Difficulty-controllable Multi-hop Question Generation From Knowledge Graphs

Vishwajeet Kumar[1,3,4], Yuncheng Hua[2,4], Ganesh Ramakrishnan[3], Guilin Qi[2,6,7], Lianli Gao[5], and Yuan-Fang Li[0000−0003−4651−2821][4⋆]

[1] IITB-Monash Research Academy, Mumbai, India
[2] School of Computer Science and Engineering, Southeast University, Nanjing, China
[3] IIT Bombay, Mumbai, India
[4] Monash University, Melbourne, Australia
[5] The University of Electronic Science and Technology of China, China
[6] Key Laboratory of Computer Network and Information Integration (Southeast University) Ministry of Education, Nanjing, China
[7] Key Laboratory of Rich-media Knowledge Organization and Service of Digital Publishing Content, SAPPRFT, Beijing, China

Abstract. Knowledge graphs have become ubiquitous data sources and their utility has been amplified by the research on ability to answer carefully crafted questions over knowledge graphs. We investigate the problem of question generation (QG) over knowledge graphs wherein, the level of difficulty of the question can be controlled. We present an end-to-end neural network-based method for automatic generation of complex multi-hop questions over knowledge graphs. Taking a subgraph and an answer as input, our transformer-based model generates a natural language question. Our model incorporates difficulty estimation based on named entity popularity, and makes use of this estimation to generate difficulty-controllable questions. We evaluate our model on two recent multi-hop QA datasets. Our evaluation shows that our model is able to generate high-quality, fluent and relevant questions. We have released our curated QG dataset and code at `https://github.com/liyuanfang/mhqg`.

Keywords: Question generation · Knowledge graph · Natural language processing · Transformer · Neural network

1 Introduction

Knowledge graphs (KG) have quickly become an indispensable information source for both research and practice in recent years. A great amount of effort has been invested into curating large KGs such as Freebase [3], DBPedia [2] and Wikidata [11]. Question answering (QA) [19,8], the task of answering natural-language questions over a KG, has attracted substantial research interest as it is an accessible, natural way of retrieving information from KGs without the need for learning complex structural query languages such as SPARQL.

⋆ Author for correspondence: `yuanfang.li@monash.edu`.

State-of-the-art KG QA models are typically based on neural networks and as a result, they are data-driven and need large amounts of training data, containing a set of triples in the form of a graph, a question and the corresponding answer. To cater to the need of training and evaluating KG QA models, a number of datasets have been created and curated over the years. These datasets include those that contain simple, single-hop information [1,4,24] as well as those that contain more complex information. These complex datasets either comprise multi-hop instances [34,35,30] or instances that are answerable only through discrete reasoning [23].
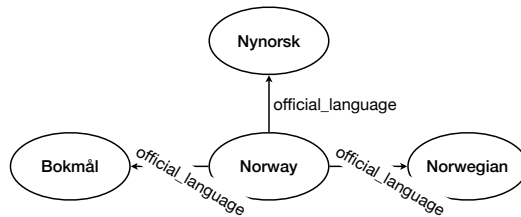
However, further improvements in KG QA have been hindered by the limited availability of data. The abundance of large-scale "simple", single-triple data does not necessarily help advance state-of-the-art. This is because, questions on such data are *easy* to answer, once correct entities and predicates are identified. In contrast, complex questions whose answering entails inference across multiple triples are naturally more difficult to answer and are therefore more valuable resources for improving KG QA models. However, complex questions are also more difficult to create, and most existing complex question datasets are created either manually or in a semi-automated manner.

Question generation (QG) over knowledge graphs poses a number of challenges. To illustrate the challenges, in Example 1 we present two subgraphs (both in visual and textual form) and the corresponding reference questions and answers from the COMPLEXWEBQUESTIONS [28] and the PathQuestion [35] datasets. The first example is from PathQuestion, which has the entities and predicates separated by a special token #, the end of the subgraph denoted by the token <end>, followed by the answer entity. It contains three entities connected by two predicates. The second example is from ComplexWebQuestions, which has the triples separated by the special token <t>. The example is a three-hop subgraph. Different from the previous example, this subgraph is not a sequence of triples but rather star-shaped, and has multiple answers.

Example 1. Two examples, each consisting of a subgraph, a question about it, together with the answer.

G: henry_i_duke_of_guise#parents#anna_deste#spouse# jacques_de_savoie_2nd_duc_de_nemours#<end># jacques_de_savoie_2nd_duc_de_nemours
Q: what is the name of the spouse of henry_i_duke_of_guise 's mom?
A: jacques_de_savoie_2nd_duc_de_nemours

| **G:** `Norway official_language Bokmål <t>` |
| `Norway official_language Norwegian <t>` |
| `Norway official_language Nynorsk` |
| **Q:** `what languages are spoken in norway?` |
| **A:** `Bokmål, Norwegian, Nynorsk` |

We address the following challenges for QG over KGs. Firstly, the input is a graph, and not necessarily a sequence of tokens. The second graph in Example 1 contains one entity with three outgoing predicates connecting to three other entities, and is obviously not structured as a sequence. Conventional text generation methods are however based on sequence-to-sequence models such as recurrent neural networks (RNN), that assume the input to be a sequence. Such a mismatch may negatively affect the quality of QG. In the same vein, for complex, multi-hop questions, a model would need to look at different parts of the graph repeatedly to generate a syntactically fluent question. Again, this is difficult for RNN-based techniques that operate sequentially. Last but not least, it is desirable to be able to generate questions of varying difficulty levels.

