Assessing the Generalization Capabilities of Neural Machine Translation Models for SPARQL Query Generation

Samuel Reyd(✉) and Amal Zouaq

Laboratoire LAMA-WeST, Polytechnique Montréal, Montreal, Canada
{samuel.reyd,amal.zouaq}@polymtl.ca
http://www.labowest.ca/

Abstract. Recent studies in the field of Neural Machine Translation for SPARQL query generation have shown rapidly rising performance. State-of-the-art models have reached almost perfect query generation for simple datasets. However, such progress raises the question of the ability of these models to generalize and deal with unseen question-query structures and entities. In this work, we propose copy-enhanced pre-trained models with question annotation and test the ability of several models to handle unknown question-query structures and URIs. To do so, we split two popular datasets based on unknown URIs and question-query structures. Our results show that the copy mechanism effectively allows non-pre-trained models to deal with unknown URIs, and that it also improves the results of some pre-trained models. Our results also show that, when exposed to unknown question-query structures on a simple dataset, pre-trained models significantly outperform non-pre-trained models, but both non-pre-trained and pre-trained models have a considerable drop in performance on a harder dataset. However, the copy mechanism significantly boosts the results of non-pre-trained models on all settings and of the BART pre-trained model, except for the template split on LC-QuAD 2.0 dataset.

Keywords: SPARQL query generation · Generalization · Unknown templates · Unknown URIs · Out-of-vocabulary problem

1 Introduction

The Knowledge Base/Graph Question Answering (KGQA) task has seen recent progress thanks to the development of Neural Machine Translation (NMT) and encoder-decoder architectures. Instead of directly generating or extracting the answer to a question, NMT models generate a SPARQL query that is run against the Knowledge Base (KB) to retrieve the answer.

Addressing English to SPARQL translation with NMT techniques can be done by considering SPARQL as another language. However, there are specific challenges that arise from the fact that SPARQL is a formal language, and substituting a word with another, even one that might appear in a similar context, might completely change the sense of the query and generate totally different answers. Notably, the most challenging issue of classic encoder-decoder models is the Out Of Vocabulary (OOV) problem. In general, for traditional NLP models, the vocabulary is fixed in advance and any word that appears in a new input or that is expected in the predicted output that has not been seen during training cannot be taken into account or be generated. This is a very real problem in SPARQL generation since most large KBs include many resources that are very rarely used and are not available to be learned by models. Recent studies address this problem by annotating the KB elements in the question and allowing models to transfer tokens or sequences of tokens from the question to the predicted query [1,10]. Even though modern technics improve the URI generation capabilities of the traditional models, this point remains one of the most challenging parts of the task, especially in the case of unseen or rare URIs. Even though some studies [10] specifically try to evaluate how their approach handles this problem, none systematically evaluate the capabilities of models to deal with unknown URIs.

Another challenge of neural SPARQL translation is the ability of state-of-the-art models to generate question-query structures unseen in training. To our knowledge, none of the previous approaches specifically evaluated this aspect. This challenge is also linked to the way most datasets are generated. Since real-world data is rarely available, most datasets feature automatically generated question-query pairs that are built by filling placeholders inside structural templates with specific KB URIs. This allows models to rely on structures that are limited. Even though some datasets feature questions that are reformulated by human experts, the training data still encourage the model to fit a finite set of question-query structures. To our knowledge, no study addresses the issue of generating queries that do not correspond to the structures seen during training.

We will therefore address the following research questions:

- How do NMT models perform with unknown URIs?
- How do NMT models perform when they face unknown question-query structures?

Our contributions are: 1) to evaluate the capabilities of modern NMT approaches (Transformers [21], ConvSeq2Seq [7], BART [15] and T5 [17]) to handle unknown URIs and question-query structures; 2) to highlight challenges in NMT-based approaches and propose an algorithm to split datasets for better generalization properties of the test set and 3) to test the ability of pretrained models enhanced with a copy mechanism to handle unknown URIs and question-query structures.

2 Related Work

KGQA with NMT Approaches. KGQA has been approached by many methods but recent studies have focused on deep learning NMT approaches. For instance, Yin et al. [22] used encoder-decoder architectures [19] to translate English questions into SPARQL queries and outperformed traditional neural architectures using a Transformer model [21] and a ConvSeq2Seq model [7].

Hirigoyen et al. [10] proposed a copy mechanism inspired by CopyNet [8] and PGN [18] to copy URIs from questions to SPARQL queries. Their copy mechanism distinguishes itself from former copy approaches by introducing a specific knowledge base vocabulary whose tokens are hidden from the encoder-decoder block and that are the only ones that the copy mechanism can act on. By annotating the questions with the URIs found in the golden queries, this method allows improved performances by a great margin and improves OOV token handling.

The most recent approaches, such as Debayan et al. [1], used pre-trained models, namely BART [15] and T5 [17]. Debayan et al. [1] also annotate questions by placing URIs at the end of the question next to their label from the KB. They also propose a model based on an LSTM [11] encoder-decoder with BERT [5] token embeddings enhanced by a copy block as defined by PGN [18]. However, they report significantly lower results with this method than with their pre-trained models.

