IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 April 2022, accepted 1 July 2022, date of publication 5 July 2022, date of current version 11 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3188714

RESEARCH ARTICLE

SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions

MD RASHAD AL HASAN RONY[1,2], UTTAM KUMAR[3], ROMAN TEUCHER[1], LIUBOV KOVRIGUINA[1], AND JENS LEHMANN[1,2]

[1]Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), 01069 Dresden, Germany
[2]Smart Data Analytics Research Group, University of Bonn, 53115 Bonn, Germany
[3]Data Science and Intelligent Systems Group, University of Bonn, 53115 Bonn, Germany

Corresponding author: Md Rashad Al Hasan Rony (rashad.rony@iais.fraunhofer.de)

This work was supported in part by the SPEAKER under Grant BMWi FKZ 01MK20011A, in part by the JOSEPH (Fraunhofer Zukunftsstiftung), in part by the OpenGPT-X under Grant BMWKFKZ 68GX21007A, in part by the Excellence Clusters ML2R under Grant BmBF FKZ01 15 18038A/B/C, in part by the ScaDS.AI under Grant IS18026A-F, and in part by the TAILOR under Grant EUGA 952215.

ABSTRACT SPARQL query generation from natural language questions is complex because it requires an understanding of both the question and underlying knowledge graph (KG) patterns. Most SPARQL query generation approaches are template-based, tailored to a specific knowledge graph and require pipelines with multiple steps, including entity and relation linking. Template-based approaches are also difficult to adapt for new KGs and require manual efforts from domain experts to construct query templates. To overcome this hurdle, we propose a new approach, dubbed SGPT, that combines the benefits of end-to-end and modular systems and leverages recent advances in large-scale language models. Specifically, we devise a novel embedding technique that can encode linguistic features from the question which enables the system to learn complex question patterns. In addition, we propose training techniques that allow the system to implicitly employ the graph-specific information (i.e., entities and relations) into the language model's parameters and generate SPARQL queries accurately. Finally, we introduce a strategy to adapt standard automatic metrics for evaluating SPARQL query generation. A comprehensive evaluation demonstrates the effectiveness of SGPT over state-of-the-art methods across several benchmark datasets.

INDEX TERMS Knowledge based systems, knowledge graph, information retrieval, query generation, language models.

I. INTRODUCTION

SPARQL is a query language used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. In recent years, the conversion of natural language questions (NLQs) to SPARQL queries gained further popularity to the growing number of graph-based applications [1]–[3]. Automatic query generation from NLQ is a long-standing research challenge with several factors contributing to its difficulty, including but not limited to understanding the complex aspects of syntax and semantics of the natural language question (i.e., ellipsis, ambiguity, lexical gap), error propagation in NLP pipelines, and skewed distribution of question types in training datasets. Additionally, changing the underlying KG requires rewriting the SPARQL query for a given NLQ as illustrated in Figure 1.

Several approaches for SPARQL query generation have been presented recently [6]–[11]. The widely adopted approaches involve query schema or template classification and filling in the slots in the templates using available sub-graph information such as linked entities and relations [9], [10], [12], [13]. A different line of research is centered around transforming natural language questions to their corresponding SPARQL queries in a sequence-to-sequence manner [8], [14]. Despite substantial research efforts, adapting these systems to arbitrary KGs and handling low-frequency question types is difficult. The main challenges can be summarized as follows: (1) SPARQL templates are usually created manually or semi-automatically by domain experts, which is both time consuming and cost intensive, (2) The query templates are tailored to a particular KG,

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Abid.

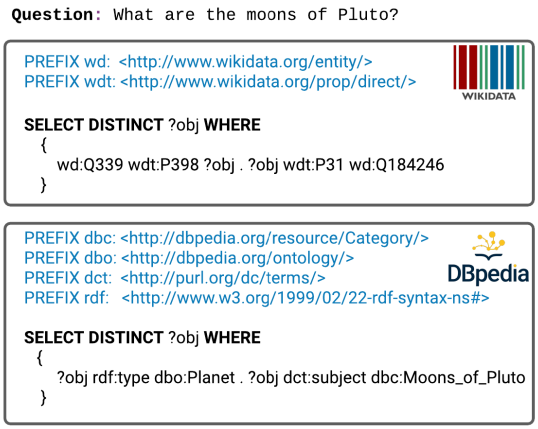

FIGURE 1. An illustration of a SPARQL query used to answer a natural question over Wikidata [4] and DBpedia [5]. Here, Q339, P398, P31, Q184246 are the Wikidata ID of *Pluto*, *child astronomical body*, *instance of*, and *moon of Pluto*, respectively.

which results in potentially changing of the whole template set when the underlying graph is changed, (3) The extension of template sets to handle new question types is performed manually or semi-automatically, and (4) In pipeline-based approaches, the SPARQL generation module is dependent on the performance of the preceding modules (i.e., entity and relation linkers as well as ranking algorithms) and, thus, suffer from error propagation.

TABLE 1. Comparison of SPARQL queries for two different questions with same wording.

Question 1:	What is the name of the actress married to the prince of England?
SPARQL 1:	SELECT ?s_label WHERE { ?s wdt:P106 wd:Q33999 . ?s wdt:P26 ?spouse . ?spouse wdt:P97 wd:Q4971429 . ?s rdfs:label ?s_label FILTER (lang (?s_label) = "en") }
Question 2:	What is the name of the prince of England married to an actress?
SPARQL 2:	SELECT ?s_label WHERE { ?spouse wdt:P106 wd:Q33999 . ?s wdt:P97 wd:Q4971429 . ?s wdt:P26 ?spouse . ?s rdfs:label ?s_label FILTER (lang (?s_label) = "en") }

Addressing the shortcomings, we propose a new approach, dubbed SGPT, for SPARQL query generation. SGPT encodes the linguistic features of an NLQ and corresponding sub-graph information (i.e, entities, if provided), and leverages a generative language model (LM) to generate SPARQL queries. We hypothesize that a deeper understanding of the NLQ is crucial for generating a correct query, since a slight deviation in the syntactic structure of the question may result in a different SPARQL query. Table 1 demonstrates such an example, where the queries are Wikidata knowledge graph-based [4], and *Q33999*, and *Q4971429* are Wikidata entity IDs of the entity *Actor* and *British Prince*, respectively. The Wikidata relation IDs *P106*, *P26* and *P97* refer to the relations *Occupation*, *Spouse*, and *Noble title*, respectively. Specifically, besides the standard word and positional embedding layers, we design special embedding layers that embed an arbitrary number of linguistic features of an NLQ, such as parts-of-speech (POS) tags and dependency tree features (i.e., dependency relations and information about tree node's children). A stack of Transformer [15]-encoders is employed to encode the linguistic features. The proposed embedding techniques facilitate SGPT to inject additional knowledge (i.e., entities) as well as allow the integration of SGPT into pipeline-based systems in a modular fashion. Furthermore, we employ the Transformer [15]-decoder based language model GPT-2 [16], to generate SPARQL queries. Our training methodology enables SGPT to embed an arbitrary KG directly into the model parameters. Moreover, the system does not require any query template or KG as input at inference time.

The evaluation of SPARQL query generation is a crucial step for developing NLQ to SPARQL systems. A widely used metric BLEU [17], was primarily designed to evaluate machine translation (MT) and later adopted for evaluating natural language generation (NLG). However, in contrast to natural language sequences, SPARQL is a formal language and includes query-specific terms, patterns and variables which the standard automatic metrics such as BLEU do not consider when computing n-gram overlaps. To overcome this shortcoming, we propose a variable normalization algorithm to adopt BLEU and F1 score for measuring the performance of SPARQL query generation. We call the adopted metrics SP-BLEU and SP-F1.

