A Zero-Shot Neuro-Symbolic Approach for Complex Knowledge Graph Question Answering

Prerna Agarwal[*1,2], **Srikanta Bedathur**[1]
[1]Yardi School of AI, IIT Delhi, India
[2]IBM Research India
preragar@in.ibm.com, srikanta@cse.iitd.ac.in

Abstract

Existing low-resource Knowledge Graph Question Answering (KGQA) methods rely heavily on Large Language Models (LLMs) for semantic parsing of natural language question to its corresponding logical form (LF) such as SPARQL, S-Expression, etc. However, LLMs becomes bottleneck for practical applications due to: (1) its high computational resource requirements; (2) limited knowledge of LLM about different LFs; (3) unavailability of low-resource annotated data for new KGs and settings. This motivates us to design a KGQA framework that can operate in a **zero-shot setting** without the need for additional resources. In this paper, we propose (NS-KGQA)[1]: a zero-shot neuro-symbolic approach based on neural KG embeddings that have demonstrated their ability to effectively model the structure of the KG without the need for additional data. We extract a link-prediction based symbolic question subgraph. We then propose a *Symbolic Resolver* that uses *Dual KG Embeddings* combined with a symbolic approach to resolve the symbolic question subgraph. Our extensive experiments on Complex KGQA benchmarks such as KQA Pro demonstrate the effectiveness of our approach. NS-KGQA outperforms all other LLM-based zero-shot baselines by 26% (avg).

1 Introduction

Research has moved towards low-resource Knowledge Graph Question Answering (KGQA) (Gu et al., 2023; Li et al., 2023b) due to the scarcity of annotated data. KGQA aims to answer a natural language question using the facts present in the Knowledge Graph (KG) in the form of triples (h, r, t) by producing a logical form (LF) such as SPARQL, S-Expression, etc. that is executed on these KG facts to retrieve the answer(s) (Shu et al., 2022; Nie et al., 2022; Neelam et al., 2022). Due to the recent surge in the complexity of the questions users pose to the systems, complex KGQA has particularly gained attention (Gu et al., 2021a; Cao et al., 2022a; Huang et al., 2023a).

LLM-based low-resource methods have shown promising results with various state-of-the-art LLMs such as GPT-3 (Brown et al., 2020), GPT-4 (OpenAI et al., 2023) variants to generate the LF. However, for practical applications, these LLMs become bottleneck due to: (1) high costs and computational resource requirements associated to it; (2) limited pre-trained knowledge about the grammar and semantics of the LF (SPARQL, S-expression, etc.) and hence limited generative capability; (3) unavailability of low-resource annotated data for new KGs and settings at cold-start. Moreover, these methods perform inferior for more complex KGQA benchmarks such as KQA Pro (Cao et al., 2022a; Huang et al., 2023a) as these benchmarks require explicit modeling of KG complexities such as concepts, attributes, qualifiers, etc. to perform joint compositional and numerical reasoning that may not be seen by the LLM.

This motivates us to design a KGQA framework that can operate in a zero-shot setting without additional data. As complex questions require joint compositional and numerical reasoning, this requires the decomposition of the question and the solution of them, providing transparency (Wei et al., 2023; Liu et al., 2022). Therefore, in this paper, we propose NS-KGQA: a zero-shot neuro-symbolic approach based on neural KG embeddings that have demonstrated their ability to effectively model KG structure without the need of additional data. We decompose the natural language question into a link-prediction based symbolic question subgraph which is then resolved by our proposed Symbolic Resolver powered by Dual KG Embeddings.

We illustrate the proposed NS-KGQA reasoning

[*]P. Agarwal is an employee at IBM Research. This work was carried out as part of PhD research at IIT Delhi

[1]Code and data is available at: https://github.com/data-iitd/NS-KGQA

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 11514–11527
November 4-9, 2025

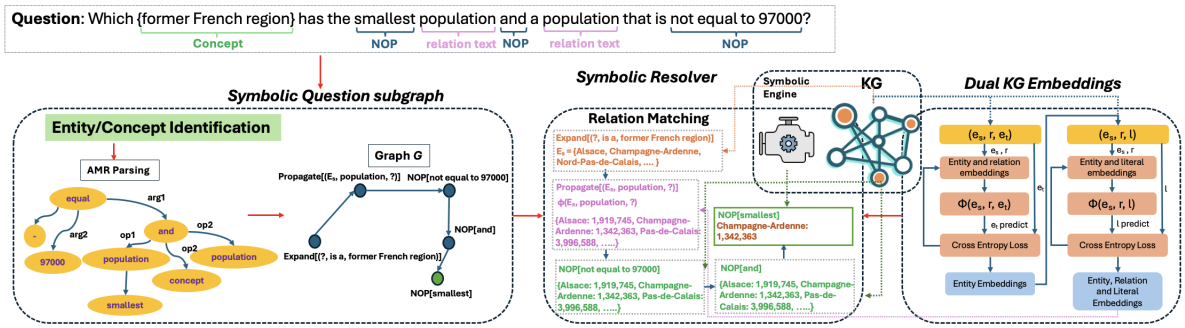

Figure 1: The overview of the proposed NS-KGQA: a zero-shot neuro-symbolic approach for complex KGQA. For a given question, it first creates a symbolic question subgraph (with the help of AMR parser) which is resolved by the Symbolic Resolver using Dual KG Embeddings.

process in Figure 1 with a complex real-world question where, *former French region* is a KG concept. The phrase *not equal to 97000* and *and* are **numerical operation phrase**s (NOP) that will be identified and satisfied during answer retrieval. Section 3.4 details the Symbolic Question Subgraph generation process. Subsequently, reasoning is conducted with a Symbolic Resolver (explained in Section 3.6) in the following steps:

1. **Expand** the KG concept *former French region* to extract entities such as `Alsace`, `Champagne-Ardenne` and `Nord-Pas-de-Calais`.
2. Retrieve and **propagate** the *population* corresponding to each entity (Section 3.5 explains the relation matching process).
3. Compare the *population* of each entity with constraint (not equal to 97000) in the **NOP**.
4. Perform *and* operation in the **NOP**. Since, there is only one input to the *and* operator, the whole input entity set with their population values becomes the output.
5. Compute the *smallest* operation in the **NOP** to obtain the entity whose population is the smallest. Here, the entity `Champagne-Ardenne` has the smallest population and is returned as the answer.

