# A Neural Approach to KGQA via SPARQL Silhouette Generation

Anonymous ACL submission

## Abstract

Semantic parsing is a predominant approach to solve the Knowledge Graph Question Answering (KGQA) task where, natural language question is translated into a logic form such as SPARQL. Semantic parsing based solutions are mostly modular/pipelined where, noise introduced by the upstream modules for entity/relation linking makes it hard to solve the complex questions. Recently, Neural Machine Translation (NMT) based approaches have emerged that are capable of handling complex questions. However, NMT-based approaches struggle with handling the large number of test entities and relations that are unseen during training. In this work,we propose a modular two-stage neural approach which combines best of both the worlds - NMT and semantic parsing pipeline. Stage-I of our approach comprises an NMT-based seq2seq module that translates a question into a sketch of the desired SPARQL, called as SPARQL silhouette. This stage also contains a noise simulator which combines the masking scheme with an entity/relation linker in a novel manner so as to take care of unseen entities/relation without blowing up the vocabulary of seq2seq module. Stage-II of our approach comprises a Neural Graph Search (NGS) module which aims to distil the SPARQL silhouette in order to reduce the entity/relation linking noise. Experimental results show that, the quality of generated SPARQL silhouette is impressive for an ideal scenario where entity/relation linker is noise-free. For the realistic scenario (i.e. noisy linker), the quality of the SPARQL silhouette drops but our NGS module recovers it considerably. We show that, our proposed approach improves state-of-the-art on LC-QuAD-1 dataset by an absolute margin of $3.72\%$ $F_1$.

## 1 Introduction

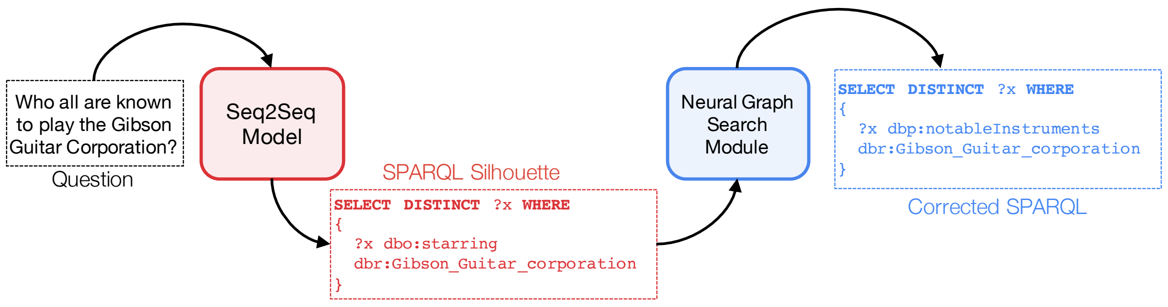
Knowledge Graph (KG) is a large collection of real world facts that are stored in the form of triples such as $\langle$*Joe Biden*, *president*, *United States*$\rangle$, where, "*Joe Biden*" and "*United States*" are entities and "*president*" is a relation between them. The task of Knowledge Graph Question Answering (KGQA) is an important application where a system is required to answer a natural language question by leveraging the facts present in the given KG. A KGQA system permits users retrieving the information from KG without any prior knowledge about KG schema or query languages such as SPARQL, SQL, etc. The availability of large-scale KGs, such as Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015), YAGO (Pellissier Tanon et al., 2020), NELL (Mitchell et al., 2015), Google's Knowledge Graph (Steiner et al., 2012), and their applicability in various business applications have made the KGQA an important research area within NLP.

There are several approaches proposed to solve the KGQA task and can be grouped into two broad categories:

1. *Semantic parsing-based*: In these approaches (Berant and Liang, 2014; Reddy et al., 2014; Dong and Lapata, 2016), a natural language question is first transformed into a structured query language or logic form such as SPARQL, SQL, $\lambda$-DCS (Liang, 2013), CCG (Zettlemoyer and Collins, 2005). Generated query is then executed against the given KG to get the answer of the given question.

2. *Information extraction-based*: These approaches (Yao and Van Durme, 2014; Dong et al., 2015; Bordes et al., 2015) extract a subgraph from the underlying KG which depends on the entities/relations present in the question. Next, they perform graph-based reasoning on the subgraph to reach the final answer directly without generating any intermediate logic form.

The popular semantic parsing-based approaches for KGQA (Singh et al., 2018; Kapanipathi et al., 2020; Liang et al., 2021) are inherently modular and pipelined in nature because they decompose



Who all are known to play the Gibson Guitar Corporation?
Question
Seq2Seq Model
SPARQL Silhouette
Neural Graph Search Module
Corrected SPARQL

```
SELECT DISTINCT ?x WHERE
{
  ?x dbo:starring
  dbr:Gibson_Guitar_corporation
}
```

```
SELECT DISTINCT ?x WHERE
{
  ?x dbp:notableInstruments
  dbr:Gibson_Guitar_corporation
}
```

Figure 1: A high level view of our proposed two stage neural architecture for KGQA.

the problem of logic form generation into several subtasks: (i) question understanding, (ii) linking mentions in the question text to the entities and relations in the KG, (iii) generating the final logic form with various constraints required to get the answer. Some of these modules, for example, entity/relation linking, are generally far from being perfect and hence induce a noticeable amount of noise in the pipeline. Because of this, such approaches (Petrochuk and Zettlemoyer, 2018) end up handling simple questions well where just a single KG fact is needed to answer the question. These approaches, however, struggle while handling complex questions (Li et al., 2016; Usbeck et al., 2017; Trivedi et al., 2017) that require multiple facts. For complex questions, one of the key challenge lies in the large sized search space when linking entities and relations. Other drawback of the pipeline-based approaches involves propagation of the noise from upstream modules into downstream modules where, usually there is no explicit provision to correct the noise.

Given the recent advancements in Neural Machine Translation (NMT) (Bahdanau et al., 2015) technologies, it is emerging as an alternative approach (Yin et al., 2021; Cai et al., 2017) for semantic parsing with a hope of alleviating the limitations of pipelined approaches. Like language translation, NMT-based approaches translate natural language questions to the logic form directly. NMT-based approaches are good at syntactic and semantic understanding of the complex questions. However, NMT approaches have their own limitations - (i) they require large amount of training data, (ii) they cannot handle unseen entities/relations at test-time due to their fixed vocabulary. Motivated by these limitations, we propose a novel *two-stage neural* approach for KGQA (see Figure 1). This approach embraces the best of both the worlds – (i) using NMT for handling complex questions, (ii) using masking technique with entity/relation linking module to handle unseen entities/relations.

