GETT-QA: Graph Embedding Based T2T Transformer for Knowledge Graph Question Answering

Debayan Banerjee[1(✉)], Pranav Ajit Nair[2], Ricardo Usbeck[1],
and Chris Biemann[1]

[1] Universität Hamburg, Hamburg, Germany
{debayan.banerjee,ricardo.usbeck,chris.biemann}@uni-hamburg.de
[2] Indian Institute of Technology (BHU), Varanasi, Varanasi, India
pranavajitnair.cse18@itbhu.ac.in

Abstract. In this work, we present an end-to-end Knowledge Graph Question Answering (KGQA) system named GETT-QA. GETT-QA uses T5, a popular text-to-text pre-trained language model. The model takes a question in natural language as input and produces a simpler form of the intended SPARQL query. In the simpler form, the model does not directly produce entity and relation IDs. Instead, it produces corresponding entity and relation labels. The labels are grounded to KG entity and relation IDs in a subsequent step. To further improve the results, we instruct the model to produce a truncated version of the KG embedding for each entity. The truncated KG embedding enables a finer search for disambiguation purposes. We find that T5 is able to learn the truncated KG embeddings without any change of loss function, improving KGQA performance. As a result, we report strong results for LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets on end-to-end KGQA over Wikidata.

1 Introduction

A Knowledge Graph (KG) is an information store where data is stored in the form of node-edge-node triples. Nodes represent entities and edges represent relationships between these entities. The aim of KGQA [21] is to produce answers from this KG given an input question in natural language, e.g., `Who is the father of Barack Obama ?`. Usually, the first step in KGQA is to perform Entity Linking (EL) where mention spans, e.g., `Barack Obama` representing the name of a person, place, etc., are linked to a KG node. The subsequent step is Relation Linking (RL), where the relationship of the entity to the potential answer in the KG is extracted, e.g., `father of`. Some KGQA systems attempt to fetch the answer based on the results of just the two steps above, which typically ends up being another entity (node) in the graph. However, for more complex questions, such as count queries or min/max aggregate queries (e.g.: `How many rivers are there in India?`) the answer does not lie in a node or edge in the

graph, but instead, a formal query must be generated as a final step. To this end, semantic parsing is relevant to the problem of KGQA. Thus, our focus in this work is to generate a final SPARQL query that can be executed on the KG.

SPARQL is a popular graph query language for querying KGs. A sample SPARQL query for the running example over the Wikidata KG looks like the following:

```
SELECT ?o WHERE { wd:Q76 wdt:P22 ?o }
```

In the query above, `wd:Q76` stands for Barack Obama, while `wdt:P22` stands for the relation `father`. The `?o` variable represents the answer from the KG.

Recent works employ text-to-text (T2T) pre-trained language models (PLMs) for generating logical queries, e.g. SPARQL, from natural language questions. If the correct entity and relation IDs are already specified in the input, the accuracy of T2T models is high [2]. However, the absence of linked entity and relation IDs in the input presents a significant challenge to such models. PLMs are adept at generating linguistic tokens from within their weights. Yet, it is an entirely different proposition to query the KG and ground the entity and relations to specific IDs, as the variability of language creates impressive richness at generation while at the same time hampers the alignment to pre-defined KG items.

In this work, we demonstrate a novel method by which a T2T PLM, namely T5 [28], not only generates SPARQL queries, but also generates truncated KG embeddings, which aid in the subsequent process of grounding entities to the correct node in the KG. Our method produces strong results for end-to-end Question Answering on the LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets over Wikidata KG. All code and data will be made available[1].

2   Related Work

Early KGQA systems could be divided on the basis of whether they can handle simple [44] or complex questions [21]. In a simple question, a node-edge-node triple is a sole basis on which a question is formed, whereas in a complex question there may be more than one such triple involved. Moreover, certain KGQA systems are built specifically to handle a certain class of questions better, e.g. temporal questions [18].

Another way of categorising KGQA systems is whether they form a formal query [1,3,8,35,39] versus whether they use graph search based methods without producing an explicit query [7,16,24,29,33,36,37,40].

Some KGQA systems work in a hybrid mode and can query from both KG and text-based sources. PullNet [36] and Graftnet [37] both use Relational-Graph Convolution Networks [34] to handle complex questions. UNIK-QA [25] verbalises all structured input from KG, tables and lists into sentences and adds them to a text corpora and proceeds to perform QA over this augmented text

---

corpora using deep passage retrieval techniques. UNIQORN [27] builds a context graph on-the-fly and retrieves question relevant pieces of evidence from KG and text corpora, using fine-tuned BERT models. They use group Steiner trees to identify the best answer in the context graph. We use the results of the KG components of these hybrid systems in our evaluation in Table 3, as reported by UNIQORN.

Platypus [39] and QAnswer [8] are two recent KGQA systems that work on Wikidata. Both of them use templates and ranking mechanisms to find the best query. We make no use of templates in our method since this inherently limits the flexibility of a system on unseen templates.

ElneuQA-ConvS2S [10] operates in a similar fashion to us, where they use a Neural Machine Translation (NMT) model to generate SPARQL queries with empty placeholders, while an entity linking and sequence labeling model fills the slots. In our case we also make use of NMT capabilities of T5 to generate a skeleton SPARQL query, however, we do not generate empty slots, and instead, generate entity and relation labels to be grounded later.