To summarize, in this paper, we make the following key contributions:

1. We propose NS-KGQA: a neuro-symbolic approach that can operate with any KG in a zero-shot (cold-start) setting.
2. A novel non-parametric link-prediction based *Symbolic Question Subgraph* generation powered by AMR (Abstract Meaning Representation) to enable joint compositional and numerical reasoning.
3. A novel *Symbolic Resolver* that uses a symbolic approach to resolve the Symbolic Question Subgraph to obtain the answer(s).
4. A *Dual KG embedding* neural module that learns embeddings of KG attributes separately from entities and then fused with the Symbolic Resolver.

To the best of our knowledge, NS-KGQA is the first one to propose a neuro-symbolic approach for complex KGQA in a zero-shot setting. Experiments demonstrates that NS-KGQA outperforms all other LLM-based zero-shot and even most of the state-of-the-art (SOTA) fully-supervised approaches (for KQA Pro dataset) by a significant margin.

2 Related Work

2.1 KG Embedding Methods

The KG embeddings (KGE) are designed for Link Prediction in KGs (Nickel et al., 2011; Yang et al., 2014; Qiu et al., 2020; Bordes et al., 2013; Sun et al., 2019b) using only KG triples. The idea is to predict the links in the KG by learning a scoring function ϕ s.t. the score for all correct triples is positive, and negative for rest of the triples.

2.2 LLM-based KGQA methods

Recent LLM-based KGQA techniques, such as RnG-KBQA (Ye et al., 2022), TIARA (Shu et al., 2022), KB_BINDER (Li et al., 2023a) convert natural language questions to S-Expression with constrained decoding using LLMs. Pangu (Gu et al., 2023), BYOKG (Agarwal et al., 2024a) leverage an LLM-based agent to understand KG information through exploration. FlexKBQA (Li et al., 2023b) introduces an execution-guided self-training method and surpasses all other few-shot methods. DecAF (Yu et al., 2023) generates LF and the final answer using LLMs. ChatKBQA (Luo et al., 2024), (Niu et al., 2023; Fang et al., 2024) fine-tunes LLMs for LF generation, but still lags behind in performance. KAPING (Baek et al., 2023) injects knowledge into LLM prompt

for QA. Some of the recent works (Wang et al., 2023b,a; Liang et al., 2023; Xiong et al., 2024) have also explored chain-of-thought (CoT) reasoning for KGQA which is a separate line of research and not the focus of our work. All these approaches are heavily based on LLMs and suffer from various limitations (as discussed in Section 1).

2.3 Fully-Supervised KGQA methods

They are broadly classified into the following:

1. **Semantic Parsing (SP) methods:** These methods (Shu et al., 2022; Nie et al., 2022; Neelam et al., 2022) transform the natural language question into LFs that are directly executable on KG to retrieve the answer. KQA Pro (Cao et al., 2022a) and GraphQIR (Nie et al., 2022) work also provides a technique for learning KoPL / SPARQL mapping questions using scratch training of sequence-to-sequence models such as BART. These works have shown SOTA on the KQA Pro dataset. Transfer learning approaches (Cao et al., 2022b) requires a large volumes of logical program annotations in the source domain (KQA Pro), and additionally a large volume of unlabeled data in the target domain (WebQSP) for Program Transfer. These methods are highly sensitive to a large amount of manual annotated Q-LF pairs. Moreover, these techniques fail if a relevant fact is missing from KG.

2. **Information Retrieval (IR) methods:** These techniques include end-to-end learning approaches that are not tied to any specific KG (Miller et al., 2016; Saxena et al., 2020; Shi et al., 2021; Zhang et al., 2022; Saxena et al., 2022). These techniques use QA pairs with KG triples to rank candidate entities given a question. Hence, these techniques can be generalized to multiple KGs.

Both categories of techniques have shown SOTA performance in specific datasets, which are highly sensitive to a large amount of manually annotated data for learning. Hence, none of these techniques can be adopted for practical applications where KG specific training data is not available, which is generally the case for cold-start problems.

2.4 Comparison with Prior AMR-Based Approaches

The following key differences distinguish our approach from others, such as (Kapanipathi et al., 2021), despite both utilizing AMR:

(a) Unlike our method, (Kapanipathi et al., 2021) does not operate in a zero-shot setting. Their approach involves manually generating AMRs for the train and dev sets of QALD and LC-QuAD 1.0 to fine-tune the AMR parser. In contrast, our method uses AMR in a lightweight fashion and does not require fine-tuning, enabling it to function entirely in a zero-shot setting.

(b) The core contribution of (Kapanipathi et al., 2021) lies in the transformation of AMR graph directly to SPARQL whereas the core contribution of our work lies in the use of AMR graph to obtain link-prediction based symbolic subgraph.

(c) (Kapanipathi et al., 2021) completely rely on the 'unknowns' identified by the AMR for query transformation, while our work relies on the AMR to identify only the source and target of the text of the relation.

In general, our work illustrates how AMR can be employed in a lightweight and efficient manner, unlike approaches such as (Kapanipathi et al., 2021) that are heavily AMR dependent.

3 NS-KGQA Framework

We introduce the details of NS-KGQA framework shown in Figure 1, that consists of three modules: (1) Learn Dual KG Embeddings; (2) Link-Prediction based Symbolic Question subgraph Generation; and (3) Symbolic Resolver.

3.1 Problem Statement

The goal of the NS-KGQA framework is to: (1) Learn neural Dual KG Embeddings using KG Triples T; (2) Generate the link-prediction based Symbolic Subgraph G; (3) Resolve G using a symbolic approach fused with Dual KGE to answer a given natural language question (Q). Here, $T \subset E \times R \times (E \cup L \cup C)$, where C, E, L, R are the set of concepts[2], entities, attributes (literals) and binary relations, respectively. The triples T are of the form (head, relation, tail).

3.2 ComplEx Embeddings

ComplEx (Trouillon et al., 2016) method is based on tensor factorization that embeds each relation $r \in R$ and each entity $e \in E$ in complex space. For a triple (h, r, t) where $h, r \in E$ and $r \in R$, ComplEx generates $\mathbb{h}$, $\mathbb{r}$ and $\mathbb{t} \in \mathbb{C}^d$ using the scoring function ϕ:

$$\phi(h, r, t) = \mathsf{Re}(\langle \mathbb{h}, \mathbb{r}, \overline{\mathbb{t}} \rangle)$$

[2]A concept is an abstraction of a set of entities

$$= \mathsf{Re}(\sum_{k=1}^{d} \mathbb{h}^{(k)}, \mathbb{r}^{(k)}, \bar{\mathbb{t}}^{(k)}) \quad (1)$$

such that $\phi(h, r, t) > 0$ for all correct triples, and $\phi(h, r, t) < 0$ for incorrect triples. Re denotes the real part of a complex number.