To assess the performance of SGPT, we conduct experiments on three publicly available datasets: LC-QuAD 2.0 [18], VQuAnDA [19] and QALD-9 [20]. We evaluate the system-generated SPARQL queries using both human and automatic metrics. Furthermore, by an ablation study we examine the impact of individual components on SGPT's overall performance to verify their effectiveness. Moreover, we conduct extensive analysis to demonstrate SGPT's capacity to comprehend diverse, complex questions and generate correct SPARQL queries. The empirical evaluation confirms that SGPT significantly outperforms state-of-the-art methods in generating SPARQL queries from natural language questions across several benchmark datasets. We open source the code and model.[1] The key contributions of this work can be summarized as follows:

- A novel embedding technique, that embeds the linguistic features of a question and graph information for the SPARQL query generation task.
- A generative system, SGPT, that utilizes the linguistic features of a natural language question and learns to embed the KG into language model's parameters. SGPT can be used as either as a standalone system or can be integrated into modular pipelines.
- An algorithm to adapt standard evaluation metrics for measuring the performance of SPARQL query generation.
- A comprehensive evaluation that demonstrates the capabilities of SGPT in generating complex and correct SPARQL queries across various knowledge graph based benchmarks.

[1] https://github.com/rashad101/SGPT-SPARQL-query-generation

Rest of the part of this paper is organized as follows. In Section II, we review the previous research efforts on various methods for natural language question to SPARQL query generation. In section III, we provide background about SPARQL query language and dependency tree, which we utilize in our research. The proposed SGPT approach is described in section IV. In Section V, details about the data, training setup, baseline systems, and evaluation metrics are provided. Additionally, the qualitative and quantitative results are also discussed. A comprehensive analysis of the proposed system is provided in Section VI, including ablation study, case studies, effectiveness of different components, error analysis and limitations. Finally, in Section VII, we summarize the key findings and identify future study areas.

II. RELATED WORK

A. MANUAL AND SEMI-AUTOMATIC APPROACHES

The early research on SPARQL query generation primarily focused on hand-crafted query construction [21]–[26]. In these approaches, SPARQL queries were manually designed to test the coverage and inference capabilities of ontology systems. A different research direction emphasised on query generation from datasets [27], [28]. Görlitz *et al.* [27] carefully explored an RDF dataset and defined a set of query characteristics for the query selection purpose. The authors employed a query generation heuristic to predict the final SPARQL representation, which checks all possible combinations of query patterns based on the defined query characteristics. In another paper, Qiao *et al.* [28] proposed a technique to construct a synthetic graph from a given RDF graph employing three separate algorithms to generate various types of SPARQL queries. However, the algorithm-generated queries are limited by six triples and can have at most two attributes. An ontology-based semi-automatic method was proposed by Dibowski *et al.* [29], where a user interface is provided to modify or select relevant concepts from the ontology for generating the SPARQL query. Nevertheless, manual efforts make it difficult to adapt these systems for large scale knowledge bases such as Wikidata [4] and DBpedia [5].

B. SCHEMA-BASED APPROACHES

Recently, to alleviate the manual efforts, a schema-based SPARQL query generation has received significant research attention [6], [7], [30]. These approaches aim to generate an intermediary schema representation (template) of the SPARQL query. The slots in the SPARQL schema are then filled up based on the defined heuristics to rank and obtain the final SPARQL query. In another schema-driven approach, Zenz *et al.* [31] proposed a method to bind domain-specific keywords to generate query template. The authors followed an incremental refinement strategy to obtain the final SPARQL query from a query template. In a different work, Unger *et al.* [30] introduced a method to generate SPARQL query templates, utilizing the semantic structure of the question. More recently, a classification based approach was proposed by Vollmers *et al.* [10], where semantically similar types of questions are classified to obtain a query template. Nevertheless, the query generation task remains limited due to the fixed number of schema. To extend the coverage of these systems for additional types of questions and queries, manual schema creation is required.

C. OTHER APPROACHES

In a different direction of solutions, Soru *et al.* [8] developed a sequence-to-sequence system that utilizes bi-directional LSTM [32] for generating SPARQL templates. An interpreter reconstructs the final SPARQL query from the query template using rule-based heuristics. However, the method cannot handle out-of-vocabulary words in the test set and lacks understanding of the question, thus frequently generating incorrect graph patterns in the query. Recently, Zafar *et al.* [9] exploited syntactic features to train a SPARQL query ranking model leveraging Tree-LSTM [33]. The similarity score between syntactic features of a question and a query is used for ranking candidate queries. Since the syntactic features are not learned and are only used to compute tree-similarity, the system does not generalize well when encountering an unseen question. Furthermore, the system needs to find all query patterns from the extracted sub-graph to predict the final SPARQL query, otherwise it fails to generate the query.

The approaches stated above are difficult to adjust for a new and arbitrary KG since they require manual construction of SPARQL queries for adaption and lack understanding of the question. In contrast to the approaches mentioned above, this research aims to encode the linguistic features of an NLQ and leverages a pre-trained language model to both learn the graph patterns and generate SPARQL queries.

III. PRELIMINARIES

A. SPARQL QUERY LANGUAGE

The term "SPARQL" stands for *SPARQL Protocol and RDF Query Language*. According to the official definition, a SPARQL query can be formally considered as a tuple $\langle GP, DS, SM, R\rangle$, where GP is a graph pattern (query pattern), DS is an RDF dataset, SM is a set of solution modifiers (ORDER, PROJECTION, DISTINCT, OFFSET, LIMIT), R is a result form (SELECT, CONSTRUCT, DESCRIBE and ASK).[2] Figure 2 illustrates the terms used in the formalization. Similar to the previous works, SGPT aims at generating the query body which includes the result form, graph pattern, and solution modifiers.

B. DEPENDENCY TREE

Dependency and constituency trees are used in computational linguistics to express syntactic dependencies between words in a sentence. A dependency tree, unlike a constituency tree, may express non-adjacent and non-projective relations in a sentence, which are common in spoken language.

[2]https://www.w3.org/2001/sw/DataAccess/rq23/defns

FIGURE 2. SPARQL query components.

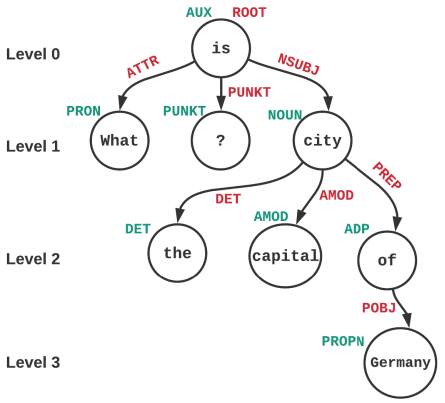

FIGURE 3. A dependency tree parsed from the question, ***What is the capital city of Germany?***. The text in red represents the dependency relation between two nodes. In this paper, the root node of the dependency tree is designated as level 0 node. The levels of other nodes are assigned with respected to distance from the root node at level 0. The POS-tags are highlighted in green for demonstration purposes only and are not part of the dependency tree.

A dependency tree is a graph with words as nodes and dependency relations as edges, as depicted in Figure 3. Every node has exactly one parent node except the root node, which has no parent [34]. In Figure 3, the annotations follow part-of-speech tagset from Universal Dependencies[3] and dependency relations from Stanford Dependencies.[4] The POS-tags, dependency relations, and information about immediate syntactic dependents of the token (level in Figure 3), are leveraged in this paper to understand the NLQ for SPARQL generation.