For simple questions, KEQA [17] targets at jointly recovering the question's head entity, predicate, and tail entity representations in the KG embedding spaces and then forming a query to retrieve the answer from a KG. Text2Graph [6] uses KEQA as a base, and improves on the embedding learning model by utilising CP tensor decomposition [15]. We include both these systems in our evaluation Table 4.

SGPT [31] and STAG [29] both use generative methods for forming the query using pre-trained language models, which is similar to what we do, however, neither of them generate the entity or relation label, or the embeddings. Instead STAG uses an external entity linking step, while SGPT attempts to generate entity and relation IDs directly. However such a method does not work well because for a KG like Wikidata, the IDs do not follow a hierarchical pattern, and hence the model is not able to predict an ID that it has not seen in training earlier.

One of our key ideas is to enable a PLM to learn KG Embeddings. There have been some recent efforts in the same direction such as KEPLER [43], K-BERT [23], KI-BERT [12], CoLAKE [38], BERT-MK [14] and JAKET [45]. These systems either try to inject KG embeddings into the input or intermediate layers of the model, or they try to augment the text corpora by including verbalised forms of the triple structural information. On the other hand, we ask the model to print the embeddings as output. This is a fundamentally different approach from what has been tried so far.

A related, yet different class of systems is that of conversational QA.

LASAGNE [19] and CARTON [26] are two notable systems in this category. They evaluate on the CSQA dataset [32], which is a conversational dataset answerable over a KG. In our case, we address only single sentence-long questions. The conversations in CSQA are arranged in sequence of turns of questions and answers. For the semantic parsing of logical forms, both LASAGNE and CARTON use a pre-defined grammar, while our approach is free of templated

grammar rules. Both LASAGNE and CARTON use a Transformer architecture to generate base logical forms, however, LASAGNE uses, a Graph Attention Network [42] to resolve entities and relations while CARTON uses stacked pointer networks.
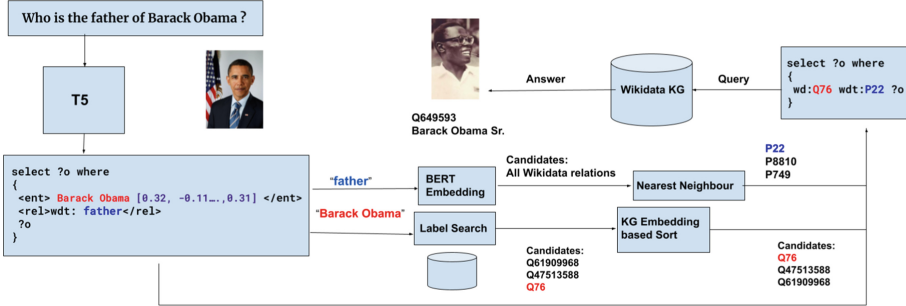

Fig. 1. Architecture of GETT-QA: T5 generates a skeleton SPARQL query with entity and relation labels and a truncated KG embedding string. The entity labels are grounded using label based search and embedding based sorting, while the relations are grounded using BERT embedding based sorting. The final query is executed against a KG to fetch the answer.

3 Method

As shown in Fig. 1, our system consists of five major steps:

- T5 generates a skeleton SPARQL query from input natural language question.
- The entity labels and truncated KG embeddings are extracted. A label search is performed to fetch entity candidates.
- The entity candidates are re-ranked using an embedding similarity metric.
- In parallel, the relation label is extracted and matched against Wikidata relations based on BERT embeddings.
- The final query is grounded to KG nodes and executed to receive the answer.

3.1 Truncated KG Embeddings

We teach T5 to generate truncated vector strings of KG embeddings. We use TransE [5] embeddings for Wikidata entities that were provided by Pytorch-BigGraph[2] [22]. These are 200-dimensional vectors of floats. The truncated KG embeddings we use are a shorter version of the same embeddings. For most of our experiments, we use the first 10 dimensions of these embeddings, and further reduce the precision of the floats to 3 digits after the decimal. We do so since T5

---

[2] https://github.com/facebookresearch/PyTorch-BigGraph.

is expected to produce these truncated KG embeddings while still in the text-to-text mode. In other words, T5 produces these vectors of floats considering them as a string. We use truncated KG embeddings instead of original embeddings to reduce the decoding load of T5. Our aim is not to learn the entire embedding space. Instead, we want to learn identifiers that can aid the entity disambiguation phase. We produce these truncated KG embeddings only for entities, not for relations.

3.2   Intuition

The initial idea behind our approach is to allow T5 to use its significant linguistic capability derived from pre-training over a large corpus for the task of semantic parsing. As a result, we use T5 to produce SPARQL tokens and entity and relation labels.

At first glance, it may appear that the production of entity labels is sufficient for grounding to KG entity nodes. However, in most KGs, several entities share the same labels. For example in Wikidata KG, the entity IDs `Q76` and `Q47513588` both share the label `Barack Obama`. In reality, `Q76` represents the President while `Q47513588` is the entity ID for a painting of the President. As a result of such collision of labels, a further step called *disambiguation* is required.

The next idea is to not just rely on T5's linguistic abilities but also to try and teach the model how to generate identifiers for the entities, which can aid the grounding and disambiguation process in a subsequent step. One way could be to generate the Wikidata IDs directly. However, the IDs do not correspond in any hierarchical way to the underlying entities. For example, while `Q76` is `Barack Obama`, `Q77` is `Uruguay`. Although the IDs are close to each other, the categories are completely different. Models cannot be expected to produce accurate IDs of this nature, especially on unseen input questions. As a result, we consider other schemes of entity identifiers that exhibit some hierarchical properties.