The Main contributions of this work are as follows:

1. In Stage-I of our approach, a NMT-based module generates the sketch of the target SPARQL for the given natural language question. We call this sketch as *SPARQL silhouette*. In this stage, we exploit the idea of *masking* to mask all the entities and relations present in the input question and thereby, freeing up the NMT module from the task of entity/relation linking. We, however, handle the entity/relation linking separately via an off-the-shelf entity/relation linker. We further leverage the masking scheme to simulate the noise level in entity/relation linking process for the purpose of ablation studies.
2. Stage-II comprises a *Neural Graph Search (NGS)* module. This module takes the SPARQL silhouette as inputs and reduces noise introduced by the entity/relation linker in Stage-I.
3. To demonstrate the effectiveness of our approach, we first simulate a scenario where entity/relation linker is noise-free (i.e. $100\%$ $F_1$ score). Then we show that, the quality of the resulting SPARQL silhouette in Satge-I is impressive – $83.08\%$ $F_1$ for LC-QuAD-1 and $55.3\%$ Macro $F_1$ QALD for QALD-9 dataset.
4. Next, we simulate a real entity/relation linker and show that as $F_1$ of this linker goes down, the quality of the resulting SPARQL silhouette drops. Finally, integrating Stage-II module with Stage-I boosts the performance significantly and improves the SOTA by an absolute margin of $3.72\%$ F1 for LC-QuAD-1 dataset.

## 2 Related Work

There is a vast body of literature on the KGQA task and its nearly impossible to cover all of them here. One can refer to survey papers (Chakraborty et al., 2019; Lan et al., 2021) for an in-depth account of KGQA literature. By and large, semantic parsing is a predominant paradigm in the KGQA literature. Semantic parsing-based approaches can be classified into two categories: (i) *neural*, and (ii) *non-neural*. Non-neural approaches (Zettlemoyer and Collins, 2007; Berant et al., 2013; Diefenbach et al., 2017) are somewhat dated now and use handcrafted features and rules. As far as neural semantic parsing approaches are concerned, they can be further divided into two subcategories: (i) ) *NMT-based*, (ii) ) *Non-NMT-based*. NMT-based approaches are the most recent ones and are inspired by the phenomenal success of neural machine translation techniques. On the other hand, majority of the non-NMT-based approaches are based on the modular pipeline architecture. Each variant has its own pros and cons as discussed in Section 1. Given our proposed approach is inspired by both NMT and pipeline-based approaches, in what follows, we give a brief account of the most relevant prior art for both.

### 2.1 Non-NMT-based Approaches

Pipeline-based approaches (Singh et al., 2018; Kapanipathi et al., 2020; Liang et al., 2021) break the problem of semantic parsing a complex question (Bao et al., 2016; Su et al., 2016; Trivedi et al., 2017; Dubey et al., 2019) into more manageable subtasks such as question understanding, entity/relation linking, logic form generation and use reusable modules for solving the subtasks. In these approaches, all intermediate modules need not be neural-based. Each of these submodules introduces its own errors, which propagate to the downstream pipeline. Another line of works (Yih et al., 2015; Maheshwari et al., 2019; Ding et al., 2019; Lan and Jiang, 2020; Chen et al., 2020) maps the problem of semantic parsing on KG to a query graph (a subgraph of KG) generation, which can be easily translated into the SPARQL.

### 2.2 NMT-based Approaches

In the last few years, NMT-based approaches are being used to translate natural language questions into SQL (Zhong et al., 2017; Yu et al., 2018; Cai et al., 2018). Recently, Yin et al. (2021) proposed a CNN-based seq2seq models to generate SPARQL queries from natural language questions. One limitation of their approach is that output vocabulary for SPARQL generation is limited to the entities/relations seen during training. As a result, their performance reduces drastically if the overlap of entities and relations in the training and test sets differ. Our proposed method can efficiently handle unseen entities/relations during test time. To the best of our knowledge, our work is the first of its kind of solving KGQA task which considers multiple relations and used NMT based approach that can handle unseen entities/relations and reduce vocabulary size by designing noise simulator with masking strategy.

## 3 The KGQA Task

In KGQA, we are given a Knowledge Graph $\mathcal{G}$ comprising of an entity set $\mathcal{E}$, a relation set $\mathcal{R}$, and a set of knowledge facts $\mathcal{F}$. The knowledge facts are expressed in the form of triples; $\mathcal{F} = \{\langle e_s, r, e_o \rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $e_s \in \mathcal{E}$ is known as *subject* or *head* entity, $e_o \in \mathcal{E}$ is known as *object* or *tail* entity, and $r$ is a relation which connects these two entities. These entities (relations) form the nodes (edges) of the KG. The task now is to identify the subset of entities from $\mathcal{E}$ that constitute the answer of a given question $Q$ in the natural language form. The most common family of approaches for the KGQA task is *semantic parsing* where, the given question $Q$ is first translated into an SPARQL query $S$ which is then executed over the KG so as to get the answer set. For developing a system to convert a question into the corresponding SPARQL query, we are given a set of training data $\{Q_i, S_i, A_i\}_{i=1}^{n}$, where $Q_i$ is a question (in natural language text), $S_i$ is the SPARQL query, and $A_i$ is the answer set obtained by executing $S_i$ on $\mathcal{G}$.

In this paper, we propose a two-stage system for KGQA. In Stage-I, seq2seq module generates a SPARQL silhouette with specific entities. Relations predicted in this module are corrected by the *neural graph search module* in Stage-II.

## 4 Stage-I: Seq2Seq Model

Figure 2 shows the architecture of Stage-I of our approach. The input question is first given to an external entity/relation linker where it detects the surface form mentions of the entities/relations in the question text and links the same to the entities/relations in the underlying KG (DBpedia herein). The output of linker, along with the question, is passed to a *noise simulator* module which masks mentions of all the entities/relations in the question text and corresponding gold SPARQL for the training data. This module employs different masking schemes depending on the desired level of noise that we wish to simulate. The masked question is then passed to a CNN-based sequence-to-sequence (seq2seq) module which converts it into a SPARQL silhouette.

Note, seq2seq models have achieved state-of-the-art performance in machine translation task (Yin and Neubig, 2017) and they can be based on RNN, CNN, or Transformer architectures. The prior research has shown (Yin et al., 2021) that CNN-based seq2seq model performs best for translating natural language to SPARQL query. In our preliminary experiments also CNN-based model performed better than the transformer based model (Table 6 of appendix). Hence, we opted CNN-based seq2seq model as our base model for Stage I.

