Location Query Answering Using Box Embeddings

Eleni Tsalapati[1], Markos Iliakis[1] and Manolis Koubarakis[1]

[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece

Abstract

A standard way to retrieve knowledge from geospatial knowledge graphs is by using the query language SPARQL and its geospatial extensions, such as GeoSPARQL and stSPARQL. However, this approach presupposes that the queried knowledge graph is complete, which is rarely the case. A promising recent approach to the problem of query answering over incomplete knowledge graphs is by employing embedding-based techniques. In this paper, we have developed the embedding-based geospatial query answering model, SQABo, which encodes conjunctive geospatial queries as boxes in an embedding space and returns as answers the entities found within the box. Experiments on two geospatial knowledge graphs (YAGO2geo and DBGeo) show that our approach offers superior performance when compared with related techniques published in the recent literature.

Keywords

Geospatial knowledge graphs, Geospatial query answering, Geospatial knowledge graph embeddings

1. Introduction

Geospatial data and knowledge have become ubiquitous on the Web today and in applications such as navigation, smart cities, Earth observation, etc. To retrieve efficiently such geospatial knowledge, several geospatial knowledge graphs (KGs) have been proposed in the literature (e.g., YAGO2geo [1], WorldKG [2] and KnowWhereGraph [3]). *Geospatial knowledge graphs* enable the representation of geospatial knowledge in a semantically enriched, formal, and structured way using ontologies and the RDF data model.

Standard query answering engines that retrieve knowledge from a geospatial KGs pose SPARQL, GeoSPARQL [4] or stSPARQL [5] queries using RDF stores that support precise geospatial data (e.g., Strabon [5], Strabo 2 [6] or GraphDB[1]). However, the evaluation of such queries can sometimes be a very time-consuming process. For instance, for identifying the countries bordering Greece, an engine like the above may compare the geometry of Greece with the geometry of *each* country of the targeted knowledge graph. On top of this, these engines assume that the targeted knowledge graph is correct and complete, which is rarely the case, for instance, in the above example, the geometries of some countries may be inaccurately defined. This can be due to the intrinsic noise of crowd-sourced data (e.g., OpenStreetMap) used to construct the queried knowledge graph, or due to the intrinsic vagueness [7] of geospatial knowledge graphs: the shape of some geospatial features (e.g., mountain or valley) cannot be

Workshop at ISWC 2023: Deep Learning for Knowledge Graphs, November 6-10, Athens, Greece.

etsalapati@gmail.com (E. Tsalapati); miliakis@di.uoa.gr (M. Iliakis); koubarak@di.uoa.gr (M. Koubarakis)

0000-0001-9464-404X (E. Tsalapati); 0000-0002-5342-2106 (M. Iliakis); 0000-0002-1954-8338 (M. Koubarakis)

CEUR Workshop Proceedings (CEUR-WS.org)

[1]http://graphdb.ontotext.com/documentation/free/

precisely defined (e.g., borders of a mountain or a valley) and in cases that these are defined, timeliness (e.g., the sizes of cities can change through time) and administrative definitions (e.g., boundaries of administrative regions can change) often affect the query answering results.

More recent approaches (e.g., [8, 9]) address KG incompleteness and vagueness by employing embedding techniques. In general, embedding-based models compute vector embeddings for the queries and identify the answers by performing nearest-neighbor search in the latent space. Hence, they can handle queries for which the data required for their answering may not be explicitly present in the KG, instead they utilize implicit semantics captured from the embeddings. SE-KGE [10] was the first embedding-based model for conjunctive query answering over geospatial KGs. SE-KGE aims to learn to encode a given query in such a way that its embedding will be the closest one to the embedding of the correct answer to the query. However, as it is pointed out by Ren et al. [11], single-point embeddings cannot enclose the set of entities related to the query, while it is unclear how to perform logical operations, like conjunction, over points in the latent space.

In this paper, we develop the novel geospatial query answering model SQABo (geoSpatial conjunctive graph Query Answering based on KG embeddings with Boxes). SQABo initially encodes entities and relations appearing in an input conjunctive query utilizing the entity encoders introduced by Mai et al. [10]. Then, these vectors are gradually projected into boxes customizing the Query2Box approach presented in [11] to also incorporate geospatial information. The answer to an input conjunctive query is computed by intersecting these boxes and returning the entities within the final box. For the intersection, we optimized the algorithm of [11] by employing the graph self-attention mechanism introduced in [12]. Experimental results over YAGO2geo [13] and over the geospatial fragment DBGeo$_{\mathcal{G}}$ of DBGeo [10] show that SQABo delivers better performance than SE-KGE: 93.5% over 87.9% macro-averaged APR score on YAGO2geo and 87.4% over 82.9% macro-averaged APR score on DBGeo$_{\mathcal{G}}$.

The remainder of the paper is structured as follows. Section 2 introduces our terminology and notation. Then, after an overview of related work in Section 3, the architecture of SQABo is presented in detail in Section 4. In Section 5 we describe the pre-processing methods used to create the dataset for training the model, the training method and our experiments. Conclusions and future work are discussed in Section 6.