IV. APPROACH: SGPT

A. PROBLEM DEFINITION

We define a knowledge graph as a multi-relational graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a set of entities in the knowledge graph and $\mathcal{R}$ is a set of relations that connects two entities in the knowledge graph. This paper proposes separate training techniques for two use cases, 1) only a natural question is available, 2) both the question and entities mentioned in the question are provided. In the second case, we consider the provided set of entities as additional knowledge $\mathcal{K}$. Given a natural language question $\mathcal{Q}$ (for the first case) or an additional knowledge $\mathcal{K}$ and a natural question $\mathcal{Q}$ (for the second case), the goal of SGPT is to generate a SPARQL query $\mathcal{S}$. We define SGPT$_{\mathcal{Q}}$ as the system for the first use case and SGPT$_{\mathcal{Q},\mathcal{K}}$ for the second use case. Formally, in SGPT$_{\mathcal{Q}}$, the probability distribution of generating a SPARQL query by the language model is defined as:

$$p_\theta(\mathcal{S}|\mathcal{Q}) = \prod_{i=1}^{n} p_\theta(s_i|s_1, \ldots, s_{i-1}, \mathcal{Q}), \tag{1}$$

and in SGPT$_{\mathcal{Q},\mathcal{K}}$ the probability distribution is as follows:

$$p_\theta(\mathcal{S}|\mathcal{Q}, \mathcal{K}) = \prod_{i=1}^{n} p_\theta(s_i|s_1, \ldots, s_{i-1}, \mathcal{Q}, \mathcal{K}), \tag{2}$$

where θ is model's parameters, n is the length of the query and s_i is the token generated at i-th time step.

We use the terms "SPARQL query" and "query" interchangeably throughout this paper. The term "SGPT" refers to both SGPT$_{\mathcal{Q}}$ and SGPT$_{\mathcal{Q},\mathcal{K}}$, if there is no design or implementation difference between them for the describe concept or operation. SGPT follows the encoder-decoder design paradigm. The approach is described in depth in the following subsections.

B. ENCODING

We design special embedding layers to embed the linguistic features of the question. The idea of special embedding was initially suggested by Devlin *et al.* [35]. The idea has been recently adopted for encoding structural information such as in table parsing [36] and graph-based dialogue generation task [37]. Unlike these prior works, in this work special embedding layers are designed to capture the linguistic characteristics of an NLQ. A stack of Transformer [15]-encoders then encodes these embeddings.

1) INPUT SEQUENCE CONSTRUCTION

A *Pre-processor* component in SGPT takes $\mathcal{Q}$ and $\mathcal{K}$ as input and constructs the input sequence. The input sequence in SGPT$_{\mathcal{Q}}$ starts with a `[BOS]` token, then the question $\mathcal{Q}$ and an `[EOS]` token that marks the end of the sequence as depicted in Figure 4.

Similarly, in SGPT$_{\mathcal{Q},\mathcal{K}}$ the input sequence starts and ends with `[BOS]` and `[EOS]` tokens, respectively. The question is separated by a `[Q]` token from the additional knowledge $\mathcal{K}$, in the input sequence. Furthermore, each entity in the additional knowledge is preceded by an `[E]` token in the input sequence (see Figure 5).

To allow generalisation, the entity positions in the question and query are masked in SGPT$_{\mathcal{Q},\mathcal{K}}$. A *Pre-processor* masks both the entities in the question and the entities (including their prefix) in the query by a generic `ENT` token for training.

[3]https://universaldependencies.org/u/pos/index.html

[4]https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf

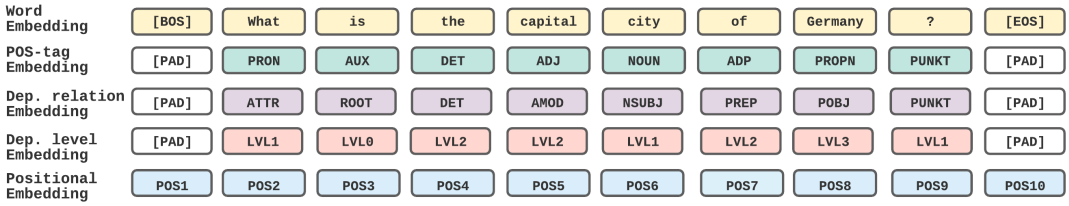

FIGURE 4. An illustration of special embedding layers used in $\text{SGPT}_{\mathcal{Q}}$.

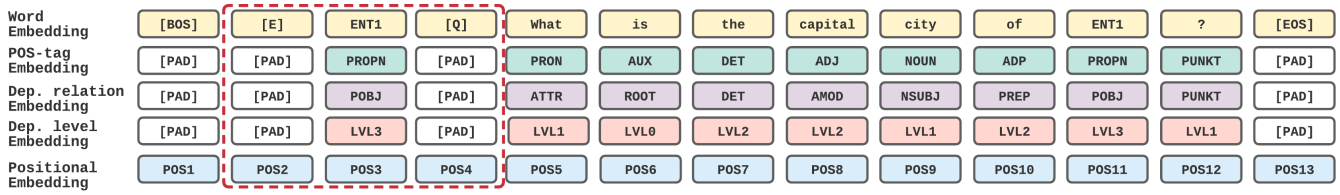

FIGURE 5. The question and knowledge embedding techniques used in $\text{SGPT}_{\mathcal{Q},\mathcal{K}}$. The dotted red box indicates the separation of additional knowledge from the question.

The entities in the query that do not appear in the question are not masked and learned in the model's parameters. For multiple entities in the question the generic masks are as follows: ENT1, ENT2,..,ENTn, where n is the number of entities present in the question. For instance, if the question contains two entities, the first is masked by the ENT1 token and the second by the ENT2 token. Figure 5 depicts a masked input sequence.

Generally, relation linking is challenging because, the surface form of the relation in the question often differs from the label of the relation in the KG [38]. This leads to relation linking-based error propagation in the pipeline-based systems [39], [40]. To alleviate the error propagation, we delegated the relation learning task to the GPT-2 model, which learns the KG (i.e., entity and relation) in its parameters.

2) EMBEDDING THE INPUT SEQUENCE

The constructed input sequence is passed through five different embedding layers to capture different properties of the input. The embedding layers are described below:

(i) **Word embedding** layer encodes the token level information of the input sequence. A pre-trained GPT-2 [16] tokenizer is used to tokenize the input sequence.

(ii) **POS-tag embedding** layer embeds the part-of-speech tag of the corresponding token in the word embedding layer. POS-tags are used to understand the use of word in the question better, since a particular word may have different meaning based on the usage in a sentence.

(iii) **Dependency relation embedding** layer encodes the dependency relations between pairs of words in the question.

(iv) **Dependency level embedding** layer embeds the information about the children of the tokens in the word embedding layer, extracted from the dependency tree.

(v) **Positional embedding** layer embeds the absolute position information of the input sequence.

SGPT computes the sum of POS-tag, dependency relation and dependency level embeddings and apply *Layer Normalization* [41] to obtain the linguistic context of the NLQ. *Layer Normalization* normalizes the embedding and prevents the model's weights from exploding. The encoding of linguistic context is discussed in the next section. The word and positional embeddings are utilized by a GPT-2 decoder, discussed in §IV-C.

