# Sequence-to-Sequence Knowledge Graph Completion and Question Answering

**Apoorv Saxena**
Indian Institute of Science
Bangalore
apoorvsaxena@iisc.ac.in

**Adrian Kochsiek**
University of Mannheim
Germany
adrian@informatik.uni-mannheim.de

**Rainer Gemulla**
University of Mannheim
Germany
rgemulla@uni-mannheim.de

## Abstract

Knowledge graph embedding (KGE) models represent each entity and relation of a knowledge graph (KG) with low-dimensional embedding vectors. These methods have recently been applied to KG link prediction and question answering over incomplete KGs (KGQA). KGEs typically create an embedding for each entity in the graph, which results in large model sizes on real-world graphs with millions of entities. For downstream tasks these atomic entity representations often need to be integrated into a multi stage pipeline, limiting their utility. We show that an off-the-shelf encoder-decoder Transformer model can serve as a scalable and versatile KGE model obtaining state-of-the-art results for KG link prediction and incomplete KG question answering. We achieve this by posing KG link prediction as a sequence-to-sequence task and exchange the triple scoring approach taken by prior KGE methods with autoregressive decoding. Such a simple but powerful method reduces the model size up to 98% compared to conventional KGE models while keeping inference time tractable. After finetuning this model on the task of KGQA over incomplete KGs, our approach outperforms baselines on multiple large-scale datasets without extensive hyperparameter tuning.[1]

## 1 Introduction

A knowledge graph (KG) is a multi-relational graph where the nodes are entities from the real world (e.g. *Barack Obama, United States*) and the named edges represent the relationships between them (e.g. *Barack Obama - born in - United States*). KGs can be either domain-specific such as WikiMovies (Miller et al., 2016) or public, cross-domain KGs encoding common knowledge such as Wikidata and DBpedia (Heist et al., 2020). These graph-structured databases play an important role in knowledge-intensive applications including web search, question answering and recommendation systems (Ji et al., 2020).

Most real-world knowledge graphs are incomplete. However, some missing facts can be inferred using existing facts in the KG (Bordes et al., 2013). This task termed knowledge graph completion (KGC)[2] has become a popular area of research in recent years (Wang et al., 2017) and is often approached using knowledge graph embedding (KGE) models. KGE models represent each entity and relation of the KG by a dense vector embedding. Using these embeddings the model is trained to distinguish correct from incorrect facts. One of the main downstream applications of KGEs is question answering over incomplete KGs (KGQA) (Choudhary et al., 2021).

Taking into account the large size of real world KGs (Wikidata contains ≈90M entities) and the applicability to downstream tasks, KGE models should fulfill the following desiderata: (i) *scalability* – i.e. have model size and inference time independent of the number of entities (ii) *quality* – reach good empirical performance (iii) *versatility* – be applicable for multiple tasks such as KGC and QA, and (iv) *simplicity* – consist of a single module with a standard architecture and training pipeline. Traditional KGE models fulfill quality and simplicity. They build upon a simple architecture and reach a high quality in terms of KGC. However, as they create a unique embedding per entity/relation, they scale linearly with the number of entities in the graph, both in model size and inference time, and offer limited versatility. Methods such as DKRL (Xie et al., 2016a) and KEPLER (Wang et al., 2021) attempt to tackle the scalability issue using compositional embeddings. However, they fail to achieve quality comparable to conventional KGEs. KG-BERT (Yao et al., 2019) utilizes pretrained BERT for link prediction and holds po-

[1] Resources are available at https://github.com/apoorvumang/kgt5

[2] We use the term KGC for the task of KG link prediction.

*Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*
*Volume 1: Long Papers*, pages 2814 - 2828
May 22-27, 2022 ©2022 Association for Computational Linguistics

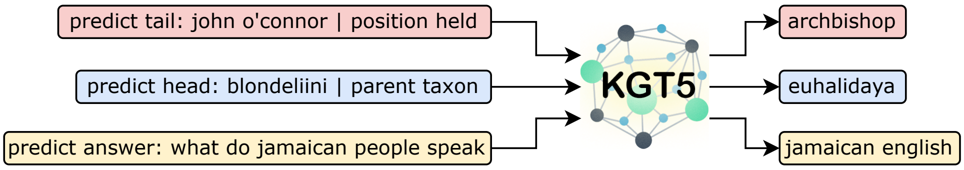
predict tail: john o'connor | position held
predict head: blondeliini | parent taxon
predict answer: what do jamaican people speak
KGT5
archbishop
euhalidaya
jamaican english

Figure 1: Overview of our method KGT5. KGT5 is first trained on the link prediction task (predicting head/tail entities, given tail/head and relation). For question answering, the same model is further finetuned using QA pairs.

tential in terms of versatility as it is applicable to downstream NLP tasks. However, it is not scalable due to its underlying cross-encoder.[3] QA methods which leverage KGEs outperform traditional KGQA approaches on incomplete KGs, but combining KGEs with the QA pipeline is a non-trivial task; models that attempt to do this often work on only limited query types (Huang et al. 2019; Sun et al. 2021; Saxena et al. 2020) or require multi-stage training and inference pipelines (Ren et al., 2021). Here, in order to achieve quality, these models have sacrificed versatility and simplicity. A comparison of approaches in terms of desiderata is summarized in Tab. 9 in the appendix.

Our paper shows that all of these desiderata can be fulfilled by a simple sequence-to-sequence (seq2seq) model. To this end, we pose KG link prediction as a seq2seq task and train an encoder-decoder Transformer model (Vaswani et al., 2017) on this task. We then use this model pretrained for link prediction and further finetune it for question answering; while finetuning for QA, we regularize with the link prediction objective. This simple but powerful approach, which we call KGT5, is visualised in Fig. 1. With such a unified seq2seq approach we achieve (i) scalability – by using compositional entity representations and autoregressive decoding (rather than scoring all entities) for inference (ii) quality – we obtain state-of-the-art performance on two tasks (iii) versatility – the same model can be used for both KGC and KGQA on multiple datasets, and (iv) simplicity – we obtain all results using an off-the-shelf model with no task or dataset-specific hyperparameter tuning.

In summary, we make the following contributions:

- We show that KG link prediction and question answering can be treated as sequence-to-sequence tasks and tackled successfully with a single encoder-decoder Transformer (with the same architecture as T5-small (Raffel et al., 2020)).
- With this simple but powerful approach called KGT5, we reduce model size for KG link prediction up to 98% while outperforming conventional KGEs on a dataset with 90M entities.
- We show the versatility of this approach through the task of KGQA over incomplete graphs. By pretraining on KG link prediction and finetuning on QA, KGT5 performs similar to or better than much more complex methods on multiple large-scale KGQA benchmarks.

## 2 Background & Related Work

Given a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$, a knowledge graph $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a collection of subject-predicate-object $(s, p, o)$ triples. Link prediction is the task of predicting missing triples in $\mathcal{K}$ by answering queries of the form of $(s, p, ?)$ and $(?, p, o)$. This is typically accomplished using knowledge graph embedding (KGE) models.

