SPARQL-to-Text Question Generation for Knowledge-Based Conversational Applications

Gwénolé Lecorvé **Morgan Veyret** **Quentin Brabant** **Lina M. Rojas-Barahona**
Orange – Lannion, France
{gwenole.lecorve, morgan.veyret, quentin.brabant, linamaria.rojasbarahona}@orange.com

Abstract

This paper focuses on the generation of natural language questions based on SPARQL queries, with an emphasis on conversational use cases (follow-up question-answering). It studies what can be achieved so far based on current deep learning models (namely pretrained T5 and BART models). To do so, 4 knowledge-based QA corpora have been homogenized for the task and a new challenge set is introduced. A first series of experiments analyzes the impact of different training setups, while a second series seeks to understand what is still difficult for these models. The results from automatic metrics and human evaluation show that simple questions and frequent templates of SPARQL queries are usually well processed whereas complex questions and conversational dimensions (coreferences and ellipses) are still difficult to handle. The experimental material is publicly available[1].

1 Introduction

Knowledge-based approaches have recently become popular in the field of question answering (QA) and dialogue, raising the task of semantic parsing that seeks to map a user's input questions to a formal representation that can be queried in a Knowledge Graph (KG). Alternatively, techniques have been proposed to verbalize small KGs, for instance to summarize information to a user. Still, the task which consists in verbalizing formal queries has been less studied. Yet, interesting applications could be derived from SPARQL-to-text question generation: for instance, the generation of tutoring systems where users can exercise on a topic, or the simulation of users for QA or dialogue systems. This is why this paper studies SPARQL-to-text question generation, with a particular consideration attached to the generation of questions in a conversational context.

[1]https://github.com/Orange-OpenSource/sparql-to-text

The objective of the paper is to study what can be achieved so far on SPARQL-to-text question generation using datasets and pretrained models available in the literature. In this regard, the contributions are the following:

1. The **release of 5 knowledge-based QA corpora** (including 2 conversational ones) that have been homogenized and prepared for the SPARQL-to-text task: 4 of them are derived from existing corpora, and the last one is a new challenge set with unseen query types and domains.
2. The **comparison of different fine-tuning approaches** for BART and T5, using different input features and training data. As a results, we show that feeding the model with the expected answer and conversational contexts helps. We also show that these information can be efficiently replaced by a paragraph when available.
3. An **in-depth analysis of the models' performance with respect to varied query types**. This highlights the limits of the current transformer-based approaches, especially to process rare types of queries, and to generate coreferences and ellipses.
4. An **evaluation of the intelligibility and relevance of the generated questions** through quizzes where the participants have to answer follow-up questions based on a short paragraph. The results show that the models are still far from human questions but they can be used for some types of queries.

After a literature review in Section 2, Section 3 and 4 present the datasets and models, respectively. Then, prototyping experiments using different training setups are described in Section 5, while a detailed analysis of the models' performance is given in Section 6.

2 Related Work

Question generation frequently refers to the task of generating a natural language questions based on a text (Zhang et al., 2021). The generation can be conditioned on the manually spotted expected answer in the text (Murakhovs'ka et al., 2022; Laban et al., 2022), whereas generating them in a free way (Duan et al., 2017), even potentially generating possible answers (Tafjord and Clark, 2021).

In the field of knowledge-based approaches, several propositions have been made for the verbalization of formal queries (in SQL, SPARQL, OWL, etc.) through rules or templates (Ngonga Ngomo et al., 2013, 2019; Kusuma et al., 2020), or intermediate representations (Guo et al., 2019; Gan et al., 2021), leading to verbalizations with a variable naturalness. Using neural approaches, several contributions have been made to generate questions from RDF triples (Han et al., 2022) or small KGs depicting multi-hop questions (Serban et al., 2016; Kumar et al., 2019). In (Bi et al., 2020), this principle is improved by enriching the entities from the triples with information from a broader KG. A limit of these approaches is that they cannot cover several features offered by query language like SPARQL (e.g., union of triples, filters, aggregation functions, etc.). Hence, to the best of our knowledge, our work is the first attempt to study the verbalization of SPARQL seeking to generate a large diversity of questions types.

Among other related work, Knowledge-Based QA (KBQA) tasks are interesting to study since they provide data with paired natural language question and formal representation (usually triples or SPARQL queries) (Bordes et al., 2015; Dubey et al., 2019; Kacupaj et al., 2020; Biswas et al., 2021; Kacupaj et al., 2021; Cui et al., 2022). It is important to note that some of these corpora overlap because they are extensions or refinements of common ancestors. Less datasets exist when considering the conversational KBQA: ConvQuestions (Christmann et al., 2019) and CSQA (Saha et al., 2018). While the former does not provide the formal representations associated to the natural language questions, the latter is relevant for our task. Finally, in the field of dialogue, propositions have also raised to enable interoperability with KGs through a formal language (Lam et al., 2022). However, annotated datasets are usually private or small. Hence, the conversational dimension in our SPARQL-to-text task is original.

3 Datasets

In this paper, 4 KBQA corpora from the literature are used: SimpleQuestions (Bordes et al., 2015), LC-QuAD 2.0 (Dubey et al., 2019), ParaQA (Kacupaj et al., 2021), and CSQA (Saha et al., 2018). They have different characteristics, and they do not overlap. Additionnaly, a new corpus is introduced to serve as a challenge set, i.e. no training data is available for it. This corpus has been generated based on the WebNLG v.3.0 corpus (Ferreira et al., 2020), and is referred to as WebNLG-QA. This section presents an overview of the 4 corpora from the literature, the generation process and resulting content of WebNLG-QA, and how all these datasets were homogenized. General statistics and examples for the 5 resulting SPARQL-to-text datasets are given in Table 1 and 2.

3.1 Existing corpora

SimpleQuestions originally does not include SPARQL queries but (subject, property, object) triples. Each triple is paired with a question whose expected answer is either the object or the subject of the triple. Hence, all questions are asking for an entity ("what is...", "which...", "who..."). The triples' elements were initially taken from FreeBase, but were ported to WikiData[2].

LC-QuAD 2.0 and **ParaQA** directly include SPARQL queries for both DBPedia (WikiData as well in LC-QuAD 2.0). Questions are more varied than in SimpleQuestions. Expected answers can be entities, numbers or booleans. Some question are even unanswerable in LC-QuAD 2.0[3]. Questions in LC-QuaD 2.0 are sometimes of poor quality as they were semi-automatically generated, whereas ParaQA's questions are more natural but the dataset is much smaller.

CSQA is a very large corpus of conversational question-answering based on Wikidata. Queries are given in a custom formalism instead of SPARQL. The questions include coreferences and ellipses, potentially with clarification steps when they are ambiguous. CSQA covers a wide range of questions types such as (single or multiple triples, entity/numeric/boolean answers, comparative questions, etc.). Nonetheless, the linguistic diversity of the questions is low and some are unnatural.

[2]https://github.com/askplatypus/wikidata-simplequestions

[3]This means that no answer can be found in the KG, not that the question does make sense. Hence, this should not bother the SPARQL-to-text models.

	SimpleQuestions	LC-QuAD 2.0	ParaQA	CSQA	WebNLG-QA
Questions (train/valid/test)	34K / 5K / 10K	21K / 3K / 6K	3.5K / 500 / 1K	1.5M / 167K / 260K	332
Dialogues (train/valid/test)	–	–	–	152K / 17K / 28K	100
Reference questions per query	1	1	1	1	2
Characters per query	70 ($\pm$ 10)	108 ($\pm$ 36)	103 ($\pm$ 27)	163 ($\pm$ 100)	100 ($\pm$ 33)
Tokens per question	7.4 ($\pm$ 2.1)	10.6 ($\pm$ 3.9)	10.3 ($\pm$ 3.7)	10.0 ($\pm$ 4.1)	8.4 ($\pm$ 4.5)

Table 1: Statistics for each SPARQL-to-text dataset. Standard deviations are given between brackets.

