SE-KGE: A Location-Aware Knowledge Graph Embedding Model for Geographic Question Answering and Spatial Semantic Lifting

Gengchen Mai[1], Krzysztof Janowicz[1], Ling Cai[1], Rui Zhu[1], Blake Regalia[1], Bo Yan[2], Meilin Shi[1], and Ni Lao[3]

[1]STKO Lab, University of California, Santa Barbara, CA, USA, 93106
{*gengchen_mai, janowicz, lingcai, ruizhu, regalia, meilinshi*}*@ucsb.edu*
[2]LinkedIn Corporation, Mountain View, CA, USA, 94043
boyan1@linkedin.com
[2]SayMosaic Inc., Palo Alto, CA, USA, 94303
ni.lao@mosaix.ai

Abstract

Learning knowledge graph (KG) embeddings is an emerging technique for a variety of downstream tasks such as summarization, link prediction, information retrieval, and question answering. However, most existing KG embedding models neglect space and, therefore, do not perform well when applied to (geo)spatial data and tasks. For those models that consider space, most of them primarily rely on some notions of distance. These models suffer from higher computational complexity during training while still losing information beyond the relative distance between entities. In this work, we propose a location-aware KG embedding model called *SE-KGE*. It directly encodes spatial information such as point coordinates or bounding boxes of geographic entities into the KG embedding space. The resulting model is capable of handling different types of spatial reasoning. We also construct a geographic knowledge graph as well as a set of geographic query-answer pairs called *DBGeo* to evaluate the performance of *SE-KGE* in comparison to multiple baselines. Evaluation results show that *SE-KGE* outperforms these baselines on the *DBGeo* dataset for geographic logic query answering task. This demonstrates the effectiveness of our spatially-explicit model and the importance of considering the scale of different geographic entities. Finally, we introduce a novel downstream task called *spatial semantic lifting* which links an arbitrary location in the study area to entities in the KG via some relations. Evaluation on *DBGeo* shows that our model outperforms the baseline by a substantial margin.

Keywords: Knowledge Graph Embedding, Location Encoding, Spatially Explicit Model, Geographic Question Answering, Spatial Semantic Lifting

1 Introduction and Motivation

The term *Knowledge Graph* typically refers to a labeled and directed multi-graph of statements (called triples) about the world. These triples often originate from heterogeneous sources across domains. According to Nickel et al. (2015), most of the widely used knowledge graphs are constructed in a curated (e.g., WordNet), collaborative (e.g., Wikidata, Freebase), or auto semi-structured (e.g., YAGO (Hoffart et al., 2013), DBpedia, Freebase) fashion rather than an automated unstructured approach (e.g., Knowledge Vault (Dong et al., 2014)). Despite containing billions of statements, these knowledge graphs suffer from *incompleteness and sparsity* (Lao et al., 2011; Dong et al., 2014; Mai et al., 2019a). To address these problems, many relational machine learning models (Nickel et al., 2015) have been developed for knowledge graph completion tasks including several embedding-based techniques such as RESCAL (Nickel et al., 2012), TransE (Bordes et al., 2013), TransH (Wang et al., 2014), HOLE (Nickel et al., 2016), R-GCN (Schlichtkrull et al., 2018), and TransGCN (Cai et al., 2019). The key idea of the embedding-based technique (Bordes et al., 2013; Wang et al., 2014; Nickel et al., 2016; Cai et al., 2019; Wang et al., 2017) is to project entities and relations in a knowledge graph onto a continuous vector space such that entities and relations can be quantitatively represented as vectors/embeddings.

The aforementioned incompleteness and sparsity problems also affect the performance of downstream tasks such as question answering (Wang et al., 2018) since missing triples or links result in certain questions becoming unanswerable (Rajpurkar et al., 2018). Consequently, researchers have recently focused on relaxing these unanswerable queries or predicting the most probable answers based on knowledge graph embedding models (Wang et al., 2018; Hamilton et al., 2018; Mai et al., 2019b).