Conventional KGEs assign an embedding vector to each entity and relation in the KG. They model the plausibility of $(s, p, o)$ triples via model specific scoring functions $f(e_s, e_p, e_o)$ using the subject ($e_s$), predicate ($e_p$) and object ($e_o$) specific embeddings. Once trained, these embeddings are used for downstream tasks such as question answering.

Knowledge graph question answering (KGQA) is the task of answering a natural language question using a KG as source of knowledge. The questions can be either simple factual questions that require single fact retrieval (e.g. *Which languages are spoken in India?*), or they can be complex questions that require reasoning over multiple facts in the KG (e.g. *What are the genres of movies, in which Leonardo DiCaprio was leading actor?*). KGEs can be utilized to perform KGQA when the background KGs are incomplete.

In the next few sections we will go into more detail about existing work on KGEs and KGQA.

### 2.1 Knowledge Graph Embeddings

**Atomic KGE models.** Multiple KGE models have been proposed in the literature, mainly differing in

[3] Shen et al. (2020) estimate it would take KG-BERT 3 days for an evaluation run on a KG with just 40k entities.

the form of their scoring function $f(e_s, e_p, e_o)$. A comprehensive survey of these models, their scoring functions, training regime and link prediction performance can be found in Wang et al. (2017) and Ruffinelli et al. (2020). It is important to note that although these models obtain superior performance in the link prediction task, they suffer from a linear scaling in model size with the number of entities in the KG, and applying them to question answering necessitates separate KGE and QA modules.

**Compositional KGE models.** To combat the linear scaling of the model size with the number of entities in a KG, entity embeddings can be composed of token embeddings. DKRL (Xie et al., 2016b) embeds entities by combining word embeddings of entity descriptions with a CNN encoder, followed by the TransE scoring function. KEPLER (Wang et al., 2021) uses a Transformer-based encoder and combines the typical KGE training objective with a masked language modeling objective. Both of these approaches encode entities and relations separately which limits the transferability of these models to downstream tasks such as question answering. MLMLM (Clouatre et al., 2021) encodes the whole query with a RoBERTa-based model and uses `[MASK]` tokens to generate predictions. However, it performs significantly worse than atomic KGE models on link prediction on large KGs, and is yet to be applied to downstream text-based tasks.

### 2.2 Knowledge Graph Question Answering

Knowledge Graph Question Answering (KGQA) has been traditionally solved using semantic parsing (Berant et al. 2013; Bast and Haussmann 2015; Das et al. 2021a) where a natural language (NL) question is converted to a symbolic query over the KG. This is problematic for incomplete KGs, where a single missing link can cause the query to fail. Recent work has focused on KGQA over incomplete KGs, which is also the focus of our work. These methods attempt to overcome KG incompleteness using KG embeddings (Huang et al. 2019; Saxena et al. 2020; Sun et al. 2021; Ren et al. 2021). In order to use KGEs for KGQA, these methods first train a KGE model on the background KG, and then integrate the learned entity and relation embeddings into the QA pipeline. This fragmented approach brings several disadvantages; for example Huang et al. (2019)'s method only works for single fact question answering, while EmQL (Sun et al., 2021) requires prior knowledge of the NL question's query structure. EmbedKGQA (Saxena et al., 2020) is capable of multi-hop question answering but is unable to deal with questions involving more than one entity. Hence, these methods are lacking in versatility. LEGO (Ren et al., 2021) can theoretically answer all first order logic based questions but requires multiple dataset dependent components including entity linking, relation pruning and branch pruning modules; here, to obtain versatility, LEGO has sacrificed simplicity.

## 3 The KGT5 Model

We pose both knowledge graph link prediction and question answering as sequence-to-sequence (seq2seq) tasks. We then train a simple encoder-decoder Transformer – that has the same architecture as T5-small (Raffel et al., 2020) but without the pretrained weights – on these tasks. While training for question answering, we regularize with the link prediction objective. This method, which we call KGT5, results in a scalable KG link prediction model with vastly fewer parameters than conventional KGE models for large KGs. This approach also confers simplicity and versatility to the model, whereby it can be easily adapted to KGQA on any dataset regardless of question complexity.

Posing KG link prediction as a seq2seq task requires textual representations of entities and relations, and a verbalization scheme to convert link prediction queries to textual queries; these are detailed in §3.1. The link prediction training procedure is explained in §3.2 and inference in §3.3. The KGQA finetuning and inference pipeline is explained in §3.4.

### 3.1 Textual Representations & Verbalization

**Text mapping.** For link prediction we require a one-to-one mapping between an entity/relation and its textual representation. For Wikidata-based KGs, we use canonical mentions of entities and relations as their textual representation, followed by a disambiguation scheme that appends descriptions and unique ids to the name.[4] For datasets used for QA only we do not enforce a one-to-one mapping as, in this case, unnecessary disambiguation can even harm model performance.[5]

[4] Please see appendix A for details on textual representations.

[5] This is because QA systems consider surface forms during evaluation, not entity IDs. For example, it will be better to have the same mention for both the single and album version of a song rather than append a unique number to their mentions.

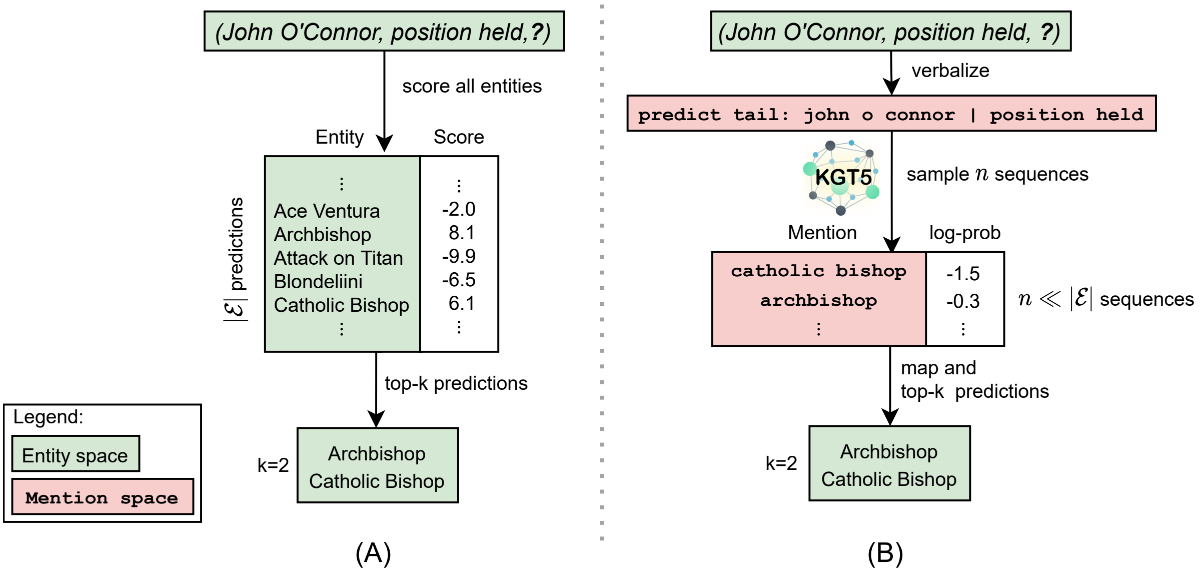

Figure 2: Inference pipeline of (A) conventional KGE models versus (B) KGT5 on the link prediction task. Given a query $(s, p, ?)$, we first verbalize it to a textual representation and then input it to the model. A fixed number of sequences are sampled from the model decoder and then mapped back to their entity IDs. This is in contrast to conventional KGEs, where each entity in the KG must be scored. Please see §3.3 for more details.