	Query	Question	Answer
SimpleQ.	SELECT DISTINCT **?f** WHERE { **?f** property:**author** resource:**Laura_Ingalls_Wilder** }	what is a book by Laura Ingalls Wilder	A Little House Traveler
LC-Q. 2.0	SELECT (COUNT (**?k**) AS **?y**) { resource:**MiG-21** property:**operator** **?k** }	How many operators does MiG-21 have?	12
ParaQA	SELECT DISTINCT **?m** WHERE { resource:**Alexa_Scimeca** property:**current_partner** **?p** . **?p** property:**former_partner** **?m** }	Who are all the people who used to figure skate with the current partner of Alexa Scimeca?	Brynn Carman, Shawnee Smith, Andrea Poapst
CSQA	SELECT DISTINCT **?x** WHERE { **?x** rdf:**type** ontology:**occupation** . resource:**Edmond_Yernaux** property:**occupation** **?x** }	Which occupation is the profession of Edmond Yernaux ?	politician
	SELECT DISTINCT **?a** WHERE { **?a** property:**main_subject** resource:**politician** . **?a** rdf:**type** ontology:**collectable** }	Which collectable has that occupation as its principal topic ?	Notitia Parliamentaria, An History of the Counties, etc.
WebNLG-QA	SELECT DISTINCT **?y** WHERE { { { resource:**Sludge_metal** property:**instrument** **?y** } UNION { resource:**Post-metal** property:**instrument** **?y** } } }	What is used as an instrument in Sludge Metal or in Post-metal?	Singing, Synthesizer
	SELECT DISTINCT **?e** WHERE { resource:**Sludge_metal** property:**instrument** **?e** }	And what about Sludge Metal in particular?	Singing
	ASK WHERE { resource:**Nord_(Year_of_No_Light_album)** property:**genre** resource:**Sludge_metal** }	Does the Year of No Light album Nord belong to this genre?	Yes

Table 2: Examples for each corpus. For conversational corpora (CSQA and WebNLG-QA), follow-up questions are shown to illustrate the notion of coreference and ellipsis.

3.2 WebNLG-QA (challenge set)

To test the generalization of the models to be trained, a new conversational QA dataset, WebNLG-QA, is proposed for the sole evaluation purpose. This corpus has been generated based on WebNLG v.3.0 (Ferreira et al., 2020), a corpus associating small KGs (1-7 triples) with several possible verbalizations (short texts transcribing the KG's information). This corpus was built in two steps. First, follow-up SPARQL queries were automatically generated for each KG from WebNLG.

The query generation algorithm allows for a wide range of query types and combinations (number of triples, logical connectors, filters, etc.). Especially, it includes mechanisms to favor coreferences and ellipses by reusing entities and triples from the last generated query. Some queries can be unanswerable based on the KG, or even be nonsensical in order to test the genericity of the models. Since the purpose is to probe the limits of the models, the algorithm permanently tries to balance the distribution over each type of queries by prioritizing the rarest ones at each new generation step.

Algorithm 1 details how this is achieved. Considering the set of elementary types $\mathcal{T}$ (line 1), we implemented a function ϕ_t for each query type $t \in \mathcal{T}$. This function reads a source knowledge graph and tries to derive a query of the given type (line 7). Depending on the type, the query can be built either from scratch, or by modifying a baseline query in order to fit the target type[4]. The dependency possibilities are listed in a specific variable (lines. 4 and 10). Furthermore, the function ϕ_t relies on a set of input constraints C, which are implemented as logical predicates on the expected query. Typically, this enables specifying the desired number of common elements (resources, properties, etc.) between the generated query and the previous ones. For instance, the types *coreference* or *ellipsis* expect certain common elements between queries, whereas other types do not (in order to prevent consecutive queries from going around in circles). The creation of an unanswerable query can be constrained such that no answer can be found in G but an answer

[4]For instance, the generation of boolean query is implemented as changing to `ASK` the verb of a `SELECT` query.

```
1: enum $\mathcal{T}$ ← {single_triple, two_triples, ..., true, false,
       coreference, ellipsis }
2: var $\Omega$ : KG ← union of all KGs
3: var $frequency$ : Dict($\mathcal{T} \rightarrow \mathbb{N}$)
4: var $dependencies$ : Dict($\mathcal{T} \rightarrow$ List($\mathcal{T}$))
5: function $\kappa_t$($Q$: list of existing queries for a given graph,
       G: KG) : Set ( Function(Query) : $\mathbb{B}$ )
6:     ▷ Build a set of conditions (predicates) that a query
       must satisfy for the type $t$ given the context of the
       generation $Q$ on a the graph $G$ to get fully validated
7: function $\phi_t$($G$ : KG, $q'$ : base query, $C$: set of predicates)
       : Query or $undefined$
8:     ▷ Try to create a query of type $t$ based on $G$, op-
       tionally from $q'$ for some types, and satisfying the
       conditions $C$. Return $undefined$ if no such query
       can be created.
9: function GENERATE($t$: Type, $G$: knowledge graph,
       $Q$: list of generated queries for G) : Query or
       $undefined$
10:    $dep\_types$ : List($\mathcal{T}$) ← $dependencies[t]$
11:    $q$ : Query ← $undefined$
12:    $q'$ : Query ← $undefined$
13:    ▷ If type $t$ requires to be build on top of another query,
         try first to build this intermediate query
14:    if $dep\_types \neq$ [] then
15:        $success$ : $\mathbb{B}$ ← $false$
16:        while $dep\_types \neq$ [] and $\neg success$ do
17:            $t'$ ← pop least frequent from $dep\_types$
18:            $C_{t'} \leftarrow \kappa_{t'}(Q, G)$
19:            $q'$ ← GENERATE($t', G, Q$)
20:            if $q' \neq undefined$ then
21:                ▷ Now try to include type $t$ in query $q'$
22:                $C_t \leftarrow \kappa_t(Q, G)$
23:                $q \leftarrow \phi_t(G, q', C_t)$
24:                $success \leftarrow true$
25:    else ▷ If no intermediate query to build, directly try to
         build for type $t$
26:        $C_t \leftarrow \kappa_t(Q, G)$
27:        $q \leftarrow \phi_t(G, q', C_t)$
28:    return q
```

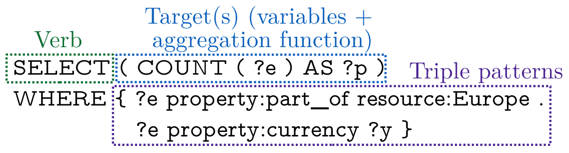
Algorithm 1: Query generation for a given type t.

exists in a larger, more general, KG, denoted as Ω (line 2). Likewise, nonsensical queries can be generated such that their elements are never observed together in any triple from Ω. All these constraints are given by auxiliary type-specific function κ_t (line 5). The generation of one query is orchestrated by the function GENERATE (lines 9-28) for the given input type t, knowledge graph G, and the previous queries Q generated on it. The balancing scheme over the type distribution is managed thanks to global statistics of all queries generated so far on all KGs (global variable $frequency$, line 3). For each KG in WebNLG, the overall process (not described in Algorithm 1) iteratively generates queries until none can be generated anymore, i.e., calls to GENERATE return $undefined$ for all types $t \in \mathcal{T}$. Examples of generated queries are given in Appendix A.1.

Then, given the whole set of resulting SPARQL queries, questions were manually annotated for the queries of a selection of 100 KGs. These KGs were selected from the test set of WebNLG such that the distribution of the query types is as uniform as possible. Two natural language questions were manually annotated by one annotator for each SPARQL query. Given a query, the annotator was asked to generate questions with different surface forms to reflect the diversity of the natural language. This results in 100 "dialogues" for a total of 332 questions (from 2 to 7 per dialogue).

3.3 Homogenization

All datasets were processed to contain SPARQL queries unified in a similar way as the following query whose verbalizaton could be "*how many currencies co-exist within the countries of Europe?*":



```
SELECT ( COUNT ( ?e ) AS ?p )
WHERE { ?e property:part_of resource:Europe .
        ?e property:currency ?y }
```

In particular, all entity IDs or URIs from WikiData or DBPedia were replaced by their label. Entities, properties and types were prefixed by `"resource:"`, `"property:"`, and `"ontology:"`, respectively. Triples were shuffled to prevent the model to learn in a biased way on the static ordering of some datasets. Variable names were anonymized with a single random letter (still prefixed by `"?"`) and some constructions were randomly replaced by equivalent forms[5].

For SimpleQuestions and CSQA, special efforts were required since they do not come with SPARQL queries. Especially for CSQA, we relied on the formalism from CARTON (Plepi et al., 2021) as an pivot representation from which SPARQL queries were generated by ourselves.

By default, the train/validation/test splits are the same as for the original datasets. In the case of LC-QuAD 2.0 and ParaQA, for which no validation set is officially provided, validation data was randomly extracted from the initial training set.

4 Models

This paper investigates the difficulty of the task for pretrained transformer models. This section first provides information about the fine-tuning process

[5]For instance, some `UNION` clauses were replaced using `VALUES` clauses. Still, some constructions could not be introduced, like `GROUP BY`, `ORDER BY` or `LIMIT`.

of these models, and then introduces several naive models used as baselines in the experiments.