Most research on knowledge graph embeddings has neglected spatial aspects such as the location of geographic entities despite the important role such entities play within knowledge graphs (Janowicz et al., 2012). In fact, most of the current knowledge graph embedding models (e.g. TransE, TransH, TransGCN, R-GCN, and HOLE) ignore triples that contain *datatype properties*, and, hence, literals for dates, texts, numbers, geometries, and so forth. Put differently, properties such as `dbo:elevation`, `dbo:populationTotal`, and `dbo:areaWater` to name but a few are not considered during the training phase. Instead, these models strictly focus on triples with *object type properties*, leading to substantial information loss in practice. A few models do consider a limited set of datatypes. LiteralE (Kristiadi et al., 2019) is one example, which encodes numeric and date information into its embedding space, while MKBE (Pezeshkpour et al., 2018) encodes images and unstructured texts. Therefore, in this work, we propose a novel technique which directly encodes spatial footprints, namely point coordinates and bounding boxes, thereby making them available while learning knowledge graph embeddings.

Geographic information forms the basis for many KG downstream tasks such as geographic knowledge graph completion (Qiu et al., 2019), geographic ontology alignment (Zhu et al., 2016), geographic entity alignment (Trisedya et al., 2019), geographic question answering (Mai et al., 2019b), and geographic knowledge graph summarization (Yan et al., 2019). In the following, we will focus on geographic logic query answering as an example and more concretely on conjunctive graph queries (CGQ) or logic queries (Hamilton et al., 2018). Due to the sparsity of information in knowledge graphs, many (geographic) queries are unanswerable *without spatial or non-spatial reasoning*. Knowledge graph embedding techniques have, therefore, been developed to handle unanswerable questions (Hamilton et al., 2018; Wang et al., 2018; Mai et al., 2019b,a) by inferring

new triples in the KG embedding space based on existing ones. However, since most KG embedding models cannot handle *datatype properties* thus cannot encode geographic information into the KG embedding space, they perform spatial reasoning tasks poorly in the KG embedding space, which in turn leads to a poor performance of handling unanswerable geographic questions.

```
SELECT ?State WHERE {
 ?RiverMouth dbo:state ?State.                    (a)
 ?River dbo:mouthPosition ?RiverMouth.               (b)
 ?River dbp:etymology dbr:Alexander_von_Humboldt.    (c)
}
```

Listing 1: Query q_A: An unanswerable SPARQL query over DBpedia which includes a partonomy relation

One example of unanswerable geographic questions that can be represented as a logic query is *which states contain the mouth of a river which is named after Alexander von Humboldt?* (Query q_A). The corresponding SPARQL query is shown in Listing 1. Running this query against the DBpedia SPARQL endpoint yields no results. In fact, two rivers are named after von Humboldt - `dbr:Humboldt_River` and `dbr:North_Fork_Humboldt_River` - and both have mouth positions as entities in DBpedia (`dbr:Humboldt_River__mouthPosition__1` and `dbr:North_Fork_Humboldt_River__sourcePosition__1`). However, the `dbo:state` (or `dbo:isPartOf`) relation between these river mouths and other geographic features such as states is missing. This makes Query q_A unanswerable (graph query pattern (a) in Listing 1). If we use the locations of the river mouths to perform a simple point-in-polygon test against the borders of all states in the US, we can deduce that `dbr:Nevada` contains both river mouths.

```
SELECT ?place WHERE {
 dbr:Yosemite_National_Park dbo:nearestCity ?place.
}
```

Listing 2: Query q_B: A SPARQL query over DBpedia which indicates a simple point-wise distance relation

Another example is the query in Listing 2, which asks for *the nearest city to Yosemite National Park* (Query q_B). If the triple `dbr:Yosemite_National_Park dbo:nearestCity dbo:Mariposa,_California` is missing from the current knowledge graph, Query q_B becomes unanswerable while it could simply be inferred by a distance-based query commonly used in GIS. Similar cases can include cardinal directions such as `dbp:north`. All these observations lead to the following research question: how could we enable spatial reasoning via *partonomic relations*, *point-wise metric relations*, and *directional relations* in the KG embedding-based systems?