In this paper, we address the important problem of automatic generation of *complex, multi-hop* questions over KGs. We propose an end-to-end, self-attentive QG method based on the Transformer [31] architecture. Our approach does not assume sequential representation of an input graph, and is naturally able to attend to different parts of a graph in an efficient manner. Moreover, we model and estimate the difficulty level of a given subgraph-question pair so that we can generate questions of varying difficulty levels.

To the best of our knowledge, this is the first work on automatic generation of complex, multi-hop questions from KGs. Our main contributions are fourfold.

1. We propose a novel model for generating complex, difficulty-controllable questions from subgraphs of multiple triples.
2. Our Transformer-based model naturally treats a subgraph (a set of triples) as a graph and avoids arbitrary *linearisation* into a sequence of triples.
3. Our evaluation over a state-of-the-art natural-language generation model on two multi-hop QA datasets shows our technique is able to generate questions of much higher quality.
4. Models, dataset and code is available[8] to facilitate reproduction and further research on KG QA research.

---

[8] `https://github.com/liyuanfang/mhqg`

2 Related work

In this section we briefly discuss prior work from the areas of question answering and question generation that are most related to our work.

2.1 Question Answering over Knowledge Graphs

Question Answering over Linked Data (QALD) [19] has been under intense investigation in recent years. A wide array of methods and techniques have been developed.Bordes *et. al.* [4] employed Memory Networks [33] to answer simple, one-hop questions, and created the SimpleQuestions dataset that contains 100k one-hop questions. Semantic parsing has also been investigated as an effective method for answering both simple [1] and complex [34] questions. Talmor and Berant proposed [28] to decompose a complex question into simpler ones as a way of answering it. Unlike in semantic parsing where the knowledge graph is used for QA, Talmor and Berant propose to answer simple questions by performing a Web search. The final answer is computed from the sequence of answers. The results are evaluated on their own ComplexWebQuestions dataset that includes SPARQL queries, answers and text snippets as evidence. As a byproduct of these methods, the datasets WebQuestions [1] and WebQuestionsSP [34] were created. An Interpretable Reasoning Network (IRN) was proposed by Zhou, Huang and Zhu [35] to answer multi-hop (path or conjunctive) questions. Two datasets, PathQuestion and WorldCup2014, of up to three hops were created for evaluating IRN.

A different type of complex questions answering task that involve discrete reasoning has recently been proposed in CSQA [23]. Unlike multi-hop questions, questions in CSQA requires a variety of different reasoning tasks, such as logical, quantitative, qualitative, and comparative reasoning.

Interested readers are referred to a recent survey [8] for further details.

2.2 Question Generation

Question generation (QG) has recently attracted significant interests in the natural language processing (NLP) and computer vision (CV) community. Given an input (e.g. a passage of text in NLP or an image in CV), optionally also an answer, the task of QG is to generate a natural-language question that is answerable from the input. Neural network-based methods [9,17,26,18,12] are the state-of-the-art in QG. These end-to-end models do not require the manual creation of templates or rules, and are able to generate high-quality, fluent questions.

For any data-driven tasks such as question answering, the availability of large, varied and challenging datasets is crucial to their continued improvements. In fact, the recent development and interests in QALD is in part driven by the creation and release of the public datasets discussed in the previous subsection. Significant manual work has been invested in the creation (e.g. by Amazon Mechanical Turk workers) and curation (e.g. by the researchers) of these datasets. Despite the

coninued efforts, constrained by available resources, these datasets are limited in their size and variability.

As a response to this issue, the (semi-)automatic generation of questions over KG has recently been investigated. Seyler at al [25] proposed a semi-automatic method of generating multi-hop quiz questions from KG. With an entity e as the starting point, the KG is queried to find all triples with the entity as either the subject or the object, using SPARQL with patterns $\langle e\ ?p\ ?o \rangle$ and $\langle ?s\ ?p\ o \rangle$. The SPARQL queries are then verbalised from a given pattern to generate quiz questions. The notion of difficulty, which is measured from entity popularity, triple pattern selectivity and coherence, is incorporated in this work.

Inspired by QALD [19], Large-Scale Complex Question Answering Dataset (LC-QuAD) [30], a QA dataset of 5,000 multi-hop questions, was recently released. Similar to the previous work, LC-QuAD's generation is semi-automatic. The starting point is a manually curated list of DBPedia entities and predicates, a list of SPARQL templates, as well as a list of natural-language question templates, one for each SPARQL template. Given a seed entity and predicates, a two-hop subgraphs are extracted from DBPedia. The subgraphs and templates are merged to create valid SPARQL queries, and in turn natural-language questions. These questions are eventually corrected and reviewed by human users.