### 4.1 Noise Simulator and Masking

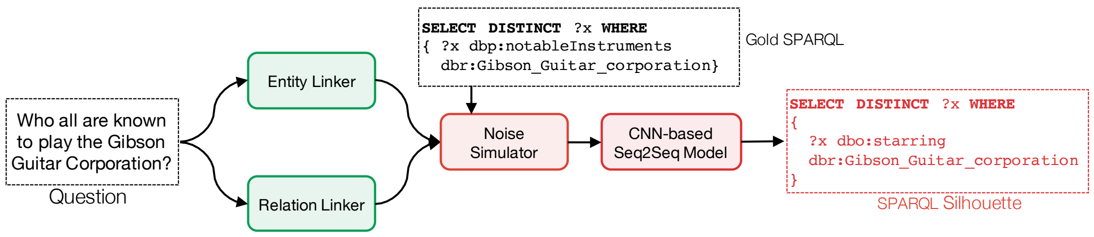
The purpose of designing noise simulator module is two-fold: (i) To mask mentions and entities/relations in the question text as well as SPARQL, (ii) To simulate varying levels of noise in the entity/relation linking process. Masking helps us in two ways: (i) handling test entities/relations that are unseen during training, (ii) reducing vocabulary size as KGs contain a large number of entities and relations. A simple neural seq2seq model which translates natural language question into a SPARQL query will struggle to output some of the entities/relations during test time that are unseen during training time and hence will not be available in the output vocabulary. In the absence of linking and masking, our elementary experiment shows the performance of seq2seq model to be quite low ($F_1$ score 16%). Table 1 further corroborates this behavior where we have captured the statistics about % of entities and relations (i.e. properties and ontology in DBpedia) in validation and test sets that are seen in the training set. This suggests that entity/relation linker along with masking is a must for any seq2seq model. Even if we were to work with perfect linker, without masking, the output vocabulary of the seq2seq model would be over growing which would become difficult to manage. To handle such difficulties, we need masking. Here, we propose three different types of masking schemes (Scenario A, B and C) and describe these in subsequent subsections .

| Dataset | Statistics | Val | Test |
|---|---|---|---|
| LC-QuAD-1 | Entities (dbr) | 52.3 | 46.8 |
| | Properties (dbp) | 97.2 | 98.3 |
| | Ontologies (dbo) | 96.5 | 94.6 |
| QALD-9 | Entities (dbr) | 27.1 | 25.9 |
| | Properties (dbp) | 0.0 | 16.9 |
| | Ontologies (dbo) | 47.8 | 38.3 |

Table 1: % of the entities and relations in val and test sets that are available within train set's gold SPARQLs.

**[Scenario 'A': Noise-Free Linking]** In this scenario, we simulate an entity/relation linker that has 100% $F_1$. For this, we pick all entities/relations from the gold SPARQL and pretend as if they were the output of the linker (see Figure 6 in appendix). We begin with extracting all the entities and relations from the gold SPARQL using their prefixes (*dbr* for entities and *dbp* or *dbo* for relations). Next, we pick these entities and relations, and align the same with *surface-form mention text* in the given question. We observe that entities match exactly with substrings in the questions most of the time (e.g. *Austin College* in Figure 6 of the appendix). For relations, an exact match is not always possible, e.g., a given relation `dbo:film` is semantically best aligned to word *movies* in the question. We use pre-trained fastext embeddings (Bojanowski et al., 2017) to represent words and relation and compute cosine similarity between each word in the question and the given relation. The highest-scoring word is considered as the aligned word. After identifying mentions of entities/relations, we mask them in question text as well as the corresponding gold SPARQL. This masked pair is subsequently fed to the seq2seq module as a training example.

**[Scenario 'B': Partly Noisy Linking]** Purpose of scenario 'B' is to allow partial noise in the entity/relation linking process. For this, we first feed the natural language question into an external entity/relation linker. The linker returns two things: (i) A set of surface form mentions for entities/relations in the question text, and (ii) Linked entities/relations for these mentions. We take linker's output and find intersection of these entities/relations with the entities/relations present in the gold SPARQL. These common entities/relations are masked in the SPARQL query. Also, their corresponding surface forms are masked in the question text. In order to mask the surface form in the question, we use exact match and string overlap based *Jaccard similarity*. Figure 7 in ap-



```
SELECT DISTINCT ?x WHERE
{ ?x dbp:notableInstruments
  dbr:Gibson_Guitar_corporation}
```

```
SELECT DISTINCT ?x WHERE
{
  ?x dbo:starring
  dbr:Gibson_Guitar_corporation
}
```

Figure 2: A detailed architecture of Stage-I.

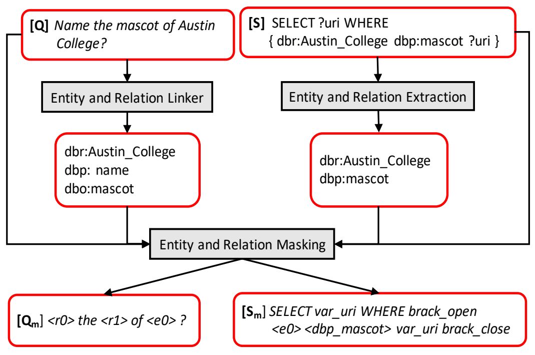

[Q] Name the mascot of Austin College?
[S] SELECT ?uri WHERE { dbr:Austin_College dbp:mascot ?uri }
Entity and Relation Linker
Entity and Relation Extraction
dbr:Austin_College
dbp: name
dbo:mascot
dbr:Austin_College
dbp:mascot
Entity and Relation Masking
[Q$_m$] <r0> the <r1> of <e0> ?
[S$_m$] SELECT var_uri WHERE brack_open <e0> <dbp_mascot> var_uri brack_close

Figure 3: An illustrative example for Scenario 'C'.

pendix illustrates this scenario.

**[Scenario 'C': Fully Noisy Linking]** Goal here is to simulate a completely realistic scenario where we rely entirely on an external entity/relation linker. For this, we feed input question to the entity/relation linker and get the suggested surface form mentions and linked entities/relations. We mask these suggested mentions using exact match and partial match. Corresponding SPARQL query's entities/relations are also masked based on the suggestions. This scenario is depicted in Figure 3.

### 4.2 Convolutional `Seq2Seq` Model

The pair of masked question and SPARQL query obtained from the noise simulator, under any noise scenario, is fed to a *Convolutional Neural Network (CNN)* based *seq2seq* model (Gehring et al., 2017). As shown in Figure 4, this model reads the entire masked question and then predicts the corresponding masked SPARQL query token-by-token in a left-to-right manner. This seq2seq model consists of the following key components.

**[Input Embedding Layer]** Both encoder and decoder consist of an embedding layer that maps each input token to a point-wise summation of its word embedding and positional embedding. The embedding of each word is initialized randomly. In order to capture the sense of order, the model is provisioned with the positional embedding.

**[Convolution + Pooling Layers]** The token embeddings obtained from the previous layer are fed to the multiple convolution and pooling layers. Each convolution layer consists of a 1-dimensional convolution followed by Gated Linear Units (GLU) (Dauphin et al., 2017). Residual connections (He et al., 2016) are added from input to the output of each convolution layer.

