs to the following entity types: *Professor*, *AssociateProfessor*, *UndergraduateStudent*, *GraduateStudent*, *TeachingAssistant*, *Publication*, *Course*; and the following relations: PUBLICATIONAUTHOR, TEACHEROF, ADVISOR, TAKESCOURSE, TEACHINGASSISTANTOF. We also add reverse edges with reverse relations for all of these to create more diverse question types in our final dataset, making $|\mathcal{R}'_{LUBM}| = 10$. We remove all literals. Using a sample size of approximately $14\,000$, we create graphs that have $\mu = 685, \sigma = 77.16$ nodes and $\mu = 4949, \sigma = 532$ edges after processing.

A.2 Statistics

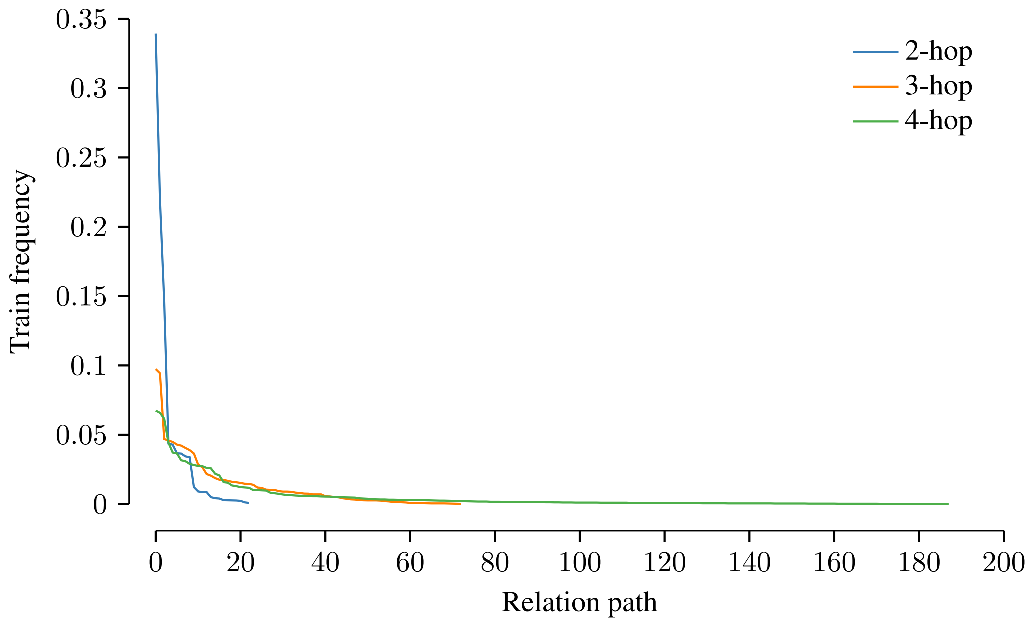

Figure 4: The relative percentage of each unique relation path to the total for 2, 3, and 4-hops.

The number of possible combinations of relations on the edges in a k-hop path increases with the number of hops. As our sampling procedure uses 10 distinct relations we have 10^k theoretical combinations of relations for any k-hop path. As our graphs are committed to the LUBM ontology the number of relations that hold between two entity types is much lower.

[3]https://swat.cse.lehigh.edu/projects/lubm/

A.3 Sampling procedure

Given a directed multigraph $G = (\mathcal{V}, \mathcal{E})$ with labeled entities $v_i \in \mathcal{V}$ and labeled edges $(v_i, r, v_j) \in \mathcal{E}$ where $r \in \mathcal{R}$ is a relation, we sample a positive and negative example for a k-hop path using the following procedure:

1. Sample two pairs of nodes (s_+ and e_+) and (s_- and e_-) from $\mathcal{V}$ with uniform probability, making sure that $s_+ \neq s_-$ or $e_+ \neq e_-$, that s_+ and s_- are of the same entity type, and that e_+ and e_- are of the same entity type.
2. Let R_+ be the set of all relation paths of length k from s_+ to e_+
3. Let R_- be the set of all relation paths of length k from s_- to e_-
4. Let P_+ be the set $R_+ \setminus R_-$
5. Let P_- be the set $R_- \setminus R_+$
6. If $P_+ \neq \emptyset$ and $P_- \neq \emptyset$:
   (a) Sample a relation path q: $q \sim \mathcal{U}(P_+)$
   (b) Return the triple (q, s_+, e_+) as a positive example and (q, s_-, e_-) as a negative example.

This procedure returns a k-hop relation path, q, that holds between s_+ and e_+ but not between s_- and e_-. However, we also make sure that there are paths that hold between s_- and e_- but not s_+ and e_+, ensuring symmetry.

In addition, we shuffle entities labels while preserving entity types before running this procedure since the LUBM generator introduces problematic patterns with its entity numbering. For example, in the original graph, Professor1 is more likely to author Paper1 than Paper5. By randomly exchanging the labels of Paper1 and Paper5, we ensure the models cannot exploit this regularity.

B Models

B.1 Parameters

All our combined models have a total of $\approx 141\,000\,000$ parameters, with the majority of them coming from the language model. Our baseline has $\approx 30\,000\,000$.

B.2 Computation infrastructure

All the experiments ran on the LUMI supercomputer,[4] using a single AMD MI250X GPU. Running all the experiments reported in the paper has a total computing budget of approximately 150 hours.

B.3 Hyperparameters

We tune the learning rate for the text encoder from $\{1 \times 10^{-6}, 1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$, the graph encoder module from $\{2 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 1 \times 10^{-5}\}$, and the MLP module from $\{1 \times 10^{-4}, 1 \times 10^{-5}\}$; the dropout value from $\{0.1, 0.2, 0.3\}$ and the epochs from $\{5, 8, 10\}$, with no early stopping. The hyperparameters are tuned on a development set targeting the substitutivity task with an equal mix of 2, 3, and 4-hop questions. We run 60 trials for each of the five model configurations and sample a random combination of these parameters for each trial, selecting the best combination for our experiments. We find that all models perform the best with no decay on the learning rate.

[4] https://www.lumi-supercomputer.eu/sustainable-future/

Hyperparameter	BASELINE	DISJOINT	GROUNDED	UNIDIRECTIONAL	BIDIRECTIONAL
LM lr	-	0.00001	0.00001	0.00001	0.00001
GNN lr	0.00001	0.0005	0.0005	0.00005	0.0005
MLP lr	0.00001	0.0001	0.0001	0.0001	0.0001
Dropout	0.3	0.3	0.3	0.3	0.3
Epochs	5	8	8	8	8

Table 5: Variable training hyperparameters for all five model configurations

Hyperparameter	Value
Pre-trained LM	bert-base-uncased (Devlin et al., 2019)
LM hidden size	768
GNN hidden size	768
GNN layers	4
Batch size	16
Sequence length	64
Learning rate decay	constant
Optimizer	AdamW

Table 6: Constant training hyperparameters for all five model configurations