3.3 Dual KG Embeddings

Here, we learn the KG Embeddings (KGE) for all C, R, E and L. The aim is to learn the orientation of literals different from entities. We extend the ComplEx (Trouillon et al., 2016) method to learn Dual KGE because it models both symmetric and asymmetric relations. However, any other KGE technique can also be used. The dual steps to learn KGE are:

(1) Using Triples T of the form (e_s, r, e_t): The entity, relation and concept embeddings $\mathbb{E}$, $\mathbb{R}$ and $\mathbb{C}$, respectively, are initialized randomly. They are then learned (shown in Figure 1) using the KGE scoring function, ϕ for all T of the form (e_s, r, e_t) where $e_s \in E, e_t \in E \cup C$ and $r \in R$ with $\mathbb{e}_\mathbb{s}$, $\mathbb{e}_\mathbb{t}$ and $\mathbb{r}$ as the respective KGE s.t.,

$$\begin{aligned} \phi(\mathbb{e}_\mathbb{s}, \mathbb{r}, \mathbb{e}_\mathbb{t}) > 0 \quad (e_s, r, e_t) \in T \\ \phi(\mathbb{e}_\mathbb{s}, \mathbb{r}, \mathbb{e}_\mathbb{t}) < 0 \quad (e_s, r, e_t) \notin T \end{aligned} \quad (2)$$

(2) Using Triples T of the form (e_s, r, l): The entity and relation embeddings, i.e. $\mathbb{E}$ and $\mathbb{R}$ are kept frozen while fine-tuning KGE $\mathbb{L}$ (initialized randomly) for L. T of the form (e_s, r, l) where $e_s \in E$, $r \in R$ and $l \in L$ (with $\mathbb{e}_\mathbb{s}$, $\mathbb{r}$ and $\mathbb{l}$ as the respective KGE) are used in this step s.t.

$$\begin{aligned} \phi(\mathbb{e}_\mathbb{s}, \mathbb{r}, \mathbb{l}) > 0 \quad (e_s, r, l) \in T \\ \phi(\mathbb{e}_\mathbb{s}, \mathbb{r}, \mathbb{l}) < 0 \quad (e_s, r, l) \notin T \end{aligned} \quad (3)$$

This ensures that the orientation of e_s w.r.t to e_t and l is learned separately. According to KG semantics, l cannot occur as the head in T. The proposed dual-step encoding ensures its preservation as opposed to the existing methods. It also enables $\mathbb{L}$ to encode type semantics along with KG structural information by initializing $\mathbb{L}$ with type-specific embeddings i.e., text or numerical embeddings (such as DICE (Sundararaman et al., 2020) for numerical reasoning (Kim et al., 2023; Bai et al., 2023)).

Note that the abstraction of concepts is required to: (1) preserve the type semantics of concepts separate to entities in the same embedding space for unified reasoning unlike other methods (Huang et al., 2023b) that encode them into different spaces; (2) "expand" all entities associated to a concept.

3.4 Link-Prediction based Symbolic Question subgraph Generation

Popular subgraph extraction methods (Sun et al., 2019a; Zhang et al., 2022) are parametric because they require a large amount of training data for learning. The non-parametric subgraph extraction methods (Das et al., 2022) are majorly dependent on kNN data points that are not available at cold start. Moreover, these methods does not support symbols/rules and are immutable.

On the other hand, question Q can provide the right cues of the KG elements involved that also eliminate the dependency on kNN samples. Link-prediction has shown promising results in modeling multi-hop relations and missing links on large KGs without any external data. Therefore, we propose a link-prediction based non-parametric method that uses parts of the question text as the seed to identify the relevant KG elements (relations, attributes, qualifiers) and form the symbolic subgraph G. We now demonstrate the generation process of G using the complex question scenarios shown in Figure 1 using the below steps:

1. *Entity and Concept Identification*: We extract the set of entities $E' \subset E$ and set of concepts $C' \subset C$ present in Q using Named Entity Recognition (NER) method. However, any off-the-shelf entity linker such as (Gu et al., 2023; Mohammed et al., 2018) can be used. As shown in Figure 1, $E' = \{\}$ and $C' = \{$*former French region*$\}$.

2. *Numerical Operation Phrase (NOP) Identification:* It identifies phrases in Q that contain operations that needs to be computed symbolically. Some examples of NOPs include *over 270 square kilometers*, *smallest*, *sum*, etc. To identify NOPs:
(a) We first mask E' and C' in Q.
(b) p-grams from this masked question are derived to encompass potential NOPs. We remove the stop words in p-grams using nltk[3] python library.
(c) The p-grams whose cosine similarity of the BERT embeddings with comparative and superlative word list (Mohammed et al., 2018) exceeds a specified threshold (0.9) are chosen. Cardinal presence (if exist) associated to them are identified using quantulum3[4] python library and appended with its corresponding p-gram to form the NOP.
Logical operators (AND, OR, NOT) and count operator (COUNT) are identified in a similar manner. BERT embeddings ensures to capture the semantic

[3] https://www.nltk.org/
[4] https://pypi.org/project/quantulum3/

similarity even if the linguistics appear differently. An NOP node for each $nop \in NOP$ is added to G with the symbolic rule:

$$G = AddNode(\texttt{NOP}[nop]) \quad \forall nop \in NOP \quad (4)$$

Figure 1 shows examples of NOP node defined using this operator.

3. *Expand*: It defines the expansion of each concept $c \in C'$ to retrieve all entities belonging to concept c. Expand node specified through the below symbolic rule is then added to G:

$$G = AddNode(\texttt{Expand}[(?^{5}, is\_a, c)]) \quad \forall c \in C' \quad (5)$$

Here, "is_a" relation depicts the *instanceOf* or *type* relation of an entity.