Transformer models. The proposed models are encoder-decoder (i.e., autoregressive) transformers, namely **BART** (Lewis et al., 2020) and **T5** (Kale and Rastogi, 2020), fine-tuned on the SPARQL-to-text task. For both architectures, the models are the "*base*" version, as provided by HuggingFace[6]. This appeared as a reasonable size since CSQA is a very large corpus and many experimental settings are considered. Hence, the impact of the size is not considered here. Tokenizers are the default ones. Input sequences longer than the length limit of 512 tokens were truncated from the beginning, and no padding was used. The T5 prefix is "`sparql to nl:` ". The fine-tuning is performed for 2 epochs with a batch size of 4 samples, which appeared to be the best setting on the development set. The optimizer is AdamW with a static learning rate of 5×10^{-5} and no weight decay. Finally, note that WebNLG data was *not* part of BART's or T5's training data for their pre-training.

Naive models. Several naive approaches are experimented to intuit the difficulty of the task and provide reasonable baselines. The simplest approach is to concatenate all terms of all triples in the query, except variables which are ignored. The order of the triples is the same as in the query—i.e., randomized, no micro-planning (Reiter and Dale, 1997, Chap. 5), hence the name **blind concatenation**. Alternatively, a rule-based micro-planning was implemented to spot the main triple in the query, that is the one on which the beginning of the question will focus[7]. Then, the main triple is placed first when concatenating. This approach is denoted as **smart concatenation**. To complete the approach, templates of questions were introduced to instanciate the triples. The most naive solution is to prefix all questions with "what" since this is the most frequent prefix in the training datasets. Another solution relies on a set of more sophisticated patterns, each being adapted to specific query configurations (query verb, target variable, shape of the main triple, etc.). This technique is called **smart concatenation + pattern**.

The next sections provide global results used to prototype a unique model for all the datasets (Section 5), and in-depth experiments to understand the current limits of the models (Section 6).

5 Prototyping Experiments

This section studies the design of a SPARQL-to-text model and provides global results. First, it studies the impact of adding input information along with the single SPARQL query. Then, the different training datasets are merged in order to investigate the generalization capacity of the models and to come up with a unique model for all the datasets. All results are presented in terms of METEOR (Banerjee and Lavie, 2005) and BERTScore (F1 score) (Zhang et al., 2020) on the test set of each corpus[8], using HuggingFace metrics.

5.1 Input features

The minimal input for the model is the SPARQL query to convert. Additionally, the model can be fed with the expected **answer** (if the question is answerable). In the case of a conversation, the **context** of the discussion can also be given, i.e. the previous questions and answers in natural language. This information is meant to be particularly helpful to properly generate coreferences and ellipses. Using all information, the model's inputs are formatted as follows: "`<context> conversational context </context> <query> SPARQL query </query> <answer> answer(s) <answer>`". The number of answers is limited to 10. Ideally, the context should be restricted to the few last turns sharing a link with the current query under study. This assumption was tested by identifying the restricted context in an oracle way using meta-information from CSQA.

Table 3 reports the impact of including the answer and the context when training the model on each corpus. First, it appears that the models are better than the naive approaches, while BART and T5 seem relatively equivalent. Then, the impact of including the answer greatly varies accross the corpora and models. Even if the best results are most frequently obtained when the answer is considered, it does not seem as useful as expected, meaning that most of the required information can probably be derived from the sole SPARQL query. The impact of the conversation context (CSQA) is more visible, with a major benefit in favor of including the context. Then, while restricting this context

[6]`https://huggingface.co/models`

[7]The rules analyze features like the presence or not of the target variable in a triple, the number of variables in this triple, the nature of the property, etc.

[8]Results are not reported on the validation sets as they were used to define several hyperparameters.

			METEOR					BERTScore-F1				
			SimQ.	LCQ2.0	ParaQA	CSQA	*Avg.*	SimQ.	LCQ2.0	ParaQA	CSQA	*Avg.*
Naive	Blind concatenation		34.8	26.8	26.8	35.5	*31.0*	89.3	88.4	88.1	87.4	*88.3*
	Blind conc. + what		39.1	29.2	27.8	36.1	*33.1*	89.8	89.5	89.1	89.3	*89.4*
	Smart conc. + what		46.6	37.1	36.5	40.5	*40.2*	91.4	89.7	89.8	89.8	*90.2*
	Smart conc. + pattern		45.1	44.7	48.3	41.8	*45.0*	91.1	90.4	90.5	90.1	*90.5*
BART	No answ.	No context	60.4	53.7	57.5	67.1	*59.7*	94.3	**93.1**	93.5	94.3	***93.8***
		Restr. cont.	–	–	–	77.3	–	–	–	–	96.4	–
		Full context	–	–	–	77.5	–	–	–	–	96.2	–
	Answer	No context	**61.0**	53.6	57.1	67.4	***59.8***	**94.4**	93.0	**93.6**	94.3	***93.8***
		Restr. cont.	–	–	–	**77.8**	–	–	–	–	**96.5**	–
		Full context	–	–	–	77.7	–	–	–	–	96.2	–
T5	No answ.	No context	58.7	**54.5**	57.7	66.0	*59.2*	94.1	**93.1**	93.5	94.1	*93.7*
		Restr. cont.	–	–	–	76.2	–	–	–	–	96.2	–
		Full context	–	–	–	76.4	–	–	–	–	96.0	–
	Answer	No context	59.7	54.3	**58.9**	66.5	***59.8***	94.2	93.0	**93.6**	94.1	*93.7*
		Restr. cont.	–	–	–	77.2	–	–	–	–	96.3	–
		Full context	–	–	–	77.1	–	–	–	–	96.0	–

Table 3: Performances on the test set when training on each dataset separately with different input settings. Best results for each dataset are in bold, and the darker the cell, the worse it is.

		METEOR					BERTScore-F1				
	↓ Training \Test →	SimQ.	LCQ2.0	ParaQA	CSQA	W.-QA	SimQ.	LCQ2.0	ParaQA	CSQA	W.-QA
Best naive		45.1	44.7	48.3	41.8	43.3	91.1	90.4	90.5	90.1	88.9
BART	Single corpus	61.0	53.6	57.1	**77.7**	–	**94.4**	**93.0**	93.6	**96.2**	–
	All corpora	**61.1**	53.2	57.6	**77.7**	40.1	**94.4**	92.9	93.5	**96.2**	89.5
	All corp. (balanced)	57.7	51.5	**60.3**	77.6	40.4	93.7	92.4	93.7	**96.2**	89.5
T5	Single corpus	59.7	**54.3**	58.9	77.1	–	94.2	**93.0**	93.6	96.0	–
	All corpora	60.1	54.1	58.1	77.1	**44.0**	94.2	**93.0**	93.7	96.1	**90.2**
	All corp. (balanced)	57.3	51.7	60.0	77.2	43.5	93.7	92.5	**93.8**	96.0	90.1

Table 4: Performances when merging the training data. Best results for each dataset are in bold, and the darker the cell, the worse it is.

seems to outperform the full (unrestricted) context on BERTScore, no conclusion can be drawn regarding METEOR. This is a useful conclusion since correctly truncating the context may not be a simple task in real conditions. In the remainder, all models are trained with the answer and the full context. Finally, for all approaches (naive and transformers), SimpleQuestions and CSQA lead to higher results, which tends to think that they are less diverse than ParaQA and LC-QuAD 2.0.

All these conclusions have been supported by back-end experiments on WebNLG-QA (detailed in Appendix A.2) regarding the impact of the answer and conversational context, as well as the poor transfer of SimpleQuestions and CSQA.

5.2 Merged training

To take advantage of the different characteristics of each corpus, fine-tuning was performed based on the merged training samples of each dataset. Since the disparity is great between the size of each corpus, a balancing strategy was tested by weighting the corpora in inverse proportion to their respective size. The results are reported in Table 4.

On the one hand, it appears that merging the training data without any balancing scheme neither improves nor degrades the overall performance on the test set of these corpora since no global trend can deduced[9]. On the contrarty, balancing the data surprisingly degrades the results. This is probably because of weights with too high values since size differences are very strong, for instance between ParaQA and CSQA (the scaling factor is more than 400). In the remainder, the models are trained on mixed corpora with no balancing.

On the other hand, the last column of Table 4 for each metric reports the performance on WebNLG-QA. First, while the score of the naive approach is comparable to the other datasets, a significant drop is reported for the transformers models, leading to similar or even worse results than the naive approach. In our opinion, this is because the models are biased towards the most frequent query structures in the training sets, while these frequency disparities are globally smoothed out in WebNLG-QA. On the contrary, the naive approach is agnostic

[9]Except for ParaQA, which is the smallest corpus. Mixing with other data probably alleviate a sparsity issue.

