Generating and Answering Simple and Complex Questions from Text and from Knowledge Graphs

Kelvin Han and **Claire Gardent**
CNRS/LORIA and Université de Lorraine
{huiyuan.han,claire.gardent}@loria.fr

Abstract

While both text and Knowledge Graphs (KG) may be used to answer a question, most current Question Answering and Generation models only work on a single modality. In this paper, we introduce a multi-task model such that questions can be generated and answered from both KG and text. The model has wide coverage and handles both simple (one KG fact) and complex (more than one KG fact) questions. Extensive internal, cross-modal and external consistency checks, and analysis of the quality of the generated questions, show that our approach outperforms previous work. Our data and modeling also leads to improvements in downstream tasks, including better performance with fine-tuning Open-Domain QA architectures and better correlation with human judgments than the Data-QuestEval metric which was previously proposed for evaluating the semantic adequacy of KG-to-Text generations.

1 Introduction

Previous work on question generation (QG) and question-answering (QA) mainly focused on a single modality such as Natural Language (NL) (Lyu et al., 2021), Knowledge Graphs (KG) (Hu et al., 2019) or images (Shah et al., 2019). Although QG and QA should be able to operate consistently across semantically equivalent sources regardless of their modality, previous work is hampered by the lack of large-scale aligned cross-modal QA-QG data that also ensures wide QG coverage. As argued in (Rebuffel et al., 2021), such cross-modal QG-QA models can also be used to assess semantic consistency between KG and Text in KG-to-Text generation (i.e. Do questions on the generated text yield the same answer on the input graph and vice-versa?). Finally, cross-modal KG/NL models are key for interacting with KGs in natural language.

Building on datasets pairing KG graphs with text, we develop a multitask, KG/NL model (QTT) that, given a text in English language or a subgraph from the Wikidata KG, can generate and answer simple and complex questions across the two modalities.

We evaluate the model in terms of QG coverage, internal, cross-modal and external QA consistency. We also examine the quality of the generated questions using human evaluation. The results show that our approach outperforms previous work across the board. We further demonstrate that our approach also brings improvements to two downstream tasks namely, better performance with fine-tuning Open-Domain QA architectures and better correlations with human judgments when used for the Data-QuestEval metric (Rebuffel et al., 2021). Our code, data and pretrained models are available at https://gitlab.inria.fr/hankelvin/quartet_qgqa.

2 Related Work

QG and QA from KG. Early rules- or template-based KGQG approaches (Olney et al., 2012; Seyler et al., 2015; Song and Zhao, 2016) required significant human effort and faced issues with generalisation, making them difficult to deploy at scale. While neural KGQG models such as (Reddy et al., 2017; Serban et al., 2016; Elsahar et al., 2018; Liu et al., 2019; Han et al., 2022) address some of the limitations, they — and KGQA ones such as (Bordes et al., 2015; Wu et al., 2019; Huang et al., 2019) — mainly focused on the generation and answering of simple questions i.e., questions which verbalise a single KG fact. More recently, some researchers have started to address complex KGQG and KGQA i.e. questions on more than one KG fact (Kumar et al., 2019; Zhang et al., 2022; Saha et al., 2018; Christmann et al., 2019; Lecorvé et al., 2022; Perez-Beltrachini et al., 2023). However most of these efforts are focused on generating and answering questions from Knowledge Graphs only.

QG and QA from NL. A large body of work utilises the SQuAD dataset (Rajpurkar et al., 2016)

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 285–304
November 1–4, 2023.

to develop neural models for text-based joint QA and QG (Wang et al., 2017; Duan et al., 2017; Lyu et al., 2021; Luo et al., 2022). Some work uses retrieval and conditions a pretrained language model on both the query and the retrieved documents to generate the answer (Lewis et al., 2020; Guu et al., 2020; Khattab et al., 2021). Other work has investigated the use of synthetically-generated data with round trip filtering techniques and shown improved QA performance (Alberti et al., 2019; Puri et al., 2020; Kwiatkowski et al., 2019). Similarly, we used data augmentation and round trip filtering to improve generalisation; however, we do this for QG and QA for both text and KG.

Cross-Modal text/graph QG and QA. An early direction (Fader et al., 2014; Das et al., 2017) for leveraging both structured (KG, tables, lists etc) and unstructured (text) information used information extraction methods such as OpenIE (Banko et al., 2007) and UniversalSchema (Yao et al., 2012) to fill the coverage gaps of KGs so as to employ semantic parsing- or rules-based KGQA methods.

More recent work instead casts structured information as text to access their knowledge through TextQA methods. (Agarwal et al., 2021) verbalise a large KG Wikidata (Vrandečić and Krötzsch, 2014) to add to a retrieval LM corpus, obtaining performance improvements on benchmark QA datasets. (Oguz et al., 2022) obtain improvements by adding Wikipedia tables and lists to the data mix.

Similar to our work, (Rebuffel et al., 2021) created synthetic multimodal-QA/QG datasets, by using a QG model trained on SQuAD to generate questions on texts paired with graphs in existing datasets. They use this and SQuAD to train multimodal models and show that the models can be used to evaluate KG-to-Text generation models; they report better correlations between their measure, the Data-QuestEval metric, with human judgments of semantic adequacy than existing automatic metrics. They however do not provide a systematic evaluation of their models; in contrast, we provide a detailed evaluation of our model and compare it against theirs. We also examine how using our model instead impacts the Data-QuestEval metric's correlations with human judgments.

3 Method

Our approach comprise the creation of graph-text aligned QG-QA datasets covering simple and complex questions (Section 4); and multimodal, multi-task training of a generative QG-QA model (Section 5) – we call this model QTT for the number (4, or **Q**uar**T**e**T**) of its main fine-tuning tasks.

WebNLG Data	
Graph	$[A]$ ⟨Akita Museum of Art, floor count, 3⟩ $[B]$ ⟨Akita Museum of Art, opening date, 2013-09-28⟩ $[C]$ ⟨Akita Museum of Art, address, 1-4-2 Nakadori⟩ $[D]$ ⟨Akita Museum of Art, floor area, 3746.66 (sqm)⟩
Text	$[E]$ The Akita Museum of Art at 142 Nakadori has 3 floors with a total area of 3746.66 square metres and was inaugurated on 28th September 2013.
QTT Data	
X = graph a_X = g ent	$[B]$, $[C]$ Akita Museum of Art
Complex Questions	{What museum opened in 2013-09-28 in Nakadori? What is the name of the museum that opened in 2013-09-28 in Nakadori?}
X = graph a_X = g ent	$[B]$ 2013-09-28
Simple Questions	{In what year was the Akita Museum of Art opened? Which year was the Akita Museum of Art opened?}
X = text a_X = t span	$[E]$ {The Akita Museum of Art}
Complex Questions	{What is the name of the museum that has 3 floors with a total area of 3746.66 square metres?}
X = text a_X = t span	$[E]$ {2013}
Simple Questions	{What year was the Akita Museum of Art inaugurated? Which year was the museum inaugurated?}

Table 1: QTT-DATA instances derived from WebNLG Data. Enclosed letters refer to the triple/text above.

Terminology and Notation. We use the term *graphs* (denoted by g) to refer to subgraphs of the Wikidata KG (Vrandečić and Krötzsch, 2014) and *texts* (t) to refer to English texts. A KG graph is a set of triples (also called facts) of the form ⟨subject, predicate, object⟩. We write X to denote the (text or graph) context of a question and X' to denote its semantically equivalent counterpart in the other modality; g' is a subgraph of g that corresponds to a question q and its answer; nf is the number of facts related to a given q (i.e. the size of its corresponding subgraph $|g'|$); $\overrightarrow{q}$ is a list of NL questions; and a_X is an answer in X whereby a graph answer a_g is either a subject or an object entity in g whereas a text answer a_t is a span in t. Table 1 contains examples of these from the QTT-DATA (Section 4) dataset we created; further examples for Q-KELM (Section 4) can be found in Table 9 in the Appendix.

4 Data

To train our QTT model, we derive a dataset of $(g, a_g, t, a_t, g', nf, q)$ tuples from two existing KG/NL datasets, KELM and WEBNLG.

▪ KELM (Agarwal et al., 2021) is a dataset of 15M (Wikidata graph, English text) pairs where texts were generated from Wikidata graphs using a T5 (Raffel et al., 2020) pre-trained model fine-tuned on TekGen, a large dataset of (g, t) pairs created using distant supervision. We use a subset of KELM filtered for (g, t) pairs where g has between 2 and 5 triples (as larger sizes lead to unnatural questions).[1]

▪ WEBNLG (Gardent et al., 2017a) is comprised of 38,872 (g, t) pairs where the graphs are from Wikidata[2] and the texts were crowdsourced to match the graphs.

We derive our training data from KELM and WEBNLG in three steps. First, we create $(g, a_g, t, a_t, g', nf, q)$ tuples by applying text-based QG and QA on the texts and heuristically aligning text answers with the corresponding graph answers – we call the resulting datasets Q-KELM and Q-WEBNLG^0. Second, we use Q-KELM to train two general multimodal QG models. Thirdly, we apply the models to WEBNLG and add to Q-WEBNLG^0, thereby extending the coverage of the data for training QTT. Figure 1 illustrates the process, and we describe the three steps below.