Generalization. Generalization in sequence-to-sequence (seq-to-seq) models has been extensively studied, and while models learn generalization as an implicit set of patterns, they may not learn explicit rules required for compositional generalization. According to Baroni [2], neural machine translation (NMT) models can adapt to unseen data, but they lack the cognitive ability to understand explicit rules. Compositional generalization refers to the ability to understand rules used for meaning construction based on how elements are composed. For instance, understanding "jump" and "jump twice" should allow generalization to "walk twice". However, NMT models struggle with explicit composition rules, as supported by experiments on the SCAN dataset [14].

In the field of SPARQL query generation from English questions, Keysers et al. [13] further specify the concept of compositional generalization by measuring the distribution of the Direct Acyclic Graphs (DAG) used for the generation of the question query pairs. They create an English-to-SPARQL dataset with appropriate compositional generalization properties and show a negative correlation between seq-to-seq models' performance and the level of compositional generalization. Gu et al. [9] defined three levels of generalization. These definitions are based on "logical forms", i.e. the underlying structures in the question-query pair. For instance, logical forms in another study [12] include the set of KB classes and properties that appear in the query, the comparison operator in the queries (e.g. $=$, $<$, etc.) and the SPARQL keyword `COUNT`. Based on these logical forms, each test set data d_0 can be associated to a level of generalization according to the following definitions. 1) the I.I.D. level: there is a question-query pair in the train set that has the same set logical forms as d_0. 2) the compositional level: all the logical forms in d_0 belong to the set of all logical forms in the train set, but no question-query pair of the train set had the same exact set of logical forms. 3) The zero-shot level: there is at least one logical form in d_0 that does not belong to the set of all logical forms in the train set.

Jiang et al. [12] conducted a survey of the KGQA datasets and how they comply with these levels. They also show that the models they tested (including former approaches [3,4] and BART [15]) do not generalize well on these new splits. In this work, we address specific NMT difficulties that can be interpreted as variations of these generalization levels. For instance, we aim at evaluating how models deal with unknown URIs, which is equivalent to the zero-shot level when considering that all URIs are logical forms. This is also equivalent to some definition of compositional generalization as proposed by Baroni [2] where a model must understand the meaning of new symbols as it understood how to compose with other symbols. Similarly, Hirigoyen et al. [10] attempted to evaluate how their models dealt with unknown URIs by generating test sets that do not share any URIs with the training sets. However, these test sets were very limited in size (250 entries). In this work, we test the ability of modern NMT models on popular SPARQL generation datasets and include pre-trained language models enhanced by a copy mechanism.

3 Methodology

3.1 Task and Data

The task of SPARQL query generation for KGQA uses pairs (called entries) of sequences, composed of English questions and expected SPARQL queries. For example: *Question:* "what is the office of richard coke ?" / *Query:* `select distinct ?uri where {dbr:Richard_Coke dbp:office ?uri }`.

All the datasets that we consider use automatically generated entries based on global templates. A global template is composed of a question template and a query template that have matching placeholders that are later filled with specific KB URIs. We refer distinctly to question templates, query templates, and global templates since some question templates or query templates might be shared by several global templates. The placeholder can be filled by any KB element, including URIs and literals. URIs can be resources, or refer to properties and classes. For instance, the above example was generated by the following template: *Question template*: "what is the `<1>` of `<2>` ?" / *Query template*: `select distinct ?uri where {<2> <1> ?uri}`.

We use three types of question annotation. 1) The raw questions are presented in the original datasets. 2) The tagged questions, where we use the global templates to replace the natural language mentions with their corresponding KB URIs, found in the corresponding placeholders in the SPARQL queries. This tagging methodology is inspired by [10]. The above example has the following tagged question: "what is the `dbp:office` of `dbr:Richard_Coke` ?" 3) The tag end questions, where each KB element is randomly placed next to its label at the end of the question, separated by `<sep>` tokens. This tagging methodology is inspired by [1]. The above example has the following tag-end question:

"what is the office of richard coke ? `<sep>` `dbr:Richard_Coke` richard coke `<sep>` `dbp:office` office"

3.2 NMT Models

Base Architectures. Following [22] and [10], we experimented with ConvSeq2Seq [7] and Transformer [21] models as well as a copy-enhanced architecture [10]. Similarly to [1], we also experimented with T5-small [17] and BART-base [15] as pre-trained models. Finally, we proposed and tested a copy-enhanced version of the pretrained language models.

Copy Mechanism. As explained in [10], adding a copy-mechanism on top of a non-pre-trained encoder-decoder can improve its performances by a great margin. The copy mechanism from [10] that we use is designed specifically to address the main difficulty of the task of SPARQL query generation, as it helps models to put the right URIs in the query. It is a block added on top of the raw encoder-decoder model. This copy mechanism is well-suited for the task of SPARQL query generation because it allows to not consider the KB elements in the model's vocabulary. For non-pretrained models, this means a significant reduction in the size of the vocabulary. For pre-trained models, it allows avoiding the tokenization of URIs and making "spelling" mistakes when generating the URIs.

We consider an annotated question $w_{0:m}$ with tokens from the natural language vocabulary W and the KB vocabulary K. The expected query $\hat{q}_{0:n}$ is composed of tokens from the SPARQL vocabulary S and the shared tokens from K. When producing a new token, the model will choose to generate a token from the SPARQL vocabulary or to copy a KB element (URI, literal) that belong to the KB vocabulary from the annotated question.

