Graph Language Models

Moritz Plenz
Computational Linguistics
Heidelberg University
plenz@cl.uni-heidelberg.de

Anette Frank
Computational Linguistics
Heidelberg University
frank@cl.uni-heidelberg.de

Abstract

While Language Models (LMs) are the workhorses of NLP, their interplay with structured knowledge graphs (KGs) is still actively researched. Current methods for encoding such graphs typically either (i) linearize them for embedding with LMs – which underutilize structural information, or (ii) use Graph Neural Networks (GNNs) to preserve the graph structure – but GNNs cannot represent text features as well as pretrained LMs. In our work we introduce a novel LM type, the *Graph Language Model* (GLM), that integrates the strengths of both approaches and mitigates their weaknesses. The GLM parameters are initialized from a pretrained LM to enhance understanding of individual graph concepts and triplets. Simultaneously, we design the GLM's architecture to incorporate graph biases, thereby promoting effective knowledge distribution within the graph. This enables GLMs to process graphs, texts, and interleaved inputs of both. Empirical evaluations on relation classification tasks show that GLM embeddings surpass both LM- and GNN-based baselines in supervised and zero-shot setting, demonstrating their versatility.[1]

1 Introduction

Knowledge Graphs (KGs) are essential for organizing vast data, to facilitate information retrieval, or revealing hidden insights for decision-making (Plenz et al., 2024). KGs excel in explicitly representing manifold relationships, so with an expanding wealth of information they become crucial tools in the digital age.

Many KGs consist of knowledge triplets, where nodes are entities and edges represent relationships holding between them. Each triplet represents a fact in pseudo-natural language, e.g., *(Thailand; Capital; Bangkok)* in DBpedia (Auer et al., 2007). Despite the (usual) simplicity of each individual triplet, complex structures emerge in KGs. We refer to such KGs as Graphs of Triplets (GoTs).

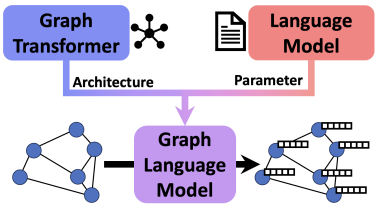


Figure 1: The GLM inherits its architecture from a Graph Transformer, and its parameters from a LM. This enables it to jointly reason over graphs and language.

To use GoTs effectively, we need meaningful encodings of their components. A natural choice is leveraging LMs, as they can capture the semantics of textually encoded entities, relations or entire triplets. But LMs are not prepared to capture graph-structured information and cannot model complex interactions in a GoT. To alleviate this problem, one can leverage graph NNs (GNNs). But GNNs are not well suited to capture meanings associated with text, and hence often LMs are used to convert nodes (and possibly edges) to language-based semantic embeddings. But in such settings, semantic encoding leveraged from LMs and structural reasoning performed by GNNs are separated and are driven by distinct underlying principles. We expect this to limit model performance if both textual and structural information are important for a task.

In this work we introduce a *Graph Language Model* (GLM) that resolves this tension through early fusion of textual and structural information. Most LMs today are transformers. Since transformers operate on sets, *Positional Encoding* (PE) is used to inform them about the inherent sequential ordering of linguistic inputs. In our GLM formulation, we modify PE and self-attention to convert LMs (i.e., sequence transformers) to graph transformers that natively operate on graphs, while preserving their LM capabilities. Usually, a new architecture requires pretraining from scratch, which is

[1] https://github.com/Heidelberg-NLP/GraphLanguageModels

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4477–4494

extremely costly. By adopting some non-invasive changes in the LM's self-attention module, we transform the LM to a Graph Transformer (GT) – while maintaining compatibility with its pretrained LM parameters. When encoding a graph, LM-like attention patterns process linearly organized textual information of individual triplets, while GT-like attention patterns aggregate information along the graph structure. Hence, the GLM inherits text understanding of triplets from the LM, while its GT architecture allows it to directly perform structural reasoning, without additional GNN layers.

Importantly, for text sequences – which can be seen as a special type of graph – the GLM is identical to the original LM. This allows the GLM to process interleaved inputs of text and GoT jointly, handling both modalities in a single framework.

Our main contributions are: (i) We propose *Graph Language Models* (GLMs) and a theoretical framework to construct them. GLMs are graph transformers, which enables graph reasoning. Simultaneously, they inherit and exploit LM weights, enabling them to represent and contextualize triplets in a GoT. Further, by encoding texts and graph components alike, it can naturally take graph and text data as interleaved inputs. (ii) Experiments on relation classification in ConceptNet subgraphs show that GLMs outperform LM- and GNN-based methods for encoding GoTs – even when the inherited LM parameters are not updated during GLM training. (iii) KG population experiments on Wikidata subgraphs and corresponding Wikipedia abstracts show that GLMs can reason over interleaved inputs of GoTs and text – again, outperforming strong LM-based methods.

2 Related Work

LMs One way to augment LMs with knowledge from KGs (Pan et al., 2024) is to formulate pretraining objectives that operate on a KG. E.g., LMs can be trained to generate parts of a KG, encouraging the LM to store KG content in its parameters. Typically, single triplets are used for pretraining (Bosselut et al., 2019; Wang et al., 2021; Hwang et al., 2021; West et al., 2023). In such cases, the graph structure is not a target of pretraining. Some works generate larger substructures, such as paths or linearized subgraphs (Wang et al., 2020a; Schmitt et al., 2020; Huguet Cabot and Navigli, 2021). In either case, the LM needs to memorize the KG, as it will not be part of the input during inference.

Another approach is to provide the linearized KG as part of the input during inference. This is common for KG-to-text generation (Schmitt et al., 2020; Ribeiro et al., 2021; Li et al., 2021), where models learn to take the linearized (typically small-sized) graphs as input. A recent trend is retrieval augmented generation, where relevant parts of a knowledge base, or KG, are retrieved, linearized and provided as part of a prompt (Gao et al., 2024).[2]

In both options the graph must be linearized to fit the input or output of a sequence-to-sequence LM. Hence, no graph priors can be enforced – instead, the LM has to learn the graph structure implicitly. By contrast, GLMs model a graph as a true graph and have inductive graph priors instilled in their architecture. This prepares a GLM for more proficient graph reasoning, compared to a LM approach.

GNNs LMs excel at representing single triplets, but struggle with structural reasoning. To alleviate this problem, LMs can be combined with GNNs. Many approaches get node and edge features from LMs and aggregate this information in the graph with GNNs (Lin et al., 2019; Malaviya et al., 2020; Zhao et al., 2023). Zhang et al. (2022); Yasunaga et al. (2022) train models consisting of a LM and a GNN that encode interleaved text and graph inputs jointly. They also use a LM to obtain node features.

While some approaches jointly train for textual understanding and graph reasoning, none offer a unified method. By contrast, our GLM formulation seamlessly integrates both in a holistic framework for embedding language and KGs.

Graph Transformers GTs, a special type of GNN (Bronstein et al., 2021), gain popularity in NLP and beyond (Min et al., 2022; Müller et al., 2023). E.g., Koncel-Kedziorski et al. (2019) and Wang et al. (2020b) train GTs to generate text from KGs and AMRs, respectively. Most relevant to our work is Schmitt et al. (2021), who use GTs for KG-to-text generation. Similar to us, they employ PE matrices, but train their model from scratch, which limits its applicability: while their model trained on WebNLG (Gardent et al., 2017) has a vocabulary size of 2,100, initializing a GLM from T5, equips it with T5's full vocabulary of 32,128 tokens.

Concurrently, and independently from our work, Li et al. (2024) also convert a LM to a graph transformer. They focus on data-to-text generation,

[2]Cf. `www.llamaindex.ai` and `www.langchain.com`

where they unify table, key-value and KG structures in a unified graph format, and apply structure-enhanced pre-training to support data-to-text generation with their structure-enhanced transformer model. They apply attention maps similar to ours to better capture the graph-structured input, which the pre-trained model rewrites into natural language. Contrary to their work, we do not resort to structure-enhanced pre-training – which is restricted in resources – but instead assess the GLMs' innate capabilities. We showcase the versatility of the inherited LM parameters in conjunction with our graph transformer architecture, by applying them to challenging reasoning tasks, where the model needs to reason over *complementary* inputs from text and graphs, and where it needs to infer information *not* present in the input, unlike data-to-text generation. Moreover, we demonstrate that our architectural changes are highly compatible with the original LM weights, via linear probing experiments, where the GLM outperforms conventional LM and Graph Transformer models.

3 Preliminary: Graph Transformers (GT)

This section briefly introduces graph transformers, focusing on architectural choices relevant for our work. We also discuss some general properties of GNNs that motivate our design choices in §4.

The attention in self-attention can be written as

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d}} + B_P + M\right)V, \qquad (1)$$

where Q, K and V are the query, key and value matrices, and d is the query and key dimension. The B_P and M matrices can be used for positional encoding and masking. Setting $B_P = M = 0$ yields the standard formulation (Vaswani et al., 2017).