4. *Propagate*: The multi-part text in between entities, concepts, and NOPs in Q is identified as the *relation text*. We use transition-based pre-trained Abstract Meaning Representation (AMR) parser (Zhou et al., 2021) to identify the source and target text for each of the *relation text*. We use AMR because it provides an efficient logical representation of the question semantics. It encodes the meaning of a sentence into a rooted directed acyclic graph where nodes represent concepts and edges represent relations. Hence, it effectively provides the source and target for each relation text in Q (Refer to Section 2.4 for distinction with other works (Kapanipathi et al., 2021) that also use AMR).

If both source text and target text are identified for the *relation text* using AMR, the Propagate node with symbolic rule is added to G as:

$$G = AddNode(\texttt{Propagate} \; [(E_s, relation\_text, e_t)]) \quad (6)$$

Here, E_s is a variable that will be populated by the Symbolic Resolver and e_t is the entity present in the target text identified by the AMR.

Propagate operator also identifies the directional edges between the NOP, Expand and Propagate nodes. A directional edge is added between the source node $G_s \in G$ (corresponding to the source text) and the target node in $G_t \in G$ (corresponding to the target text).

$$G = AddEdge(getNode(G_s), getNode(G_t)) \quad (7)$$

If the target text is not identified, then the Propagate node is added to G with following rule:

$$G = AddNode(\texttt{Propagate} \; [(E_s, relation\_text, ?)]) \quad (8)$$

In this case, the node in G corresponding to the most proximal text in order (i.e., towards right) is identified as the target node G_t. A directional edge is then added using Eq. 7.

The symbolic rules in Propagate node are passed to the link-prediction scoring function ϕ to obtain the value of the variables or missing value(s) or to verify for a true triple.

Our link prediction based subgraph generation approach provides additional novel advantages as compared to existing subgraph extraction methods when: (1) backward relation inference is required; (2) the surface mention of the relation is a composition of a set of relations in the KG; (3) concepts with multiple entities are involved in the reasoning process; (4) answer(s) are not present in the subgraph (where the answer has to be computed numerically); (5) the number of hops is very large (or not known beforehand). Hence, to the best of our knowledge, *our work is the first to leverage link-prediction for non-parametric symbolic question subgraph generation.*

3.5 Relation Matching

The KGE obtained for R i.e., $\mathbb{R}$ is often ignored by the previous methods, however, it can carry important information on how a source entity is related to a target. The generated question subgraph will contain natural language text from Q as *relation text*. Therefore, to retrieve the KGE embeddings for *relation text* r_q to be used in scoring function ϕ, we perform relation matching as follows:

(1) We first obtain the sentence embedding for relation text r_q and each relation $r \in R$ using BERT.

(2) We create the vector-based index using ChromaDB[6] with the sentence embedding as features. We chose ChromaDB due to its superior performance ($5 - 7\%$) as compared to popular BM25-based retrievers (Robertson and Zaragoza, 2009) after experimentation.

(3) We retrieve the closest relation r_c based on the cosine similarity of the embedding of r_q and R.

(4) The Dual KGE of r_c is thus retrieved.

[5] ? denotes the value in the triple to be retrieved from KG

[6] https://docs.trychroma.com/

3.6 Symbolic Resolver

The symbolic question subgraph G is now resolved in the post-order traversal order as the link-prediction problem using Dual KGE scoring function ϕ, simultaneously interpreting and executing the symbolic operators as follows:

• **NOP**: The *nop* phrase in each `NOP` node G_{nop} is resolved by first parsing it to identify either the logical or a numeric operator with the numerical quantity and unit. Consider an example of a NOP phrase: *over 270 square kilometers*:

– *Numerical Quantity Identification*: the *nop* with a numeric operator contain a numerical value (in decimal form) and its unit (in SI unit form, if present) identified (see 2c. in Section 3.4) i.e., *270 kilometers* is identified as `{value: 270.0, unit: square kilometer}`.

– *Unit Conversion:* The numerical value is normalized to the unit present in the question for numeric operation execution. We achieve this by using SymPy[7] Python library.

– *Numeric operator Identification:* The p-gram of the *nop* (see 2c. in Section 3.4) is mapped to its corresponding numeric operator. For example: *greater than* is mapped to `operator.gt`, *smallest* is mapped to `np.argmin`, *not equal to* is mapped to `opertor.neq`, etc. with the help of operators[8] and numpy[9] libraries in python.

– *Retrieve Answers:* The entities that satisfy the numeric operator is populated as the entity set E_s for the next rule to be executed.

Now, consider an example of an *nop* phrase containing a logical operator OR: The intermediate output E_n of the nodes G_n that has in-degree to G_{nop} will undergo set *union* i.e.,

$$E_s = \cup E_n \quad G_n \in in\_degree(G_{nop}) \tag{9}$$

The entity set E_s is then populated for the next rule to be executed.

• **Expand**: The rule of each `Expand` node is directly executed on the KG to obtain the entity set E_s that are *instanceOf* concept $c \in C'$. The entity set E_s is then populated for the next rule to be executed.

• **Propagate**: The KGE of r_c i.e., $\mathbb{r}_{\mathbb{c}}$ is obtained for the *relation text* using Section 3.5. For the `Propagate` node with rule: (E_s, *relation text*, e_t), the embedding for e_t i.e., $\mathbb{e}_{\mathbb{t}}$ and $\mathbb{e}$ for $e \in E_s$ is obtained from the Dual KGE module. These embeddings are then passed to the KGE scoring function ϕ to retrieve E'_s (initially empty set):

$$score_e = \phi(\mathbb{e}, \mathbb{r}_{\mathbb{c}}, \mathbb{e}_{\mathbb{t}}) \quad \forall e \in E_s \tag{10}$$

$$E'_s = E'_s \cup \{e\} \text{ iff } \quad score_e > 0 \quad \forall e \in E_s \tag{11}$$

For the `Propagate` node with rule: (E_s, *relation text*, ?), the embedding $\mathbb{e}$ for $e \in E_s$ is obtained using the Dual KGE module and then passed to the scoring function ϕ to retrieve the answer e_{ans} for e to form E'_s (initially empty set). For the relations that link 2 entities:

$$e_{ans} = max(\forall_{e' \in E}\ \phi(\mathbb{e}, \mathbb{r}_{\mathbb{c}}, \mathbb{e}')) \tag{12}$$

To re-iterate, here, E is the set of all entities in the KG. For the relations that link attributes of entities:

$$e_{ans} = max(\forall_{l \in L}\ \phi(\mathbb{e}, \mathbb{r}_{\mathbb{c}}, \mathbb{l})) \tag{13}$$

Here, L is the set of attributes (literals) in the KG.

$$E_{ans} = E_{ans} \cup \{e_{ans}\} \quad \forall e \in E_s \tag{14}$$

To our knowledge, *it is the first approach to fuse neural KGE with a symbolic approach to enable joint compositional and numerical reasoning.*

4 Experiments

Our experiments answer the following research questions:

1. How does NS-KGQA compare against existing LLM-based KGQA frameworks in zero-shot setting?
2. What are the contributions of each module of NS-KGQA?
3. What are the different sources of errors?