Different from the above works, our method is end-to-end and fully automated without the need of manually created templates or patterns. Our method only requires a subgraph (similar to a text passage and an image in other settings) and optionally an answer, from which a natural-language question is generated.

The 30M Factoid Question Answer Corpus [24] is possibly the earliest work using neural networks to generate questions over KG, and the largest dataset of single-hop questions. With SimpleQuestions [4] as the training set, they employ a standard encoder-decoder architecture to embed facts (triples), from which questions are generated. Reddy et al [22] also uses a standard sequence-to-sequence model to generate single-hop questions from a set of keywords, extracted from a KG using rules. Elsahar et al recently proposed a method [10] of generating single-hop questions from KG. Their method supports the generation involving unseen predicates and types, which is achieved by incorporating side information, in this case textual context from Wikipedia articles. Employing the encoder-decoder architecture with GRUs (gated recurrent units), the decoder module makes use of triple attention and textural attention to generate the next, possibly unseen token.

Our method differs from the above in a number of important ways. (1) Our model generates complex multi-hop questions, whilst all of the above neural network-based methods generate single-hop questions. (2) Our end-to-end model estimates and controls difficulty levels of generated questions. (3) We employ the Transformer [31] as our base model. The Transformer architecture allows us to naturally treat a graph as a graph, insted of a sequence of triples. Moreover, compared to variants of recurrent neural networks (e.g. LSTM [15] and GRU [6]), training on the Transformer is more efficient.

More broadly speaking, question generation from KG is a special case of text generation from KG, which has also been investigated recently [20,29,32]. These techniques encode a set of triples using either customised LSTM or GCN (graph convolutional network), and are typically evaluated on the WebNLG dataset [13]. A main difference is that these work do not take into account the answer or the difficulty level, which are important in the task of QG. Moreover, compared to RNN-based methods, our technique is more effective and efficient in handling larger contexts and is able to attend to multiple places in the context.

3 Our Approach

We model the problem of question generation over knowledge graphs as a sequence-to-sequence (Seq2Seq) learning problem. We assume a background knowledge graph $\mathcal{G}$, comprising a set of triples (facts). Given a subgraph $G = \{f_1, \ldots, f_n\} \subseteq \mathcal{G}$ of n facts, a set of entities E_A that appears in some triple(s) in G that represents the *answer*, our model will generate a natural-language question $Q = (w_1, \ldots, w_m)$, i.e. a sequence of m words, such that

$$Q^* = \arg\max_Q P(Q \mid G, E_A; \Theta) \tag{1}$$

$$= \arg\max_{w_1, \ldots, w_m} \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}, G, E_A; \Theta) \tag{2}$$

where Θ denotes model parameters.

The high-level architecture of our model can be seen in Figure 1. It uses the Transformer as the base architecture. The encoder (Section 3.1) consists of a stack of Transformers, taking as input the subgraph, the answer entities and estimated difficulty level of the subgraph. Difficulty modelling and estimation is described in detail in Section 3.2. The decoder (Section 3.3) is another stack of Transformers and decodes a multi-hop question given the encoder output, conditioned on the user-specified difficulty setting.

3.1 Knowledge Graph Encoding

For a subgraph G, the encoder takes its embedding as input, which in turn is constructed from the embeddings of the triples in G. Let d_e denote the dimension of entity/relation embeddings and d_g denote the dimension of triple embeddings. At initialisation, the embedding of a triple is the concatenation of the embeddings of the subject, the predicate and the object of the triple, with the rest of the values randomly initialised to match triple embedding dimension d_g. Each answer entity in E_A is additionally embedded into a d_e-dimensional vector, learned from whether it is an answer entity through an MLP (multi-layer perceptron). Element-wise addition is then performed on the answer embedding and the original entity embedding to obtain the final emmedding for each answer entity.
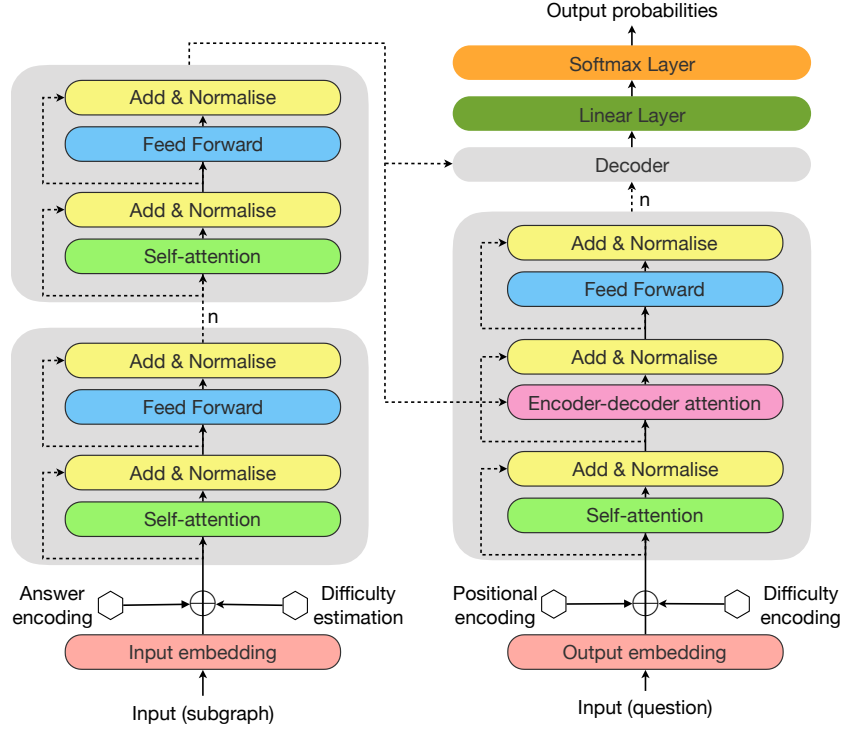
Fig. 1: The high-level architecture of our multi-hop question generation framework. The encoder stack of Transformers is on the left and the decoder stack is on the right.

