A Two-Stage Approach towards Generalization in Knowledge Base Question Answering

Anonymous ACL submission

Abstract

Most existing approaches for Knowledge Base Question Answering (KBQA) focus on a specific underlying knowledge base either because of inherent assumptions in the approach, or because evaluating it on a different knowledge base requires non-trivial changes. However, many popular knowledge bases share similarities in their underlying schemas that can be leveraged to facilitate generalization across knowledge bases. To achieve this generalization, we introduce a KBQA framework based on a 2-stage architecture that explicitly separates semantic parsing from the knowledge base interaction, facilitating transfer learning across datasets and knowledge graphs. We show that pretraining on datasets with a different underlying knowledge base can nevertheless provide significant performance gains and reduce sample complexity. Our approach achieves comparable or state-of-the-art performance for LC-QuAD (DBpedia), WebQSP (Freebase), SimpleQuestions (Wikidata) and MetaQA (Wikimovies-KG).

1 Introduction

Knowledge Base Question Answering (KBQA) has gained significant popularity in recent times due to its real-world applications, facilitating access to rich Knowledge Graphs (KGs) without the need for technical query-syntax. Given a natural language question, a KBQA system is required to find an answer based on the facts available in the KG. For example, given the question "Who is the director of the film Titanic", a KBQA system should retrieve the entity corresponding to "James Cameron". This would be `dbr:James_Cameron`[1] in DBpedia (Auer et al., 2007), `wd:Q42574`[2] in Wikidata (Vrandečić and Krötzsch, 2014), and `fb:m.03_gd`[3] in Freebase (Bollacker et al., 2008).

KBQA has been evaluated on multiple different KGs such as Freebase (Bollacker et al., 2008), Wikidata (Vrandečić and Krötzsch, 2014), DBpedia (Auer et al., 2007), and MetaQA (Zhang et al., 2018). Most existing heuristic-based KBQA approaches such as NSQA (Kapanipathi et al., 2020), gAnswer (Zou et al., 2014), and QAmp (Vakulenko et al., 2019) are typically tuned for a specific underlying knowledge base making it non-trivial to generalize and adapt it to other knowledge graphs. On the other hand, WDAqua (Diefenbach et al., 2017a), a system with a focus on being generalizable, ignores question syntax, thereby showing reduced performance on datasets with complex multi-hop questions.

Recently, there has been a surge in end-to-end learning approaches that are not tied to specific KGs or heuristics, and hence can generalize to multiple KGs. GrailQA (Gu et al., 2021) in particular categorized different forms of generalization, such as novel relation compositionality and zero-shot generalization. They also demonstrated transfer across QA datasets, but within the same KG. On the other hand, GraftNet (Sun et al., 2018) and EmbedKGQA (Saxena et al., 2020) demonstrated their ability to generalize over multiple KGs by demonstrating state-of-the-art performance on MetaQA (Wikimovies) as well as WebQSP (Freebase). The two techniques, however, are highly sensitive to the training data; failing to generalize in terms of relation compositionality within a KG. EmbedKGQA and GraftNet show significant drops (between 23-50%) in performance on relation compositions that are not seen during training. Furthermore, it is unclear how these systems transfer across KGs because of their tight-integration with KG-specific embeddings.

In this work, we present a novel generalizable KBQA approach STaG-QA (Semantic parsing for Transfer and Generalization) that works seamlessly with multiple KGs, and demonstrate transfer

[1] dbr: `http://dbpedia.org/resource/`
[2] wd: `http://www.wikidata.org/entity`
[3] fb: `http://rdf.freebase.com/ns/`

even across QA datasets with different underlying KGs. Our approach attempts to separate aspects of KBQA systems that are softly tied to the KG but generalizable, from the parts more strongly tied to a specific KG. Concretely, our approach has two stages: 1) The first stage is a generative model that predicts a query skeleton, which includes the query pattern, the different SPARQL operators in it, as well as partial relations based on label semantics that can be generic to most knowledge graphs. 2) The second stage converts the output of the first stage to a final query that includes entity and relations mapped to a specific KG to retrieve the final answer.