Positional Encoding The self-attention mechanism of transformer models is permutation invariant, i.e., it doesn't have any notion of the order of its input elements. Thus, positional Encoding (PE) is used to inform LMs of the ordering of tokens in a text (Dufter et al., 2022). Most approaches employ either *absolute PE*, where absolute token positions are encoded (Vaswani et al., 2017; Gehring et al., 2017) or *relative PE*, which encodes the relative position between pairs of tokens (Shaw et al., 2018; Raffel et al., 2020; Su et al., 2021; Press et al., 2022). Absolute PE is typically combined with the input sequence and hence, the PE does not need to be encoded in self-attention ($B_P = 0$). For relative PE, B_P encodes a bias depending on the relative distances between pairs of tokens – for example, by learning one scalar for each possible distance:

$$B_P = f(P), \qquad (2)$$

where P is a matrix of relative distances and $f(\cdot)$ an elementwise function.

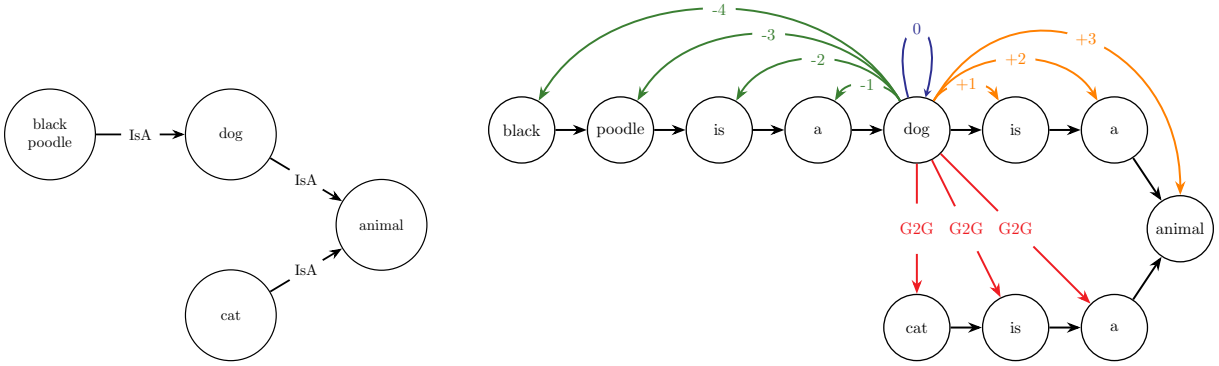
Similarly, GTs use PEs to encode the structure of the input, and hence, their PE has to encode a graph structure, as opposed to a sequence. This can again be done with absolute or relative PEs. However, defining an "absolute position" of a node or edge in a graph is not straightforward. While many methods exist, they are not directly compatible with the usual (absolute) "counting position" known from sequence encoding in LMs. In this work we thus focus on relative PE. Given a directed acyclic path in a graph, we can define the (signed) distance between any pair of nodes along a path simply as the number of hops between the nodes. The sign can be set by the direction of the path. Thus, by finding a consistent set of such paths in §4, we obtain relative distances and hence the graph's PE.

Masked Attention In a vanilla transformer, self-attention is computed for all possible pairs of tokens in the input. By contrast, nodes typically only attend to adjacent nodes in GNNs. Therefore, information between more distant nodes has to be propagated across multiple GNN layers. For graphs, such sparse message passing approaches are sometimes preferred, as in most graphs the neighborhood size increases exponentially with increasing radius, which can cause loss of information due to over-smoothing (Chen et al., 2020). Thus, in GTs it can be beneficial to introduce graph priors, for example by restricting self-attention to local neighborhoods. This can be realized by setting elements of M to 0 for pairs of tokens that should be connected, and to $-\infty$ otherwise.

On the other hand, it has been shown that a global view of the graph can enable efficient, long-ranged information flow (Alon and Yahav, 2021; Ribeiro et al., 2020). We will therefore present two model variants in §4 – a *local* and a *global* GLM.

4 Graph Language Model

GLM vs. GT We aim to design an architecture that can efficiently and jointly reason over text and graph-structured data. GTs can offer desired graph priors, but they lack language understanding.



(a) Original GoT.

(b) Extended Levi graph of GoT (with relative distances P for *dog*).

Figure 2: Example of graph preprocessing in our GLM. Fig 2b shows relative distances for *dog*, i.e., when *dog* is attending to other tokens. The red Graph-to-Graph (G2G) connections only exist for the gGLM, not for the ℓGLM.

One intuitive approach to bridge this gap is to pretrain a GT from scratch (Schmitt et al., 2021). But pretraining is costly and the necessary data bound to be scarce. We thus take a different avenue. We hypothesize that for reasoning over GoTs a model needs language understanding capabilities similar to those used for reasoning over text. Intuitively this should be the case, since (i) GoTs are designed to be understandable by humans and (ii) literate people can "read" and understand GoTs.

By initializing a GT with parameters from a compatible LM, we obtain our Graph Language Model (GLM). The GT architecture introduces graph priors, while parameter initialization from the LM gives it language understanding capabilities. In the following we explain the necessary modifications to the input graph and to the model, to make this work. The general idea is that (verbalized) triplets should resemble natural language as much as possible to enable LM weights to capture them, while graph reasoning should work via message passing.

Graph preprocessing A LM tokenizer converts text into a sequence of tokens from the LM vocabulary. Similarly, we process GoTs, such that the GLM can process the graphs "as a LM would do" (cf. Fig. 2). To achieve this, we first convert the GoT to its so-called Levi graph (Schmitt et al., 2021), i.e., we replace each edge with a node that contains the relation name as text feature, and connect the new node to the head and tail of the original edge via unlabeled edges, preserving the direction of the original edge. Next, we tokenize each node and split each node into multiple nodes, such that every new node corresponds to a single token. New edges connect adjacent nodes, again preserving the original direction. This yields the extended Levi graph (see Fig. 2b). In this representation, each triplet is represented as a sequence of tokens – just as it would be for a standard LM.[3]

	black	poodle	is	a	dog	is	a	animal	cat	is	a
black	0	1	2	3	4	G2G	G2G	G2G	G2G	G2G	G2G
poodle	-1	0	1	2	3	G2G	G2G	G2G	G2G	G2G	G2G
is	-2	-1	0	1	2	G2G	G2G	G2G	G2G	G2G	G2G
a	-3	-2	-1	0	1	G2G	G2G	G2G	G2G	G2G	G2G
dog	-4	-3	-2	-1	0	1	2	3	G2G	G2G	G2G
is	G2G	G2G	G2G	G2G	-1	0	1	2	G2G	G2G	G2G
a	G2G	G2G	G2G	G2G	-2	-1	0	1	G2G	G2G	G2G
animal	G2G	G2G	G2G	G2G	-3	-2	-1	0	-3	-2	-1
cat	G2G	G2G	G2G	G2G	G2G	G2G	G2G	3	0	1	2
is	G2G	G2G	G2G	G2G	G2G	G2G	G2G	2	-1	0	1
a	G2G	G2G	G2G	G2G	G2G	G2G	G2G	1	-2	-1	0

Figure 3: Relative position matrix P for tokens in Fig. 2b. Entries with *G2G* have no relative position (ℓGLM) or are initialized from $+\infty$ (gGLM). Cf. §A.

Positional Encodings As discussed in §3, we prefer PEs that encode the relative position between pairs of tokens, determined by their signed distance. We can directly adopt this method to encode the relative position between pairs of tokens occurring within the same triplet – by simply considering the triplet as a piece of text, and counting the token distance in this text. Note that a single token can occur

[3]Note that the token sequence of the converted GoT is not necessarily perfectly identical to the token sequence that corresponds to the input triplets. We tokenize each node in the Levi graph individually, to ensure consistent tokenization of concepts shared by multiple triplets. This removes whitespace between concepts and edges, which impacts tokenization. We leave investigation of the impact of this effect to future work.

in multiple triplets, leading to, e.g., multiple "left-hand side neighbors" (cf. *animal* in Fig. 2b and 3). While this does not occur in ordinary sequential text, it does not impose a problem for relative PE.

Yet, the approach above breaks when tokens do not belong to the same triplet. To determine the distance between such pairs of tokens, previous work considered, e.g., the length of the shortest path between them (Schmitt et al., 2021). However, this results in PEs that do not come natural to a LM, since a triplet would appear in reversed order, if it is traversed in "the wrong direction" in the shortest path.[4] We therefore omit structure-informed PE between tokens that do not belong to the same triplet and instead propose two GLM variants: a local (ℓGLM) and a global (gGLM) one.

Local and global GLM Fig. 3 shows the relative token position matrix P for the graph in Fig. 2b. In the ℓGLM the self-attention mechanism is restricted to tokens from the same triplet. This means that attention to any token located beyond the local triplet is set to 0 – and hence does not require PE. Still, in such configurations, messages can propagate through the graph across multiple layers, since tokens belonging to a concept can be shared by multiple triplets. This is analogous to standard message passing in GNNs, where non-adjacent nodes have no direct connection, but can still share information via message passing. For example, the representation of *dog* is contextualized by the triplets *black poodle is a dog* and *dog is a animal* after the first ℓGLM layer. Hence in the second layer, when *animal* attends to *dog*, the *animal* embedding gets impacted by *black poodle*, even though there is no direct connection from *animal* to *black poodle*.