Therefore, for graph G containing n triples, it is represented as a matrix $\boldsymbol{G} \in \mathbb{R}^{n \times d_g}$. Taking $\boldsymbol{G}$ as input, the Transformer encoder maps it to a sequence of continuous representations $\boldsymbol{Z} = (z_1, \cdots, z_n) \in \mathbb{R}^{n \times d_v}$.

Let $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{n \times d_k}, \boldsymbol{V} \in \mathbb{R}^{n \times d_v}$ denote the *query*, the *key* and the *value* matrices of the encoder Transformer. Given input $\boldsymbol{G}$, we use the query matrix $\boldsymbol{V}$ to soft select the relevant triples with the scaled dot-product attention:

$$\text{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{3}$$

where $\boldsymbol{K}^T$ is $\boldsymbol{K}$'s transpose, $\boldsymbol{Q} = \boldsymbol{G}\boldsymbol{W}^Q, \boldsymbol{K} = \boldsymbol{G}\boldsymbol{W}^K, \boldsymbol{V} = \boldsymbol{G}\boldsymbol{W}^V$, and $\boldsymbol{W}^Q, \boldsymbol{W}^K \in \mathbb{R}^{d_g \times d_k}, \boldsymbol{W}^V \in \mathbb{R}^{d_g \times d_v}$ are trainable model parameters.

To be able to attend to information from different triples in different representation subspaces, we use the multi-head attention with k heads and aggregate them as follows:

$$\text{MultiAtt}(\boldsymbol{Q}, \boldsymbol{G}) = \text{concat}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k)\boldsymbol{W}^O \tag{4}$$

where $\boldsymbol{W}^O \in \mathbb{R}^{kd_v \times d_g}$, and $\boldsymbol{a}_i = \text{Att}(\boldsymbol{QW}_i^Q, \boldsymbol{GW}_i^K, \boldsymbol{GW}_i^V), i \in [1, k]$ as defined in Formula 3.

Aggregated multi-head attention output is passed to a feed-forward neural net (FFNN) in each of the encoder stack of Transformers, where the output $\boldsymbol{x}$ goes through two linear transformations:

$$\text{FFNN}(\boldsymbol{x}) = \max(0, \boldsymbol{xW}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{5}$$

In all the encoder layers other than 1st layer we directly feed output of the previous encoder layer as input. The output of the top of the encoder is finally transformed into two attention matrices, the keys $\boldsymbol{K}_{encdec}$ and the values $\boldsymbol{V}_{encdec}$.

The original Transformer is designed for handling sequences, and for that purpose it provides *positional encoding* to encode the position of each input token. As a subgraph does not necessarily form a sequence of triples (cf the second graph in Example 1), we do not use positional encoding in our encoder.

The subgraph embedding is augmented with the answering (entity) encoding, as well as a difficulty estimation, which is described in the next subsection.

3.2 Difficulty Level Modelling & Estimation

Given a subgraph, it is desirable to generate questions of different difficulty levels in different situations. As there is no ground truth for question difficulty level, we estimate the difficulty using characteristics of the question and the subgraph, namely (1) the *confidence* of entity linking in the question, and (2) the *selectivity* of the surface forms of entities in the subgraph.

Confidence. We employ NER (named entity recognition) systems to perform entity recognition and linking. Intuitively, high confidence of an NER system about an entity-mention linking may be due to the low ambiguity of the mention and the high differentiability of the context, both of which would make the subgraph easy to understand and the generated question easy to answer. For example, given the mention "`Cold War`", if the NER system returns a higher confidence score for the entity freebase:m.034y4w than freebase:m.011l309l, a question that contains the former would be more likely to be correct, hence easier to answer.

Selectivity. On the other hand, the less selective a mention is (e.g. "`John Smith`" vs "`Elon Musk`") , the more confusing it would be for the question containing that mention. We query Wikidpedia with the each mention, and use the number of returned hits as an estimation of its *selectivity*, where he higher the number of hits, the lower the selectivity, thus the more difficult the question.