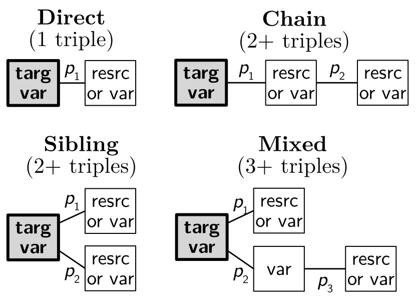


Figure 1: Topologies of the conjunctive queries.

to these considerations. Finally, it seems that T5 is more robust than BART. For this reason, BART is discarded in the next section where deeper investigations are conducted to understand what the model learns and what is still difficult for it.

6 Detailed Analysis

This section first analyses how the T5 model behaves on different query types. Then, a human evaluation on a real application is presented to evaluate the intelligibility and effectiveness of the generated questions. The focus is given on the challenge set WebNLG-QA but complementary results for the other datasets are reported in different appendices.

6.1 Robustness over the query types

Queries are categorized according to[10]:

The triples. They can mainly vary w.r.t. the *number of triples* (with the assumption that the more triples a question contains, the more complex it is), and the *logical connectors* between them (by default, logical *AND*s but potentially disjunctions with logical *OR*s, or exclusion like *triple*$_1$ *AND NOT triple*$_2$). In the conjunctive case (i.e., *AND* connectors) , the variables can interconnect the triples following different *topologies* w.r.t. the position of the target variable, as depicted in Figure 1. Additionnaly, *type* information can be given for the variables. Although this information is also written as a triple, "typing triples" (with a special property "`rdf:type`") are not considered as regular triples when counting the number of triples in the query in our statistics. Finally, constraints on the possible values for the variables can enable expressing *comparisons* to static values (`FILTER` clauses on string, numbers or dates).

The expected answer(s). Queries vary also according to the *type of the expected answer(s)* (entities, numbers or booleans), the *number of answers* (1, more or even 0 if the question cannot be answered), and the *number of target variables* (1, 2 or even 0 when simply checking a fact).

[10]If needed, more details can be found in Appendix A.3.

The conversational context. In a conversation, consecutive turns may re-use information from the previous turns, potentially leading to *coreferences* (replacing an entity by an equivalent pronoun or noun phrase to avoid repetition) and *ellipses* (skipping a syntagm that can be deduced from the previous sentences). While generating these can bring a more natural flow of questions, it can also bring ambiguity. If no coreference and no ellipsis is present, the question is denoted as *self-sufficient*.

The meaningfulness. Whereas queries are expected to make sense, it is worth observing how the model behaves when facing non-sensical questions.

Table 5.a presents the METEOR and BERTScore results for all categories and subsequent query types in WebNLG-QA using the T5 model fine-tuned on all merged corpora, and with the expected answers and the conversational context. This is compared to the best naive approach. Color shades depict the difference with the average performance for each dataset separately (red means lower than the average, green means greater). In complement, Table 5.b reports the standard deviation within each category of query types in order to evaluate the robustness against each variability factor. For the sake of completeness, results on all the datasets are in Appendix A.4. From Table 5.a, it appears that difficult types are those for which concurrent types can co-exist. For instance, queries with 2 triples can represent multiple configurations like sibling or chain topologies, conjunctive or disjunctive connectors, etc. On the contrary, queries with 1 or 3+ triples do not allow this diversity and they are better predicted. This is the same when the expected answer is an open entity (i.e., which is not part of closed list of choices in the query). Then, the model seems to also struggle when several target variables are considered. Finally, both tables show that handling the dialogue context is difficult for the model. Counter-intuitively, especially w.r.t. the results of Sec. 5.1, the results of the naive approach may even encourage one not to consider it.

6.2 Evaluation in a real application

To verify that the generated questions are understandable and lead to the expected answers, they were integrated in quizzes. As a reminder, each

		Nb of quest.	METEOR W.-QA (T5)	METEOR W.-QA (Naive)	BERTScore-F1 W.-QA (T5)	BERTScore-F1 W.-QA (Naive)
Average		332	44.0	43.3	90.2	88.9
Number of triplets	1	181	47.0	46.7	90.7	89.3
	2	127	39.3	39.7	89.2	88.3
	More	24	46.9	37.4	91.2	88.3
Logical connector	Conjunction	323	44.0	43.2	90.1	88.8
	Disjunction	6	48.6	47.9	93.2	90.5
	Exclusion	10	45.0	47.8	89.2	87.9
Topology	Direct	153	41.5	48.2	89.7	89.1
	Sibling	32	38.8	29.3	87.6	86.0
	Chain	28	48.6	38.8	91.8	90.1
	Mixed	23	46.9	37.4	91.2	88.3
	Other	96	47.7	42.0	90.9	89.0
Variable typing	None	282	43.7	44.6	90.2	89.1
	Target var.	18	49.6	32.8	89.5	86.3
	Internal var.	31	43.4	37.4	90.2	88.3
Compar-isons	None	283	44.2	45.2	90.1	88.9
	String	22	37.9	31.7	90.0	88.1
	Number	13	40.1	36.1	89.4	89.0
	Date	14	51.6	34.1	91.9	89.0
Answer type	Entity (open)	177	42.4	38.7	89.7	88.4
	Entity (closed)	5	62.1	73.8	89.7	91.7
	Number	33	55.7	58.0	92.5	89.5
	Boolean	90	47.4	43.1	90.9	89.2
Answer cardinality	0 (unanswer.)	35	47.1	57.3	90.4	90.5
	1	281	46.2	43.4	90.5	88.9
	More	51	33.3	42.7	88.4	88.9
Nb. target variables	0 (⇒ ASK)	90	47.4	43.1	90.9	89.2
	1	217	44.9	42.5	90.1	88.7
	2	25	24.6	51.4	87.5	89.4
Dialogue context	Self-sufficient	144	53.3	46.7	92.6	90.3
	Coreference	154	36.5	40.6	88.4	87.9
	Ellipsis	90	35.6	34.9	87.3	86.6
Meaning	Meaningful	302	43.4	41.7	90.1	88.7
	Non-sense	30	49.9	59.5	90.6	90.7

(a) Average for each query type of each category (red/green means "worse/better than average for the given model")

Query type's Category	METEOR W.-QA (T5)	METEOR W.-QA (Naive)	BERTScore-F1 W.-QA (T5)	BERTScore-F1 W.-QA (Naive)
All categories	6.8	8.8	1.38	1.18
Number of triplets	4.4	4.8	1.05	0.61
Logical connector	2.4	2.7	2.08	1.33
Topology	4.3	6.9	1.66	1.53
Variable typing	3.5	5.9	0.38	1.46
Comparisons	6.1	5.9	1.09	0.43
Answer type	8.7	15.9	1.34	1.43
Answer cardinality	7.7	8.2	1.17	0.97
Nb. target variables	12.5	5.0	1.82	0.40
Dialogue context	10.0	5.9	2.79	1.86
Meaning	4.6	12.6	0.33	1.46

(b) Standard deviation for each category (black/white cells mean "higher/lower than the global std. dev. of the model")

Table 5: Average (a) and standard deviations (b) of METEOR and BERTScore for all query type categories.

sample in WebNLG-QA includes a small KG and the corresponding paragraphs provided by the original WebNLG corpus. For each sample, follow-up tuples (query, question, answer) can be used to quiz a user that would have read the paragraph. Before assessing the effectiveness of the generated questions in these quizzes, prior experiments are conducted.

Parag.	Answ.	Context	METEOR Naive	METEOR BART	METEOR T5	BERTScore-F1 Naive	BERTScore-F1 BART	BERTScore-F1 T5
No	No	None	43.3	41.5	48.2	88.9	90.1	**91.1**
		Restr.		37.4	44.4		89.2	90.4
		Full		36.4	44.0		88.9	90.2
	Yes	None		44.8	47.9		90.6	**91.1**
		Restr.		40.8	44.8		89.8	90.4
		Full		40.1	44.0		89.5	90.2
Yes	No	None		42.3	**50.0**		89.9	**91.1**
		Restr.		38.4	44.5		89.1	90.2
		Full		37.9	43.4		88.9	90.0
	Yes	None		45.7	49.6		90.1	**91.1**
		Restr.		40.9	45.1		89.5	90.3
		Full		39.6	43.9		89.2	90.0

Table 6: Impact of changing the input features at *inference time* on WebNLG-QA using T5 fine-tuned on all merged corpora with full context and answers.

