

Article

Geographic Knowledge Base Question Answering over OpenStreetMap

Jonghyeon Yang ^{1,†} , Hanme Jang ^{1,†}  and Kiyun Yu ^{1,2,*}

¹ Department of Civil and Environmental Engineering, Seoul National University, Seoul 08826, Republic of Korea; yangjonghyeon@snu.ac.kr (J.Y.); janghanie1@snu.ac.kr (H.J.)

² Institute of Engineering Research, Seoul National University, Seoul 08826, Republic of Korea

* Correspondence: kiyun@snu.ac.kr

† These authors contributed equally to this work.

Abstract: In recent years, question answering on knowledge bases (KBQA) has emerged as a promising approach for providing unified, user-friendly access to knowledge bases. Nevertheless, existing KBQA systems struggle to answer spatial-related questions, prompting the introduction of geographic knowledge base question answering (GeoKBQA) to address such challenges. Current GeoKBQA systems face three primary issues: (1) the limited scale of questions, restricting the effective application of neural networks; (2) reliance on rule-based approaches dependent on predefined templates, resulting in coverage and scalability challenges; and (3) the assumption of the availability of a golden entity, limiting the practicality of GeoKBQA systems. In this work, we aim to address these three critical issues to develop a practical GeoKBQA system. We construct a large-scale, high-quality GeoKBQA dataset and link mentions in the questions to entities in OpenStreetMap using an end-to-end entity-linking method. Additionally, we develop a query generator that translates natural language questions, along with the entities predicted by entity linking into corresponding GeoSPARQL queries. To the best of our knowledge, this work presents the first purely neural-based GeoKBQA system with potential for real-world application.

Keywords: GeoKBQA; KBQA; semantic parsing



Citation: Yang, J.; Jang, H.; Yu, K. Geographic Knowledge Base Question Answering over OpenStreetMap. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 10. <https://doi.org/10.3390/ijgi13010010>

Academic Editors: Wolfgang Kainz, Guohui Xiao, Yu Feng, Linfang Ding and Younes Hamdani

Received: 19 October 2023

Revised: 11 December 2023

Accepted: 21 December 2023

Published: 26 December 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Modern knowledge bases (KBs) contain a wealth of structured knowledge. For example, FREEBASE [1] contains over 45 million entities and 3 billion facts across more than 100 domains, while GOOGLE KNOWLEDGE GRAPH has amassed over 500 billion facts about 5 billion entities. The emergence of question answering on knowledge bases (KBQA) represents a significant advancement, offering users unified and simplified access to the vast and diverse knowledge stored in KBs while shielding them from their inherent complexities. As the scale and coverage of KBs increase, KBQA is becoming even more important due to the increasing difficulty of writing structured queries like SPARQL [2].

KBQA systems serve as intuitive natural language interfaces for users, transforming natural language queries into formalized query language. Initially, KBQA systems predominantly operated on rule-based mechanisms. They rely on predefined rules or templates to parse questions into logical forms [3,4], suffering from coverage and scalability problems. Recently, there has been a noticeable shift towards neural semantic parsing approaches in research. Particularly, the introduction of GrailQA [2] has marked a significant advancement in this field, featuring a large-scale, high-quality dataset consisting of 64,331 questions [2,5–9]. GrailQA diverges from pre-existing datasets, such as WebQSP [10] and ComplexWebQSP [11], which predominantly focused on i.i.d. assumptions. Instead, GrailQA boasts three levels of generalization, allowing the KBQA model to generalize novel compositions of seen schema items or even in a zero-shot fashion. Such characteristics

have prompted a wide array of research initiatives focuses toward resolving the intrinsic challenges in KBQA using GrailQA.

Geographic knowledge base question answering (GeoKBQA) is an extension of KBQA, specifically adapted to the geographic domain. Just like KBQA, GeoKBQA aims to provide a unified and user-friendly access to geographic knowledge bases, enabling users to extract detailed geographic information efficiently. It extends the functionality of KBQA by not only encompassing general knowledge but also integrating specific geographic knowledge about the world. For example, a user could ask questions like “How many pharmacies are in 200 m radius of High Street in Oxford?”. Structured data stored in KBs are more suitable than unstructured text retrieval to answer such questions. Using structured data, geographic question answering (GeoQA) systems can perform spatial (i.e., 200 m radius) and nonspatial operations (i.e., count how many), and retrieve information based on specific criteria (e.g., High Street in Oxford) [12].

Research in geographic knowledge base question answering (GeoKBQA) has been limited, with only a handful of studies addressing this area [12–14]. Ref. [13] pioneered work in this domain by developing GeoKBQA systems based on a template-based query generator, employing linked geospatial data. They introduced the GeoQuestions201 dataset, featuring 201 place-related question pairs, each paired with SPARQL/GeoSPARQL queries, and utilized templates to convert natural language questions into SPARQL/GeoSPARQL queries. Ref. [12] employed deep neural networks (DNNs), such as BERT, to transform natural language questions into sequences of encodings, using a template-based approach for converting these sequences into GeoSPARQL queries. Further contributing to the field, ref. [14] improved the GeoQuestions201 dataset, leading to the creation of GeoQuestions1089. This improved dataset includes more complex questions, demanding a more sophisticated understanding of both natural language and GeoSPARQL. Ref. [14] also enhanced the question-answering system introduced by ref. [13], which incorporated an entity linker that primarily utilizes string matching for entity disambiguation. All research in this domain consistently utilizes a template-based approach for query generation. Ref. [14] introduced a string-match-based entity linker in their system. This approach, despite facilitating the association of mentions with relevant entities in the knowledge bases, has its limitations owing due to the simplicity and inherent nature of string matching.

Compared to existing KBQA studies, GeoKBQA faces three principal challenges. First, GeoQuestions201 and GeoQuestions1089 are the only datasets specifically available for querying the geospatial knowledge base, are limited by their relatively small size. (When we say GeoKBQA dataset, we exclude the dataset with the query targeted on specific APIs such as the overpass API.) These datasets are insufficiently comprehensive for the practical development of KBQA systems based on DNNs. Second, the dominant methodologies in current studies are predominantly rule-based, concentrating on template-based translation of natural language into query language. These methods face significant challenges concerning coverage and scalability. Finally, most studies lack an entity linker, an essential component for GeoKBQA, with the only existing study employing hand-crafted features and rules. This limitation constrains the practical applicability of GeoKBQA systems.

In this study, we aim to address these challenges by developing a robust and practical GeoKBQA system. Our goal is to create an expansive GeoKBQA dataset centered around OpenStreetMap (OSM) and to establish a neural-based GeoKBQA system. To the best of our knowledge, our work is pioneering, being the first to introduce a neural-based approach supported by a substantial dataset.

The contributions of our work include:

- The creation of an unprecedented, extensive GeoKBQA dataset, encompassing 4468 questions enriched with diverse spatial functions, schema item compositions, and paraphrased natural language questions;

- The integration of advanced entity linking in GeoKBQA systems, enhancing their practical applicability. This is a notable advancement over previous works that either lacked an entity linker or relied on entity linkers with hand-crafted features and rules;
- The introduction of a novel neural-based GeoKBQA system, signifying a substantial departure from traditional rule-based systems, and augmented by our comprehensive dataset.

This paper is organized as follows: Section 2 reviews previous works related to KBQA, GeoKBQA, and geographic question datasets. In Section 3, we describe the process of constructing the GeoKBQA datasets and outline the GeoKBQA pipeline, providing detailed explanations for each stage of the process. Section 4 discusses the setup of GeoKBQA and the methodologies employed for evaluating the system's results. In Section 5, we present and analyze the results of the GeoKBQA system. Finally, Section 6 concludes the paper with a summary of our findings and an overview of the potential applications and limitations of our work. It also suggests directions for future research.