Our contributions are as follows:

- A simple SEQ2SEQ architecture for KBQA that separates aspects of the output that are generalizable across KGs, from those that are strongly tied to a specific KG.
- To the best of our knowledge, our approach is the first to evaluate on and achieve state-of-the-art or comparable performance on KBQA datasets corresponding to four different knowledge graphs, i.e, LC-QuAD (DBpedia), WebQSP (Freebase), SimpleQuestions (Wikidata) and MetaQA (Wikimovies).
- Our extensive experimental results shows that the proposed architecture: (a) facilitates transfer with significant performance gains in low-resource setting; (b) generalizes significantly better (23-50%) to unseen relation combinations in comparison to state-of-the-art approaches.

We make our code and pretrained models available[4]

2 Proposed Architecture

The KBQA task involves finding an answer for a natural language question from a given KG. Following the semantic parsing techniques for KBQA (Chen et al., 2021; Kapanipathi et al., 2020; Yih et al., 2015), we attempt to solve this task by predicting the correct structured SPARQL query that can retrieve the required answer(s) from the KG, i.e, by estimating a probability distribution over possible SPARQL queries given the natural language question.

In this work, we aim to design a model architecture that generalises across different KGs such as DBpedia, Wikidata, and Freebase. In order to achieve this goal, we have a 2-stage approach as shown in Figure 1, where we separate generic SPARQL query-sketch learning from KG-specific mapping of concepts. Specifically, the following 2-stages are:

KG	Query Graph Structure
DBpedia	`?var <director> <entity>` `?var <language> ?ans`
Wikimovies	`?var <directed by> <entity>` `?var <In language> ?ans`
Wikidata	`?var <director> <entity>` `?var <original language of film> ?ans`

Table 1: Query sketch for the question "The films directed by John Krasinski are in which language?"

***Softly-tied query sketch*:** This is the first stage of our approach where we learn aspects of the SPARQL query generation that are generic to any KG. Specifically, we observe the following: (i) multi-hop patterns are mostly generic to question answering over KGs. (ii) across many KGs, analogous relations have semantic or lexical overlap. Therefore, we focus on 2 sub-tasks in this stage, query skeleton generation and partial relation linking. We call the output of this stage a *softly-tied* semantic parse, because the exact output is partially dependent on the specific KG in use, but our choice of representations and architecture ensures that transfer across KGs is a natural consequence.

***KG alignment*:** This is the next step where we introduce all vocabulary specific to the KG in order to generate an executable SPARQL query. To do so, we bind the softy-tied semantic parse strongly to the KG to find the answer by (i) resolving the textual relations to KG relations, (ii) introducing KG specific entities into the SPARQL skeleton, and (iii) rank the obtained SPARQL queries based on its groundings in the KG.

2.1 Softly-tied Query Sketch

As mentioned above, the goal is to create a representation and architecture that can generalize easily not only across examples within a dataset, but also across KGs. To accomplish this, we define 2 subtasks: (a) Skeleton Generation, and (b) Partial relation linking.

Skeleton Generation: A SPARQL's skeleton captures the operators needed to answer the question; i.e. ASK, SELECT, COUNT or FILTER, as well as