3) LINGUISTIC CONTEXT ENCODING

A stack of Transformer-encoders [15] is employed in this paper to encode the linguistic context. The output of the l-th encoder layer is formalized as follows:

$$\begin{aligned} h_i^l &= \sum_{j=1}^{N} \alpha_{ij}^l (h_j^{l-1} W^V) \\ \alpha_{ij}^l &= \frac{exp(t_{ij}^l)}{\sum_{p=1}^{N} exp(t_{ip}^l)} \\ t_{ij}^l &= \frac{(h_i^{l-1} W^Q)(h_j^{l-1} W^K)}{\sqrt{d}} \\ i &= 1, 2, \ldots, N \end{aligned} \tag{3}$$

where W^Q, W^K, and W^V are trainable weights, N is the sequence length, and d is the dimension of query, key and value vectors. The output is then passed to a Feed-Forward Neural Network (FFNN), preceded and followed by residual connections and normalization layers as follows:

$$\begin{aligned} h'^l_i &= LayerNorm(h_i^l + h_i^{l-1}) \\ h''^l_i &= W_2^l ReLU(W_1^{l+1} h'^l_i + b_1) + b_2 \\ \hat{h}_i^l &= LayerNorm(h'^l_i + h''^l_i), \end{aligned} \tag{4}$$

where W_2^l and W_1^{l+1} are trainable weights and b_1 and b_2 are bias terms. A rectified linear unit (ReLU) is employed

as the activation function in the FFNN network. The output of the last encoder layer $\hat{h}_i^l$ is then passed to a GPT-2 model for decoding. Figure 6 illustrates a high-level architecture of SGPT.

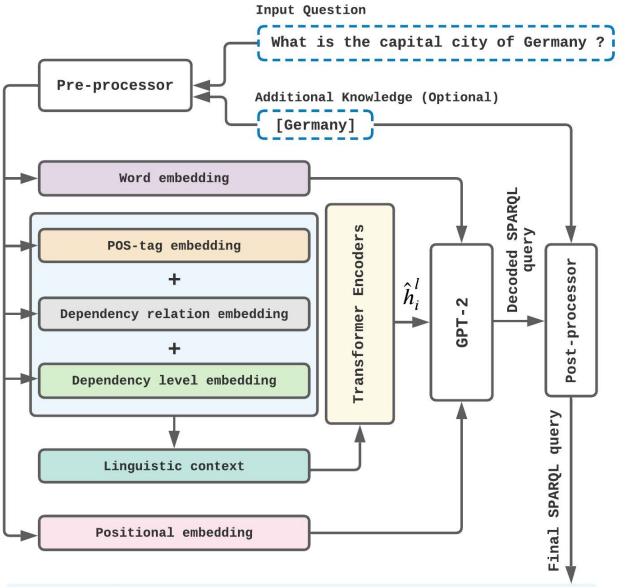

FIGURE 6. System architecture.

C. DECODING

A GPT-2 [16] language model is used in this paper to model SPARQL query generation. However, any Transformer [15] decoder-based LM can be used. GTP-2 is a multi-headed attention-based language model. The attention, computed in each of GPT-2's heads is formalized as follows:

$$\mathcal{F}(Q, K, V) = softmax(\frac{1}{\sqrt{d_k}}(QK^T) + M)V,$$
$$H_i = \mathcal{F}(QW_i^Q, KW_i^K, VW_i^V), \quad (5)$$

where $\mathcal{F}(\cdot)$ computes the masked attention. The attention mask is denoted as M, where H_i is the i-th head and d_k=d_m/h. Here, d_m is the model's dimension and h denotes the number of heads. Q, K and V are query, key and value where W_i^Q, W_i^K, W_i^V are trainable weights. Model parameters in θ are trained to minimize the negative log-likelihood $\mathcal{L}_\mathcal{Q}$ (for SGPT$_\mathcal{Q}$) and $\mathcal{L}_{\mathcal{Q},\mathcal{K}}$ (for SGPT$_{\mathcal{Q},\mathcal{K}}$) for next-token prediction. Formally, the loss $\mathcal{L}_\mathcal{Q}$ and $\mathcal{L}_{\mathcal{Q},\mathcal{K}}$ are defined as follows:

$$\mathcal{L}_\mathcal{Q} = -\sum_i^n \log p(s_i|s_1, \ldots, s_{i-1}, \mathcal{Q}),$$
$$\mathcal{L}_{\mathcal{Q},\mathcal{K}} = -\sum_i^n \log p(s_i|s_1, \ldots, s_{i-1}, \mathcal{Q}, \mathcal{K}), \quad (6)$$

where n is the maximum query length. During inference, *Top-k sampling* decoding [42] is utilized to generate a word token at each time step. It is noteworthy that the entities that were masked by the pre-processor of SGPT$_{\mathcal{Q},\mathcal{K}}$ in the question also appear masked in the decoded query. Once the sequence decoding is completed, the *Post-processor* component replaces the entity masks with their corresponding entity identifier.

V. EXPERIMENTS AND RESULTS

A. DATA

We evaluate SGPT on three publicly available datasets.

1) **LC-QuAD 2.0** [18]: A large-scale question answering dataset, which includes for each complex natural language question its corresponding query template, SPARQL query and annotations. We chose LC-QuAD 2.0 to evaluate Wikidata-based questions.

2) **VQuAnDa** [19]: A verbalization dataset which contains natural language questions and their corresponding SPARQL queries for extracting answers. VQuAnDa contains DBpedia-based questions.

3) **QALD-9** [20]: QALD-9 is a small yet challenging multilingual question answering dataset based on DBpedia. The dataset contains questions in 3 to 8 different languages. Within the scope of this paper, we select the English data.

TABLE 2. Dataset statistics.

	LC-QuAD 2.0	VQuAnDA	QALD-9
# train	21,497	3,500	350
# validation	2,389	500	58
# test	5,969	1,000	150
Avg. # tokens in the question	10.55	11.09	7.48
Avg. # tokens in the query	13.68	12.42	13.20
Avg. # keywords in the query	2.08	1.96	2.21

TABLE 3. Statistics of question types.

	LC-QuAD 2.0		VQuAnDa		QALD-9	
Question Type	Train	Test	Train	Test	Train	Test
Boolean	2,111	1,433	328	82	40	5
Count	1,134	281	676	181	58	32
Rank	905	204	134	29	59	13
Simple	3,216	824	617	172	158	52
String	5,920	1,433	1	-	32	17
Two hops	20,283	5,049	3,052	744	203	88
Two intents	5,062	1,223	-	-	17	11

The dataset statistics are summarized in Table 2. The original train and test splits of LC-QuAD 2.0, VQuAnDa and QALD-9 are 24,180/6,046, 4,000/1,000 and 408/150, respectively. For the validation during the training, we split the training set and use 10%-15% data as the validation set, based on the dataset size. In LC-QuAD, 2.0 we removed the data from train set with empty question and query field. Natural language questions are treated as complex when answering them requires multiple graph patterns. Depending on the complexity, the following question types are distinguished [18]:

- *Boolean*: Question where the answer is either True or False.
- *Count*: That computes the number of occurrence of a particular thing.
- *Rank*: Questions seek answer which is in a particular order.
- *Simple*: Questions correspond to semantics of natural language question that is obtained by matching just one hop relations of the entity.
- *String*: Questions, for which answers contain a particular word or letter.
- *Two Hop*: Questions, in which semantic interpretation corresponds to two hop of the entity's connection in the knowledge graph i.e. two set of triples in the where clause of the SPARQL query.
- *Two Intent*: Questions seek for minimum two answer for the same question for example, mother of a person and also the child of same person.

The question types statistics of the benchmark datasets are reported in Table 3.

B. TRAINING SETTINGS

We use a stack of Transformer-encoders with 8 heads and 6 layers to encode linguistic features. For decoding we employ the GPT-2 [16] model with 117M parameters throughout this paper. As an optimizer, AdamW [43] with ϵ = 1e-8 and a value of 6.25e-5 is used as learning rate. GELU [44] is used as the activation function. The optimum hyper-parameters for each dataset were determined using grid search based on the performance on the validation set. We used spaCy[5] to annotate the NLQ with the POS-tags and dependency relations, based on the work of [45]. All experiments were run in a distributed training environment with 2 GPUs, each with 12 GBs of RAM. The training takes 215, 125 and 35 minutes on LC-QuAD 2.0, VQuAnDa, and QALD-9, respectively.