**[Multi-Step Attention]** Each decoder layer comprises a convolution layer followed by a multi-step attention layer. This multi-step attention is used to find the attention scores from a particular decoder state to the source tokens. Attention between decoder state $d_i$ (after $i^{th}$ layer) of the last token in generated sequence so far and state $z_j$ of the $j^{th}$ source element (after last encoder layer) is computed as: $a^i_j = exp(d_i \cdot z_j)/\sum_{t=1}^{m} exp(d_i \cdot z_t)$ where, $m$ is the number of source elements. The context vector, $c_i$, is now computed as, $c_i = [\sum_{j=1}^{m} a^i_j(z_j + e_j)] + d_i$ where, $e_j$ is the input embedding for the source element $j$.

**[Output Layer]** Finally, output at a particular time step is calculated over all the $Z$ possible tokens, $P(z_{t+1}|z_1, \ldots, z_t, X) = softmax(Wd_L + b)$ where $P(z_{t+1}|\cdot) \in \mathbb{R}^Z$, and $W$, $b$ are trainable parameters. $d_L$ is the decoder state of last target element at the last layer $L$. $X$ is the input sequence.

**[Training Loss:]** The model is trained using *label smoothed cross-entropy loss* given by following expression (for single training example) $L(\theta) = -(1/N) \cdot \sum_{n=1}^{N} \sum_{z=1}^{Z} q(y_n = z|y_{n-1}) \cdot \log P_\theta(y_n = z|y_{n-1})$ where, $N$ is the number of words in output sequence and $y_n$ is the first $n$ tokens of output sequence. $P_\theta(y_n = z|y_{n-1})$ is model's probability to output token $z$ given $y_{n-1}$ sequence generated so far. The quantity $q(y_n = z|y_{n-1})$ is equal to $\gamma$ if $f(y_n) = z$ and $(1-\gamma)/(Z-1)$ o/w, where $\gamma \in [0, 1]$, $\gamma > 1/Z$.

## 5 Stage-II: Neural Graph Search Module

Our error analysis on output of Stage-I revealed that entity linking performance is reasonably good but the same is not true for relation linking. Exist-

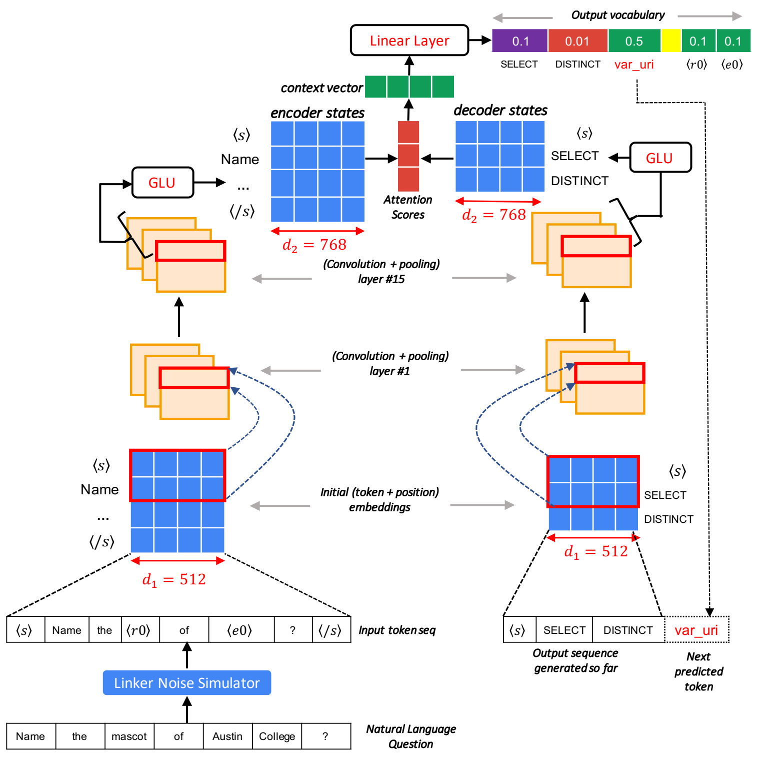

Output vocabulary
Linear Layer
0.1 0.01 0.5 0.1 0.1
SELECT DISTINCT var_uri ⟨r0⟩ ⟨e0⟩
context vector
encoder states
decoder states
⟨s⟩ Name … ⟨/s⟩
⟨s⟩ SELECT DISTINCT
GLU
GLU
Attention Scores
$d_2 = 768$
$d_2 = 768$
(Convolution + pooling) layer #15
(Convolution + pooling) layer #1
Initial (token + position) embeddings
$d_1 = 512$
$d_1 = 512$
⟨s⟩ Name the ⟨r0⟩ of ⟨e0⟩ ? ⟨/s⟩
Input token seq
⟨s⟩ SELECT DISTINCT var_uri
Output sequence generated so far
Next predicted token
Linker Noise Simulator
Name the mascot of Austin College ?
Natural Language Question

Figure 4: A CNN-based Seq2Seq model for KGQA. We have assumed noise-free linking scenario here.

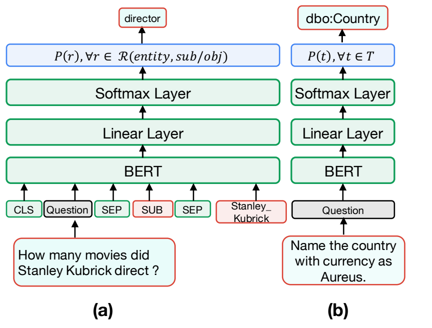

director
$P(r), \forall r \in \mathcal{R}(entity, sub/obj)$
Softmax Layer
Linear Layer
BERT
CLS Question SEP SUB SEP Stanley_Kubrick
How many movies did Stanley Kubrick direct ?
(a)
dbo:Country
$P(t), \forall t \in T$
Softmax Layer
Linear Layer
BERT
Question
Name the country with currency as Aureus.
(b)

Figure 5: Architecture of neural graph search module. (a) Relation Classifier. This module predicts relation for a given entity (b) Ontology Type Classifier. This module predicts *rdf:type* ontology class.

ing literature (Wu et al., 2020; Li et al., 2020) also show enough evidences of achieving high performance on the entity linking task, whereas relation linking turns out to be harder due to complexity of natural language. Because of this, we have most of the entities within a SPARQL silhouette generated by Stage-I as correct but the relations are incorrect. Graph search module in Stage-II takes a SPARQL silhouette as input and produces an improved version of the same by replacing incorrect relations (see Figure 8 in appendix for an example).[1] This is a BERT-based module and its architecture is shown in Figure 5. This module works as follows.

1. We consider each triple $\langle e_s, r, e_o \rangle$ in the SPARQL silhouette in which at least one of the entity is an existential variable unless the silhouette is with *rdf:type* relation which we handle separately. We prepare input in the following format: [CLS] $Q$ [SEP] [SUB (or OBJ)] [SEP] $e_s$ (or $e_o$). Here, $Q$ is token sequence of input question text and [SUB (or OBJ)] is special token depending on whether the grounded entity is in subject (or object) position (refer Figure 5a). We also pass grounded entity ($e_s$ or $e_o$) as the last element of this input. [CLS] and [SEP] are special tokens from BERT vocabulary.