To perform this choice between generating and copying, we first mask the KB token from the annotated question before feeding it to the encoder-decoder. Then, the copy block will compute a copy probability by applying a linear transformation to the decoder logits. This copy probability will then be used to weight the copy distribution and the generation distribution. The generation distribution is obtained by applying a softmax on the decoder logits. This distribution therefore only covers the SPARQL vocabulary. The copy distribution is computed by applying a softmax on the cross-attention weights (from the last attention layer) between the token to be produced and the masked tokens of the question. This distribution therefore only covers the KB elements that appear in the question.

3.3 Evaluation of Generalization Capabilities

Generalization Properties of a Dataset. To evaluate the generalization capabilities of our models, we define generalization properties of our datasets and their split based on some characteristics of the question-query pairs. The characteristics of the entries are simply elements of interest, such as the global template or the URIs that the entry features. The generalization properties are assessments on the distribution of these characteristics with respect to the dataset split. More formally, let D be a dataset, let D_1 be a reference subset of this dataset (usually the test set), and $\overline{D}_1$ be the rest of the dataset (usually the train set). For all $d \in D$ let $c(d)$ be the set of characteristics of the data (in our case either the URIs or the global template of a question-query pair), we say that a generalization property gp holds if any data point d_1 of D_1 has at least one characteristic that does not appear in $\overline{D}_1$, i.e.

$$gp(D, D_1) \equiv \forall d_1 \in D_1 \; \exists c_1 \in c(d_1) : c_1 \notin \cup_{d \in \overline{D}_1}(c(d))$$

.

We then define two properties of our datasets that are useful for assessing the generalization capabilities of NMT models.

We first evaluate the problem of unknown URIs, since generating the right URIs is necessary to produce correct queries. This generalization capability is very important because real knowledge bases cover very large types of subjects and contain a very large number of URIs. Even though the datasets might aim at covering as many areas and concepts as possible, none may feature enough data to contain multiple examples of each possible URI. Hence, our first generalization property is: *a test set must feature only queries that have at least one URI that is unseen during training.*

We then aim to evaluate the generation of queries with structures that differ from the ones seen during training. The handling of unknown question-query structures is particularly important because it is unclear if models really generate queries based on the sentence semantics, or if they simply map them to known structures of queries. Hence, our second generalization property is: *a test set must feature only question-query pairs that were generated with different global templates than the ones used in the train set.*

Generation of Test Sets with Generalization Properties. We first partition our dataset into subsets that we call groups, such that any set of groups can constitute a reference subset of the dataset that has a generalization property. More formally, given a dataset D, a partition of this dataset into groups $\{g_1, ..., g_k\}$ and a generalization property gp, for any set of these groups $\mathcal{G} \subset \{g_1, ..., g_k\}$ we have that $gp(D, \bigcup \mathcal{G})$ holds. We then define the procedure to build these groups for each of our two generalization properties.

Groups for Unknown Templates Split. In the case of unknown question-query templates (aka unknown template split), the groups are composed of question-query pairs that share the same question-query template. These groups can then be assigned to the train set or test set. This ensures that there is no overlap.

Groups for Unknown URI Split. The URIs are exponentially distributed amongst the question-query pairs. Therefore, most pairs share common URIs and most pairs feature rare URIs that are not shared by many other pairs. The minimum frequency of a URI to be considered rare is an experimental hyperparameter of the algorithm. We explain how we choose it in Sect. 4.1. We first set aside the few question-query pairs in the dataset D that only feature common URIs, which are by default assigned to the train set. Thus our dataset $\tilde{D}$ contains only question-query pairs that feature at least one rare URI. We then define a graph where each node is a question-query pair of $\tilde{D}$, and there is an edge between two nodes if the two question-query pairs share a rare URI. We define our groups as the connected components of this graph. If we take any set of these groups, all the question-query pairs within these groups will feature at least one URI that does not appear in the rest of the groups. Indeed, if we take any rare URI in any entry of these groups, it cannot appear in other groups because if it was the case, there would have been an edge between entries from two different groups, which can't append since they are connected components of our graph.

Desired Number of Question-Query Pairs to Assign to the Train Set. For both settings, we aim at assigning 80% of the data to the train set and 20% to the test set. We define N as the desired number of question-query pairs in the groups that are assigned to the train set. In the case of the unknown template split, $N = 0.8 \times |D|$ with D being the dataset we want to split. In the case of unknown URIs, since we reserved the question-query pairs without rare URIs to the train set, we get $N = 0.8 \times |D| - (|D| - |\tilde{D}|) = |\tilde{D}| - 0.2 \times |D|$.

Split Procedure. We then execute the following procedure. We initialize the train set $train$ and the test set $test$ as empty sets. We then iterate over each group g. If the size of $train$ is larger than N, we assign the group to $test$. Reversely, if the size of $test$ is larger than $|D| - N$ we assign the group to $train$. Finally, if neither of these conditions is true, we compute a probability $p = \frac{N - |train|}{|D| - |train| - |test|}$ and assign the group to $train$ with probability p or to $test$ with probability $1 - p$.