Given a training instance (G, q) of a subgraph G and a question q, we denote the confidence of $Con(q)$, by averaging over all identified mentions in q and min-max normlisation over the training corpus. We denote the selectivity of the subgraph $Sel(G)$, by averaging over all entities in G and min-max normlisation. We estimate the difficulty level of a given subgraph and a question, $Dif(G, q)$,

as follows:

$$\text{Dif}(G, q) = \frac{1 + Sel(G)}{1 + Con(q)} \tag{6}$$

The difficulty estimation $\text{Dif}(G, q)$ is then normalised into the closed interval $[0, 1]$, and finally converted into a binary vector $\boldsymbol{x} \in \{0, 1\}^2$ by thresholding, where $(0, 1)$ and $(1, 0)$ represent easy and difficult respectively. We randomly observe values of around 200 instances of *easy* difficulty level and we choose the maximum of those values as threshold. We note that we use a one-hot vector to represent difficulty levels so that it is easy to generalise it to multiple difficulty levels (e.g. easy, medium, hard).

3.3 Complex Question Decoder

Our decoder is a stack of Transformer decoders, conditioned on the difficulty level of the question to be generated. Similar to the encoder, the decoder also has multiple scaled dot-product attention layers along with feed forward neural network layers. Besides the self-attention, the decoder uses the final (top in Figure 1) encoder Transformer's output attention matrices ($\boldsymbol{K}_{encdec}$ and $\boldsymbol{V}_{encdec}$) in its encoder-decoder attention layer, which helps decoder in attending to (focusing on) important triples in the input subgraph.

The encoder-decoder attention layer works very similarly to the multi-head self attention layer described in Section 3.1 above. The main difference is that the encoder-decoder attention layer computes the query matrix ($\boldsymbol{Q}$) using the layer below it in the decoder and takes the key ($\boldsymbol{K}$) and values ($\boldsymbol{V}$) matrices from encoder output.

The output vector from the decoder stack is fed to a fully connected neural network (linear layer) which projects it to logits vector. Finally the softmax layer converts this logits vectors into a probability distribution over vocabulary, from which the question is decoded.

We encode difficulty into the decoder using a multi-layer perceptron DE consisting of an input linear layer followed by a rectified linear unit (ReLU) layer and an output linear layer. The input to DE is a length-two vector $\boldsymbol{x}$ representing a given difficulty level, as described in the previous subsection.

$$\text{DE}(\boldsymbol{x}) = Linear(ReLU(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})) \tag{7}$$

where $\boldsymbol{x}$ the difficulty level and $\boldsymbol{W} \in \mathbb{R}^{d_g \times 2}$ and $\boldsymbol{b} \in \mathbb{R}^{d_g}$ are trainable model parameters. we sum $\text{DE}(\boldsymbol{x})$ with decoder input to condition decoder to generate question of encoded difficulty.

At the decoder side, the order of question words is important. Therefore, we inject/add sequence order information to the decoder input. To represent order of the sequence we use fixed *positional encoding* with sine and cosine functions of different frequencies:

$$\text{PE}_{(pos,2i)} = \sin\left(pos/10000^{2i/d_g}\right) \tag{8}$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_g}\right) \tag{9}$$

where pos is the position and i is the index of dimension.

Label smoothing Label smoothing [27] has shown great impact especially in Transformers with multi-head attention. Adding label smoothing reduces expected calibration error. Motivated by previous work [7], we use label smoothing for regularisation with an uncertainty of 0.1. Our label smoothing technique is based on the Kullback-Leibler divergence loss. Instead of using the vanilla one-hot question word distribution, we build a distribution that has confidence of the correct word and distributes the rest of the smoothing mass throughout the output vocabulary.

4 Evaluation

Dataset & Preprocessing. We collected data from three recent multi-hop question answering datasets: WebQuestionsSP [34], ComplexWebQuestions [28], and PathQuestion [35][9], all of which are based on Freebase.

Each instance in WebQuestionsSP and ComplexWebQuestions contains a natural-language question, a corresponding SPARQ query and the answer entity and some other auxiliary information. For each instance, we convert its SPARQL query to return a subgraph instead of the answer entity, by changing it from a `SELECT` query to a `CONSTRUCT` query. An example `CONSTRUCT` query and its returned graph from the WebQuestionsSP dataset is shown in Example 2 below. We combine these two datasets and refer to them as WQ hereafter.

Example 2. An example `CONSTRUCT` query and the corresponding returned graph.

```
PREFIX ns: <http://rdf.freebase.com/ns/>
CONSTRUCT WHERE { FILTER (?x != ns:m.02189)
FILTER (!isLiteral(?x) OR lang(?x) = '' OR langMatches(lang(?x), 'en'))
ns:m.02189 ns:organization.organization.founders ?x .
?x ns:medicine.notable_person_with_medical_condition.condition ns:m.0g02vk .
}
```

The subgraph returned after executing the above query is given below. Note the Freebase prefix is omitted for brevity reasons.

```
m.02189 organization.organization.founders m.04xzm .
m.04xzm medicine.notable_person_with_medical_condition.condition m.0g02vk .
```

---

[9] Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=52763, https://www.tau-nlp.org/compwebq, and https://github.com/zmtkeke/IRN/tree/master/PathQuestion respectively.