2. Preliminaries

A *knowledge graph* $\mathcal{G}\langle \mathcal{E} \cup \mathcal{L}, \mathcal{R} \rangle$ over a vocabulary consisting of disjoint sets of entities $\mathcal{E}$, literal values $\mathcal{L}$ and relations $\mathcal{R}$, is a finite set of *triples* (alternatively, *facts*) $\langle e_1, r, e_2 \rangle$ where $e_1 \in \mathcal{E}$, $e_2 \in \mathcal{E} \cup \mathcal{L}$ and $r \in \mathcal{R}$. Entity e_1 is called the *subject* (alternatively, *head*), entity e_2 is called the *object* (alternatively, *tail*) and r is called the relation of the triple. The set of entities $\mathcal{E}$ consists of a disjoint set of *classes* $\mathcal{C}$ and *individuals* $\mathcal{I}$.

In GIS terminology [14], a *geographic feature*, or simply *feature*, is an abstraction of a real-world phenomenon (e.g., the airport of Athens) and can have various attributes that describe its *thematic* (e.g., name, the company that manages it) and *spatial* (e.g., location on Earth) characteristics. A *point* $\mathbf{p}$ is a pair $(x, y) \in \mathbb{R}^2$. Points represent locations in the Cartesian coordinate space, which is the coordinate space used in this paper. A *bounding box* is the

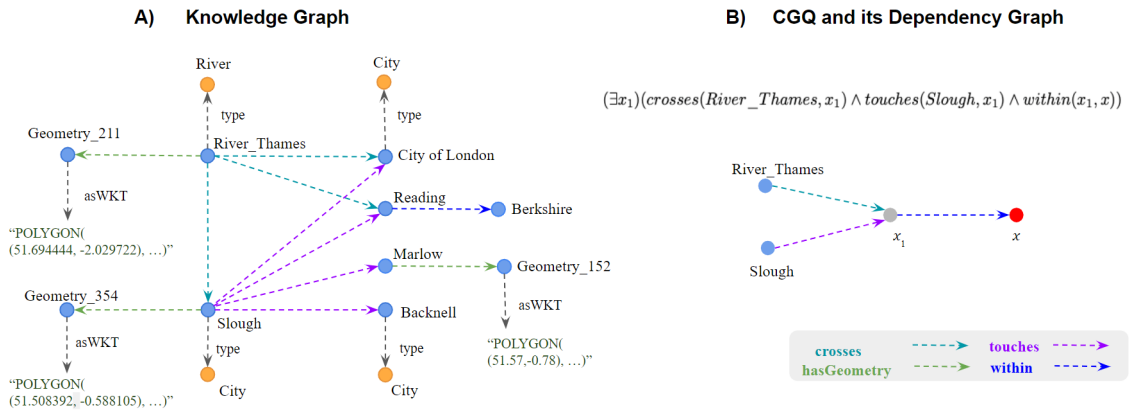

Figure 1: A) A geospatial KG example containing knowledge about the river Thames and some of the places it crosses. The geospatial entities of the KG are represented as blue nodes and the types (classes) of the entities as orange nodes. B) Formalisation of the query: *"Which places contain areas that are crossed by Thames and border Slough?"* and the corresponding dependency graph. With blue nodes we represent the entities of the KG, with grey the existential variables and with red the answer variable (target node).

smallest (with respect to the subset relation) rectangle enclosing a geographic feature. Naturally, for any bounding box $\mathcal{A}$, we have $\mathcal{A} \subset \mathbb{R}^2$. As it is standard [10], a bounding box is represented by the pair $[\vec{x}; \vec{y}] \in \mathbb{R}^4$, where $\vec{x}, \vec{y} \in \mathbb{R}^2$ are its southwest and northeast points. A *study area* is a bounding box that contains geographic features under study.

A *geospatial knowledge graph* over a study area $\mathcal{A}$ is a knowledge graph $\mathcal{G}\langle \mathcal{E} \cup \mathcal{L}, \mathcal{R} \rangle$ such that the set of entities $\mathcal{E}$ contains a non-empty set $\mathcal{E}_\mathcal{G}$ of *geospatial entities*. The geospatial entities in $\mathcal{E}_\mathcal{G}$ are of two kinds: locations represented by a point in the study area $\mathcal{A}$ or regions represented by a bounding box $[\vec{x}; \vec{y}]$, with $\vec{x}, \vec{y} \in \mathcal{A}$.

An example of a geospatial KG is presented in Figure 1A), where knowledge about river Thames, the areas that it crosses and areas neighbouring to Slough is illustrated. Here, $\mathcal{E}$ represents all nodes appearing in the graph (e.g., City, River, River_Thames, Geometry_211), $\mathcal{L}$ the polygons (POLYGON((51.69444,-2.029722),..)) and $\mathcal{C}$ the classes (City, River, denoted with orange nodes) of the KG. *Conjunctive graph queries* (CGQ) is a subclass of first-order logic queries over a knowledge graph that may include only the existential quantifier ($\exists$), the conjunction operator ($\wedge$), and have a single answer variable. CGQs can be defined as follows. Let $\mathcal{E}$ be a set of entities, $\mathcal{R}$ a set of relations and $\mathcal{V}$ a set of variables. We assume that the three sets $\mathcal{E}, \mathcal{R}$ and $\mathcal{V}$ are pairwise disjoint. A *CGQ query* $Q(x)$ with answer variable (or *target node*) $x \in \mathcal{V}$ is a formula of the form

$$(\exists x_1)(\exists x_2) \cdots (\exists x_n)(\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_m) \tag{1}$$

where for each $1 \leq i \leq m$, ϕ_i is of the form $r(e, x_k)$ or of the form $r(x_k, x_l)$. We also have $r \in \mathcal{R}$, $e \in \mathcal{E}$, $x_k, x_l \in \mathcal{V}$, $x_k \neq x_l$ and x appears in some of the ϕ_i.

