A Simple and Effective Model for Multi-Hop Question Generation

Anonymous ACL submission

Abstract

Previous research on automated question generation has almost exclusively focused on generating factoid questions whose answers can be extracted from a single document. However, there is an increasing interest in developing systems that are capable of more complex *multi-hop question generation* (QG), where answering the question requires reasoning over multiple documents. In this work, we propose a simple and effective approach based on the transformer model for multi-hop QG. Our approach consists of specialized input representations, a supporting sentence classification objective, and training data weighting. Prior work on multi-hop QG considers the simplified setting of shorter documents and also advocates the use of entity-based graph structures as essential ingredients in model design. On the contrary, we showcase that our model can scale to the challenging setting of longer documents as input, does not rely on graph structures, and substantially outperforms the state-of-the-art approaches as measured by automated metrics and human evaluation.

1 Introduction

Motivated by the process of human inquiry and learning, the field of question generation (QG) requires a model to generate natural language questions in context. QG has wide applicability in automated dialog systems (Mostafazadeh et al., 2016; Fitzpatrick et al., 2017), language assessment (Settles et al., 2020), data augmentation (Tang et al., 2017), and the development of annotated data sets for question answering (QA) research.

Most prior research on QG has focused on generating relatively simple *factoid-based* questions, where answering the question simply requires extracting a span of text from a single reference document (Zhao et al., 2018; Kumar et al., 2019; Chen et al., 2020). However, motivated by the desire to build NLP systems that are capable of more sophisticated forms of reasoning and understanding (Kaushik and Lipton, 2018; Sinha et al., 2019), there is an increasing interest in developing systems for *multi-hop* question answering and generation (Zhang et al., 2018; Welbl et al., 2018; Yang et al., 2018; Dhingra et al., 2020), where answering the questions requires reasoning over the content in multiple documents (see Figure 1 for an example).

Document 1: *Byron Edmund Walker*
[**1**] Sir Byron Edmund Walker, CVO (14 October 1848 – 27 March 1924) was a Canadian banker. [**2**] He was the president of the Canadian Bank of Commerce from 1907 to 1924, and a generous patron of the arts, helping to found and nurture many of Canada's cultural and educational institutions, including the University of Toronto, National Gallery of Canada, ...
Document 2: *University of Toronto*
[**1**] The University of Toronto (U of T, UToronto, or Toronto) is a public research university in Toronto, ... [2] It was founded by royal charter in 1827 as ... [**3**] Originally controlled by the Church of England, the university assumed the present name ....

Answer: The University of Toronto

Supporting Facts
Document 1: {1, 2}, Document 2: {1, 3}

Question
Which *Byron Edmund Walker founded institution* was *originally controlled* by the *Church of England*?

Figure 1: An example illustrating the multi-hop QG task. The inputs are the two documents, answer, and supporting facts. The task is to generate a question such that it is answerable only after reading both the documents. Entities and predicates relevant to the task are underlined while supporting facts are shown in color.

Unlike standard QG, generating multi-hop questions requires the model to understand the relationship between disjoint pieces of information in multiple context documents. Compared to standard QG, multi-hop questions tend to be substantially longer, contain a higher density of named entities, and—perhaps most importantly—high-quality multi-hop questions involve complex chains of predicates connecting the mentioned entities (see Appendix A for supporting statistics.)

To address these challenges, existing research on multi-hop QG primarily relies on graph-to-sequence (G2S) methods (Pan et al., 2020; Yu et al., 2020; Su et al., 2020). These approaches construct graph inputs by augmenting the original text with structural information (e.g., entity mentions and dependency parses) and then apply graph neural networks (GNNs) (Hamilton, 2020) whose embeddings are then fed to an autoregressive sequence decoder. However, the necessity of these complex G2S approaches—which require designing hand-crafted graph extractors—is not entirely clear, especially when standard transformer-based sequence-to-sequence models (Vaswani et al., 2017) can induce a strong relational inductive bias among its inputs (Battaglia et al., 2018). Due to this, one might imagine that the transformer model alone would suffice for the relational reasoning requirements of multi-hop QG, *i.e.* to reason about relationships between entities in the text.

This work: We show that a simple training approach based on the transformer model is sufficient to outperform previous methods, which rely on graph-based augmentations, on the multi-hop QG task. Our training method consists of specialized input representations to impart useful inductive biases for answer conditioning, a supporting sentence classification objective to identify salient sentences in a document, and a training data weighting approach to pay higher importance to relevant training examples. These techniques help our model obtain new state-of-the-art results outperforming the previous records by more than 4 BLEU points on the widely used HotpotQA dataset (Yang et al., 2018). Our model is also robust to the challenging setting of long document inputs, on which it obtains a gain of 7.5 BLEU points. In addition, we also propose a graph-augmented transformer encoder (GATE)—which integrates explicit graph structure information into the transformer. However, we show that the gains induced by the graph augmentations are relatively small compared to other enhancements in our training pipeline.

2 Methods

We first formalize the multi-hop QG task and describe how we adapt the standard transformer architecture to multi-hop QG (§ 2.2). Then, we present two techniques that are critical to achieving strong performance: an auxiliary supporting sentence classification objective (§ 2.3) and a training data reweighting approach (§ 2.4). Lastly, we outline an approach for augmenting the transformer model with graph-structured data (§ 2.5).

Background The input to the multi-hop QG task is a set of context documents $\{c_1, \ldots, c_k\}$ and an answer a. These documents can be long containing multiple sentences, $c_j = [s_1, \ldots, s_n]$, where each s_i is a sequence of words. Sentences across different documents contain common *named entities*, which are also known as *bridge entities*. The answer a spans one or multiple tokens in one document. The desired goal of a multi-hop QG model is to generate a question (q) conditioned on the context and the answer, such that answering the question requires reasoning about the content utilizing multiple context documents.

2.1 Proposed Model

We formulate multi-hop QG as a sequence-to-sequence (S2S) learning task, where our network is a transformer-based model (Vaswani et al., 2017). The input to the transformer are all the context documents $\{c_1, \ldots, c_k\}$ and the provided answer (a). In the transformer model, both the encoder and decoder consist of self-attention and feed-forward sublayers,[1] which are trained using teaching-forcing and a negative log-likelihood loss (Williams and Zipser, 1989). We found that achieving strong performance with a transformer requires careful design decisions in terms of how the input is represented, which we present next.