It turns out KG embeddings fulfill these requirements handsomely, and hence we decide to use truncated KG embeddings as the "soft" identifier of our choice. Another possibility would be to generate entity descriptions instead of truncated KG embeddings, however, roughly 50% of entities in Wikidata do not have corresponding descriptions (e.g.: Q67395985[3]), hence we focus on generating truncated KG embeddings instead.

While the production of such truncated embeddings may also aid the grounding of relations, we do not attempt this, since Wikidata only contains a few thousand relations, while the number of entities run into several millions. For the grounding of relations we use simpler text embedding based methods, as described later in Sect. 3.7.

3.3   Models

T5 [28], or text-to-text transfer transformer, is a transformer [41] based encoder-decoder architecture that uses a text-to-text approach. Every task, including

---

translation, question answering, and classification, is cast as feeding the model text as input and training it to generate some target text. This allows using the same model, loss function, hyper-parameters, etc., across a diverse set of tasks. We experiment with fine-tuning two variants of the T5 model: `T5-Small` with 60 million parameters and `T5-Base` with 220 million parameters. For the GETT-QA system results reported in Tables 3 and 4 we use the `T5-Base` based model, whereas in the analysis Sect. 7.2 we present a comparative study against `T5-Small`.

3.4   Skeleton SPARQL

As shown in Fig. 1, the first step of our KGQA system is to generate a skeleton SPARQL query from the given natural language question. The skeleton query consists of SPARQL tokens such as `SELECT, WHERE,` {}, entity and relation labels, and truncated KG embeddings, which are an array of `float`s. Some additional tokens are used to surround the entity and relation labels, such as `<ent>, </ent>,<rel>,</rel>` so that in a later step their extraction can be performed using regular-expression operations. The extraction of the labels and the truncated KG embedding are essential for the subsequent grounding step. Notably, entity and relation IDs are not a part of a skeleton SPARQL query.

During training via fine-tuning, pairs of questions and skeleton SPARQL queries are presented to T5. For this purpose, we pre-process the original dataset, which contains gold SPARQL queries for each question. The SPARQL query is converted to a skeleton SPARQL query by replacing the entity and relation IDs with their gold labels while appending the entity labels with a truncated KG embedding. Hence, the following gold SPARQL query:

```
select ?o where {  wd:Q76 wdt:P22 ?o }
```

is converted to:

```
select ?o where
{
 <ent>Barack Obama [...] </ent>
 <rel>father</rel>
 ?o
}
```

for the purposes of training T5, where the truncated KG embedding is represented by `[...]`.

3.5   Entity Candidates

During inference, when T5 generates a skeleton query, all entity and relation labels, as well as truncated KG embeddings, are extracted using regular expressions. For the entity labels, a BM-25-based [30] label search is performed on a

database of all Wikidata entity labels, out of which top-`k` candidates are retrieved per entity label. For this text search we use the Elasticsearch database[4]. For our experiments, we fix `k` at 100.

3.6 Entity Candidates Re-ranking and Ordering

The top-3 entity candidates based on label matching are retained. For the next 3 candidates, we resort to truncated KG embedding-based sorting. For each item in the list of 100 entity candidates fetched, we also fetch their gold KG embeddings, and convert them to truncated KG embeddings. For the truncated KG embedding generated by T5, we compute its dot product against the gold truncated KG embeddings fetched and re-rank them in descending order. The dot product is used as a comparator because this was the same function that was used during the production of the TransE embeddings. From this re-ranked list based on truncated KG embedding similarity, top-3 candidates are retained.

We append these `top-3` truncated KG embedding-sorted candidates to the `top-3` label-sorted candidates, and proceed to the subsequent steps with a list of 6 candidate entities for each entity label.

3.7 Relation Candidates

We generate no truncated KG embeddings for relation IDs, as their numbers are orders of magnitudes smaller when compared to entities in Wikidata. From the relation labels generated by T5, we compute their BERT embeddings and compute the cosine similarity against the BERT embeddings of all Wikidata properties. The list of properties is sorted based on this similarity score, and the top-3 matches are considered for the subsequent steps.

3.8 Candidate Combinations

For a generated query, each entity label and each relation label has 6 and 3 candidates each, respectively. We preserve the serial order of the entities and relations as produced in the query, and generate all possible combinations of the entities and relations, which generates several queries of the same structure but different entity and relation IDs. For example, if the query contains just one entity and one relation, the number of possible SPARQL queries generated would be $6 \times 3 = 18$. We execute each query on the KG in sorted order of entity and relation IDs received in previous steps. We stop when the KG returns a non-empty response. This response is considered the output of our KGQA system. We consider the top 3 beams produced by T5 decoder as probable queries. The first beam producing a valid response from the KG is considered the output of our KGQA system.

---

[4] https://www.elastic.co/.

4   Dataset

We evaluate our approach on the LC-QuAD 2.0 [11] dataset, which consists of approximately 30,000 questions based on the Wikidata KG. Each question contains the corresponding SPARQL query as gold annotation. The dataset consists of a wide variety of questions, such as simple, complex, multi-hop, count, min/max, dual and boolean types. This dataset also uses the recently introduced hyper-relational [13] structure of the Wikidata KG.