2. Related Works

2.1. KBQA

Knowledge base question answering (KBQA) has attracted significant attention recently due to its capacity to facilitate direct access to extensive knowledge graphs (KGs) without requiring specialized query-syntax knowledge. In the context of KBQA, given a natural language question, the system aims to retrieve an appropriate answer from the facts stored within the KG [15]. Most recent studies in the field of KBQA predominantly utilize datasets, such as WebQSP and GrailQA, to benchmark system performance. These specific datasets are primarily constructed based on Freebase, which includes over 100 domains, 45 million entities, and 3 billion facts, making it a robust resource for this purpose, despite its lack of current update [2].

KBQA systems typically employ semantic parsing approaches, where a natural language question is converted into an executable command in various formal languages such as SPARQL, lambda calculus, or S-expressions [9]. This approach encompasses several processes, including entity linking, schema retrieval, logical form enumeration, and transducer [5,6,8]. (Note that these methodologies employ neural networks to translate natural language into S-expressions but subsequently utilize a rule-based approach for converting S-expressions into SPARQL queries). Entity linking is a crucial process in KBQA, where explicit entity mentions are matched with corresponding entities in a KB. This process encompasses three main stages: mention detection, candidate generation, and entity disambiguation. Mention detection involves identifying entity mentions within questions. Candidate generation is concerned with finding possible entities that correspond to each mention, and entity disambiguation involves selecting the most suitable entity from the generated candidates. The schema retriever is another essential component, responsible for extracting relevant schema items that are either explicitly or implicitly referred to in the question. After the necessary KB items (entities, schema items) are retrieved, the transducer generates the logical form. Certain studies introduce logical form enumeration as a technique to facilitate the transducer in understanding the syntax of logical forms. Additionally, to ensure the syntactic correctness of the output program, mechanisms like checkers, grammar-based decoding, or constrained decoding have been utilized in some research.

ReTraCk [5] integrates three central components: the retriever, the transducer, and the checker. The retriever, which includes an entity linker and schema retriever, follows the entity-linking methodology described in ref. [2]. Starting with the identification of entity mentions through a BERT-based named entity recognition (NER) system, it then generates candidate entities along with their prior scores, utilizing an alias map mined from the KB and FACC1 [16]. For entity disambiguation, a prior baseline approach is employed, selecting the entity with the most substantial prior score. The schema retriever operates using a biencoder architecture with two distinct BERT-base encoders. The transducer

employs a bidirectional long short-term memory neural network (BiLSTM) encoder, paired with LSTM decoders. The checker component incorporates features like instance-level and ontology-level checking, as well as real and virtual execution capabilities.

RnG-KBQA [6] employs BERT-NER systems for mention detection and utilizes FACC1 for candidate generation. In their approach, the entity-disambiguation problem is transformed into an entity-ranking task, applying a BERT-base model as a cross-encoder. RnG-KBQA introduced a methodology involving logical form enumeration and ranking. This methodology enumerates potential logical forms, selects the top-k forms, and utilizes them as supplementary inputs for the transducer. In the transduction process, RnG-KBQA utilizes the T5 (text-to-text transfer transformer) in a sequence-to-sequence model. Ref. [8] adopts a pipeline akin to that of RnG-KBQA but integrates outputs from the entity linker, schema retriever, and logical form enumerator, directing them as inputs to the transducer model. Unlike RnG-KBQA, ref. [8] treats mention detection as a span-classification task, enhancing the performance of entity linking. The T5 is employed as a transducer, with the addition of constrained decoding implemented based on a prefix tree.

Contrastingly, refs. [7,9] adopt a dynamic program induction methodology to translate natural language into logical forms. This approach involves generating intricate programs through the cumulative expansion of a list of subprograms. Ref. [7] applies BERT as an encoder and LSTM as a decoder, interacting with the KB to incorporate information regarding permissible tokens during the encoding process. While decoding, the vocabulary is limited to a concise set of allowable tokens, influenced by the decoding history and conforming to predefined rules. In a slight variation, ref. [9] employs a rule-based logical form generator and utilizes language models primarily for ranking purposes, rather than token generation.

2.2. GeoKBQA

GeoKBQA has recently attracted increased interest from researchers due to its ability to handle place-related questions that necessitate sophisticated spatial reasoning, an aspect not thoroughly supported by conventional factoid question retrieval methods [12]. Recent advancements in GeoKBQA have predominantly utilized the GeoQuestions201 dataset for their research. GeoQuestions201 is interlinked with geospatial databases, such as DBpedia, OpenStreetMap, and the GADM dataset of global administrative areas. Unlike other KBQA datasets, GeoQuestions201 presents limitations in scale and quality, comprising only 201 questions. These questions were primarily crafted by third-year artificial intelligence students who were instructed to formulate “simple” questions by envisioning scenarios where geospatial information was essential. The unique constraints of GeoQuestions201 may pose challenges to the broader development and practical applicability of GeoKBQA systems. The introduction of the GeoQuestions1089 dataset marks a significant progression, incorporating questions that require an advanced understanding of both natural language and GeoSPARQL, thereby elevating semantic complexity. Despite these enhancements, certain methodological limitations persist. The significant reliance on human intervention in question generation introduces intrinsic restrictions. This approach results in a limited diversity of paraphrased natural language questions and reduces the scalability and diversity of schema items, which could inhibit the comprehensive development of GeoKBQA systems.

Similar to KBQA, GeoKBQA is also typically modeled as semantic parsing. In ref. [13], the authors implemented the first QA engine capable of answering questions with a geospatial dimension. They proposed a template-based query translator, translating natural language into SPARQL/GeoSPARQL queries using predefined templates. Additionally, they utilized AGDISTIS [17] for entity disambiguation, leveraging manually crafted features based on knowledge-graph structures and string similarity. Ref. [12] employed a rule-based query generator, utilizing neural-based constituency and dependency parsing methods to discern the structure of input questions and the relations among their tokens. They incorporated neural networks to determine each encoding in the parsed tree. How-

ever, despite utilizing DNNs across various components, the core of their query generator remained template based. Ref. [14] introduced GeoQA2, aligning with ref. [12] in employing dependency and constituency parsing to identify the structure of input questions and the relations among tokens. They also used a rule-based query generator, applying predefined templates filled with instances and concepts, to construct the final GeoSPARQL queries. A distinctive feature of GeoQA2 is the introduction of an entity linker based on TAGME [18], operating with hand-crafted features and rules. Ref. [19] compared GeoQA2 from refs. [12,14] using the GeoQuestions1089 dataset, underscoring the significance of the entity linker in [14]. Nonetheless, ref. [14] encounters limitations with TAGME, which relies on hand-crafted features and rules. This presents challenges in practical applications, as TAGME's contextual comprehension and adaptability are restrained compared to the DNN approaches that utilize learning-based features.

Note that, unlike existing KBQA systems that translate natural language into logical forms like s-expressions and then convert them into SPARQL queries using a rule-based approach, GeoKBQA studies primarily focus on converting natural language directly into executable GeoSPARQL queries.

2.3. Geographic Question Dataset

Research on geographic questions has typically employed datasets comprising a range of several hundred to a few thousand questions. Notable early contributions were made by [20,21], who focused on methods to parse approximately 880 questions that could interact with GEOBASE. GEOBASE is a specialized database that houses around 1000 geospatial facts related to the US geography domain, covering objects such as cities, states, rivers, and specific geometric properties like area and length. The questions were designed to extract information on various dimensions, such as the number of objects, population, and location, as well as topics related to humanities and social sciences.

Ref. [22] extracted 2500 questions randomly from approximately 1 million query logs of actual searches conducted on the Excite search engine and identified 464 as geo-related questions. A question was classified as geo related if it contained elements such as a place name, zip code, adjectives of place (e.g., international, western, north of), or terms such as city, site, street, island, or lake. Sanderson also organized specific prepositions into three categories: inclusion (in/at/from), direction (north/south/east/west), and vicinity (near/surrounding).