**Verbalization.** We convert $(s, p, ?)$ query answering to a sequence-to-sequence task by *verbalizing* the query $(s, p, ?)$ to a textual representation. This is similar to the verbalization performed by Petroni et al. (2019), except there is no relation-specific template. For example, given a query *(barack obama, born in, ?)*, we first obtain the textual mentions of the entity and relation and then verbalize it as `'predict tail: barack obama | born in'`. This sequence is input to the model, and output sequence is expected to be the answer to this query, `'united states'`, which is the unique mention of entity *United States*.

### 3.2 Training KGT5 for Link Prediction

To train KGT5, we need a set of (input, output) sequences. For each triple $(s, p, o)$ in the training graph, we verbalize the queries $(s, p, ?)$ and $(?, p, o)$ according to §3.1 to obtain two input sequences. The corresponding output sequences are the text mentions of $o$ and $s$ respectively. KGT5 is trained with teacher forcing (Williams and Zipser, 1989) and cross entropy loss.[6]

| Dataset | Entities | Rels | Edges | Token. vocab |
|---|---|---|---|---|
| WikiKG90Mv2 | 91M | 1,387 | 601M | 32k |
| Wikidata5M | 4.8M | 828 | 21M | 30k |
| MetaQA | 43k | 9 | 70k | 10k |
| WQSP$^\dagger$ | 158k | 816 | 376k | 32k |
| CWQ$^\dagger$ | 3.9M | 326 | 6.9M | 32k |

Table 1: Statistics of the KGs used. $^\dagger$We use subsets of FreeBase (Google, 2015) for WebQuestionsSP (WQSP) and ComplexWebQuestions (CWQ).

One thing to note is that unlike standard KGE models, we train *without explicit negative sampling*. At each step of decoding, the model produces a probability distribution over possible next tokens. While training, this distribution is penalised for being different from the 'true' distribution (i.e. a probability of 1 for the true next token, 0 for all other tokens) using cross entropy loss. Hence, this training procedure is most similar to the 1vsAll + CE loss in Ruffinelli et al. (2020), except instead of scoring the true entity against all other entities, we are scoring the true token against all other tokens at each step, and the process is repeated as many times as the length of the tokenized true entity. This avoids the need for many negatives, and is independent of the number of entities.

### 3.3 Link Prediction Inference

In conventional KGE models, we answer a query $(s, p, ?)$ by finding the score $f(s, p, o)\ \forall o \in \mathcal{E}$, where $f$ is the model-specific scoring function. The entities $o$ are then ranked according to the scores.

In our approach, given query $(s, p, ?)$, we first

[6] More details about training are available in Appendix B

verbalize it (§3.1) before feeding it to KGT5. We then *sample* a fixed number of sequences from the decoder,[7] which are then mapped to their entity ids.[8] By using such a generative model, we are able to approximate (with high confidence) top-$m$ model predictions without having to score all entities in the KG, as is done by conventional KGE models. For each decoded entity we assign a score equal to the (log) probability of decoding its sequence. This gives us a set of (entity, score) pairs. To calculate the final ranking metrics comparable to traditional KGE models, we assign a score of $-\infty$ for all entities not encountered during the sampling procedure. A comparison of inference strategy of conventional KGE models and KGT5 is shown in Figure 2.

### 3.4 KGQA Training and Inference

For KGQA, we pretrain the model on the background KG using the link prediction task (§3.2). This pretraining strategy is analogous to 'KGE module training' used in other KGQA works (Sun et al. 2021; Ren et al. 2021). The same model is then finetuned for question answering. Hereby, we employ the same strategy as Roberts et al. (2020): we concatenate a new task prefix (`predict answer:`) with the input question and define the mention string of the answer entity as output. This unified approach allows us to apply KGT5 to any KGQA dataset regardless of question complexity, and without the need for sub-modules such as entity linking.

To combat overfitting during QA finetuning (especially on tasks with small KGs) we devise a regularisation scheme: we add link prediction sequences sampled randomly from the background KG to each batch such that a batch consists of an equal number of QA and link prediction sequences. For inference, we use beam search followed by neighbourhood-based reranking (§4.3) to obtain the model's prediction which is a single answer.

## 4 Experimental Study

We investigate whether KGT5–i.e. a simple seq2seq Transformer model–can be jointly trained to perform both knowledge graph link prediction as well as question answering. Hereby, we first describe the used datasets (§4.1), the baselines we compared to (§4.2) and the experimental setup (§4.3). The results of our experiments are analysed in §4.4-§4.8. Before going into detail, we summarize our key findings:

1. For link prediction on large KGs, the text-based approach of KGT5 reduces model size to comparable KGE models by up to 98% and reaches or outperforms current state-of-the-art.
2. On the task of KGQA over incomplete KGs, our simple seq2seq approach obtains better results than the current state-of-the-art across multiple datasets.
3. KG link prediction training might be more beneficial than language modeling pretraining on knowledge intensive tasks such as KGQA.
4. Although KGT5 is good at generalizing to unseen facts, it is rather poor at memorizing facts. This problem can be alleviated, if needed, by using an ensemble of KGT5 and conventional link prediction or KGQA systems.

### 4.1 Datasets

We evaluate the link prediction capability of KGT5 on Wikidata5M (Wang et al., 2021) and WikiKG90Mv2 (Hu et al., 2021), two of the largest publicly available benchmark KGs. Although KGT5 is designed for large problems, we evaluate on the smaller benchmark KGs FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018) and YAGO3-10 (Dettmers et al., 2018) for comparability.

We evaluate the QA capabilities of KGT5 on three large-scale KGQA benchmark datasets: MetaQA (Zhang et al., 2018), WebQuestionsSP (WQSP) (Yih et al., 2016) and ComplexWebQuestions (CWQ) (Talmor and Berant, 2018). Questions in MetaQA span from 1-hop to 3-hop questions requiring path-based reasoning on a KG based on WikiMovies (Miller et al., 2016). WQSP contains both 1-hop and 2-hop path based questions while CWQ contains questions requiring steps such as compositional, conjunctive, comparative and superlative reasoning. Both WQSP and CWQ can be answered using Freebase (Google, 2015) as the background KG. We create subsets of Freebase using the scheme proposed by Ren et al. (2021) which results in KGs that are much smaller than Freebase but can still be used to answer all ques-

[7] See Appendix C for additional details on sampling and our choice of decoding strategy.