By design of the groups, this ensures that the test set has the intended generalization properties. We define a metric $\delta = \frac{|N - |train||}{|D|}$ that we wish to minimize. In our procedure, once the train or test set has reached or exceeded its desired size, no more groups can be assigned to it. Thus, we obtain $\delta < |g_f|/|D|$ where g_f is the last group to be assigned. We run the procedure multiple times and keep the run with the lowest δ, i.e. the one where the last assigned group is the smallest.

4 Experiments

4.1 Data

We experimented with a simpler and a harder dataset, namely LC-QuAD 1.0 [20] and LC-QuAD 2.0 [6]. They both provide question-query pairs from English to SPARQL. We use processed versions that include the raw questions and queries, as well as two annotated versions of the questions.

Table 1 reports the number of URIs, global templates, and question query pairs of the entire dataset and each of its subsets, as well as the number of unseen elements in the test set. As can be noticed, while many URIs are unknown in LC-QuAD 2.0, global templates are all seen in training. Question-query pairs refer to specific questions and queries. Those in the test set have not, by definition, been seen in training.

Table 1. Number of URIs, global templates and question-query pairs in the datasets for the original split

	Total	Train	Validation	Test	Unseen
All URIs					
LC-QuAD 1.0	4751	4150	1068	1065	318
LC-QuAD 2.0	31018	25064	4978	9992	6724
Global templates					
LC-QuAD 1.0	35	35	32	31	0
LC-QuAD 2.0	30	30	30	30	0
Entries					
LC-QuAD 1.0	5000	4000	500	500	500
LC-QuAD 2.0	30225	21761	2418	6046	6046

We chose an 80-10-10 split proportion for the train, validation, and test sets. We used the split methodology described in Sect. 3.3 to get an 80–20 division of our datasets [6,20] with the generalization properties introduced in the same section. We then uniformly split the 20% into 10% for the validation set and 10% for the test set.

We obtained our unknown template splits in seconds by running our split procedure around a hundred times and keeping the split with lowest δ as explained in Sect. 3.3. For the unknown URI split, we set the minimum frequency for a URI to be rare at 5. Figure 1 shows that we have the most connected component and the largest size of $\tilde{D}$ ($\tilde{D}$ being the set of entries with at least one rare URI, see Sect. 3.3) around a minimum frequency of 5.

Finding the graph's connected components took less than a second for LC-QuAD 1.0 [20] and around 1.5min for LC-QuAD 2.0 [6]. We then ran the split procedure around a hundred times, which took less than a second. We get $\delta = 6.6e^{-5}$ for the template split on LC-QuAD 2.0 [6] and $\delta = 0$ for the template split on LC-QuAD 1.0 [20] and for the URI split on both datasets [6,20].

Table 2 reports the number of URIs, global templates, and question query pairs of the train/val/test sets obtained for the unknown URIs split and the unknown template split. We can observe that we have fewer unknown URIs in the test set of the unknown URI spit than in the test set of the original split. This is because the original split did not use an 80-10-10 split and therefore had around twice as many entries in its test set. However, our new split ensures that 100% entries of the test feature at least one unknown URI which is not the case in the original split.

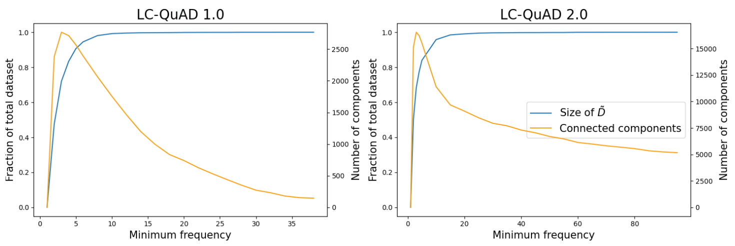

Fig. 1. Size of $\tilde{D}$ (see Sect. 3.3) and number of connected components as we raise the minimum frequency for considering a URI rare

Table 2. Number of URIs, global templates, and question-query pairs in the datasets for our splits

	Total	Train	Val	Test	Unseen	Train	Val	Test	Unseen
		Unknown template split				Unknown URIs split			
		All URIs							
LC-QuAD 1.0	4751	4183	925	912	282	3872	909	948	535
LC-QuAD 2.0	31018	27986	4403	4429	1619	25121	4738	4708	3437
		Global templates							
LC-QuAD 1.0	35	31	4	4	4	35	32	30	0
LC-QuAD 2.0	30	24	6	6	6	30	30	30	0
		Entries							
LC-QuAD 1.0	5000	4000	500	500	500	4000	500	500	500
LC-QuAD 2.0	30225	24178	3023	3024	3024	24180	3022	3023	3023

4.2 Models Training and Evaluation

Our non-pre-trained model architectures follow the ones of [22] and [10] in terms of the number of layers (6 for Transformer and 15 for ConvSe2Seq), and number of hidden units (1024 for Transformer and 512 for ConvSeq2Seq). The training methodology is also the same with similar optimizer (ADAM for Transformer and SGD for ConvSeq2Seq), learning rates (0.0005 for Transformer and 0.5 for ConvSeq2Seq) and dropout (0.3 for Transformer and 0.2 for ConvSeq2Seq). Our pre-trained models are T5-small and BART-base following [1]. We also adopt their training parameters, using the ADAM optimizer for both models with a learning rate of 0.000015 for BART and 0.0015 for T5, and a polynomial decay schedule with warmup.