2.2 Input Representations

Sentence markers As the sentences present in the document context are expected to play a crucial role in training, we add additional annotations to the input in order to learn sentence embeddings. Learning these sentence embeddings adds a form of implicit regularization, and we also leverage these embeddings in our auxiliary objective (§2.3). In particular, we add a *sentence id token* before the first token of each sentence. In practice, as the number of sentences varies between examples, to make the model more robust to this difference, we *tie* the sentence id token embedding weights for all the sentences and refer to it as the <SENT> token.

Answer span representation To provide answer tokens as input to the encoder, the prevalent technique in QG approaches is to append the answer

[1] Due to space limitations, we include a description of the self-attention and feed-forward sublayers in Appendix B.

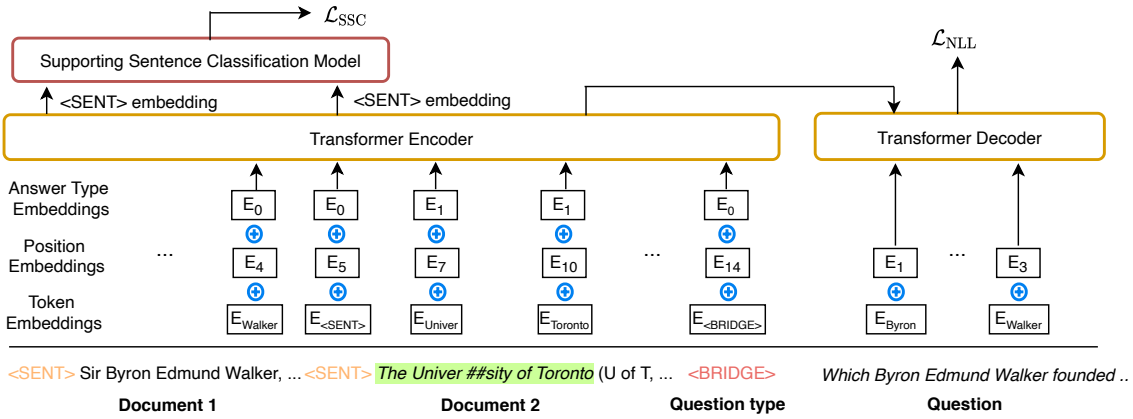

Figure 2: Schematic diagram illustrating our proposed model for multi-hop QG. <SENT> token is prepended before every sentence. The tokens highlighted in green denotes the answer tokens.

tokens after the context (Dong et al., 2019). However, we found this approach to substantially underperform in multi-hop QG. A possible reason is that concatenation of the answer tokens imparts poor inductive biases to the decoder. To overcome this limitation, we define indicator *answer type id* tokens in which the value of type ids is `1` for the answer span tokens (within the context) and `0` for the remaining tokens. We introduce a new embedding layer for answer type ids called *answer type embeddings*. For a token, its input representation to the transformer encoder is the sum of token, position, and answer type embeddings.

A schematic diagram of the proposed model is shown in Figure 2.

2.3 Training Objective

To train our S2S transformer model, we combine two loss functions. The first loss function is the standard S2S log-likelihood loss, while the second loss function trains the model to detect salient sentences within the document context.

2.3.1 Negative log-likelihood objective

The primary training signal for our S2S approach comes from a standard negative log-likelihood loss

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{K}\sum_{k=1}^{K}\log p\left(q_k \mid c, q_{1:k-1}; \theta\right),$$

where the parametric distribution $p(q \mid c; \theta)$ models the conditional probability of question (q) given the context (c) and K is the number of question tokens. As is common practice in the literature, we use teacher-forcing while training with this loss.

2.3.2 Auxiliary Supporting Sentence Classification Objective

To compliment our standard likelihood objective, we also design an auxiliary objective, which trains the model to detect the occurrence of *supporting facts* in the multi-document context.

Supporting facts in multi-hop QA As multi-hop questions require reasoning over some salient sentences across long documents, a unique challenge of multi-hop QA is the presence of a large number of irrelevant sentences in context. Thus, as a common practice, researchers annotate which sentences are necessary to answer each question, called *supporting facts* (Yang et al., 2018). Prior work on multi-hop QG leveraged these annotations by simply discarding all irrelevant sentences and training only on sentences with supporting facts (Pan et al., 2020). Instead, we propose a supporting sentence classification objective, which allows us to leverage these annotations during training while still receiving full document contexts at inference.

Supporting sentence classification (SSC) Our classifier training setup utilizes sentences contained in the supporting facts as positive examples while we consider all the remaining sentences in the context to be negative examples. We use only the sentence id embedding (h_i) for training *i.e.*, the embedding corresponding to the <SENT> token; see §2.2. The training loss is defined in terms of the binary cross-entropy loss formulation as

$$\mathcal{L}_{\text{SSC}} = \frac{-1}{T}\left(\sum_{i=1}^{P}\log \mathcal{D}\left(h_i; \phi\right) + \sum_{j=1}^{N}\log\left(1 - \mathcal{D}\left(h_j; \phi\right)\right)\right),$$

where $\mathcal{D}(\phi)$ is a binary classifier consisting of a two-layer MLP with ReLU activation and a final

sigmoid layer, P and N are the number of positive and negative training sentences in the context documents respectively, and $T = P + N$. During evaluation, we predict the supporting facts using the binary classifier. This objective is added as an additional term to the main likelihood loss, leading to the following composite objective:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{NLL}} + (1 - \lambda)\mathcal{L}_{\text{SSC}},$$

where λ is a hyperparameter.

2.4 Training Data Weighting

Another key component of our training pipeline is a data weighting approach. This aspect specifically addresses challenges arising from the question-length distribution in the widely used HotpotQA benchmark (Yang et al., 2018), which is also used in this work. Nonetheless, despite the fact that this approach is motivated directly by the statistics of HotpotQA, we expect the general principle to be applicable to future multi-hop datasets as well.

Motivation HotpotQA's training set contains three categories of questions: *train-easy*, *train-medium*, and *train-hard*. Train-easy questions are essentially single-hop requiring one context document to answer them while train-medium and train-hard questions are multi-hop requiring multiple context documents. *However, both the dev and test sets in HotpotQA consist of hard multi-hop questions*. While the additional *train-easy* and *train-medium* examples have proved useful as training signals in the question-answering setting (Yang et al., 2018), our QG experiments reveal that naively training the model using all the questions leads to a big drop in BLEU scores (Papineni et al., 2002). The reason for the low scores is that the generated questions are almost 80% longer than the reference questions, and thus are less precise.