The *dependency graph* of a CGQ $Q(x)$ is a directed graph whose nodes are the terms (i.e., entities or variables) occurring in the query and where there is a directed edge from term t_1 to term t_2 for each atom of the form $r(t_1, t_2)$ occurring as a sub-formula in the query. According

to [15], for a CGQ to be *valid* (i.e., there are no contradictions or redundancies), its dependency graph must be a *directed acyclic graph (DAG)*, where each source node in the graph is an entity (thus not a variable) and the answer variable is the single sink node. An example CGQ with its dependency graph is presented in Figure 1B).

3. Related Work

A simple query answering task is link prediction: calculate the most likely answers to a query of the form $r(e, x)$ or $r(x, e)$ where x is an answer variable, r is a relation and e is an entity in the knowledge graph. There are three main approaches for link prediction: (1) translation-based methods (e.g., TransE [16] and TransH [17]), which translate the head entity and the tail entity of a triple using the relation of the triple and, then, they use a distance function to evaluate the embedding or to score the reliability of the predicted fact; (2) semantic matching-based methods (e.g., RESCAL [18] and ComplEx [19]) that use similarity-based scoring functions, by matching latent semantics of entities and relations embodied in their vector space representations; and (3) methods that embed the entities based, also, on the local structure of the graph using graph convolutional networks [20] (e.g., R-GCN [21] and TransGCN [22]). Further details about link prediction models can be found in [23], [24] and [25].

The main idea of knowledge graph embedding-based query answering is to predict the embedding of the answer variable by utilizing the embeddings of the entities appearing in the query. Recent approaches include GQE [15], CGA [12], Query2box [11], CQD [26] and Var2Vec [8]. GQE was one of the first such models that could handle conjunctive graph queries. Mai et al. [12] developed CGA which extends GQE by dealing with the variability of contributions from different query paths. Query2box [11] models the queries by using box instead of point embeddings in a vector space. In this way, it manages to support with higher accuracy (compared to GQE) logical queries that may include the existential quantifier and both the conjunction and disjunction ($\vee$) operators. CQD supports the same fragment of FOL queries, and exploits trained neural link predictors and fuzzy logic operators limiting in this way the need for a large and complex training dataset. However, as it is discussed by Wang et al. [8] it has low inference efficiency. Var2Vec [8] supports in a scalable manner also queries with negations.

Although the sparsity of geospatial knowledge graphs calls for solutions utilizing knowledge graph embedding-based techniques, to the best of our knowledge, only Wang et al. [27] have utilized embedding techniques for link prediction and Mai et al. [10] for conjunctive query answering. SE-KGE [10] encodes the absolute positions of the geospatial entities. Until SE-KGE, geospatial information was encoded into the KG embedding space in a very limited fashion, mainly by leveraging only the geospatial distances between geospatial entities (e.g., [28, 29]).

SE-KGE utilizes the GQE and CGA models to encode the class information of the geospatial entities (e.g., whether a geospatial entity is a country or a city). In addition, it encodes the geospatial information of an entity by using the location encoder Space2Vec [30]. Space2Vec first encodes a location as a multi-scale periodic representation by using sinusoidal functions with different frequencies and then feeds the resulting embedding into a N-layer feed-forward neural network. In this way, it preserves global position information of the geospatial entities of the knowledge graph as well as relative distance and direction.

4. The SQABo Model

In this section we present the architecture of the SQABo model. SQABo is composed of: (i) the geospatial encoder, which encodes the entities appearing an input query $\mathcal{Q}$ taking into consideration the classes in which they belong and their geometry; (ii) the geometric projection operator that creates, moves and enlarges the query box embeddings based on the components of $\mathcal{Q}$; and (iii) the intersection operator which intersects the final generated boxes to return the final box containing the set of points corresponding to the of answer entities of $\mathcal{Q}$.

4.1. Geospatial Entity Encoder

For the encoding of the geospatial entities we follow the methodology proposed by Mai et al. [10], which, for better understanding of the paper, it is described in brief next.

The *geospatial entity encoder* $Enc()$ consists of an *entity class encoder* $Enc^{(c)}()$ and an *entity space encoder* $Enc^{(x)}()$. The entity class encoder consists of class-specific embedding matrices which are learned during training. This part helps the model to learn class information about each geospatial entity (e.g., that the entity is a country or a city).

The entity space encoder utilizes the model Space2Vec [30] to enrich the final representation with the geospatial information of the entity. For entities that are locations and have point representations, the input to the module is the entity's co-ordinates. If an entity is a region, the input is a point selected uniformly at random to be inside the bounding box of the entity. The intuition behind this is that during training the encoder will learn a uniform distribution over the entity's bounding box. After constructing the input, the initial space representation passes through a feed-forward neural network to get the space embedding. In the end, the class embedding and the space embedding are concatenated resulting in the entity embedding which will have both class and geospatial information embedded.