However, it has been shown that a global view can have benefits (Ribeiro et al., 2020). Hence, we also formalize the gGLM, as an alternative where self-attention can connect any node to every other node. For this setting we need to assign a PE to any pair of tokens, including those that do not occur within the same triplet. For these pairs we introduce a new graph-to-graph (G2G) relative position. LMs don't have learned parameters for G2G connections, so we initialize the parameters with the corresponding parameters of a relative position of $+\infty$. In a LM a relative position of $+\infty$ means that the respective tokens occur somewhere "far" away in a remote text passage. LMs learn a *proximity bias* during pretraining, i.e., they tend to have higher attention scores between tokens that are close to each other in the text. This means that tokens with a high relative distance tend to have low attention scores. For our gGLM this corresponds to a graph bias where distant nodes are less important, but are still accessible.[5] Note that unlike in the ℓGLM, this bias is *not* part of the architecture. It originates from the pretrained parameters, meaning that the gGLM can learn to attend to distant tokens.

Along with P and M, the GLM takes a sequence of all tokens in the extended Levi graph as input. For this, we technically need to "linearize" the graph. However, the order of tokens in the resulting sequence does not matter: relative positions in P are determined by distances in the graph, not in the sequence. Permuting the input sequence simply means that rows and columns of P and M need to be permuted accordingly, but the resulting token embeddings remain unchanged. See example matrices for P and M for ℓGLM and gGLM in §A.

Being transformers, GLMs have the same computational complexity as their respective LM. For sparse graphs the ℓGLM could make use of sparse matrix multiplication, making it more efficient than a corresponding LM or gGLM. However, for our experiments this was not necessary.

Joint graph and text encoding If we use normal matrices for P and M, the GLM is identical to its underlying LM. Hence, GLMs can be applied to texts and – more interestingly – interleaved inputs of text and graph. In this joint setting, P and M each consists of four sub-matrices that correspond to self-attention between tokens from (i) graph-to-graph, (ii) text-to-text, (iii) text-to-graph and (iv) graph-to-text. Graph-to-graph sub-matrices are formatted as described above for ℓGLM and gGLM, respectively. Text-to-text sub-matrices are standard matrices from conventional sequence transformers. We introduce new T2G and G2T relative positions for text-to-graph, and graph-to-text connections, respectively. With this, the model can learn interaction strength between the two modalities. Similar to G2G in gGLM, we initialize T2G and G2T parameters from $+\infty$. See example matrices in §A.

Uni- and Bidirectional LMs If a LM's self-attention is unidirectional, information can only propagate along the direction of arrows in Fig. 2b for

[4] For example, *cat* would see the graph as the following sequence: *cat is a animal a is dog a is poodle black*.

[5] Preliminary experiments showed that initializing G2G parameters from $+\infty$ outperforms random initialization, which outperforms initialization from 0.

the ℓGLM. This means that, e.g., the representation of the node *black poodle* is independent of the rest of the graph. We could augment the graph with inverse relations to enable bidirectional information flow with unidirectional LMs, but in this work, we restrict our analysis to bidirectional models.

T5 We use T5 (Raffel et al., 2020) – a bidirectional encoder with unidirectional decoder – as base LM to instantiate GLMs. In T5, relative distances in P group into so-called buckets, and each bucket maps to one learned positional bias in B_P for each head. Positional biases are shared across layers. The decoder is not needed to encode graphs, but can be used to generate sequences, such as text or linearized graphs in future work.

5 Experiments

We assess the GLMs' capabilities for embedding GoTs in two experiments on relation (label) classification, i.e., classifying which relation belongs to a given head and tail entity. One experiment uses ConceptNet (CN; Speer et al., 2017) subgraphs that we construct to enable analysis of the impact of structural graph properties. In a second experiment on Wikidata (Vrandečić and Krötzsch, 2014) subgraphs and associated Wikipedia abstracts we test GLMs on interleaved inputs of text and graph.

5.1 Representing and reasoning over Graphs

We construct a balanced dataset of English CN subgraphs consisting of 13,600 train, 1,700 dev and 1,700 test instances with 17 distinct relations as labels. We replace the relation to be predicted with `<extra_id_0>`, T5's first mask token.

To investigate the impact of varying graph complexities, we experiment with different graph sizes denoted by their radius r. We ensure that small graphs are strict subgraphs of larger graphs, such that potential performance gains in larger graphs must stem from additional long-ranged context.

To evaluate model effectiveness when long-ranged connections are crucial, we mask complete subgraphs around the relation to be predicted. The size of a masked subgraph is m, where $m = 0$ means no mask, $m = 1$ masks neighboring concepts, $m = 2$ masks neighboring concepts and the next relations, etc. We replace each masked concept and relation with a different mask token. Construction details and statistics are shown in §B.1.1.

5.1.1 Experimental setup

The input to our model is a CN subgraph. The relation to be predicted is replaced with `<extra_id_0>`. The GLM encodes the graphs as in §4, producing an embedding for each token. A linear classification head gives the final prediction from the mask's embedding. We verbalize unmasked relations using static templates (Plenz et al., 2023), shown in §B, Table 4.

In a **finetuning** setting we train the GLM and the classification head jointly. However, since the GLM is initialized from a LM, we hypothesize that it should produce meaningful embeddings, even without any training. To test this hypothesis, we train only the classification head, i.e., we only train a **linear probe**. In this setting, the GLM was never trained on any graph data, similar to a zero-shot setting. The linear probe only extracts linear features and hence, can only achieve high performance if the GLM embeddings show expressive features.

We report mean accuracy across 5 different runs. See §B.1.2 for hyperparameters. Unless stated otherwise, we use T5-small to allow many baselines.

5.1.2 Baselines

We compare to several baselines inspired by related work. For all baselines we utilize the T5 encoder as underlying LM. This allows us to focus on the architectural design of different model types.

LM For LM-based approaches we linearize the input graphs to a sequence, by concatenating the verbalized triplets. There are structured ways to linearize graphs, but such graph traversals generally require the graph to be directed and acyclic – which makes them inapplicable to linearizing GoTs. Instead, we order the triplets either randomly (**T5 set**) or alphabetically (**T5 list**). For *T5 set*, triplets are shuffled randomly in every training epoch such that the model can learn to generalize to unseen orderings. The concatenated triplets are passed to the T5 encoder, and the embedding of `<extra_id_0>` is presented to the classification head.

GNN For GNN baselines we encode each node of the original graph (cf. Fig. 2a) with the T5 encoder, and train a GNN using these static embeddings. After the final layer, the GNN returns 17 logits for each node. As final logits, we take the mean logit of the two nodes adjacent to the relation to be predicted. We experiment with different variants as GNN layers: **GCN** (Kipf and Welling, 2017) and **GAT** (Veličković et al., 2018). Since GNNs do not

	Model	r	1	2	3	4	5	4	4	4	4	4
		m	0	0	0	0	0	1	2	3	4	5
Lin. Prob.	ℓGLM		**55.4±0.3**	57.1±0.3	56.8±0.6	56.9±0.4	57.0±0.4	30.4±0.4	17.8±0.2	14.0±0.3	11.4±0.5	11.9±0.3
	gGLM		**55.4±0.3**	**58.6±0.7**	**58.8±0.6**	**59.3±0.7**	**59.5±0.4**	**41.8±0.8**	**25.6±0.9**	**22.0±0.6**	**19.4±0.5**	**17.0±0.2**
	T5 (list)		53.7±0.3	56.8±1.1	56.5±1.2	55.8±0.6	55.3±0.5	20.3±0.6	19.9±0.4	15.3±0.6	14.0±1.1	10.2±1.2
	T5 (set)		53.1±0.6	52.8±1.2	54.6±0.6	53.9±0.5	53.1±0.8	18.2±0.6	16.7±0.5	13.1±0.7	12.3±0.6	9.7±0.9
Finetuning	ℓGLM		64.0±1.3	64.0±1.0	64.4±0.7	**64.1±0.9**	64.2±1.1	47.9±0.4	26.8±0.8	23.8±0.9	19.8±1.1	18.1±0.7
	gGLM		63.2±0.9	64.4±1.1	64.6±1.2	**64.1±1.3**	**65.3±0.7**	**48.0±0.6**	**27.2±0.7**	**24.2±0.7**	**20.2±1.4**	**19.2±0.7**
	T5 (list)		**64.9±1.0**	64.9±1.2	**64.9±1.3**	63.9±0.9	64.0±0.6	40.4±0.8	21.8±0.8	17.8±1.0	15.4±0.3	12.8±0.5
	T5 (set)		63.9±0.7	**65.8±0.8**	64.0±0.3	**64.1±1.2**	64.3±1.1	40.3±1.2	21.8±0.7	18.0±0.6	15.5±0.6	13.1±0.7
	GCN		44.3±0.9	37.1±1.0	34.4±1.2	36.5±0.6	36.8±1.4	22.2±1.2	21.9±0.8	12.1±3.5	9.0±4.3	5.9±0.0
	GAT		44.5±0.9	40.6±1.3	36.3±1.3	37.0±0.8	37.0±0.8	20.0±0.7	20.8±0.2	14.0±0.6	13.8±0.8	11.0±0.6
	ℓGT		24.2±3.4	35.0±1.2	34.7±1.3	32.7±2.9	34.5±2.8	30.1±2.6	12.8±2.4	15.5±0.3	9.5±1.3	10.0±1.6
	gGT		27.6±1.9	29.0±0.8	23.4±1.2	19.2±1.2	15.6±1.5	18.6±0.7	13.2±1.1	14.5±0.6	12.4±1.3	12.1±1.7

Table 1: Relation label classification accuracy on CN in %. Results are shown for *Linear Probing* and *Finetuning*.

come with pretrained weights, we only apply them in finetuning, when training all parameters.