Ref. [23] crafted queries to extract geo-related objects from DBpedia and OSM data. These queries encompassed a variety of spatial questions, including proximity questions such as "Find hospitals outside and within 10 km of the city of Cardiff," crossing questions that ascertain whether a river or road intersects a specific administrative area, and containment questions that determine whether a spatial feature resides within a particular administrative region. Specifically tailored for Ohio State students, ref. [24] formulated spatial questions for the GeoQA system. A compilation of 800 questions was developed, encapsulating essential information about cities within Ohio, including coordinates, population, and other descriptive details. These questions were systematically categorized into five types, enhancing the efficacy of natural language processing and facilitating queries pertaining to names, coordinates, locations, and populations.

Ref. [13] collected spatial-related questions from third-year undergraduates and manually crafted SPARQL/GeoSPARQL queries for each question. The dataset included a total of 201 questions, broadly categorized into seven types such as location, spatial relationship, and attribute comparison. Following a similar methodology to [13,14] instructed third-year students in an AI course to develop 50 geospatial questions each, tailored to be compatible with querying databases on Wikipedia and OSM. From the accumulated 9335 questions, approximately 1000 were randomly selected, for which SPARQL/GeoSPARQL queries were manually written. These questions were systematically categorized into nine distinct types utilizing patterns, attributes, classes, instances, and aggregations.

3. Methodology

3.1. Basic Idea

Our methodology is organized into three critical subsections: dataset construction, entity retrieval, and target query generation. In dataset construction, the process of curating the dataset specifically for GeoKBQA is detailed. Following this, in entity retrieval, our methodology for linking entities is elucidated. The final subsection, target query generation, is dedicated to explaining our method for converting natural language into specific target queries. Figure 1 represents our pipeline for the GeoKBQA system, incorporating elements of entity retrieval and query generation. It is noteworthy that our approach diverges from conventional KBQA methods, which typically transition from natural language to s-expression, followed by the application of rule-based techniques to convert s-expression to SPARQL. Instead, our methodology leverages a transducer to directly translate natural language into corresponding SPARQL/GeoSPARQL queries. This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

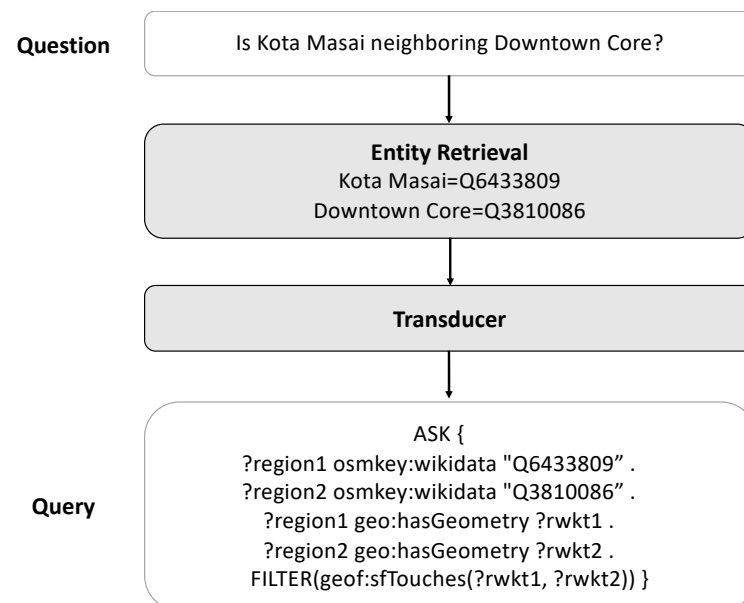


Figure 1. Our pipeline for the GeoKBQA system.

3.2. Dataset Construction

In this section, we present the methodology employed in the construction of our novel geospatial question dataset. A comprehensive analysis was conducted on existing geospatial question datasets to enhance the diversity of schema items, scale, and array of spatial functions, which are essential for the effective training of DNNs. Among the available geospatial question datasets, Questions1089 was selected as the baseline for our dataset due to its extensive collection of geospatial questions paired with SPARQL/GeoSPARQL queries. The procedure for constructing our dataset is organized as follows: 1. examination and analysis of existing geospatial questions to create foundational templates; 2. the substitution of entities, classes, and spatial functions to augment the diversity of schema items and spatial functions, and to expand the scale of the dataset; 3. paraphrasing of geospatial questions using ChatGPT (GPT-3.5 Turbo) to enhance natural language diversity; 4. selection of question–query pairs based on successful queries to the OpenStreetMap database, choosing pairs that yield results. Figure 2 illustrates the pipeline of our dataset-construction process.

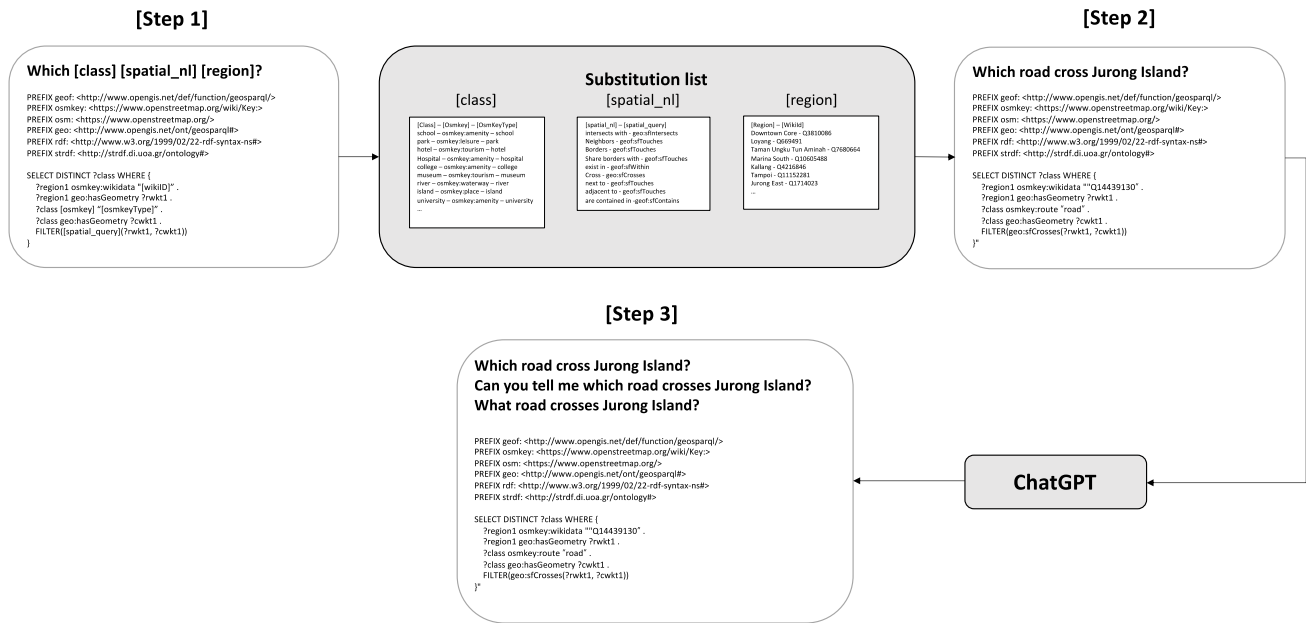


Figure 2. Dataset-construction pipeline.

In the initial step, we thoroughly analyzed the GeoQuestions1089 dataset to create foundational templates. Although questions in GeoQuestions1089 were grouped into nine categories, the corresponding Geo-SPARQL queries exhibited a lack of uniform structure within each category. This discrepancy necessitated a reclassification of questions based on the structural variance in their GeoSPARQL queries, leading to more in-depth analysis and the development of novel query templates. As depicted in Figure 1, masks were designed for each template, facilitating the substitution of geospatial entities, classes, and spatial functions in subsequent steps. In the second step, elements from predefined lists were randomly selected to replace each mask. Lists included geospatial entities such as countries, cities, and regions. In the case of geospatial classes, types like parks, universities, and banks were utilized, following the approach by [13]. A manually curated list of spatial functions was also applied to each template. The third step involved using ChatGPT to paraphrase questions, aiming to increase the diversity of natural language expressions in the dataset. This is crucial as GeoKBQA systems should effectively handle a variety of paraphrased sentences that maintain consistent meanings and the corresponding SPARQL/GeoSPARQL queries. In the final step, queries returning nonempty results were prioritized, allowing the exclusion of unrealistic questions from the dataset, thereby ensuring its relevance and applicability in practical scenarios.