Input features at inference time. While including the answer and the conversational context has been decided at *training time* based on results of Section 5.1 (and Appendix A.2), previous conclusions from Section 6.1 have led us to study the impact of different inputs at *inference time*. Hence, Table 6 reports the scores obtained by the T5 model trained with the answers and contexts when feeding these two elements or not at inference time. This experiment also test the inclusion of the paragraph in input to provide contextualized knowledge to the model, even though the latter was not trained using such information. For a better analysis, results for BART are reported as well. Regarding the conversational context, these numbers show different trends as those reported during the prototyping experiments since including the context brings worse results for both models. Then, the T5 no longer benefits from the answer either (whereas BART clearly does). Finally, using the paragraph improves the results for T5 in terms of METEOR but not BERTScore, while this degrades the results for BART. These surprising conclusions call for more investigation. Currently, one may think that (*i*) T5 used the conversational context and answers during training to learn how to parse the SPARQL and then does not need the information later on, and (*ii*) that the multi-task pretraining of T5 included text comprehension task (summary, text-based QA, etc.) helps the model understanding the paragraph even after fine-tuning on the SPARQL-to-text task.

Human evaluation on quizzes Questions for the quizzes were either the reference or generated

	Reference	Best Naive	T5 (Query+Answ. +Context)	T5 (Query +Paragraph)
Answer accuracy (%)	**78.6** (41.0)	32.8 (47.0)	53.0 (50.0)	60.5 (48.9)
Linguistic correctness	**4.72** (0.75)	2.78 (1.43)	4.06 (1.20)	4.45 (0.90)
Dialogue naturalness	***3.74** (1.18)	2.15 (1.02)	3.15 (1.19)	*3.52 (1.03)

(a) Global results of the human evaluation (standard deviation between brackets). Difference between values marked with * is *not* statistically significant (Student paired t-test with $p = 0.05$). All others are.

	Answer accuracy				Linguistic correctness			
Query type's category	Ref.	Best Naive	T5 (Query +Answ. +Ctxt)	T5 (Query +Para.)	Ref.	Best Naive	T5 (Query +Answ. +Ctxt)	T5 (Query +Para.)
All categories	15.3	22.8	19.9	16.9	0.24	0.44	0.33	0.20
Number of triplets	3.6	29.1	19.6	15.7	0.06	0.55	0.37	0.20
Logical connector	25.6	19.5	10.1	28.2	0.30	0.23	0.23	0.40
Topology	5.0	18.6	11.4	12.5	0.08	0.58	0.26	0.17
Variable typing	6.5	16.4	14.0	9.4	0.07	0.37	0.22	0.10
Comparisons	10.9	18.7	22.9	14.5	0.14	0.59	0.75	0.16
Answer type	29.7	11.7	28.5	1.5	0.57	0.41	0.32	0.17
Answer cardinality	14.3	35.5	30.6	24.0	0.20	0.44	0.07	0.10
Nb. target variables	6.3	20.7	27.4	10.6	0.14	0.29	0.38	0.17
Dialogue context	4.9	11.3	6.1	3.9	0.03	0.27	0.11	0.05
Meaning	14.0	47.0	33.9	28.1	0.35	0.59	0.23	0.09

(b) Standard deviations for each category of query type, the darker, the higher (across all models).

Table 7: Results of the human evaluation (quizzes).

using the naive approach, or T5. For T5, two types of input were provided at *inference time*: with the answer and context (as in the training setup), or only with the paragraph. 2 examples of quizzes are provided in Appendix A.5. There are 100 quizzes for each setup, based on the same 100 paragraphs. 20 users took part in the evaluation. All quizzes and their answers were seen exactly once. Users had to select their answers in a closed list of possibilities ("Yes", "No", 0, 1, 2, . . . , or entities from the paragraph). They could also report that the question cannot be answered because the paragraph did not contain the answer or the question was not understandable. By comparing with the expected and collected answer(s), accuracies were computed for each setup. After answering a quiz, users also had to rate the linguistic correctness of each question and the overall naturalness of the quiz (flow of questions). Both scores range between 1 (very bad) and 5 (excellent).

Table 7 reports the average results for each setup (7.a) and the variability of the answer accuracy and linguistic correctness within each category of query types (7.b). Exhaustive values for all query types are provided in Appendix A.6. As expected, it appears that the reference questions rank first for all the metrics. While the linguistic correctness is excellent, it is worth noting that the answer accuracy is not perfect. A manual analysis shows that this comes from confusions of the users, for instance between entity question (what, who. . .) and some boolean questions (is there. . .), or cascaded errors. Likewise, the naturalness of the flow of questions is not perfect because some questions are unanswerable. Then, the ranking is the same as with METEOR and BERTScore. Nonetheless, the difference between the naive approach and the T5 models is much clearer, which highlights the limits of automatic metrics for the task. By the way, this confirms that feeding the T5 model with the paragraph is significantly helpful. Compared to T5 with answer and context, the questions are more robust against almost all variability factors (Table 7.b).

7 Conclusion and Future Work

In this paper, we have studied in depth the problem of generating questions from SPARQL queries, in particular in order to be able to integrate these questions in a conversational knowledge-based application such as a QA system or a task-oriented dialogue. Contributions stand in the proposed corpora, including a new challenge set (WebNLG-QA), and in the multiple experiments conducted to highlight the limits of the popular pretrained models BART and T5 for the SPARQL-to-text task. These experiments show that, although the linguistic quality of the generated questions is good, the task only really works well for unambiguous and frequent situations, generally conforming to what has been seen in training.

In the future, it would be interesting to evaluate the questions generated with a QA system. Although the varying performance of these systems may bring uncertainty in the interpretation of the results, this would complement the human evaluation results and provide another basis for other researchers to compare their own question generation models. Then, several limitations remain to be overcome. First of all, a better generation of coreferences and ellipses should be investigated, as well as a better transfer capacity from one corpus to another. Then, apart from the use of other KBQA corpora than those used in this paper, it is likely that the use of unsupervised approaches, i.e. not requiring aligned questions and queries, is a challenging avenue to explore. In particular, this could favor help mixing knowledge-based and text-based approaches, as called by our last results.

References

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. Association for Computational Linguistics.

Sheng Bi, Xiya Cheng, Yuan-Fang Li, Yongzhen Wang, and Guilin Qi. 2020. Knowledge-enriched, type-constrained and grammar-guided question generation over knowledge bases. In *Proceedings of the International Conference on Computational Linguistics (CICLing)*, pages 2776–2786.

Debanjali Biswas, Mohnish Dubey, Md Rashad Al Hasan Rony, and Jens Lehmann. 2021. Vanilla: Verbalized answers in natural language at large scale. *arXiv preprint arXiv:2105.11407*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *arXiv:1506.02075 [cs]*. ArXiv: 1506.02075.

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.

Ruixiang Cui, Rahul Aralikatte, Heather Lent, and Daniel Hershcovich. 2022. Compositional generalization in multilingual semantic parsing over wikidata. *Transactions of the Association for Computational Linguistics*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question Generation for Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 866–874. Association for Computational Linguistics.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In *Proceedings of the The Semantic Web (ISWC)*, pages 69–78. Springer International Publishing.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris Van Der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task Overview and Evaluation Results (WebNLG+ 2020). In *Proceedings of the International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL easier to infer from natural language specifications. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2030–2042. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4524–4535. Association for Computational Linguistics.

Kelvin Han, Thiago Castro Ferreira, and Claire Gardent. 2022. Generating questions from wikidata triples. In *Proceedings of the Language Resource and Evaluation Conference (LREC)*. ELDA.

Endri Kacupaj, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. 2021. ParaQA: A Question Answering Dataset with Paraphrase Responses for Single-Turn Conversation. In *Proceedings of the The Semantic Web (ISWC)*, pages 598–613. Springer International Publishing.

Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. 2020. Vquanda: Verbalization question answering dataset. In *European Semantic Web Conference*, pages 531–547. Springer.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the International Conference on Natural Language Generation (INLG)*, pages 97–102.

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 *Proceedings of The Semantic Web (ISWC)*, pages 382–398. Springer International Publishing.

Selvia Ferdiana Kusuma, Daniel O Siahaan, and Chastine Fatichah. 2020. Automatic Question Generation In Education Domain Based On Ontology. In *Proceedings of the International Conference on Computer Engineering, Network, and Intelligent Multimedia*, pages 251–256.

Philippe Laban, Chien-Sheng Wu, Lidiya Murakhovs' ka, Wenhao Liu, and Caiming Xiong. 2022. Quiz design task: Helping teachers create quizzes with automated question generation. In *Proceedings of the North American Chaapter of the ACL (NAACL)*.

Monica S Lam, Giovanni Campagna, Mehrad Moradshahi, Sina J Semnani, and Silei Xu. 2022. Thingtalk: An extensible, executable representation language for task-oriented dialogues. *arXiv preprint arXiv:2203.12751*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7871–7880.