Data weighting to prevent distributional mismatch We hypothesize that the model generates long questions because of the *negative exposure bias* that it receives from the train-easy questions. We analyze the question-length distribution in the training set in Figure 3 and observe that a significant number of train-easy questions are much longer than train-medium and train-hard—while most of the train-medium and train-hard questions are 30 words long, train-easy questions can be as long as 70 words. Therefore, we strive to match the training-evaluation question-length distribution

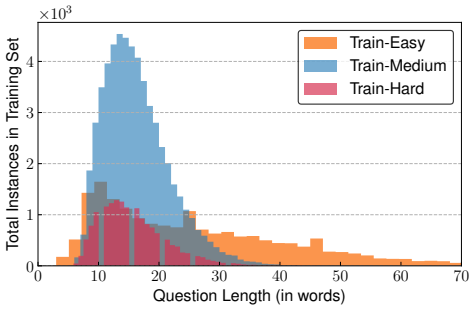

Figure 3: Question length distribution according to its difficulty level in the HotpotQA training set. Plot reveals that *train-easy* questions are much longer than *train-medium* and *train-hard* questions.

by down-weighting the importance of examples whose question length is more than 30 words during the training process. More formally, let there be m examples in a batch, let $|q|$ denote the question length, then the log-likelihood loss is calculated as

$$\mathcal{L}_{\text{NLL}} \propto \sum_{i=1}^{m} (1 - \epsilon)\mathbb{1}(|q_i| \leq t)\mathcal{L}_{\text{NLL}}^{(i)} + \epsilon \mathbb{1}(|q_i| > t)\mathcal{L}_{\text{NLL}}^{(i)},$$

where ϵ is a small constant ($\epsilon = 0.05$) and t is a length threshold ($t = 30$). As our analysis indicates, most of the down-weighted examples are train-easy questions. A special case is when we set $\epsilon = 0$, i.e. *data filtering*, where we ignore all the train-easy questions from the training process.

2.5 Graph Augmentations to Transformer Encoder (GATE)

The context contains useful structural information such as *named entities* and *relations* among them. Recent approaches represent this information with multi-attribute graphs and apply GNNs to learn useful representations from them (Pan et al., 2020). Following these trends, to ascertain the usefulness of graphs, we extract graph structure from the input context. We consider three types of nodes—*named-entity mentions*, *coreferent-entities*, and *sentence-ids*—and construct a *multi-relational graph* with three relation types over these nodes (Figure 4).

Based on prior work (Sachan et al., 2021), we introduce augmentations to the transformer encoder enabling it to leverage graph inputs and refer to this as the GATE model. Specifically, we introduce two additional sublayers: (1) *graph-attention*, which is similar to self-attention but attends to connecting nodes (Veličković et al., 2018) (2) *fused-attention sublayer*, where we combine the outputs of self-attention and graph-attention by passing them through a one-layer MLP. For more details,

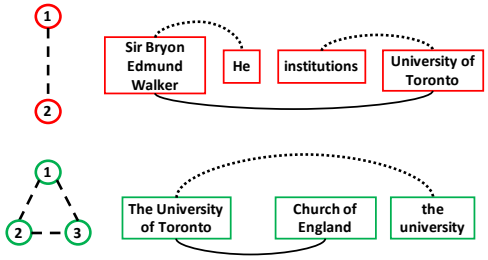

Figure 4: Entity-centric graph corresponding to the example in Figure 1. The sentence nodes are drawn in circles and the entities in rectangles. Entity edges are drawn in bold, coreference edges are dotted and sentence edges are dashed.

we refer the reader to Appendix C, where we elucidate the architecture of the GATE model.

3 Experimental Setup

3.1 Dataset Preprocessing and Evaluation

We use the HotpotQA dataset for experiments as it is the only multi-hop QA dataset that contains questions in textual form.[2] HotpotQA is a large-scale crowd-sourced dataset constructed from Wikipedia articles and contains over 100K questions. We use its *distractor* setting that contains 2 gold and 8 distractor paragraphs for a question. Following prior work on multi-hop QG, we limit the context size to the 2 gold paragraphs, as the distractor paragraphs are irrelevant to the generation task (Pan et al., 2020). The questions can be either of type *bridge*- or *comparison*-based. The answer span is not explicitly specified in the context documents rather the answer tokens are provided. Hence, we use approximate text-similarity algorithms to search for the best matching answer span in the context. For some of the comparison questions whose answer is either *yes* or *no*, we append it to the context.

To train and evaluate the models, we use the standard training and dev sets.[3] We pre-process the dataset by excluding examples with spurious annotations. As the official dev set is used as a test set, we reserve 500 examples from the training set to be used as a dev set. Overall, our training set consists of 84,000 examples, and the test set consists of 7,399 examples. We follow the evaluation protocol of Pan et al. (2020) and report scores on standard automated evaluation metrics common in QG: BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Banerjee and Lavie, 2005). We also performed human evaluation studies to assess our model performance.

[2]We also explored WikiHop (Welbl et al., 2018) and WikiData (Dhingra et al., 2020), but they contain questions in triple format and thus are outside the scope of this work.

[3]As the test set is hidden for HotpotQA.

Model	BLEU-4	ROUGE-L	METEOR
Encoder Input: Supporting Facts Sentences			
NQG++†	11.50	32.01	16.96
ASs2s†	11.29	32.88	16.78
MP-GSA†	13.48	34.51	18.39
SRL-Graph†	15.03	36.24	19.73
DP-Graph†	15.53	36.94	20.15
TE_{NLL}	**19.33**	**39.00**	**22.21**
Encoder Input: Full Document Context			
TE_{NLL}	17.13	38.13	21.34
$TE_{NLL+SSC}$	**19.60**	**39.23**	**22.50**

Table 1: Results of multi-hop QG on HotpotQA. NQG++ is from Zhou et al. (2018), ASs2s is from Kim et al. (2019), MP-GSA is from Zhao et al. (2018), SRL-Graph and DP-Graph are from Pan et al. (2020). † denotes that the results are taken from Pan et al. (2020). Best results in each section are highlighted in bold.