C. EVALUATION METRICS

1) AUTOMATIC METRICS

Following the baseline models, we use BLEU [17] and F1 score as automatic metrics for the evaluation. Generally, SPARQL queries in the used data sets were created manually or semi-automatically by domain experts. This means that the choice of variable names depends on the domain experts and can vary. Hence, we argue that these metrics are incapable of capturing the variations in the variables used in a the query, since BLEU computes n-gram overlaps and F1 is computed by token-level precision and recall. We propose an adaption, where variables in both reference and predicted queries are normalized before the standard evaluation performed by BLEU and F1 and named them SP-BLEU and SP-F1, respectively. The proposed normalization technique is shown in Algorithm 1. Our proposed variable name normalization technique allows the automatic metric to evaluate a predicted query regardless of the annotated variable names. An example of query normalization is demonstrated in Table 8. The normalisation results in the metrics more closely reflecting actual performance. We compare all systems on both the standard automated metrics and the proposed metrics (discussed in §V-E).

[5]https://spacy.io/

Algorithm 1: Query Normalization

Input: A SPARQL query $\mathcal{S}$
Output: A normalized SPARQL query $\mathcal{S}_n$
$V_o \leftarrow \emptyset, V_n \leftarrow \emptyset$ ▷ initializing empty sets
$i \leftarrow 0, i_o \leftarrow 0, \mathcal{S}_n \leftarrow \emptyset$
for $w \in \mathcal{S}$ **do**
  if w *startswith '?'* **then**
    $t_c \leftarrow \emptyset$
    if $w \in V_o$ **then**
      $i_o \leftarrow indexOf(w, V_o)$ ▷ position of w in V_o
      $t_c \leftarrow V_o[i_o]$
    else
      $V_o \leftarrow add(w, V_o)$▷ adds a value w to the set V_o
      $i \leftarrow i + 1$
      $t_c \leftarrow string(?var\ i)$ ▷ converts to string
    $\mathcal{S}_n \leftarrow concat(\mathcal{S}_n, t_c)$ ▷ string concatenation
  else
    $\mathcal{S}_n \leftarrow concat(\mathcal{S}_n, t_c)$ ▷ string concatenation
return $\mathcal{S}_n$

2) HUMAN EVALUATION

We further conducted a human evaluation to manually assess the quality of generated queries. We randomly chose 75 examples (25 from each dataset) and asked two domain experts to evaluate the system generated queries based on the following criteria: 1) *Syntax validity* - how structurally correct the generate queries are, and 2) *Content validity* - how correct the entities and relations are. We asked the reviewers to rate the system generated queries on a scale of 1 to 5 (higher is better). The inter-annotator agreement score (Cohen's kappa κ) of the annotated data is 0.86.

D. BASELINES

We compare SGPT with both sequence-to-sequence and template-based methods.

SQG [9]: A set of candidate queries are created in SQG based on the sub-graph patterns, which are then ranked and arranged based on structural similarity, utilizing Tree-LSTM [33].

NSpM [8]: A sequence-to-sequence strategy in which a Bidirectional Long Short-Term Memory Network (Bi-LSTM) learns to generate a template SPARQL from a natural language question.

TeBaQA [10]: The TeBaQA model depends on template classes which it generates from the training dataset, to predict the SPARQL query. To generate the SPARQL query,

TABLE 4. Performance of SGPT and baseline models on three benchmark datasets. Best scores are in **bold**.

Models	with $\mathcal{K}$	LC-QuAD 2.0 BLEU	LC-QuAD 2.0 F1	LC-QuAD 2.0 SP-BLEU	LC-QuAD 2.0 SP-F1	VQuAnDa BLEU	VQuAnDa F1	VQuAnDa SP-BLEU	VQuAnDa SP-F1	QALD-9 BLEU	QALD-9 F1	QALD-9 SP-BLEU	QALD-9 SP-F1
NSpM [8]	✗	34.74	66.47	38.39	70.78	37.75	59.96	37.75	59.96	18.23	45.34	24.18	50.53
SGPT$_{\mathcal{Q}}$ (ours)	✗	**60.50**	**83.45**	**63.59**	**86.22**	**63.82**	**87.08**	**63.82**	**87.08**	**29.95**	**60.22**	**32.12**	**64.57**
SQG [9]	✓	-	-	-	-	5.09	37.70	33.86	44.67	4.44	27.85	22.14	39.39
TeBaQA [10]	✓	-	-	-	-	13.30	22.41	13.30	22.41	12.82	28.81	17.48	32.24
SGPT$_{\mathcal{Q},\mathcal{K}}$ (ours)	✓	**73.78**	**89.04**	**77.85**	**92.27**	**72.58**	**88.87**	**72.58**	**88.87**	**35.68**	**67.82**	**41.88**	**72.98**

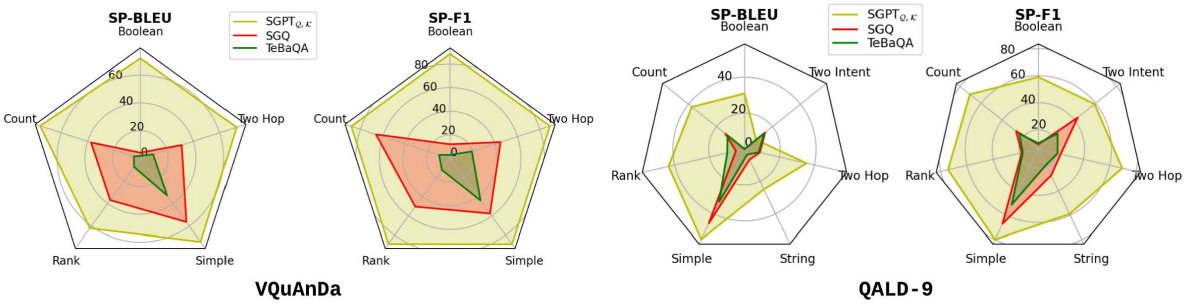

FIGURE 7. Question type-wise performance of SGPT$_{\mathcal{Q},\mathcal{K}}$ and baseline models on the test set of VQuAnDa and QALD-9.

TABLE 5. Results on data where baseline models could generate queries.

	VQuAnDa SP-BLEU	VQuAnDa SP-F1	QALD-9 SP-BLEU	QALD-9 SP-F1
SQG	31.11	46.40	24.30	55.60
SGPT$_{\mathcal{Q},\mathcal{K}}$	55.98	80.45	22.21	57.47
TeBaQA	30.55	44.81	25.57	55.14
SGPT$_{\mathcal{Q},\mathcal{K}}$	62.92	83.65	33.10	71.89

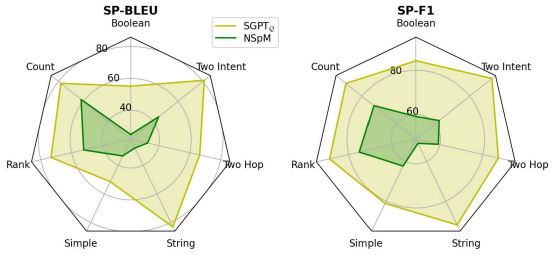

FIGURE 8. Question type-wise performance of SGPT$_{\mathcal{Q}}$ on LC-QuAD 2.0 test set.

the model first classifies the input question and predicts a template class. The slots in the template class are filled in by indexed entities and relations guided by a rule-based lookup. The generated queries are then ranked to obtain the final SPARQL query representation.

We train and evaluate the baseline models with their recommended settings.