4.1 Datasets

We experiment with 3 datasets having different: (a) question complexities; (b) KG domain; and (c) number of inference hops required. The statistics of these datasets are shown in Table 1.

1. **KQA Pro** (Cao et al., 2022a): This recent dataset contains much **harder and complex QA pairs** with numerical quantities, concepts, and entities for multi-hop compositional and numerical reasoning unlike other datasets (Gu et al., 2021b; Dutt et al., 2023). It contains questions that require up to 10-hops answerable through Wikidata KG. Its test set is closed, hence, we use dev set as test.

[7] https://pypi.org/project/sympy
[8] https://pypi.org/project/operators
[9] https://pypi.org/project/numpy

Dataset	Train	Val	Test
KQA Pro	94,376	11,797	11,797
MetaQA 1-hop	96,106	9,992	9,947
MetaQA 2-hop	118,948	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274
WebQSP	2,998	100	1,639

Table 1: Statistics of the Datasets used

2. **MetaQA** (Zhang et al., 2018): It is the largest multi-hop dataset widely used (Choi et al., 2023; Agarwal et al., 2024a,b) for **domain-specific** (movie) KGQA. It provides upto 3-hop QA pairs.

3. **WebQSP** (Yih et al., 2016): This dataset is based on Freebase KG that contains questions upto 2-hops from **Google query logs**. It exhibits more complex structures i.e., Isomorphisms (ISO-4) as compared to other datasets (Gu et al., 2021b).

4.2 Baselines

We compare against the following latest LLM-based techniques that support zero-shot setting:

1. KQA Pro: FlexKBQA (Li et al., 2023b), InteractiveKBQA (Xiong et al., 2024).

2. WebQSP: FlexKBQA (Li et al., 2023b), KAPING (Baek et al., 2023), KQG-COT+ (Liang et al., 2023).

3. MetaQA: BYOKG (Agarwal et al., 2024a).

No other LLM-based work enables zero-shot setting, hence, can't be used as baselines. Alternatively, we compare the performance of our approach on KQA Pro with the most powerful closed-source[13] and open-source LLMs (see Table 4) when used directly to answer the questions.

We additionally contrast our approach against SOTA IR based fully-supervised techniques for each dataset. It includes KVMemNet (Miller et al., 2016), EmbedKGQA (Saxena et al., 2020), SRN (Qiu et al., 2020), RGCN (Schlichtkrull et al., 2018), and Subgraph Retrieval (Zhang et al., 2022). SRN and Subgraph Retrieval are based on subgraph extraction(SE).

Although comparison with SP-based KGQA methods is not apples-to-apples, we still include two SOTA, fully-supervised SP-based methods i.e., GraphQ IR (Nie et al., 2022) and BART-based KoPL (Cao et al., 2022a) for KQA Pro. Refer to Appendix A.2 for more details of each baseline.

4.3 Implementation and Hyper-parameters

We use the PyTorch[14] framework in Python to implement NS-KGQA using Nvidia A100 GPUs. Dual KGE are learned using Cross Entropy Loss with Adam Optimizer for 500 epochs with a learning rate of 5e-3 and a batch size of 256. The dimension of KGE is set to 400. all-distilroberta-v1[15] BERT model is used during NOP identification and relation matching. We use Accuracy[16] for KQA Pro, Hits@1 for MetaQA, and the F1 score for WebQSP as the evaluation metrics. Section 5.5 provides the compute resources utilized.

5 Results and Analysis

The results on each dataset are discussed below. Qualitative analysis is provided in Appendix A.1.

5.1 Results: KQA Pro

The results of NS-KGQA compared with baselines are shown in Table 3. Detailed category-wise performance is provided in Table 2. The comparison with SOTA LLMs is shown in Table 4.

As shown, NS-KGQA outperforms all zero-shot baselines and SOTA LLMs to *achieve a new SOTA in the zero-shot setting for KQA Pro*. It beats FlexKBQA and Interactive KBQA by 24.56% and 27.79% respectively. This performance gap is attributed to the dependency of the baselines on LLMs to generate the LF. Additionally, most of the LLMs struggles to handle complex questions that require numerical reasoning (Tan et al., 2023). Also, note that even though the core step of InteractiveKBQA for interaction i.e., deconstruction of question into sub-query triples is similar to the proposed NS-KGQA, but, in Interactive KGQA, this step is manual and hence, not scalable. Whereas, in NS-KGQA, this step is powered by AMR and symbols, making it robust and scalable.

NS-KGQA outperforming SOTA LLMs again indicates the difficulty of answering complex questions solely by LLMs. NS-KGQA additionally beats fully-supervised IR-based methods as well. This is because these baselines are unable to handle a richer variety of complex questions as provided by KQA Pro. This demonstrates that NS-KGQA

[10] https://huggingface.co/codellama/CodeLlama-34b-hf

[11] https://huggingface.co/meta-llama/Meta-Llama-3-70B

[12] https://ai.google.dev/gemini-api/docs/models#gemini-1.5-pro

[13] on popular models ONLY due to high costs associated

[14] https://pypi.org/project/torch/

[15] https://huggingface.co/sentence-transformers/all-distilroberta-v1

[16] Each question in KQA Pro has only one answer, hence Hits@1 and Accuracy values will be the same.