PathQuestion (referred to as PQ hereinafter) is similar to WQ. However, PQ only contains *verbalised* entities and predicates but not their Freebase IDs. As a result, we process this dataset differently.

Conceptually, each of WQ and PQ is a set of tuples $\{(Q_t, G, E_A)\}$, where Q_t is a natural-language question, G is the subgraph from which the question is derived, and E_A is the set of answer entities to question Q_t. Brief statistics of the two datasets can be found in Table 1.

For each dataset (WQ and PQ), we split it into 80%, 10% 10% for training, validation and testing.

Table 1: Brief statistics of the collected datasets.

Dataset	# entities	# predicates	# hops	# instances
WQ [34,28]	25,703	672	2 to 100	22,989
PQ [35]	7,250	378	2, 3	9,731
Total	32,953	1,050	–	32,720

Implementation Details. We have implemented our multi-hop question generation model, denoted MHQG[10], in the PyTorch framework. For the state-of-the-art question generation model Learning to Ask (L2A) [9] used for comparison, we used its publicly available code.

We set triple embedding dimension for the Transformers to 512, i.e. $d_g = 512$. The triple embeddings are fine-tuned during training. We use 8 parallel attention heads with d_k and d_v set to 64. The dimension of encoder and decoder's fully connected feed forward neural nets is set to 2048. We set the number of layers of Transformers to six ($n = 6$ in Figure 1) for both the encoder and the decoder.

For the WQ dasatet, we obtain the pre-trained 50-dimensional TransE [5] embeddings for Freebase from OpenKE [14]. The embedding of a triple is the 150-dimensional concatenated embeddings of its components (subject, predicate and object). For initialisation, we extend the 150-dimensional embedding vectors to 512-dimensional with random values for the remaining 362 dimensions.

As PQ only contains lexicalised entities and predicates but not their IDs, we resort to using the pre-trained 300-dimensional GloVe [21] word embeddings for this dataset. For each entity/predicate, its embedding is the average of the embeddings of its words. We extend the embedding vectors to 512-dimensional with random values for the remaining 212 dimensions. As in WQ, the triple embeddings are fine-tuned during training.

For example, for the following (2-triple) subgraph in PQ `claudius#parents# nero_claudius_drusus#nationality#roman_empire`, the embedding of the entity `roman_empire` is the average of the GloVe embeddings of words `roman` and

---

[10] Available at `https://github.com/liyuanfang/mhqg`

`empire`. We mask entities in questions to a generic entity form such as "ENT1" to handle unseen entities, thus resolving the out of vocabulary issue. Entity masking also helps model learn better generalisations for similar training instances.

We use TAGME[11] for entity recognition and entity linking, and obtain the TAGME *confidence* score in difficulty estimation (Section 3.2).

We used the Adam optimiser [16] with $\beta_2 = 0.998$, initialised with learning rate 2 to optimise model parameters. The learning rate increases linearly for the first 800 training steps (warmup steps) and decreases thereafter proportionally to the inverse square root of the step number. For regularisation we apply dropout to the output of each layer with the dropout probability set to 0.1. We use beam search in the decoder with beam size of 10 for decoding question words. These parameters are empirically chosen using grid search.

All our models are trained on a single P100 GPU. We train for 15 epochs, and select the model with the minimum perplexity on the validation set to generate question on the test set for evaluation.

4.1 Results and Discussion

To the best of our knowledge, this work is the first to address multi-hop question generation from KG. Therefore, there is no other model that we can compare with directly. Existing text generation models such as GTR-LSTM [29] and GCN [20] are trained on a different KG (DBPedia instead of Freebase). Comparing them requires significant additional data preprocessing work, including entity linking, triple embedding, etc., as well as mandatory additional data (e.g. entity types) that are not available to us. As a result, we leave comparing with them to future work.

Instead, we use a state-of-the-art natural language QG model, Learning to Ask [9] (referred to as L2A hereinafter), as a baseline model for comparison. L2A is a recent LSTM-based Seq2Seq model that takes a sentence as input and generates a question. We train L2A using the linearised subgraph with 300-dimensional embeddings, which are fine-tuned during training. We use a 2 layer Bi-LSTM encoder and decoder with hidden unit size set to 600. The other hyper-parameters are set exactly the same as described in L2A [9]

We perform two experiments to evaluate the effectiveness of our proposed model: automatic evaluation using widely-used metrics including BLEU, GLEU and METEOR and human evaluation. We compare two variants of our model against L2A: with (MHQG+AE) or without (MHQG) answer encoding. The automatic evaluation is performed on the full test sets, whereas a subset of 50 randomly selected questions, 25 from each of WQ and PQ, are used in human evaluation.