Additionally, we evaluate our approach on the SimpleQuestions-Wikidata [9][5] dataset, which consists of 34,374 train questions and 9,961 test questions. This dataset is derived from the original SimpleQuestions dataset [4], which was later aligned with the Wikidata KG. A sample question from each dataset can be seen in Tables 1 and 2.

Table 1. Sample question from LC-QuAD 2.0

Question	SPARQL
Tell me the female beauty pageant that operates in all countries and contains the word model in it's name?	```SELECT DISTINCT ?sbj ?sbj_label WHERE { ?sbj wdt:P31 wd:Q58863414 . ?sbj wdt:P2541 wd:Q62900839 . ?sbj rdfs:label ?sbj_label . FILTER(CONTAINS(lcase(?sbj_label), "model")) . FILTER (lang(?sbj_label) = "en") } LIMIT 25```

Table 2. Sample question from SimpleQuestions-Wikidata

Question	SPARQL
What type of music does David Ruffin play?	`SELECT ?x WHERE { wd:Q1176417 wdt:P136 ?x }`

5   Evaluation

In Table 3, the results for UNIQORN, QAnswer, UNIK-QA, Pullnet, and Platypus are taken from UNIQORN [27]. UNIQORN uses a test split of 4,921 questions from the original LC-QuAD 2.0 test set of 6,046 questions for all the systems. We evaluate our approach on the same split as UNIQORN. Despite our best efforts we were unable to acquire the precise KG snapshot that UNIQORN used for evaluation. UNIQORN used a Wikidata dump dated 20 April 2021, which is no longer available either in the official Wikidata repository[6], or with the authors

---

[5] https://github.com/askplatypus/wikidata-simplequestions.
[6] https://dumps.wikimedia.org/wikidatawiki/entities/.

of UNIQORN. As a result, we ran the 4,921 questions against the NLIWOD[7] Wikidata dump[8,9], which is hosted on the docker hub for easy deployment, and also hosted as an API by the Universität Hamburg's SEMS group.

In Table 4 for SimpleQuestions-Wikidata, results for KEQA and Text2Graph are taken from MEKER [6]. They evaluate both systems on a smaller split of the SimpleQuestions-Wikidata test set. This subset contains those questions which are valid on a custom Wikidata version they call Wiki4M. We were provided the KG by the authors of MEKER and we evaluated our system on the same.

In Table 3, we report macro Precision@1 based on UNIQORN's reporting preference. In Table 4 we report macro F1 in line with MEKER. To compute metrics, we take the gold SPARQL and predicted SPARQL query and query the KG with both. We compare the results from the KG to compute true positives, false positives, and false negatives (TP, FP, FN).

Table 3. Results on LC-QuAD 2.0

	P@1
UNIK-QA	0.005
Pullnet	0.011
Platypus	0.036
QAnswer	0.308
UNIQORN	0.331
GETT-QA without truncated embeddings	0.327 ± 0.002
GETT-QA (with truncated embeddings)	$\mathbf{0.403} \pm 0.0$

Table 4. Results on SimpleQuestions-Wikidata

	F1
KEQA	0.405
Text2Graph	0.618
GETT-QA without truncated embeddings	0.752 ± 0.004
GETT-QA (with truncated embeddings)	$\mathbf{0.761} \pm 0.002$

6   Results

In Table 3, the bottom two rows contain the results of our system in two different settings for LC-QuAD 2.0. In the first case, our KGQA system uses the top-6 entity candidates based on label match, without the use of truncated KG

---

[7] https://www.nliwod.org/challenge.
[8] https://hub.docker.com/r/debayanin/hdt-query-service.
[9] https://skynet.coypu.org/#/dataset/wikidata/query.

embeddings for re-ranking. In the second case, we keep the top-3 entity candidates based on label match and append to it the top-3 candidates based on truncated KG embedding match. This is the same setting as described in Subsect. 3.6. The relation candidates in both cases remain top-3 as described in Subsect. 3.7.

The key finding in Table 3 is that when we compare the last two rows, our system performs better with an absolute gain of approximately 8% when truncated KG embeddings are used.

In Table 4, for SimpleQuestions-Wikidata our method without truncated KG embeddings already outperforms the nearest competitor by an absolute margin of 13%. This demonstrates the natural ability of T5 to predict the correct entity and relation labels given a question. Since the query structure of all the questions in this dataset is the same, no challenge is posed to T5 in trying to learn the query itself. It is noteworthy however, that after the inclusion of truncated KG embedding re-ranking, the performance remains similar. An insignificant margin of absolute improvement of 0.9% is seen. To investigate this gap when compared to the 8% improvement of LC-QuAD 2.0, we delve further into the nature of the two datasets and run some analysis. We find that in the case of LC-QuAD 2.0, the correct entity is found in the top-1 position of the candidate list based on text match 60% of the time, whereas in the case of SimpleQuestions-Wikidata, this number is significantly higher at 82%. This is so because the questions in SimpleQuestions-Wikidata contain the entity names in almost the exact form as their gold entity annotations, whereas in LC-QuAD 2.0, several entity labels are modified, misspelled or shortened by the human annotators. Hence, in the case of SimpleQuestions-Wikidata, label-only matching is in most cases sufficient, whereas in LC-QuAD 2.0, truncated KG embedding-based disambiguation holds greater importance.