Lidiya Murakhovs'ka, Chien-Sheng Wu, Tong Niu, Wenhao Liu, and Caiming Xiong. 2022. MixQG: Neural Question Generation with Mixed Answer Types. In *Proceedings of the North American Chapter of the ACL (NAACL)*.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 977–988. Association for Computing Machinery.

Axel-Cyrille Ngonga Ngomo, Diego Moussallem, and Lorenz Bühmann. 2019. A Holistic Natural Language Generation Framework for the Semantic Web. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 819–828. INCOMA Ltd.

Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. 2021. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 356–371. Springer.

Ehud Reiter and Robert Dale. 1997. *Building Natural Language Generation Systems*. Cambridge University Press.

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, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with macaw. *arXiv preprint arXiv:2109.02593*.

Ruqing Zhang, Jiafeng Guo, Lu Chen, Yixing Fan, and Xueqi Cheng. 2021. A Review on Question Generation from Natural Language Text. *ACM Transactions on Information Systems*, 40(1):14:1–14:43.

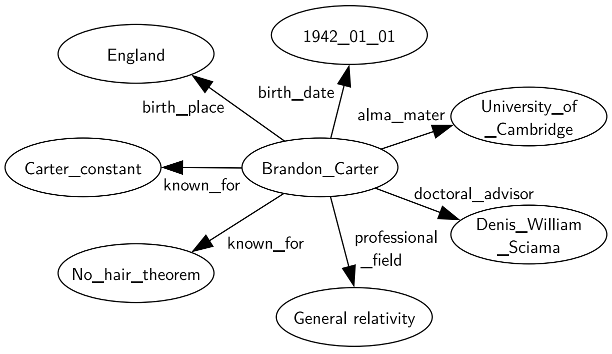


Figure 2: Example of knowledge graph from WebNLG.

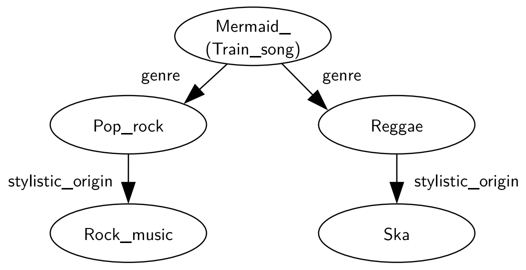


Figure 3: Example of another knowledge graph from WebNLG.

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

A Appendices

A.1 Examples of generated SPARQL queries

This sections presents sequences of SPARQL queries generated as exposed in Section 3.2 and Algorithm 1 based on 2 sample KGs, depicted in Figures and 3.

Using the graph of Figure A.1, the resulting sequence of SPARQL queries is the following:

1. `SELECT DISTINCT ?d WHERE { ?d property:birth_date ?k . FILTER ( CONTAINS ( YEAR ( ?k ) , '1942' ) ) . ?d property:known_for resource:No_hair_theorem }`

2. `SELECT DISTINCT ( COUNT ( ?m ) AS ?g ) WHERE { resource:Brandon_Carter property:known_for ?m }`

3. `SELECT DISTINCT ?m WHERE { resource:Brandon_Carter property:known_for`

`?m . FILTER ( ?m != resource:No_hair_theorem ) }`

4. `SELECT DISTINCT ?b WHERE { resource:Brandon_Carter property:birth_place ?b . FILTER ( STRSTARTS ( LCASE ( ?b ) , 'e' ) ) }`

5. `SELECT DISTINCT ?t ?g WHERE { resource:Brandon_Carter property:alma_mater ?g . resource:Brandon_Carter property:doctoral_advisor ?t }`

6. `SELECT DISTINCT ?x WHERE { resource:Brandon_Carter property:sports_offered ?x }`

Using the graph of Figure 3, the generated queries are:

1. `SELECT DISTINCT ?k WHERE { { { ?k property:stylistic_origin resource:Ska } UNION { ?k property:stylistic_origin resource:Rock_music } } }`

2. `SELECT DISTINCT ?k WHERE { ?k property:stylistic_origin resource:Rock_music }`

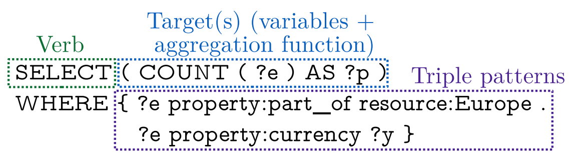
3. `ASK WHERE { resource:Mermaid_(Train_song) property:genre resource:Pop_rock }`

A.2 Performance of each separate dataset on WebNLG-QA

This appendix details how the models trained on each dataset separately transfer to the WebNLG-QA challenge set. Results reported in Table 8 show the same trends as observed on the test sets, respectively: the impact of including the answer is not obvious, while including the context help for the model trained on CSQA. The results also show that SimpleQuestions and CSQA cannot beat the naive approaches with expert micro-planning (*smart concatenation*). For SimpleQuestions, this seems obvious since most query types in WebNLG-QA are absent in SimpleQuestions. Regarding CSQA, this is probably due to the lack of linguistic diversity in the way to verbalize questions in this dataset (again, CSQA was generated semi-automatically). Results from Section 5.2 show that mixing the datasets solves this problem.

A.3 Details on the types of queries

As a reminder, a SPARQL query is as follows:



```
SELECT ( COUNT ( ?e ) AS ?p )
WHERE { ?e property:part_of resource:Europe .
  ?e property:currency ?y }
```

It mainly relies on triple patterns of the form (*subject*, *property*, *object*), where each element can refer to an entity (resource, literal, type, property) from the KG or represent a variable to be solved (prefixed by "?"). The query also specifies the nature of the answer(s) to be derived from these triple patterns using a verb (`SELECT` or `ASK`), target variables and possibly aggregation functions on the values taken by these variables. This section details variability factors on these various elements, as well as the possible values as reported in the paper's tables.

A.3.1 Structure of the triple patterns

Mainly, the pattern consists of cloze triples where potential values for the blanks are designated through variables prefixed with a `?` sign. Below is a list of variability factors on the organisation of these triples.

Number of triples: Queries can include 1, 2 and more triplets. This reflects the complexity of the question. As far as what we observed, it is rare that more than 2 triplets are implied in real life questions as this becomes difficult to formulate within one sentence.

Logical connectors: The default connector between triples is the conjunction ($triple_1 \wedge triple_2$), but it can also be a disjunction ($triple_1 \vee triple_2$) or an exclusion ($triple_1 \wedge \neg triple_2$). Since the default connector in SPARQL is the conjunction, disjunctive and exclusive queries are more verbose.

Topology of the pattern: When triples are connected with a conjunction, they represent a connected graph where nodes are resources or variables and edges are properties. Assuming that only one variable is the target variable (which is the most frequent case), regularities can be observed in the topology of this graph w.r.t. the target variable, illustrated in Figure 4 and defined as follows:

↓ Training setup \ Training corpus →			METEOR SimQ.	METEOR LCQ2.0	METEOR ParaQA	METEOR CSQA	BERTScore-F1 SimQ.	BERTScore-F1 LCQ2.0	BERTScore-F1 ParaQA	BERTScore-F1 CSQA
Naive	Blind concatenation		33.6				86.6			
Naive	Blind conc. + what		35.8				87.7			
Naive	Smart conc. + what		41.2				88.9			
Naive	Smart conc. + pattern		43.3				88.9			
BART	No answ.	No context	**30.3**	44.1	45.1	31.7	88.9	90.7	90.8	87.9
BART	No answ.	Restr. cont.	–	–	–	30.9	–	–	–	88.6
BART	No answ.	Full context	–	–	–	33.4	–	–	–	88.2
BART	Answer	No context	29.5	45.4	45.3	31.5	88.8	90.8	90.4	88.1
BART	Answer	Restr. cont.	–	–	–	31.0	–	–	–	88.7
BART	Answer	Full context	–	–	–	33.4	–	–	–	88.3
T5	No answ.	No context	29.7	44.2	**45.9**	32.8	**89.1**	90.9	**90.9**	88.4
T5	No answ.	Restr. cont.	–	–	–	35.5	–	–	–	**89.7**
T5	No answ.	Full context	–	–	–	37.9	–	–	–	89.1
T5	Answer	No context	29.1	**45.7**	**45.9**	33.5	88.8	**91.0**	90.8	88.5
T5	Answer	Restr. cont.	–	–	–	35.5	–	–	–	**89.7**
T5	Answer	Full context	–	–	–	**38.3**	–	–	–	89.0

Table 8: METEOR and BERTScore (F1) on WebNLG-QA when training on SimpleQuestions, LC-QuAD 2.0, ParaQA, and CSQA independently. The darker, the worse.