E. QUANTITATIVE RESULTS

Table 4 summarises the performance of SGPT and baseline systems. In the first set of results where additional knowledge $\mathcal{K}$ is not provided, SGPT$_{\mathcal{Q}}$ outperformed the other generative system, NSpM, significantly across all metrics. In many cases NSpM failed to recognize the correct question types, thus frequently generated wrong queries. In the second set of results, using additional knowledge, the baseline models obtained very low scores because of their limited template coverage. We further investigated the performance of baseline models and observed that SQG managed to generate queries for 46% and 45.33% of the test NLQs of VQuAnDa and QALD-9, respectively. TeBaQA could generate queries for 30.55% and 40.67% of the same test NLQs. These template-based systems fail to generate queries primarily for two reasons: 1) They could not classify or find a suitable template for a give question 2) They failed to fill in all the slots in the selected template, resulting in no query predicted. We report the comparison of performance on the data where baseline models could generate queries in Table 5. The results suggest that SGPT outperformed the baseline models significantly in most cases. Despite given the correct subjects detected from the question, template-based systems failed to generate queries frequently. The main reason is that SPARQL queries oftentimes include intermediary entities which leads to correct answer but do not appear in the question. Our proposed training technique allows SGPT to learn those entities in the model's parameters and thus can effectively generate correct SPARQL queries.

Finally, we investigated the capabilities of SGPT and the baselines models on diverse types of questions, depicted in Figure 7 and 8. The improvements over all the baselines across benchmark datasets confirms SGPT's capacity to handling diverse types of questions.

TABLE 6. Case study showing a comparison between SGPT and baseline system's outputs.

Question	System	SPARQL query
Does cobalt have a time-weighted average exposure limit of .1?	Reference	**ASK WHERE** { wd:Q740 wdt:P2404 ?obj **FILTER**(?obj = 0.1) }
	NSpM	**ASK WHERE** { wd:Q1049389 wdt:P3737 ?obj **FILTER**(?obj = 12) }
	SGPT$_{\mathcal{Q}}$	**ASK WHERE** { wd:Q740 wdt:P2404 ?obj **FILTER** (?obj = 0.1) }
Which countries have places with more than two caves?	Reference	**SELECT DISTINCT** ?uri **WHERE** { ?cave rdf:type dbo:Cave ; dbo:location ?uri . ?uri rdf:type dbo:Country } **GROUP BY** ?uri **HAVING** (**COUNT**(?cave) > 2)
	SQG	(no results found)
	TeBaQA	(no results found)
	SGPT$_{\mathcal{Q},\mathcal{K}}$	**SELECT DISTINCT** ?uri **WHERE** { ?cave rdf:type dbo:Cave ; dbo:location ?uri . ?uri rdf:type dbo:Country } **GROUP BY** ?uri **HAVING** (**COUNT** (?cave) > 2)

TABLE 7. Ablation study.

Approach	SP-BLEU	Δ	SP-F1	Δ
SGPT$_{\mathcal{Q}}$ (seq2seq)	50.40	-	71.66	-
+ POS-tag emb.	56.74	6.34↑	76.92	5.26↑
+ Dep. relation emb.	62.91	6.17↑	84.21	7.29↑
+ Dep. level emb.	63.28	0.37↑	86.17	1.96↑
SGPT$_{\mathcal{Q},\mathcal{K}}$ (seq2seq)	67.76	-	82.00	-
+ POS-tag emb.	72.84	5.08↑	88.15	6.15↑
+ Dep. relation emb.	77.16	4.32↑	91.39	3.24↑
+ Dep. level emb.	77.73	0.57↑	92.27	0.88↑

TABLE 8. An illustration of query normalization.

Question	How many grand-children did Jacques Cousteau have ?
Reference	SELECT COUNT (DISTINCT **?y** as **?y**) WHERE { dbr:Jacques_Cousteau dbo:child **?x** . **?x** dbo:child **?y** . }
Normalized Reference	SELECT COUNT (DISTINCT **?var1** as **?var1**) WHERE { dbr:Jacques_Cousteau dbo:child **?var2** . **?var2** dbo:child **?var1** . }
Prediction	SELECT COUNT (DISTINCT **?string** as **?string**) WHERE { dbr:Jacques_Cousteau dbo:child **?uri** .**?uri** dbo:child **?string** . }
Normalized Prediction	SELECT COUNT (DISTINCT **?var1** as **?var1**) WHERE { dbr:Jacques_Cousteau dbo:child **?var2** . **?var2** dbo:child **?var1** . }

TABLE 9. Human evaluation results.

System	with $\mathcal{K}$	Syntax validity	Content validity
NSpM [8]	✗	4.17	3.00
SGPT$_{\mathcal{Q}}$ (ours)	✗	**4.96**	**4.10**
SQG [9]	✓	3.42	2.73
TeBaQA [10]	✓	3.85	2.99
SGPT$_{\mathcal{K},\mathcal{Q}}$ (ours)	✓	**5.00**	**4.26**

F. QUALITATIVE RESULTS

We conducted a human evaluation to assess the system generated SPARQL queries. We observed that SGPT is capable of generating queries with correct syntax, reported in Table 9. We also noticed that template-based approaches obtained comparatively low *syntax validity* score because they failed to generate queries in some cases. Overall, generative systems received a high *syntax validity* score as they learned the SPARQL pattern well. Figure 9 depicts the score distribution of human annotation and corresponding BLEU and SP-BLEU scores from automatic metrics. Human judgements are normalized to a scale of 0 to 100. The Spearman correlation co-efficient between BLEU and human judgement is 0.94, where for SP-BLEU and human judgement it is 0.97. This confirms that our proposed normalization algorithm enables the metric to correlate better with human judgement.

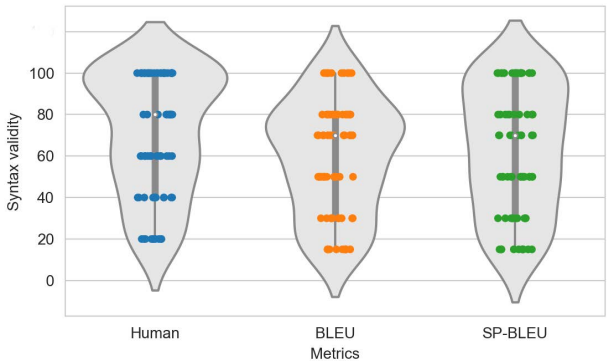

FIGURE 9. Human evaluation score distribution.

VI. ANALYSIS

A. ABLATION STUDY

Table 7 summarizes the results of the ablation study conducted to investigate how various components of SGPT affect its overall performance. The *seq2seq* approach denotes the SGPT model without the special layers: POS-tag embedding, dependency relation embedding and dependency level embedding. The results in Table 7 exhibit that adding syntactic features improves SGPT's capability to understand the question and generate the correct query. A remarkable gain in the performance is noticeable after adding of the POS-tag and dependency relation embedding layers, in both SGPT$_{\mathcal{K}}$ and SGPT$_{\mathcal{K},\mathcal{Q}}$. Adding dependency level embedding which captures information about token's immediate syntactic dependents, however, only slightly improved the results further.

B. CASE STUDY

Table 6 shows two NLQs with corresponding reference query and SPARQL queries, generated by the compared systems. The first NLQ is from LCQuAD 2.0 (Wikidata-based), and the second is from the QALD-9 dataset (DBpedia-based). In the first case where no additional knowledge is provided,

TABLE 10. Three error cases where the texts highlighted in green indicate the correct entry in the reference query and red indicating wrong predication in the system generated query. The text in yellow shows the masked entity.