Additionally, as mentioned in Sect. 2, STAG [29] and ElNeuQA-ConvS2S [10] are comparable generative KGQA systems. For STAG, no code, data or KG versions have been made public, while for ElNeuKGQA we were unable to run their code[10] as no instructions on how to run the code exists. For STAG on SimpleQuestions [4], on a test split of 2280 questions they report F1 61.0 while we report F1 78.1 on a larger test split of 9961 questions. Lastly, ElNeuQA-ConvS2S reports an F1 of 12.9 on WikidataQA while we report 17.8. WikidataQA is a 100 question test-subset created by the authors of ElNeuQA-ConvS2S. On LC-QuAD 2.0, they report F1 of 26.9 while we report 40.3.

6.1   Limitations

Although GETT-QA performs the best in Tables 3 and 4, we do not claim state-of-the-art results on the respective datasets. This is due to a variety of reasons: as mentioned in Subsect. 5, we could not procure the precise Wikidata KG version as the competing systems for LC-QuAD 2.0. In the case of LC-QuAD 2.0 and SimpleQuestions-Wikidata, evaluation was performed on a truncated subset of

---

[10] https://github.com/thesemanticwebhero/ElNeuKGQA.

the original test split of this dataset. As a result we can not claim that we have the best results on the entire dataset. Additionally, we could not find the code for, or run majority of systems we evaluated against, and hence resorted to using the results as reported by them.

7 Analysis

7.1 Error Analysis

In an attempt to find the common source of errors in LC-QuAD 2.0 we find that by far the largest cause of incorrect answers is the improper grounding of entities and relations to nodes in the KG. More than 95% of questions where the correct entities and relations were in the top-6 and top-3 candidates respectively, the right answer was eventually produced by the KG. Unfortunately, only in 41% of the questions, the correct entities and relations were found within the top-k candidates. This suggests that greater focus in the area of entity and relation linking will produce better results. One may also increase the size of k, at the cost of increased run time.

Less than 1% of the queries generated had incorrect truncated KG embedding length (e.g.: 11 instead of 10) however these were handled in code appropriately. Less than 1% of queries generated were improper SPARQL, where a critical keyword was missing rendering the query syntactically incorrect. This suggests that T5 learns how to generate valid SPARQL queries to a large extent. This is consistent with the findings of [2] where T5 crosses 90% accuracy when provided with grounded entity and relation IDs with their labels.

To explore the issue of lack of correct entities and relations in candidate lists, we observe that while 60% of questions contain the correct entity ID in the top-100 label-based search candidates, by the time we reduce this list to top-6, only 49% questions remain with the correct entity ID in the candidates list. On the other hand, for relation candidates, 45% of questions contain the correct relation IDs in the top-3 relation candidates.

When looking at the category of questions that return incorrect KG responses, irrespective of entity and relation grounding, we find that `COUNT` queries are the most common. This happens due to a quirk in SPARQL format. If a `COUNT` query is built around a family of triples that do not exist in the KG, the KG responds with `count = 0`, instead of producing a `NULL` value or an `ERROR`. This means that the very first query to be executed on the KG with the correct `COUNT` SPARQL syntax will return a valid response, even if it is `count = 0` and we no longer explore subsequently ranked queries.

In the case of SimpleQuestions-Wikidata, all errors are either due to incorrect entity or relation linking. The model produced an accuracy of 70% for entity linking and 94% for relation linking.

7.2 Truncated KG Embedding Learning

We discover that T5 is able to produce a vector of floats while still in the text-to-text mode of decoding. For this functionality, no change of loss function or
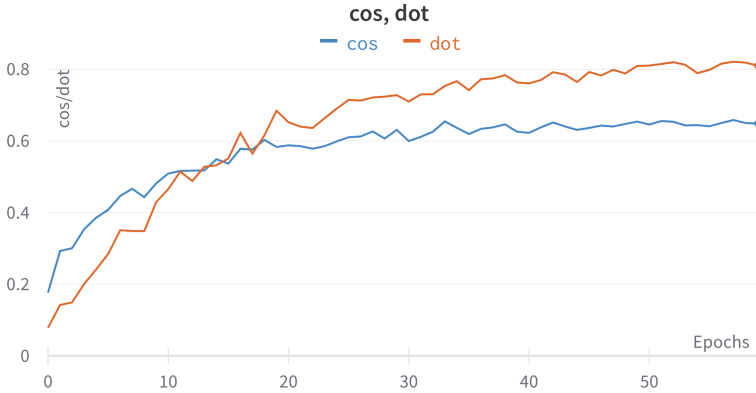
Fig. 2. Cosine and Dot Product based similarities of truncated KG embeddings

decoding scheme is made in our experiments. It effectively learns a simplified embedding space, but with certain limitations. To further explore this ability of T5, we performed some additional experiments on the 200 questions dev set of LC-QuAD 2.0 with `T5-Small`.