3.3. Entity Retrieval

In contrast to existing KBQA studies, which treat entity linking as separate tasks—mention detection, candidate generation, and entity disambiguation—we adopted ELQ, the end-to-end entity-linking method proposed by [25], which jointly handles mention detection and entity disambiguation. To link mentions in the questions to the entities in OSM, we utilized entity information from Wikipedia. Specifically, for a given OSM entity, we followed its Wikipedia links to gather information, which was then provided to the entity-linking models. Figure 3 illustrates the question in conjunction with its corresponding entity-linking outcomes.

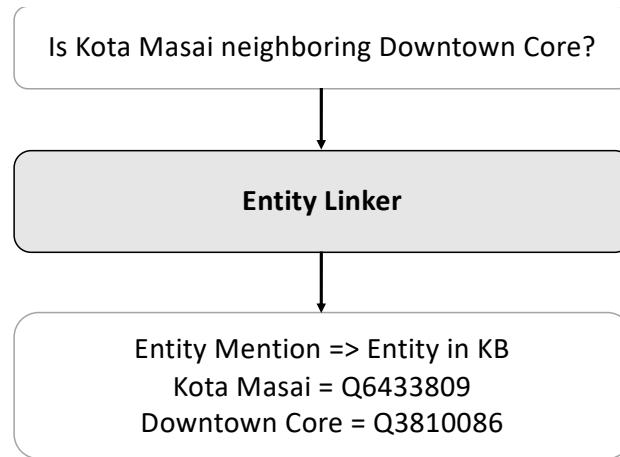


Figure 3. Entity-linking example.

3.3.1. Mention Detection

Given a question $[q_1 \dots q_n]^T = \text{BERT}([\text{CLS}]q_1 \dots q_n[\text{SEP}]) \in \mathbb{R}^{n \times h}$, we consider candidate mentions as all spans $[i, j]$ (i-th to j-th tokens of q) in the text up to length L .

To calculate the likelihood score of a candidate $[i, j]$ being an entity mention, we determine scores for each token, evaluating whether it is the start, part, or end of a mention using the following equations.

$$s_{\text{start}}(i) = w_{\text{start}}^T q_i \quad (1)$$

$$s_{\text{mention}}(t) = w_{\text{mention}}^T q_t \quad (2)$$

$$s_{\text{end}}(j) = w_{\text{end}}^T q_j \quad (3)$$

where $w_{\text{start}}^T, w_{\text{mention}}^T, w_{\text{end}}^T \in \mathbb{R}^h$ are learnable vectors.

The mention probabilities can be computed as

$$p([i, j]) = \sigma(s_{\text{start}}(i) + s_{\text{end}}(j) + \sum_{t=i}^j s_{\text{mention}}(t)) \quad (4)$$

3.3.2. Entity Disambiguation

For every $e_i \in \varepsilon$, we obtain entity representations

$$x_e = \text{BERT}_{[\text{CLS}]}([\text{CLS}]t(e_i)[\text{ENT}]d(e_i)[\text{SEP}]) \in \mathbb{R}^h \quad (5)$$

where $\text{BERT}_{[\text{CLS}]}$ indicates that we select the representation of $[\text{CLS}]$ token of the BERT embeddings, $t(e_i)$ is a title of the entity in Wikipedia, and $d(e_i)$ is a text description of the entity in Wikipedia.

To obtain a mention representation for each mention candidate $[i, j]$, we average $q_i \dots q_j$, and compute a similarity score s between the mention candidate and an entity candidate $e \in \varepsilon$

$$y_{i,j} = \frac{1}{(j-i+1)} \sum_{t=i}^j q_t \in \mathbb{R}^h \quad (6)$$

$$s(e, [i, j]) = x_e^T y_{i,j} \quad (7)$$

We then compute a likelihood distribution over all entities, conditioned on the mention $[i, j]$

$$p(e|[i, j]) = \frac{\exp(s(e, [i, j]))}{\sum_{e' \in \varepsilon} \exp(s(e', [i, j]))} \quad (8)$$

3.3.3. Training and Inference

We train the mention-detection and entity-disambiguation components jointly by optimizing the sum of their losses. A binary cross-entropy loss is used across all mention candidates.

$$L_{MD} = -\frac{1}{N} \sum_{1 \leq i \leq j \leq \min(i+L-1, n)} (y_{[i,j]} \log p([i,j]) + (1 - y_{[i,j]}) \log(1 - p([i,j]))) \quad (9)$$

whereby $y_{[i,j]} = 1$ if $[i, j]$ is a gold mention span, and 0 otherwise. N is the total number of candidates we consider.

The loss for entity disambiguation is expressed as:

$$L_{ED} = -\log p(e_g | [i, j]) \quad (10)$$

where e_g is the gold entity corresponding to mention $[i, j]$.

The inference step is as follows. Given an input question q , we use our mention-detection model to obtain our mention set

$$\mathcal{M} = \{[i, j] : 1 \leq i \leq j \leq \min(i + L - 1, n), p([i, j]) > \gamma\} \quad (11)$$

where γ is a threshold.

Then, for each $[i, j] \in \mathcal{M}$, and threshold according to γ , probability of mention $[i, j]$ linked to entity e is

$$p(e, [i, j]) = p(e | [i, j]) p([i, j]). \quad (12)$$

3.4. Target Query Generation

Our target query generation takes the question and entity-linking results as inputs, generating corresponding GeoSPARQL queries, as depicted in Figure 4. Following the standard pipeline of KBQA research, we utilized T5 [26]—a transformer-based seq2seq model—as the foundational transducer architecture. The input sequence is a concatenation of the natural language instruction, the input question, and the retrieved entities. For the natural language instruction, we used: “Translate the place-related questions into SPARQL/GeoSPARQL queries based on their semantics.” The T5 model is fine-tuned to generate the target sequence, minimizing the cross-entropy loss.

$$L_{gen} = -\sum_{t=1}^n \log(p(y_t | y_{<t}, x, e)) \quad (13)$$

where x denotes the input question (concatenated with the natural instructions), e denotes the retrieved entities, y denotes the target GeoSPARQL sequence, and n is the length of the target sequence.

While existing KBQA studies translate natural language into s-expressions, which are simpler than SPARQL queries, our work translates natural language into SPARQL/GeoSPARQL. To enhance the performance of translating natural language into GeoSPARQL, we experimented with variants of T5, which exhibited improved performance in various tasks. We tested three variants of T5: FLAN-T5 [27], CodeT5 [28], and CodeT5+ [29]. Since T5 was not originally pretrained with natural language instructions, we evaluated whether models fine tuned with instructions could enhance the performance of target query generation. Furthermore, viewing the translation of natural language into GeoSPARQL queries as a variant of code generation, we tested the CodeT5+ models, which are specifically trained for code understanding and generation.

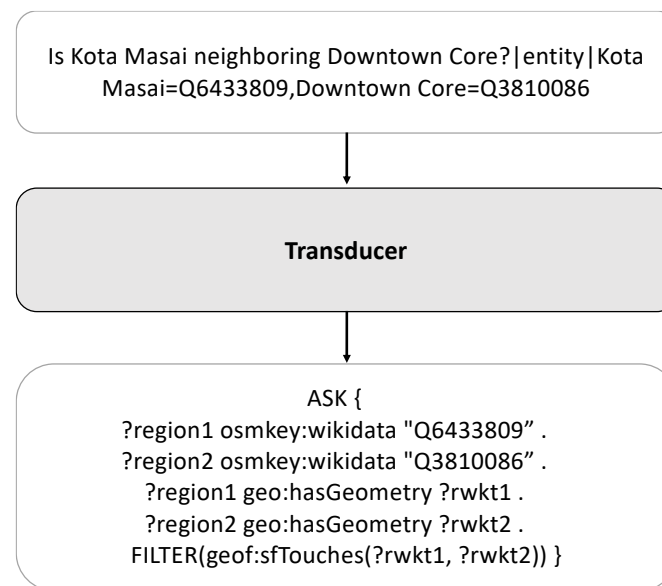


Figure 4. Query generation example.