[4] https://github.ibm.com/Srini/text2sparql

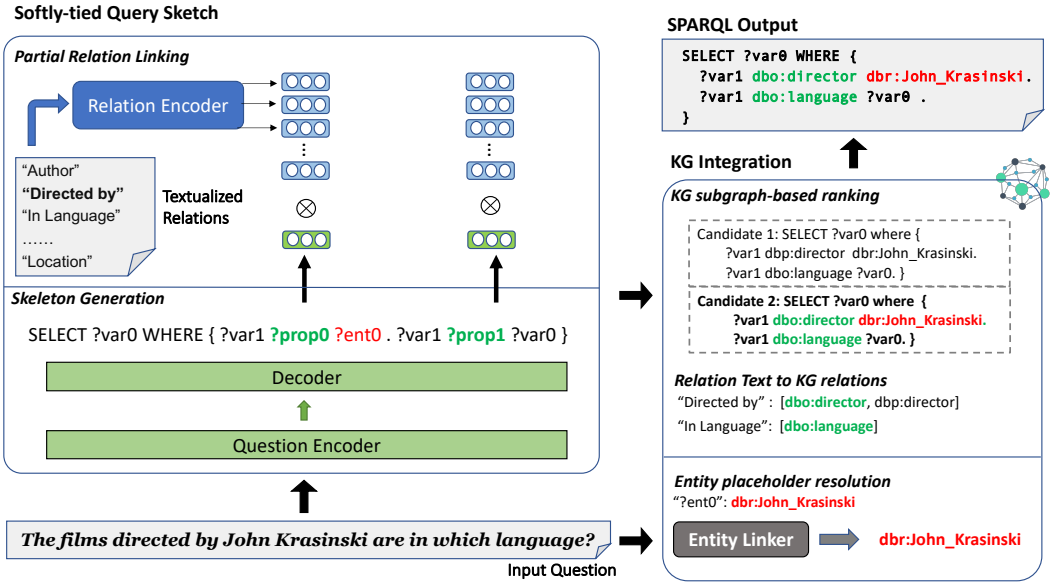

Figure 1: Two-stage system architecture that comprises of: (a) On the left: Softly-tied semantic parse generation that takes an input question return a KG-agnostic parse, and (b) On the right: Knowledge Graph Integration process to eventually return the SPARQL query.

the query graph structure, with placeholder nodes for entities (e.g. `:ent0`), relations (e.g. `:prop0`) and variables (e.g. `?var0`). For many questions, the generated SPARQL skeletons across different KGs are similar, if not identical. The skeleton structures unique to a KG, e.g. reification (present in Wikidata but not DBpedia), can be learnt when fine-tuning on a dataset with that underlying KG. An example of a SPARQL skeleton for our running example in Figure 1 "The films directed by John Krasinski are in which language?" is:

```
SELECT ?var0 WHERE
{ ?var1 :prop0 :ent0 .
  ?var1 :prop1 ?var0 . }
```

As shown in Figure 1, the question is passed through a transformer-based SEQ2SEQ model which is trained to produce the SPARQL skeleton corresponding to the question text. We use a BERT-base (Devlin et al., 2018) encoder, while the decoder has a similar architecture to BERT-base but with added cross-attention layers.

Given a question text, we tokenize it using BERT tokenizer and add special `[CLS]` and `[SEP]` symbols in the beginning and the end of the question, respectively. This tokenized input is passed through a transformer encoder, producing encoder hidden states for each token at each layer. The encoder is initialized with pretrained BERT model (Devlin et al., 2018), which helps generalization with respect to different question syntax. We then use a transformer decoder with cross attention mechanism. At each time step i, the decoder considers the encoder states via cross-attention and previous decoder states via self attention. It produces a distribution over possible skeleton output tokens. The decoder output vocabulary $\mathcal{V}$ comprises of entity place holder tokens $\mathcal{V}_e$, relation place holder tokens $\mathcal{V}_r$ and SPARQL operators $\mathcal{V}_o$; each of these is a small closed set of tokens. The output of each decoding step is a softmax over possible tokens $s_i \in \mathcal{V}$. Unlike the encoder, no pre-trained model is used for the decoder, and parameters are initialized randomly.

Consider a question answering dataset $\mathcal{Q}$ with question-SPARQL pairs (q, s); let the tokenized form be $(s_1, s_2 ... s_t)$. Then the skeleton generation loss is given by:

$$\mathcal{L}_1(\mathcal{Q}) = -\sum_{(q,s)\in\mathcal{Q}} \frac{1}{t}\sum_{i=1}^{t} \log P(s_i|q, s_1..s_{(i-1)}; \Theta_1) \quad (1)$$

Partial Relation Linking: For each relation placeholder in the SPARQL skeleton (`:prop0`, `:prop1`, etc), we need to identify the appropriate relation that can replace the placeholder to produce the correct semantic representation of the query. We have noted previously that relations across KGs share

lexical and semantic similarities. For example, in Table 1 the three KGs (DBpedia, Wikimovies, and Wikidata) represent the relationship “Directed by" with very similar lexical terms “Director" and “Directed by". We can thus leverage large pre-trained language models to allow generalization and transfer of such relations across KGs. In each KG, we first map the relations to their respective surface forms, using either `label` relations from the KG, or by extracting some semantically meaningful surface form from the relation URI. These are the “textualized relations" shown in Figure 1. Table 2 shows some more examples of relation labels for 3 KGs. Note that this mapping can be many-to-one. For example, both `dbo:language` and `dbp:language` map to the same relation label “language".