2. We feed above input sequence of tokens into the BERT layer of graph search module. The output is passed through a linear layer followed by a *softmax* layer. This softmax layer induces a probability score $p_r$ for each relation $r \in \mathcal{R}$ in the given KG. While training, we use the following loss function (given for single example): $\ell = (1-\alpha)*(\ell_c)+(\alpha)*(\ell_{gs})$. Here, $\ell_c$ denotes standard cross entropy loss between predicted probabilities $\{p_r\}_{r\in\mathcal{R}}$ and the gold relation. The graph search loss term $\ell_{gs}$ forces the predicted probabilities to be low for all those relations which are invalid relations (in the given KG) for corresponding input entity $e_s$ (or $e_o$) in the input position (subject or object). For this, we assume a uniform probability distribution over all such valid relations and compute its cross entropy loss with $\{p_r\}_{r\in\mathcal{R}}$. $\alpha$ is a hyperparameter.

3. During inference, at softmax layer, we restrict the outputs only to those relations $r \in \mathcal{R}$ which are valid relation for the input entity as being subject or object. For example, if input grounded entity is $e_s$ then we restrict prediction to only those relations $r$ for which $\langle e_s, r, ?x \rangle$

[1]It is easy to extend this idea and perform an iterative graph searching when entity linker performance is also low.

is a valid triple for some grounding of $?x$. In DBpedia same relation can exist in the form of *'dbo'* and *'dbp'* for a specific entity. In such cases, we pick the *'dbo'* version. Prediction is made based out of 61623 relations available in DBpedia.

4. We have a separate version of the NGS module (refer Figure 5b) if the relation $r$ in a given triple is *rdf:type*. Note, in DBpedia, a triple containing *rdf:type* relation looks like this $\langle ?x, \textit{rdf:type}, \textit{dbo:type} \rangle$ where, $?x$ is a variable and *dbo:type* is the DBpedia ontology class of the entity $?x$. For such triples, input to NGS module is [CLS] Q. We need to predict the corresponding ontology class *dbo:type*. DBpedia ontology contains 761 classes and hence, in this model, prediction is one of these 761 classes. This module is trained with standard cross-entropy loss. An example of the *rdf:type* classification would be to predict *dbo:Country* for the question *'Name the country with currency as Aureus?'*.

## 6 Experiments and Results

**Datasets:** We work with two different KGQA datasets based on DBpedia: LC-QuAD-1 (Trivedi et al., 2017) and QALD-9 (Ngomo, 2018). LC-QuAD-1 contains 5000 examples and is based on the *04-2016 version* of the DBpedia. We split this dataset into 70% training, 10% validation, and 20% test sets (same as the leaderboard). QALD-9 is a multilingual dataset and is based on the *10-2016 version* of the DBpedia. Questions in this dataset vary in terms of reasoning nature (e.g. counting, temporal, superlative, comparative, etc.) and therefore, in terms of the SPARQL aggregation functions as well. This dataset contains 408 training and 150 test examples. We split the training set into 90% training and 10% validation sets.

**Evaluation Metric:** Performance is evaluated based on the standard precision, recall, $F_1$ score for KGQA systems. For more detail please refer to section C of appendix. We also measure a metric called Answer Match (AM).

**Answer Match (AM)**: For a question $Q$, when executing the predicted SPARQL in the underlying KG, if we get $S_p = S_g$ then we say AM=1 otherwise AM=0. Here, $S_g$ and $S_p$ are gold and predicted answer set respectively.

**Baselines:** We compare our approach with three baselines: WDAqua (Diefenbach et al., 2020), QAmp (Vakulenko et al., 2019) and gAnswer (Zou et al., 2014). WDAqua is a graph based approach to generate SPARQL query based on predefined patterns. These SPARQL candidates are then ranked. QAmp uses text similarity and graph structure based on an unsupervised message-passing algorithm. gAnswer is graph data driven approach and generate query graph to represent user intention. WDAqua and QAmp are top entries in the LC-QuAD-1 leaderboard [2] whereas, WDAqua and gAnswer in QALD-9 challenge (Ngomo, 2018).

**Experimental Setup:**
*1) Stage-I:* We use *Falcon* (Sakor et al., 2019) for entity/relation linking and experiment with all 3 noise scenarios. We use *fairseq*[3] library for implementation of CNN-based seq2seq model (Gehring et al., 2017) comprising of 15 layers[4]. and used Nesterov Accelerated Gradient (NAG) optimizer. We experimented with different values of hyperparameters and report results for the values yielding the best performance on the validation set. Details about tuning ranges and optimal values of all these hyperparameters are given in Table 4 of appendix. We used 2 Tesla v100 GPUs for training seq2seq model.
*2) Stage-II:* For neural graph search module, we work with a pre-trained *BERT-base uncased model*. Figure 9 of appendix captures change in validation accuracy with hyperparameter $\alpha$. It consists of 12 transformer layers, 12 self-attention heads, and 768 hidden dimension. We used 1 Tesla v100 GPU for training.

**Results and Discussions:** Table 2 and 3 capture performance of our approach compared to state-of-the-art on the LC-QuAD-1 and QALD-9 datasets. Our results in the last two rows of both the tables in stage-II are under realistic scenario or full noise setting for entity/relation linking. Our approach achieves state-of-the-art performance in case of LC-QuAD-1 dataset by improving an absolute margin of 3.72% F1. Our seq2seq model can achieve upto 83.08% $F_1$ for LC-QuAD-1 and 55.3% Macro $F_1$ QALD for QALD-9 dataset if the entity/relation linker were to be 100% correct. The gap between the performance of *No Noise* linking (upper bound) and *Full Noise* linking (lower bound) illustrates how the performance of entity/relation linker impacts the overall performance of KGQA. Poor per-

[2]http://lc-quad.sda.tech/lcquad1.0.html
[3]https://github.com/pytorch/fairseq
[4]We will release our code after the review period.

| Model Type | Model Name | AM | Prec. | Recall | $F_1$ |
|---|---|---|---|---|---|
| Baseline | WDAqua | - | 22.00 | 38.00 | 28.00 |
| | QAmp | - | 25.00 | **50.00** | 33.33 |
| Stage-I (Ours) | No Noise | 82.88 | 83.11 | 83.04 | 83.08 |
| | Part Noise | 41.34 | 42.40 | 42.26 | 42.33 |
| | Full Noise | 24.92 | **25.54** | 25.64 | 25.59 |
| Stage-II (Ours) | w/o type | 30.63 | **32.17** | 32.20 | 32.18 |
| | w/ type | 34.83 | **37.03** | 37.06 | **37.05** |