Graph transformer Finally we compare GLMs to models with the same architecture, but random weight initialization (normal graph transformers). This allows us to assess the impact of weight initialization from a LM with two further baselines: ***ℓ*GT** and ***g*GT**. We only consider GTs with finetuning.

5.1.3 Results

Linear probing Tab. 1 shows the relation label prediction accuracy for linear probing, i.e., when training only the classification head. Our first observation is that gGLM is consistently the best, outperforming ℓGLM and the LM baselines. For a radius of $r = 1$ we have exactly one triplet, which has almost the same representation in the GLM and LM approaches. The only difference is that the LM baselines have an end-of-sentence token, which the GLM does not have. Surprisingly, not having the end-of-sentence token seems to be an advantage with linear probing, but we will see later that this changes when updating model weights.

For $r \geq 3$, LM baselines show decreasing performance with increasing radii. By contrast, both ℓGLM and gGLM show increasing performances with increasing radii. This indicates that GLMs can utilize the additional context. But LM baselines don't have any inbuilt methods to grasp distances in the graph, which could cause them to fail at distinguishing relevant from less relevant information.

The performance gap between gGLM and LM models tends to increase for larger m, i.e., when larger sub-structures are masked. However, the ℓGLM underperforms for large m, highlighting the advantage of the global view in gGLM when long-ranged connections are necessary.

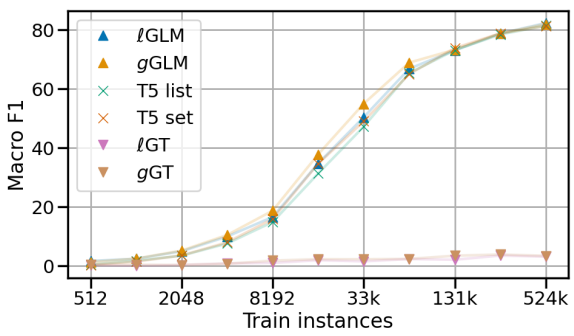
The overall high performance of GLMs confirms our assumption that GLMs are compatible with LM weights, even without any training. Increasing performance with increasing radii further shows that GLMs have good inductive graph biases. When long-range connections are relevant, the representations learned by gGLM outperform the locally constrained ℓGLM – which showcases the strength of the global view that the gGLM is able to take.

Finetuning Tab. 1 shows results when training all parameters. In this setting, models can adjust to the task and learn to reason over graphs through parameter updates. In addition, GLMs can tune parameters to better match the novel input structure.

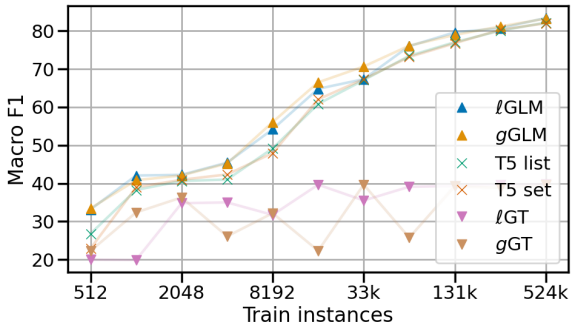
The GLM and LM variants are consistently better than GNN and GT methods, which indicates that linguistic understanding is potentially more important than graph reasoning for this task. Models outperform their linear probing scores, which shows that finetuning is, as expected, beneficial.

Overall, the GLMs perform best, while GTs perform the worst. The only difference between the two model groups is weight initialization – the GLMs are initialized from T5, while the GTs are randomly initialized. Further, we observe that for $r \geq 1$ and $m = 0$ the local GT (ℓGT) significantly outperforms its global counterpart gGT. For the GLM the global version is on par, or even better than the local one. This shows the effectiveness of T5's attention mechanism: thanks to its weight initialization, gGLM attends to relevant tokens even in large context windows, while gGT suffers from potentially distracting long-ranged information.

For $m = 0$ the differences between GLM and LM approaches are small, with a slight trend for GLMs to outperform LMs on large graphs, and vice versa for small graphs. However, when graph rea-



(a) Relation label classification.



(b) Source classification.

Figure 4: KG population test results during training. gGLM outperforms T5 set by up to 6 points in 4a.

soning is more important due to masking ($m \geq 1$), then GLMs consistently and significantly outperform all other baselines. This indicates that LMs can learn to do simple graph reasoning through parameter updates, but underperform in more complex graph reasoning tasks where either graphs are larger, or long-ranged connections are required.

For $m \geq 1$, the gGLM outperforms ℓGLM due to its global connections. In contrast to the linear probing setting, the ℓGLM outperforms other baselines for all non-zero levels of masking. This indicates that ℓGLM can learn to use long-ranged information during training, if the task requires it.

Impact of model size To investigate the effect of model size, we train the most promising approaches (GLM and LM) in 3 different sizes. Tab. 6 in §B.1.3 shows that overall larger models perform better. Surprisingly, the base models sometimes outperform the larger models for settings that require more graph reasoning, i.e., larger m. However, these differences are small and non-significant. In most cases, gGLM large or base are the best model.

5.2 Jointly representing Graph and Text

We now investigate GLM capabilities to process interleaved inputs of text and graph in a KG population setup, i.e., extending a KG with new relation instances. Subtask 1 performs text-guided *relation* classification where some relations may be inferrable from the text, while others may exploit graph knowledge to make predictions. In Subtask 2, models classify the *source* of a predicted relation, i.e., whether it can be inferred from the text, or whether it requires (additional) graph knowledge.

We construct our data from Huguet Cabot and Navigli (2021), who offer a corpus of Wikipedia abstracts that are linked to Wikidata via entity linking. Their focus is relation extraction, so they filter the graphs using NLI, such that all triplets are entailed by the text. We augment the entailed triplets with further triplets from Wikidata that are not entailed by the text. For a given text, subgraph, head and tail entity, models will jointly predict the *relation* and the *source*. We adopt the 220 most common relations in our train graphs and a "no-relation" label. For source labels we have 3 classes: entailed by the text, not entailed and no-relation. No-relation is the correct label iff the relation is also no-relation. §B.2.1 shows statistics and construction details.

5.2.1 Experimental setup and baselines

Unlike §5.1.1, models now receive text *and* graph data as input. We train two distinct classification heads on the mask's embedding for relation and source classification. While the mask is part of the graph, its embedding depends on both modalities. The final loss is the sum of the relation classification and the source prediction loss, weighted by 0.9 and 0.1. We use T5-large, but otherwise baselines are as in §5.1.2. §B.2.2 shows the training details.

5.2.2 Results

Fig. 4 and Tab. 8 show test set performance for a) relation and b) source classification, at different training stages. gGLM performs the best overall, followed by ℓGLM. LM baselines are competitive, but lag behind at early stages and for source prediction. Again, GT baselines perform poorly, showcasing the advantage of weight initialization in GLM – even with large-scale training data. For all models, training plateaus beyond $\sim$ 500k seen instances (cf. Fig. 8 in §B.2.3), so we stop training at this cut-off.

Tab. 2 gives results for ablating different input modalities to GLMs. Since source prediction always requires text input, we test relation classifi-

Ablation	Relation classification		Source classification	
	ℓGLM	gGLM	ℓGLM	gGLM
w/ text & graph	82.63	82.25	83.39	83.21
w/o text	-6.22	-5.84	–	–
w/o graph	-6.05	-5.10	-4.67	-4.49
w/o text & graph	-19.62	-19.24	–	–

Table 2: Ablations for KG population in macro F1.

cation w/o source prediction. Ablating the text or graph lowers performance by similar amounts, indicating that GLMs utilize both modalities. Training curves in Fig. 10 reveal that first, the model almost exclusively utilizes text data, but quickly learns to make use of the graph. For textually entailed triplets, text is more impactful than the graph, and vice versa for other triplets (cf. Tab. 9). Ablating graphs lowers source prediction by $\sim$4.5 points, which shows that GLMs benefit from graph information even for predominantly text oriented tasks.

The results show that GLMs can efficiently reason over interleaved inputs of graph and text, especially with limited training data. This makes GLMs a promising new model type for knowledge-intense NLP tasks, such as KG population or Q&A.

6 Conclusion

We present the Graph Language Model (GLM) – a graph transformer initialized with weights from a LM. It excels at graph reasoning, while simultaneously encoding textual triplets in the graph as LMs do, thereby bridging the gap between LMs and GNNs. GLMs can natively reason over joint inputs from texts and graphs, leveraging and enhancing each modality. Experiments show the GLM's advantage over LM and GNN based baselines, even in a linear probing setting. In particular, GLMs greatly outperform graph transformers. This highlights the need for pretrained LM weights, even for graph reasoning. We therefore advocate GLMs as a valuable tool for advancing research in embedding and leveraging knowledge graphs for NLP tasks.

Limitations

While GLMs are designed as general purpose tools for knowledge-intense NLP tasks, our evaluation is limited to English knowledge graphs. However, we explore various types of knowledge graphs (commonsense and factual) and tasks (relation classification, text-guided relation classification, and source prediction), broadening our empirical assessment. Confirming GLMs improved text and graph reasoning skills for different languages, domains and tasks is left for future work.