▪ **Associating Graphs and Texts with Questions and Answers (Step 1).** Given (g, t), an instance of any aligned KG-to-Text dataset, we create synthetic Multimodal QG-QA Data by: (i) generating a question q from t using text-based QG; (ii) extracting the text answer a_t using QA to obtain (t, a_t, q);[3] and heuristically aligning a_t with the corresponding graph answer a_g. To improve quality, we filter out any questions that QA found unanswerable, or whose text answer cannot be aligned with a graph entity. We also heuristically align each generated question with the matching subgraph $g' \subseteq g$ and label it with its size nf i.e., the number of facts each question denotes. Appendix C.1 contains the implementation details for these steps.

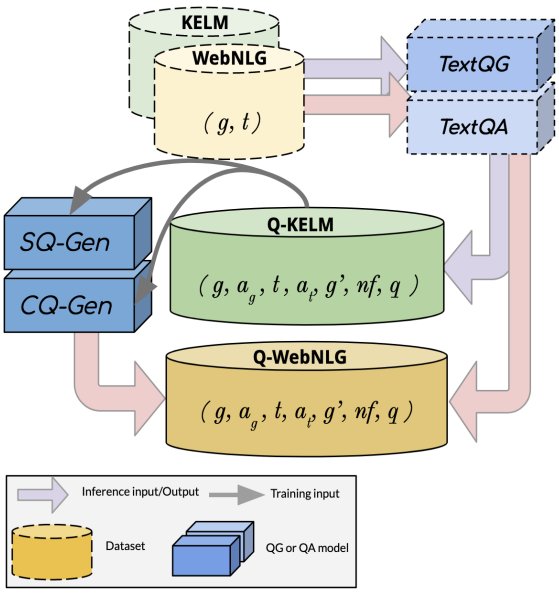

Figure 1: **Procedure for generating Q-WebNLG.**

The size information permits distinguishing between simple (SQ) and complex (CQ) questions which allows us to take a differentiated approach in Step 2 and generate more QA-QG data.

Our filtering above comes however at the cost of question coverage – when our full data generation procedure is applied to WEBNLG, only 2,044 CQs remain (Table 2). Nonetheless, we keep these (as stated earlier, we call this set Q-WEBNLG^0) to have as wide coverage as possible.[4] Applying the procedure on KELM, we get much larger sets of SQs and CQs, which we call SQ-KELM and CQ-KELM, and which together form Q-KELM. Some examples of Q-KELM instances are in Table 9 in Appendix A. Table 2 shows the sizes of the resulting datasets.

# Facts	Q-KELM	Q-WEBNLG^0	QTT-DATA
1	544,464	–	19,467
2	341,082	1,858	61,346
3	234,170	175	25,170
4	22,607	11	13,918
5	7,031	-	-
TOTAL	1,149,354	2,044	119,901

Table 2: **Data Statistics.** Number of questions in the QA datasets; nf: the size of the question (no. of facts). Training multimodal QG models on Q-KELM and applying them to WEBNLG drastically enlarges the Q-WEBNLG^0 training data.

[1] We also filtered out the KELM (g, t) pairs that have: (i) properties not found in the Wikidata SPARQL endpoint or have a functional nature, e.g. containing terms such as 'identifier', 'image of', which tend to have superfluous t in KELM; and (ii) t with low fidelity to their g, using a contrastive loss-trained similarity measure for RDF graph-text pairs.

[2] In WEBNLG, the graphs are from the DBpedia KG. Here we use a version where some of the DBPedia graphs have been mapped to Wikidata (Han et al., 2022), or else removed of underscores and camelcase to align with the Wikidata format.

[3] In our work, we used `t5-base-e2e-qg`, a T5-base QG model fine-tuned on SQuAD 1.0 data and the deepset RoBERTA-based QA model fine-tuned on SQuAD 2.0.

[4] This was not necessary for SQs since SQ-GEN in our next step (3) generates SQs across varying q_{type} and facts.

▪ **Training QG Models on Q-KELM (Step 2)** We use Q-KELM to train two multimodal QG models, one (SQ-GEN) fine-tuned on SQ-KELM for simple questions, and another (CQ-GEN) tuned on CQ-KELM for complex ones. Both are based on the T5-base public checkpoint, and are able to generate questions from text and from graph.

Using a set of textual prompts (see Table 11 in Appendix C.1) at the start of the input sequence to train the CQ-GEN model allows us to controllably generate a complex question from input of the form (X, a_X, a_{type}, nf). Here, a_{type} is the semantic type of the answer entity detected by `ELQ` (Li et al., 2020) or `Duckling`; it is retrieved from the 2021-12-29 Wikidata RDF dump (for entities) or the prediction from `Duckling` (for values). a_{type} is added to improve QG. SQ-GEN is similar to CQ-GEN except that the question type (q_{type}) is used in the input to increase the number and variety of the generated questions.

▪ **Extending Q-WebNLG^0 (Step 3)** By applying the controllable QG models from Step 2 to WEBNLG, we can extend Q-WEBNLG^0 from Step 1. This gives us the final training data for QTT and we call it QTT-DATA.

Our QG models (CQ-GEN, SQ-GEN) generate a question given a context and an answer. Hence, a set of answers must first be selected from the context. For a given X in WEBNLG, we use the same answer selection method as (Rebuffel et al., 2021). For g, the set of possible graph answers is comprised of the subjects and objects in g. For t, it is the set of named entities (NEs) and noun phrases (NPs) detected in t using the `spacy` package.

For SQs from graphs, we follow (Han et al., 2022)'s work on SQ generation from RDF triples and use their q_{type} prediction model, which returns the set of plausible q_{type} for an answer given its position in the triple and its semantic type.

Finally, we add an answerability+consistency filter on the generated questions by posing them to two QA models[5] and keep only questions where both QA models return an answer which (i) has a confidence score $\geqslant 0.7$, and (ii) shares at least a token overlap with the other model's answer and with the answer used to condition QG.

In sum, by iterating over possible answers, nf from 1 to 4, and q_{type} for SQ-GEN, our controllable approach to QG drastically increases the number of generated questions. A breakdown of QTT-DATA's composition is in Table 2.

[5]The deepset QA above and one based on DeBERTaV3.

	Text		Graph	
nf	avg/min/max	# Qs	avg/min/max	# Qs
1	2.9 / 1 / 7	10,205	2.9 / 1 / 10	9,262
2	2.2 / 1 / 9	52,517	1.6 / 1 / 11	8,829
3	1.5 / 1 / 6	17,734	1.9 / 1 / 17	7,436
4	1.4 / 1 / 6	8,841	1.9 / 1 / 21	5,077

Table 3: **QTT DATA**. Average, minimum and maximum number of questions for text and graph inputs of size nf (the size is the number of facts matched by the question)

5 QTT, a multimodal QG-QA Model

Our model (QTT) is trained in a multi-task manner to handle both QA and QG. It is based on the T5-small checkpoint (60.5M parameters), allowing for direct comparison with (Rebuffel et al., 2021). We fine-tune on QTT-DATA using four main and four auxiliary tasks, all of which are cast in a sequence-to-sequence manner. Using a single Nvidia A40 GPU, it takes approximately 20 hours to fine-tune QTT. The four main and four auxiliary tasks are:

▪ ***QG from text/graph*** Given (X, a_X, nf), generate a set of questions $\overrightarrow{q}$. We obtain this set by first gathering together questions in QTT-DATA that were generated from a given context X, and which share the same size nf and answer, and then adding to these the questions generated from other "smaller" pieces of contexts (whose information is fully contained in X), and also sharing the same attributes (nf and answer). This gathering process is detailed in Appendix D.1.1.

▪ ***QA from text/graph*** Given (q, X), generate an answer $\hat{a}_X$. We leverage sets of (X, a_X, q) from QTT-DATA for training these tasks. Additionally, to maximise the use of the data for training QA, we also associate questions answerable by a text t to larger pieces of text that semantically contain t (details in Appendix D.1.2). To allow QTT to abstain from an answer if the question cannot be answered from the context, we use two strategies (details in Appendix D.1.4) to generate negative unanswerable $(q, \neg X)$ pairs.

▪ ***KG-to-Text/Text-to-KG*** These auxiliary tasks consist in either verbalising a graph or deriving a graph from a text. We instantiate each of the WEBNLG graph-text pairs as training instances.

▪ ***Entity typing on graph/text*** These auxiliary tasks combine entity detection and typing. Given a context X, the task identifies the entities/values mentioned in X, and their semantic types. We use `BLINK` (Wu et al., 2020), `Duckling` and Wikidata to obtain the target information for the tasks.

As the number of instances vary across QTT-DATA, we upsampled between modalities and tasks to balance them. Details of each task, and upsampling, can be found in Appendices D.2 and D.1.5. Also, when the input is a graph, we linearise it using the same format for data-to-text tasks in the GEM benchmark (Gehrmann et al., 2021). During training, we use cross-entropy loss as the objective. At inference, we generate with greedy search.

6 Experiments

We compare QTT to the original models (DQE, hereafter) used in the Data-QuestEval metric (Rebuffel et al., 2021) in terms of QG coverage, QA accuracy and consistency as well as performance in two downstream tasks. In the following, we describe DQE, our evaluation data and methodology.