In Fig. 2, we compare how the model learns the embedding space with each epoch of training. The TransE embeddings have an angular component and a magnitude component. Since the dot operation was used to train the original embeddings, the magnitude of each embedding may be greater than 1. It appears in Fig. 2 that around step 20 the angular component of the embeddings has been learned by T5 to the best of its abilities, and it proceeds to learn the magnitude component (denoted by the orange line) further.
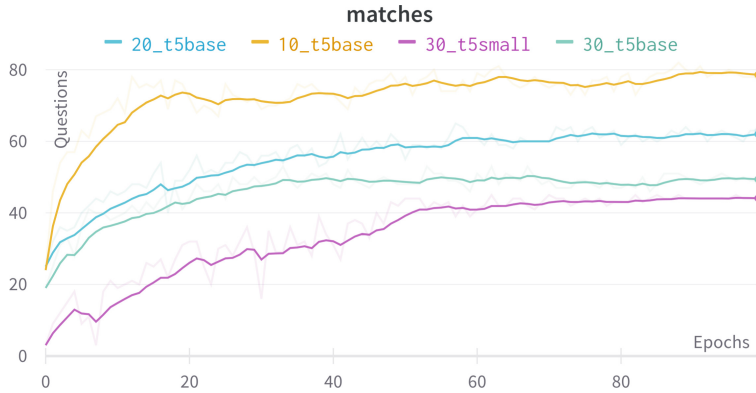

Fig. 3. Dev Set matches for varying truncated KG embedding lengths (Color figure online)
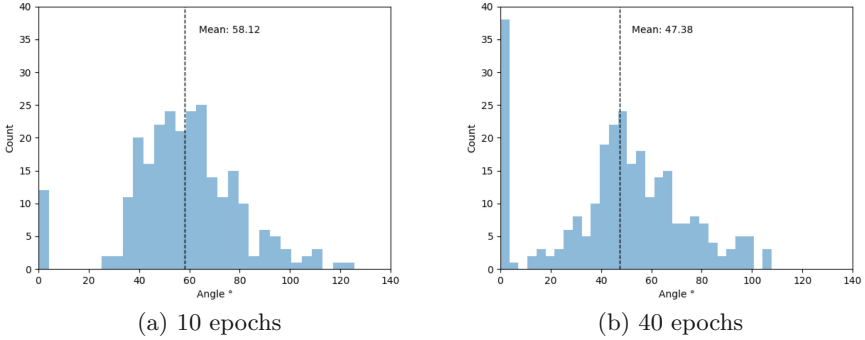
(a) 10 epochs   (b) 40 epochs

Fig. 4. Distribution of angular difference between gold and predicted truncated KG embeddings on LC-QuAD 2.0 dev set. The mean angular difference can be seen reducing as the epochs progress, suggesting that the model is learning the embedding space.

In Fig. 3 we chart the LC-QuAD 2.0 dev set performance of `T5-Small` and `T5-Base` in varying truncated KG embedding lengths. For `T5-Small` (pink line) we set the truncated KG embedding length at 30, so this should only directly be compared against `T5-Base` with truncated KG embedding length 30 (green line). In the matches metric, which only looks at the keywords and labels produced in the skeleton SPARQL query (truncated KG embeddings have been removed from the generated query), the two reach similar performance. This suggests that a larger number of parameters helps learn the embedding space better, but for the textual component the extra number of parameters of `T5-Base` remain unused.

With a truncated KG embedding length of 10 (yellow line), we see the best label match accuracy and hence we persist with this family of models for reporting results in Table 3.

In Fig. 4, we plot the distribution of the angular difference between the gold and predicted truncated embeddings on the LC-QuAD 2.0 dev set. The model seems to learn the embedding space in two distinctly different manners: firstly, for several entities it is able to print the exact embedding, with an angular difference of 0°. Secondly, for the entities which it is unable to learn the embedding of exactly, it produces a more familiar distribution, where the mean shifts every few epochs, reducing the angular difference. This suggests that the model is learning the embedding space effectively.

Table 5. Effects of ordering entity candidates differently. LS = Label sorted, TS = Truncated KG embedding sorted

	F1
3 LS + 3 TS	0.365
3 TS + 3 LS	0.331
3 LS + 0 TS	0.289
0 LS + 3 TS	0.236
6 LS + 0 TS	0.319
0 LS + 6 TS	0.256

7.3   Candidate Ordering

As mentioned in Subsect. 3.6, our results reported in Table 3 come from a configuration of our system where the entity candidates are layered in two parts: the first three candidates are sorted based on label match, while the bottom three candidates are sorted based on the truncated KG embedding dot product similarity. It is observed in Table 5 that the ordering of these two categories affects the eventual accuracy strongly. We take 200 questions at random from the LC-QuAD 2.0 test set and perform experiments to ascertain how the ordering of candidates affects accuracy. In the first row of the table, three entity candidates based on label sorting are followed by three candidates of truncated KG embedding sorting, while the next row of the table shows the result when we keep truncated KG embedding-sorted candidates above label-sorted candidates. The results show that keeping truncated KG embedding-sorted candidates at the top reduces accuracy, and hence, label-based matching for entities remains a stronger mode of fetching correct candidates. This is no surprise, since not all labels have multiple entity candidates requiring disambiguation. Kindly note that the accuracy drops because an earlier query formed due to candidate combinations (as explained in Subsect. 3.8) returns a non-empty result from the KG and this response turns out to be incorrect.

In the next two rows, we see how excluding either label-sorted candidates, or truncated KG embedding-sorted candidates entirely affects accuracy. Once more we see that label-sorted candidates still perform better when used in isolation. However, the crux of the table is the first row, i.e., when both the categories are appropriately sequenced and used in tandem, the accuracy is best.

In the bottom two rows, we see the effect of changing the number of candidates. It is no surprise that increasing k from 3 to 6 increases accuracy since more correct entities are included in the list. However, some systems also perform worse in such settings as the noise may increase adding to the disambiguation load. In our system, this does not seem to be the case.