Our GLM framework supports instantiation from any LM with relative positional encoding, including rotary positional encoding. Comprehensive comparisons to determine the most suitable models for the GLM framework remain for future investigation. Nonetheless, bidirectional LMs are expected to perform best in the novel framework, because unidirectional LMs necessitate additional inverse relations, as discussed in §4.

Ethical considerations

We do not foresee immediate ethical concerns for our research, as we rely on well-established datasets. However, even established datasets can contain undesirable biases which our method could potentially spread and amplify.

Looking ahead, our focus lies in enriching knowledge graph integration within language models, with the aim of enhancing factuality and mitigating hallucination. This advancement is expected to bolster the reliability and controllability of LMs, leading to positive societal impacts. Furthermore, LMs relying on knowledge graphs may facilitate easier maintenance, potentially reducing the need for frequent retraining of deployed models, thereby promoting sustainability in NLP practices.

Acknowledgements

We want to thank Letiția Pârcălăbescu for providing feedback on our manuscript.

This work was funded by DFG, the German Research Foundation, within the project "ACCEPT: Perspectivized Argument Knowledge Graphs for Deliberation", as part of the priority program "RATIO: Robust Argumentation Machines" (SPP-1999).

References

Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi.

2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3438–3445.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2022. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. 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 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot Knowledge Graph-to-Text Generation with Pre-trained Language Models. In *ACL Findings*.

Shujie Li, Liang Li, Ruiying Geng, Min Yang, Binhua Li, Guanghu Yuan, Wanwei He, Shao Yuan, Can Ma, Fei Huang, and Yongbin Li. 2024. Unifying Structured Data as Graph for Data-to-Text Pre-Training. *Transactions of the Association for Computational Linguistics*, 12:210–228.

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.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wen bing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for graphs: An overview from architecture perspective. *ArXiv*, abs/2202.08455.

Luis Müller, Christopher Morris, Mikhail Galkin, and Ladislav Rampášek. 2023. Attending to Graph Transformers. *Arxiv preprint*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

Moritz Plenz, Philipp Heinisch, Anette Frank, and Philipp Cimiano. 2024. Pakt: Perspectivized argumentation knowledge graph and tool for deliberation analysis.

Moritz Plenz, Juri Opitz, Philipp Heinisch, Philipp Cimiano, and Anette Frank. 2023. Similarity-weighted construction of contextualized commonsense knowledge graphs for knowledge-intense argumentation tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6130–6158, Toronto, Canada. Association for Computational Linguistics.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21, Mexico City, Mexico. Association for Computational Linguistics.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.

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

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020a. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020b. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Peter West, Ronan Bras, Taylor Sorensen, Bill Lin, Liwei Jiang, Ximing Lu, Khyathi Chandu, Jack Hessel, Ashutosh Baheti, Chandra Bhagavatula, and Yejin Choi. 2023. NovaCOMET: Open commonsense foundation models with symbolic knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1127–1149, Singapore. Association for Computational Linguistics.

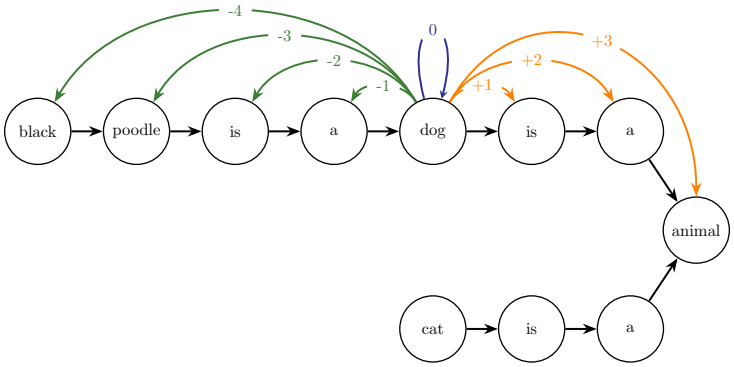
Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *Advances in Neural Information Processing Systems*.

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*.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*.

A Model

Fig. 5 shows the matrices P and M for ℓGLM and gGLM. Fig. 6 shows the same matrices for joint encoding of text and graph data.



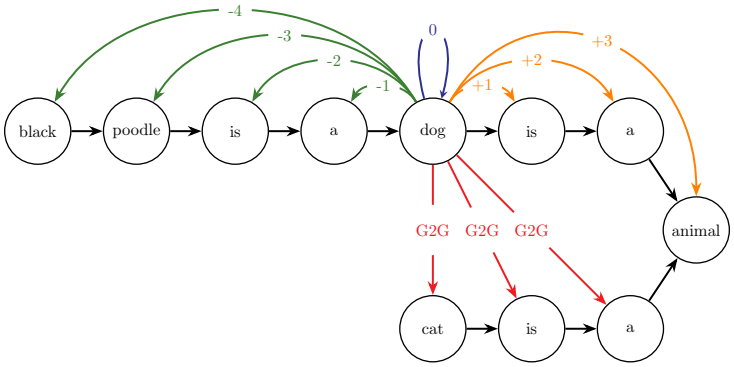
(a) Relative positions P for *dog* in ℓGLM.

	black	poodle	is	a	dog	is	a	animal	cat	is	a
black	0	1	2	3	4						
poodle	-1	0	1	2	3						
is	-2	-1	0	1	2						
a	-3	-2	-1	0	1						
dog	-4	-3	-2	-1	0	1	2	3			
is					-1	0	1	2			
a					-2	-1	0	1			
animal					-3	-2	-1	0	-3	-2	-1
cat								3	0	1	2
is								2	-1	0	1
a								1	-2	-1	0

(b) Relative position matrix P for ℓGLM

	black	poodle	is	a	dog	is	a	animal	cat	is	a
black	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞
poodle	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞
is	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞
a	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞
dog	0	0	0	0	0	0	0	0	-∞	-∞	-∞
is	-∞	-∞	-∞	-∞	0	0	0	0	-∞	-∞	-∞
a	-∞	-∞	-∞	-∞	0	0	0	0	-∞	-∞	-∞
animal	-∞	-∞	-∞	-∞	0	0	0	0	0	0	0
cat	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0
is	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0
a	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0

(c) Mask matrix M for ℓGLM.



(d) Relative position P for *dog* in gGLM.

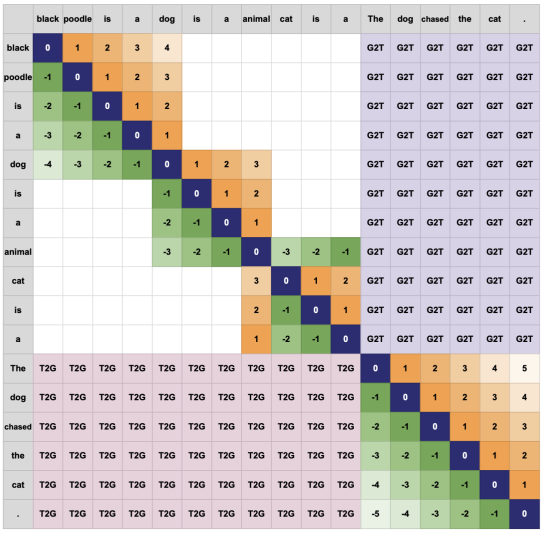
	black	poodle	is	a	dog	is	a	animal	cat	is	a
black	0	1	2	3	4	G2G	G2G	G2G	G2G	G2G	G2G
poodle	-1	0	1	2	3	G2G	G2G	G2G	G2G	G2G	G2G
is	-2	-1	0	1	2	G2G	G2G	G2G	G2G	G2G	G2G
a	-3	-2	-1	0	1	G2G	G2G	G2G	G2G	G2G	G2G
dog	-4	-3	-2	-1	0	1	2	3	G2G	G2G	G2G
is	G2G	G2G	G2G	G2G	-1	0	1	2	G2G	G2G	G2G
a	G2G	G2G	G2G	G2G	-2	-1	0	1	G2G	G2G	G2G
animal	G2G	G2G	G2G	G2G	-3	-2	-1	0	-3	-2	-1
cat	G2G	G2G	G2G	G2G	G2G	G2G	G2G	3	0	1	2
is	G2G	G2G	G2G	G2G	G2G	G2G	G2G	2	-1	0	1
a	G2G	G2G	G2G	G2G	G2G	G2G	G2G	1	-2	-1	0

(e) Relative position matrix P for gGLM

	black	poodle	is	a	dog	is	a	animal	cat	is	a
black	0	0	0	0	0	0	0	0	0	0	0
poodle	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0
dog	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0
animal	0	0	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0

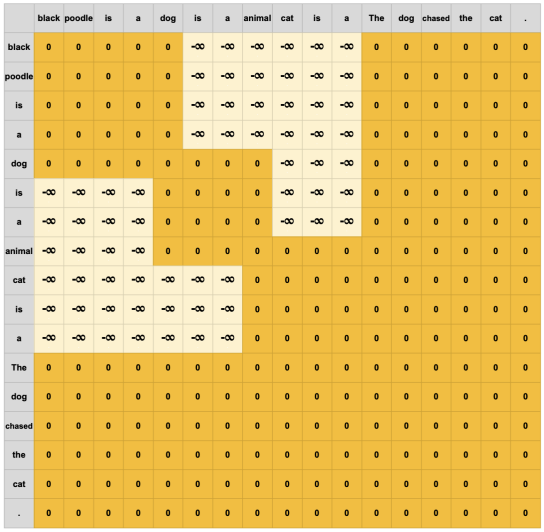
(f) Mask matrix M for gGLM.