Baseline: the DQE models DQE comprises four T5-small (Raffel et al., 2020) models fine-tuned for QG-QA from graph and text. For text, their QA model (`DQE-TextQA`) was fine-tuned on SQuAD 2.0 and the QG model (`DQE-TextQG`) on SQuAD 1.0. For graphs, both their QG (`DQE-KGQG`) and QA models (`DQE-KGQA`) were fine-tuned on a synthetic QG dataset of (g, a_g, q) triples created by applying `DQE-TextQG` to a (g, t) corpus.

Evaluation data We reserved the test part of WEBNLG, which comprise 1,779 (graph, text) instances, for evaluation. The parallel (g, t) data here ensures that a question can be answered using graph or text, allowing us to check the models' cross-modal consistency (Section 6). We apply both DQE and our model to the test set and generate questions for every graph and text.

Evaluating QG Coverage We compare the coverage of QTT against DQE by measuring the number of unique questions they each generated on the WEBNLG test set. We also compare the semantic coverage of the questions using BERTScore (BSc) (Zhang et al., 2020), by taking one model's question for a given entry as prediction and the other's generated questions for the same entry as multi-references. This is repeated with both approaches swapped. The intuition is that if approach A scores higher with approach B's questions as references than vice-versa, A's questions are "contained" in B's and conversely, B has wider semantic coverage.

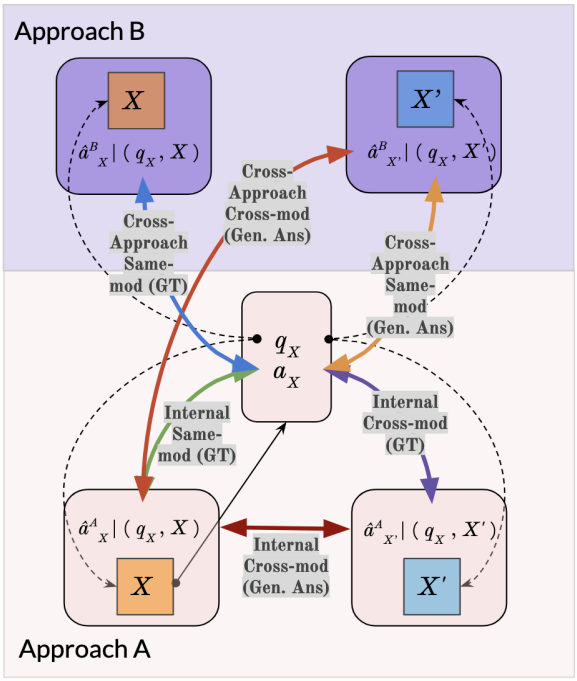

Figure 2: **QA Accuracy**. Bold lines denote QA comparisons within/between modalities and/or approaches. Dotted arrows indicate the context X or X' that the question (q_X) is posed against to obtain the answers.

Evaluating QA Consistency In what follows, we refer to a_X, the answer used to condition the generation of q_X, as the ground truth (GT) or the reference answer. We use $\hat{a}_X$ to denote a generated answer. For a given question, $\hat{a}_X$ is the answer derived from modality X and $\hat{a}_{X'}$ is from modality X'. We use superscripts (e.g. $\hat{a}^A_X$ and $\hat{a}^B_X$) to distinguish answers generated by different models for the same question q and input context X.

Accounting for the various ways in which a question can be answered (i.e. $a_X, \hat{a}_X, \hat{a}_{X'}$), we evaluate the quality of multimodal QG-QA models by computing three consistency metrics.[6] *Internal Same-mod (GT)* compares the generated answers $\hat{a}_X$ against the reference answers a_X, indicating the approach's self-consistency. *Internal X-mod (GT)* compares the ground truth a_X with the answer derived from the other modality $\hat{a}_{X'}$. Finally, *Internal X-mod (Gen Ans)* compares $\hat{a}_{X'}$ and $\hat{a}_X$, the answers derived from each modality.

[6] QTT and DQE generates differing numbers of questions for a given X; to ensure a fair evaluation, when an Approach A generates more questions for X, we randomly sample from its set as many questions that Approach B generates for X.

We also investigate QA across approaches, allowing an external indication of each's QG-QA capabilities. Here, we examine the two answers that can be generated by approach B ($\hat{a}^B_X$ and $\hat{a}^B_{X'}$) when given q^A_X, a question generated by A. We do this on three levels: (i) *X-Appr Same-mod (GT)*, by comparing $\hat{a}^B_X$ against a^A_X on the same modality that q^A_X came from; (ii) *X-Appr X-mod (GT)* comparing B's generated answer with the reference answer across modalities (i.e. $\hat{a}^B_{X'}$ vs a^A_X); and (iii) *X-Appr X-mod (Gen Ans)* where $\hat{a}^B_{X'}$ is compared against $\hat{a}^A_X$. A graphical overview of these QA consistency comparisons can be found in Figure 2.

Downstream: QA with FiD For another external verification of QTT's (and DQE's) QG, we conducted experiments with Fusion-in-Decoder (FiD) (Izacard and Grave, 2021). We use a checkpoint that was trained on TriviaQA (Joshi et al., 2017) as the questions there are factual in nature, i.e. compatible with the texts in WebNLG. To investigate the quality of QTT-DATA, we also fine-tune the same FiD checkpoint using either QTT-DATA or DQE's training data and use these for QA.[7] Similar to *X-Appr Same-mod (GT)* above, we compare $\hat{a}^B_X$ against a^A_X, except that B in this case is a given fine-tuned (or not) FiD QA model while A is DQE or QTT. Though such a setting gives upper-bound FiD scores,[8] the differences in scores – when varying fine-tuning data and QG – independently validates QTT's QG vs DQE's and QTT-DATA too.

Downstream: Data-QuestEval metric Since DQE was originally used in the Data-QuestEval metric, we also compare the correlation of the resulting Data-QuestEval metric with human judgments when DQE is replaced with QTT. For this, we compute the correlations with the judgments collected on 2,007 outputs from 9 participating systems in the WebNLG Challenge (Gardent et al., 2017b). Following (Rebuffel et al., 2021), we compute Pearson's r , but also report Spearman's ρ.[9]

Evaluation settings The following describes and explains the settings for our evaluations.

▪ ***Answer selection for QG inference*** To have a direct comparison with (Rebuffel et al., 2021)'s models, we follow their use of the `spacy` pipeline for answer selection on t (see Section 4), and report results with this setting in Sections 6 and 7. However this approach is noisy and often segments NEs with nouns or NPs in them (e.g. 'English' being extracted from 'English Without Tears'), leading to ill-formed questions. We trained an answer selection model for texts (see Appendix D.2) using QTT-DATA, ensuring that the answers for QG are meaningful spans in t. We conducted our ablation experiments and human evaluation (Sections 8 and 9) using this answer selection method for text.

▪ ***Automatic metric*** Following (Rebuffel et al., 2021), we use BERTScore (BSc) for evaluation, to address the restrictiveness of token F1 for cross-modality QA. We use the same settings, except the following for a clearer analysis: (i) BScs were rescaled against the official BSc baseline for a wider spread, (ii) "unanswerable" strings were set to an empty string to avoid non-zero BSc for these and "over-counting" them, and (iii) lowercasing.

▪ ***Self-consistency filter*** Since both QTT and DQE are trained on synthetic data, some generated questions may be ill-formed and pose an impact on QA. We therefore filter from both QTT and DQE the questions: (i) which cannot be answered from their source context; or (ii) whose generated answer $\hat{a}_X$ has a BSc < 0.7 when compared against the reference a_X. We focus our analysis for QA Consistency and the Downstream Evaluations on the results after filtering as this is the upper bound of the approaches' performance; the impact of removing this filter is included in our ablations (Section 8). For congruence with our QG Coverage analysis, if the filtering will leave a given approach A – and therefore B as well – with no QA pairs (i.e. no coverage), we keep one QA pair for A.

We note also that the self-consistency filter above is important in the downstream Data-QuestEval evaluation (see above) – given how the Data-QuestEval metric is computed (i.e. the generated answer is compared against the GT answer), if a generated question cannot be answered by the source context and yet still posed to the other modality, it will not capture the factuality comparison accurately (i.e. it will skew the metric and affect its reliability).

[7] Here, when the context is a graph, we use the linearisation scheme from (Oguz et al., 2022) to utilise FiD for KGQA.

[8] i.e. the information to answer the question is in a single document and this gold document is being provided to FiD.

[9] The latter may be appropriate since the system outputs for the WebNLG 2017 challenge evaluation were selected to cover a spread of automatic scores and are therefore unlikely to be normally distributed.

7 Results

QG Coverage QTT generates three times as many questions as DQE when the input is a text (14,141 vs 42,959) and 10 times more when it is a graph (7,272 vs 75,906). Figure 3 provides a fine-grained view of the QG coverage by the (graph-based) size of the context.

This higher coverage results from three modeling choices differentiating QTT from DQE: (i) QTT is trained to generate multiple questions from a given X; (ii) the enlarged size and coverage of QTT's training data by applying Q-KELM-trained QG models to WEBNLG; and (iii) the use of q_{type} controls in SQ-GEN, permitting multiple SQs of various types to be generated from a single X.