3.2 Training Protocols

We follow the same training process for all the experiments. We encode the context and question with 32K subwords units by applying the *sentencepiece* toolkit (Kudo and Richardson, 2018). For a fair comparison with previous work, we experiment with smaller models. We use a 2-layer transformer with 8 attention heads, 512-D model size, and 2048-D hidden layer. Token embedding weights are shared between the encoder, decoder, and generation layer. For variance control, we average our results over 5 independent runs. For reproducibility, we describe model training details in Appendix D.

4 Results and Analysis

We report the performance of our proposed transformer encoder (TE) model in Table 1, comparing with a number of recent QG models.

Performance with supporting facts as input We first consider a *simplified version* of the task when *only the supporting facts are used during training and testing* (top section in Table 1). In other words, in this setting, we remove all sentences of the context documents that have not been annotated as supporting facts. This is an overly simplified setting since supporting fact annotations are not always available at test time. However, this is the setting used in previous work on multi-hop QG (Pan et al., 2020), which we directly compare to. In this setting, we see that our TE model scores 19.3 BLEU, an absolute gain of around 4 points over the

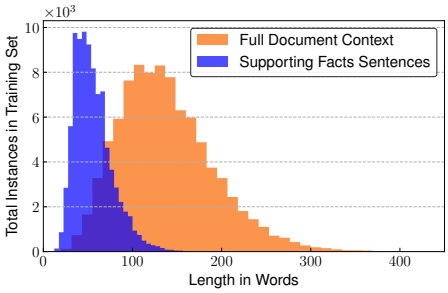

Figure 5: Length distribution of the full document context and supporting facts sentences in HotpotQA. The plot reveals that the full document context is almost three times longer than supporting facts.

Setting	BLEU-4
$\text{TE}_{\text{NLL+SSC}}$	19.60
– w/o SSC	17.13
– w/o Training data weighting	14.50
– w/o Training data weighting & SSC	11.90
– w/o Answer type embeddings	7.81

Table 2: Ablation studies when the encoders' input is the full document context. SSC: Supporting sentence classification objective.

previous best result. Note that as the SSC training is not applicable here, we just use teacher-forcing.

Performance with full context as input In a more realistic setting, when the supporting facts are not available at test time, the model needs to processes the full context. As the average document context is three times the size of the supporting facts in HotpotQA (Figure 5), this setting is much more challenging, which is also evident from the results (bottom section in Table 1). We notice that if using only the teacher-forcing objective, the performance drops by 2 points. However, when trained with the composite objective, our model obtains a BLEU score of 19.6, which is in fact slightly higher than what it could achieve in the simplified setting. We hypothesize that the additional training signal from the longer contexts combined with the SSC objective actually benefits the models.

4.1 Ablation Studies

We perform ablations to understand what components are essential for strong performance when the input is the full document context (Table 2).

SSC training We see that SSC training improves the BLEU score by 2.5 points and it also helps the model attain around 75 F1 and 35 Exact Match scores in predicting the supporting facts sentences. This highlights that—besides being helpful in QG—SSC training additionally imparts useful signals to

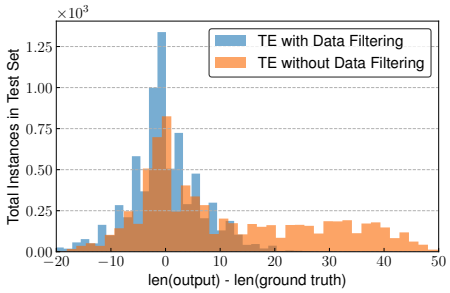

Figure 6: Effect of data filtering ($\epsilon = 0$) on QG.

the encoder such that its supporting facts predictions are *interpretable*.

Training data weighting We showcase that weighting of questions less than 30 words is a critical step in our training pipeline. On the other hand, without data weighting ($\epsilon = 0.5$), the accuracy drops by 5 points to 14.5 BLEU. In addition, if we just use teacher-forcing objective without data weighting, the performance further drops to 11.9 BLEU, signifying that data weighting and SSC training are independently useful and essential. To highlight the effect of data weighting on the generation quality, we plot the generated question length's deviation from the ground truth in Figure 6. We observe that when a model is trained without data weighting, a substantial number of generated questions are at least 15 words longer than the ground truth, thus leading to lower quality output, while this effect is not as pronounced otherwise.

Answer span representation We demonstrate that effective encoding of the answer span is of utmost importance in multi-hop QG as the decoder needs to condition generation on both the context and the answer. As mentioned previously, the common approach in standard QG is to append the answer tokens after the context. We see that this approach results in a drop of 12 points to 7.8 BLEU, which is quite low. Our proposed approach of encoding the answer span in the context with answer type embeddings is a much stronger methodology.

4.2 Discussion

Overall, the above results demonstrate that all of our proposed enhancements to the transformer model and training pipeline—input representations, SSC objective, and effective data weighting during training—helps in obtaining state-of-the-art results. Compared to the previous work of (Pan et al., 2020) who leverage semantic and dependency graphs on the unrealistic setting of shorter supporting facts, our model works well with full document contexts

Model	BLEU-4	ROUGE-L	METEOR
Encoder Input: Full Document Context			
$\text{TE}_{\text{NLL+SSC}}$	19.60	39.23	22.50
$\text{GATE}_{\text{NLL+SSC}}$	20.02	39.49	22.40
Ensemble	**21.34**	**40.36**	**23.24**

Table 3: Performance comparison of the TE and GATE models and their ensemble.

Model	G	SC	A	QC
DP-Graph†	3.42	3.26	3.26	2.06
$\text{Ensemble}_{\text{TE+GATE}}$	**4.56**	**4.66**	**4.08**	**3.26**
Ground truth	4.66	4.78	4.36	3.24

Table 4: Human evaluation results on 300 example generations from the HotpotQA test set. The table shows the average rating from our annotators on a scale from 1-5, where 1 indicates the lowest score and 5 the highest score on four criteria: **Grammaticality (G)**, **Semantic Correctness (SC)**, **Answerability (A)**, and **Question Complexity (QC)**. Please refer to Appendix E for more details on these criteria. †DP-Graph is from Pan et al. (2020). We use the generations from their best released checkpoint to do human evaluation.

that are almost three times as long, is simpler, and more accurate. We believe that scaling up the training to larger and deeper models would help further improve the generation accuracy. Our strong results without the dependence on structured graphs also point to the hypothesis that the transformer architecture can facilitate relational reasoning among its input, which we empirically analyze next.