Table 2: Test set performance on LC-QuAD-1 dataset.

| Model Type | Model Name | AM | Mac. Prec. | Mac. Rec. | Mac. $F_1$ | Mac. $F_1$ QALD |
|---|---|---|---|---|---|---|
| Baseline | WDAqua | - | 26.1 | 26.7 | 25.0 | 28.9 |
| | gAnswer | - | 29.3 | **32.7** | 29.8 | **43.0** |
| Stage-I (Ours) | No Noise | 29.9 | 80.4 | 42.1 | 40.9 | 55.3 |
| | Part Noise | 13.1 | 63.9 | 28.7 | 22.4 | 39.6 |
| | Full Noise | 11.1 | **82.6** | 23.0 | 20.6 | 36.0 |
| Stage-II (Ours) | w/o type | 15.3 | **59.4** | 26.1 | 23.3 | 36.2 |
| | w/ type | 15.3 | **59.4** | 26.1 | 23.3 | 36.2 |

Table 3: Test set performance on `QALD-9` dataset. Here Mac. means Macro and Rec. means Recall.

formance of Falcon on relation linking (Table 5 of appendix) also justifies this gap. Further, the performance of Stage-II demonstrates we gain 11.46% in $F_1$ for LC-QuAD-1 and 0.2% in Macro $F_1$ QALD for QALD-9 dataset.

By manually analysing examples, we find that our method can learn complex patterns of queries including imperative, interrogative, questions with count (aggregation) and involving multiple relations. Proposed model is a simple neural approach which does not need question interpretation step unlike QAmp and also can learn complex templates without requiring a large number of templates unlike WDAqua. As our seq2seq model translates natural language query to SPARQL, it does not need to resolve any ambiguity in the natural language questions to reach to answer unlike approaches (gAnswer) that generates query graph . Another advantage of our method is it is KG agnostic method.

**Error Analysis:** We randomly picked 50 examples where predicted answer by our model is wrong. We see that, there are four types of scenarios where our model fails to generate correct SPARQL: (i) two very similar looking relations such as "placeOfDeath" (gold) Vs "deathPlace" (predicted), "product" (gold) Vs "products" (predicted) (details in Table 9 of appendix) that exist in the KG for an entity (ii) inconsistencies in KG (iii) gold SPARQL comprises infrequent SPARQL keywords (iv) classes of rdf:type belong to other than DBPedia classes. The performance numbers in the last two rows of Table 3 are same because in the dataset there are only two such gold examples with rdf:type classes with *YAGO* ontology that our model does not support. Further analysis shows that the reason for QALD-9 having low upper bound is its training set size being too small (408) and has large variety of SPARQL keywords within a small training set. There are only 32 queries with FILTER and 4 queries with GROUP BY keyword in the training set of 408 to represent comparative/superlative questions that is too small for any neural model to learn from. Training on more data can further improve the performance. Because of the inconsistencies in KG , presence of classes in *YAGO* ontology and infrequent SPARQL keywords, generated SPARQL silhouette in QALD-9 dataset has errors other than incorrect entity/relation. Therefore, Stage-II offers much smaller gain for QALD-9. Combining LC-QuAD-1 training data with QALD-9 (Table 7 and 8 of appendix)) did not improve the performance of QALD-9 dataset because the nature of the SPARQL is very different in both the datasets.

## 7 Conclusions

We proposed a simple sequential two-stage NMT-based approach to solve the KGQA task. Stage-I translates natural language query to SPARQL silhouette. To train seq2seq module, we introduced various noise scenarios with masking schemes to handle unseen entities/relations and reduce vocabulary size. We also introduce Neural Graph Search Module in Stage-II to improve the quality of SPARQL silhouette generated in the realistic scenario at Stage I. We demonstrated that, though in ideal linking scenario, our Stage-I can generate high quality SPARQL, in realistic scenario quality of SPARQL silhouette drops. Integrating NGS module with Stage-I, enhances the quality of generated final SPARQL thereby improving state-of-the-art performance for LC-QuAD-1 dataset. We believe, this research demonstrates great potential of NMT-based approaches to solve the KGQA task and opens up a new research direction.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proc. of COLING*, pages 2503–2514.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. of EMNLP*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proc. of ACL*, pages 1415–1425.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. of ACM SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. 2017. An encoder-decoder framework translating natural language to database queries. *arXiv preprint arXiv:1711.06061*.

Ruichu Cai, Boyan Xu, Zhenjie Zhang, Xiaoyan Yang, Zijian Li, and Zhihao Liang. 2018. An encoder-decoder framework translating natural language to database queries. In *Proc. of IJCAI*, pages 3977–3983.

Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. 2019. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*.

Yongrui Chen, Huiying Li, Yuncheng Hua, and Guilin Qi. 2020. Formal query building with query structure prediction for complex question answering over knowledge base. In *Proc. of IJCAI*, pages 3751–3758.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proc. of ICML*, pages 933–941.

Dennis Diefenbach, Andreas Both, Kamal Deep Singh, and Pierre Maret. 2020. Towards a question answering system over the semantic web. *Semantic Web*, pages 421–439.

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

Jiwei Ding, Wei Hu, Qixin Xu, and Yuzhong Qu. 2019. Leveraging frequent query substructures to generate formal queries for complex question answering. In *Proc. of EMNLP*, pages 2614–2622.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proc. of ACL*, pages 33–43.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proc. of ACL*, pages 260–269.

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 *Proc. of ISWC*, pages 69–78.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proc. of ICML*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778.

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*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proc. of IJCAI*, pages 4483–4491. Survey Track.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proc. of ACL*, pages 969–974.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia – A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Belinda Z. Li, Sewon Min, Srini Iyer, Yashar Mehdad, and Wen tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proc. of EMNLP*.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv:1607.06275*.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. 2021. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(1):1–23.

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 *Proc. of ISWC*, pages 487–504.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proc. of AAAI*.

Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (QALD-9). *Language*, 7(1).

Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. 2020. YAGO 4: A reason-able knowledge base. In *Proc. of ESWC*, pages 583–596.

Michael Petrochuk and Luke Zettlemoyer. 2018. SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach. In *Proc. of EMNLP*, pages 554–558.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Ahmad Sakor, Kuldeep Singh, and M E Vidal. 2019. Falcon: An entity and relation linking framework over DBpedia. In *Proc. of CEUR Workshop*, volume 2456, pages 265–268.

Kuldeep Singh, Andreas Both, Arun Sethupat, and Saeedeh Shekarpour. 2018. Frankenstein: A platform enabling reuse of question answering components. In *Proc. of ESWC*, pages 624–638. Springer.

Thomas Steiner, Ruben Verborgh, Raphaël Troncy, Joaquim Gabarro, and Rik Van de Walle. 2012. Adding realtime coverage to the google knowledge graph. In *Proc. of ISWC*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proc. of EMNLP*, pages 562–572.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *Proc. of ISWC*, 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.