We ran our models for a fixed number of epochs three times, keeping the model with the best validation loss at each run, and report the mean of the three best models. The number of epochs and the batch size are fixed based on our physical device's abilities. For non-pre-trained models, we used 500 epochs and a batch size of 32 for LC-QuAD 1.0 [20] and 150 epochs and a batch size of 16 for LC-QuAD 2.0 [6]. For pre-trained models, we used 200 epochs and a batch size of 16 for LC-QuAD 1.0 [20] and 50 epochs and a batch size of 8 for LC-QuAD 2.0 [6].

During the evaluation, we generate the outputs greedily and compute several evaluation measures, namely BLEU score [16], answer accuracy and F1 score. We first compare the predicted queries to the gold queries using the BLEU score. We also run the gold queries against a 2016 dump of DBPedia-based endpoint for LC-QuAD 1.0 [20] and on the current public endpoint for LC-QuAD 2.0 [6], and we only keep queries that return non-empty answers. We then run our predicted queries on the same endpoints and compare expected answers with predicted answers using answer accuracy, which measures if the two sets of answers are identical and using the F1-score.

5 Results

We report results rounded to integers for an easier comparison between tables.

5.1 Original Results

All results for the original split are reported in Table 3.

For non-pre-trained models, we can notice that the copy mechanism managed to bring the performance for LC-QuAD 1.0 [20] close to 100%. For LC-QuAD 2.0 [6], even with copy, the task remains really challenging. Moreover, we can see that in most settings, the Transformer architecture struggles with LC-QuAD 2.0 [6].

Contrary to non-pre-trained models, the original results of pre-trained models are quite heterogeneous. Both T5 [17] and BART [15] have low results on raw questions (even though they are better than any setting of non-pre-trained models without the copy mechanism). BART [15] without the copy mechanism is the only pre-trained architecture that has performances noticeably lower than ConSeq2Seq with the copy mechanism. In the case of LC-QuAD 1.0 [20] in its original split, both BART [15] with the copy mechanism and T5 [17] with and without the copy mechanism perform well. In the case of LC-QuAD 2.0 [6], T5 [17] performs well without the copy mechanism, whereas BART [15] performs well only with the copy mechanism.

5.2 Unknown URIs Split

All results for the unknown URIs split are reported in Table 4.

The results for non-pre-trained models show very clearly the impact of the copy mechanism on the handling of unknown URIs and also clearly highlight that our split creates a real difficulty for non-pre-trained models compared to the original dataset.

Without the copy mechanism, we can note that no non-pre-trained model reaches 10% of accuracy on LC-QuAD 1.0 [20] nor 2% on LC-QuAD 2.0 [6]. Even

Table 3. Results for the original split

		No copy			Copy			No copy			Copy		
		BLEU	Acc	F1	BLEU	Acc	F1	BLEU	Acc	F1	BLEU	Acc	F1
		Transformer						Conv					
LC-QuAD 1.0	raw	69	33	37	/	/	/	69	23	27	/	/	/
	tagged	82	41	44	98	95	95	78	28	32	98	95	95
	tag end	74	38	41	49	1.9	1.9	75	25	28	91	77	78
		BART						T5					
	raw	83	59	62	/	/	/	77	42	46	/	/	/
	tagged	96	85	85	99	97	97	98	96	96	98	96	96
	tag end	96	85	85	98	96	96	98	95	95	97	93	93
		Transformer						Conv					
LC-QuAD 2.0	raw	57	0.9	1.1	/	/	/	76	10	11	/	/	/
	tagged	59	1.5	1.6	83	69	69	76	8.8	10	88	69	70
	tag end	66	1.4	1.7	57	2.2	2.2	78	14	15	91	66	66
		BART						T5					
	raw	70	2.2	2.3	/	/	/	79	13	13	/	/	/
	tagged	88	80	80	89	72	73	92	87	87	89	72	72
	tag end	84	67	67	95	85	85	90	85	85	87	58	58

though the tagged questions lead to slightly higher performance, the annotation impact is much less noticeable than for other settings.

On the contrary, with the copy mechanism, we clearly notice that the performance of non-pre-trained models (based on all metrics) using tagged questions is much higher than without the copy mechanism. We reach and even outperform the results from the original split. This suggests that the unknown URIs are handled, particularly using tagged questions. Indeed, the introduction of unknown URIs in each test entry without any F1-score loss (as seen by comparing Table 3 and Table 4) suggests that the errors made might not be caused by unknown URIs. The tag end setting does not however help the models.

For pre-trained models, with annotated questions, the results remain similar to the original split for T5 [17]. BART [15] loses some F1 score points on LC-QuAD 2.0 [6]. However, we see a great increase in performance between raw data and annotated questions. This suggests that with question annotation, pre-trained models are able to deal with unknown URIs to a good extent. We can also see that T5 [17] with the copy mechanism slightly increases its performance on LC-QuAD 1.0 [20].

5.3 Unknown Templates Split

All results for the unknown templates split are reported in Table 5.