[8] The decoded sequence may or may not be an entity mention. We experimented with constrained decoding (Cao et al., 2021) to force the decoder to output only entity mentions; however, we found this unnecessary since the model almost always outputs an entity mention, and increasing the number of samples was enough to solve the issue.

| **Model** | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** | **Params** |
|---|---|---|---|---|---|
| TransE (Bordes et al., 2013) † | 0.253 | 0.170 | 0.311 | 0.392 | 2,400M |
| DistMult (Yang et al., 2015) † | 0.253 | 0.209 | 0.278 | 0.334 | 2,400M |
| SimplE (Kazemi and Poole, 2018) † | 0.296 | 0.252 | 0.317 | 0.377 | 2,400M |
| RotatE (Sun et al., 2019b) † | 0.290 | 0.234 | 0.322 | 0.390 | 2,400M |
| QuatE (Zhang et al., 2019) † | 0.276 | 0.227 | 0.301 | 0.359 | 2,400M |
| ComplEx (Trouillon et al., 2016) $ | **0.308** | **0.255** | - | **0.398** | 614M |
| KGT5 (Our method) | **0.300** | **0.267** | **0.318** | **0.365** | 60M |
| ComplEx 14-dim ‡ | 0.201 | 0.161 | 0.211 | 0.275 | 67M |
| ComplEx 26-dim ‡ | 0.239 | 0.187 | 0.261 | 0.342 | 125M |
| KEPLER (Wang et al., 2021) †† | 0.210 | 0.173 | 0.224 | 0.277 | 125M |
| DKRL (Xie et al., 2016a) †† | 0.160 | 0.120 | 0.181 | 0.229 | 20M |
| MLMLM (Clouatre et al., 2021) ‡‡ | 0.223 | 0.201 | 0.232 | 0.264 | 355M |
| KGT5-ComplEx Ensemble | **0.336** | **0.286** | **0.362** | **0.426** | 674M |

Table 2: Link prediction results on Wikidata5M . † results are from the best pre-trained models made available by Graphvite (Zhu et al., 2019) . ‡ results were obtained through a hyperparameter search with LibKGE (Broscheit et al., 2020). $ results are from (Kochsiek and Gemulla, 2021). †† results are from Wang et al. (2021). ‡‡ results are from Clouatre et al. (2021). For more details, please see §4.4.

| **Model** | **Test MRR** | **Valid MRR** | **Params** |
|---|---|---|---|
| TransE-Concat | **0.176** | 0.206 | 18.2B |
| ComplEx-Concat | **0.176** | 0.205 | 18.2B |
| ComplEx-MPNet | 0.099 | 0.126 | 307K |
| ComplEx | 0.098 | 0.115 | 18.2B |
| TransE-MPNet | 0.086 | 0.113 | 307K |
| TransE | 0.082 | 0.110 | 18.2B |
| KGT5 (Our method) | -[13] | **0.221** | 60M |

Table 3: Link prediction results on WikiKG90Mv2. Baseline numbers are from the official leaderboard of OGB-LSC (Hu et al., 2021). For more details, please see §4.4.

tions in CWQ and WQSP.

Following prior work (Sun et al., 2019a) we randomly drop 50% of edges from all KGs to simulate KG incompleteness. This stochasticity causes different works to have different KGs, making it hard to compare results without re-implementing methods. Ren et al. (2021) implemented all comparison methods using their own KG splits which they have not yet published.[9] Our KG split is available along with our implementation[1] and we encourage further studies to use it. We do not re-implement comparison methods but instead report the numbers for our methods and baselines separately. We also report the accuracy obtained by executing the ground truth SPARQL queries (GT query) for test questions. GT query serves as an estimate of the hardness of a KG split and helps us compare model performance across KG splits. Note that for training all models, we only use (NL question, answer entity) pairs - *no ground truth query information is used for training*. Statistics of the KGs used in our experiments are shown in Tab. 1. Statistics of the QA datasets are shown in Tab. 11.

### 4.2 Comparison Models

For KG completion on Wikidata5M, we compared with several standard KGE models that have been shown to achieve good performance across multiple datasets (Ruffinelli et al., 2020) but with a large number of parameters. Among low-parameter models, we compared to the text based approaches KEPLER (Wang et al., 2021), DKRL (Xie et al., 2016a) and MLMLM (Clouatre et al., 2021). We also consider low-dimensional versions of the state-of-the-art method ComplEx. For the small benchmark KGs we compared with the currently best performing model NBFNet (Zhu et al., 2021).

For KGQA, we compared against several methods that have been shown to achieve SOTA on QA over incomplete KGs. These include PullNet (Sun et al., 2019a), EmQL (Sun et al., 2021), EmbedKGQA (Saxena et al., 2020) and LEGO (Ren et al., 2021). Additionally, for the MetaQA datasets, we compared with a relation-path finding baseline, which we call PathPred. This simple

[9] Through private communication with the authors we were able to obtain the same KG split for WQSP.

| Model | CWQ | WQSP |
|---|---|---|
| GT query | 25.2 | 56.9 |
| Pullnet | 26.8 (+1.6) | 47.4 (-9.5) |
| EmbedKGQA | - | 42.5 (-14.4) |
| LEGO | 29.4 (+4.2) | 48.5 (-8.4) |
| GT query | 24.5 | 56.9 |
| KGT5 | **34.5 (+10.0)** | **50.5 (-6.4)** |

Table 4: Hits@1 (gain vs GT query) on ComplexWebQuestions (CWQ) and WebQuestionsSP (WQSP) datasets in the 50% KG setting. Baseline results are from Ren et al. (2021). We use the same KG as used by the baselines for WQSP and a slightly *harder* KG for CWQ. Please see §4.5 for more details.

method maps a NL question to a relation path using distantly supervised data obtained from QA pairs in the training set.[10]

### 4.3 Experimental Setup

In all our main experiments we used a model with the same architecture as T5-small (∼60M parameters) but without the pretrained weights. For tokenizing sequences, we trained a BPE tokenizer using the SentencePiece (Kudo and Richardson, 2018) library on the verbalised KGs (see Tab. 1 for tokenizer statistics).