One may argue that classical spatial reasoning can be used instead of direct location encoding to obtain answers to aforementioned questions. This is partially true for data and query endpoints that support GeoSPARQL and for datasets that are clean and complete. However, in some cases even GeoSPARQL-enabled query endpoints cannot accommodate spatial reasoning due to inherent

challenges of representing spatial data in knowledge graphs. These challenges stem from principles of conceptual vagueness and uncertainty (Regalia et al., 2019), and are further complicated by technical limitations. In this study we aim at enabling the model to perform *implicit spatial reasoning* in the hidden embedding space. Instead of performing classical spatial reasoning by explicitly carrying out spatial operations during query time, the spatial information (points or bounding boxes) of geographic entities (e.g., $Indianapolis$) are directly encoded into the entity embeddings which are jointly optimized with relation embeddings (e.g, $isPartOf$). The trained embeddings of geographic entities encode their spatial information while by embedding the spatial relations, we also hope to capture some of their implicit semantics for simple spatial reasoning tasks. At query time, a normal link prediction process can be used to answer geographic questions and no explicit spatial reasoning is needed. Find more detail of this example in Section 7.

Existing approaches are only able to incorporate spatial information into the KG embedding space in a very limited fashion, e.g., through their training procedures. Furthermore, they estimate entity similarities based on some form of distance measures among entities, and ignore their absolute positions or relative directions. For example, Trisedya et al. (2019) treated geographic coordinates as strings (a sequence of characters) and used a compositional function to encode these coordinate strings for geographic entities alignment. In order to incorporate distance relations between geographic entities, both Mai et al. (2019b) and Qiu et al. (2019) borrowed the translation assumption from TransE (Bordes et al., 2013). For each geographic triple $s = (h, r, t)$ in the KG, where h and t are geographic entities, the geospatial distance between h and t determines the frequency of resampling this triple such that triples containing two closer geographic entities are sampled more frequently, and thus these two geographic entities are closer in the embedding space. Similarly, Yan et al. (2019) used distance information to construct virtual spatial relations between geographic entities during the knowledge graph summarization process. This data conversion process (coordinates to pairwise distances) is unnecessarily expensive and causes information loss, e.g., absolute positions and relative directional information. In this work, we explore to directly encode entity locations into a high dimensional vector space, which preserves richer spatial information than distance measures. These location embeddings can be trained jointly with knowledge graph embedding.

Location encoders (Mac Aodha et al., 2019; Chu et al., 2019; Mai et al., 2020) refer to the neural network models which encode a pair of coordinates into a high dimensional embedding which can be used in downstream tasks such as geo-aware fine-grained image classification (Mac Aodha et al., 2019; Chu et al., 2019; Mai et al., 2020) and Point of Interest (POI) type classification (Mai et al., 2020). Mai et al. (2020) showed that multi-scale grid cell representation outperforms commonly used kernel based methods (e.g., RBF) as well as the single scale location encoding approaches. Given the success of location encoding in other machine learning tasks, the question is whether we can incorporate the location encoder architecture into a knowledge graph embedding model to make it spatially explicit (Mai et al., 2019b). One initial idea is directly using a location encoder as the entity encoder which encodes the spatial footprint (e.g., coordinates) of a geographic entity into a high dimensional vector. Such entity embeddings can be used in different decoder architectures for different tasks. However, several challenges remain to be solved for this initial approach.