Question	System	SPARQL query
Where did the designer of ENT1 die ?	Reference	SELECT DISTINCT ?uri WHERE { ENT1 dbo:designer ?x . ?x dbp:placeOfDeath ?uri . }
	$\text{SGPT}_{\mathcal{Q},\mathcal{K}}$	SELECT DISTINCT ?uri WHERE { ENT1 dbo:designer ?x . ?x dbo:deathPlace ?uri . }
List all the faiths that British Columbian politicians follow ?	Reference	SELECT DISTINCT ?uri WHERE { ?x dbp:residence dbr:British_Columbia . ?x dbp:religion ?uri . ?x <http://www.w3.org/1999/02/22-rdf-syntax-nstype> dbo:Politician }
	$\text{SGPT}_{\mathcal{Q}}$	SELECT DISTINCT ?uri WHERE { ?x dbp:residence dbr:British_Columbia_republic . ?x dbp:religion ?uri . ?x <http://www.w3.org/1999/02/22-rdf-syntax-nstype> dbo:Politician }
When did socialist state for contains administrative territorial entity of Beijing ?	Reference	SELECT DISTINCT ?sbj WHERE { ?sbj wdt:P150 wd:Q956 . ?sbj wdt:P31 wd:Q842112 }
	$\text{SGPT}_{\mathcal{Q}}$	SELECT DISTINCT ?sbj WHERE { ?sbj wdt:P150 wd:Q956 . ?sbj wdt:P31 wd:Q3624078 }

TABLE 11. Performance of entity and relation generation.

	LC-QuAD 2.0		VQuAnDa		QALD-9	
	F1 (E)	F1 (R)	F1 (E)	F1 (R)	F1 (E)	F1 (R)
NSpM	41.52	51.09	29.19	34.62	33.46	30.56
$\text{SGPT}_{\mathcal{Q}}$ (ours)	67.22	83.38	89.95	69.26	40.27	45.21
SQG	-	-	45.31	45.69	55.00	47.17
TeBaQA	-	-	20.71	16.93	48.25	33.21
$\text{SGPT}_{\mathcal{Q},\mathcal{K}}$ (ours)	97.75	83.60	97.74	70.88	79.14	48.39

$\text{SGPT}_{\mathcal{Q}}$ generated a query with correct content and syntax, whereas NSpM failed to generate the correct content. This demonstrates SGPT's capabilities of understanding the question and generating a query with correct content from the KG. Despite having additional knowledge provided for a challenging question (second case), SQG and TeBaQA failed to generate a correct query. They could not find a template that could both classify and fill in all the slots correctly. This exhibits the advantage of SGPT over template-based and slot-filling approaches in handling complex query patterns.

C. EFFECTIVENESS OF ENTITY MASKING STRATEGY

Masking entities and relations in the question is a widely adopted strategy for generating and classifying SPARQL query templates. In NSpM [8], all entities in a question are masked with a generic `<A>` token. During inference, the final query is obtained by replacing `<A>` with all possible entity combinations and ranking. Similarly, in TeBaQA the slots in the predicted template are filled in by checking all possible indexed entities and relations. In contrast, our proposed masking strategy in $\text{SGPT}_{\mathcal{Q},\mathcal{K}}$ eliminates the need for any slot-filling component. The entity masking strategy used in $\text{SGPT}_{\mathcal{Q},\mathcal{K}}$'s training allows the system to learn the patterns of entity positions in the question and generate corresponding correct query. $\text{SGPT}_{\mathcal{Q},\mathcal{K}}$ achieves an absolute 3.8%, 1.9%, and 1.1% increase of BLEU score on LC-QuAD 2.0, VQuAnDA, and QALD-9 respectively, when the entities are masked in the input sequence instead of keeping their initial mentions.

D. EFFECTIVE ENTITY AND RELATION GENERATION

Table 11 shows the study results, which we conducted to investigate how well our proposed model learns the KG in its parameters. The performance suggest that SGPT can learn the knowledge graph in its parameters with high accuracy. The metric F1 (E) denotes the F1 scores between the entity sets of ground truth and system generated queries. Similarly, F1 (R) shows the performance of relation prediction. Despite given the correct knowledge, SQG and TeBaQA failed to achieve high F1-scores for entity and relation linking. This is due to the fact that the generated query may include entities from the NLQ as well as intermediate entities and relations that are not explicitly present in the NLQ. The intermediary entities and relations are required to resolve the answers, which is dependent on the complexity of the question and hence cannot be specified in a template-based setting.

E. ERROR ANALYSIS AND LIMITATIONS

We performed an error analysis to inspect whether SGPT has not generated correct SPARQL queries. Table 10 shows such erroneous examples where the first one shows an error of $\text{SGPT}_{\mathcal{Q},\mathcal{K}}$ in generating the wrong masked query. Although the system could infer that the question is about death, it predicted the wrong, though similar relation `dbo:deathPlace` instead of `dbp:placeOfDeath`. The first two error cases are from DBpedia-based questions where the third example is based on Wikidata. In the second case, $\text{SGPT}_{\mathcal{Q}}$ correctly detected the query type and the topic about British Columbia. However, it generated the wrong entity `dbr:British_Columbia_republic` instead of `dbr:British_Columbia`. Similarly, in the third example, the system could infer, that the question is about a state, but predicted a Wikidata entity ID with the wrong type of state Q3624078 (sovereign state) instate of Q842112 (socialist state). Despite the failed cases, the generated queries in Table 10 confirm SPGT's capability of generating queries with correct syntax and query type.

Since SGPT learns the graph patterns in the model's parameters, fine-tuning is required if the graph is updated. Nevertheless, the proposed training techniques allow the system to learn intermediary graph patterns required to generate a complete SPARQL query, that are not detectable from the input question. The current version of this work only supports English language. To adapt SGPT for other languages, a POS-tagger, a dependency parser and a pre-trained language model of the target language are required. Despite the limitations, SGPT comes with the advantages of a training facility without

query templates, adaptable to arbitrary KG, and extendable for pipeline-based systems.

VII. CONCLUSION

We have presented SGPT, a SPARQL query generation system, improving the state-of-the-art across multiple benchmark datasets. Our proposed training technique eliminates the need for manual annotation and is applicable to arbitrary RDF datasets. The key contributions of SGPT include **1)** a new encoding technique for the linguistic features of a question and (optionally) entities in the question, that allows deeper question understanding during SPARQL generation, **2)** training techniques that leverage a pre-trained language model to generate a SPARQL query and can be adapted to questions from different knowledge graphs, **3)** improved evaluation metrics to measure the performance of SPARQL query generation. An extensive empirical assessment confirms SGPT effectiveness in handling diverse types of questions and generating correct SPARQL queries. In future, we intend to integrate an automatic knowledge retrieval component and provide support for multiple languages.

REFERENCES

[1] Y. Khan, M. Saleem, A. Iqbal, M. Mehdi, A. Hogan, A.-C. N. Ngomo, S. Decker, and R. Sahay, "SAFE: Policy aware SPARQL query federation over RDF data cubes," in *Proc. 7th Int. Workshop Semantic Web Appl. Tools Life Sci.*, Berlin, Germany, Dec. 2014, pp. 1–3.

[2] M. Kulmanov, S. Kafkas, A. Karwath, A. Malic, G. Gkoutos, M. Dumontier, and R. Hoehndorf, "Vec2SPARQL: Integrating SPARQL queries and knowledge graph embeddings," in *Proc. 11th Int. Conf. Semantic Web Appl. Tools Life Sci. (SWATLS)*, Antwerp, Belgium, vol. 2275, 2018.

[3] S. Rudolph, L. Schweizer, and Z. Yao, "SPARQL queries over ontologies under the fixed-domain semantics," in *Proc. Pacific Rim Int. Conf. Artif. Intell.* Cham, Switzerland: Springer, 2019, pp. 486–499.

[4] D. Vrandečić, "Wikidata: A new platform for collaborative data collection," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 1063–1064.

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[6] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee, "Diversified stress testing of RDF data management systems," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2014, pp. 197–212.

[7] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat, "GMark: Schema-driven generation of graphs and queries," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 4, pp. 856–869, Apr. 2017.

[8] T. Soru, E. Marx, A. Valdestilhas, D. Esteves, D. Moussallem, and G. Publio, "Neural machine translation for query construction and composition," 2018.