We used AdaFactor (Shazeer and Stern, 2018) with a learning rate warmup schedule for link prediction training, batch size 320 and 10% dropout. We adopted the same procedure as Roberts et al. (2020) for QA finetuning - we halved the batch size and fixed the learning rate to 0.001. All experiments were performed using 4 Nvidia 1080Ti GPUs and models were implemented using the HuggingFace library (Wolf et al., 2019). *We performed no dataset-specific hyperparameter tuning* for KGT5 and used the same architecture, batch size, dropout and learning rate schedule throughout all experiments.[11] All models were trained until validation accuracy did not significantly increase for 10k steps.[12]

For inference, we used sampling size = 500 for link prediction and beam size = 4 for KGQA. We further performed a neighbourhood-based reranking for KGQA: given question $q$, topic entity from question $e$, predicted answer entity $a$ and (log) probability of predicted entity $p_a$, we compute score for $a$ being answer as

$$\begin{aligned} score(a) &= p_a + \alpha \quad \text{if } a \in \mathcal{N}(e) \\ &= p_a \qquad\quad \text{otherwise} \end{aligned} \tag{1}$$

where $\alpha$ is a constant hyperparameter and $\mathcal{N}(e)$ is the $n$-hop neighbourhood of the topic entity ($n$ = 1, 2 or 3). Re-ranking was only done on datasets where topic entity annotation is available as part of test questions.

| Model | 1-hop | 2-hop | 3-hop |
|---|---|---|---|
| GT query | 63.3 | 45.8 | 45.3 |
| PullNet | 65.1 (+1.8) | 52.1 (+6.3) | 59.7 (+14.4) |
| EmbedKGQA | **70.6 (+7.3)** | 54.3 (+8.5) | 53.5 (+8.2) |
| EmQL | 63.8 (+0.5) | 47.6 (+1.8) | 48.1 (+2.8) |
| LEGO | 69.3 (+6.0) | **57.8 (+12.0)** | 63.8 (+18.5) |
| GT query | 67.7 | 48.7 | 44.4 |
| PathPred | 67.7 (+0.0) | 48.7 (+0.0) | 44.4 (+0.0) |
| KGT5 | **75.0 (+7.3)** | 36.2 (-8.2) | **64.4 (+20.0)** |
| KGT5-PP-Ens. | **76.0 (+8.3)** | **65.4 (+16.7)** | **76.6 (+32.2)** |

Table 5: Hits@1 (gain vs GT query) on MetaQA in the 50% KG setting. Baseline results are from Ren et al. (2021). There are two ground truth query (GT query) rows since the KG used by baseline models is different from ours. KGT5-PP-Ens. is the KGT5-PathPred ensemble model. Please see §4.5 for more details.

### 4.4 Link Prediction with KGT5

Tab. 3 shows link prediction performance on WikiKG90Mv2, one of the largest benchmark KGs available. Here we compare against TransE, ComplEx and their variants. *-MPNet and *-concat methods use text embeddings as part of entity representations, and operate on the same textual data as KGT5. KGT5 achieves the highest MRR on validation set while having 98% fewer parameters than the next best performing model on the leaderboard.[13]

Tab. 2 shows link prediction performance on Wikidata5M, a smaller but better studied KG. We see that KGT5 outperformed all low-parameter count models on all metrics. When compared to the large ComplEx model, there is a drop of 0.008 points in MRR and a gain of 0.012 points in hits@1.

We performed a more fine-grained analysis of

[10] Please see Appendix D for details of PathPred.

[11] The vocabulary size for MetaQA is 10k, compared to ∼30k for other datasets. This was necessary in order to train a BPE tokenizer on such a small KG.

[12] ∼5M steps for large KGs (WD5M, W90M), ∼500k steps for smaller KGs and ∼30k steps for QA finetuning

[13] The authors of OGB-LSC did not provide us with scores on the hidden test set because we used the entity mentions that were provided with the dataset. These entity mentions have now been removed; we provide them for reproducibility on our resource website.

model predictions according to the type of query for Wikidata5M (Tab. 13 in the appendix). We found that KGT5 excelled at answering queries which have none or only a few correct answers in the train set; performance dropped when several entities can be correct for a query. This could be due to the nature of sampling: low probability sequences are harder to sample and also harder to rank correctly. Additionally, the limited sampling (§3.3) may not even provide the correct answer if there exist more known positives than sampled answers.

Based on these observations we created an ensemble of ComplEx and KGT5 which answers queries as follows: if the query does not have answers in the train KG, use KGT5; otherwise use ComplEx (614M). As shown in Tab. 2, the ensemble created by this simple rule outperformed all other single models and achieved the state-of-the-art on Wikidata5M.[14,15] Such an ensemble neither achieves the goal of scalability nor versatility but instead serves as an ablation to point out weak spots of KGT5.

Tab. 10 in the appendix shows link prediction performance on KGs with $\leq$ 150k entities. Here KGT5 sometimes falls behind the baselines; Transformer models are known to struggle when data is scarce, and this could be the reason for poor performance on these small datasets.

### 4.5 QA over Incomplete KGs with KGT5

Due to the lack of public KG splits, we compared KGQA methods using *gain over ground truth query model*, which is available for both the comparison methods (from Ren et al. 2021) as well as our methods.[16] Tab. 4 shows hits@1 performance on Freebase-based datasets ComplexWebQuestions and WebQuestionsSP. On both datasets, KGT5 outperformed all baselines. The gains were the largest on ComplexWebQuestions which is the hardest dataset in terms of complexity and KG size.

Tab. 5 shows hits@1 performance on the MetaQA datasets. On MetaQA 1- and 3-hop, KGT5 was either equal or better than all baselines (in terms of gain). On MetaQA 2-hop however, the performance was significantly worse compared to the baselines, and even worse than ground truth querying. We did a more fine-grained analysis of the performance of KGT5 on different question types (Tab. 15-16 in the appendix). We found that KGT5 performance suffered most on questions where the head and answer entity were of the same type (for e.g. *actor* → *movie* → *actor* questions). These question types are absent in the 1-hop and 3-hop datasets. When head and answer entities had different types (for e.g. *director* → *movie* → *language* questions), KGT5 was able to answer them better than GT query.

| Model | MetaQA | | | WQSP |
|---|---|---|---|---|
| | 1-hop | 2-hop | 3-hop | |
| KGT5 | 75.0 | 36.2 | 64.4 | 50.5 |
| – reranking | 73.1 | 35.8 | 63.3 | 47.2 |

Table 6: Effect of neighbourhood reranking on KGQA with 50% KG. The numbers reported are hits@1.

To remedy this issue and create a model more faithful towards the knowledge present in the incomplete KG, we devised an ensemble of KGT5 with the PathPred baseline. The ensemble works as follows: Given a question $q$, try to answer it using PathPred. If this returns an empty set, use KGT5. This ensemble outperformed all single models on all MetaQA datasets, often by large margins (Tab. 5).

Additionally, we performed an ablation to study the effect of neighbourhood reranking on KGQA performance (Tab. 6). We found that reranking gave small but consistent gains on all datasets.

### 4.6 Relation to Knowledge Probing

Knowledge probing works such as LAMA (Petroni et al., 2019) aim to answer the following question: can models (e.g. BERT) which are pretrained on *generic text corpora* with a *language modeling objective* be used as knowledge bases? In our case, the model has been explicitly trained with the *link prediction objective*, and a knowledge probing experiment would be akin to checking train set performance of link prediction (which is discussed in §4.8). Furthermore, we do not claim that KGT5 is as general purpose as large LMs, or that it contains generic world knowledge. Hence we do not perform knowledge probing experiments on datasets such as T-Rex or Google-RE (Petroni et al., 2019).