For non-pre-trained models, we can observe that models without the copy mechanism consistently suffer a huge drop in performance compared to the original split. For LC-QuAD 1.0 [20], we still see a consistent increase in performance with question annotation but the answer accuracy still remains below 55%. For

Table 4. Results for non-pre-trained models on the unknown URIs split

		No copy			Copy			No copy			Copy		
		Bleu	Acc	F1	Bleu	Acc	F1	Bleu	Acc	F1	Bleu	Acc	F1
		Transformer						Conv					
LC-QuAD 1.0	raw	59	7.0	10	/	/	/	64	6.9	10	/	/	/
	tagged	70	8.0	12	98	95	95	71	8.4	12	98	96	96
	tag end	67	6.1	8.6	46	0.2	0.2	70	6.9	9.6	86	60	62
		BART						T5					
	raw	77	38	41	/	/	/	72	24	26	/	/	/
	tagged	96	85	85	99	96	96	98	95	95	99	97	98
	tag end	95	83	83	96	93	94	98	94	94	98	96	96
		Transformer						Conv					
LC-QuAD 2.0	raw	61	0.3	0.5	/	/	/	75	1.8	2.7	/	/	/
	tagged	64	0.3	0.4	86	72	73	75	2.0	3.0	90	72	73
	tag end	62	0.4	0.6	55	1.8	1.8	74	1.9	3.0	88	53	53
		BART						T5					
	raw	68	0.5	0.6	/	/	/	74	0.7	0.9	/	/	/
	tagged	87	72	72	86	62	62	93	87	88	85	68	68
	tag end	86	71	71	92	78	78	91	85	86	92	77	78

LC-QuAD 2.0 [6] with Transformer, the results are even lower than in the original split and the answers are almost always wrong. For ConvSeq2Seq, the model has better performances but they remain low.

However, we can observe a noticeable impact of the copy mechanism on the non-pre-trained models' results. Even though they remain below those of the original split, they are much better than without the copy mechanism. Moreover, compared to the original split, the BLEU score is notably lower especially if we also consider the answer accuracy. This would suggest that the structure of the predicted queries does not always match the structure of the expected ones while still providing correct answers. Explanations of why this might happen are given in Sect. 6. Overall, non-pre-trained models seem to really struggle with global templates unseen during training.

Contrarily to non-pre-trained models, pre-trained models appear to handle unknown templates for simple datasets.

For LC-QuAD 1.0 [20], BART [15] and T5 [17] without the copy mechanism both remain consistent in terms of answer metrics, while they show a significant drop in BLEU score compared to the original split. However, we can observe a huge drop in performance for T5 [17] with copy on both question annotation settings (tagged, tag-end) and for BART [15] with copy on the tag-end questions. Yet, BART [15] with copy on tagged questions demonstrates impressive and almost perfect answers metrics.

In the LC-QuAD 2.0 [6] case, we can note that the results are low. No model reaches 50% accuracy. In this case, both models perform better without the copy mechanism.

Table 5. Results for the unknown template split

		No copy			Copy			No copy			Copy		
		Bleu	Acc	F1	Bleu	Acc	F1	Bleu	Acc	F1	Bleu	Acc	F1
		Transformer						Conv					
LC-QuAD 1.0	raw	37	6.9	8.0	/	/	/	43	17	19	/	/	/
	tagged	36	6.3	6.4	50	24	24	46	19	22	50	41	41
	tag end	42	11	12	37	2.8	2.8	50	18	20	60	55	55
		BART						T5					
	raw	62	41	42	/	/	/	65	52	53	/	/	/
	tagged	79	87	88	63	99	99	80	92	92	57	50	50
	tag end	80	90	90	61	42	43	75	78	79	66	51	51
		Transformer						Conv					
LC-QuAD 2.0	raw	43	0.2	0.3	/	/	/	51	4.1	4.7	/	/	/
	tagged	39	0.1	0.2	64	32	33	52	1.4	2.0	63	34	34
	tag end	41	0.2	0.2	37	0.3	0.3	55	2.7	2.9	57	13	14
		BART						T5					
	raw	50	0.4	0.4	/	/	/	59	1.1	1.2	/	/	/
	tagged	70	47	47	65	34	34	69	48	48	61	14	14
	tag end	52	21	22	62	12	12	68	47	47	62	7.9	8.2

6 Discussion

We showed that classic NMT approaches cannot handle unknown URIs or unknown question-query structures. On the contrary, adding a copy mechanism or using pre-trained models combined with question annotation allows the handling of unknown URIs since having all test instances featuring unknown URIs almost did not lower the performances. Indeed, the results of pre-trained models on the unknown URIs split remain roughly similar compared to the original split with tagged questions (-2 F1 point on average). Significant performance gains are obtained only with tagged questions on the unknown URIs split, as we observe a raise of 67.9 F1 points on average when using tagged questions instead of raw questions for pre-trained models.

We also showed that the copy mechanism or the use of pre-trained models, and sometimes specifically the combination of both, can allow models to deal with unknown question-query structures. Yet, this type of generalization remains a challenge for SPARQL query generation with NMT as our results don't show consistent handling across models, datasets, and data annotation.

Low Performances for Non-pre-trained Models on tag-End Questions in the Unknown URIs Split. We observed that the copy mechanism allowed the non-pre-trained models to reach the same performance on the unknown URIs split as on the original split for tagged questions but not on tag-end questions. This can be explained by the fact that the copy mechanism uses the label of the URI, at the end of the question to map it to its position in the question and to its position in the query. But in these settings, there are many unknown URIs, and thus a very high probability that the label reported next to the URI is also an unknown token. Even though the copy mechanism is able to overcome unknown URIs, it needs an anchor to map the position of the given URI to the position of its natural language mention in the question, which is often impossible in the tag-end questions.