4. Experimental Setup

4.1. Dataset

We constructed the dataset using the OSM TTL RDF dataset [30] in the Singapore region. Unlike Questions1089, which chose Great Britain as the research area, we selected Singapore—a slightly smaller region but one rich in urban spatial features and with an English-speaking demographic. We extracted entities that have Wikidata IDs in the OSM TTL. Following the dataset-construction pipeline demonstrated in Section 3.1, we created a dataset with 4468 geographic question and query pairs. For each question, we also constructed its mention and corresponding Wikidata ID for entity linking. We split the dataset into a training set with 3574 entries, a validation set of 446, and a test set of 448.

4.2. Evaluation Metrics

For performance evaluation, we utilized the string-based exact match (EM) score. Compared to the standard evaluation pipeline of [2], which used a graph-isomorphism EM score, we opted for the string-based EM score. This decision was made because our queries are in SPARQL/GeoSPARQL language, which is not a graph query language, and hence, implementing a graph-isomorphism exact match is not straightforward.

4.3. Implementation Details

Our models were implemented using PyTorch and Hugging Face transformers. For entity linking, we utilized the BERT-base [31] model across all cases, and for target query generation, we employed the T5-base model, following the approaches of [2,5,8]. Additionally, we conducted experiments using the T5-large model to assess whether scaling up the model size enhances performance. Each model was trained for 50 epochs, utilizing a learning rate of 5×10^{-5} and the AdamW optimizer [32]. We adopted a total batch size of 128, with gradient accumulation steps set to 2, and warm-up steps set to 200. The number of beams used was four. For training, we utilized bf16 16-bit mixed precision. For other hyperparameters, we adhered to the default options provided by Hugging Face transformers.

5. Results

5.1. Dataset-Construction Results

In our study, we generated a dataset of 4468 geographic question and query pairs using a specialized pipeline, as illustrated in Figure 5. This pipeline generates templates in

both natural language ('template_nl') and query format ('template_query'). Each template includes specific placeholders, denoted as 'masks', which indicate the type of data to be substituted. For instance, the mask '[country]' in the question template 'What is the population of [country]?' signifies that it should be replaced with an actual country name. The 'substitution list' serves two crucial functions. Firstly, it maps the masks in 'template_nl' to their counterparts in 'template_query', ensuring that a mask like '[country]' in the natural language template is appropriately paired with a '[wikiID]' mask in the query template. Secondly, this list specifies the potential substitutions for these masks, drawing from a range of entities, classes in the OpenStreetMap's TTL format, and spatial functions. As an example, the '[country]/[wikiID]' pair could be replaced with 'Indonesia' and 'Q252', respectively, in the respective masks.

Template_n1	Template_query	Substitution list
What is the population of [country]?	<p>PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key> PREFIX osm: <https://www.openstreetmap.org/></p> <p>SELECT DISTINCT ?population where { ?o osmkey:wikidata "[wikid]" . ?o osmkey:population ?population. }</p>	<div>[country/wikid]</div> <div> <div>[Country:]wikid]</div> <div>Indonesia - Q212</div> <div>Malaysia - Q833</div> </div>
Is [region_1] [spatial_n1] [region_2]?	<p>PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#></p> <p>ASK { ?region1 osmkey:wikidata "[wikid_1]" . ?region2 osmkey:wikidata "[wikid_2]" . ?region1 geo:hasGeometry ?rwk1 . ?region2 geo:hasGeometry ?rwk2 . FILTER(spatial_query{?rwk1, ?rwk2}) }</p>	<div>[region/wikid]</div> <div> <div>[Region:]wikid]</div> <div>Downtown Core - Q3810086</div> <div>Loyang - Q569493</div> <div>Taman Ungku Tun Amiah - Q7680664</div> <div>Marina South - Q39050488</div> <div>Kallang - Q4216846</div> <div>Tampoi - Q11512281</div> <div>Jurong East - Q1714023</div> <div>...</div> </div> <div>[spatial_n/spatial_query]</div> <div> <div>[spatial_n1] - [spatial_query_n1]</div> <div>near - >5000</div> <div>Close to - <5000</div> <div>Not far from - <5000</div> </div>
Which [class] [spatial_n1] [region]?	<p>PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#></p> <p>SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata "[wikid1]" . ?region1 geo:hasGeometry ?rwk1 . ?class [osmkey] ["osmkeyType"] . ?class geo:hasGeometry ?rwk1 . FILTER(spatial_query{?rwk1, ?rwk1}) }</p>	<div>[class/osmkey,osmkeytype]</div> <div> <div>[Class:]osmkey]</div> <div>[osmkeyType]</div> <div>school - osmkey:amenity - school</div> <div>park - osmkey:leisure - park</div> <div>hotel - osmkey:tourism - hotel</div> <div>Hospital - osmkey:amenity - hospital</div> <div>college - osmkey:amenity - college</div> <div>museum - osmkey:tourism - museum</div> <div>river - osmkey:waterway - river</div> <div>island - osmkey:place - island</div> <div>university - osmkey:amenity - university</div> <div>...</div> </div> <div>[spatial_n/spatial_query]</div> <div> <div>[spatial_n1] - [spatial_query_n1]</div> <div>intersects with - geo:intersects</div> <div>neighbors - geo:touches</div> <div>border - geo:touches</div> <div>Share borders with - geo:touches</div> <div>exist in - geo:within</div> <div>Cross - geo:crosses</div> <div>near to - geo:touches</div> <div>adjacent to - geo:touches</div> <div>are contained in - geo:contains</div> </div> <div>[region/wikid]</div> <div> <div>[Region:]wikid]</div> <div>Downtown Core - Q3810086</div> <div>Loyang - Q569493</div> <div>Taman Ungku Tun Amiah - Q7680664</div> <div>Marina South - Q39050488</div> <div>Kallang - Q4216846</div> <div>Tampoi - Q11512281</div> <div>Jurong East - Q1714023</div> <div>...</div> </div>
Are there any [class] in [region]	<p>PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#></p> <p>ASK WHERE { ?region1 osmkey:wikidata "[wikid1]" . ?region1 geo:hasGeometry ?rwk1 . ?class [osmkey] ["osmkeyType"] . ?class geo:hasGeometry ?rwk1 . FILTER(spatial_query{?rwk1, ?rwk1}) }</p>	<div>[class/osmkey,osmkeytype]</div> <div> <div>[Class:]osmkey]</div> <div>[osmkeyType]</div> <div>school - osmkey:amenity - school</div> <div>park - osmkey:leisure - park</div> <div>hotel - osmkey:tourism - hotel</div> <div>Hospital - osmkey:amenity - hospital</div> <div>college - osmkey:amenity - college</div> <div>museum - osmkey:tourism - museum</div> <div>river - osmkey:waterway - river</div> <div>island - osmkey:place - island</div> <div>university - osmkey:amenity - univer sity</div> <div>...</div> </div> <div>[x/spatial_query]</div> <div> <div>[x] - [spatial_query]</div> <div>geo:intersects</div> <div>geo:within</div> </div> <div>[region/wikid]</div> <div> <div>[Region:]wikid]</div> <div>Downtown Core - Q3810086</div> <div>Loyang - Q569493</div> <div>Taman Ungku Tun Amiah - Q7680664</div> <div>Marina South - Q39050488</div> <div>Kallang - Q4216846</div> <div>Tampoi - Q11512281</div> <div>Jurong East - Q1714023</div> <div>...</div> </div>
Which [class] are [distance_n1] [region]?	<p>PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/></p> <p>SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata "[wikid1]" . ?region1 geo:hasGeometry ?rwk1 . ?class [osmkey] ["osmkeyType"] . ?class geo:hasGeometry ?rwk1 . FILTER(geof:distance{?rwk1,?rwk1,uom:metre}[distance_query]) }</p>	<div>[class/osmkey,osmkeytype]</div> <div> <div>[Class:]osmkey]</div> <div>[osmkeyType]</div> <div>school - osmkey:amenity - school</div> <div>park - osmkey:leisure - park</div> <div>hotel - osmkey:tourism - hotel</div> <div>Hospital - osmkey:amenity - hospital</div> <div>college - osmkey:amenity - college</div> <div>museum - osmkey:tourism - museum</div> <div>river - osmkey:waterway - river</div> <div>island - osmkey:place - island</div> <div>university - osmkey:amenity - university</div> <div>...</div> </div> <div>[distance_n1/distance_query]</div> <div> <div>[distance_n1] - [distance_query]</div> <div>a short distance away from - > 5000</div> <div>less than 50m away from - < 5000</div> <div>within 20km of - > 20000</div> <div>near - > 5000</div> <div>far from - > 5000</div> <div>...</div> </div> <div>[region/wikid]</div> <div> <div>[Region:]wikid]</div> <div>Downtown Core - Q3810086</div> <div>Loyang - Q569493</div> <div>Taman Ungku Tun Amiah - Q7680664</div> <div>Marina South - Q39050488</div> <div>Kallang - Q4216846</div> <div>Tampoi - Q11512281</div> <div>Jurong East - Q1714023</div> <div>...</div> </div>