Our goal is to identify which relation surface form best matches each relation placeholder in the skeleton. We thus train the SEQ2SEQ decoder and relation encoder to project into the same space. Concretely, the decoder hidden state corresponding to each relation placeholder is optimised to be closest to the encoded representation of the correct relation, using a cross-entropy loss.

Let the output layer embeddings corresponding to the decoder tokens $(s_1, s_2..s_t)$ be $(h_1, h_2..h_t)$. Let $\mathcal{Z}$ be the indices corresponding to the placeholder tokens in s, and Y_i be the correct relation corresponding to a placeholder token s_i. Then the partial relation linking loss is defined as:

$$\mathcal{L}_2(\mathcal{Q}) = -\sum_{(q,h)\in\mathcal{Q}}\sum_{i\in\mathcal{Z}} \log P(Y_i|h_i;\Theta_2) \quad (2)$$

$$P(Y_i|h_i;\Theta_2) = \frac{e^{r_{Y_i}.h_i}}{\sum_{k\in\mathcal{R}} e^{r_k.h_i}} \quad (3)$$

where r_{Y_i} denotes the relation embedding obtained from the relation encoder, corresponding to Y_i in the relation dictionary $\mathcal{R}$. For example, in Figure 1, the decoder state for `:prop0` should have maximum inner product with the encoded representation for the relation surface form “Directed by", compared to the encoded representations of all other relations. Our relation encoder is a transformer model whose parameters are initialized with pretrained BERT model. Given that BERT-based representations of lexically or semantically similar relations across KGs will be close, it is easy to see why transfer across KG is possible. The final outcome of partial relation linking is a ranked list of relation surface forms for each placeholder in the skeleton.

A linear combination of the skeleton generation loss and partial relation linking loss is optimized.

$$\mathcal{L}(\mathcal{Q}) = \lambda\mathcal{L}_1(\mathcal{Q}) + (1-\lambda)\mathcal{L}_2(\mathcal{Q}) \quad (4)$$

The SPARQL skeleton together with the partial relation linking produces a ranked list of softly-tied query sketches. In the case of multiple placeholders, the score of each pair of relation surface forms is the product of their individual scores. Sometimes this phase produces multiple semantic interpretations, either due to noisy surface forms (for instance, DBpedia KG includes Wikipedia infobox keys “as is” when they can not be mapped to the ontology relations) or due to the presence of semantically identical or similar relations with distinct identifiers (eg. `dbo:language` and `dbp:language`). For the example, “The films directed by John Krasinski are in which language?", this stage will produce the following sketches:

```
P=0.87    SELECT ?var0 where {
            ?var1 director :ent0.
            ?var1 language ?var0.}
P=0.76    SELECT ?var0 where {
            ?var1 director :ent0.
            ?var1 languages ?var0.}
...
```

2.2 KG Interaction

In order to generate an executable SPARQL query, we need to introduce vocabulary specific to the KG. The KG interaction stage performs this task. Concretely, given a list of candidate query sketches, this stage performs the following steps to produce the final question answer: 1) link the different entities to their corresponding placeholders in the skeleton, 2) disambiguate relations' textual form and link it to the specific KG relations, and 3) select the correct SPARQL based on the actual facts in the KG.

In our approach, we leverage a pre-trained off-the-shelf entity linker, BLINK (Wu et al., 2020). BLINK provides tuples of (surface form, linked entity) pairs. The entity placeholder resolution step aligns the entities with the entity place holders in the query sketch. In the example above, `:ent0` will be linked to `dbr:John_Krasinski` in DBpedia, or `wd:Q313039` in Wikidata. When multiple entities are present in the question, the position of the corresponding textual span defines the alignment to the entity placeholder variable. During training, the first entity in the question corresponds to `:ent0`, the second entity by `:ent1`, etc. This pattern is repeated by the system when decoding during inference,

KG	KG Relation	Derived Surface Form
DBpedia	dbo:language	language
	dbp:language	language
	dbp:languages	languages
Wikidata	P397	official language
	P364	original language of film or TV show
FreeBase	people.ethnicity.languages_spoken	languages spoken
	location.country.languages_spoken	languages spoken

Table 2: Examples of textualized relations for different KGs, obtained either using the relation label from the KG (DBpedia, Wikidata) or by extracting a part of the relation URI (Freebase)

making entity placeholder resolution trivial.