4.3 Importance of explicit graph-structures

From the results in Table 3, we find that our graph-augmented model (GATE) leads to a very small performance gain, achieving a BLEU of 20.02, which is quite close to the TE model. This suggests that incorporating graph-structures may not be an essential component in the model design, as opposed to the prevailing formalism in the literature.

Model ensemble We notice that the GATE model seems to provide complementary strengths compared to the TE model, which is evident when we ensemble them during decoding. At every step, the likelihood is computed using the linearly interpolated likelihood scores of both the models,

$$p\left(q_k \mid c, q_{1:k-1}\right) = \alpha \cdot p_{\text{TE}}\left(q_k \mid c, q_{1:k-1}\right) + (1-\alpha) \cdot p_{\text{GATE}}\left(q_k \mid c, q_{1:k-1}\right),$$

where $\alpha \in [0, 1]$ is a hyperparameter. We find that $\alpha = 0.5$ works the best. From Table 3, we see that the ensemble of both models results in an improvement of 1.7 points over the TE model. This suggests that—while the gains from graph-augmentations are relatively small—there is complementary information in the graph structures.

Question-level scores comparison In order to further understand how the GATE model is different in performance from the TE model, we perform an analysis of the generated questions on the test set. We analyze the distribution of the difference in their question-level GLEU scores (Wu et al., 2016)[4] and observe that on 397 (5.4%) test examples, the GATE model achieves a GLEU score of 20 points or more than that of the TE, while on 377 (5.1%) examples the TE model achieves at least 20 points higher. Therefore, this complementary performance is the reason for the gain that we see in Table 3 when the two models are ensembled.

[4]GLEU is a sentence-level metric that is calculated as the minimum of the precision or recall between the reference and the hypothesis.

4.4 Human Evaluation

We present the results of human evaluation in Table 4, where predictions from the models are assigned a rating out of 1-5 for four criteria to assess different aspects of multi-hop QG which are: to which degree the questions are (a) grammatically, and (b) semantically correct, (c) conditioned on the provided answer, and (d) complex, i.e. if they require reasoning over multiple supporting sentences or not. We compare the predictions from our best performing model with the previous state-of-the-art model from literature (Pan et al., 2020) and also provide human evaluation scores for ground truth data. We observe that despite being trained on much longer context documents, the generation quality of our ensemble model is vastly superior on all the four criteria compared to the previous best model and in fact comes very close to the ground truth ratings. It is also worth mentioning that the questions from our model are *more complex* and *better conditioned on the answer text*.

We also provide evaluation scores using pre-trained language models in Appendix F.

4.5 Qualitative Analysis of Data Weighting

In Table 5, we present example generations from the $\text{TE}_{\text{NLL+SSC}}$ model when it is trained with and without data weighting. We observe that when trained with data weighting, the model is able to generate a question that closely matches the ground truth. On the other hand, the model trained with-

Model	Document Contexts and Generated Questions
Document context	**Document 1**: Pinhead Gunpowder is an American punk rock band that formed in East Bay, California, in 1990. The band currently consists of Aaron Cometbus (drums, lyrics), Bill Schneider (bass), Billie Joe Armstrong (guitar, vocals) and Jason White (guitar, vocals). The band's name comes . . . **Document 2**: Billie Joe Armstrong (born February 17, 1972) is an American musician, singer, songwriter and actor who is best known as the lead vocalist, primary songwriter, and guitarist of the punk rock band Green Day, which he co-founded with Mike Dirnt. He is also a guitarist and vocalist for the punk rock band Pinhead . . .
Answer	band
Ground truth	*What kind of group does Pinhead Gunpowder and Billie Joe Armstrong have in common?*
$TE_{NLL+SSC}$	*What kind of group does Pinhead Gunpowder and Billie Joe Armstrong have in common?*
$TE_{NLL+SSC}$ w/o data weighting	*Pinhead Gunpowder is an American punk rock band that formed in East Bay, California, in 1990, the band currently consists of Aaron Cometbus (drums, lyrics), an American musician, singer, songwriter and actor who is best known as the lead vocalist, primary songwriter, and guitarist of the punk rock Green Day, which he co-founded with*
Document context	**Document 1**: Seesaw is a musical with a book by Michael Bennett, music by Cy Coleman, and lyrics by Dorothy Fields. **Document 2**: Michael Bennett (April 8, 1943 – July 2, 1987) was an American musical theatre director, writer, choreographer, and dancer. He won seven Tony Awards for his choreography and direction of Broadway shows and was nominated for an additional eleven.
Answer	April 8, 1943
Ground truth	*When was the writer of Seesaw born?*
$TE_{NLL+SSC}$	*When was the writer of Seesaw born?*
$TE_{NLL+SSC}$ w/o data weighting	*When was the American musical theatre director, writer, choreographer, and dancer born who's choreography and direction of Seesaw?*

Table 5: Example illustrating the effect of data weighting on multi-hop QG. Without data weighting the generated question is long and does not resemble a valid multi-hop question.

out it has considerably longer generations, often covering many pieces of information from input documents. It is also evident that sometimes these long generations are based on one document, and therefore are not true multi-hop questions.

5 Related Work

Most work on QG has focused on generating one-hop questions using neural S2S models (Du et al., 2017), pre-trained transformers (Dong et al., 2019), reinforcement learning-based query reformulation (Buck et al., 2018), and G2S model (Chen et al., 2020). Previously, (Pan et al., 2020; Yu et al., 2020) also propose approaches for multi-hop QG. They incorporate an entity-graph to capture information about entities and their contextual relations within as well as across documents.

In parallel, there have been advances in multi-hop question answering models (Tu et al., 2019; Chen et al., 2019; Tu et al., 2020; Groeneveld et al., 2020). In this task, GNN models applied over the extracted graph structures have led to improvements (De Cao et al., 2019; Fang et al., 2019; Zhao et al., 2020). Our work examines the complementary task of multi-hop QG and provides evidence that transformer models could, in fact, achieve competitive results in this task, compared to these GNN-based models that use explicit graph structures.