Figure 5: Relative positions P and masking M for ℓGLM and gGLM.



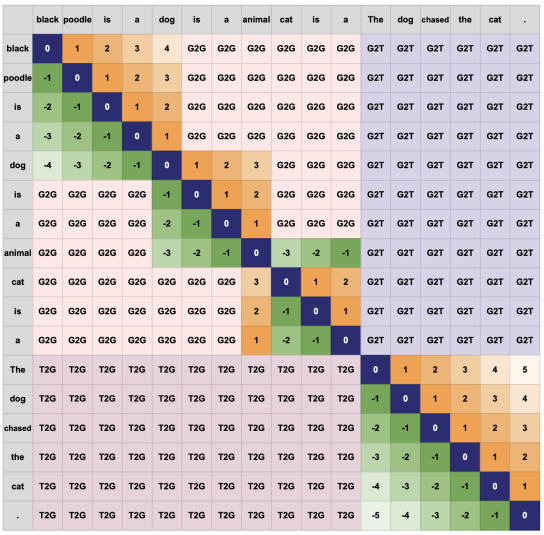
	black	poodle	is	a	dog	is	a	animal	cat	is	a	The	dog	chased	the	cat	.
black	0	1	2	3	4							G2T	G2T	G2T	G2T	G2T	G2T
poodle	-1	0	1	2	3							G2T	G2T	G2T	G2T	G2T	G2T
is	-2	-1	0	1	2							G2T	G2T	G2T	G2T	G2T	G2T
a	-3	-2	-1	0	1							G2T	G2T	G2T	G2T	G2T	G2T
dog	-4	-3	-2	-1	0	1	2	3				G2T	G2T	G2T	G2T	G2T	G2T
is					-1	0	1	2				G2T	G2T	G2T	G2T	G2T	G2T
a					-2	-1	0	1				G2T	G2T	G2T	G2T	G2T	G2T
animal					-3	-2	-1	0	-3	-2	-1	G2T	G2T	G2T	G2T	G2T	G2T
cat								3	0	1	2	G2T	G2T	G2T	G2T	G2T	G2T
is								2	-1	0	1	G2T	G2T	G2T	G2T	G2T	G2T
a								1	-2	-1	0	G2T	G2T	G2T	G2T	G2T	G2T
The	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	0	1	2	3	4	5
dog	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-1	0	1	2	3	4
chased	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-2	-1	0	1	2	3
the	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-3	-2	-1	0	1	2
cat	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-4	-3	-2	-1	0	1
.	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-5	-4	-3	-2	-1	0

(a) Relative position matrix P for ℓGLM.



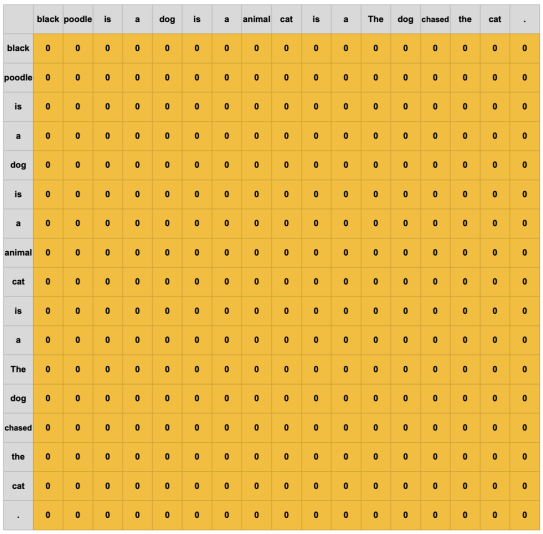
	black	poodle	is	a	dog	is	a	animal	cat	is	a	The	dog	chased	the	cat	.
black	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0
poodle	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0
is	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0
a	0	0	0	0	0	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0
dog	0	0	0	0	0	0	0	0	-∞	-∞	-∞	0	0	0	0	0	0
is	-∞	-∞	-∞	-∞	0	0	0	0	-∞	-∞	-∞	0	0	0	0	0	0
a	-∞	-∞	-∞	-∞	0	0	0	0	-∞	-∞	-∞	0	0	0	0	0	0
animal	-∞	-∞	-∞	-∞	0	0	0	0	0	0	0	0	0	0	0	0	0
cat	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0	0	0	0	0
is	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0	0	0	0	0
a	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	0	0	0	0	0	0	0	0	0
The	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dog	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
chased	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Mask matrix M for ℓGLM.



	black	poodle	is	a	dog	is	a	animal	cat	is	a	The	dog	chased	the	cat	.
black	0	1	2	3	4	G2G	G2G	G2G	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
poodle	-1	0	1	2	3	G2G	G2G	G2G	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
is	-2	-1	0	1	2	G2G	G2G	G2G	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
a	-3	-2	-1	0	1	G2G	G2G	G2G	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
dog	-4	-3	-2	-1	0	1	2	3	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
is	G2G	G2G	G2G	G2G	-1	0	1	2	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
a	G2G	G2G	G2G	G2G	-2	-1	0	1	G2G	G2G	G2G	G2T	G2T	G2T	G2T	G2T	G2T
animal	G2G	G2G	G2G	G2G	-3	-2	-1	0	-3	-2	-1	G2T	G2T	G2T	G2T	G2T	G2T
cat	G2G	G2G	G2G	G2G	G2G	G2G	G2G	3	0	1	2	G2T	G2T	G2T	G2T	G2T	G2T
is	G2G	G2G	G2G	G2G	G2G	G2G	G2G	2	-1	0	1	G2T	G2T	G2T	G2T	G2T	G2T
a	G2G	G2G	G2G	G2G	G2G	G2G	G2G	1	-2	-1	0	G2T	G2T	G2T	G2T	G2T	G2T
The	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	0	1	2	3	4	5
dog	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-1	0	1	2	3	4
chased	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-2	-1	0	1	2	3
the	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-3	-2	-1	0	1	2
cat	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-4	-3	-2	-1	0	1
.	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	T2G	-5	-4	-3	-2	-1	0

(c) Relative position matrix P for gGLM.



	black	poodle	is	a	dog	is	a	animal	cat	is	a	The	dog	chased	the	cat	.
black	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
poodle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dog	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
animal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
is	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
The	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dog	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
chased	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
the	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(d) Mask matrix M for gGLM.

Figure 6: Relative positions P and masking M for ℓGLM and gGLM when encoding text and graph jointly. The example sentence is "*The dog chased the cat.*"

B Experiments

B.1 ConceptNet

B.1.1 Dataset

We experiment on randomly selected subgraphs from the largest connected component of the English part of CN version 5.7 (Speer et al., 2017), which consists of 125,661 concepts and 1,025,802 triplets. We select 17 distinct relation label classes (cf. Tab. 4), ensuring sufficient frequency and semantic dissimilarity. For each class, we randomly sample 1,000 triplets, allowing only cases where exactly one triplet connects the head and tail entities, to reduce label ambiguity. These 1,000 instances are split into train (800), dev (100), and test (100). This creates a balanced dataset of 13,600 train, 1,700 dev, and 1,700 test instances. To predict relation labels, we replace them with `<extra_id_0>`, T5's first mask token. For our experiments, we replace CN (unmasked) relations with more natural verbalizations. Tab. 4 shows the static verbalization for each relation.

During graph construction we control the graph size, parameterized by the radius r. We start with a radius of $r = 1$, when we consider only the two concepts (head and tail) in the target triplet. To create a larger graph context, we randomly select 4 adjacent triplets – 2 for the head, and 2 for the tail entity of the original triplet. A graph with radius $r = 2$ is formed by the subgraph spanned by all entities used in these 5 triplets. For $r = 3$ we again randomly select 2 triplets for each of the outer (up to) 4 entities, yielding (up to) 13 triplets. To avoid accidentally adding more short-ranged information, we restrict the new triplets to triplets that actually extend the radius of the graph. This enables us to control graph size and complexity, while still enabling sufficient diversity in the graph structure. Further, the graphs are created such that graphs for smaller radii are strict subgraphs of graphs with larger radii. This ensures that performance changes with increasing radii are due to long-ranged connections, and not due to potentially different short-ranged information. Tab. 3 shows structural properties of CN subgraphs, depending on their radius.

When masking subgraphs, we mask complete subgraphs of a certain size around the target to be predicted. The size of the masked subgraph is denoted by m, where $m = 0$ means no masking, $m = 1$ masks neighboring concepts, $m = 2$ masks neighboring concepts and the next relations, and so on. Formally, m denotes the radius of the masked graph in Levi representation, which should not be confused with the extended Levi graph, nor the normal graph representation. We replace each concept and relation in the masked subgraph with a different mask token. This in principle enables LM baselines to internally reconstruct the graph.

B.1.2 Experimental setup and baselines

Tab. 5 shows our hyperparameters. For the GNNs, we tested different numbers of layers (2, 3, 4, 5), hidden channel dimensions (32, 64, 128), and non-linearities (ReLU, leaky ReLU) in preliminary experiments.