4.2. Geometric Projection Operator

The geometric projection operator goes a step beyond simple edge prediction and incorporates the idea of box embeddings proposed by Ren et al. [11], but we extend it by encoding also the geospatial information. The intuition behind the box embedding approach is that, instead of encoding the queries as points in a latent space, encode them as hyper-rectangles. The points inside the generated hyper-rectangle correspond to the answer entities of the query. The hyper-rectangle is defined by a center embedding (of the box of the query) and an offset embedding (from the center of the box).

To incorporate the geospatial information present in a geospatial knowledge graph, the geometric projection operator encodes each relation r to a relation embedding $\mathbf{r} = (Cen(\mathbf{r}), Off(\mathbf{r})) \in \mathbb{R}^{2d}$. $Cen(\mathbf{r})$ represents the *relation box center embedding* and it is calculated by using two trainable matrices: the *relation feature matrix* $\mathbf{r}^{(c)}$ and the *geospatial embedding matrix* $\mathbf{r}^{(x)}$ which are applied on the entity class embedding and the entity space embeddings, respectively. The relation box center embedding is resulted from the concatenation of these two matrices: $Cen(\mathbf{r})$= $([\mathbf{r}^{(c)}; \mathbf{r}^{(x)}])$. $Off(\mathbf{r})$, with $Off(\mathbf{r}) \succeq 0$, is the trainable *relation box offset embedding matrix*, which, when trained, represents the correct size of the box (i.e., the distance of its endpoints from its center).

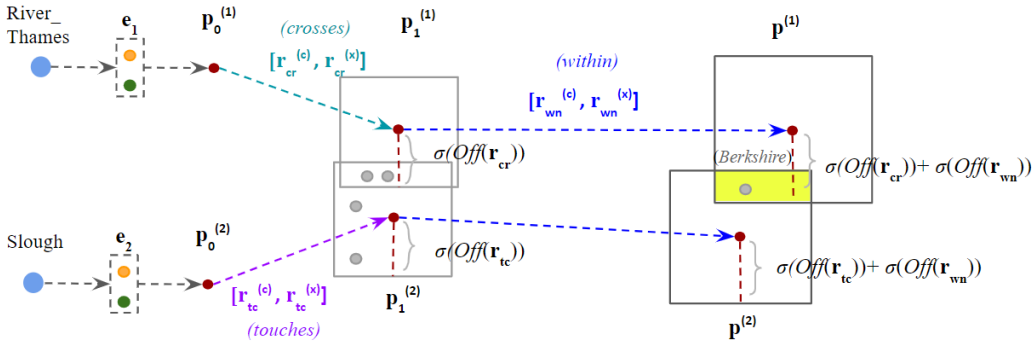

Figure 2: The embedding process of the example query: "Which place contains an area that is crossed by Thames and is near Slough ?"

.

Let $\mathcal{G} = \langle \mathcal{E} \cup \mathcal{L}, \mathcal{R} \rangle$ be a geospatial knowledge graph, $\mathcal{Q}(x)$ a query over $\mathcal{G}$, $\mathcal{G}_\mathcal{Q}$ the DAG of $\mathcal{Q}(x)$, $\mathcal{P}$ a path of $\mathcal{G}_\mathcal{Q}$ from a leaf entity $e \in \mathcal{E}$ to the root x, and $\mathcal{Q}_\mathcal{P}(x) = r_1(e, x_1) \wedge r_2(x_1, x_2) \wedge \ldots \wedge r_{n+1}(x_n, x)$ the subformula of $\mathcal{Q}(x)$ corresponding to $\mathcal{P}$. Then, supposing that we operate on $\mathbb{R}^d$, the geometric projection operator calculates the box embedding $\mathbf{p} = (\mathit{Cen}(\mathbf{p}), \mathit{Off}(\mathbf{p})) \in \mathbb{R}^{2d}$ of $\mathcal{P}$ as follows.

First, in line with [11], each relation $r \in \mathcal{R}$ is associated with a relation embedding $\mathbf{r} = (\mathit{Cen}(\mathbf{r}), \mathit{Off}(\mathbf{r})) \in \mathbb{R}^{2d}$ with $\mathit{Off}(\mathbf{r}) \succeq \mathbf{0}$ as the contribution of the relation to the existing box may enlarge it or only move it in the vector space.

Then, starting from the leaf entity e the box embedding $\mathbf{p}_0 = (\mathbf{e}, \mathbf{0})$, with $\mathit{Enc}(e) = \mathbf{e}, \mathbf{e} \in \mathbb{R}^d$, will be calculated and the node e in $\mathcal{P}$ will be marked as visited. After this, the following process will be repeated until all nodes in $\mathcal{P}$ are marked as visited:

- Iterate through each unvisited node x_j of $\mathcal{P}$ for which there is an i $(1 \leq i \leq n)$ such that $r_i(t, x_j)$ appears in $\mathcal{Q}(x)$ and t is a visited node.
- Compute box $\mathbf{p}_i = (\mathit{Cen}(\mathbf{p}_{i-1}) + [\mathbf{r}_i{}^{(c)}; \mathbf{r}_i{}^{(x)}], \mathit{Off}(\mathbf{p}_{i-1}) + \sigma(\mathit{Off}(\mathbf{r}_i)))$, where σ is the sigmoid activation function.
- Mark x_j as visited.

Finally, $\mathbf{p} = \mathbf{p}_{n+1}$.

If the query contains multiple entities (i.e., its DAG contains multiple leaves), then the geometric projection operator generates multiple boxes.