The results of automatic evaluation are shown in Table 2. As can be seen, on both datasets, both our models outperform L2A substantially across all three evaluation metrics. On both datasets, our models outperform L2A on BLEU for 5.56 and 8.99 absolute points, representing a 93% and 53% respectively. On the

---

[11] https://tagme.d4science.org/tagme/

Table 2: Results of automatic evaluation. Best results for each metric is **bolded**.

Model	WQ			PQ		
	BLEU	ROUGE-L	METEOR	BLEU	ROUGE-L	METEOR
L2A	6.01	26.95	25.24	17.00	50.38	19.72
MHQG	11.49	34.61	27.65	24.98	58.08	31.32
MHQG+AE	**11.57**	**35.53**	**29.69**	**25.99**	**58.94**	**33.16**

PQ dataset, the differences in ROUGE-L and METEOR are more substantial than on the WQ dataset. Moreover, MHQG+AE, the model with answer encoding, also consistently exhibits better performance than without it.

Table 3: Results of human evaluation, showing percentages of questions with correct syntax, semantics and difficulty level for the two datasets and each model. The numbers in parentheses are the percentage of agreement between participants. Best results for each metric is **bolded**.

Model	WQ			PQ		
	Syntax	Semantics	Difficulty	Syntax	Semantics	Difficulty
L2A	78 (97)	80 (95)	48 (59)	67 (65)	65 (73)	58 (50)
MHQG	79 (75)	83 (81)	**60** (49)	64 (75)	70 (69)	**68** (44)
MHQG+AE	**98** (73)	**97** (76)	56 (53)	**78** (70)	**74** (62)	**68** (49)

In human evaluation, four participants were asked to judge the correctness of syntax, semantics and difficulty level of the questions generated by L2A and our models. The results are averaged across the four participants and are summarised in Table 3. For all evaluation criteria, both of our models outperform L2A. Notably, on WQ, our model MHQG+AE achieves 98% and 97% of syntactic and semantic correctness respectively. Overall MHQG+AE achieves best result, except the slightly lower percentage of correct difficulty level for WQ. This is consistent with the results in automatic evaluation in Table 2, where MHQG+AE also shows best performance.

Below in Example 3 we show two questions, one easy, and one hard, generated by our model MHQG+AE on a same graph. Fro brevity reasons only the localname of the freebase predicates are shown. As can be seen, the difference in hardness is obvious, showing the effectiveness of our model in controlling question difficulty.

Example 3. An example graph with two questions of different difficulty levels generated by our model MHQG+AE.

Graph:	m.0gtqy5p location m.0r0m6 <t> m.0gtqxxq location m.0fpzwf <t> m.01vrncs places_lived m.03pnpl8 <t> m.01vrncs film.film_subject.films m.0djlxb <t> m.03pnpl8 location m.0h1k6 <t> m.0gtqxxk location m.02_286 <t> m.01vrncs places_lived m.0gtqy5p <t> m.01vrncs places_lived m.0gtqxxk <t> m.0gtqy5h location m.0wjjx<t> m.01vrncs places_lived m.0gtqxxq <t> m.01vrncs places_lived m.0gtqy5h <t>
Easy:	where did bob dylan live?
Hard:	where did the subject of the film "I'm Not There" live?

Example 4 below shows two complex, 7-hop and 4-hop, subgraphs from WQ and the questions generated on them, by L2A and our two models. As can be seen, our models generate questions of much higher quality than L2A.

Example 4. Two subgraphs and questions generated by different models.

Graph:	m.0jjl89y office_position_or_title m.0j6tpbb <t> m.0hqg6pb office_position_or_title m.0j6tpbb <t> m.03gj2 official_language m.02ztjwg <t> m.03gj2 governing_officials m.0hqg6m3 <t> m.03gj2 governing_officials m.0jjl89y <t> m.03gj2 governing_officials m.0hqg6pb <t> m.0hqg6m3 office_position_or_title m.0j6tpbb <t>
L2A:	what language is spoken in the governmental jurisdiction?
MHQG:	what is the spoken language in the country with governmental position prime minister of hungary?
MHQG+AE:	what language is spoken in the governmental jurisdiction where prime minister of hungary holds office?

Graph:	m.0d04z6 currency_used m.049p2z <t> m.0d04z6 national_anthem m.048z_y1 <t> m.0d04z6 currency_used m.049p6c <t> m.048z_y1 anthem m.01lg5j <t>
L2A:	the country that contains uses what type of currency?
MHQG:	what is the currency used in the country with la bayamesa as its national anthem?
MHQG+AE:	what currency is used in the country with national anthem la bayamesa?

5 Conclusion

In this paper we present a novel technique for the automatic generation of complex, multi-hop questions from knowledge graphs. Our technique takes a subgraph as input, encodes the answer, estimates the difficulty level, and generates a natural-language question from the subgraph. We employ a Transformer-based encoder-decoder model that is conditioned on the difficulty level. Experiments

were performed on three recent multi-hop question-answering datasets to assess the quality of generated questions, by both widely-used evaluation metrics and human judgements. Compared to a state-of-the-art text question generation technique, our method generates questions that are more fluent and relevant with tunable difficulty levels.