Increasing the value of k further imposes a large cost on the run time of the system affecting the user experience adversely. Since the candidate combination step has exponential complexity, which depends on the number of entities and

relations in a query, we need to keep the number of candidates in check. Too many candidates will produce too many SPARQL queries and the user must wait for all of them to be executed on the KG till one of them responds validly.

In our choice of setting `k=6` for entities and `k=3` for relations, we observe that on the test set of LC-QuAD 2.0, our system has an average response time of $1.2\,\mathrm{s}$ per question, which from a user experience perspective seems like an acceptable response time.

8   Hyperparameters and Hardware

For the evaluation of LC-QuAD 2.0 in Table 3, we fine-tune our models for 50 epochs with a learning rate of `1e-04` with the Adam optimizer [20]. For SimpleQuestions in Table 4 we fine-tune for 25 epochs, roughly half of LC-QuAD 2.0, since the train set is roughly twice as large as LC-QuAD 2.0. We use a batch size of 8. During this phase we had access to `NVIDIA GeForce RTX 2080 Ti/1080 Ti` graphics cards with approximately 11 GB of video memory. We do not fix a seed during training, and train and infer three times. We report mean and standard deviation for the three runs in the respective tables.

For the analysis in Sect. 7, we fine-tune our models for 100 epochs with a learning rate of `1e-05` with the Adam Optimizer [20] and we use a batch size of 20. During this phase we had access to larger GPUs, namely `NVIDIA RTX A6000` with 48 GB of memory and RTX A5000 with 24 GB memory.

9   Conclusion and Future Work

In this work, we presented a novel KGQA pipeline called GETT-QA. We use no external entity or relation linking tool, and still achieve strong results on LC-QuAD 2.0 and SimpleQuestions-Wikidata datasets. Additionally, we discover the ability of T5 to learn KG embeddings. We demonstrate that in certain situations this ability helps in better question answering performance.

In future work, we will explore the ability of T5 in generating similar truncated KG embedding based queries with modified loss functions and a customised architecture towards the penultimate layers of the models, so that embeddings can be generated with more standard loss functions meant specifically for learning embeddings. Additionally, suitable identifiers other than embeddings can also be explored, for example, text description based identifiers.

References

1. Abujabal, A., Roy, R.S., Yahya, M., Weikum, G.: Never-ending learning for open-domain question answering over knowledge bases. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, pp. 1053–1062. ACM (2018). https://doi.org/10.1145/3178876.3186004
2. Banerjee, D., Nair, P., Kaur, J.N., Usbeck, R., Biemann, C.: Modern baselines for SPARQL semantic parsing. In: SIGIR 2022: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid. ACM (2022). https://arxiv.org/abs/2204.12793
3. Bhutani, N., Zheng, X., Jagadish, H.V.: Learning to answer complex questions over knowledge bases with query composition. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, pp. 739–748. ACM (2019). https://doi.org/10.1145/3357384.3358033
4. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. CoRR abs/1506.02075 (2015). https://dblp.uni-trier.de/db/journals/corr/corr1506.html#BordesUCW15
5. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, Nevada, United States, pp. 2787–2795 (2013). https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html
6. Chekalina, V., Razzhigaev, A., Sayapin, A., Frolov, E., Panchenko, A.: MEKER: memory efficient knowledge embedding representation for link prediction and question answering. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, Dublin, Ireland, pp. 355–365. Association for Computational Linguistics (2022). https://aclanthology.org/2022.acl-srw.27
7. Christmann, P., Roy, R.S., Abujabal, A., Singh, J., Weikum, G.: Look before you hop: conversational question answering over knowledge graphs using judicious context expansion. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, pp. 729–738. ACM (2019). https://doi.org/10.1145/3357384.3358016
8. Diefenbach, D., Both, A., Singh, K.D., Maret, P.: Towards a question answering system over the semantic web. Semant. Web Interoperability Usability Applicability **11**(3), 421–439 (2020). https://hal.archives-ouvertes.fr/hal-02013956
9. Diefenbach, D., Tanon, T.P., Singh, K.D., Maret, P.: Question answering benchmarks for Wikidata. In: Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks Co-Located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria (2017). https://ceur-ws.org/Vol-1963/paper555.pdf
10. Diomedi, D., Hogan, A.: Question answering over knowledge graphs with neural machine translation and entity linking. arXiv preprint abs/2107.02865 (2021). https://arxiv.org/abs/2107.02865
11. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: a large dataset for complex question answering over Wikidata and DBpedia. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 69–78. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_5
12. Faldu, K., Sheth, A.P., Kikani, P., Akabari, H.: KI-BERT: infusing knowledge context for better language and domain understanding. arXiv preprint abs/2104.08145 (2021). https://arxiv.org/abs/2104.08145

13. Galkin, M., Trivedi, P., Maheshwari, G., Usbeck, R., Lehmann, J.: Message passing for hyper-relational knowledge graphs. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 7346–7359. Association for Computational Linguistics (2020). https://aclanthology.org/2020.emnlp-main.596

14. He, B., et al.: BERT-MK: integrating graph contextualized knowledge into pre-trained language models. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 2281–2290. Association for Computational Linguistics (2020). https://aclanthology.org/2020.findings-emnlp.207

15. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys. **6**(1–4), 164–189 (1927). https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164

16. Huang, X., Zhang, J., Li, D., Li, P.: Knowledge graph embedding based question answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, pp. 105–113. ACM (2019). https://doi.org/10.1145/3289600.3290956