Models	Overall	Multi-hop	Comparison	Logical	Count	Verify	Qualifier
			fully-supervised models				
KVMemNet	6.9	0.05	0.08	1.64	0.15	54.48	0.05
SRN	-	11.84	-	-	-	-	-
RGCN	29.12	10.5	11.86	20.69	36.82	66.41	22.89
EmbedKGQA	20.27	13.22	11.84	12.28	20.49	40.43	14.87
Subgraph Retrieval	22.82	12.44	14.72	18.34	21.92	63.39	15.42
			zero-shot models				
FlexKBQA	28.48	13.81	29.75	22.44	**29.36**	62.54	**21.81**
InteractiveKBQA	25.25	-	-	-	-	-	-
NS-KGQA	**53.04**	**34.04**	**69.5**	**44.04**	27.49	**99.33**	10.29

Table 2: Category-wise accuracy of different models on KQA Pro dataset on dev set (The best is highlighted in bold)

Method	Models	Acc%
Fully-Supervised (IR-based)	KVMemNet	6.90
	EmbedKGQA	20.27
	SRN (SE)	11.84
	RGCN	29.12
	Subgraph Retrieval (SE)	22.82
Fully-Supervised (SP-based)	BART + KoPL	83.28
	GraphQ IR	79.13
Zero-Shot	FlexKBQA	28.48
	Interactive KBQA	25.25
	NS-KGQA	**53.04**

Table 3: Results of KQA Pro on dev set (baselines reproduced on dev set, hence, numbers will differ from original work)

Closed-source	Acc%	Open-source	Acc%
ChatGPT[17]	24.96	CodeLlama Ins. 34B[10]	28.7
Davinci-003	31.02	Llama-3 70B[11]	33.2
GPT-4	37.43	Gemini 1.5 Pro[12]	27.5
	NS-KGQA	**53.04**	

Table 4: Comparison with SOTA LLMs (on KQA Pro)

is generalized to perform any type of compositional and numerical reasoning, and hence, can answer any complex question in a zero-shot setting. NS-KGQA essentially divides the question into multiple grounded steps making the *reasoning process more transparent*.

The performance gap with fully-supervised SP-based methods is attributed to the use of a large amount of LF such as SPARQL as strong supervision signals, whereas, our framework addresses a more challenging setting, assuming no QA and LF data is available at cold-start for new KGs.

5.2 Results: MetaQA and WebQSP

The results of NS-KGQA on MetaQA are shown in Table 5b. *NS-KGQA achieves a new SOTA on all hops in the zero-shot setting for MetaQA*. Specifically, it beats BYOKG by 31%. The results on WebQSP are shown in Table 5a. NS-KGQA beats KAPING and KQG-COT+ by a significant margin. FlexKBQA outperforms NS-KGQA on WebQSP due to its dependency on the ChatGPT[17] model, which is trained on a large body of knowledge.

The performance gap with fully-supervised IR-based methods for these datasets is attributed to the availability of a large number of QA pairs for learning the model. To re-iterate, *our framework addresses a more challenging zero-shot setting where fully-supervised baselines are inapplicable.*

5.3 Ablation Study

We analyze the effect of different modules of NS-KGQA, mainly: (1) Dual KG Embedding and (2) Link-Prediction based Symbolic Question subgraph Generator. The Symbolic Resolver is the core execution engine of our proposed framework and hence cannot be ablated. Table 6 shows the performance for the following settings:

1. NS-KGQA - Dual KGE: Here, we replace the Dual KGE with traditional KG embeddings (T-KGE) with single encoding where literals are treated the same as entities. We observed that the overall performance degrades by $\sim$ 15%. This demonstrates that the proposed Dual KGE provides advantages by encoding literals separately with the help of a dual encoder. We particularly gained performance in answering the questions where literals were involved in the reasoning process.

2. NS-KGQA + Oracle SQG: Here, we quantify how far the proposed link-prediction based *S*ymbolic *Q*uestion sub*G*raph (SQG) generator is from the Oracle SQG. To obtain the Oracle SQG, we convert the KoPL program of each question (provided with the dataset) to the link-prediction based SQG. We observe an improvement of $\sim$ 18% when the Oracle SQG is used in our framework. This depicts that our proposed link-prediction based SQG generation method is worth further exploration to reach closer to the Oracle SQG.

5.4 Error Analysis

We analyze the following sources of errors:

[17] https://chat.openai.com

Method	Models	F1%
Fully-Supervised (IR-based)	EmbedKGQA	66.6
	Subgraph Retrieval (SE)	66.7
	KVMemNet	46.7
	RGCN	37.2
Zero-Shot	FlexKBQA	**46.2**
	KAPING (T5-11B)	24.91
	KQG-COT+	35.22
	NS-KGQA	<u>39.31</u>

(a) WebQSP

Method	Models	Hits@1		
		1-hop	2-hop	3-hop
Fully-Supervised (IR-based)	KVMemNet	96.2	82.7	48.9
	EmbedKGQA	97.5	98.8	94.8
	SRN (SE)	97.0	95.1	75.2
	RGCN	-	-	-
	Subgraph Retrieval (SE)	-	-	-
Zero-Shot	BYOKG	<u>41.67</u>	<u>11.42</u>	<u>27.84</u>
	NS-KGQA	**62.54**	**64.19**	**60.32**

(b) MetaQA

Table 5: NS-KGQA Results (Bold and underline denote best and second best performance **among zero-shot setting**)

Model Settings	Acc%
NS-KGQA	53.04
NS-KGQA - Dual KGE	37.4 (↓ 15.64)
NS-KGQA + Oracle SQG	71.3 (↑ 18.26)

Table 6: Effect of different modules (for KQA Pro)

1. *Entity Linking*: Observed only in KQA Pro, for 0.69% of questions.

2. *External tools*: There can be multiple ways to represent a numerical quantity in natural language. For eg: square kilometers can be represented as `sq km`, `km2`, etc. out of which some could be difficult to detect by *quantulum3* and *SymPy* python libraries for quantity extraction and unit conversion respectively. Hence, we observe approx. 8% error from these tools in NOP identification.

3. *Incorrect Symbolic Question subgraph Generation*: We found out that:

(a) Some questions are also difficult to understand by humans. For example: "`Seymour Cassel got nominated for Academy Award for best supporting actor, tell me the subject of this statement`". The phrase `subject of this statement` makes the question unclear as to what exactly is meant by the `subject` here. There is no relation that depicts the `subject` in the KG, making it even more difficult to understand and retrieve the correct answer.

(b) Some questions are confusing as to what exactly needs to be retrieved. For example: "`What town in the United States has the postal code [06880] and what is the area?`". Here, the `town` and `area` are retrieved but gets confused about what to return as the answer because both `town` and `area` are asked in the question.