First, point location encoding can handle point-wise metric relations such as distance (e.g., `dbo:nearestCity`) as well as directional relations (e.g., `dbp:north`, `dbp:south`) in knowledge graphs, but it is not easy to encode regions which are critical for relations

such as containment (e.g., dbo:isPartOf, dbo:location, dbo:city, dbo:state, and dbo:country). For example, in Query q_A, the location encoder can encode dbr:Yosemite_National_Park and dbo:Mariposa,_California as two high dimensional embeddings based on which distance relations can be computed since the location embeddings preserve the relative distance information between locations (Mai et al., 2020). However, point locations and location embeddings are insufficient to capture more complex relations between geographic entities such as containment as these require more complex spatial footprints (e.g., polygons). This indicates that we need to find a way to represent geographic entities as *regions* instead of points in the embedding space based on location encoders, especially for large scale geographic entities such as dbr:California, which is represented as a single pair of coordinates (a point) in many widely used KGs. We call this *scale effect* to emphasize the necessity of encoding the spatial extents of geographic entities instead of points, especially for large scale geographic entities.

The second challenge is how to seamlessly handle geographic and non-geographic entities together in the same entity encoder framework. Since location encoder is an essential *component* of the entity encoder, how should we deal with non-geographic entities that do not have spatial footprints? This is a non-trivial problem. For example, in order to weight triples using distance during KG embedding training, Qiu et al. (2019) constructed a geographic knowledge graph which only contains geographic entities. Mai et al. (2019b) partially solved the problem by using a lower bound l as the lowest triple weight to handle non-geographic triples. However, this mechanism cannot distinguish triples involving both geographic and non-geographic entities from triples that only contain non-geographic entities.

The third challenge is how to capture the spatial and other semantic aspects at the same time when designing spatially explicit KG embedding model based on location encoders. The embedding of a geographic entity is expected to capture both its spatial (e.g., spatial extent) and other semantic information (e.g., type information) since both of them are necessary to answer geographic questions. Take Query q_A in Listing 1 as an example. Intuitively, to answer this query, the spatial information is necessary to perform partonomical reasoning to select geographic entities which *contain* a given river mouth, while type information is required to filter the answers and get entities with type $state$. Therefore, we need both spatial and type information encoded in the entity embeddings to answer this question. The traditional KG embedding models fail to capture the spatial information which leads to a lower performance in geographic question answering.

Finally, thanks to the inductive learning nature of the location encoder, another interesting question is how to design a spatially-explicit KG embedding model so that it can be used to infer new relations between entities in a KG and any arbitrary location in the study area. We call this task **spatial semantic lifting** as an analogy to traditional *semantic lifting* which refers to the process of associating unstructured content to semantic knowledge resources (De Nicola et al., 2008). For example, given any location $\mathbf{x}_i$, we may want to ask *which radio station broadcasts at* $\mathbf{x}_i$ i.e., to infer dbo:broadcastArea. None of the existing KG embedding models can solve this task.

In this work, we develop a spatially-explicit knowledge graph embedding model, *SE-KGE*, which directly solves those challenges. **The contributions of our work are as follow**:

1. We develop a spatially-explicit knowledge graph embedding model (*SE-KGE*), which applies a location encoder to incorporate spatial information (coordinates and spatial extents) of geographic entities. To the best of our knowledge, this is the first KG embedding model

that can incorporate spatial information, especially spatial extents, of geographic entities into the model architecture.

2. *SE-KGE* is extended to an end-to-end geographic logic query answering model which predicts the most probable answers to unanswerable geographic logic queries over KG.

3. We apply *SE-KGE* on a novel task called *spatially semantic lifting*. Evaluations show that our model can substantially outperform the baseline by 9.86% on AUC and 9.59% on APR for the *DBGeo* dataset. Furthermore, our analysis shows that this model can achieve implicit spatial reasoning for different types of spatial relations.

The rest of this paper is structured as follow. We briefly summarize related work in Section 2. Then basic concepts are discussed in Section 3. In Section 4, we formalize the query answering and spatial semantic lifting task. Then, in Section 5, we give an overview of the logic query answering task before introducing our method. Section 6 describes the *SE-KGE* architecture. Experiments and evaluations are summarized in Section 7. Finally, we conclude our work in Section 8.