The BScs are also higher with QTT's questions as references (89.7 vs 85.3 for text; 90.4 vs 82.5 for graph), suggesting that QTT is not just generating more questions but also ones that "contains" DQE's as well as semantically different ones from DQE's.

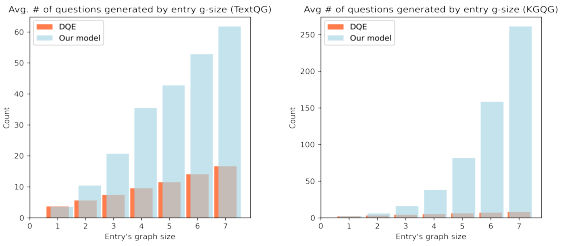

Figure 3: **Comparative QG coverage DQE vs. QTT.** Total and average number of generated questions is much higher for QTT across both modality and input size. The delta increases with the size of the input.

QA Consistency Table 4 summarises the comparisons between QTT and DQE on QA accuracy. A finer-grained analysis of QTT's Internal performance across question complexity can be found in Table 10 in Appendix B.

▪ ***QTT outperforms DQE on self-consistency*** Despite QTT and DQE both starting fine-tuning from the same T5-small checkpoint, QTT gains over DQE in *Internal Same-mod (GT)* (+8.0 BSc for text, +3.8 for graph). This shows that using QTT-DATA – which provides aligned wide-coverage QA-QG data – for training in a multi-modal multi-task manner enables QG and QA with greater internal roundtrip consistency.

▪ ***Our synthetic in-domain data improves performance*** QTT's self-consistency gains over DQE for text also stems from our procedure for creating in-domain data. `DQE-TextQG` and `DQE-TextQA` were fine-tuned on SQuAD only, leading to a drop in scores when `DQE-TextQA` is applied on WEBNLG (vs `DQE-KGQA`'s 95.0). This out-of-domain effect also shows when it answers a question QTT generated with text (95.4 to 56.8, Table 4); whereas, in the reverse case, the drop for QTT when it answers a DQE question generated is much less (87.4 to 81.4).

	Internal		X-Appr	
QG: / **QA:**	DQE / DQE	QTT / QTT	DQE / QTT	QTT / DQE
	Same-mod (GT)			
T → T	$87.4_{(\pm 0.01)}$	$\mathbf{95.4}_{(\pm 0.14)}$	$81.4^{(\mathbf{-6.0})}_{(\pm 0.02)}$	$56.8^{(-38.6)}_{(\pm 0.31)}$
G → G	$95.0_{(\pm 0.01)}$	$\mathbf{98.8}_{(\pm 0.04)}$	$76.2^{(\mathbf{-18.8})}_{(\pm 0.05)}$	$79.4^{(-19.4)}_{(\pm 0.47)}$
	X-mod (GT)			
G → T	$51.4_{(\pm 0.04)}$	$\mathbf{75.8}_{(\pm 0.35)}$	$60.3^{(\mathbf{+8.9})}_{(\pm 0.04)}$	$48.2^{(-27.6)}_{(\pm 0.52)}$
T → G	$60.6_{(\pm 0.02)}$	$\mathbf{69.7}_{(\pm 0.35)}$	$59.8^{(\mathbf{-0.8})}_{(\pm 0.02)}$	$51.6^{(-18.1)}_{(\pm 0.27)}$
	X-mod (Gen Ans)			
G → T	$51.8_{(\pm 0.04)}$	$\mathbf{76.4}_{(\pm 0.35)}$	$58.2^{(\mathbf{+6.4})}_{(\pm 0.04)}$	$48.4^{(-28.0)}_{(\pm 0.51)}$
T → G	$63.9_{(\pm 0.01)}$	$\mathbf{72.1}_{(\pm 0.38)}$	$61.2^{(\mathbf{-2.7})}_{(\pm 0.02)}$	$53.1^{(-19.0)}_{(\pm 0.28)}$

Table 4: **Consistency Results** Avg. of BScs between answers. In $_{\text{subscripts}}$ are std. dev. across 5 random runs; $^{\text{superscripts}}$ are the difference between X-Appr and Internal. X/Y indicates the QG/QA model used. QTT betters DQE on all consistency tests and for all modalities.

▪ ***Q-DATA with multi-task training improves cross-modality performance*** QTT outperforms DQE by at least 9.1 BSc (e.g. 69.7 vs 60.6 for T → G) in *Internal X-mod (GT)* showing that it can more accurately answer questions cross-modally with respect to the reference answer, and/or generate questions that allow this. This is beneficial when using the QG-QA model(s) for evaluation such as the Data-QuestEval metric where this comparison ($\hat{a}_{X'}$ and a_X) is relied on to assess the semantic concordance between the data input and generated text. Furthermore, a low discrepancy in the *Internal X-mod (Gen Ans)* performances between the cross-modal directions (i.e. $G \rightarrow T$ vs $T \rightarrow G$) is ideal under our parallel data evaluation setting, as it shows that the QG-QA model(s) is able to reflect the agreement between the (g, t) pairs. QTT's 4.3 BSc gap here narrows by nearly two-thirds the 12.1 BSc gap faced by DQE (i.e. 76.4-72.1 vs 51.8-63.9).

▪ QTT***'s performance is consistent across question complexity and externally validated*** We find that QTT also performs consistently for all nf, (the number of facts q relates to) for text and for graph (Table 10 in Appendix B). Our findings above on QTT's internal performance also hold when examined cross-approach (*X-Appr*, i.e. external); whenever QTT is used to answer questions generated by DQE, the drop in QA accuracy is significantly lower (or in some cases a gain) than vice-versa.

Downstream Evaluations QTT also outperforms in two downstream tasks. Our FiD experiments (Table 5) show that the QTT's questions can be answered with higher accuracy than DQE's for both modalities – likely because QTT's training data permits it to generate more CQs than DQE, which is closer to the forms in TriviaQA. The combination of fine-tuning FiD with QTT-DATA and QTT QG also betters every other combination with DQE and/or its training data. These validate (i) our procedure for creating QTT-DATA, and (ii) the use of it with multimodal multi-task modeling for improved QG. Besides that, using QTT for computing the Data-QuestEval metric also boosts by >10 points the score's correlation (Spearman's ρ) with human judgments of semantic adequacy (Table 6). This is likely due to the improved QG (quality and coverage) for both modalities, allowing QA-based evaluation to more accurately assess the information content of the data and the generated text.

	Fusion-in-Decoder					
QG: **QA:**	DQE FiD^0	DQE FiD^D	DQE FiD^Q	QTT FiD^0	QTT FiD^D	QTT FiD^Q
	Same-mod (GT)					
T → T	68.94 (0.01)	86.24 (0.04)	86.60 (0.02)	77.06 (0.42)	88.89 (0.26)	**91.48** (0.18)
G → G	74.25 (0.03)	91.07 (0.02)	88.66 (0.05)	82.14 (0.33)	91.10 (0.27)	**94.99** (0.17)

Table 5: **Consistency Results with FiD** (Similar to Table 4.) FiD^0 is the public FiD checkpoint trained on TriviaQA. FiD^D/FiD^Q denotes that checkpoint fine-tuned on training data from DQE/QTT for that modality.

Measure	DQE	QTT
Spearman's ρ	47.9 (1.47e-104)	**58.6** (4.81e-168)
Pearson's r	51.8 (2.69e-125)	**61.8** (1.24e-191)

Table 6: **Correlations with human judgments.** Comparing when DQE and QTT are used in the computation of the Data-QuestEval metric. All (p-values) « 0.001.

8 Ablation

We also studied the impact of four variations to the data and modeling: (i) *X-Filt* removes the question self-consistency filter (Section 6); (ii) *Data* uses a similar data setting as (Rebuffel et al., 2021) i.e. synthetic QG-QA data on WEBNLG plus SQuAD, without Q-KELM and Steps 2 & 3 in Section 4; (iii) *SPO* uses another graph linearisation, where each (s, p, o) in g is shown as they appear; and (iv) *X-Aux*, removes all auxiliary tasks. All these models were trained for 383,445 steps. These ablation results can be found in Table 7.

	QTT	X-Filt	Data	SPO	X-Aux
	Same-mod (GT)				
T → T	**97.6**	76.6	95.7	97.4	97.4
G → G	98.8	86.0	**99.5**	97.8	98.9
	X-mod (GT)				
G → T	**75.5**	70.3	71.4	75.3	75.4
T → G	**77.7**	63.8	64.4	74.9	75.8
	X-mod (Gen Ans)				
G → T	76.0	**76.3**	71.7	76.0	76.0
T → G	**78.3**	72.3	66.6	75.5	76.7

Table 7: **Ablation results.** All models here use the text answer selector. Comp. denotes consistency comparison, Mod. denotes QG and QA modalities respectively.

The ablations show that, other than the question self-consistency filter and using QTT-DATA, the other variations have relatively limited impact on QTT's *Same-mod (GT)*. It also shows that using our data generation procedure leads to improvements in QA; especially in cross-modal settings (*X-mod (GT)* and *X-mod (Gen Ans)*).

9 Human evaluation

Modality	Consistency	Naturalness	Complexity
Text	0.76 (0.53)	0.77 (0.45)	0.75 (0.72)
Graph	0.64 (0.56)	0.67 (0.55)	0.93 (0.86)

Table 8: **Human evaluation for QG.** Each score is the average of all annotators' ratings. Values in brackets are the Fleiss' kappa coefficient for that particular aspect.