Also related to our work is the recent line of work on graph-to-text transduction (Xu et al., 2018; Koncel-Kedziorski et al., 2019; Zhu et al., 2019; Cai and Lam, 2020; Chen et al., 2020). However, these works seek to generate text from a structured input, rather than the setting we examine, which involves taking long context text as the input.

6 Conclusion and Future Work

We propose a simple and effective transformer-based model for multi-hop QG. We introduce architectural enhancements such as answer type embeddings, a supporting sentence classification objective to identify the salient facts, and a training data weighting approach. Experiments on HotpotQA showcase that our model outperforms the current best model by 4 BLEU points, which is further validated by human evaluation. Our analysis reveals that graph-based modeling may not be the most critical component in improving performance.

We present several research directions for future work on this topic. One direction is to leverage recent pre-trained generative language models such as GPT-3 (Brown et al., 2020) or T5 (Raffel et al., 2020) and finetune them using our proposed techniques. Another direction is to pre-train multi-hop QA systems by generating synthetic multi-hop questions using the ideas in this work.

Broader Impact and Ethics Statement

In terms of the ethical context of our work, it is important to consider the real-world use cases, implications, and potential stakeholders (e.g., potential individuals who may interact with systems built on the methods we propose). The primary real-world application of our methods is in dialogue or virtual assistant applications, where our techniques could be used to improve the question-asking ability of such systems. However, we note that we do not intend our trained systems to be employed off-the-shelf in such applications, given that our models were trained on the HotPotQA dataset with the goal of matching that data distribution. Real-world applications built on our work should be re-trained using a training dataset that is relevant to the task at hand.

Moreover, while our system is not tuned for any specific real-world application, our methods could be used in sensitive contexts such as legal or healthcare settings, and it is essential that any such applications undertake extensive quality-assurance and robustness testing, as our system is not designed to meet stringent robustness requirements (e.g., for not stating false facts or meeting legal requirements). More generally, in any language generation setting, there is the possibility of (potentially harmful) social biases that can be introduced in training data. Again, as we did not specifically control or regularize our model to remove the possibility of such biases, we would urge downstream users to undertake the necessary quality-assurance testing to evaluate the extent to which such biases might be present and impacting their trained system and to make modifications to their training data and procedures accordingly.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *International Conference on Learning Representations*.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019. Multi-hop question answering via reasoning chains. *arXiv preprint arXiv:1910.02610*.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *Eighth International Conference on Learning Representations (ICLR)*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial. *JMIR Ment Health*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*.

Dirk Groeneveld, Tushar Khot, Ashish Sabharwal, et al. 2020. A simple yet strong pipeline for HotpotQA. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

William L Hamilton. 2020. *Graph Representation Learning*. Morgan & Claypool.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Divyansh Kaushik and Zachary C. Lipton. 2018. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. 2019. Improving neural question generation using answer separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 2015 International Conference for Learning Representations*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Vishwajeet Kumar, Raktim Chaki, Sai Teja Talluri, Ganesh Ramakrishnan, Yuan-Fang Li, and Gholamreza Haffari. 2019. Question generation from paragraphs: A tale of two hierarchical models. *arXiv preprint arXiv:1911.03407*.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Liangming Pan, Yuxi Xie, Yansong Feng, Tat-Seng Chua, and Min-Yen Kan. 2020. Semantic graphs for generating deep questions. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2021. Do syntax trees help pretrained transformers extract information? In *The 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Burr Settles, Geoffrey T. LaFlair, and Masato Hagiwara. 2020. Machine learning–driven language assessment. *Transactions of the Association for Computational Linguistics*, 8.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Dan Su, Yan Xu, Wenliang Dai, Ziwei Ji, Tiezheng Yu, and Pascale Fung. 2020. Multi-hop question generation with graph convolutional network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

Ming Tu, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Graph sequential network for reasoning over sequences. *arXiv preprint arXiv:2004.02001*.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2019. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jianxing Yu, Xiaojun Quan, Qinliang Su, and Jian Yin. 2020. Generating multi-hop reasoning questions to improve machine reading comprehension. In *Proceedings of The Web Conference 2020*, WWW '20.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *The Eighth International Conference on Learning Representations (ICLR 2020)*.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2018. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing*. Springer International Publishing.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

QG task	Words	Entities	Predicates
Standard (SQuAD)	10.22	1.12	1.75
Multi-Hop (HotpotQA)	15.58	2.34	2.07

Table 6: Comparison of questions' properties in standard and multi-hop QG datasets. We show the average number of words, entities, and predicates per question.

A Standard vs Multi-Hop QG

In this section, we present our results to illustrate the relative complexity of standard and multi-hop QG tasks. For this analysis, we compare three properties of expected output *i.e.* questions: total words, named entities, and predicates, as we believe these represent the *sufficient statistics* of the question. As a benchmark dataset of standard QG, we use the development set from SQuAD (Rajpurkar et al., 2018) and for multi-hop QG, we use the development set from HotpotQA. We extract named entities using spaCy[5] and predicates using Open IE (Stanovsky et al., 2018).

From the results in Table 6, we see that multi-hop questions are almost 1.5 times longer than standard ones and also contain twice the number of entities. These results suggest that in multi-hop QG the decoder needs to generate longer sequences containing more entity-specific information making it considerably more challenging than standard QG. We also observe that multi-hop questions contain roughly 2 predicates in 15 words while standard questions contain 1.75 predicates in 10 words—highlighting that there are fewer predicates per word in multi-hop questions compared with standard ones. This highlights that information is more densely packed within the multi-hop question as they are not expected to contain latent (or bridge) entity information.

B Transformer Model

In this section, we will describe the self-attention and feed-forward sublayers of the widely-used Transformer model (Vaswani et al., 2017).

Self-attention sublayer The self-attention sublayer performs *dot-product self-attention*. Let the input to the sublayer be token embeddings $x = (x_1, \ldots, x_T)$ and the output be $z = (z_1, \ldots, z_T)$, where $x_i, z_i \in \mathbb{R}^d$. First, the input is linearly transformed to obtain key ($k_i = x_i \boldsymbol{W}_K$), value ($v_i = x_i \boldsymbol{W}_V$), and query ($q_i = x_i \boldsymbol{W}_Q$) vectors.