Our technique allows the generation of complex questions over a large knowledge without any manual intervention. This ability can facilitate the continued improvements of knowledge graph question answering methods by providing substantial amount of new training data with minimal cost.

We have planned a number of further research directions. Firstly, we will investigate a more refined estimation of difficulty levels, taking into account more comprehensive information such as predicates and the graph itself, but not only entities. Secondly, taking into account additional information sources such as background ontologies as entity and predicate definitions is also worth investigating.

References

1. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on Freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1533–1544. Association for Computational Linguistics (2013)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia-a crystallization point for the web of data. Web Semantics: Science, Services and Agents on the World Wide Web **7**(3), 154–165 (2009)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250. AcM (2008)
4. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
5. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
6. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014)
7. Chorowski, J., Jaitly, N.: Towards better decoding and language model integration in sequence to sequence models. In: Proc. Interspeech 2017. pp. 523–527 (2017)
8. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowledge and Information Systems **55**(3), 529–569 (2018)
9. Du, X., Shao, J., Cardie, C.: Learning to ask: Neural question generation for reading comprehension. In: ACL. vol. 1, pp. 1342–1352 (2017)
10. Elsahar, H., Gravier, C., Laforest, F.: Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In: Proceedings of the 2018 Conference

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 218–228. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018)

11. Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., Vrandečić, D.: Introducing Wikidata to the Linked Data web. In: International Semantic Web Conference. pp. 50–65. Springer (2014)

12. Fan, Z., Wei, Z., Li, P., Lan, Y., Huang, X.: A question type driven framework to diversify visual question generation. In: IJCAI. pp. 4048–4054 (2018)

13. Gardent, C., Shimorina, A., Narayan, S., Perez-Beltrachini, L.: Creating training corpora for nlg micro-planners. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 179–188. Association for Computational Linguistics, Vancouver, Canada (Jul 2017)

14. Han, X., Cao, S., Lv, X., Lin, Y., Liu, Z., Sun, M., Li, J.: OpenKE: An open toolkit for knowledge embedding. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 139–144. Association for Computational Linguistics, Brussels, Belgium (Nov 2018)

15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)

16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

17. Kumar, V., Boorla, K., Meena, Y., Ramakrishnan, G., Li, Y.F.: Automating reading comprehension by generating question and answer pairs. In: 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) (2018)

18. Li, Y., Duan, N., Zhou, B., Chu, X., Ouyang, W., Wang, X., Zhou, M.: Visual question generation as dual task of visual question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6116–6124 (2018)

19. Lopez, V., Unger, C., Cimiano, P., Motta, E.: Evaluating question answering over Linked Data. Web Semantics Science Services And Agents On The World Wide Web 21, 3–13 (2013)

20. Marcheggiani, D., Perez-Beltrachini, L.: Deep graph convolutional encoders for structured data to text generation. In: Proceedings of the 11th International Conference on Natural Language Generation. pp. 1–9. Association for Computational Linguistics, Tilburg University, The Netherlands (Nov 2018)

21. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014)

22. Reddy, S., Raghu, D., Khapra, M.M., Joshi, S.: Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pp. 376–385. Association for Computational Linguistics (2017)

23. Saha, A., Pahuja, V., Khapra, M.M., Sankaranarayanan, K., Chandar, S.: Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

24. Serban, I.V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., Bengio, Y.: Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. arXiv preprint arXiv:1603.06807 (2016)

25. Seyler, D., Yahya, M., Berberich, K.: Generating quiz questions from knowledge graphs. In: Proceedings of the 24th International Conference on World Wide Web. pp. 113–114. WWW '15 Companion, ACM, New York, NY, USA (2015)

26. Song, L., Wang, Z., Hamza, W., Zhang, Y., Gildea, D.: Leveraging context information for natural question generation. In: NAACL (Short Papers). vol. 2, pp. 569–574 (2018)

27. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)

28. Talmor, A., Berant, J.: The Web as a knowledge-base for answering complex questions. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 641–651. Association for Computational Linguistics (2018)

29. Trisedya, B.D., Qi, J., Zhang, R., Wang, W.: GTR-LSTM: A triple encoder for sentence generation from rdf data. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1627–1637. Association for Computational Linguistics, Melbourne, Australia (Jul 2018)

30. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: A corpus for complex question answering over knowledge graphs. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) The Semantic Web – ISWC 2017. pp. 210–218. Springer International Publishing, Cham (2017)

31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NIPS). pp. 5998–6008 (2017)

32. Vougiouklis, P., Elsahar, H., Kaffee, L.A., Gravier, C., Laforest, F., Hare, J., Simperl, E.: Neural Wikipedian: Generating textual summaries from knowledge base triples. Journal of Web Semantics **52-53**, 1 – 15 (2018)

33. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)

34. Yih, W.t., Richardson, M., Meek, C., Chang, M.W., Suh, J.: 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). pp. 201–206. Association for Computational Linguistics (2016)

35. Zhou, M., Huang, M., Zhu, X.: An interpretable reasoning network for multi-relation question answering. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 2010–2022. Association for Computational Linguistics (2018)