Figure 5. Generated template example.

Figure 6 presents a sample question and query pair produced by our pipeline. In this figure, the initial sentence under ‘question’ is the direct output from our substitution process. The subsequent sentences are paraphrases generated by ChatGPT, designed to maintain the original meaning. For instance, ‘Can you tell me the timezone of Singapore?’ is a paraphrase of the original question ‘What timezone is Singapore in?’. This demonstrates the consistency in meaning across different formulations of the question. In the ‘entity’ column, the ground truth entities used in the substitution are listed. It is important to note that this column exclusively contains entities and does not include spatial functions or classes that were part of the substitution process. To illustrate, the question ‘Which park is a short distance away from Bukit Indah?’ (refer to Figure 6) originates from the template ‘Which [class] are [distance_nl] from [region]?’ (refer to Figure 5). During the generation process, placeholders such as [class], [distance_nl], and [region] were substituted. However, in the ‘entity’ column, only the [region] placeholder is represented, as [class] and [distance_nl] do not constitute entities. The ‘query’ column displays the query generated

from template substitution. Here, the correspondence between the masks in ‘question’ and the substituted elements in the ‘query’ is evident. For example, when ‘Pengerang’ replaces the [region] mask, its corresponding [wikiID] (Q7162711) is used in the query, demonstrating the direct mapping between natural language placeholders and their query counterparts.

Question	Entity	Query
What timezone is Singapore in? Can you tell me the timezone of Singapore? Which timezone does Singapore belong to?	singapore=Q334	PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key:> PREFIX osm: <https://www.openstreetmap.org/> SELECT ?o where { ?country osmkey:wikidata "Q334". ?country osmkey:timezone ?o. }
What is the location of Pengerang? Can you tell me where Pengerang is located? Where can I find Pengerang? Where is Pontian located?	Pengerang=Q7162711	PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key:> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> SELECT ?rwkt WHERE { ?city osmkey:wikidata "Q7162711". ?city geo:hasGeometry ?rwkt. ?city rdf:type osm:node. }
Is Outram not far from Tanjong Pagar? Is Outram near Tanjong Pagar?	Outram=Q15077913,Tanjong Pagar=Q868368	PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key:> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/> ASK { ?region1 osmkey:wikidata ""Q15077913"". ?region2 osmkey:wikidata ""Q868368"". ?region1 geo:hasGeometry ?rwkt1. ?region2 geo:hasGeometry ?rwkt2. FILTER(geof:distance(?rwkt1,?rwkt2,uom:metre) < 5000) }
Which region exist in Seri Alam? What are the regions in Seri Alam?	Seri Alam=Q4854416	PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key:> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q4854416"". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:place "region". ?class geo:hasGeometry ?cwkt1. FILTER(geo:sfWithin(?rwkt1,?cwkt1)) }
Which park are a short distance away from Bukit Indah?	Bukit Indah=Q4986902	PREFIX geof: <http://www.opengis.net/def/function/geosparql/> PREFIX osmkey: <https://www.openstreetmap.org/wiki/Key:> PREFIX osm: <https://www.openstreetmap.org/> PREFIX geo: <http://www.opengis.net/ont/geosparql#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/> SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q4986902"". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:leisure "park". ?class geo:hasGeometry ?cwkt1. FILTER(geof:distance(?rwkt1,?cwkt1,uom:metre) <= 5000) }

Figure 6. Generated dataset example.

5.2. GeoKBQA Results

In evaluating our models, two distinct scenarios were considered: first, utilizing outputs from the entity-linking process as inputs, and second, employing golden entities

(accurate, ground truth entities) directly within the transducer. Figure 7 provides several illustrative examples of the operational flow in our GeoKBQA pipeline. For instance, upon receiving an input question such as ‘Which hotel is the nearest to Kangkar Pulai?’, the entity linker processes this input and yields a paired result, specifically the mention ‘Kangkar Pulai’ and its corresponding entity ID ‘Q6362781’. Subsequently, the input question is combined with the predictions from the entity linker. This is achieved by appending the identified entity information to the query, resulting in a combined input format: ‘Which hotel is nearest to Kangkar Pulai? | entity: Kangkar Pulai = Q6362781’. Moreover, in adherence to the original T5 model’s protocol, which involves prefixing inputs, we incorporated a specific input prefix. We prepend each question with the specific input prefix ‘translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question,’ indicating the task type being performed.

Results
<p>Input question Which hotel is the nearest to Kangkar Pulai?</p> <p>Entity prediction kangkar pulai=Q6362781</p> <p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: Which hotel is the nearest to Kangkar Pulai? entity : kangkar pulai=Q6362781</p> <p>Prediction SELECT DISTINCT?class WHERE { ?region1 osmkey:wikidata ""Q6362781". ?region1 geo:hasGeometry?rwkt1. ?class osmkey:tourism "hotel". Results ?class geo:hasGeometry?cwkt1. BIND(geof:distance(?rwkt1,?cwkt1,uom:metre) AS?distance) } ORDER BY ASC(?distance) LIMIT 1</p>
<p>Input question Which hospitals are within a short distance from Seletar?</p> <p>Entity prediction seletar=Q7447926</p> <p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: Which hospitals are within a short distance from Seletar? entity : seletar=Q7447926</p> <p>Prediction SELECT DISTINCT?class WHERE { ?region1 osmkey:wikidata ""Q7447926". ?region1 geo:hasGeometry?rwkt1. ?class osmkey:building "hospital". ?class geo:hasGeometry?cwkt1. FILTER(geof:distance(?rwkt1,?cwkt1,uom:metre) <= 5000) } }</p>
<p>Input question What are the forest and river of Potong Pasir?</p> <p>Entity prediction potong pasir=Q7235101</p> <p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: What are the forest and river of Potong Pasir? entity : potong pasir=Q7235101</p> <p>Prediction SELECT DISTINCT?class1?class2 WHERE { ?region1 osmkey:wikidata ""Q7235101". ?region1 geo:hasGeometry?rwkt1. ?class1 osmkey:landuse "forest". ?class1 geo:hasGeometry?cwkt1. ?class2 osmkey:waterway "river". ?class2 geo:hasGeometry?cwkt2. FILTER(geo:sfintersects(?rwkt1,?cwkt1) && geo:sfintersects(?rwkt1,?cwkt2)) } }</p>

Figure 7. GeoKBQA system prediction samples. The samples are the result of CodeT5+ 770M models. We only demonstrated the correct predictions and, therefore, omit the ground truth.