2 Related Work

In this section, we briefly review related work on knowledge graph embeddings, query answering, and location encoding.

2.1 Knowledge Graph Embedding

Learning knowledge graph embeddings (KGE) is an emerging topic in both the Semantic Web and machine learning fields. The idea is to represent entities and relations as vectors or matrices within an embedding spaces such that these distributed representations can be easily used in downstream tasks such as KG completion and question answering. Many KG embedding models have been proposed such as RESCAL (Nickel et al., 2012), TransE (Bordes et al., 2013), and TransH (Wang et al., 2014). Most of these approaches cannot handle triples with data type properties nor triples involving spatial footprints.

The only KG embedding methods considering distance decay between geographic entities are Qiu et al. (2019) and Mai et al. (2019b). Mai et al. (2019b) computed the weight of each geographic triple $s = (h, r, t)$ as $\max(\ln \frac{D}{dis(h,t)+\varepsilon}, l)$ where h and t are geographic entities, and D is the longest (simplified) earth surface distance. ε is a hyperparameter to avoid zero denominator and l is the lowest edge weight we allow for each triple. As for non-geographic triples, l is used as the triple weight. Then this knowledge graph is treated as an undirected, unlabeled, edge-weighted multigraph. An edge-weighted PageRank is applied on this multigraph. The PageRank score for each node/entity captures the the structure information of the original KG as well as the distance decay effect among geographic entities. These scores are used in turn as weights to sample the entity context from the 1-degree neighborhood of each entity which is used in the KG embedding training process. As for Qiu et al. (2019), the distance decay effect was deployed in a triple negative sampling process. Given a triple $s = (h, r, t)$ in the KG, each negative triple $s' = (h', r, t')$ of it was assigned a weight based on $w_{geo} = \frac{1}{1 + \left| log_{10} \frac{dis(h,t)+\theta}{dis(h',t')+\theta} \right|}$ where θ is a hyperparameter to avoid

a zero denominator. w_{geo} is used in the max-margin loss function for the embedding model training. Note that non-geographic triples are not considered in Qiu et al. (2019). We can see that, instead of directly encoding an entity's location, they rely on some form of distance measures as weights for triple resampling. This process is computationally expensive and does not preserve other spatial properties such as direction. In contrast, our work introduces a direct encoding approach to handle spatial information.

2.2 Query Answering

Compared to link prediction (Bordes et al., 2013), query answering (Wang et al., 2018; Hamilton et al., 2018; Mai et al., 2019a) focuses on a more complex problem since answering a query requires a system to consider multiple triple patterns together. Wang et al. (2018) designed an algorithm to answer a subset of SPARQL queries based on a pretrained KG embedding model. However, this is not an end-to-end model since the KG embedding training and query answering process are separated. Hamilton et al. (2018) proposed an end-to-end logic query answering model, GQE, which can answer conjunctive graph queries. CGA (Mai et al., 2019a) further improved GQE by using a self-attention based intersection operator. In our work, we will utilize GQE and CGA (Mai et al., 2019a) as the underlying logic query answering baseline. We provide an overview about logic query answering in Section 5.

2.3 Location Encoding

Generating representations of points/locations that can benefit representation learning is a long-standing problem in machine learning. There are many well-established methods such as the *Kernel trick* (Schölkopf, 2001) widely used in SVM classification and regression. However, these location representation methods use the positions of training examples as the centers of Gaussian kernels and thus need to memorize the training examples.

Kejriwal and Szekely (2017) proposed a graph embedding approach to representing GeoNames locations as high dimensional embeddings. They converted the locations in GeoNames into a weighted graph where locations are nodes and the weight of each edge is computed based on the distance between two locations. Then a Glove (Pennington et al., 2014) word embedding model is applied on this generated graph to obtain the embedding for each location. Despite its novelty, this model is a transductive learning based model which means if new locations are added, the weighted graph has to be regenerated and the whole model needs to be retrained. In other words, this embedding approach can not be easily generalized to unseen locations. This calls for inductive learning (Hamilton et al., 2017a) based models.