[5] https://spacy.io

Next, interaction scores (s_{ij}) between query and key vectors are computed by performing a dot-product operation

$$s_{ij} = \frac{1}{\sqrt{d}} q_i k_j^T.$$

Then, attention coefficients (α_{ij}) are computed by applying softmax function over these interaction scores

$$\alpha_{ij} = \frac{\exp s_{ij}}{\sum_{l=1}^{T} \exp s_{il}}.$$

Finally, self-attention embeddings (z_i) are computed by the weighted combination of attention coefficients with value vectors followed by a linear transformation

$$z_i = (\sum_{j=1}^{T} \alpha_{ij} v_j) \boldsymbol{W}_F.$$

Feed-forward sublayer To enable the model to have higher representational capacity, we further apply *position-wise non-linear transformations* to the token embeddings. In the feed-forward sublayer, we pass as input the embeddings of all the tokens to a two-layer MLP with ReLU activation.

$$h_i = \max(0,\ z_i \boldsymbol{W}_{\boldsymbol{L_1}} + b_1) \boldsymbol{W}_{\boldsymbol{L_2}} + b_2,$$

where $\boldsymbol{W}_{\boldsymbol{L_1}} \in \mathbb{R}^{d \times d'}$, $\boldsymbol{W}_{\boldsymbol{L_2}} \in \mathbb{R}^{d' \times d}$. These embeddings ($h_i$) are given as input to the next layer.

In the above descriptions, all the weight matrices (denoted by $\boldsymbol{W}_*$) and biases (denoted by b_*) are trainable parameters. To ease optimization during the training process, we apply layer normalization (Ba et al., 2016) to the input and residual connections (He et al., 2016) to the output of each sublayer.

C Graph-Augmented Transformer Encoder (GATE)

We will now discuss how we augment the transformer by including explicit graph-structure extracted from the input context. In addition to the document-level structure such as paragraphs and sentences, the context also contains structural information contained in *entities* and *relations* among them. A popular approach in the multi-hop setting is to use graph neural networks (GNNs) to encode this structural information (Pan et al., 2020). In this work, we leverage the graph-structure information by introducing augmentations to the transformer

architecture—as in our preliminary experiments we found it to outperform other graph-to-sequence alternatives. We refer to this approach as the graph-augmented transformer encoder (GATE).

C.1 Graph Representation of Documents

To extract graph structure from the input context, we consider three types of nodes—*named-entity mentions*, *coreferent-entities*, and *sentence-ids*—and we extract a *multi-relational graph* with three types of relations over these nodes (Figure 4). First, we extract named entities present in the context and introduce edges between them.[6] Next, we extract coreferent words in a document and connect them with edges.[7] Finally, we introduce edges between all sentence nodes in the context. As entities comprise the nodes of this graph, we refer to it as "*context-entity graph*".

C.2 GATE Sublayers

We leverage the context-entity graph by defining two new types of transformer sublayers: a *graph-attention sublayer* and a *fused-attention sublayer*. These two sublayers are intended to be used in sequence with each other and in conjunction with the usual self-attention and fully-connected sublayers of a transformer.

Graph-attention sublayer The graph-attention sublayer performs *relational dot-product graph-attention*. The input to this sublayer are node embeddings[8] from the context-entity graph $v = (v_1, \ldots, v_N)$. Here, we aggregate information from the connected nodes instead of all the tokens. First, interaction scores ($\tilde{s}_{ij}$) are computed for all the edges by performing dot-product on the adjacent projected nodes embeddings

$$\tilde{s}_{ij} = (v_i \widetilde{\boldsymbol{W}}_Q)(\tilde{v}_j \widetilde{\boldsymbol{W}}_K + \gamma_{ij})^\top.$$

In this step, we additionally account for the relation between the two nodes by learning embeddings ($\gamma \in \mathbb{R}^d$) for each relation type (Shaw et al., 2018), where γ_{ij} denotes the relation type between nodes i and j. Next, we compute attention score ($\tilde{\alpha}_{ij}$) for each node by applying softmax over the interaction scores from all its connecting edges

$$\tilde{\alpha}_{ij} = \frac{\exp(\tilde{s}_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(\tilde{s}_{ik})},$$

where $\mathcal{N}_i$ refers to the set of nodes connected to the i^{th} node. Graph-attention embeddings ($\tilde{z}_i$) are computed by the aggregation of attention scores followed by a linear transformation

$$\tilde{z}_i = \left(\sum_{j \in \mathcal{N}_i} \tilde{\alpha}_{ij} (\tilde{v}_j \widetilde{\boldsymbol{W}}_V + \gamma_{ij}) \right) \widetilde{\boldsymbol{W}}_F.$$

Fused-attention sublayer After running both the graph-attention sublayer described above as well as the standard self-attention sublayer, context tokens which belong to the vertex set of context-entity graph have two embeddings: z_i from self-attention and $\tilde{z}_i$ from graph-attention. To effectively integrate information from sequence- and graph-views, we concatenate these two embeddings and apply a parametric function f, an MLP with ReLU non-linearity (Glorot et al., 2011), which we term as the *fused-attention sublayer*

$$z_i = f([z_i, \tilde{z}_i]\boldsymbol{W}_M + b),$$

where $z_i \in \mathbb{R}^d$.

D Training Details

We mostly follow the model training details as outlined in (Sachan and Neubig, 2018), which we also describe here for convenience. The word embedding layer is initialized according to the Gaussian distribution $\mathcal{N}(0, d^{-1/2})$, while other model parameters are initialized using *LeCun uniform initialization* (LeCun et al., 1998). For optimization, we use Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.997$, $\epsilon = 1e^{-9}$. The learning rate is scheduled as: $2d^{-0.5}\min\left(step^{-0.5}, step \cdot 16000^{-1.5}\right)$. During training, the mini-batch contains $12,000$ source and target tokens. For regularization, we use label smoothing (with $\epsilon = 0.1$) (Pereyra et al., 2017) and apply dropout (with $p = 0.1$) (Srivastava et al., 2014) to the word embeddings, attention coefficients, ReLU activation, and to the output of each sublayer before the residual connection. For decoding, we use beam search with width 5 and length normalization following (Wu et al., 2016) with $\alpha = 1$. We also use $\lambda = 0.5$ when performing joint NLL and SSC training.

[6] We use the English NER model provided by the spaCy toolkit, which was trained on OntoNotes-5.0 and covers 18 classes.