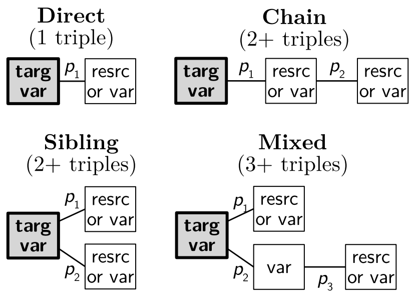


Figure 4: Topologies of the conjunctive query graphs.

1. A *direct* topology refers to a graph with only 2 nodes (i.e. 1 triplet).

2. *chain* denotes the situation where the graph is linear with more than 2 nodes and the target variable is at one of its extremities.

3. *sibling* refers to a graph the target variable is directly linked to 2 or more resources (whatever the orientation of the edges), i.e. the graph is a star of depth 1.

4. *mixed* is a mixture of the sibling and chain structures, that is a star topology centered on the target variable and with at least one branch of the star whose depth is more than 1.

Variable typing: Associating types to concepts (target of internal variables) in a question is sometimes critical to help understand a question. In the remainder, we consider typing as a specific case of property. Thus, triplets about typing are not counted as regular triplets.

Comparisons: Filtering clauses can be append to the triplets to restrict the range of their variables. Based on the corpora used in this paper, this comparisons can be numbers, strings or dates.

Superlatives: A specific case of comparison is when a minimal or maximal value is asked, or (most frequently) the entity associated with this extremum. While `MIN` and `MAX` are predefined aggregation functions in SPARQL, retrieving the is less trivial since it requires nested queries.

A.3.2 Answers

Queries vary also according to the expected answer.

Data type: Most usually, answers are *entities* but they can also be *numbers* (typically a count over entities) or *booleans* when facts are asked to be checked.

Number of intentions: Queries can include a variable number of target variables. This is referred to the number of intentions. While one intention is the most frequent situation, corpora also include questions with two intentions, as well as no intention (i.e. no target variable, when a fact is to be checked).

Number of answers: For each target variable, the number of answer can also vary depending on the information in the KG and the cardinality of the query properties. This may be zero if entity matches the query in the KG. Then, for a given person in subject, the property `birth_date` should lead to a single answer, while `parent_of` may return several objects.

A.3.3 Conversational context

Finally, in the context of conversations, the discussion may re-use information from the previous turns, potentially leading to coreferences and ellipses. Coreferences are the act of replacing an entity already mentioned in the discussion by a pronoun or another equivalent noun phrase in subsequent occurrences. Second, an ellipsis is the omission of a sentence segment deemed useless by the speaker because it can be deducted from previous turns, typically because the omitted segment (and no longer just an entity) would be a raw repetition. These linguistic phenomena are guided by the will to be brief by not repeating information, and constrained by the need to remain unambiguous. These linguistic phenomena are complex because they are not systematic. Hence, a coreference may link a pronoun with an entity mentioned several turns ago if there is not difficult to infer this link. At the opposite, a repetition in two consecutive turns may be required to avoid ambiguity. The same applies to ellipses with an even higher degree of complexity since ellipses require to rely on the syntact structure of a previous turn. Hence, generating coreferences and ellipses can be improve naturalness, it can also bring ambiguity.

A.4 Details on query types for all the datasets

Table 9 presents the METEOR and BERTScore results for all query types on each corpus using the T5 model fine-tuned on all merged corpora, and with the expected answers and the conversational context. For each test set, color shades depict the distance to the average performance on this dataset (red means lower than the average, green means greater). For WebNLG-QA, values are reported for the naive approach as well, since the average results are close (see Section 5.2).

Table 10 examines the impact of each category of query types from Table 9 in order to evaluate the robustness of the model.

A.5 Examples of quizzes

Tables 11 and 12 present two examples of quizzes. The first example is related to the queries of Figure A.1 from Appendix A.1.

- It can clearly be observed that the references regularly use coreferences or ellipses (in bold) to make the questions shorter and more fluent, and that the T5 models rarely generate such linguistic phenomena (in Q2 of Example 1, T5 generates "that person").
- Other limits of the transformers can be noticed. For instance, the underlying query of Q3 contains an exclusion ("Except the No-hair Theorem, what is Brandon Carter known for?"), which T5 does not generate at all.
- In Q1 of the second example, the underlying query is an `ASK` query with a variable, which has never been observed in any of the training corpora. While T5 with answer and context tries to combine elements from the sole query, T5 with the paragraph uses the text to produce a meaningful query (even if this is not the correct question).

A.6 Detailed results of the human evaluation for each type of query

Table 13 reports the details of the answer accuracy and linguistic correctness with respect to each query type. These results show that, except for a few situations, using the paragraph as an input to the model is always better than using the answer and the context.

		METEOR						BERTScore-F1					
		SimQ.	LCQ2.0	ParaQA	CSQA	W.-QA (T5)	W.-QA (Naive)	SimQ.	LCQ2.0	ParaQA	CSQA	W.-QA (T5)	W.-QA (Naive)
Number of triplets	1	60.1	57.0	59.4	74.3	47.0	46.7	94.2	93.5	94.5	95.7	90.7	89.3
	2	–	48.2	57.3	85.8	39.3	39.7	–	92.0	93.3	97.3	89.2	88.3
	More	–	54.0	–	82.0	46.9	37.4	–	93.3	–	96.4	91.2	88.3
Logical connector	Conjunction	60.1	54.1	58.1	75.7	44.0	43.2	94.2	93.0	93.7	95.9	90.1	88.8
	Disjunction	–	–	–	84.6	48.6	47.9	–	–	–	97.0	93.2	90.5
	Exclusion	–	–	–	91.9	45.0	47.8	–	–	–	98.4	89.2	87.9
Topology	Direct	60.1	57.6	57.2	74.4	41.5	48.2	94.2	93.5	94.2	95.7	89.7	89.1
	Sibling	–	48.2	63.3	86.8	38.8	29.3	–	91.7	93.9	97.6	87.6	86.0
	Chain	–	51.7	53.2	91.4	48.6	38.8	–	92.8	92.9	98.6	91.8	90.1
	Mixed	–	44.6	–	–	46.9	37.4	–	91.4	–	–	91.2	88.3
	Other	–	55.9	67.2	83.2	47.7	42.0	–	93.4	95.3	96.9	90.9	89.0
Variable typing	None	60.1	55.9	59.2	82.2	43.7	44.6	94.2	93.3	93.9	96.8	90.2	89.1
	Target var.	–	48.8	59.3	76.0	49.6	32.8	–	92.2	93.7	95.9	89.5	86.3
	Internal var.	–	32.3	50.0	85.4	43.4	37.4	–	92.2	92.5	97.0	90.2	88.3
Comparisons	None	60.1	54.1	58.1	76.8	44.2	45.2	94.2	93.1	93.7	96.0	90.1	88.9
	String	–	53.1	–	–	37.9	31.7	–	91.8	–	–	90.0	88.1
	Number	–	55.6	–	83.7	40.1	36.1	–	93.3	–	97.1	89.4	89.0
	Date	–	52.9	–	–	51.6	34.1	–	93.2	–	–	91.9	89.0
Superlative	No	60.1	54.1	58.1	77.0	44.0	43.3	94.2	93.0	93.7	96.1	90.2	88.9
	Yes	–	–	–	85.8	–	–	–	–	–	97.2	–	–
Answer type	Entity (open)	60.1	54.2	57.9	73.7	42.4	38.7	94.2	93.0	93.7	95.6	89.7	88.4
	Entity (closed)	–	–	–	83.5	62.1	73.8	–	–	–	97.0	89.7	91.7
	Number	–	47.4	–	85.1	55.7	58.0	–	92.7	–	97.0	92.5	89.5
	Boolean	–	59.8	67.2	81.5	47.4	43.1	–	94.0	95.3	96.8	90.9	89.2
Answer cardinality	0 (unanswer.)	–	56.8	–	–	47.1	57.3	–	93.4	–	–	90.4	90.5
	1	60.1	55.2	57.8	75.7	46.2	43.4	94.2	93.2	93.7	95.9	90.5	88.9
	More	–	50.7	58.8	80.6	33.3	42.7	–	92.5	93.8	96.6	88.4	88.9
Nb. target variables	0 ($\Rightarrow$ ASK)	–	59.8	67.2	81.5	47.4	43.1	–	94.0	95.3	96.8	90.9	89.2
	1	60.1	53.8	57.3	76.6	44.9	42.5	94.2	93.0	93.6	96.0	90.1	88.7
	2	–	50.7	–	–	24.6	51.4	–	92.8	–	–	87.5	89.4
Dialogue context	Self-sufficient	60.1	54.1	58.1	76.6	53.3	46.7	94.2	93.0	93.7	96.3	92.6	90.3
	Coreference	–	–	–	77.3	36.5	40.6	–	–	–	95.8	88.4	87.9
	Ellipsis	–	–	–	80.3	35.6	34.9	–	–	–	95.5	87.3	86.6
Meaning	Meaningful	60.1	54.1	58.1	77.1	43.4	41.7	94.2	93.0	93.7	96.1	90.1	88.7
	Non-sense	–	–	–	–	49.9	59.5	–	–	–	–	90.6	90.7

Table 9: METEOR and BERTScore (F1) on the test set for the T5 model according to the type of query for each dataset. Independently for each dataset, white means a median result, red means "worse" and green means "better".