Drop of BLEU Score for the Unknown Template Split. We observed that in most cases, the margin between the BLEU score and answer-based scores is lowered if not reversed on the unknown template splits. This behavior suggests that the models predict queries that do not match the expected ones but still produce the gold answers.

We found a significant example of such behavior in the case of BART with the copy mechanism on LC-QuAD 1.0 [20] with tagged questions. When we look at the generated queries, we can see that they always follow the structure of a query template in the training set. We conclude that the model has integrated the task of mapping a question template to a query template together with placeholder filling. When it is given a question from the test set, it tries to associate the unknown structure to a known one from the training set and then generates the corresponding query. An example of outputs from BART with copy and tagged questions from the most common global template in the test set of LC-QuAD 1.0 [20] can be found in Table 6. For instance, for the test question "what is the `dbp:hubs` of `dbr:Cascade_Airways`?" the model predicted `select distinct ?uri where { dbr:Cascade_Airways dbp:hubs ?uri . dbr:Cascade_Airways dbp:hubs ?uri }` instead of `select distinct ?uri where { dbr:Cascade_Airways dbp:hubs ?uri }`.

Table 6. Example of outputs generated by BART for the most common global template of the test set (376 occurrences) which is *Question:* what is the <1> of <2> ? / *Query:* `select distinct ?uri where { <2> <1> ?uri }` (one run of BART tagged copy on LC-QuAD 2.0)

Generated structure from: what is the `<1>` of `<2>` ?	Training global template it might come from
`select distinct ?uri where { <mask> <mask> ?uri . <mask> <mask> ?uri }` (298 occurrences)	- what is the <1> of the <2> and <3> ? - `select distinct ?uri where { <2> <1> ?uri. <3> <1> ?uri }`
	- who is the <1> of the <2> and <3> of the <4> ? - `select distinct ?uri where { <2> <1> ?uri. <4> <3> ?uri }`
`select distinct ?uri where { <mask> <mask> ?uri . <mask> <mask> ?uri . }` (77 occurrences)	- what is the <1> of the <2> and <3> ? - `select distinct ?uri where { <2> <1> ?uri. <3> <1> ?uri }`
	- who is the <1> of the <2> and <3> of the <4> ? - `select distinct ?uri where { <2> <1> ?uri. <4> <3> ?uri }`
`select distinct ?uri where { ?uri <mask> <mask> . }` (1 occurrence)	- what are the <0> whose <1> is <2> ? - `select distinct ?uri where {?uri <1> <2> }`

Challenging Cases. Either thanks to the copy mechanism, to pre-trained models or to the combination of both, we manage to obtain at least one model that reaches above 80% of F1 score on almost each split of each dataset. The only exception is the case of the unknown template split for LC-QuAD 2.0. In this case, no model manages to reach above 48% of F1 score. We can notably see a very strong drop of performance from the unknown template split of LC-QuAD 1.0 compared to the unknown template split of LC-QuAD 2.0. In particular, we can observe that BART's performance with the copy mechanism and tagged questions decrease from 99% of F1 score to 34% of F1 score. On both LC-QuAD 2.0 and LC-QuAD 1.0, we note that the model generates queries that match query templates from the training set that are associated to training questions close to the test questions. In the case of LC-QuAD 1.0, this allowed to generate queries that return the expected answers due to equivalent queries with different structures (see the above example) but it is not the case for LC-QuAD 2.0.

Table 7 shows an example of how BART generated incorrect queries for questions with a specific structure and used the same SPARQL templates as those in the training set.

Table 7. Example of how BART with the copy mechanism and tagged questions handles an unknown template in LC-QuAD 2.0

Question structure from the test set: How many <1> are by <2>? *Example:* how many child are by gaia ?
Expected query structures for this question structure: `select ( count ( ?sub ) as ?value ) { ?sub <1> <2> }` *Example:* `select ( count ( ?sub ) as ?value ) { ?sub wdt:P40 wd:Q93172 }`
Predicted query structures for this question structure: `select ( count ( ?obj ) as ?value ) { <2> <1> ?obj }` *Example:* `select ( count ( ?obj ) as ?value ) { wd:Q270503 wdt:P400 ?obj }`
Question template associated to this query template in the train set: how many <1> are for <2> ? *Example:* how many platform are for tomb raider ?

Standard Deviation of Performance between Runs with the Copy Mechanism. For each result that we report, we averaged the results of three different runs where the models are trained and evaluated with different random seeds. We noted that there is, in some cases, a significant standard deviation between the runs when we use models with the copy mechanism.

Complexity of Finding Unknown URIs Groups. Despite the speed of the split procedure, our methodology for unknown URI splits has a higher time complexity. Indeed, it requires finding the connected components in the graph, which is a quadratic process in the size of the dataset.

7 Conclusion and Future Work

In this study, we defined two major issues of the NMT approach of SPARQL query generation from English questions. We first show how they are related to common definitions of generalization in the context of SPARQL query generation. We then presented a split algorithm to obtain datasets that test these challenges in their train/test distribution: the unknown template split and the unknown URI split. Finally, we tested pre-trained models and copy-enhanced models with question annotation on a simple and a hard dataset. We also compared these results to non-pre-trained models.