We conducted a human evaluation on the quality of QTT's questions since there are no reference questions. To have a broad study, we sampled 10 questions each from bins combining these characteristics: (i) modality - whether QG from graph or text; (ii) complexity - the question's size (nf); and (iii) QA accuracy, where the questions were separated into three quantiles based on their *Internal*

Same-mod (GT) score.[10]

Three doctoral candidates in NLP fluent in English, were shown the 240 questions and their contexts, and asked to rate each question on three aspects with binary choice: whether it is (i) consistent with the context; (ii) natural-sounding; and (iii) a CQ. For the last aspect, we report whether their rating matches the control (nf=1 or nf>1) used for generation. The results can be found in Table 8.

The overall agreement between the annotators is substantial (Fleiss' kappa: 0.65 for KGQG; 0.61 for TextQG). For TextQG, QTT scores $\geqslant 0.75$ on all aspects. In KGQG, we observe lower scores for *Consistency* and *Naturalness*. This is likely due to the increased challenge when generating from graph (which is under-specified and requires a semantic gap to be overcome), and is compounded by unseen properties in the WEBNLG test set. There is also a difference in the performance for *Complexity* between the graph and text modalities; we found that this can be attributed to the challenge of accurately judging KG facts in text.[11]

10 Conclusion

We propose an approach (QTT), which we show generates more questions that cover more information compared to previous work (DQE). Unlike existing approaches, our data and architecture allows us to generate multiple questions for a given input. Extensive internal, cross-modal and external checks show that QTT outperforms DQE on QA consistency. The quality of our generated questions was verified by human evaluation for semantic consistency, naturalness and adherence to our complexity controls. Finally, the use of our approach also leads to improvements against DQE in two downstream evaluations (QA with Fusion-in-Decoder and the Data-QuestEval metric).

Our main contributions are (i) a large multimodal general QG-QA dataset (Q-KELM) which will be made available on publication, (ii) a data generation procedure including two general multimodal QG models enabling controllable generation of in-domain synthetic QG-QA datasets, and (iii) a multimodal multi-task QG-QA model that can generate and answer questions from text and from graph.

11 Acknowledgments

We thank the anonymous reviewers for their feedback. This research was supported by ANR Project QUANTUM (Project-ANR-19-CE23-0025). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

[10] e.g. 1 bin is {G, nf=1, 1st quant.} i.e. SQs from graph, with BSc for $\hat{a}_g$ vs a_g in top 33% of all SQs from graph.

[11] For e.g. the question "Who was born in Los Angeles in California?" generated from the text "The birthplace of X is Los Angeles in California." corresponds to the single KG fact $\langle$ X; born in; Los Angeles, California $\rangle$ but may be judged as being a CQ of more than 1 fact (e.g. $\langle$ X; born in; Los Angeles > and < Los Angeles; located in; California $\rangle$).

12 Limitations

Our data generation process relies on the KELM dataset (Agarwal et al., 2021), which was generated using a single pretrained language model (T5-large). As such, KELM — and hence the texts in our generated data (Q-KELM) too — reflects only the set of factual or linguistics characteristics in this model. For instance, there is certainly more than one way a KG graph (or set of facts) can be lexicalised in text, however KELM only contains one lexicalisation for each of the KG subgraph within it. In addition, we only use a single QG model for generating our initial sets of synthetic QA pairs (Q-KELM, Q-WEBNLG^0). Although by using Q-KELM to train our two general QG models, we may have introduced new varieties of questions into QTT-DATA, it is unlikely we have obtained the full range of questions possible for a given context-answer pair. This limitation of KELM may however be alleviated by for example, ensembling the KELM dataset with generations from different LMs (following fine-tuning or with in-context learning) on the same subgraphs used to generate the KELM sentences/passages.

Secondly, so as to ensure QG and QA fidelity, we made sure to exclude questions generated from text in Step 3 of the data creation process (Section 4) when constructing the KGQG and KGQA instances for QTT-DATA (Section 5). This is because the answers for questions generated from text may not be constrained to a single KG entity. As a result of this, the sets of complex questions for these two tasks are smaller than for their text counterparts[12]. This impacts QTT's capabilities for generating multiple complex questions for a single input instance in the KGQG task[13]. This limitation could potentially be alleviated by using beam search (or variants of it such as diverse beam search (Vijayakumar et al., 2016) and constrained beam search (Post and Vilar, 2018)) to increase the generation of complex questions from graph using CQ-GEN.

Finally, although we used the agreement of two state-of-the-art QA models when checking for QG acceptability, we cannot be certain that questions rejected by the QA models are not actually valid questions — in such cases, it means that the coverage of QA pairs in our datasets is constrained.

[12]On the other hand, given the parallel nature of WEBNLG, the answer for a question generated from graph can be found in the text, allowing us to include such questions when constructing the TextQG and TextQA instances for QTT-DATA.

[13]The input context here is $g' \in g$, where $2 \leqslant \mid g \mid \leqslant 4$

13 Ethics Statement

As advancements in generative technologies accelerate in terms of capabilities, scale and public access, so too must the need for the the ability to understand if such machine-generated information are reliable.

We believe that our KG/NL-aligned QG-QA data creation method and cross-modal QG-QA model has the potential to contribute positively in the following areas: (i) a QG-QA model with cross-modal consistency can aid in tasks that includes but are not limited to automated fact verification, KG-to-text/text-to-KG quality estimation and knowledge graph completion; and (ii) the ability to generate in-domain QA data can help improve downstream QA performance and dialogue systems to aid human-machine interactions with KGs. On the other hand, a direct application of our method and model for a task such as fact verification could lead to a failure to capture misinformation, which have the potential for substantive societal harm. This risk arises because KELM is based on a snapshot of the Wikidata KG from circa 2019. Additionally, the T5 pretrained language models used in producing KELM, and also in all our models, were trained with data up to 2020. These constrain the extent and validity of facts in our model up to these points in time.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3554–3565, Online. Association for Computational Linguistics.

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, page 2670–2676,

San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. 2019. Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion. In Proceedings of the ACM International Conference on Information and Knowledge Management, pages 729–738. Association for Computing Machinery.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 358–365, Vancouver, Canada. Association for Computational Linguistics.

Kaustubh D. Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Srivastava, Samson Tan, Tongshuang Wu, Jascha Sohl-Dickstein, Jinho D. Choi, Eduard Hovy, Ondrej Dusek, Sebastian Ruder, Sajant Anand, Nagender Aneja, Rabin Banjade, Lisa Barthe, Hanna Behnke, Ian Berlot-Attwell, Connor Boyle, Caroline Brun, Marco Antonio Sobrevilla Cabezudo, Samuel Cahyawijaya, Emile Chapuis, Wanxiang Che, Mukund Choudhary, Christian Clauss, Pierre Colombo, Filip Cornell, Gautier Dagan, Mayukh Das, Tanay Dixit, Thomas Dopierre, Paul-Alexis Dray, Suchitra Dubey, Tatiana Ekeinhor, Marco Di Giovanni, Rishabh Gupta, Rishabh Gupta, Louanes Hamla, Sang Han, Fabrice Harel-Canada, Antoine Honore, Ishan Jindal, Przemyslaw K. Joniak, Denis Kleyko, Venelin Kovatchev, Kalpesh Krishna, Ashutosh Kumar, Stefan Langer, Seungjae Ryan Lee, Corey James Levinson, Hualou Liang, Kaizhao Liang, Zhexiong Liu, Andrey Lukyanenko, Vukosi Marivate, Gerard de Melo, Simon Meoni, Maxime Meyer, Afnan Mir, Nafise Sadat Moosavi, Niklas Muennighoff, Timothy Sum Hon Mun, Kenton Murray, Marcin Namysl, Maria Obedkova, Priti Oli, Nivranshu Pasricha, Jan Pfister, Richard Plant, Vinay Prabhu, Vasile Pais, Libo Qin, Shahab Raji, Pawan Kumar Rajpoot, Vikas Raunak, Roy Rinberg, Nicolas Roberts, Juan Diego Rodriguez, Claude Roux, Vasconcellos P. H. S., Ananya B. Sai, Robin M. Schmidt, Thomas Scialom, Tshephisho Sefara, Saqib N. Shamsi, Xudong Shen, Haoyue Shi, Yiwen Shi, Anna Shvets, Nick Siegel, Damien Sileo, Jamie Simon, Chandan Singh, Roman Sitelew, Priyank Soni, Taylor Sorensen, William Soto, Aman Srivastava, KV Aditya Srivatsa, Tony Sun, Mukund Varma T, A Tabassum, Fiona Anting Tan, Ryan Teehan, Mo Tiwari, Marie Tolkiehn, Athena Wang, Zijian Wang, Gloria Wang, Zijie J. Wang, Fuxuan Wei, Bryan Wilie, Genta Indra Winata, Xinyi Wu, Witold Wydmański, Tianbao Xie, Usama Yaseen, M. Yee, Jing Zhang, and Yue Zhang. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. 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), pages 218–228, New Orleans, Louisiana. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, page 1156–1165, New York, NY, USA. Association for Computing Machinery.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In Proceedings of the 10th International Conference on Natural Language Generation, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc,

Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM benchmark: Natural language generation, its evaluation and metrics. In Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021), pages 96–120, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3929–3938. PMLR.