B.1.3 Results

Tab. 6 shows performance on CN for different model-sizes.

B.2 Wikidata and Wikipedia

B.2.1 Dataset

Huguet Cabot and Navigli (2021) propose a large-scale corpus of aligned Wikipedia abstracts and Wikidata (Vrandečić and Krötzsch, 2014) triplets. They first extract Wikidata entities from the abstract, and then link these entities with triplets in Wikidata. They are interested in triplets that are entailed by the text, so they use a NLI model to filter out all other triplets. They publicly released the extracted entities and the filtered triplets.

For our purpose, we are interested in aligned graphs and texts, but triplets in the graph do not necessarily have to be entailed by the text. Hence, we find all triplets between the extracted entities using the Wikidata Query Service.[6] From Huguet Cabot and Navigli (2021) we know which triplets in our graphs are entailed by the text.

Similar to Huguet Cabot and Navigli (2021) we consider the 220 most common relations in the train split as our relation labels. Additionally, we add a "no-relation" label, yielding 221 relation classes.

For 10 % of the graphs we randomly add a new triplet between previously unconnected head and tail entity, and the mask token as relation. For these graphs "no-relation" is the correct relation label. For the other 90 % graphs we replace a random existing relation with the mask token, while making sure that (i) the existing relation is in our 220 labels and that (ii) there is no other triplet connecting the respective head and tail entities. We remove

[6] https://query.wikidata.org, accessed in Jan. 2024.

Metric	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$
#nodes	2.00 ± 0.00	5.77 ± 0.46	12.28 ± 1.67	23.47 ± 4.33	42.90 ± 9.57
#edges	1.00 ± 0.00	8.25 ± 2.74	19.19 ± 5.33	36.41 ± 9.09	66.06 ± 16.77
mean degree	1.00 ± 0.00	2.87 ± 0.96	3.14 ± 0.82	3.11 ± 0.59	3.08 ± 0.42

Table 3: Structural statistics of ConceptNet (§5.1) train graphs.

	Relation	Verbalization
Used as relation label	Antonym	is an antonym of
	AtLocation	is in
	CapableOf	is capable of
	Causes	causes
	CausesDesire	causes desire
	DistinctFrom	is distinct from
	FormOf	is a form of
	HasContext	has context
	HasPrerequisite	has prerequisite
	HasProperty	is
	HasSubevent	has subevent
	IsA	is a
	MannerOf	is a manner of
	MotivatedByGoal	is motivated by
	PartOf	is a part of
	Synonym	is a synonym of
	UsedFor	is used for
Not used as relation label	CreatedBy	is created by
	DefinedAs	is defined as
	Desires	desires
	Entails	entails
	HasA	has
	HasFirstSubevent	starts with
	HasLastSubevent	ends with
	InstanceOf	is an instance of
	LocatedNear	is near
	MadeOf	is made of
	NotCapableOf	is not capable of
	NotDesires	does not desire
	NotHasProperty	is not
	ReceivesAction	receives action
	RelatedTo	is related to
	SymbolOf	is a symbol of

Table 4: Verbalization templates for relations in ConceptNet. The upper part of the relations are the 17 classes in the classification task.

	Parameter	Value
GLM, LM & GT	Loss	cross entropy loss
	Optimizer	AdamW
	Learning rate	1e−4 (FT) & 5e−3 (LP)
	Batchsize	32
	Max. # epochs	50
	Early stopping criterion	dev loss
	Early stopping # epochs	5
	# parameters in small	35B (FT) & 8k (LP)
	# parameters in base	110B (FT) & 13k (LP)
	# parameters in large	335B (FT) & 17k (LP)
	# encoder layers in small	6
	# encoder layers in base	12
	# encoder layers in large	24
GNN	Loss	cross entropy loss
	Optimizer	AdamW
	Learning rate	5e−3
	Batchsize	32
	Max. # epochs	50
	Early stopping criterion	dev loss
	Early stopping # epochs	5
	# layers	3
	hidden channel dimension	64
	non-linearity	ReLU

Table 5: Hyperparameters for §5.1. FT stands for fine-tuning and LP stand for linear probing. "# parameters" is the number of trainable parameters.

Model	r	1	2	3	4	5	4	4	4	4	4
	m	0	0	0	0	0	1	2	3	4	5
ℓGLM	small	64.0±1.3	64.0±1.0	64.4±0.7	64.1±0.9	64.2±1.1	47.9±0.4	26.8±0.8	23.8±0.9	19.8±1.1	18.1±0.7
	base	67.6±0.8	69.6±0.9	69.8±0.5	69.8±1.3	69.6±0.7	**49.2±0.8**	29.3±0.8	24.4±0.3	**20.8±0.9**	19.6±0.8
	large	**72.0±1.0**	**71.4±1.5**	**72.2±1.0**	**72.7±0.8**	**71.5±1.8**	48.4±1.1	**29.7±1.6**	**24.8±1.6**	20.0±0.9	**20.3±0.5**
gGLM	small	63.2±0.9	64.4±1.1	64.6±1.2	64.1±1.3	65.3±0.7	48.0±0.6	27.2±0.7	24.2±0.7	20.2±1.4	19.2±0.7
	base	67.8±0.7	71.3±1.0	70.5±1.2	71.5±1.1	71.1±0.4	49.7±1.2	30.2±0.8	**25.5±0.8**	**21.4±1.2**	**20.1±0.2**
	large	**72.1±1.1**	**73.9±0.7**	**74.2±0.6**	**74.8±0.8**	**73.9±0.7**	**50.1±0.5**	**31.9±1.2**	24.4±1.5	21.2±0.6	19.6±0.8
T5 list	small	64.9±1.0	64.9±1.2	64.9±1.3	63.9±0.9	64.0±0.6	40.4±0.8	21.8±0.8	17.8±1.0	15.4±0.3	12.8±0.5
	base	71.2±0.9	69.5±0.7	69.5±1.0	70.4±1.6	70.4±0.7	40.7±0.9	25.5±1.2	17.8±0.2	16.4±1.3	**13.9±0.7**
	large	**74.5±0.4**	**73.7±0.4**	**73.5±0.6**	**73.6±0.8**	**73.3±1.0**	**41.2±1.5**	**27.9±1.0**	**18.3±0.9**	**17.0±0.5**	13.0±0.9
T5 set	small	63.9±0.7	65.8±0.8	64.0±0.3	64.1±1.2	64.3±1.1	40.3±1.2	21.8±0.7	18.0±0.6	15.5±0.6	13.1±0.7
	base	71.2±0.6	69.8±0.6	69.5±0.6	70.1±0.7	69.8±1.4	40.4±0.9	23.9±1.1	**18.5±1.1**	**16.3±0.3**	**14.3±0.7**
	large	**74.9±0.3**	**73.0±0.5**	**73.1±0.8**	**72.5±1.1**	**73.5±0.4**	**41.2±1.3**	**25.1±1.3**	17.4±0.9	15.9±0.5	13.2±0.8

Table 6: Relation label classification accuracy on ConceptNet (§5.1) when training all parameters. Best score per model family is boldfaced, and best score overall is highlighted in yellow.

Metric	train	test
#nodes	5.59 ± 3.77	5.60 ± 3.78
#edges	8.71 ± 11.99	8.71 ± 12.01
mean degree	2.66 ± 1.58	2.66 ± 1.58

Table 7: Structural statistics of Wikidata (§5.2) subgraphs.

instances where no suitable triplet is available. This yields a dataset with 2,449,582 train, 135,828 val and 135,923 test instances.

Fig. 7 shows the label distributions for relation and source for train and test. Out of the 221 relations, only 195 and 194 relations occur in the train and test set, respectively. All relations in the test set also occur in the train set.

Tab. 7 shows graph statistics. Compared to CN subgraphs (c.f. Tab. 3) the graphs are relatively small, matching the size of $r = 2$. On CN we found that LMs can perform well on such small graphs, so we expect that the performance gap between GLMs and LM baselines on Wikidata would be larger if Wikidata subgraphs were larger.

B.2.2 Experimental setup and baselines

For these experiments we omit GNNs as a baseline, since they can't natively process texts.

The other models all compute an embedding of the mask token, and then two separate classification heads produce predictions for the relations (221 classes) and the source (3 classes). For each prediction, we compute the Cross Entropy Loss. The final loss is the weighted sum of these losses, weighted by 0.9 and 0.1 respectively. The relation classification has a higher weight since it has many more classes and hence, is potentially more difficult. This means that model parameters are optimized for both objectives jointly, while only the linear classification heads can specialize on their respective task.

The dataset is unbalanced (c.f. Fig 7), so report macro F1 scores instead of accuracy. This means that models only achieve high scores if they perform well on all classes, including minority classes.

We assume that classifying one out of 221 relations requires fine grained text understanding, so we initialize models from T5-large instead of T5-small. To reduce computational load, we only train one model per setting. Further, we enable efficient batching by restricting inputs to a maximum of 512 tokens. This truncates $2.8\,\%$ of train instances for GLMs and $5.1\,\%$ for LM baselines due to their less efficient graph encoding.

Hyperparameters are identical to Tab. 5, except that (i) we reduce batch size to 8, (ii) train for at most 1 epoch and (iii) don't use early stopping.