Table 1 presents a detailed evaluation of SPARQL/GeoSPARQL generation within the GeoKBQA pipeline, as depicted in Figure 7. The column labeled ‘ELQ’ specifically identifies instances where ELQ functioned as the entity linker. The analysis reveals a distinct and consistent trajectory of performance improvement across various model configurations. This progressive enhancement, evident in the transition from the T5 base model through FLAN-T5, CodeT5, and culminating in CodeT5+, is also reflected in their respective large model counterparts. Such a pattern robustly illustrates the correlation between advancements in pretraining strategies, spanning from T5 to CodeT5+, and the increased efficiency in translating questions into SPARQL/GeoSPARQL queries.

Table 1. Query generation results of T5-base and its variants.

	T5 Base Model (220M)		T5 Large Model (770M)	
	ELQ	Golden	ELQ	Golden
T5	51.7857	71.6518	75.8929	79.0179
FLAN-T5	62.2768	73.4375	78.5714	80.5804
CodeT5	84.375	83.4821	91.5179	92.6339
CodeT5+	93.3036	92.4107	93.0804	94.1964

Moreover, a key observation from the data is the consistent improvement in performance observed when scaling up within each model series. Specifically, for the T5, FLAN-T5, CodeT5, and CodeT5+ models, moving from the base version to the larger variant is invariably linked with enhanced performance. This trend highlights the direct impact of model scaling on overall performance, demonstrating that increasing the model size within the same series significantly contributes to improved efficiency in translating questions into SPARQL/GeoSPARQL queries.

Significantly, the performance disparity between the ELQ and golden entity scenarios narrows as the model’s performance improves. This can be attributed to the presence of paraphrased questions in the training set that mirror queries in the test set. As the model’s size increases, its capacity to memorize entities from the training set improves. This phenomenon and its implications are further exemplified in Figure 8. To avoid overestimating performance, it is crucial to strategically and deliberately divide the training and test sets, rather than depending on random segmentation.

The ‘Golden’ column in Table 1 illustrates the transducer’s performance when provided with accurate (‘golden’) entity inputs. Consistent with previous analyses, this column reaffirms that advancements in pretraining strategies—from T5 to FLAN-T5, CodeT5, and then to CodeT5+—correlate positively with the efficiency of translating questions into queries. Additionally, an increase in model capacity (either through improved pretraining or scaling), appears to further improve performance, likely due to better memorization of schema items. Examples demonstrating this capability and its potential challenges are also presented.

Figure 8 presents several failure cases encountered in our GeoKBQA system. In the first example, the entity is not correctly identified, leading to an input that lacks specific entity information. Consequently, the T5-base 220M model fails to accurately translate the natural language into the correct query, instead substituting a miscellaneous entity ‘Q14489267’. In contrast, the CodeT5+ 220M model, with its enhanced capacity to memorize entity information from the training set, successfully retrieves the correct entities and generates accurate results. This discrepancy highlights a potential overestimation of performance in advanced models.

The second example illustrates a different challenge: the T5-base 220M model’s inability to identify the correct schema. It incorrectly predicts ‘?class2 osmkey:natural “park”’, whereas the ground truth is ‘?class2 osmkey:leisure “park”’. This error underscores the struggles of smaller models in accurately retrieving the correct schema, suggesting the

potential need for an additional schema retriever. Addressing these issues is crucial for advancing the effectiveness of such systems in future research.

Results	
T5-base 220M	CodeT5+ 220M
<p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: Can you tell me which villages are within 20km of the Western Islands? entity ;</p> <p>Target SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q7987985". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:place "village". ?class geo:hasGeometry ?cwkt1. FILTER (geo:distance(?rwkt1,?cwkt1,uom:metre) <= 20000) }</p> <p>Prediction SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q14489267". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:place "village". ?class geo:hasGeometry ?cwkt1. FILTER (geo:distance(?rwkt1,?cwkt1,uom:metre) <= 20000) }"</p>	<p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: Can you tell me which villages are within 20km of the Western Islands? entity ;</p> <p>Target SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q7987985". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:place "village". ?class geo:hasGeometry ?cwkt1. FILTER (geo:distance(?rwkt1,?cwkt1,uom:metre) <= 20000) }</p> <p>Prediction SELECT DISTINCT ?class WHERE { ?region1 osmkey:wikidata ""Q7987985". ?region1 geo:hasGeometry ?rwkt1. ?class osmkey:place "village". ?class geo:hasGeometry ?cwkt1. FILTER (geo:distance(?rwkt1,?cwkt1,uom:metre) <= 20000) }"</p>
<p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: What are the names of the forest and park in Tuas? entity : tuas=Q7850658</p> <p>Target SELECT DISTINCT ?class1 ?class2 WHERE { ?region1 osmkey:wikidata ""Q7850658". ?region1 geo:hasGeometry ?rwkt1. ?class1 osmkey:landuse "forest". ?class1 geo:hasGeometry ?cwkt1. ?class2 osmkey:leisure "park". ?class2 geo:hasGeometry ?cwkt2. FILTER (geo:sfIntersects(?rwkt1,?cwkt1) && geo:sfIntersects(?rwkt1,?cwkt2)) }</p> <p>Prediction SELECT DISTINCT ?class1 ?class2 WHERE { ?region1 osmkey:wikidata ""Q7850658". ?region1 geo:hasGeometry ?rwkt1. ?class1 osmkey:landuse "forest". ?class1 geo:hasGeometry ?cwkt1. ?class2 osmkey:natural "park". ?class2 geo:hasGeometry ?cwkt2. FILTER (geo:sfIntersects(?rwkt1,?cwkt1) && geo:sfIntersects(?rwkt1,?cwkt2)) }"</p>	<p>Input translate the place-related questions into SPARQL/GeoSPARQL queries depending on their semantics and specific requirements of each question: What are the names of the forest and park in Tuas? entity : tuas=Q7850658</p> <p>Target SELECT DISTINCT ?class1 ?class2 WHERE { ?region1 osmkey:wikidata ""Q7850658". ?region1 geo:hasGeometry ?rwkt1. ?class1 osmkey:landuse "forest". ?class1 geo:hasGeometry ?cwkt1. ?class2 osmkey:leisure "park". ?class2 geo:hasGeometry ?cwkt2. FILTER (geo:sfIntersects(?rwkt1,?cwkt1) && geo:sfIntersects(?rwkt1,?cwkt2)) }</p> <p>Prediction SELECT DISTINCT ?class1 ?class2 WHERE { ?region1 osmkey:wikidata ""Q7850658". ?region1 geo:hasGeometry ?rwkt1. ?class1 osmkey:landuse "forest". ?class1 geo:hasGeometry ?cwkt1. ?class2 osmkey:leisure "park". ?class2 geo:hasGeometry ?cwkt2. FILTER (geo:sfIntersects(?rwkt1,?cwkt1) && geo:sfIntersects(?rwkt1,?cwkt2)) }"</p>

Figure 8. GeoKBQA system failure cases.

6. Conclusions

This study constructed the first practical GeoKBQA dataset and GeoKBQA systems employing DNNs. Prior studies had constructed GeoKBQA datasets that were small scale and of low quality. They also developed rule-based GeoKBQA systems, which were impractical due to their limited coverage and scalability issues. Specifically, these previous systems utilized a rule-based approach to identify entities and generate queries from place-related questions. To our knowledge, this is the first work to construct a fully neural-based GeoKBQA system using large-scale datasets.

To build a practical GeoKBQA system, we initially constructed a large-scale GeoKBQA dataset, enriched with diverse spatial functions, schema item compositions, and paraphrased natural language questions. In the GeoKBQA system, we first conducted entity linking on the natural language questions. Subsequently, we trained the query generation model using input questions concatenated with entity-linking results and the ground truth SPARQL/GeoSPARQL queries. For entity linking, we utilized information from Wikidata that is connected to the OSM object, performing an end-to-end entity-linking process. For query generation, we employed various versions of T5 to enhance the performance of query generation.