[9] H. Zafar, G. Napolitano, and J. Lehmann, "Formal query generation for question answering over knowledge bases," in *Proc. Eur. Semantic Web Conf.* Springer, 2018, pp. 714–728.

[10] D. Vollmers, R. Jalota, D. Moussallem, H. Topiwala, A.-C. Ngonga Ngomo, and R. Usbeck, "Knowledge graph question answering using graph-pattern isomorphism," 2021, *arXiv:2103.06752*.

[11] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 824–837, May 2018.

[12] S. Purkayastha, S. Dana, D. Garg, D. Khandelwal, and G. P. Shrivatsa Bhargav, "Knowledge graph question answering via SPARQL silhouette generation," 2021, *arXiv:2109.09475*.

[13] Y. Chen, H. Li, G. Qi, T. Wu, and T. Wang, "Outlining and filling: Hierarchical query graph generation for answering complex questions over knowledge graph," 2021, *arXiv:2111.00732*.

[14] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019, *arXiv:1910.13461*.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI Blog, Tech. Rep., 2019, p. 9, vol. 1, no. 8.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[18] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia," in *Proc. Int. Semantic Web Conf.* Auckland, New Zealand: Springer, 2019, pp. 69–78.

[19] E. Kacupaj, H. Zafar, J. Lehmann, and M. Maleshkova, "VQuAnDa: Verbalization question answering dataset," in *Proc. Eur. Semantic Web Conf.* Heraklion, Greece: Springer, 2020, pp. 531–547.

[20] N. Ngomo, "9th challenge on question answering over linked data (QALD-9)," *Language*, vol. 7, no. 1, pp. 58–64, 2018.

[21] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," *J. Web Semantics*, vol. 3, nos. 2–3, pp. 158–182, 2005.

[22] A. Owens and N. Gibbins, "Effective benchmarking for RDF stores using synthetic data," in *Proc. 7th Int. Semantic Web Conf.*, Karlsruhe, Germany, 2008.

[23] C. Bizer and A. Schultz, "Benchmarking the performance of storage systems that expose SPARQL endpoints," in *Proc. 4th Int. Workshop Scalable Semantic Web Knowl. Base Syst. (SSWS)*, 2008, p. 39.

[24] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "SP^{2}Bench: A SPARQL performance benchmark," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar./Apr. 2009, pp. 222–233.

[25] P. Haase, T. Mathäß, and M. Ziller, "An evaluation of approaches to federated query processing over linked data," in *Proc. 6th Int. Conf. Semantic Syst. (I-SEMANTICS)*, 2010, pp. 1–9.

[26] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri, "Test-driven evaluation of linked data quality," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, 2014, pp. 747–758.

[27] O. Görlitz, M. Thimm, and S. Staab, "SPLODGE: Systematic generation of SPARQL benchmark queries for linked open data," in *Proc. Int. Semantic Web Conf.* Berlin, Germany: Springer, 2012, pp. 116–132.

[28] S. Qiao and Z. M. Özsoyoğlu, "RBench: Application-specific RDF benchmarking," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1825–1838.

[29] H. Dibowski and K. Kabitzsch, "Ontology-based device descriptions and device repository for building automation devices," *EURASIP J. Embedded Syst.*, vol. 2011, pp. 1–17, Jan. 2011.

[30] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," in *Proc. 21st Int. Conf. World Wide Web*, Apr. 2012, pp. 639–648.

[31] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl, "From keywords to semantic queries—Incremental query construction on the semantic web," *J. Web Semantics*, vol. 7, no. 3, pp. 166–176, 2009.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] K. S. Tai, R. Socher, and D. Christopher Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process*, Beijing, China, vol. 1, Jul. 2015, pp. 1556–1566.

[34] K. Hall and V. Novák, "Corrective dependency parsing," in *Trends in Parsing Technology*. Dordrecht, The Netherlands: Springer, 2010, pp. 151–167.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, Minneapolis, MN, USA, vol. 1, Jun. 2019, pp. 4171–4186.

[36] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos, "TaPas: Weakly supervised table parsing via pre-training," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 4320–4333.

[37] F. Galetzka, J. Rose, D. Schlangen, and J. Lehmann, "Space efficient context encoding for non-task-oriented dialogue generation with graph attention transformer," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2021, pp. 7028–7041.

[38] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann, "EARL: Joint entity and relation linking for question answering over knowledge graphs," in *Proc. Int. Semantic Web Conf.* Monterey, CA, USA: Springer, 2018, pp. 108–126.
[39] S. W.-T. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proc. 53rd Annu. Meeting ACL 7th Int. Joint Conf. Natural Lang. Process. (AFNLP)*, 2015, pp. 1–11.
[40] D. Sorokin, "Knowledge graphs and graph neural networks for semantic parsing," Ph.D. thesis, Technische Univ., Darmstadt, Germany, 2021.
[41] L. J. Ba, J. R. Kiros, and E. Geoffrey Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[42] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, vol. 1, Jul. 2018, pp. 889–898.
[43] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–19.
[44] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[45] M. Honnibal and M. Johnson, "An improved non-monotonic transition system for dependency parsing," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, Sep. 2015, pp. 1373–1378.

MD RASHAD AL HASAN RONY received the bachelor's degree from BRAC University, Bangladesh, and the master's degree from the University of Bonn, Germany, where he is currently pursuing the Ph.D. degree with the Smart Data Analysis Group. He is also a Research Scientist working at the Fraunhofer IAIS, Dresden. He worked on several research and industry projects related to dialogue systems, question answering, and machine reading comprehension. His research interests include knowledge graph-based dialogue systems, question answering systems, evaluation of generative systems, machine reading comprehension, and language models.

UTTAM KUMAR received the Bachelor of Technology degree in computer science from the National Institute of Technology, Jamshedpur, India, and the master's degree in computer science from the University of Bonn, Germany, where he is currently pursuing the Ph.D. degree with the Data Science and Intelligent Systems Group. Previously, he worked as Research Engineer for Knowledge Graph and Question Answering at the Fraunhofer IAIS, Dresden, and earlier as a Senior Data Analyst at the Analytics Department, Tata Steel, India.

ROMAN TEUCHER received the M.A. degree in linguistics from the University of Technology, Chemnitz. He is currently working as a Research Engineer at the Fraunhofer IAIS. His research interests include question answering over knowledge graphs, knowledge extraction, and machine reading comprehension.

LIUBOV KOVRIGUINA received the Candidate degree from St. Petersburg State University, in 2015. She has been working as an Assistant Professor at the ITMO University, Russia, and a Postdoctoral Researcher at InfAI, Leipzig, in 2020, before joining Fraunhofer IAIS, in 2020. She is currently a Research Engineer at the Fraunhofer IAIS, also focusing on incorporating knowledge graphs to dialogue systems and combining structured linguistic representations with neural language models. She took part in several research and industry projects, including voice interfaces for the Internet of Things, spoken language parsing, multimodal corpora development, and ASR evaluation. Her main research interests include NLP, artificial intelligence in general, machine learning, dialogue systems architectures, incompleteness handling in NLP pipelines, and spoken language understanding.

JENS LEHMANN received the Ph.D. degree *(summa cum laude)* from the University of Leipzig and the joint master's degree in computer science from the Technical University of Dresden and the University of Bristol. He is currently the Head of the Smart Data Analysis Research Group, a Full Professor with the University of Bonn, and a Lead Scientist with the Fraunhofer IAIS. He authored more than 100 publications, which were cited more than 18,000 times and have won 12 international awards. He contributed to various open-source projects, such as DL-Learner, SANSA, LinkedGeoData, and DBpedia. His research interests include semantic web technologies, question answering, machine learning, and knowledge graph analysis.