The next step is to disambiguate relations' textual form and link them to the specific KG relations. Recall from Table 2 that each surface form in a query sketch can map to one or more KG relations. In our example using DBpedia as a KG, the surface form "director" could map to both [`dbo:director`, `dbp:director`] whereas "language" could map to both [`dbo:language`, `dbp:language`]. The semantic parsing stage cannot hope to distinguish between these, and thus we rely on the KG to determine the specific relation that should be chosen. Concretely, we replace every relation surface form with each of the possible KG relations it could map to. Thus, each softly-tied query sketch produces one or more fully executable SPARQLs. For example, the 2 softly-tied sketches from the previous stage in our example produce 4 possible SPARQLs, see Table 3. As the final step, we execute the candidate SPARQL queries against the KB and choose the highest-ranked SPARQL that produces an answer for SELECT queries. Since ASK queries do not necessarily have to be valid in the KG, we only consider the model score in such cases.

Ranked SPARQL query predictions
`1. SELECT ?var0 where { ?var1 dbo:director dbr:John_Krasinski. ?var1 dbo:language ?var0. }`
`1. SELECT ?var0 where { ?var1 dbp:director dbr:John_Krasinski. ?var1 dbo:language ?var0. }`
`2. SELECT ?var0 where { ?var1 dbo:director dbr:John_Krasinski. ?var1 dbp:languages ?var0. }`
`2. SELECT ?var0 where { ?var1 dbp:director dbr:John_Krasinski. ?var1 dbp:languages ?var0. }`

Table 3: Top predicted SPARQL queries for the question "The films directed by John Krasinski are in which language?"

Dataset	Train	Valid	Test
LC-QuAD 1.0	3,650	200	1,000
SimpleQuestion	15,000	2,000	2,280
WQSP-FB	2898	200	1,596
MetaQA 1-hop	86,470	9,992	9,947
MetaQA 2-hop	118,980	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274

Table 4: Dataset Statistics

3 Experiments

In this section, we validate 2 claims: (1) STaG-QA achieves state-of-the-art or comparable performance on a variety of datasets and KGs. (2) STaG-QA generalizes across KBs and hence facilitating transfer. All experiments were run on a machine with an Intel Xeon E5-2690 CPU and 1 x P100 GPU. SPARQL Virtuoso endpoints were setup on the local network for all KBs. Runtime was approximately 2.28 seconds per question. The results show that pre-training our system even on a different KG achieves improvement in performance with better gains in low-resource and unseen relation combination settings.

3.1 Datasets

To evaluate the generality of our approach, we used datasets across a wide variety of KGs including Wikimovies-KG, Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), and Wikidata (Vrandečić and Krötzsch, 2014). In particular, we used the following datasets (Table 4 shows detailed statistics for each dataset): (a) **MetaQA**

(Wikimovies-KG) (Zhang et al., 2018) is a large-scale complex-query answering dataset on a KG with 135k triples, 43k entities, and nine relations. It contains more than 400K questions for both single and multi-hop reasoning. (b) **WQSP-FB** (Freebase) (Yih et al., 2016) provides a subset of WebQuestions with semantic parses, with 4737 questions in total. (c) **LC-QuAD 1.0** (DBpedia) (Trivedi et al., 2017): A dataset with 5,000 questions (4,000 train and 1,000 test) based on templates. It includes simple, multi-hop, as well as aggregation-type questions. LC-QuAD 2.0 is another version of LC-QuAD based on Wikidata. It has 30K question in total and also template-based. Due to the larger underlying KB and the extensive pattern covered, we used LC-QuAD 2.0 dataset for pretraining and showing our transfer results. (d) **SimpleQuestions-Wiki** (Wikidata) (Diefenbach et al., 2017b): a mapping of the popular Freebase's SimpleQuestions dataset to Wikidata KB with 21K answerable questions.