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 *Proc. of CIKM*, pages 1431–1440.

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

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proc. of ACL*, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proc. of ACL*, pages 1321–1331.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proc. of ACL*, pages 440–450.

Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. 2021. Neural machine translating from natural language to sparql. *Future Generation Computer Systems*, 117:510–519.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proc. of NAACL-HLT*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proc. of EMNLP*, pages 678–687.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proc. of UAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv:1709.00103*.

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 *Proc. of ACM SIGMOD*, pages 313–324.

## A Noise Simulator and Masking

Details about all the masking schemes and noise scenarios are given in the main paper. Here we explain scenario ’A’ and scenario ’B’ with the examples in Figure 6 and Figure 7 respectively.

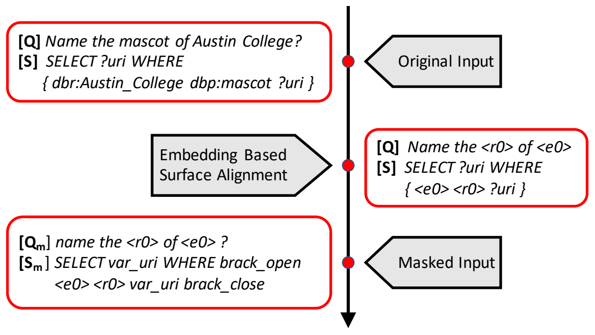

[Q] Name the mascot of Austin College?
[S] SELECT ?uri WHERE { dbr:Austin_College dbp:mascot ?uri }
Original Input
Embedding Based Surface Alignment
[Q] Name the <r0> of <e0>
[S] SELECT ?uri WHERE { <e0> <r0> ?uri }
[Q$_m$] name the <r0> of <e0> ?
[S$_m$] SELECT var_uri WHERE brack_open <e0> <r0> var_uri brack_close
Masked Input

Figure 6: An illustrative example for Scenario 'A': Noise-Free Linking. To align the surface forms of the entities/relations mentions in the given question text, we used word embedding as it offers higher alignment $F_1$. We used Falcon as a linker.

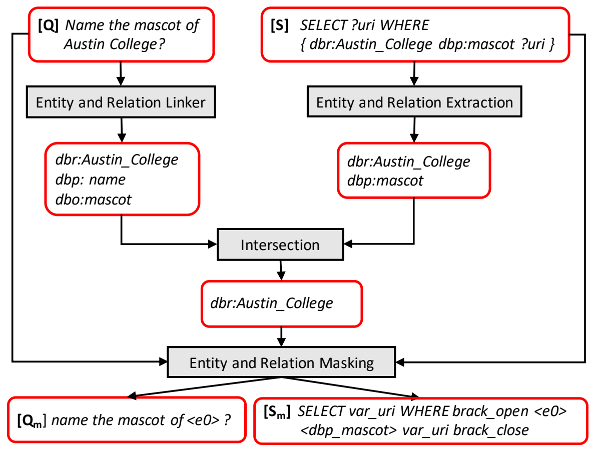

[Q] Name the mascot of Austin College?
[S] SELECT ?uri WHERE { dbr:Austin_College dbp:mascot ?uri }
Entity and Relation Linker
Entity and Relation Extraction
dbr:Austin_College
dbp: name
dbo:mascot
dbr:Austin_College
dbp:mascot
Intersection
dbr:Austin_College
Entity and Relation Masking
[Q$_m$] name the mascot of <e0> ?
[S$_m$] SELECT var_uri WHERE brack_open <e0> <dbp_mascot> var_uri brack_close

Figure 7: An illustrative example for Scenario 'B': Partly Noisy Linking. To align the surface forms of the entities/relations mentions in the given question text, we used exact match as well as string overlap based *Jaccard similarity* with a threshold of 0.7. We used Falcon as a linker.

## B Neural Graph Search Module

Figure 8 shows an example of input to the neural graph search module in Stage-II.

## C Evaluation Metric

*1) Precision, Recall, and $F_1$ for Single Question:* For single question $Q$, we compute precision $P$, recall $R$, and $F_1$ using the set of *gold answer entities* $S_g$ and *predicted answer entities* $S_p$. While computing these metrics, we handle boundary cases as follows. If $S_g = S_p = \emptyset$ then we take $P = R = F_1 = 1$. If only $S_g = \emptyset$ then we take $R = F_1 = 0$.
*2) Macro Precision, Macro Recall, Macro $F_1$, and Macro $F_1$ QALD:* These metrics are defined for the whole dataset. For this, we first compute $P$, $R$, and $F_1$ at individual question level and average of these numbers across entire dataset gives us the *macro* version of these metrics. For $F_1$, if use the boundary condition of having $P = 1$ when $S_p = \emptyset, S_g \neq \emptyset$ then such a Macro $F_1$ is called as *Macro $F_1$ QALD* as per Ngomo (2018). But if we instead use $P = 0$ then it is called Macro $F_1$.
*3) Precision, Recall, and $F_1$ for the whole set:* For whole set, $P$ and $R$ are same as *macro* version of these metrics. $F_1$, however, is computed by taking Harmonic mean of these $P$ and $R$. The reported metrics for the LC-QuAD-1 dataset were computed in this manner.

## D Hyperparameters Tuning

Table 4 shows the range of hyperparameters used for our experiments and the best value when tuned on validation set. Figure 9 captures change in validation accuracy with hyperparameter $\alpha$ used in the loss function in stage II.

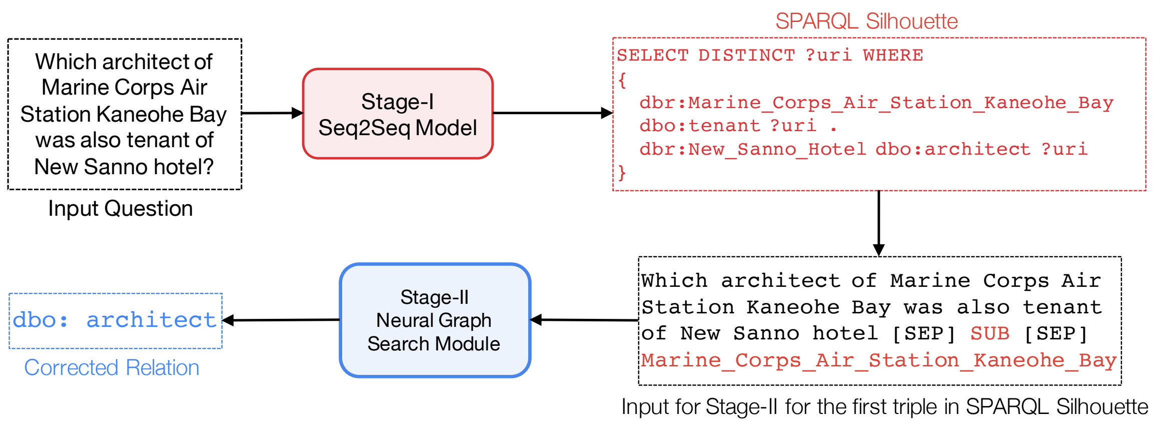
## E Experiments and Results

Apart from the experiments in the main paper, we also performed experiments taking transformer as the seq2seq model in the stage-I. Table 6 captures performance of transformer model. From this table and our performance table in main paper (Table 2 and Table 3 in main paper), one can notice that, performance of CNN model is better than the transformer model. So, we opted CNN as our model.