[14] In this ensemble KGT5 was used to answer 42% of the queries; the rest were answered by ComplEx

[15] To the best of our knowledge current state-of-the-art on Wikidata5M is ComplEx published with Kochsiek and Gemulla (2021) presented in Tab. 2.

[16] Details about KGs used by us compared to baselines can be seen in Tab. 14.

| Method | WQSP | CWQ |
|---|---|---|
| T5-small + QA finetuning | 31.3 | 27.1 |
| KGT5 (50% KG pretraining) | 50.5 | 34.5 |
| KGT5 (full KG pretraining) | 56.1 | 36.5 |
| EmbedKGQA | 66.6 | - |
| CBR-KGQA (Das et al., 2021b) | **73.1** | **70.4** |

Table 7: Hits@1 in the full-KG KGQA setting. For details please see §4.8.

| Model | Test MRR | Train MRR | Params |
|---|---|---|---|
| ComplEx | 0.308 | 0.721 | 614M |
| KGT5 | 0.300 | 0.304 | 60M |

Table 8: Train vs. test performance on link prediction on Wikidata5M. Please see §4.8 for details.

### 4.7 KG vs LM pretraining

We analyzed how generic corpora pretraining performed compared to KG link prediction training for the task of KGQA. We compared with T5-small (Raffel et al., 2020), which has the same architecture as KGT5 but pretrained on a mixture of tasks, most notably language modeling on web text. From Tab. 7 we see that KGT5 vastly outperformed T5-small. This is not surprising: the data for KGT5 pretraining was tailored towards the task performed–KGQA–which was not the case for T5-small. However, this shows that it is the link prediction pretraining that is responsible for the excellent KGQA performance of KGT5.

### 4.8 Limitations

**Full-KG Question Answering.** Tab. 7 shows hits@1 performance in the full KG setting. KGT5 performance only marginally improves when pretrained on full KG compared to 50% KG, and lags far behind both EmbedKGQA (a ComplEx-based method) as well as CBR-KGQA (a semantic parsing method that uses (NL-query, SPARQL-query) parallel data). This indicates that although KGT5 excels at generalizing to unseen facts, it may not be good at memorizing facts. This is further supported by the *train set* link prediction performance of KGT5 (Tab. 8); although both ComplEx and KGT5 have comparable test MRR, train MRR of ComplEx is significantly better. One possible explanation could be that the reduced model capacity of KGT5 – which has only 60M parameters – does not allow it to memorize facts seen during pretraining, leading to poor train MRR and full-KG KGQA performance. Hence we recommend against using KGT5 as a standalone KGQA method, and it should be used only when query-parsing does not yield good results.

**Use of textual mentions.** Since KGT5 requires textual representations for every entity, it cannot be directly applied to all KGs, and is especially unsuitable for KGs that contain CVT nodes as entities (e.g. full Freebase). Also, care must be taken when comparing models that make use of entity names/descriptions with those that do not. In our experiments, we noticed a significant proportion of validation triples in WikiKG90Mv2 required just text processing (eg. `<Giovanni Bensi, family name, Bensi>`) and we found a few cases of potential data leakage when definitions are used in WN18RR (eg. `<hylidae - the amphibian family of tree frogs, hypernym, amphibian family>`). However, from a practical perspective, models which can leverage text data could be more advantageous, and one must assess the pros and cons of a technique before applying it.

## 5 Conclusion and Future Work

We have shown that KG link prediction and question answering can be treated as seq2seq tasks and tackled successfully with a single encoder-decoder Transformer model. We did this by training a Transformer model with the same architecture as T5-small on the link prediction task, and then finetuning it on the QA task. This simple but powerful approach, which we call KGT5, performed competitively with the state-of-the-art methods for KG completion on large KGs while using upto 98% fewer parameters. On the task of KGQA on incomplete KGs, we found that our unified approach outperformed baselines on multiple large-scale benchmark datasets. Additionally, we compared language modeling pretraining with KG link prediction training and found that for knowledge-intensive tasks such as KGQA, link prediction training could be more beneficial.

One promising direction for future exploration would be to see whether KG link prediction training could be considered as an additional pretraining objective when training large seq2seq models. Furthermore, the impact of model size, and whether larger Transformer models can indeed store more relational information should be investigated.

## References

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 1431–1440, New York, NY, USA. Association for Computing Machinery.

Jonathan Berant, Andrew K. Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. LibKGE - A knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 165–174.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Niel Chah. 2017. Freebase-triples: A methodology for processing the freebase data dumps. *CoRR*, abs/1712.08707.

Shivani Choudhary, Tarun Luthra, Ashima Mittal, and Rajat Singh. 2021. A survey of knowledge graph embedding and their applications. *CoRR*, abs/2107.07842.

Louis Clouatre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. 2021. MLMLM: Link prediction with mean likelihood masked language model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4321–4331, Online. Association for Computational Linguistics.

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

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

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*.

Google. 2015. Freebase data dumps.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711.

Nicolas Heist, Sven Hertling, Daniel Ringler, and Heiko Paulheim. 2020. Knowledge graphs on the web - an overview. *CoRR*, abs/2003.00719.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 105–113, New York, NY, USA. Association for Computing Machinery.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*.

Adrian Kochsiek and Rainer Gemulla. 2021. Parallel training of knowledge graph embedding models: a comparison of techniques. *Proceedings of the VLDB Endowment*, 15(3):633–645.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. *CoRR*, abs/1808.10006.

Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *CoRR*, abs/1909.01066.

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.

Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *Proceedings of* *the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8959–8970. PMLR.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.

Tao Shen, Yi Mao, Pengcheng He, Guodong Long, Adam Trischler, and Weizhu Chen. 2020. Exploiting structured knowledge in text via graph-guided representation learning. *CoRR*, abs/2004.14224.

Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. 2021. Faithful embeddings for knowledge base queries.

Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019a. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *CoRR*, abs/1908.04319.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016a. Representation learning of knowledge graphs with entity descriptions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016b. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. *CoRR*, abs/1808.09582.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embedding. *arXiv preprint arXiv:1904.10281*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.

Zhaocheng Zhu, Shizhen Xu, Meng Qu, and Jian Tang. 2019. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504. ACM.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34.

| | scalability | quality | versatility | simplicity |
|---|---|---|---|---|
| Traditional KGE | | ✓ | | ✓ |
| DKRL | ✓ | | | ✓ |
| KEPLER | ✓ | | | ✓ |
| KG-Bert | | | ✓ | ✓ |
| MLMLM | ✓ | | ✓ | ✓ |
| KGE based KGQA | | ✓ | | |
| KGT5 | ✓ | ✓ | ✓ | ✓ |

Table 9: Comparison of related work in terms of the desiderata described in §1.

## A Textual representations of entities and relations

For Wikidata based datasets we obtain canonical mentions of entities and relations from the corresponding Wikidata page titles (canonical names). However, multiple entities can have identical canonical mentions; we disambiguate such entities by appending the name with their 1-line description if available. In all other cases of identical canonical mentions we extend each mention with a unique id. This results in a one-to-one mapping between entities and their textual representations. For WikiKG90Mv2 we used the entity names and descriptions provided as part of OGB v1.3.2 data dump. For Wikidata5M, these were extracted from a 2019 WikiData dump.