Recently, location encoding technique (Mac Aodha et al., 2019; Chu et al., 2019; Mai et al., 2020) has been proposed to directly encode a location (a pair of coordinates) $\mathbf{x}$ as a high-dimension vector which can be incorporated into multiple downstream tasks. As shown by Mai et al. (2020), the advantages of location encoding is that **1)** it can preserve absolute position information as well as relative distance and direction information between locations; **2)** it does not need to memorize the positions of training examples as all kernel based methods do (Scholkopf et al., 1997); **3)** In contrast to many transductive learning models, it is an inductive learning model (Battaglia et al., 2018) which can encode any location/point no matter it appears in the training dataset or not.

In theory, we can adopt any location encoder (Chu et al., 2019; Mac Aodha et al., 2019; Mai et al., 2020) to capture the spatial information of each geographic entity e_i in a knowledge graph $\mathcal{G}$. In this work, we utilize the $Space2Vec$ (Mai et al., 2020) location encoder, which is inspired by Nobel Prize-winning neuroscience research about grid cells (Abbott and Callaway, 2014) as well as the position encoding module of the Transformer model (Vaswani et al., 2017). $Space2Vec$ first encodes a location $\mathbf{x}$ as a multi-scale periodic representation $PE(\mathbf{x})$ by using sinusoidal functions with different frequencies and then feeds the resulting embedding into a N layer feed forward neural network $\mathbf{NN}()$.

$$LocEnc^{(x)}(\mathbf{x}) = \mathbf{NN}(PE(\mathbf{x})) \tag{1}$$

The advantages of such location encoder compared to previous work (Chu et al., 2019; Mac Aodha et al., 2019) are that **1)** it can be shown that location embeddings from $Space2Vec$ are able to preserve global position information as well as relative distance and direction, and that **2)** multi-scale representation learning approach outperforms traditional kernel-based methods (e.g., RBF) as well as single-scale location encoding approaches (Chu et al., 2019; Mac Aodha et al., 2019) for several machine learning tasks. In the following, we will use $LocEnc^{(x)}()$ to denote the $Space2Vec$ model.

3 Basic Concepts

Definition 1 (Geographic Knowledge Graph)**.** *A geographic knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed edge and node labeled multigraph where $\mathcal{V}$ is a set of entities/nodes and $\mathcal{E}$ is the set of directed edges. Any directed and labeled edge will be called a triple $s = (h, r, t)$ where the nodes become heads $h \in \mathcal{V}$ and tails $t \in \mathcal{V}$, and the role label $r \in \mathcal{R}$ will be called the relationship between them. The set of triples/statements contained by $\mathcal{G}$ is denoted as $\mathcal{T}$ and $\mathcal{R}$ denotes as the set of relations (predicates, edge labels) in $\mathcal{G}$. Each triple can also be represented as $r(h, t)$, or $r^{-1}(t, h)$ where r^{-1} indicates the inverse relation of r. $Domain(r)$ and $Range(r)$ indicate the domain and range of relation r.*

$\Gamma() : \mathcal{V} \rightarrow \mathcal{C}$ is a function which maps an entity $e \in \mathcal{V}$ to a unique type $c \in \mathcal{C}$, where $\mathcal{C}$ is the set of all entity types in $\mathcal{G}$.[1]

The geographic entity set $\mathcal{V}_{pt}$ is a subset of $\mathcal{V}$ ($\mathcal{V}_{pt} \subseteq \mathcal{V}$). $\mathcal{PT}(\cdot)$ is a mapping function that maps any geographic entity $e \in \mathcal{V}_{pt}$ to its geographic location (coordinates) $\mathcal{PT}(e) = \mathbf{x}$ where $\mathbf{x} \in \mathcal{A} \subseteq \mathbb{R}^2$. Here $\mathcal{A}$ denotes the bounding box containing all geographic entities in the studied knowledge graph $\mathcal{G}$. We call it study area.