3.2 Baselines

In this work, we evaluate against 10 different KBQA systems. (1) NSQA (Kapanipathi et al., 2020; Abdelaziz et al., 2021) is state-of-the-art system for KBQA on DBpedia datasets. (2) QAMP (Vakulenko et al., 2019) is an unsupervised message passing approach that provides competitive performance on LC-QuAD 1.0 dataset. (3) WDAqua (Diefenbach et al., 2017a) is another system that generalises well across a variety of knowledge graphs. (4) Falcon 2.0 (Sakor et al., 2020) is a heuristics-based approach for joint detection of entities and relations in Wikidata. Since this approach does not predict the query structure, we tested it on SimpleQuestions dataset only. (5) SYGMA (Neelam et al., 2021) is a modular approach facilitating multiple reasoning types, (6) EDGQA (Hu et al., 2021) which leverages a novel graph structure called Entity Description Graph (EDG) to represent the structure of complex questions, (7) EmbedKGQA (Saxena et al., 2020) is the state-of-the-art KBQA system on MetaQA and WebQSP datasets, (8) PullNet (Sun et al., 2019) is recent approach evaluated on MetaQA and WebQSP datasets, (9) GraftNet (Sun et al., 2018) infuses both text and KG into a heterogeneous graph and uses GCN for question answering, and (10) EmQL (Sun et al., 2020) is a query embedding approach that was successfully integrated into a KBQA system and evaluated on WebQSP and MetaQA datasets.

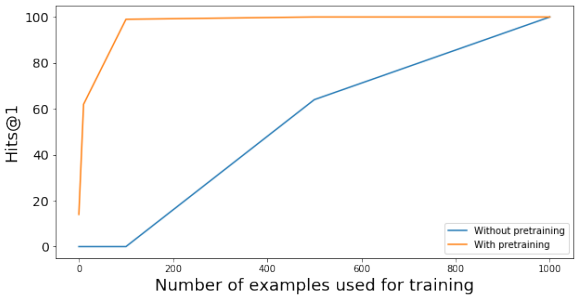

Figure 2: System performance on MetaQA 2-hop questions using different number of training examples, with and without pretraining on LC-QuAD 2.0.

3.3 Results

Table 5 shows our system results on all four datasets in comparison to existing approaches. We show two versions of our system, one pre-trained with LC-QuAD 2.0 datatset (Dubey et al., 2019) (STaG-QA$_{pre}$) and another trained from scratch on the target dataset only (STaG-QA). As noted earlier, to the best of our knowledge, we are the first to show generality across knowledge graphs by evaluating on datasets from DBpedia, Wikidata, Freebase, and Wikimovies-KG.

On SimpleQuestions-Wiki, our approach achieves significantly better performance compared to Falcon 2.0 and SYGMA with 24% and 16% better F1 score respectively. On MetaQA dataset, our system achieves near perfect scores on all 3 subsets. On LC-QuAD 1.0, our approach is on par with EDGQA, the state-of-the-art system for DBpedia. As for WebQSP, both versions of our approach are inferior compared to EmQL, which can leverage KBC embeddings unlike STaG-QA. Overall, the results show that STaG-QA shows better or competitive performance on 3 out of four datasets and when pretrained on another dataset, the performance improves across all datasets.

3.4 Effect of Pretraining

Our architecture is designed to allow transfer learning between entirely different QA dataset/KG pairs. We consider low-resource settings to highlight the benefit of transfer, even across KGs. This is particularly applicable when there is scarcity of training data for a new target KG. We investigate the benefit of pretraining the semantic parsing stage using LC-QuaD 2.0 (Wikidata), before training on the 2-hop dataset in MetaQA (Wikimovies-KG) and

System	SimpleQuestions-Wiki			LC-QuAD 1.0			WebQSP	MetaQA		
	P	R	F1	P	R	F1	Hits@1	Hits@1 1-Hop	Hits@1 2-Hop	Hits@1 3-Hop
WDAqua	-	-	-	22.0	38.0	28.0	-	-	-	-
QAMP	-	-	-	25.0	50.0	33.0	-	-	-	-
NSQA	-	-	-	44.8	45.8	44.4	-	-	-	-
EDGQA	-	-	-	50.5	**56.0**	53.1	-	-	-	-
Falcon 2.0	34	41.1	36.3	-	-	-	-	-	-	-
SYGMA	42.0	55.0	44 .0	47.0	48.0	47.0	-	-	-	-
GraftNet	-	-	-	-	-	-	70.3	97.0	99.9	91.4
PullNet	-	-	-	-	-	-	69.7	97.0	99.9	91.4
EmbedKGQA	-	-	-	-	-	-	66.6	97.5	98.8	94.8
EMQL	-	-	-	-	-	-	**75.5**	-	98.6	99.1
STaG-QA	59.4	62.7	61.0	**76.5**	52.8	51.4	65.9	100.0	100.0	99.9
STaG-QA$_{pre}$	**60.2**	**63.2**	**61.7**	74.5	54.8	**53.6**	68.5	**100.0**	**100.0**	**100.0**

Table 5: Performance against previous state-of-the-art approaches. Following these techniques, we report precision, recall and F1 scores on SimpleQuestions and LC-QuAD 1.0, and Hits@1 performance on WebQSP and MetaQA datasets. The subscript pre indicates the "pre-trained" version of our system using LC-QuAD 2.0 dataset.