For the Freebase based question answering datasets, such as WQSP and CWQ, we use the *identifier triples* (Chah, 2017) to retrieve mention strings. In particular, we use the canonical name (in English) connected by the relation type `/type/object/name`. Furthermore, we disambiguate similar to the Wikidata based datasets with an alias retrieved via the relation `/common/topic/alias` or append part of the description `/common/topic/description` if available.

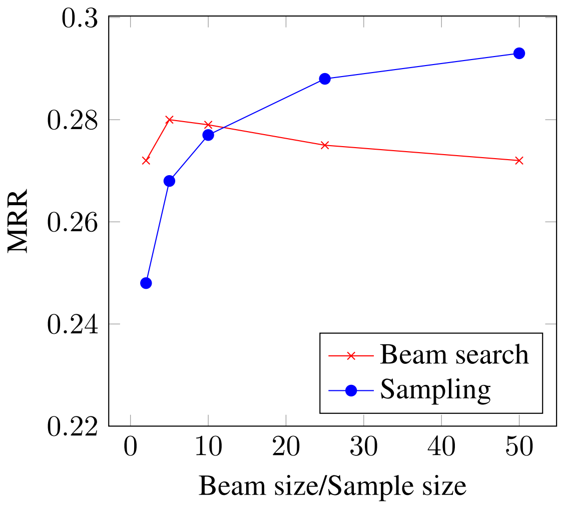

Figure 3: Link prediction performance on Wikidata5M. Increasing the sample size steadily increases MRR for the sampling strategy; the opposite effect is seen with beam size $\geq 5$ and beam search.

## B Teacher forcing

At each step of decoding, the model produces a probability distribution over possible next tokens. While training, this distribution is penalised for being different from the 'true' distribution (i.e. a probability of 1 for the true next token, 0 for all other tokens) using cross entropy loss. In teacher forcing (Williams and Zipser, 1989) the target token is used as the next token during decoding.

An entity usually consists of multiple tokens. Consider an input sequence $input$, target entity mention tokenized as $[w_1, w_2, .., w_T]$ and vocabulary $[v_1, v_2, ..., v_M]$. Then

$$y_{t,c} = \mathbb{1}_{c=w_t}$$

$$p_{t,c} = \mathbb{P}(v_c | input, w_1, w_2, ..., w_{t-1})$$

$$J_t = -\sum_{c=1}^{M} y_{t,c} \log p_{t,c}$$

$$Loss = \frac{1}{T} \sum_{t=1}^{T} J_t$$

where $\mathbb{P}$ is the model's output distribution.

## C Sampling strategy for link prediction

At each step of decoding we get a probability distribution over tokens. We sample a token from this distribution and then autoregressively decode until the 'stop' token. By repeating this sampling procedure multiple times we can get multiple predictions for the same input sequence. The score for a sequence is the sum of log probabilities for

| Model | WN18RR |  |  | FB15k-237 |  |  | YAGO3-10 |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  | **MRR** | **H@1** | **H@10** | **MRR** | **H@1** | **H@10** | **MRR** | **H@1** | **H@10** |
| ComplEx | 0.475 | 0.438 | 0.547 | 0.348 | 0.253 | 0.536 | **0.551** | **0.476** | **0.682** |
| NBFNet (Zhu et al., 2021) | **0.551** | **0.497** | **0.666** | **0.415** | **0.321** | **0.599** | - | - | - |
| KGT5 (Our method) | 0.508 | 0.487 | 0.544 | 0.276 | 0.210 | 0.414 | 0.426 | 0.368 | 0.528 |
| KGT5-ComplEx Ensemble | 0.542 | **0.507** | 0.607 | 0.343 | 0.252 | 0.377 | **0.552** | **0.481** | 0.680 |

Table 10: Link prediction results on small KGs ($\leq$ 150k entities). KGT5 is generally worse than both NBFNet and ComplEx on FB15k-237 and YAGO3-10 datasets. Performance on WN18RR is somewhat better; however a part of this could be due to the use entity definitions (see §4.8). Please see §4.4 for more details.

| **Dataset** | **Train** | **Validation** | **Test** |
|---|---|---|---|
| MetaQA 1-hop | 96,106 | 9,992 | 9,947 |
| MetaQA 2-hop | 118,980 | 14,872 | 14,872 |
| MetaQA 3-hop | 114,196 | 14,274 | 14,274 |
| WQSP | 2,998 | 100 | 1,639 |
| CWQ | 27,639 | 3,519 | 3,531 |

Table 11: Numbers of questions in the KGQA datasets used in our experiments.

| Dataset | Train<br>Questions | Distinct<br>Qtypes | Distinct<br>NL questions | Train<br>QA pairs |
|---|---|---|---|---|
| 1-hop | 96,106 | 11 | 161 | 184,884 |
| 2-hop | 118,980 | 21 | 210 | 739,782 |
| 3-hop | 114,196 | 15 | 150 | 1,521,495 |

Table 12: Statistics for MetaQA QA datasets. Since it is a template-based dataset, there is very little linguistic variation - for each linguistic variation, there are more than 1,000 QA pairs on average in the 1-hop dataset. This is further amplified for 2-hop and 3-hop datasets since there are more correct answers on average per question.

its tokens. For an input sequence $input$, and an entity mention tokenized as $[w_1, w_2, ..., w_T]$, the score for the entity would be

$$\sum_{t=1}^{T} \log(\mathbb{P}(w_t | input, w_1, w_2, ..., w_{t-1}))$$

where $\mathbb{P}$ is the model's output distribution.

Another way to obtain large number predictions could have been beam search (Graves, 2012). This would also have the advantage of being deterministic and guaranteed to produce as many predictions as we want. Although in theory wider beam sizes should give improved performance, it has been observed that for beam sizes larger than 5, performance of generative models suffers drastically (Yang et al., 2018) and sampling generally produces better results. We observe the same phenomenon in our work where beam size 50 produces far worse results than sampling 50 times (fig. 3). Modifying the stopping criteron (Murray and Chiang, 2018) or training method (Welleck et al., 2019) might be helpful solutions that we hope to explore in future work.

## D Path Predictor on MetaQA

Being an artificially generated template-based dataset, MetaQA has far more questions than any other dataset that we compare with (Tab. 11). It also has very little variety in the forms of questions (Tab. 12). Hence we try to answer the following question: Can we create a simple model that maps a NL question to a relation path, and then does KG traversal with this path to answer questions? We achieve this by using distant supervision to get the question → path mapping data, which is then processed to get the final model. We call this model PathPred. *We do not use ground truth queries to create this data.*

A question in MetaQA consists of the question text $q_{text}$, a topic entity $h$ and a set of answers $\{a_1, a_2, ...\}$ (answers only in train set). Since the topic entity annotation is present for all questions (including test set), we can replace the entity in the question to get a base template $q_{base}$.[17]

Given a training tuple of $(q_{base}, h, a)$, we find all the k-hop relation paths $[r_1, .., r_k]$ between $h$ and $a$ (k=1,2 or 3 depending on the dataset). We then aggregate these paths for each distinct $q_{base}$, and take the most frequent path as the mapping from $q_{base}$ to relation path. This mapping from question template $q_{base}$ to a relation path $[r_1, .., r_k]$ constitutes the PathPred model.