$\mathcal{V}_{pn}$ is a subset of $\mathcal{V}_{pt}$ ($\mathcal{V}_{pn} \subseteq \mathcal{V}_{pt}$) which represents the set of large-scale geographic entities whose spatial extent cannot be ignored. In this work, we use a bounding box to represent a geographic entity's spatial footprint. $\mathcal{PN}(\cdot)$ is a mapping function defined on $\mathcal{V}_{pn}$ that maps a geographic entity $e \in \mathcal{V}_{pn}$ to its spatial extent $\mathcal{PN}(e)$ and $\mathcal{PN}(e) = [\mathbf{x}^{min}; \mathbf{x}^{max}] \in \mathbb{R}^4$. In the vector concatenation above, $\mathbf{x}^{min}$, $\mathbf{x}^{max} \in \mathcal{A} \subseteq \mathbb{R}^2$ indicate the southwest and northeast point of the entity's bounding box.

[1] Note that, in many knowledge graphs (e.g., DBpedia, Wikidata), an entity can belong to multiple types. We use this definition to be in line with many existing work (Hamilton et al., 2018; Mai et al., 2019a) so that we can compare our results. It is easy to relax this requirement which we will discuss in Section 6.1.

Note that in many existing knowledge graphs, a triple can include a datatype property (e.g., `dbo:abstract`) implying that the tail is a literal. In line with related work (Bordes et al., 2013; Nickel et al., 2016, 2015; Wang et al., 2017), we do not consider this kind of triples here in general. However, we do consider datatype properties about the spatial footprints of geographic entities implicitly by using $\mathcal{PT}(\cdot)$ *or* $\mathcal{PN}(\cdot)$.[2] While we do not model them directly as triples, we use the spatial footprints of geographic entities as input features for the entity encoder.

Definition 2 (Conjunctive Graph Query (CGQ))**.** *A query $q \in Q(\mathcal{G})$ that can be written as follows:*

$$
\begin{aligned}
& q = V_? . \exists V_1, V_2, .., V_m : b_1 \wedge b_2 \wedge ... \wedge b_n \\
\textit{where} \quad & b_i = r_i(e_k, V_l), V_l \in \{V_?, V_1, V_2, .., V_m\}, e_k \in \mathcal{V}, r \in \mathcal{R} \\
\textit{or} \quad & b_i = r_i(V_k, V_l), V_k, V_l \in \{V_?, V_1, V_2, .., V_m\}, k \neq l, r \in \mathcal{R}
\end{aligned}
$$

Conjunctive graph queries are also called logic queries. Here $Q(\mathcal{G})$ is a set of all conjunctive graph queries that can be asked over $\mathcal{G}$. $V_?$ denotes the target variable of query q (*target node*) which will be replaced with the answer entity a, while $V_1, V_2, .., V_m$ are existentially quantified bound variables (*bound nodes*). $\{e_k | e_k \ in \ q\}$ is a set of anchor nodes and b_i is a basic graph pattern in this CGQ. We define the dependency graph of q as the graph with basic graph pattern $\{b_1, ..., b_n\}$ formed between the anchor nodes $\{e_k | e_k \ in \ q\}$ and the variable nodes $V_?, V_1, V_2, .., V_m$ (Figure 1). Each conjunctive graph query can be written as a SPARQL query.[3]

Note that the dependency graph of q represents computations on the KG and is commonly assumed to be a *directed acyclic graph (DAG)* (Hamilton et al., 2018) where the entities (*anchor nodes*) e_k in q are the source nodes and the target variable $V_?$ is the unique sink node. This restriction makes the logic query answering task in line with the usual question answering set up (e.g., semantic parsing (Berant et al., 2013; Liang et al., 2017)).