Example 4.1. Consider the following query $\mathcal{Q}(x)$ presented in Figure 1:

$$(\exists x_1)(\mathit{crosses}(\mathit{River\_Thames}, x_1) \wedge \mathit{touches}(\mathit{Slough}, x_1) \wedge \mathit{within}(x_1, x))$$

The embedding process of $\mathcal{Q}(x)$ is presented in Figure 2. The query has two leaf-to-root paths: $\mathcal{P}_1 = \{\mathit{River\_Thames}, x_1, x\}$, $\mathcal{P}_2 = \{\mathit{Slough}, x_1, x\}$. For $\mathcal{P}_1$, initially the embedding $\mathbf{e}_1$ of *River_Thames* will be calculated and, then, the zero-offset "box" $\mathbf{p}_0^{(1)}$ with center $\mathbf{e}_1$ $(\mathbf{p}_0^{(1)} = (\mathbf{e}_1, \mathbf{0}))$ will be generated. Then, the geometric projection operator visits the node x_1,

encodes the relation *crosses* ($[\mathbf{r}_{cr}{}^{(c)}; \mathbf{r}_{cr}{}^{(x)}]$) and randomly initializes $\mathit{Off}(\mathbf{r}_{cr})$. In this way, it creates the box $\mathbf{p}_1^{(1)}$ by combining the embeddings with the previous zero-size initial box:

$$\mathbf{p}_1^{(1)} = (\mathbf{e}_1 + [\mathbf{r}_{cr}{}^{(c)}; \mathbf{r}_{cr}{}^{(x)}], \sigma(\mathit{Off}(\mathbf{r}_{cr})))$$

In the same way, the final box $\mathbf{p}^{(1)} = \mathbf{p}_2^{(1)}$ is generated by encoding the relation *within* ($\mathbf{r}_{wn}$) and combining it with the box $\mathbf{p}_1^{(1)}$. The same process is followed to create $\mathbf{p}^{(2)}$ for path $\mathcal{P}_2$.

Next, we describe the box intersection operator, which intersects the generated boxes from each path of the query to result in a single embedding box that will represent the set of answers to the input query.

4.3. Box Intersection Operator

The box intersection operator produces a single final answer box embedding by calculating a final box center and the final box offset from the boxes $\mathbf{p}^{(1)}$, $\mathbf{p}^{(2)}$, ..., $\mathbf{p}^{(m)}$ generated by the geometric projection operator.

4.3.1. Intersection of Box Centers

Instead of employing a simple attentions mechanism, as in Query2Box, `SQABo` employs the graph self-attention mechanism introduced by Mai et al. [12], which is implemented by using a multi-head attention layer and a feed-forward neural network layer having normalization layers in between. In the end, the attention-weighted box center embedding is computed as the weighted average of different input box center embeddings, while the weights are automatically learned by the multi-head attention mechanism.

Formally, let $\mathcal{G}$ be a geospatial KG, $\mathcal{Q}(x)$ be a query over $\mathcal{G}$ and $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(m)}$ generated by the geometric projection operator, then the center of the intersected box is calculated as:

$$\mathit{Cen}(\mathcal{Q}) = \mathit{CGA}(\mathit{Cen}(\mathbf{p}^{(1)}), \ldots, \mathit{Cen}(\mathbf{p}^{(m)})) =$$
$$\mathit{LayerNorm}_2(\mathbf{W}_c \mathit{LayerNorm}_1(\mathbf{e}_{\mathit{attn}} + \mathbf{e}_{\mathit{init}}) + \mathbf{B}_c + \mathit{LayerNorm}_1(\mathbf{e}_{\mathit{attn}} + \mathbf{e}_{\mathit{init}}))$$

where $\mathit{LayerNorm}_1()$, $\mathit{LayerNorm}_2()$ are normalization layers, $\mathbf{W}_c \in \mathbb{R}^{d \times d}$ is a trainable entity class specific weight matrix (since all box centers are forced to have the same entity class C) and $\mathbf{B}_c \in \mathbb{R}^d$ a bias vector in a feed-forward neural network. Also, $\mathbf{e}_{\mathit{init}}$ is a permutation invariant transformation of the initial box center embeddings. The attention weighted embedding $\mathbf{e}_{\mathit{attn}}$ is computed as the weighted average of different input embeddings, while the weights are automatically learned by the multi-head attention mechanism:

$$\mathbf{e}_{\mathit{attn}} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{m} a_{ik} \mathit{Cen}(\mathbf{p}^{(i)}))$$

where K is the number of attention heads, $\sigma()$ is the sigmoid activation function, m is the number of all answer box centers to be intersected, and a_{ik} the attention coefficient [12] for the center of each $\mathbf{p}^{(i)}$ box in the k^{th} attention head.

Table 1
Statistics of the dataset used in SQABo.

	YAGO2geo	DBGeo$_G$
Triples	17,353,031	176,671
Relations	9	227
Entities	772,143	25,980
Training Queries	1,000,000	1,000,000
Validation Queries	1,000	1,000
Testing Queries	10,000	10,000
Places	UK / Ireland / Greece	United States (DBpedia)

4.3.2. Intersection of Box Offsets

For the intersection of box offsets we use the method proposed by Ren et al. [11], which generates a smaller box that lies inside the intersected boxes. For this purpose, the permutation-invariant deep architecture Deepsets [31] is utilized.