For a test question $(q_{text}, h)$, we first get $q_{base}$ from $q_{text}$. We then use the aforementioned map-

[17] As an example given a $q_{text}$ '*who are the co-actors of Brad Pitt*' and topic entity annotation '*Brad Pitt*', we can get a base template $q_{base}$ as '*who are the co-actors of NE*' where *NE* (named entity) is the substitution string.

| Model | MRR | | | | Hits@1 | | | |
|---|---|---|---|---|---|---|---|---|
| | No. of entities to filter | | | All | No. of entities to filter | | | All |
| | 0 | 1 to 10 | >10 | queries | 0 | 1 to 10 | >10 | queries |
| ComplEx | 0.534 | **0.351** | **0.045** | 0.296 | 0.464 | **0.233** | **0.027** | 0.241 |
| KGT5 | **0.624** | 0.215 | 0.015 | 0.300 | **0.567** | 0.164 | 0.011 | 0.267 |

Table 13: For a test query $(s, r, ?)$, there can be multiple entities $o$ such that $(s, r, o)$ is in train set. These entities need to be 'filtered' before evaluation. This table shows model performance on queries requiring different amounts of filtering. Dataset is Wikidata5M. The ComplEx checkpoint used in this analysis is slightly worse than the SOTA.

| Model(s) | MetaQA | | | WQSP | CWQ |
|---|---|---|---|---|---|
| | 1-hop | 2-hop | 3-hop | | |
| Baselines (LEGO, EmbedKGQA, EMQL, PullNet) | 63.3 | 45.8 | 45.3 | 56.9 | 25.2 |
| Ours (KGT5, KGT5 Ensemble) | 67.7 | 48.7 | 44.4 | 56.9 | 24.5 |

Table 14: Percentage of questions answerable using ground truth query. For the baselines that we compare with, we do not have access to the exact same 50% KG split used by them. This table lists the percentage of questions answerable using GT query, for the KGs used by the comparison models (LEGO, EmbedKGQA, EMQL, PullNet) as well as by our models (KGT5, KGT5 + PathPred Ensemble). The GT query numbers for baselines were made available by Ren et al. 2021.

ping to get a relation path using $q_{base}$. This relation path is then used to traverse the KG starting from $h$ to arrive at the answer(s).

In the KGT5 + PathPred ensemble (§4.5, Tab. 5), we first apply the PathPred technique; if the resulting answer set is empty – which can happen due to KG incompleteness – we apply KGT5 to get the answer.

| Question type | GTQ | KGT5 | Gain |
|---|---|---|---|
| actor→movie | 0.96 | 0.95 | -0.01 |
| director→movie | 0.84 | 0.92 | 0.08 |
| movie→actor | 0.79 | 0.77 | -0.02 |
| movie→director | 0.52 | 0.64 | 0.12 |
| movie→genre | 0.48 | 0.63 | 0.15 |
| movie→language | 0.49 | 0.63 | 0.14 |
| movie→tags | 0.72 | 0.7 | -0.02 |
| movie→writer | 0.66 | 0.8 | 0.14 |
| movie→year | 0.46 | 0.45 | -0.01 |
| tag→movie | 1 | 0.96 | -0.04 |
| writer→movie | 0.88 | 0.94 | 0.06 |
| All | 0.678 | 0.732 | 0.054 |

| Question type | GTQ | KGT5 | Gain |
|---|---|---|---|
| actor→movie→director | 0.44 | 0.39 | -0.05 |
| director→movie→director | 0.34 | 0.62 | 0.28 |
| director→movie→language | 0.37 | 0.77 | 0.4 |
| writer→movie→writer | 0.39 | 0.39 | 0 |
| actor→movie→genre | 0.48 | 0.55 | 0.07 |
| director→movie→genre | 0.46 | 0.7 | 0.24 |
| actor→movie→actor | 0.57 | 0.09 | -0.48 |
| writer→movie→actor | 0.51 | 0.31 | -0.2 |
| actor→movie→writer | 0.48 | 0.44 | -0.04 |
| movie→director→movie | 0.45 | 0.21 | -0.24 |
| actor→movie→year | 0.48 | 0.23 | -0.25 |
| writer→movie→genre | 0.4 | 0.59 | 0.19 |
| director→movie→actor | 0.51 | 0.5 | -0.01 |
| movie→actor→movie | 0.73 | 0.06 | -0.67 |
| writer→movie→year | 0.37 | 0.35 | -0.02 |
| director→movie→year | 0.45 | 0.51 | 0.06 |
| director→movie→writer | 0.47 | 0.44 | -0.03 |
| movie→writer→movie | 0.5 | 0.3 | -0.2 |
| writer→movie→director | 0.33 | 0.31 | -0.02 |
| writer→movie→language | 0.32 | 0.66 | 0.34 |
| actor→movie→language | 0.4 | 0.54 | 0.14 |
| All | 0.471 | 0.363 | -0.108 |

Table 15: Hits@1 performance on MetaQA 1-hop (left) and 2-hop (right) validation dataset, 50% KG setting. GTQ refers to ground truth querying.

| Question type | GTQ | KGT5 | Gain |
|---|---|---|---|
| movie→director→movie→language | 0.17 | 0.85 | 0.68 |
| movie→director→movie→actor | 0.37 | 0.54 | 0.17 |
| movie→actor→movie→language | 0.29 | 0.8 | 0.51 |
| movie→writer→movie→year | 0.31 | 0.47 | 0.16 |
| movie→actor→movie→director | 0.65 | 0.57 | -0.08 |
| movie→director→movie→genre | 0.37 | 0.82 | 0.45 |
| movie→writer→movie→director | 0.4 | 0.52 | 0.12 |
| movie→actor→movie→year | 0.63 | 0.72 | 0.09 |
| movie→actor→movie→writer | 0.63 | 0.51 | -0.12 |
| movie→actor→movie→genre | 0.65 | 0.83 | 0.18 |
| movie→director→movie→writer | 0.39 | 0.55 | 0.16 |
| movie→writer→movie→genre | 0.42 | 0.75 | 0.33 |
| movie→writer→movie→actor | 0.41 | 0.43 | 0.02 |
| movie→director→movie→year | 0.32 | 0.56 | 0.24 |
| movie→writer→movie→language | 0.27 | 0.74 | 0.47 |
| All | 0.443 | 0.634 | 0.191 |

Table 16: Hits@1 performance on MetaQA 3-hop validation dataset, 50% KG setting. GTQ refers to ground truth querying.