Kelvin Han, Thiago Castro Ferreira, and Claire Gardent. 2022. Generating questions from Wikidata triples. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, pages 277–290, Marseille, France. European Language Resources Association.

Sen Hu, Lei Zou, and Zhanxing Zhu. 2019. How question generation can help question answering over knowledge base. In Natural Language Processing and Chinese Computing, pages 80–92. Springer International Publishing.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 874–880, Online. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Relevance-guided supervision for OpenQA with ColBERT. Transactions of the Association for Computational Linguistics, 9:929–944.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In International Semantic Web Conference, pages 382–398. Springer.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466.

Gwénolé Lecorvé, Morgan Veyret, Quentin Brabant, and Lina M. Rojas Barahona. 2022. SPARQL-to-text question generation for knowledge-based conversational applications. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 131–147, Online only. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In Advances in Neural Information Processing Systems, volume 33, pages 9459–9474. Curran Associates, Inc.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6433–6441, Online. Association for Computational Linguistics.

Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie, and Jun Zhao. 2019. Generating questions for knowledge bases via incorporating diversified contexts and answer-aware loss. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2431–2441, Hong Kong, China. Association for Computational Linguistics.

Hongyin Luo, Shang-Wen Li, Mingye Gao, Seunghak Yu, and James Glass. 2022. Cooperative self-training of machine reading comprehension. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 244–257, Seattle, United States. Association for Computational Linguistics.

Chenyang Lyu, Lifeng Shang, Yvette Graham, Jennifer Foster, Xin Jiang, and Qun Liu. 2021. Improving unsupervised question answering via summarization-informed question generation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4134–4148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. UniK-QA: Unified representations of structured and unstructured knowledge for open-domain question answering. In Findings of the Association for Computational Linguistics: NAACL 2022, pages 1535–1546, Seattle, United States. Association for Computational Linguistics.

Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. Dialogue & Discourse, 3(2):75–99.

Laura Perez-Beltrachini, Parag Jain, Emilio Monti, and Mirella Lapata. 2023. Semantic parsing for conversational question answering over knowledge graphs. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2507–2522, Dubrovnik, Croatia. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5811–5826, Online. Association for Computational Linguistics.

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, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Clement Rebuffel, Thomas Scialom, Laure Soulier, Benjamin Piwowarski, Sylvain Lamprier, Jacopo Staiano, Geoffrey Scoutheeten, and Patrick Gallinari. 2021. Data-QuestEval: A referenceless metric for data-to-text semantic evaluation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8029–8036, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. 2017. 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, pages 376–385, Valencia, Spain. Association for Computational Linguistics.

Amrita Saha, Vardaan Pahuja, Mitesh Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 588–598, Berlin, Germany. Association for Computational Linguistics.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In Proceedings of the 24th International Conference on World Wide Web, pages 113–114.

Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. 2019. Cycle-consistency for robust visual question answering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6649–6658.

Linfeng Song and Lin Zhao. 2016. Question generation from a knowledge base with web exploration. arXiv preprint arXiv:1610.03807.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods, 17(3):261–272.

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

Tong Wang, Xingdi (Eric) Yuan, and Adam Trischler. 2017. A joint model for question answering and question generation. In Learning to generate natural language workshop, ICML 2017.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In Proceedings of the 2020 Conference on Empirical

Methods in Natural Language Processing (EMNLP), pages 6397–6407, Online. Association for Computational Linguistics.

Peng Wu, Shujian Huang, Rongxiang Weng, Zaixiang Zheng, Jianbing Zhang, Xiaohui Yan, and Jiajun Chen. 2019. Learning representation mapping for relation detection in knowledge base question answering. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6130–6139, Florence, Italy. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Probabilistic databases of universal schema. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), pages 116–121, Montréal, Canada. Association for Computational Linguistics.

Kun Zhang, Yunqi Qiu, Yuanzhuo Wang, Long Bai, Wei Li, Xuhui Jiang, Huawei Shen, and Xueqi Cheng. 2022. Meta-CQG: A meta-learning framework for complex question generation over knowledge bases. In Proceedings of the 29th International Conference on Computational Linguistics, pages 6105–6114, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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.

A Q-KELM examples

Table 9 shows examples of instances from Q-KELM.

B Detailed results

Finer-grained analysis of QTT QA performance Table 10 provides a finer-grained view of QTT's same-mod and cross-mod QA consistency performance; it is evaluated for QA performance on the set of questions relating to varying number of facts ($1 \leqslant nf \leqslant 4$).

C Implementation details: CQ-Gen and SQ-Gen

C.1 Training data

Q-KELM is used as the training data for the CQ-GEN and SQ-GEN models (respectively using CQ-KELM and SQ-KELM from Q-KELM). Each training instance in CQ-GEN and SQ-GEN is assembled in the manner described in the following paragraphs; examples for them can be found in Table 11 and 12.

Context X When generating from text t, the entire t is used as input for both CQ-GEN and SQ-GEN. For generation from KG subgraph, the input to both models is g', a subgraph of g of size nf where $1 \leqslant nf \leqslant 4$. For SQ-GEN, the size of g' is always 1; for CQ-GEN, it is $2 \leqslant nf \leqslant 4$.

Answer a_X For generation from text, a_t the text answer that was extracted by the RoBERTa QA model (see Section 4) is used. When generating from KG subgraph, the graph answer a_g (the entity/value in g that is aligned with a_t (see below) is used.

▪ ***Aligning*** a_g The graph answer a_g is the entity in g which either exactly matches, contains or else has the smallest edit distance to a_t. If a_t cannot be matched to a graph entity, the (q, a_t) is rejected, together with those unanswerable by the QA model.

Answer semantic type a_{type} The answer entity semantic type a_{type} is used for QG when generating from text as well as from graph. It is obtained by using the graph entity (a_g) aligned with the answer by querying Wikidata for the set of entities/values that a_g has the 'instance of' and/or 'subclass of' property with.

KELM		
Instance	**Description**	**Example**
Graph g	▪ g: a Wikidata graph	⟨JNR Class DE15, subclass of, diesel-hydraulic locomotive⟩, ⟨JNR Class DE15, instance of, locomotive class⟩, ⟨JNR Class DE15, manufacturer, Kawasaki Heavy Industries⟩, ⟨JNR Class DE15, service entry, 00 1967⟩
Text t	▪ t: an English text generated from g	The JNR Class DE15 is a diesel-hydraulic locomotive class made by Kawasaki Heavy Industries, which entered service in 1967.
Q-KELM		
Instance	**Description**	**Example**
Text (CQ) (t, q, a_t)	▪ q: generated from t using Text QG (Step 1) ▪ a_t: answer derived from (t, q) using Text QA (Step 1)	What is the name of the diesel-hydraulic locomotive class made by Kawasaki Heavy Industries? JNR Class DE15
Text (SQ) (t, q, a_t)	▪ q: generated from t using Text QG (Step 1) ▪ a_t: answer derived from (t, q) using Text QA (Step 1)	When did the JNR Class DE15 enter service? 1967
Graph (CQ) (g', q, a_g)	▪ $g' \subseteq g$: a subgraph of g heuristically aligned with q ▪ q: generated from t using Text QG (Step 1) ▪ a_g: graph answer i.e. entity in g' heuristically aligned with a_t ▪	⟨JNR Class DE15, subclass of, diesel-hydraulic locomotive⟩, ⟨JNR Class DE15, instance of, locomotive class⟩, ⟨JNR Class DE15, manufacturer, Kawasaki Heavy Industries⟩ What is the name of the diesel-hydraulic locomotive class made by Kawasaki Heavy Industries? JNR Class DE15
Graph (SQ) $(g'^{\lvert 1\rvert}, q, a_g)$	▪ $g'^{\lvert 1\rvert} \in g$: a subgraph of g, of size one ▪ q: generated from t using Text QG (Step 1) ▪ a_g: graph answer i.e. entity in $g'^{\lvert 1\rvert}$ heuristically aligned with a_t	⟨JNR Class DE15, service entry, 00 1967⟩ When did the JNR Class DE15 enter service? 00 1967

Table 9: Q-KELM **Dataset.** For each (g, t) pairs in the filtered version of KELM (see Section 4), QA pairs are created for both t and $g' \subseteq g$. The question q is generated from t, heuristically aligned with the corresponding subgraph g' and both the text and the graph answer are extracted from t and g' respectively.

Question complexity control nf In CQ-KELM and SQ-KELM each (q_X, a_X) pair is associated with a subgraph g' obtained with heuristic matching (see below). Since g and t are parallel, nf (the size of g') is used as the control for the complexity of the question for generation from both text and graph.

▪ ***Determining*** nf The size (nf) of the question is determined by matching a question and its answer to the corresponding subgraph $g' \subseteq g$ where g' is the set of triples $\langle s, p, o \rangle$ in g such that: either s and o have an overlap of $\geqslant 1$ token with $q + a_t$, and/or they can be detected in $q + a_t$.