The intersection offset is given by:

$$Off(\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(m)}) = Min(Off(\mathbf{p}^{(1)}), \ldots, Off(\mathbf{p}^{(m)})) \cdot \sigma(DeepSets(\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(m)}))$$

5. Experiments

We have implemented SQABo and made it available publicly as open source.[2] We also evaluated the performance of SQABo against SE-KGE [10], which is, to the best of our knowledge, the only embedding-based query answering model over geospatial knowledge graphs proposed so far. In this section, we describe the results of our evaluation over the geospatial knowledge graphs DBGeo[3], which was used for the evaluation of SE-KGE, and YAGO2geo[4]. The statistics of DBGeo and YAGO2geo and the respective generated QA datasets are presented in Table 1.

Knowledge Graphs. To evaluate DBGeo with SQABo, we extracted its geospatial fragment (i.e., the fragment of DBGeo that contains only geospatial entities) named here DBGeo$_G$. To follow the entity encoder architecture of SE-KGE [10], we transformed the polygons represented YAGO2geo into bounding boxes by initially calculating the minimum and maximum vertices of each polygon and finally computing from these vertices the height, width, rotation angle and center of each bounding box.

Query Dataset. In line with [11, 10], we consider the seven query patterns (DAG structures) shown in Figure 3. Query pattern 1-chain represents the most simple query with an entity, a relation and an answer variable, 2-chain represents a query with a single existential variable and a single answer variable, the 2-inter a query with two entities and a single answer variable. The remaining patterns represent increasingly complex query structures.

[2] https://github.com/markos-iliakis/GeospatialKGEmbeddings
[3] https://github.com/gengchenmai/se-kge
[4] https://yago2geo.di.uoa.gr/

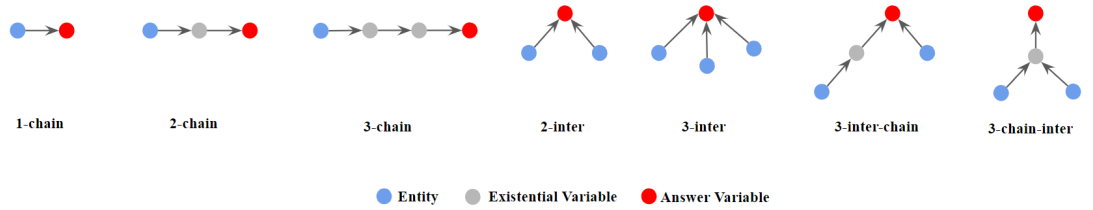

Figure 3: Query patterns considered in the experiments. In the naming of the query structures, *chain* stands for Projection and *inter* stands for Intersection.

As in [11, 10], for the generation of the training and validation queries, we selected 10% of the edges in a uniformly random manner and removed them from the graph and, then, we performed sampling on this down-sampled training graph, taking $1,000,000$ samples of the seven different DAG structures. To make the test queries, we sampled them from the original graph, but we ensured that at least the answer nodes were not part of the training graph and, therefore, the model should not have used them in the training phase.

Model Training. The training of the model is supervised as we sampled the query-answer pairs from the graph. In the training phase, we sample in total $1,000,000$ conjunctive query-answer pairs, equally distributed among the DAG structures, and k negative answers for each query. We utilized the loss function used for the `Query2Box` model:

$$L = -\log \sigma(\gamma - dist_{box}(\mathbf{v}; \mathbf{q})) - \sum_{i=1}^{k} \frac{1}{k} \log \sigma(dist_{box}(\mathbf{v}'; \mathbf{q}) - \gamma)$$

where γ represents a fixed scalar margin, and $dist_{box}()$ [11] the distance between the entity vector $\mathbf{v}$ representing the answer v (positive entity) to the query box embedding $\mathbf{q}$. Also, $\mathbf{v}'$ represents the entity vector of the i-th negative entity (non-answer to $\mathcal{Q}$) v', and k is the number of negative entities.

The number of iterations was $30,000$ and the embedding size of both the entity encoder feature and spatial embedding was 128 as well as the size of the relation embeddings. The hidden size of the feed-forward NN was 512, the dropout 0.5 with skip connections and learning rate 0.01. `Space2Vec` was used with a minimum radius of 50, maximum of 5,400,000, and frequency of 16. The graph attention mechanism had 2 heads.

We implemented all models in PyTorch and trained/evaluated each model on a Ubuntu machine with 1 GeForce GTX 1080 Nvidia GPU core, which has 10GB memory.

Evaluation Metrics. In line with Mai et al. [10], to measure the performance of `SQABo`, i.e., how representative are the final embeddings of the target nodes (answers), we use Average Percentile Rank (*APR*). We calculate *APR* for each query by getting the average percentile rank of the correct answer among all negative answers based on the prediction of the model:

$$APR = \frac{CF - (0.5 * F)}{N} * 100$$

Table 2
Results (APR score) of SE−KGE model versus SQABo model evaluated on DBGeo and YAGO2geo

KG	Model	1-chain	2-chain	3-chain	2-inter	3-inter	3-inter -chain	3-chain -inter	Macro-Average
DBGeo	SE−KGE	89.74	79.28	70.82	98.5	99.45	90.37	98.08	88.6
DBGeo$_{\mathcal{G}}$	SE−KGE	81.81	64.95	55.38	**99.26**	99.69	87.12	92.75	82.9
	SQABo	**82.21**	**71.57**	**68.84**	98.11	**99.96**	**96.07**	**95.58**	**87.4**
YAGO2geo	SE−KGE	84.52	88.09	85.97	85.83	86.74	91.5	92.99	87.9
	SQABo	**90.09**	**91.14**	**87.72**	**93.9**	**95.46**	**98.18**	**98.3**	**93.5**
	SQABo-noGA	86.02	89.10	87.36	88.01	86.74	92.6	95.2	89.29

where *CF* (cumulative frequency) is the count of all scores less than or equal to the score of interest and F is the frequency for the score of interest.