Table 7 and 8 capture test performance of stage-I module when we train our CNN model with combining the training data of both the datasets for LC-QuAD-1 and QALD-9 respectively. From these two tables and the main performance tables (Table 2 and Table 3 in main paper), we can see that, performance of Stage-I is better when the model is trained in same domain dataset rather than when it is trained with data combining both the datasets.

## F Anecdotal Examples

Table 9 shows examples from LC-QuAD-1 test set where our neural graph search module is unable to disambiguate between two very similar looking (placeOfDeath Vs deathPlace, mouthPlace Vs sourceRegion, product Vs products) relations that exist in DBpedia for an entity. As an example for the first question in Table 9, the entity *Essex* has both the relations mouthPlace and mouthRegion present in underlying KG DBpedia. So, it is very


Which architect of Marine Corps Air Station Kaneohe Bay was also tenant of New Sanno hotel?
Input Question
Stage-I Seq2Seq Model
SPARQL Silhouette
Stage-II Neural Graph Search Module
dbo: architect
Corrected Relation
Input for Stage-II for the first triple in SPARQL Silhouette

```
SELECT DISTINCT ?uri WHERE
{
  dbr:Marine_Corps_Air_Station_Kaneohe_Bay
  dbo:tenant ?uri .
  dbr:New_Sanno_Hotel dbo:architect ?uri
}
```

```
Which architect of Marine Corps Air
Station Kaneohe Bay was also tenant
of New Sanno hotel [SEP] SUB [SEP]
Marine_Corps_Air_Station_Kaneohe_Bay
```

Figure 8: An example of input to the *neural graph search module*.

| Hyperparameter | Tuning Range | Best Value |
|---|---|---|
| $\eta$ for Stage-I | $[1 \times 10^{-1}, 2 \times 10^{-1}, 2.5 \times 10^{-1}, 5.0 \times 10^{-1}]$ | 0.25 |
| $\eta$ for Stage-II | $[10^{-4}, 10^{-5}, 10^{-6}]$ | $10^{-5}$ |
| $b$ for both stages | 8 | 8 |
| $\alpha$ for LC-QuAD-1 | $[1 \times 10^{-1}, 4 \times 10^{-1}, 6 \times 10^{-1}, 7 \times 10^{-1}]$ | $4 \times 10^{-1}$ |
| $\alpha$ for QALD-9 | $[1 \times 10^{-1}, 4 \times 10^{-1}, 6 \times 10^{-1}, 7 \times 10^{-1}]$ | $6 \times 10^{-1}$ |

Table 4: Tuning range and the final chosen best values of various hyperparameters. $\eta$ means learning rate and $b$ means batch size. $\alpha$ is hyperparameter in loss function of Graph Search Module in Stage-II.

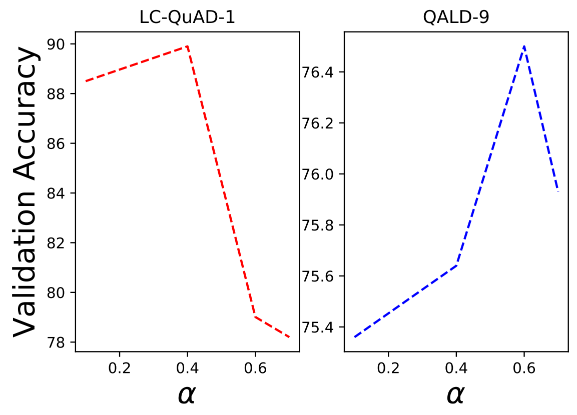


Figure 9: Change in validation set accuracy with hyper-parameter $\alpha$

difficult for any seq2seq model to disambiguate these two relations that are semantically same and looks very similar.

| Dataset | E/R | Precision (%) | Recall (%) |
|---|---|---|---|
| LC-QuAD-1 | E | 79.19 | 85.60 |
| | R | 43.74 | 44.99 |
| QALD-9 | E | 78.00 | 98.55 |
| | R | 41.05 | 37.17 |

Table 5: *Falcon* performance on entity (E) and relation (R) linking on test sets.

| Model Type | Model Name | AM | Prec. | Recall | $F_1$ |
|---|---|---|---|---|---|
| Stage-I (Transformer) | No Noise | 77.58 | 77.82 | 77.78 | 77.79 |
| | Part Noise | 34.73 | 35.87 | 35.92 | 35.89 |
| | Full Noise | 20.72 | 21.49 | 21.43 | 21.46 |

Table 6: Test set performance on LC-QuAD-1 dataset when Transformer is used in Stage- I.

| Model Type | Model Name | AM | Prec. | Recall | $F_1$ |
|---|---|---|---|---|---|
| Stage-I | No Noise | 80.78 | 81.03 | 81.09 | 81.06 |
| | Part Noise | 39.24 | 40.14 | 40.39 | 40.26 |
| | Full Noise | 23.92 | 24.70 | 24.91 | 24.80 |

Table 7: Test set performance on LC-QuAD-1 dataset when CNN is trained on training set of combining LC-QuAD-1 and QALD-9 in Stage I.

| Model Type | Model Name | AM | Prec. | Recall | $F_1$ |
|---|---|---|---|---|---|
| Stage-I | No Noise | 30.66 | 79.0 | 42.9 | 39.5 |
| | Part Noise | 18.24 | 72.6 | 31.8 | 44.3 |
| | Full Noise | 07.29 | 72.5 | 18.6 | 29.6 |

Table 8: Test set performance on QALD-9 dataset when CNN is trained on training set of combining LC-QuAD-1 and QALD-9 in Stage I.

| Question | Gold Relation | Predicted Relation |
|---|---|---|
| Name the rivers who originate from **Essex**? | mouthPlace | sourceRegion |
| Where was the person born who died in **Bryn Mawr Hospital**? | placeOfDeath | deathPlace |
| Name the artist who made Dream Dancing and is often associated with **Joe Pass**. | associatedBand | associatedMusicalArtist |
| What is used as money for French Southern and Antarctic Lands is also the product of the **Karafarin Bank** ? | product | products |

Table 9: Anecdotal examples from LC-QuAD-1 test set where graph search module is unable to disambiguate between two closely related relations (gold and predicted) that are available for the highlighted entities in DBpedia.