(c) There are two sub-types of *Count* questions:
i. Count Entities: counting the retrieved entities;
ii. Retrieve Count: retrieve count present in KG.
The Symbolic question subgraph generator is unable to distinguish among them. For Count Entities, it retrieves all entities that have a score > 0 (See Eq. 11) and performs a count on top of it. Hence, the set of entities whose score > 0 becomes a superset of the answer, and even if we consider top-k entities, it may become the subset of the answer. Hence, our approach does not guarantee that the entity set on which the count will be performed is indeed the right subset even if the intermediate answers are correct. Most of the existing information retrieval based techniques fail to handle this scenario. endasparaenum These issues constitute about 24% of errors.

5.5 Inference Latency

We use NVIDIA V100 GPU with 32 GB RAM to learn Dual KGE that takes approx. 6 GPU hours. After learning KGE, the entire NS-KGQA pipeline runs on a CPU with average latency of 2s/question. On the other hand, most of the LLM-based methods have more average latency per API call itself (approx. 8s/call) and their entire pipeline entirely runs on GPU that adds up approx. 5s/question additionally.

6 Conclusion

We propose NS-KGQA, a novel zero-shot neuro-symbolic approach for complex KGQA. To the best of our knowledge, it is the first one to fuse the neural KGE approach with the symbolic approach without the need for any external data. The core of NS-KGQA is to preserve the query capability of the KG through the proposed link-prediction based reasoning (neural approximation). Only the "symbolic" parts essential to answer the complex question are abstracted while leveraging Dual KG Embeddings for querying. Our extensive experiments show that NS-KGQA outperforms LLM-based zero-shot baselines by a significant margin, setting a new benchmark for various Complex KGQA datasets. It discovers the weakness of current KGQA techniques in handling complex question types. NS-KGQA is independent of any Logical Form and, hence, applicable to any new KG and settings at cold-start while being interpretable.

7 Limitations

Despite harnessing the interpretability of Symbols and robustness of the neural model, obtaining a step-wise question subgraph is still prone to errors, which impacts the performance of NS-KGQA, as discussed in Section 5.4. Most of the errors stem from an incorrect interpretation and decomposition of the question w.r.t the KG. Fortunately, if the framework provides a wrong answer, the step that went wrong can be easily interpreted and can be rectified with human intervention.

8 Risks

Our work does not have any obvious risks that we are aware of.

Acknowledgements

Srikanta Bedathur acknowledges the DS Chair Professor of AI fellowship. This work was partially supported by an IITD-IBM AIHN collaborative project. The authors thank the IIT Delhi HPC facility for computational resources.

References

Dhruv Agarwal, Rajarshi Das, Sopan Khosla, and Rashmi Gangadharaiah. 2024a. Bring your own KG: Self-supervised program synthesis for zero-shot KGQA. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 896–919, Mexico City, Mexico. Association for Computational Linguistics.

Prerna Agarwal, Nishant Kumar, and Srikanta Bedathur. 2024b. SymKGQA: Few-shot knowledge graph question answering via symbolic program generation and execution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10119–10140, Bangkok, Thailand. Association for Computational Linguistics.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada. Association for Computational Linguistics.

Jiaxin Bai, Chen Luo, zheng li, Qingyu Yin, Bing Yin, and Yangqiu Song. 2023. Knowledge graph reasoning over entities and numerical values. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 57–68, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022b. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

Hyeong Kyu Choi, Seunghun Lee, Jaewon Chu, and Hyunwoo J Kim. 2023. Nutrea: Neural tree search for context-guided multi-hop kgqa. In *Advances in Neural Information Processing Systems*, volume 36, pages 35954–35965. Curran Associates, Inc.

Rajarshi Das, Ameya Godbole, Ankita Naik, Elliot Tower, Robin Jia, Manzil Zaheer, Hannaneh Hajishirzi, and Andrew McCallum. 2022. Knowledge base question answering by case-based reasoning over subgraphs. In *ICML*.

Ritam Dutt, Sopan Khosla, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiah. 2023. GrailQA++: A challenging zero-shot benchmark for knowledge base question answering. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–909, Nusa Dua, Bali. Association for Computational Linguistics.

Haishuo Fang, Xiaodan Zhu, and Iryna Gurevych. 2024. DARA: Decomposition-alignment-reasoning autonomous language agent for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3406–3432, Bangkok, Thailand. Association for Computational Linguistics.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021a. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

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

Xiang Huang, Sitao Cheng, Yuheng Bao, Shanshan Huang, and Yuzhong Qu. 2023a. MarkQA: A large scale KBQA dataset with numerical reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10241–10259, Singapore. Association for Computational Linguistics.

Zijie Huang, Daheng Wang, Binxuan Huang, Chenwei Zhang, Jingbo Shang, Yan Liang, Zhengyang Wang, Xian Li, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2023b. Concept2Box: Joint geometric embeddings for learning two-view knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10105–10118, Toronto, Canada. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Gayeong Kim, Sookyung Kim, Ko Keun Kim, Suchan Park, Heesoo Jung, and Hogun Park. 2023. Exploiting relation-aware attribute representation learning in knowledge graph embedding for numerical reasoning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 1086–1096, New York, NY, USA. Association for Computing Machinery.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023a. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2023b. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering.

Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. 2023. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4329–4343, Singapore. Association for Computational Linguistics.

Zhixuan Liu, Zihao Wang, Yuan Lin, and Hang Li. 2022. A neural-symbolic approach to natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2159–2172, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2039–2056, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana. Association for Computational Linguistics.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G P Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme Lima, Ryan Riegel, Francois Luus, and L V Subramaniam. 2022. SYGMA: A system for generalizable and modular question answering over knowledge bases. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3866–3879, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. ICML'11, page 809–816, Madison, WI, USA. Omnipress.

Lunyiu Nie, Shulin Cao, Jiaxin Shi, Jiuding Sun, Qi Tian, Lei Hou, Juanzi Li, and Jidong Zhai. 2022. GraphQ IR: Unifying the semantic parsing of graph query languages with one intermediate representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5848–5865, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yilin Niu, Fei Huang, Wei Liu, Jianwei Cui, Bin Wang, and Minlie Huang. 2023. Bridging the gap between synthetic and natural questions via sentence decomposition

for semantic parsing. *Transactions of the Association for Computational Linguistics*, 11:367–383.