17. Huang, X., Zhang, J., Li, D., Li, P.: Knowledge graph embedding based question answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, pp. 105–113. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3289600.3290956

18. Jia, Z., Pramanik, S., Saha Roy, R., Weikum, G.: Complex temporal question answering on knowledge graphs. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, CIKM 2021, pp. 792–802. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3459637.3482416

19. Kacupaj, E., Plepi, J., Singh, K., Thakkar, H., Lehmann, J., Maleshkova, M.: Conversational question answering over knowledge graphs with transformer and graph attention networks. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 850–862. Association for Computational Linguistics (2021). https://www.aclweb.org/anthology/2021.eacl-main.72

20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings (2015). https://arxiv.org/abs/1412.6980

21. Lan, Y., He, G., Jiang, J., Jiang, J., Zhao, W.X., Wen, J.R.: A survey on complex knowledge base question answering: methods, challenges and solutions. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-2021, pp. 4483–4491. International Joint Conferences on Artificial Intelligence Organization (2021). https://doi.org/10.24963/ijcai.2021/611. Survey track

22. Lerer, A., et al.: PyTorch-BigGraph: a large scale graph embedding system. In: Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA. mlsys.org (2019). https://proceedings.mlsys.org/book/282.pdf

23. Liu, W., et al.: K-BERT: enabling language representation with knowledge graph. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 03, pp. 2901–2908 (2020)

24. Neelam, S., et al.: SYGMA: a system for generalizable and modular question answering over knowledge bases. In: Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, pp. 3866–3879. Association for Computational Linguistics, December 2022. https://aclanthology.org/2022.findings-emnlp.284

25. Oguz, B., et al.: UniK-QA: unified representations of structured and unstructured knowledge for open-domain question answering (2020). https://arxiv.org/abs/2012.14610

26. Plepi, J., Kacupaj, E., Singh, K., Thakkar, H., Lehmann, J.: Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 356–371. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_21

27. Pramanik, S., Alabi, J., Roy, R.S., Weikum, G.: UNIQORN: unified question answering over RDF knowledge graphs and natural language text. arXiv preprint abs/2108.08614 (2021). https://arxiv.org/abs/2108.08614

28. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1), 5485–5551 (2020)

29. Ravishankar, S., et al.: A two-stage approach towards generalization in knowledge base question answering. CoRR abs/2111.05825 (2021). https://arxiv.org/abs/2111.05825

30. Robertson, S.E., Walker, S.: Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In: Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1994)

31. Rony, M.R.A.H., Kumar, U., Teucher, R., Kovriguina, L., Lehmann, J.: SGPT: a generative approach for SPARQL query generation from natural language questions. IEEE Access **10**, 70712–70723 (2022). https://doi.org/10.1109/ACCESS.2022.3188714

32. Saha, A., Pahuja, V., Khapra, M.M., Sankaranarayanan, K., Chandar, S.: Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: AAAI 2018/IAAI 2018/EAAI 2018. AAAI Press (2018)

33. Saxena, A., Tripathi, A., Talukdar, P.: 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, pp. 4498–4507. Association for Computational Linguistics (2020). https://aclanthology.org/2020.acl-main.412

34. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38

35. Shen, T., et al.: Multi-task learning for conversational question answering over a large-scale knowledge base. 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), Hong Kong, China, pp. 2442–2451. Association for Computational Linguistics (2019). https://aclanthology.org/D19-1248

36. Sun, H., Bedrax-Weiss, T., Cohen, W.: PullNet: open domain question answering with iterative retrieval on knowledge bases and text. 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), Hong Kong, China, pp. 2380–2390. Association for Computational Linguistics (2019). https://aclanthology.org/D19-1242

37. Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., Cohen, W.: Open domain question answering using early fusion of knowledge bases and text. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language

Processing, Brussels, Belgium, pp. 4231–4242. Association for Computational Linguistics (2018). https://aclanthology.org/D18-1455

38. Sun, T., et al.: CoLAKE: contextualized language and knowledge embedding. In: Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, pp. 3660–3670. International Committee on Computational Linguistics (2020). https://aclanthology.org/2020.coling-main.327

39. Pellissier Tanon, T., de Assunção, M.D., Caron, E., Suchanek, F.M.: Demoing platypus – a multilingual question answering platform for Wikidata. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 111–116. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_21

40. Vakulenko, S., Garcia, J.D.F., Polleres, A., de Rijke, M., Cochez, M.: Message passing for complex question answering over knowledge graphs. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, pp. 1431–1440. ACM (2019). https://doi.org/10.1145/3357384.3358026

41. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, pp. 5998–6008 (2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

42. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lió, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations (2017)

43. Wang, X., et al.: KEPLER: a unified model for knowledge embedding and pretrained language representation. Trans. Assoc. Comput. Linguist. **9**, 176–194 (2021). https://aclanthology.org/2021.tacl-1.11

44. Yani, M., Krisnadhi, A.A.: Challenges, techniques, and trends of simple knowledge graph question answering: a survey. Information **12**(7) (2021). https://www.mdpi.com/2078-2489/12/7/271

45. Yu, D., Zhu, C., Yang, Y., Zeng, M.: JAKET: joint pre-training of knowledge graph and language understanding. In: AAAI 2022 (2022). https://www.microsoft.com/en-us/research/publication/jaket-joint-pre-training-of-knowledge-graph-and-language-understanding/