Controls A textual prompt is added to both the input and the target to control the question genera-

	Num Facts			
	1	2	3	4
	Same-mod (GT)			
T → T	96.3	98.5	98.3	98.1
G → G	97.8	99.1	99.2	99.3
	X-mod (GT)			
T → G	77.7	77.9	77.7	77.2
G → T	73.2	77.9	75.1	74.6
	X-mod (Gen Ans)			
T → G	79.2	77.9	78.1	77.1
G → T	74.3	78.2	75.5	75.0

Table 10: **Fine-grained analysis of QTT's QA performance (BSc). Num Facts** denote the number of facts (nf) the set of QA-pairs relate to (i.e. 1 denotes an SQ of 1 fact, 2 denotes a CQ of 2 facts etc...). The nf sets are mutually exclusive.

tion; the prompt differs for CQ-GEN and SQ-GEN. SQs are those where q + a_t contain only 2 entities, whereas we define CQs as those with at least 7 tokens and such that q + a_t contain > 2 entities and the size of the matching subgraph is at least 2 (i.e. $nf \geqslant 2$). A set of special tokens (added to the T5 tokeniser vocabulary) are used to demarcate the components in the input and target. These prompts and tokens can be seen in the examples found in Tables 11 and 12.

Question type q_{type} The question type, which is only used in SQ-GEN, is detected using the question type filter in the NL-Augmenter framework (Dhole et al., 2021) [14].

Target The target is a single question for a given input for both CQ-GEN and SQ-GEN.

C.2 Technical details

CQ-GEN and SQ-GEN are each T5-base pre-trained models that were fine-tuned from their public checkpoints. Each of them was tuned for up to 10 epochs with early stopping (on the loss for the validation set for when it stops decreasing, with a patience of 3 epochs). A learning rate of 2e-4 was used, together with a linear warmup ratio on 10% of the total training steps, and an effective batch size of 144. For both models, we used the HuggingFace `transformers` library. We used the Lightning integration of the DeepSpeed framework for efficient training and used bf16 precision together with the DeepSpeedCPUAdam optimizer.

[14] https://github.com/GEM-benchmark/NL-Augmenter

Modality		
Text	Input	generate 1 complex question of 2 facts from text [ANS] 1988 [Sp2] value duration | value distance [INP] Peter Vermes represented the United States, where he began his career in 1988 and ended it in 1997.
	Target	cq 1 2 text[ANS] 1988 [QST] When did Peter Vermes begin representing the United States?
Graph	Input	generate 1 complex question of 3 facts from rdf [ANS] SoundApp [Sp2] software [INP] [SUB] SoundApp [PRP] instance of [OBJ] Software [SUB] SoundApp [PRP] operating system [OBJ] System 7 [SUB] SoundApp [PRP] license [OBJ] Freeware
	Target	cq 1 3 rdf[ANS] SoundApp [QST] What is the name of the freeware software program for the operating system of System 7?

Table 11: **CQ-GEN** Examples of inputs and targets for complex questions training instances, for text and for graph. [ANS], [INP], [QST], [Sp1], and [Sp2] are special tokens we use to demarcate parts of the input/target. [SUB], [PRP] and [OBJ] are used in graph inputs to demarcate subject, property and object elements of a triple.

Modality		
Text	Input	generate 1 simple question of 1 fact from text [ANS] 1977 [QST] when [Sp2] time [INP] The Lasse Viren Finnish Invitational, which was created in 1977, is part of the sport of athletics.
	Target	sq 1 text 1 [ANS] 1977 [QST] When was the Lasse Viren Finnish Invitational created?
Graph	Input	generate 1 simple question of 1 fact from rdf [ANS] Pennsylvania Avenue [QST] where [Sp2] street [INP] [SUB] National Archives Building [PRP] located on street [OBJ] Pennsylvania Avenue
	Target	sq 1 rdf 1 [ANS] Pennsylvania Avenue [QST] Where is the National Archives Building located?

Table 12: **SQ-GEN** Examples of inputs and targets for simple questions training instances for text and for graph. [ANS], [INP], [QST], [Sp1], and [Sp2] are special tokens we use to demarcate parts of the input/target. [SUB], [PRP] and [OBJ] are used in graph inputs to demarcate subject, property and object elements of a triple.

D Implementation details: QTT

D.1 Training data

QTT-DATA was used as the training data for QTT. Examples of the training instances for each task/subtask can be found in the following tables: TextQG and KGQG can be found in Tables 13 and 14, complex and simple TextQA as well as KGQA can be found in Table 15, the KG-to-Text as well as Text-to-KG tasks can be found in Table 16; and the EntType tasks can be found in Table 17.

D.1.1 Obtaining sequence of questions for QTT QG

For questions grounded in text, since each WEBNLG text t is associated with a graph g, we use it to gather the set of questions in QTT-DATA generated from g itself and its sub-graphs (and the corresponding sub-texts of t that is present in WEBNLG). From this set, we gather all nf-sized questions which share an a_t to form the set of questions associated with (t, a_t, nf).

For questions grounded in graphs, two treatments are used to gather questions since the inputs to QTT differs for generation of CQs and SQs (see Appendix D.2). For CQs, given g', which is a subgraph of g (a graph occurring in WEBNLG), we gather all questions with size nf and answer a_g that are associated with g'. For SQs, given a WEBNLG graph g, we gather all questions of size one and answer a_g which can be computed from a triple contained in g.

Finally, the target in the TextQG and KGQG tasks is typically a set of 3 questions drawn from $\overrightarrow{q}$ without replacement. If $|\ \overrightarrow{q}\ | > 3$, $\overrightarrow{q}$ is padded to ensure $|\ \overrightarrow{q}\ |\ \%\ 3 = 0$. If however $|\ \overrightarrow{q}\ | \leqslant 3$ — as it happens in the complex KGQG setting — the sequence of questions in the target is simply $\overrightarrow{q}$. Each set of $\leqslant 3$ questions is then instantiated as a new training instance. This is done in order to train the QG models to generate multiple questions for a given context while still staying within the T5 model's maximum input length.

D.1.2 Deriving and maximising QA data

We derive QA data by creating for each $(X, a_X, \overrightarrow{q})$ tuple in QTT-DATA as many QA training instances of the form (X, a_X, q) as there are questions in $\overrightarrow{q}$.

In addition, if a question q with answer a_t is answered by a text t (resp. graph g by a_g) which is strictly contained within another larger text t^+ (resp. g^+) in WEBNLG, we also associate (q, a_t) with t^+ (resp. (q, a_g) with g^+). Table 3 shows the number of questions associated with the various input and nf size.

D.1.3 Creating EntType instances

The target for the **EntType** auxiliary task is the set of entities and values ($eset$) detected in the input context, and each of the $e \in eset$ are paired with their semantic types.

For text, $eset_t$ is detected by applying the `BLINK` entity linker for text (Wu et al., 2020), and `Duckling` on the text t. For entities, the semantic type information is retrieved from Wikidata (from the RDF dump dated 29 December 2021). Any type that contains the strings "MediaWiki", "Wikimedia" or "disambiguation" are excluded. For values, we use the type information predicted by `Duckling`.

For graph, the set of entities/values ($eset_g$) that are present in g is used. For values in g, such as dates, times, monetary sums etc we leverage the predictions from `Duckling` (that was applied on the t that is associated with g). We identify from the `Duckling` prediction set the one: with the lowest edit distance to the value, which is an alphanumeric string, and has a normalised edit distance < 0.5 (if there is one such).

Finally, we exclude the following from the training instances: (i) when an answer semantic type for an entity cannot be found; and (ii) when the difference in the number of entity/values sets between the g and its t is more than 2. The latter is so as to avoid semantically similar g and t instances having significantly different targets, which is particularly important since we train in a multi-task setting where all tasks are seen simultaneously.

D.1.4 Negative sampling for QA

For the TextQA and KGQA tasks, negative examples were created so as to allow the model to recognise questions that are unanswerable given the context. These samples are created using a random (i.e. 50-50) assignment to one of these two strategies:

Strategy 1: simple negatives for a given (X, q, a_X) instance are created by picking another (X_{other}, q', a'_X) instance in the QA training set where X and X_{other} do not share any common entities/values between them. If X is a text, we use the graph that it is paired with in WEBNLG to check for common entities. A new instance $(X, q',$ "unanswerable") is then created in the training data.

Strategy 2: hard negatives are created in the following manner: a mapping M is first created ahead of time where every entity e that can be found in the training data is associated with the set of all the (X, q, a_X) instances where e is mentioned in X ($e \in X$). If X is a text, we use the graph it is paired with in WEBNLG when creating M.

For a given (X, q, a_X) instance, another instance, i.e. $(X_{other}, q', a_{X_{other}})$ is randomly chosen using M and a random $e \in X$. If a single token from the answer $a_{X_{other}}$ overlaps with (i) any of the tokens for any e (an entity or value) found in X or (ii) any of the tokens of X, then this $(X_{other}, q', a_{X_{other}})$ candidate is rejected. Otherwise, a new instance $(X, q'$, "unanswerable") is created in the training data using the instance and the process for (X, q, a_X) terminates. When a candidate instance is rejected, a new $(X_{other'}, q', a_{X_{other'}})$ is drawn from M using another $e \in X$. After 10 tries or when all $e \in X$ has been exhausted, a simple negative is created instead.