OpenAI, :, Josh Achiam, and Steven Adler. 2023. Gpt-4 technical report.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. WSDM '20, page 474–482, New York, NY, USA. Association for Computing Machinery.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

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, Online. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. TransferNet: An effective and transparent framework for multi-hop question answering over relation graph. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019a. 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)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753, Online. Association for Computational Linguistics.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. In *The Semantic Web – ISWC 2023*, pages 348–367, Cham. Springer Nature Switzerland.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org.

Chaojie Wang, Yishi Xu, Zhong Peng, Chenxi Zhang, Bo Chen, Xinrun Wang, Lei Feng, and Bo An. 2023a. keqing: knowledge-based question answering is a nature chain-of-thought mentor of llm.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023b. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-KBQA: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10561–10582, Bangkok, Thailand. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

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.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784, Dublin, Ireland. Association for Computational Linguistics.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. Association for Computational Linguistics.

A Appendix

A.1 Qualitative Analysis

Examples demonstrating different question types handled by NS-KGQA are provided in Table 7.

A.2 Baselines

- **KVMemNet** (Miller et al., 2016) performs QA by first storing facts in a key-value structured memory before reasoning on them to predict an answer. At each reasoning step, the information collected from the memory is cumulatively added to the original query to build context for the next reasoning iteration.
- **SRN** (Qiu et al., 2020) model starts from the question entity and uses a path search technique to predict the relation path sequence to reach the target entity.
- **RGCN** (Schlichtkrull et al., 2018) uses a graph convolution network-based technique to encode KG in graph form and perform QA.
- **EmbedKGQA** (Saxena et al., 2020) uses KG embeddings to perform multi-hop reasoning using a RoBERTa-based question encoder.
- **Subgraph Retrieval** (Zhang et al., 2022) use a dual-encoder that provides better retrieval compared to existing retrieval methods.
- **BART + KoPL** (Cao et al., 2022a) is an end-to-end generation model that directly produces the corresponding KoPL program steps given a question. It is worth noting that the pre-trained BART model is forced to have the capability to memorize the relations and entities present in the KG.
- **GraphQ IR** (Nie et al., 2022) proposes a unified intermediate representation for graph query languages, named GraphQ IR. It has a natural-language-like expression that bridges the semantic gap and formally defined syntax that maintains the graph structure. A neural semantic parser is used to convert user queries into GraphQ IR, which can be later losslessly compiled into various downstream graph query languages such as SPARQL, Lambda DCS, etc.
- **FlexKBQA** (Li et al., 2023b) utilizing Large Language Models (LLMs) as program translators. It leverages automated algorithms to sample diverse programs, such as SPARQL queries, from the knowledge base, which are subsequently converted into natural language questions via LLMs. They used this synthetic data set to facilitate training of a specialized lightweight model for a KG.
- **BYOKG (Bring your own KG)** (Agarwal et al., 2024a) leverage an agent based on LLM to understand the information of KG through exploration.
- **KAPING** (Baek et al., 2023) Knowledge Augmented language model PromptING (KAPING) framework first retrieves the top-K similar triples to the question with the knowledge retriever and then augments them as the form of the prompt.
- **KQG-CoT+** (Liang et al., 2023) first retrieves supportive logical forms from the unlabeled data set, taking into account the characteristics of the logical form. Then, write a prompt to explicit the reasoning chain of generating complicated questions based on the selected demonstrations. To ensure prompt quality, they sort the logical forms by their complexity for in-context learning.
- **InteractiveKBQA** (Xiong et al., 2024) generates logical forms through direct interaction with KG. They use examples to guide LLMs through the reasoning processes. They deconstruct the question into sub-query triples but, this step is manual and, hence not scalable.

Question Type		Example	Good/Bad
Multi-hop	Question	Which {feature film} was distributed by [Walt Disney Pictures]?	
	SQG*	[Expand(?, is a, feature film), Propagate(E_s, was distributed by, Walt Disney Pictures)]	Good
	Question	What {war} has the participant [Hannibal] who died from suicide	
	SQG	[Expand(?, is a, war), Propagate(E_s, participant, Hannibal), Propagate(E_s, died from suicide, Hannibal)]	Bad
Comparison	Question	Does [My Neighbor Totoro] or [Hannah Arendt], possess the longer run-time?	
	SQG	[Propagate(My Neighbor Totoro, run-time, ?), Propagate(Hannah Arendt, run-time, ?), NOP(longer)]	Good
	Question	Which one has more population between [Boston] and [Rocky Mount]?	
	SQG	[Propagate(Boston, one moe population, ?), Propagate(Rocky Mount, more population, ?)]	Bad
Logical	Question	Which {former French region} has the smallest population and a population that is not equal to 97000?	
	SQG	[Expand(?, is a, former French region), Propagate(E_s, population,?), NOP(smallest), NOP(AND), Propagate(E_s, population, ?), NOP(not equal to 97000)]	Good
	Question	Which one among the {census-designated place}, with the population of more or less than 110,000, is the highest above sea level?	
	SQG	[Expand(?, is a, census-designated place), Propagate(E_s, population, ?), NOP(more), NOP(OR), NOP (less than 110000), Propagate(E_s, above sea level, ?), NOP(highest)]	Bad
Count	Question	How many {Pennsylvania counties} have a population greater than 7800 or a population less than 40000000?	
	SQG	[Expand(?, is a, Pennsylvania county), Propagate(E_s, population, ?), NOP(greater than 7800), NOP(OR), NOP(less than 40000000), NOP(COUNT)]	Good
	Question	How many episodes does the children's movie [Bewitched] contain?	
	SQG	[Propagate(Bewitched, episodes, ?), NOP(COUNT)]	Bad
Verify	Question	Is [http://www.west-chester.com] the official website of the {animated feature film} titled [Waltz with Bashir]?	
	SQG	[Expand(?, is a, animated feature film), Propagate(E_s, official website, http://www.west-chester.com), Propagate(E_s, titled, Waltz with Bashir)]	Good
	Question	Does Libris-URI [sq468r1b0m8d5g4] designate the [Caldecott Medal] winning individual for [Where the Wild Things Are]?	
	SQG	[Propagate(sq468r1b0m8d5g4, designate the, Caldecott Medal), Propagate(Caldecott Medal, winning individual for, Where the Wild Things Are)]	Bad

Table 7: Qualitative Analysis on KQA Pro (*SQG denotes the post-order traversal of Symbolic Question subgraph (SQG) generated by NS-KGQA)