[7] We use the coreference resolution model trained on OntoNotes-5.0 hosted in spaCy Universe.

[8] Node embeddings are obtained from the entity's token embeddings.

E Human Evaluation Details

We also assess the generation quality of our multi-hop QG model with human evaluation. To put the human evaluation scores in perspective, we also do human evaluation of the recent best performing model DP-Graph proposed in (Pan et al., 2020)) and ground truth data. For human evaluation, from each system, we randomly selected 100 predicted samples on the HotpotQA test set. We recruited eight human annotators to assign ratings for a subset of examples from each system such that each example gets two sets of ratings. Annotators were asked to assign ratings from 1-5 (inclusive) according to the following four criteria.

Grammaticality Does the question have proper English syntax? In other words, is it a well-formed English question? Note that a question can be grammatically correct but nonsensical (e.g., "Where did the spoon take off?"). Scale: 1: complete nonsense, 5: perfect grammar.

Semantic Correctness Does the question make sense semantically? In other words, is the question meaningful and interpretable? Note that a question can be semantically meaningful but have grammar or syntax errors (e.g., "Where has the girl going?"). Scale: 1: complete nonsense, 5: perfectly understandable.

Answerability Is the question answerable based on the given context: Scale: 1: not at all, 5: the answer is unambiguous.

Question Complexity Does the question require reasoning about multiple sentences and entities in the context in order to find the answer? Scale: 1: trivial to answer, 5: requires non-trivial reasoning across multiple sentences.

We report the average of ratings for each system and each criterion in Table 4.

F BERTScore Evaluation

We evaluate the performance using automated metrics computed from BERT language model (Devlin et al., 2018), whose use is increasingly becoming more common as its results have shown to be correlated with human judgments. Specifically, we use BERTSCORE tool (Zhang et al., 2020),[9] to evaluate the generation quality. From the results in Table 7, we see that our model improves over the previous best model by 2 F_1 points.

[9] https://github.com/Tiiiger/bert_score

Model	P	R	F_1
DP-Graph†	87.66	87.70	87.65
TE_{NLL} w/o data weighting	87.09	88.91	87.96
$Ensemble_{TE+GATE}$	**89.66**	**89.47**	**89.55**

Table 7: Results of multi-hop QG on HotpotQA computed using BERTSCORE tool (Zhang et al., 2020). † indicates that DP-Graph is from (Pan et al., 2020).

G Reproducibility Checklist

G.1 For all reported experimental results

- *A clear description of the mathematical setting, algorithm, and/or model*: This is provided in the main paper in §2.
- *A link to a downloadable source code, with specification of all dependencies, including external libraries (recommended for camera ready, though welcome for initial submission)*: We will open-source the code at a later date.
- *A description of computing infrastructure used*: We run experiments on a machine with these specifications: number of CPUs: 6, CPU RAM: 50GB, GPU model: RTX8000, GPU architecture and memory: turing/48GB, Arch: x86_64, and Disk size: 3.6TB.
- *The average runtime for each model or algorithm, or estimated energy cost*: The average runtime of our TE model was within 10 hours while that of the GATE model was around 15 hours.
- *The number of parameters in each model*: Our model parameters are in the range of 30M-40M.
- *Corresponding validation performance for each reported test result*: Validation set performance is currently not reported in the main paper, as the validation set is non-standard. The HotpotQA dataset does not provide its full test set data, so the convention is to use the validation set as the test set. Therefore, for validation, we select 500 examples from the training set. We include some of these details in the main paper as well in §3.1.
- *A clear definition of the specific evaluation measure or statistics used to report results*: We report results using standard evaluation

metrics that are widely used for evaluation in generation tasks: BLEU, ROUGE-L, and METEOR. We provide the URLs of their code implementations:
BLEU: https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/bin/t2t_bleu.py
ROUGE-L: https://github.com/google-research/google-research/tree/master/rouge
METEOR: https://www.cs.cmu.edu/~alavie/METEOR/README.html

G.2 For all results involving multiple experiments, such as hyperparameter search

- *The exact number of training and evaluation runs* For each experiments, we train the models until convergence, which generally around 30 epochs in our case. We evaluate the performance of the model after each epoch and save the best checkpoint according to BLEU score performance on the validation set.

- *Hyperparameter configurations for best-performing models*: We use most of the code and hyperparameter settings from the transformer model as described in (Sachan and Neubig, 2018). The optimizer and training hyperparameters are also listed in Appendix D. We also provide model hyperparameters in §3.2.

- *The bounds for each hyperparameter*: As described in Appendix D, our model and training setting uses standard hyperparameters such as different dropouts $\in [0, 1)$, weighting or smoothing parameters, $\lambda, \alpha, \epsilon \in [0, 1]$, and optimizer settings such as learning rate $\in [1e-3, 1e-5]$. The model hyperparameter includes model dimensions $d \in \{512, 768\}$, number of layers $\in \{1, 2, 3, 4\}$.

- *The method of choosing hyperparameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy)*: We performed manual hyperparameter tuning. We tuned the source and target sides words within a minibatch as transformer models are sensitive to these. We also performed tuning of the number of warmup steps for the Adam optimizer. We selected the best hyperparameter using the BLEU score on the validation set.

- *Summary statistics of the results (e.g. mean, variance, error bars, etc.)*: The reported results on the performance of our GATE and TE models are the mean of 5 experimental runs. Currently, we don't provide the variance or error bars for these these runs.

G.3 For all datasets used

- *Details of train/validation/test splits*: We use the standard training split provided by HotpotQA dataset creators. As the official test set is blind, we use the validation set as the test set. For validation, we constructed a validation set from 500 examples from the training set.

- *Relevant statistics such as number of examples and label distributions*: We provide dataset statistics details in §3.1. As our task is a generation task, label distribution is not applicable.

- *An explanation of any data that were excluded, and all pre-processing steps*: We include pre-processing details in the main paper in §3.1. We also include the data pre-processing code with the submission for reproducibility.

- *For natural language data, the name of the language(s)*: Our datasets are in English language.

- *A link to a downloadable version of the dataset or simulation environment*: HotpotQA dataset is open-source and is available at: https://hotpotqa.github.io/. We have included the pre-processing code with the submission.

- *For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control*: This is not applicable to this work.