Mai et al. utilize, also, the *AUC* (Area Under Roc Curve) but since *APR* uses all negative samples for each query, as opposed to the *AUC*, which uses only one negative sample per query, we consider *APR* a more robust evaluation metric, and hence we omit *AUC*.

Main Results. The evaluation results of SQABo against SE−KGE are presented in Table 2. We compare the two models with two experiments per model (one for each KG) plus one for the original SE−KGE with all of its data as a reference point (illustrated in the first row of the table). For the case of DBGeo$_{\mathcal{G}}$, for every single query structure, except for 2-inter (where SE−KGE outperformed SQABo only by 1.15%), SQABo demonstrates better APR, resulting in an APR score difference of 4.5%, when macro averaged. The results for YAGO2geo were even better, with SQABo being better in every query structure and having a macro averaged APR score difference of 5.6%. It is worth noting that, for 3-chain queries over DBGeo$_{\mathcal{G}}$, SQABo outperformed SE−KGE by 13.46%. While the maximum difference between the two models for YAGO2geo KG was in 3-inter queries, by 8.72%.

To analyse the influence of the contextual graph attention mechanism used for the intersection of the center of the boxes, we replaced it with simple attention, which is used by Ren et al. [11]. As it is shown in Table 2, the resulting model, SQABo−noGA still performed better than SE−KGE with an average APR of 89.29%, but worse than the model with contextual graph attention showing the importance of its use.

6. Conclusions and Future Work

We presented and evaluated the novel embedding-based geospatial query answering model SQABo. SQABo encodes the geospatial and non-geospatial features of the entities and relations appearing in any incoming conjunctive graph query. Then, these encodings are gradually projected into boxes. The answer to an input conjunctive graph query is computed by intersecting these boxes using graph attention [12] and returning the entities inside the boxes. Experimental results on two geospatial KGs demonstrate that SQABo outperforms existing related work.

In future work, we plan to increase the accuracy of our results by employing geospatial encoding techniques that appeared very recently in the literature [32, 33]. We, also, plan to extend the expressivity of the queries that SQABo can support to cover disjunction [11] and negation [8]. Finally, we plan to extend this work to support questions expressed in natural

language by translating them to formal queries, utilizing our previous work [34].

Acknowledgments

This work was supported by the European Union's Horizon 2020 R&I programme under the Marie Skłodowska-Curie GA (No 101032307), by the First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant (Pr. No: HFRI-FM17-2351), by ESA project DA4DTE (contract no. 4000139212/22/I-EF).

References

[1] N. Karalis, G. M. Mandilaras, M. Koubarakis, Extending the YAGO2 knowledge graph with precise geospatial knowledge, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II, volume 11779 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 181–197. URL: https://doi.org/10.1007/978-3-030-30796-7_12. doi:10.1007/978-3-030-30796-7_12.

[2] A. Dsouza, N. Tempelmeier, R. Yu, S. Gottschalk, E. Demidova, WorldKG: A world-scale geographic knowledge graph, in: CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021, ACM, 2021, pp. 4475–4484.

[3] K. Janowicz, P. Hitzler, W. Li, D. Rehberger, M. Schildhauer, R. Zhu, C. Shimizu, C. K. Fisher, L. Cai, G. Mai, J. Zalewski, L. Zhou, S. Stephen, S. G. Estrecha, B. D. Mecum, A. Lopez-Carr, A. Schroeder, D. Smith, D. J. Wright, S. Wang, Y. Tian, Z. Liu, M. Shi, A. D'Onofrio, Z. Gu, K. Currier, Know, know where, knowwheregraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence, AI Mag. 43 (2022) 30–39. URL: https://doi.org/10.1609/aimag.v43i1.19120. doi:10.1609/aimag.v43i1.19120.

[4] Matthew Perry, John Herring, OGC GeoSPARQL - A Geographic Query Language for RDF Data, OGC Implementation Standard OGC 11-052r4, Open Geospatial Consortium, 2012. URL: http://www.opengis.net/doc/IS/geosparql/1.0.

[5] K. Kyzirakos, M. Karpathiotakis, M. Koubarakis, Strabon: A semantic geospatial DBMS, in: P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I, volume 7649 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 295–311. URL: https://doi.org/10.1007/978-3-642-35176-1_19. doi:10.1007/978-3-642-35176-1_19.

[6] D. Bilidas, T. Ioannidis, N. Mamoulis, M. Koubarakis, Strabo 2: Distributed management of massive geospatial rdf datasets, Springer-Verlag, Berlin, Heidelberg, 2022, p. 411–427. URL: https://doi.org/10.1007/978-3-031-19433-7_24. doi:10.1007/978-3-031-19433-7_24.