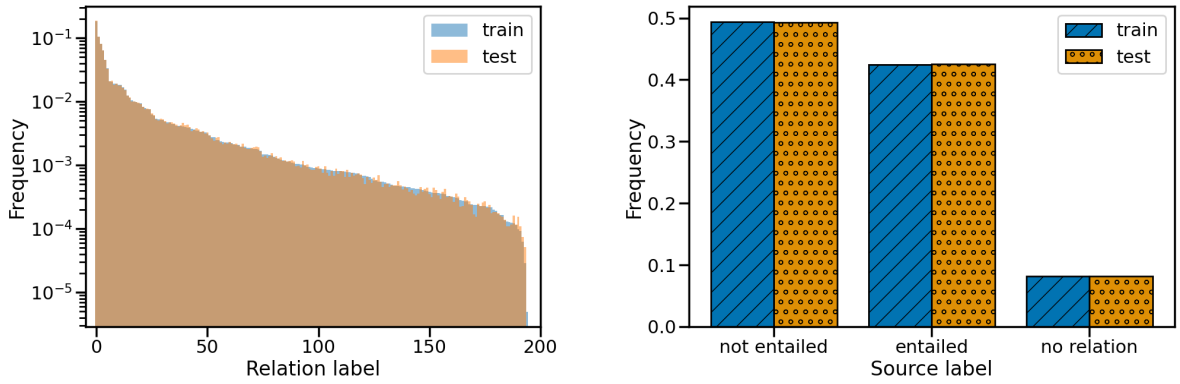
B.2.3 Results

Fig. 8 shows the training curve when training for an entire epoch, i.e., 2,499,582. We observe that performances plateau beyond ~ 0.2 epochs, so we stop training after 524,288 instances in our other experiments.

Tab. 8 shows concrete numbers for the models in Figures 4 and 8.

Fig. 9 shows confusion matrices for source prediction.

Fig. 10 shows the test performance in relation classification of ablated models during different training steps. Table 9 shows relation classification scores for (i) triplets entailed by text and for (ii)



(a) Relation label.

(b) Source label.

Figure 7: Label distributions for Wikidata (§5.2) train and test sets.

Model	524,288 train instances		2,449,582 train instances	
	Relation	Source	Relation	Source
ℓGLM	82.35	83.39	85.06	86.20
gGLM	81.98	83.21	85.28	86.17
T5 list	81.45	82.17	85.36	85.83
T5 set	81.29	82.00	85.04	85.53
ℓGT	3.19	39.81	1.50	37.83
gGT	3.47	39.58	3.40	39.37

Table 8: Macro F1 scores on Wikidata test set for relation classification and source classification. Scores are shown for models after training on different numbers of train instances.

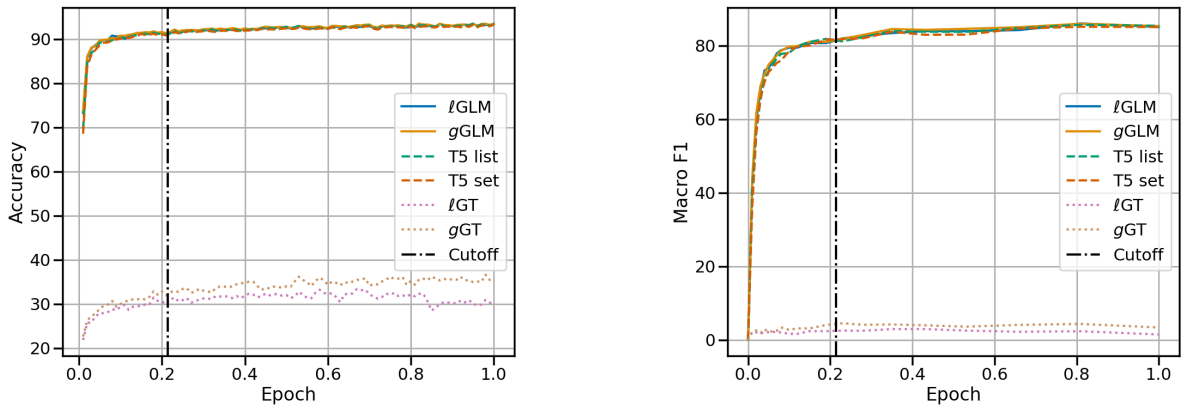
Ablation	Entailed		Not entailed	
	ℓGLM	gGLM	ℓGLM	gGLM
w/ text & graph	85.46	84.85	78.47	78.46
w/o text	-8.40	-6.75	-4.57	-4.28
w/o graph	-4.56	-3.94	-7.56	-7.55
w/o text & graph	-20.52	-19.90	-20.08	-20.07

Table 9: Ablations for KG population (§5.2). Scores are macro F1 for relation label classification on (i) triplets that are entailed by the text and (ii) all other triplets. Models are trained w/o source prediction.

other triplets.

C Usage of AI assistants

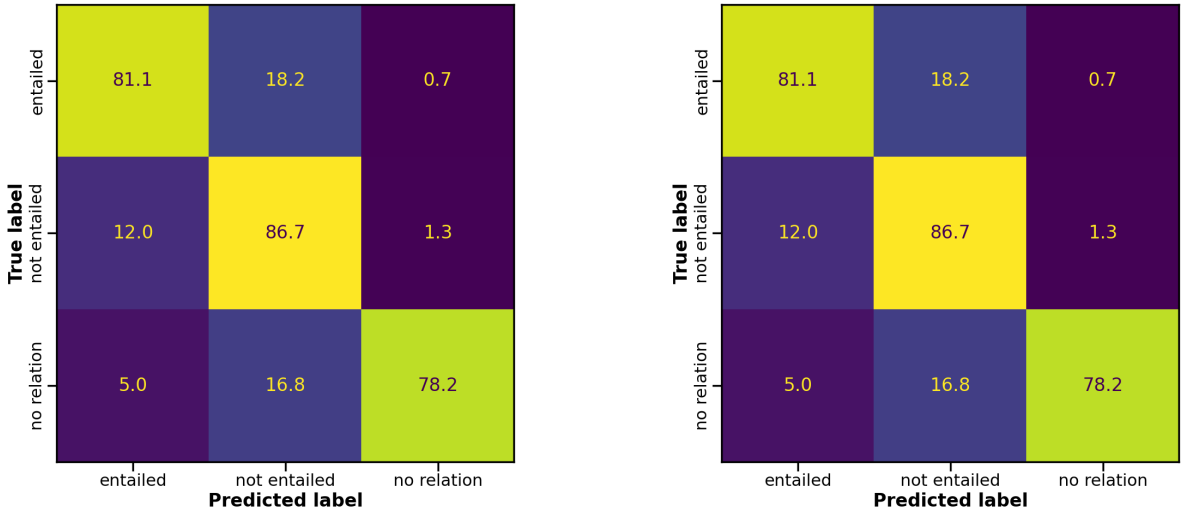
We use GitHub Copilot (https://github.com/features/copilot) for speeding up programming, and ChatGPT 3.5 (https://chat.openai.com) to aid with reformulations. The content of this work is our own, and not inspired by AI assistants.



(a) Evaluation on train set.

(b) Evaluation on test set.

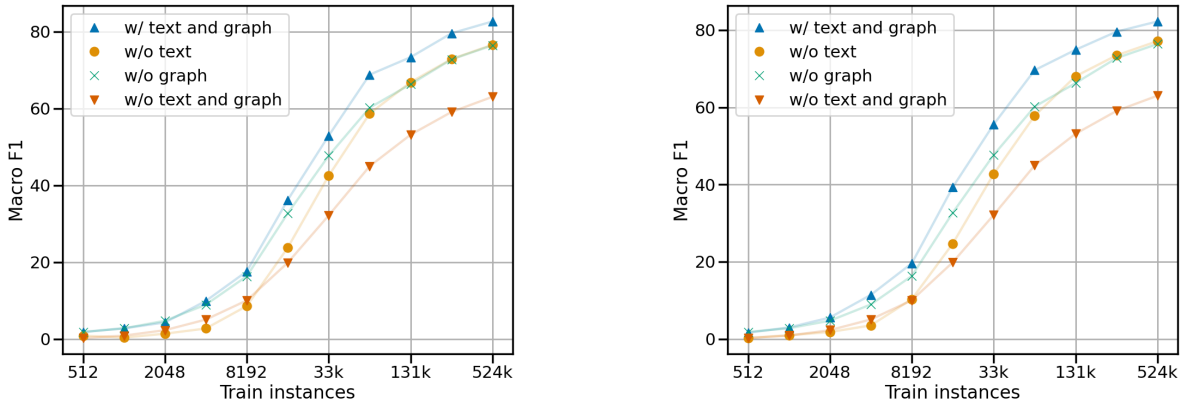
Figure 8: Training curves (§5.2) when training for a whole epoch, i.e., 2,449,582 train instances. Performances are for relation classification. On the train set we did not compute macro F1, so we report accuracy instead.



(a) ℓGLM.

(b) gGLM.

Figure 9: Confusion matrices source prediction on Wikidata (§5.2).



(a) ℓGLM.

(b) gGLM.

Figure 10: Ablation of different input modalities to GLMs. All runs are done without source prediction (besides ℓGLM and gGLM). Scores are for relation classification on Wikidata (§5.2).