D.1.5 Upsampling

We carry out upsampling on two levels when preparing QTT-DATA (the training data for QTT).

Between modalities for QG subtasks and between modalities for the same task For QG this is done between complex TextQG and complex KGQG, simple TextQG and simple KGQG to ensure that the QG subtasks are balanced. It is also done between modalities for the QA tasks (e.g. TextQA and KGQA) to ensure that the tasks are balanced between modalities. The negative samples for the QA tasks are also upsampled in the same way.

For instance, m_1 and m_2 are the sets of samples for modality 1 and modality 2 respectively, and suppose $| m_1 | > | m_2 |$.

If $| m_2 | / | m_1 | \geqslant 0.5$, we randomly sample $| m_1 |$ - $| m_2 |$ from m_2 to balance them.

If however $| m_2 | / | m_1 | < 0.5$, we upsample m_2 up to at most $1/3 \cdot | m_1 |$. This is to ensure that we do not overrepresent m_2 in the data and overfit on it during training.

Globally for certain tasks This was done for simple QG as a whole (i.e. after simple TextQG and simple KGQG have been consolidated as one), KG-to-Text, Text-to-KG, and EntType tasks. This is because they are significantly less instantiated samples of these tasks in the data than the rest. The number of samples for each of these tasks were tripled.

D.2 Technical details

QA with FiD We fine-tune with the base version of the publicly released FiD checkpoint[15] that is trained on the TriviaQA dataset (Joshi et al., 2017). Using the training data for either QTT (i.e. QTT-DATA) or DQE (Section 6), we further fine-tune this checkpoint for 15,000 steps to give four different models (FiD_t^D trained on DQE's training data for text; FiD_g^D trained on DQE's synthetic training data for graph; FiD_t^Q and FiD_g^Q trained with QTT-DATA with the appropriate context X used, i.e. text and graph respectively). For evaluation, we use the same set of generated questions (q_X) and reference answers, i.e. the answer used to condition QG (a_X) produced by each of DQE and QTT; this is the same set of questions and reference answers used to compute the results in Table 4

Correlations Following (Rebuffel et al., 2021), both the Spearman's ρ and Pearson's r correlations in this paper were computed with the SciPy python library (Virtanen et al., 2020)

Text answer selector model This is a T5-base model that is fine-tuned on the task of answer selection on text. The input to the model is a text t, and the target is a set of answer spans in QTT-DATA that are in t. The answer spans comprise: (i) the set of all a_t in QTT-DATA for a given t, together with the set of all mention spans obtained from `BLINK`/`Duckling` (which were also used in the **EntType** auxiliary tasks). The answer spans are sorted by the sequence of appearance in t. The model was trained for 10 epochs with early stopping (patience of 3 epochs) on the development set loss. A batch size of 32 and a learning rate of 2e-4 (with a linear warmup of 10% of the training steps) was used.

[15] https://dl.fbaipublicfiles.com/FiD/pretrained_models/tqa_reader_base.tar.gz

Modality		
Complex TextQG	Input	cqg task [Sp1] text 2 [ANS] Aarhus University [INP] The School of Business and Social Sciences at Aarhus University (in Denmark) is affiliated to the European University Association (HQ in Brussels). Denmark's leader is Lars Lokke Rasmussen.
	Target	cqg task [Sp1] text 2 [ANS] Aarhus University [QST] Lars Lokke Rasmussen is the leader of Denmark, which is the home of the School of Business Studies and Social Sciences at what university? [QST] The School of Business and Social Sciences at what university is affiliated to the European University Association? [QST] What is the name of the university in Denmark that is home to the School of Business and Social Sciences?
Complex KGQG	Input	cqg task [Sp1] rdf 3 [ANS] 28.0 (metres) [INP] entity [3Arena], height [28.0 (metres)], located in the administrative territorial entity [North Wall Quay], building type [Concert and events venue] [TEND]
	Target	cqg task [Sp1] rdf 3 [ANS] 28.0 (metres) [QST] What is the height of the concert and events venue in North Wall Quay?

Table 13: **QTT** Examples of inputs and targets for **complex** TextQG and KGQG training instances. [ANS], [INP], [QST], [Sp1], and [TEND] are special tokens we use to demarcate parts of the input/target.

Modality		
Simple TextQG	Input	sqg task [Sp1] text 1 [ANS] 1985 [INP] 200 Public Square, Cleveland, with 45 floors covering 111484 square metres, was completed in 1985
	Target	sqg task [Sp1] text 1 [ANS] 1985 [QST] In what year was 200 Public Square completed? [QST] How many years was 200 Public Square completed? [QST] Which year was 200 Public Square completed?
Simple KGQG	Input	sqg task [Sp1] rdf 1 [ANS] Abraham A. Ribicoff [INP] entity [Abraham A. Ribicoff], spouse [Ruth Ribicoff][TEND]
	Target	sqg task [Sp1] rdf 1[ANS] Abraham A. Ribicoff [QST] What was Ruth Ribicoff's husband's name? [QST] Who was Ruth Ribicoff's husband? [QST] What is the name of the man Ruth Ribicoff married to?

Table 14: **QTT** Examples of inputs and targets for **simple** TextQG and KGQG training instances. [ANS], [INP], [QST], [Sp1], and [TEND] are special tokens we use to demarcate parts of the input/target.

Modality		
Text	Input	textqa task [Sp1] Alaa Abdul Zahra has played for AL Kharaitiyat SC and for what sports club of the Qatar Stars League? [INP] Alaa Abdul Zahra has played for AL Kharaitiyat SC and for Al-Khor Sports Club of the Qatar Stars League. Al Kharaitiyat SC play at Al Khor and are managed by Amar Osim.
	Target	textqa task [Sp1] Al-Khor Sports Club
	Input	textqa task [Sp1] What is the name of the monument that was constructed in 2000 in Adams County, Pennsylvania? [INP] In Soldevanahalli, Acharya Dr. Sarvapalli Radhakrishnan Road, Hessarghatta Main Road, Bangalore – 560090 is the location of the Acharya Institute of Technology in India which is affiliated with the Visvesvaraya Technological University. The motto of the Institute which was established in the year 2000 is "Nurturing Excellence" and there are 700 postgraduate students.
	Target	textqa task [Sp1] unanswerable
Graph	Input	kbqa task [Sp1] In what city in Switzerland is the accademie di architetura di mendresio located? [INP] entity [Accademia di Architettura di Mendrisio], country [Switzerland], number of students [600], established [1996], city [Mendrisio], location [Ticino] [TEND] , entity [Switzerland], leader [Johann Schneider - Ammann] [TEND]
	Target	kbqa task [Sp1] Mendrisio
	Input	kbqa task [Sp1] In what year was the Ataturk Monument opened? [INP] entity [Turkey], leader title [President of Turkey], leader [Ahmet Davutoglu], largest city [Istanbul], currency [Turkish lira] [TEND] , entity [Ataturk Monument (Izmir)], designer [Pietro Canonica], material [Bronze], location [Turkey] [TEND]
	Target	kbqa task [Sp1] unanswerable

Table 15: **QTT** Examples of inputs and targets for TextQA and KGQA training instances. [INP], [Sp1], and [TEND] are special tokens we use to demarcate parts of the input/target.

Modality		
KG-to-Text	Input	data2text task [Sp1] entity [Allama Iqbal International Airport], operating organisation [Pakistan Civil Aviation Authority], location [Punjab, Pakistan], city served [Lahore][TEND] , entity [Pakistan], leader [Mamnoon Hussain] [TEND] , entity [Punjab, Pakistan], country [Pakistan] [TEND]
	Target	data2text task [Sp1] Allama Iqbal International airport is located in Punjab, Pakistan and is operated by The Pakistan Civil Aviation Authority. Lahore city is served by the airport. Mamnoon Hussain is the leader of Pakistan.
Text-to-KG	Input	text2data task [Sp1] A Long Long Way was written in Ireland and published by Penguin Random House (parent company Viking Press).
	Target	text2data task [Sp1] entity [A Long Long Way], country [Ireland], publisher [Viking Press] [TEND] , entity [Viking Press], parent company [Penguin Random House] [TEND]

Table 16: **QTT** Examples of inputs and targets for KG-to-Text and Text-to-KG training instances. [INP], [Sp1], and [TEND] are special tokens we use to demarcate parts of the input/target.

Modality		
EntType_t	Input	entlink task [Sp1] Abner currently plays football for Real Madrid Castilla.
	Target	entlink task [Sp1] [INP] Abner [Sp3] Homo sapiens; person; natural person; omnivore [INP] Real Madrid Castilla [Sp3] football team; sports team
EntType_g	Input	entlink task [Sp1] entity [Buzz Aldrin], place of birth [Glen Ridge], country of citizenship [United States of America], status [Retired] [TEND]
	Target	entlink task [Sp1] [INP] Buzz Aldrin [Sp3] Homo sapiens; person; natural person; omnivore [INP] Glen Ridge [Sp3] borough in the United States; municipality of New Jersey [INP] United States of America [Sp3] state; country; political territorial entity; republic; federation; historical country; state with limited recognition; democracy; nation

Table 17: **QTT** Examples of inputs and targets for EntType (on text and on graph) training instances. [INP], [Sp1], [Sp3], and [TEND] are special tokens we use to demarcate parts of the input/target.