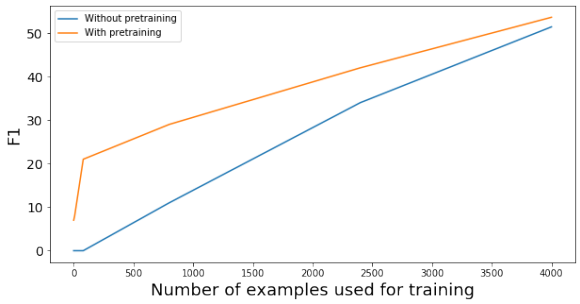

Figure 3: System performance on LC-QuAD 1.0 using different number of training examples, with and without pretraining on LC-QuAD 2.0.

the LC-QuAD 1.0 dataset (DBpedia). Figures 2 and 3 show the performance of STaG-QA on each dataset with and without pre-training. We make note of the following observations.

First, without any fine-tuning on either datasets, the pre-trained version STaG-QA$_{pre}$ is able to achieve 18% Hits@1 on MetaQA and 8% F1 on LC-QuAD 1.0, demonstrating zero-shot transfer across KGs. Second, the pre-trained version shows significantly low sample complexity. For example, in MetaQA (Figure 2), STaG-QA$_{pre}$ was able to reach almost 100% Hits@1 with 100 training examples only. To reach the comparable performance, STaG-QA without pretraining required $\sim 1,000$ examples, an order of magnitude more training data. The same behaviour can be observed on LC-QuAD 1.0.

3.5 Generalization to novel relation composition

Common KBs have a large number of relations. For example, DBpedia (v2016-10) has around ∼60K relations, Wikidata (as of March 2020) has ∼8K relations, whereas Freebase contains ∼25K relations. In multi-hop queries, these relations can be arranged as paths (e.g., *director* → *language*) where possible path combinations grow combinatorially. Seeing all possible relation combinations during training is impractical with most KBs as it would require significantly large training data to cover all combinations. Instead, an effective KBQA system should be able to generalise to unseen relation paths. However, we find that some prominent KBQA datasets do not effectively evaluate this property of a QA system.

We show in Table 6 the number of test questions in LC-QuAD 1.0, MetaQA and WebQSP datasets that contain relation combinations never seen at training. MetaQA does not have any unseen relation paths, and WebQSP contains only 2.06% of such questions. In contrast, in LC-QuAD 1.0 roughly half of the test questions contain novel relation compositions.

MetaQA Unseen Challenge Set: In order to investigate how this issue affects evaluation of KBQA systems, we modified the train and dev sets of MetaQA as follows: From the 2-hop training set, we removed training examples containing two randomly chosen relation paths (`actor_to_movie_to_director` and

Dataset	# unseen path questions
LC-QuAD 1.0	490/1,000 (49 %)
WebQSP	45/1,638 (2 %)
MetaQA 2-hop	0/14,872 (0 %)
MetaQA 3-hop	0/14,274 (0 %)

Table 6: Novel combinations in Test of seen relations

System	2-Hop Seen	2-hop Unseen
EmbedKGQA	99.00	50.00
GraftNet-KB	97.90	75.2
GraftNet-Text	51.2	43.3
GraftNet-Both	99.13	95.41
STaG-QA	**99.9**	**99.7**

Table 7: MetaQA Unseen Challenge Set Setting

`director_to_movie_to_actor`) and split the dev set into two, one containing 13,510 questions with all seen relations combinations in training and another containing 1,361 questions with all unseen relation paths.

We then trained STaG-QA, EmbedKGQA and GraftNet on the new reduced training set and tested the performance on our new development sets (seen and unseen). Shown in Table 7, the results clearly demonstrate that there is a significant drop in performance in methods that rank directly across entities in the KG to predict answers. This is most clearly observed in EmbedKGQA, as well as GraftNet-KB, though the use of text alleviates this issue. In contrast, our approach is able to maintain exactly the same level of performance for novel relation compositions using KB information alone.