	METEOR					BERTScore-F1				
Query type's category	LCQ2.0	ParaQA	CSQA	W.-QA (T5)	W.-QA (Naive)	LCQ2.0	ParaQA	CSQA	W.-QA (T5)	W.-QA (Naive)
All categories	5.4	4.3	5.0	6.8	8.8	0.66	0.74	0.79	1.38	1.18
Number of triplets	4.5	1.5	5.8	4.4	4.8	0.85	0.83	0.77	1.05	0.61
Logical connector	–	–	8.1	2.4	2.7	–	–	1.27	2.08	1.33
Topology	5.4	6.2	7.2	4.3	6.9	0.99	1.01	1.21	1.66	1.53
Variable typing	12.1	5.3	4.8	3.5	5.9	0.66	0.78	0.58	0.38	1.46
Comparisons	1.3	–	4.9	6.1	5.9	0.68	–	0.74	1.09	0.43
Superlative	–	–	6.2	–	–	–	–	0.79	–	–
Answer type	6.2	6.6	5.0	8.7	15.9	0.72	1.18	0.66	1.34	1.43
Answer cardinality	3.2	0.7	3.5	7.7	8.2	0.48	0.12	0.49	1.17	0.97
Nb. target variables	4.6	7.0	3.4	12.5	5.0	0.66	1.25	0.57	1.82	0.40
Dialogue context	–	–	2.0	10.0	5.9	–	–	0.44	2.79	1.86
Meaning	–	–	–	4.6	12.6	–	–	–	0.33	1.46

Table 10: Standard deviation of the METEOR and BERTScore values for each category of query for all corpora. The darker, the worse.

Paragraph

Contributor to the no-hair theorem and developer of the Carter Constant, Brandon Carter, works in the field of General Relativity. He was born in England on January, 1 1942. He graduated from the University of Cambridge where he was under the doctoral advisement of Dennis William Sciama.

	Reference	**Naive**	**T5** (Query + Answ + Ctxt)	**T5** (Query + Paragraph)
Q1	Which person born in 1942 is known for the No-hair theorem?	What birth date known for No-hair theorem?	Name a person born in 1942 who is known for No-hair theory.	What is the name of a person born in 1942 that is known for No-hair theorem?
A1	Brandon Carter			
Q2	How many things is **he** generally known for?	How many known for does Brandon Carter?	How many things are **that person** known for?	How many things is Brandon Carter known for?
A2	2			
Q3	What is **the second**?	What is the known for of Brandon Carter No hair theorem?	What is Brandon Carter known for and what is No-hair theorem known for?	What is Brandon Carter known for proving the no hair theorem?
A3	Carter constant			
Q4	In which place beginning with E was **he** born?	What is the birth place of Brandon Carter?	Which country was Brandon Carter born in?	What is the birth place of Brandon Carter that begins with the letter e
A4	England			
Q5	Who was **his** doctoral advisor and what is **his** alma mater?	What is the doctoral advisor of Brandon Carter alma mater?	Which people were the doctoral advisors of Brandon Carter and are the alma mater of Brandon Carter?	What is the alma mater and doctoral advisor of Brandon Carter?
A5	Dennis William Sciama, University of Cambridge			
Q6	What sports does **he** offer?	What is the sports offered of Brandon Carter ?	What is the sports offered by Brandon Carter ?	What sports does Brandon Carter play?
A6	No answer (nonsensical question)			

Table 11: An example of a quiz.

Paragraph

Nie Haisheng was born in Zaoyang, in the Hubei province of the People's Republic of China, on October 13th, 1964. He was part of the Shenzhou 6 mission and the Shenzhou 10 mission.

	Reference	**Naive**	**T5** (Query + Answ + Ctxt)	**T5** (Query + Paragraph)
Q1	Was anybody born in Reşadiye?	Does something birth place Reşadiye?	Was Reşadiye born?	Is Reşadiye the birthplace of Nie Haisheng?
A1	No			
Q2	**What about in Zaoyang?**	Does something birth place Zaoyang?	Was Zaoyang born?	Is Zaoyang the birthplace of Nie Haisheng?
A2	Yes			
Q3	Who is **it**?	What birth place Zaoyang?	Who was born at Zaoyang?	Who was born in Zaoyang?
A3	Nie Haisheng			
Q4	How many missions did **he** participate in?	How many mission does Nie Haisheng?	How many missions did Nie Haisheng participate in?	How many missions did Nie Haisheng participate in?
A4	2			
Q5	**Which missions?**	What is the mission of Nie Haisheng?	What are the mission of Nie Haisheng?	What mission did Nie Haisheng participate in?
A5	Shenzhou 10, and Shenzhou 6			

Table 12: Another example of a quiz.

		Nb of quest.	Answer accuracy				Linguistic correctness			
			Reference	Best Naive	T5 (Query+Answ. +Context)	T5 (Query +Paragraph)	Reference	Best Naive	T5 (Query+Answ. +Context)	T5 (Query +Paragraph)
All		332	79	33	53	**61**	4.7	2.8	4.1	**4.4**
Number of triplets	1	181	76	54	69	**73**	4.7	3.2	4.3	**4.6**
	2	127	81	9	33	**46**	4.8	2.4	3.7	**4.3**
	More	24	83	0	38	**46**	4.7	2.1	**4.3**	4.2
Logical connector	Conjunction	323	79	34	53	**62**	4.7	2.8	4.0	**4.5**
	Disjunction	6	67	0	**33**	17	5.0	2.3	**4.3**	3.7
	Exclusion	10	30	0	**40**	10	4.4	2.6	**4.5**	4.1
Topology	Direct	153	73	44	61	**67**	4.6	3.2	4.2	**4.4**
	Sibling	32	78	9	53	53	4.8	2.1	4.2	**4.4**
	Chain	28	79	14	32	**36**	4.7	3.1	4.3	**4.5**
	Mixed	23	83	0	39	**48**	4.7	2.0	**4.3**	4.1
	Other	96	86	35	50	**64**	4.8	2.4	3.7	**4.6**
Variable typing	None	282	78	37	54	**61**	4.7	2.9	4.1	**4.4**
	Target var.	18	89	11	67	67	4.8	2.1	4.3	**4.6**
	Internal var.	31	77	6	39	**48**	4.7	2.4	3.9	**4.6**
Comparisons	None	283	78	35	54	**62**	4.7	2.8	4.0	**4.5**
	String	22	91	36	**68**	59	4.9	3.5	**4.6**	4.5
	Number	13	77	8	15	**31**	4.6	2.5	3.1	**4.2**
	Date	14	64	0	36	**57**	4.9	2.1	**4.7**	4.5
Answer type	Entity (open)	177	79	33	60	**62**	4.7	2.9	4.2	**4.4**
	Entity (closed)	5	20	20	0	**60**	3.6	2.0	4.2	**4.8**
	Number	33	61	48	58	**61**	4.7	2.8	4.5	4.5
	Boolean	90	87	37	51	**63**	4.8	2.5	3.7	**4.6**
Answer cardinality	0 (unanswer.)	35	97	89	**97**	94	4.4	3.6	4.0	**4.5**
	1	281	80	35	56	**63**	4.7	2.7	4.1	**4.5**
	More	51	69	22	37	**47**	4.7	3.1	4.1	**4.3**
Nb. target variables	0 (⇒ `ASK`)	90	87	37	51	**63**	4.8	2.5	3.7	**4.6**
	1	217	75	35	59	**61**	4.6	2.9	4.3	**4.4**
	2	25	84	0	8	**44**	4.9	3.0	3.6	**4.2**
Dialogue context	Self-sufficient	144	79	22	55	**59**	4.7	2.5	4.2	**4.4**
	Coreference	154	80	44	57	**66**	4.7	3.1	4.1	**4.5**
	Ellipsis	90	71	32	46	**60**	4.7	2.6	3.9	**4.5**
Meaning	Meaningful	302	77	27	49	**57**	4.8	2.7	4.1	**4.5**
	Non-sense	30	97	93	97	97	4.3	3.5	3.8	**4.3**

Table 13: Results of the human evaluation for each type of query. The darker, the worse. Bold refers to the best non human result.