We showed that unknown templates are very often an issue for model training and testing since most results are low and only very specific combinations of question annotation and model architecture (BART [15] with copy and tagged questions and T5 [17] without copy) allow good results on LC-QuAD 1.0 [20]. Moreover, even when models generate queries that have correct answers, they often follow structures matching those in the training set, which is an important limitation. However, we showed that even though the performance remains low, the copy mechanism allowed a significant improvement for non-pre-trained models on LC-QuAD 2.0 [6]. We also showed that pre-training and question annotation or usage of the copy mechanism allows the handling of unknown URIs contrary to non-pre-trained models (without copy).

Our future work will include other criteria to evaluate the ability of current natural language to SPARQL datasets to test generative models. For instance, LC-QuAD 1.0 [20] and LC-QuAD 2.0 [6] both include reformulated questions that are produced by humans, which can constitute a good test set for models trained on template questions. We also plan to consider other splitting criteria that would show how models generalize to harder data, based on characteristics such as the length of the query or the question, the number of placeholders, or the number of triples in the query. Finally, we plan to enhance the performance of the split algorithm for unknown URIs, by considering other graph algorithms to find groups.

Supplemental Material Statement: The code and data for this paper can be found at this link[1].

Acknowledgements. This research has been funded by the NSERC Discovery Grant Program. The authors acknowledge support from Compute Canada for providing computational resources. We would like to thank Karou Diallo for setting the DBpedia SPARQL endpoint used in this paper.

[1] Link to our GitHub.

References

1. Banerjee, D., Nair, P.A., Kaur, J.N., Usbeck, R., Biemann, C.: Modern baselines for SPARQL semantic parsing. In: Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Shane Culpepper, J., Kazai, G. (eds.) SIGIR 2022: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022, pp. 2260–2265. ACM (2022)
2. Baroni, M.: Linguistic generalization and compositionality in modern artificial neural networks. Philos. Trans. Royal Soc. B: Biol. Sci. **375**(1791), 20190307 (2019)
3. IAIS bFraunhofer. Knowledge graph question answering using graph-pattern isomorphism. In: Further with Knowledge Graphs: Proceedings of the 17th International Conference on Semantic Systems, 6–9 September 2021, Amsterdam, The Netherlands, vol. 53, p. 103. IOS Press (2021)
4. Chen, Y., Li, H., Qi, G., Wu, T., Wang, T.: Outlining and filling: hierarchical query graph generation for answering complex questions over knowledge graphs. arXiv preprint arXiv:2111.00732 (2021)
5. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics (June 2019)
6. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: a large dataset for complex question answering over Wikidata and DBpedia. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 69–78. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_5
7. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, vol. 70 , pp. 1243–1252. PMLR (2017)
8. Gu, J., Lu, Z., Li, H., Li, V.O.K.: Incorporating copying mechanism in sequence-to-sequence learning. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, 7–12 August 2016, Berlin, Germany, Volume 1: Long Papers, pp. 1631–1640. The Association for Computer Linguistics (2016)
9. Gu, Y., et al.: Three levels of generalization for question answering on knowledge bases. In: Proceedings of the Web Conference 2021, WWW 2021, pp. 3477–3488. Association for Computing Machinery, New York (2021)
10. Hirigoyen, R., Zouaq, A., Reyd, S.: A copy mechanism for handling knowledge base elements in SPARQL neural machine translation. In: Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022, pp. 226–236, Online only. Association for Computational Linguistics (November 2022)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
12. Jiang, L., Usbeck, R.: Knowledge graph question answering datasets and their generalizability: Are they enough for future research? In: Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Shane Culpepper, J., Kazai, G. (eds.) SIGIR 2022: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022, pp. 3209–3218. ACM (2022)

13. Keysers, D.: Measuring Compositional Generalization: A Comprehensive Method on Realistic Data . arXiv:1912.09713 ((June 2020)
14. Lake, B., Baroni, M.: Generalization without systematicity: on the compositional skills of sequence-to-sequence recurrent networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, 10–15 Jul. Proceedings of Machine Learning Research, vol. 80, pp. 2873–2882. PMLR (2018)
15. Lewis, M.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020, pp. 7871–7880. Association for Computational Linguistics (2020)
16. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 6–12 July 2002, Philadelphia, PA, USA, pp. 311–318. ACL (2002)
17. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140:1–140:67 (2020)
18. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: Barzilay, R., Kan, M.-Y. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, 30 July - 4 August, Volume 1: Long Papers, pp. 1073–1083. Association for Computational Linguistics (2017)
19. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 December 2014, Montreal, Quebec, Canada, pp. 3104–3112. Curran Associates Inc (2014)
20. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: a corpus for complex question answering over knowledge graphs. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 210–218. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_22
21. Vaswani, A.: Attention is all you need. In: Guyon, I., et al.: (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp. 5998–6008. Curran Associates Inc (2017)
22. Yin, X., Gromann, D., Rudolph, S.: Neural machine translating from natural language to SPARQL. Future Gener. Comput. Syst. **117**, 510–519 (2021)