4 Related Work

There have been a wide variety of Knowledge Base Question Answering (KBQA) systems trained on datasets that are either question-SPARQL pairs (strong supervision) or question-answer pairs (weak supervision). Some of the approaches are rule based, and depend on generic language based syntactic (Zou et al., 2014) or semantic parses (Abdelaziz et al., 2021; Kapanipathi et al., 2020) of the question and build rules on it to obtain a query graph that represents the SPARQL query. NSQA, the state of the art approach for DBpedia based datasets such as LC-QuAD-1.0 (Trivedi et al., 2017) and QALD-9 (Usbeck et al., 2017), falls in this category. Many of these systems have components or aspects that are specific to the KG they evaluate on, and do not trivially generalize to other KGs. In particular GAnswer, NSQA, and QAmp are specific to DBpedia and do not evaluate their approaches on any other KGs.

Closest to our architecture is MaSP, a multi-task end-to-end learning approach that focuses of dialog-based KGQA setup. MaSP uses a predicate classifier which makes transfer across KGs non-trivial. A prominent work (Maheshwari et al., 2019) ranks all candidate graph patterns retrieved from the knowledge graph based on the grounded entity. In multi-hop settings, as in MetaQA with 3-hop questions, retrieving all possible candidates upto n-hops (for an arbitrary choice of n) and then ranking across all of them is expensive. In contrast, our work focuses on a generative approach to model query graph patterns. EmbedKGQA (Saxena et al., 2020) and GraftNet are two such approaches that directly ranks across entities in the knowledge base to predict an answer, by leveraging either KG embeddings from Knowledge Base Completion (KBC); or creating a unified graph from KB and text. However, these approaches do not generalize well to novel relation compositions not seen during training. Finally, it is unclear how to transfer KBC embedding-based approaches such as EmbedKGQA across KGs since the learnt KG embeddings are tightly coupled with the specific KG in question.

5 Conclusion

In this work, we show that a simple 2-stage architecture which explicitly separates the KG-agnostic semantic parsing stage from the KG-specific interaction can generalize across a range of datasets and KGs. We evaluated our approach on four KG/QA pairs, obtaining state-of-the-art performance on MetaQA, LC-QuAD 1.0, and SimpleQuestions-Wiki; as well as competitive performance on WebQSP. We also successfully demonstrate transfer learning across KGs by showing that pre-training the semantic parsing stage on an existing KG/QA-dataset pair can help improve performance in low-resource settings for a new target KG; as well as greatly reduce the number of examples required to achieve state-of-the-art performance. Finally, we show that some popular benchmark datasets do not evaluate generalization to unseen combinations of seen relations (compositionality), an important requirement for a question answering system.

References

Ibrahim Abdelaziz, Srinivas Ravishankar, Pavan Kapanipathi, Salim Roukos, and Alexander Gray. 2021. A semantic parsing and reasoning-based approach to knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15985–15987.

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 325–336, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2017a. Wdaqua-core0: A question answering component for the research community. In *Semantic Web Evaluation Challenge*, pages 84–89. Springer.

Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. 2017b. 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, October 23rd - to - 25th, 2017*.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International semantic web conference*, pages 69–78. Springer.

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

Xixin Hu, Yiheng Shu, Xiang Huang, and Yuzhong Qu. 2021. Edg-based question decomposition for complex question answering over knowledge bases. In *International Semantic Web Conference*, pages 128–145. Springer.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, et al. 2020. Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. *arXiv preprint arXiv:2012.01707*.

Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. 2019. Learning to rank query graphs for complex question answering over knowledge graphs. In *International semantic web conference*, pages 487–504. Springer.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, et al. 2021. Sygma: System for generalizable modular question answering over-knowledge bases. *arXiv preprint arXiv:2109.13430*.

Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. 2020. Falcon 2.0: An entity and relation linking tool over wikidata. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3141–3148.

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.

Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. 2020. Faithful embeddings for knowledge base queries.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *EMNLP*.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC 2017*, pages 210–218.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (qald-7). In *Semantic Web Evaluation Challenge*, pages 59–69. Springer.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1431–1440.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*, pages 201–206.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324.