By training various T5 models, we concluded that using pretrained models with capabilities in code understanding and generation tasks substantially aids in generating precise queries. Additionally, we found that scaling the model size effectively improves the

quality of query generation. Utilizing CodeT5+ 220M with ELQ enabled us to develop a robust GeoKBQA system capable of proficiently translating place-related questions into GeoSPARQL queries.

Despite being the pioneer in proposing a neural-based, practical GeoKBQA system, this research has encountered several limitations. Due to the limited schema of OpenStreetMap, the schema retriever component was not included in this study. Furthermore, we did not deliberate on how to ensure that the seq2seq models conform precisely with the SPARQL/GeoSPARQL syntax. Lastly, a careful and strategic division of the dataset is crucial for an accurate evaluation of the GeoKBQA systems' performance. Our subsequent works aim to resolve these existing challenges.

Author Contributions: Conceptualization, Jonghyeon Yang and Hanme Jang; methodology, Jonghyeon Yang and Hanme Jang; software, Jonghyeon Yang and Hanme Jang; validation, Jonghyeon Yang and Hanme Jang; formal analysis, Jonghyeon Yang and Hanme Jang; investigation, Jonghyeon Yang and Hanme Jang; resources, Kiyun Yu; data curation, Jonghyeon Yang and Hanme Jang; writing—original draft preparation, Jonghyeon Yang and Hanme Jang; writing—review & editing, Jonghyeon Yang and Hanme Jang; visualization, Jonghyeon Yang and Hanme Jang; supervision, Kiyun Yu; project administration, Kiyun Yu; funding acquisition, Kiyun Yu. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2022-00143336).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008; pp. 1247–1250.
2. Gu, Y.; Kase, S.; Vanni, M.; Sadler, B.; Liang, P.; Yan, X.; Su, Y. Beyond IID: Three levels of generalization for question answering on knowledge bases. In Proceedings of the Web Conference, Ljubljana, Slovenia, 19–23 April 2021; Volume 2021, pp. 3477–3488.
3. Cabrio, E.; Cojan, J.; Aprosio, A.P.; Magnini, B.; Lavelli, A.; Gandon, F. QAKiS: An open domain QA system based on relational patterns. In Proceedings of the International Semantic Web Conference, Boston, MA, USA, 11–15 November 2012.
4. Abujabal, A.; Roy, R.S.; Yahya, M.; Weikum, G. Quint: Interpretable question answering over knowledge bases. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Copenhagen, Denmark, 9–11 September 2017; pp. 61–66.
5. Chen, S.; Liu, Q.; Yu, Z.; Lin, C.Y.; Lou, J.G.; Jiang, F. ReTraCk: A flexible and efficient framework for knowledge base question answering. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, Online, 1–6 August 2021; pp. 325–336.
6. Ye, X.; Yavuz, S.; Hashimoto, K.; Zhou, Y.; Xiong, C. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv* **2021**, arXiv:2109.08678.
7. Gu, Y.; Su, Y. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. *arXiv* **2022**, arXiv:2204.08109.
8. Shu, Y.; Yu, Z.; Li, Y.; Karlsson, B.F.; Ma, T.; Qu, Y.; Lin, C.Y. Tiara: Multi-grained retrieval for robust question answering over large knowledge bases. *arXiv* **2022**, arXiv:2210.12925.
9. Gu, Y.; Deng, X.; Su, Y. Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments. *arXiv* **2022**, arXiv:2212.09736.
10. Yih, W.T.; Richardson, M.; Meek, C.; Chang, M.W.; Suh, J. The value of semantic parse labeling for knowledge base question answering. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; Volume 2, pp. 201–206.
11. Talmor, A.; Berant, J. The web as a knowledge-base for answering complex questions. *arXiv* **2018**, arXiv:1803.06643.
12. Hamzei, E.; Tomko, M.; Winter, S. Translating place-related questions to GeoSPARQL queries. In Proceedings of the ACM Web Conference, Lyon, France, 25 April 2022; pp. 902–911.
13. Punjani, D.; Singh, K.; Both, A.; Koubarakis, M.; Angelidis, I.; Bereta, K.; Beris, T.; Bilidas, D.; Ioannidis, T.; Stamoulis, G.; et al. Template-based question answering over linked geospatial data. In Proceedings of the 12th Workshop on Geographic Information Retrieval, Seattle, WA, USA, 6 November 2018; pp. 1–10.

14. Kefalidis, S.A.; Punjani, D.; Tsalapati, E.; Plas, K.; Pollali, M.; Mitsios, M.; Tsokanaridou, M.; Koubarakis, M.; Maret, P. Benchmarking geospatial question answering engines using the dataset GeoQuestions1089. In *International Semantic Web Conference*; Springer: Cham, Switzerland, 2023.
15. Ravishankar, S.; Thai, J.; Abdelaziz, I.; Mihidukulasooriya, N.; Naseem, T.; Kapanipathi, P.; Rossiello, G.; Fokoue, A. A two-stage approach towards generalization in knowledge base question answering. *arXiv* **2021**, arXiv:2111.05825.
16. Gabrilovich, E.; Ringgaard, M.; Subramanya, A. *FACC1: Freebase Annotation of ClueWeb Corpora*, Version 1. 2013.
17. Usbeck, R.; Ngonga Ngomo, A.C.; Röder, M.; Gerber, D.; Coelho, S.A.; Auer, S.; Both, A. AGDISTIS-graph-based disambiguation of named entities using linked data. In *Proceedings of the Semantic Web–ISWC 2014: 13th International Semantic Web Conference*, Riva del Garda, Italy, 19–23 October 2014; Part I 13. pp. 457–471.
18. Ferragina, P.; Scaiella, U. Tagme: On-the-fly annotation of short text fragments (by Wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, Shanghai, China, 3–7 November 2010; pp. 1625–1628.
19. Punjani, D.; Kefalidis, S.A.; Plas, K.; Tsalapati, E.; Koubarakis, M.; Maret, P. The Question Answering System GeoQA2. In *Proceedings of the 2nd International Workshop on Geospatial Knowledge Graphs and GeoAI: Methods, Models, and Resources*, Leeds, UK, 12 September 2023.
20. Zelle, J.M.; Mooney, R.J. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR, USA, 4–8 August 1996; pp. 1050–1055.
21. Tang, L.R.; Mooney, R.J. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, Freiburg Germany, 5–7 September 2001; pp. 466–477.
22. Sanderson, M.; Kohler, J. Analyzing geographic queries. In *Proceedings of the ACM SIGIR Workshop on Geographic Information Retrieval*, Sheffield, UK, 25–29 July 2004; Volume 2, pp. 8–10.
23. Younis, E.M.; Jones, C.B.; Tanasescu, V.; Abdelmoty, A.I. Hybrid geo-spatial query methods on the Semantic Web with a spatially-enhanced index of DBpedia. In *Proceedings of the Geographic Information Science: 7th International Conference, GIScience 2012*, Columbus, OH, USA, 18–21 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7478, pp. 340–353.
24. Chen, W. Developing a Framework for Geographic Question Answering Systems Using GIS, Natural Language Processing, Machine Learning, and Ontologies. Ph.D. Thesis, The Ohio State University, Columbus, OH, USA, 2014.
25. Li, B.Z.; Min, S.; Iyer, S.; Mehdad, Y.; Yih, W.T. Efficient one-pass end-to-end entity linking for questions. *arXiv* **2020**, arXiv:2010.02413.
26. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
27. Chung, H.W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. Scaling instruction-finetuned language models. *arXiv* **2022**, arXiv:2210.11416.
28. Wang, Y.; Wang, W.; Joty, S.; Hoi, S.C. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv* **2021**, arXiv:2109.00859.
29. Wang, Y.; Le, H.; Gotmare, A.D.; Bui, N.D.; Li, J.; Hoi, S.C. Codet5+: Open code large language models for code understanding and generation. *arXiv* **2023**, arXiv:2305.07922.
30. Bast, H.; Brosi, P.; Kalmbach, J.; Lehmann, A. An efficient RDF converter and SPARQL endpoint for the complete OpenStreetMap data. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, Beijing, China, 2–5 November 2021; pp. 536–539.
31. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.