[7] B. Regalia, K. Janowicz, G. McKenzie, Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data, Transactions in GIS 23 (2019) 601–619. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12548. doi:https://doi.org/10.1111/tgis.12548. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/tgis.12548.
[8] D. Wang, Y. Chen, B. Cuenca Grau, Efficient embeddings of logical variables for query answering over incomplete knowledge graphs (2022).
[9] F. Li, M. Chen, R. Dong, Multi-hop question answering with knowledge graph embedding in a similar semantic space, in: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 01–07.
[10] G. Mai, K. Janowicz, L. Cai, R. Zhu, B. Regalia, B. Yan, M. Shi, N. Lao, Se-kge: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting, Transactions in GIS 24 (2020) 623–655.
[11] H. Ren, W. Hu, J. Leskovec, Query2box: Reasoning over knowledge graphs in vector space using box embeddings, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJgr4kSFDS.
[12] G. Mai, K. Janowicz, B. Yan, R. Zhu, L. Cai, N. Lao, Contextual graph attention for answering logical queries over incomplete knowledge graphs, in: Proceedings of the 10th International Conference on Knowledge Capture, 2019, pp. 171–178.
[13] N. Karalis, G. Mandilaras, M. Koubarakis, Extending the yago2 knowledge graph with precise geospatial knowledge, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 181–197.
[14] P. A. Longley, M. F. Goodchild, D. J. Maguire, D. W. Rhind, Geographic Information Science and Systems, 4th edition, John Wiley and Sons, 2015.
[15] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, J. Leskovec, Embedding logical queries on knowledge graphs, Advances in neural information processing systems 31 (2018).
[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).
[17] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.
[18] M. Nickel, V. Tresp, H.-P. Kriegel, Factorizing yago: scalable machine learning for linked data, in: Proceedings of the 21st international conference on World Wide Web, 2012, pp. 271–280.
[19] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: M. F. Balcan, K. Q. Weinberger (Eds.), Proceedings of The 33rd International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, 2016, pp. 2071–2080. URL: https://proceedings.mlr.press/v48/trouillon16.html.
[20] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France,

April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.
[21] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European semantic web conference, Springer, 2018, pp. 593–607.
[22] L. Cai, B. Yan, G. Mai, K. Janowicz, R. Zhu, Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction, in: Proceedings of the 10th International Conference on Knowledge Capture, 2019, pp. 131–138.
[23] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2724–2743. doi:10.1109/TKDE.2017.2754499.
[24] S. Choudhary, T. Luthra, A. Mittal, R. Singh, A survey of knowledge graph embedding and their applications, CoRR abs/2107.07842 (2021). URL: https://arxiv.org/abs/2107.07842. arXiv:2107.07842.
[25] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: A comparative analysis, ACM Trans. Knowl. Discov. Data 15 (2021). URL: https://doi.org/10.1145/3424672. doi:10.1145/3424672.
[26] E. Arakelyan, D. Daza, P. Minervini, M. Cochez, Complex query answering with neural link predictors, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=Mos9F9kDwkz.
[27] Y. Wang, H. Zhang, H. Xie, Geography-enhanced link prediction framework for knowledge graph completion, in: Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding: 4th China Conference, CCKS 2019, Hangzhou, China, August 24–27, 2019, Revised Selected Papers 4, Springer, 2019, pp. 198–210.
[28] P. Qiu, J. Gao, L. Yu, F. Lu, Knowledge embedding with geospatial distance restriction for geographic knowledge graph completion, ISPRS International Journal of Geo-Information 8 (2019). URL: https://www.mdpi.com/2220-9964/8/6/254. doi:10.3390/ijgi8060254.
[29] G. Mai, B. Yan, K. Janowicz, R. Zhu, Relaxing unanswerable geographic questions using a spatially explicit knowledge graph embedding model, in: P. Kyriakidis, D. Hadjimitsis, D. Skarlatos, A. Mansourian (Eds.), Geospatial Technologies for Local and Regional Development, Springer International Publishing, Cham, 2020, pp. 21–39.
[30] G. Mai, K. Janowicz, B. Yan, R. Zhu, L. Cai, N. Lao, Multi-scale representation learning for spatial feature distributions using grid cells, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020. URL: https://openreview.net/forum?id=rJljdh4KDH.
[31] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, A. Smola, Deep sets, 2017. URL: https://arxiv.org/abs/1703.06114. doi:10.48550/ARXIV.1703.06114.
[32] G. Mai, C. Jiang, W. Sun, R. Zhu, Y. Xuan, L. Cai, K. Janowicz, S. Ermon, N. Lao, Towards general-purpose representation learning of polygonal geometries, 2022. URL: https://arxiv.org/abs/2209.15458. doi:10.48550/ARXIV.2209.15458.
[33] G. Mai, Y. Xuan, W. Zuo, K. Janowicz, N. Lao, Sphere2vec: Multi-scale representation learning over a spherical surface for geospatial predictions, 2022. URL: https://arxiv.org/abs/2201.10489. doi:10.48550/ARXIV.2201.10489.
[34] D. Punjani, E. Tsalapati, Question Answering Engines for Geospatial Knowledge Graphs, 1

ed., Association for Computing Machinery, New York, NY, USA, 2023, p. 257–282. URL: https://doi.org/10.1145/3581906.3581922.