Definition 3 (Geographic Conjunctive Graph Query (GCGQ))**.** *A conjunctive graph query $q \in Q(\mathcal{G})$ is said to be a geographic conjunctive graph query if the answer entity a corresponding to the target variable $V_?$ is a geographic entity, i.e., $a = \varphi(\mathcal{G}, q) \wedge a \in \mathcal{V}_{pt}$ where $\varphi(\mathcal{G}, q)$ indicates the answer when executing query q on $\mathcal{G}$. We denote all possible geographic CGQ on $\mathcal{G}$ as $Q_{geo}(\mathcal{G}) \subseteq Q(\mathcal{G})$.*

An example geographic conjunctive query q_C is shown in Figure 1 whose corresponding SPARQL query is shown in Listing 3. The corresponding natural language question is *[which city in Alameda County, California is the assembly place of Chevrolet Eagle and the nearest city to San Francisco Bay]*. This query is especially interesting since it includes a non-spatial relation (`dbo:assembly`), a point-wise metric spatial relation (`dbo:nearestCity`) and a partonomy relation (`dbo:isPartOf`). Note that executing each basic graph pattern in Query q_C over DBpedia will yield multiple answers. For example, b_1 will return all subdivisions of Alameda County, California. b_2 matches multiple assembly places of Chevrolet Eagle such as `dbr:Oakland,_California`, `dbr:Oakland_Assembly`, and `dbr:Flint,_Michigan`. Interestingly, `dbr:Oakland_Assembly` should be located in `dbr:Oakland,_California` while there are no relationship between them in DBpedia

[2] It is worth mentioning that most KG to date merely store point geometries even for features such as the United States.

[3] For the detail comparison between CQG and SPARQL 1.1 query, please refer to Section 2.1 of Mai et al. (2019a).

except for their spatial footprints which can be inferred that they are closed to each other. b_3 will return three entities[4] - `dbr:San_Francisco`, `dbr:San_Jose,_California` and `dbr:Oakland,_California`. Combining these three basic graph patterns will yield one answer `dbr:Oakland,_California`. In our knowledge graph, both triple s_1 (See Figure 1) and triple s_2 are missing which makes Query q_C an unanswerable geographic query.

```
SELECT ?place  WHERE{
 ?place dbo:isPartOf dbr:Alameda_County,_California.      (1)
 dbr:Chevrolet_Eagle dbo:assembly ?place.                 (2)
 dbr:San_Francisco_Bay dbo:nearestCity ?place.            (3)
}
```

Listing 3: Query q_C: A geographic conjunctive query which is rewritten as a SPARQL query over DBpedia including both non-spatial relations and different types of spatial relation.

4 Problem Statement

In this work, we focus on two geospatial tasks - *geographic logic query answering* and *spatial semantic lifting*.

Task 1 (Logic Query Answering)**.** *Given a geographic knowledge graph $\mathcal{G}$ and an unanswerable conjunctive graph query $q \in Q(\mathcal{G})$ (i.e., $\varphi(\mathcal{G}, q) = \varnothing$), a query embedding function $\mathbf{\Phi}_{\mathcal{G},\theta}(q) : Q(\mathcal{G}) \rightarrow \mathbb{R}^d$, which is parameterized by θ, is defined to map q to a vector representation of d dimension. The most probable answer a' to q is the entity nearest to $\mathbf{q} = \mathbf{\Phi}_{\mathcal{G},\theta}(q)$ in the embedding space:*

$$a' = \arg\max_{e_i \in \mathcal{V}} \mathbf{\Omega}(\mathbf{\Phi}_{\mathcal{G},\theta}(q), Enc(e_i)) = \arg\max_{e_i \in \mathcal{V}} \mathbf{\Omega}(\mathbf{q}, \mathbf{e}_i) \tag{2}$$

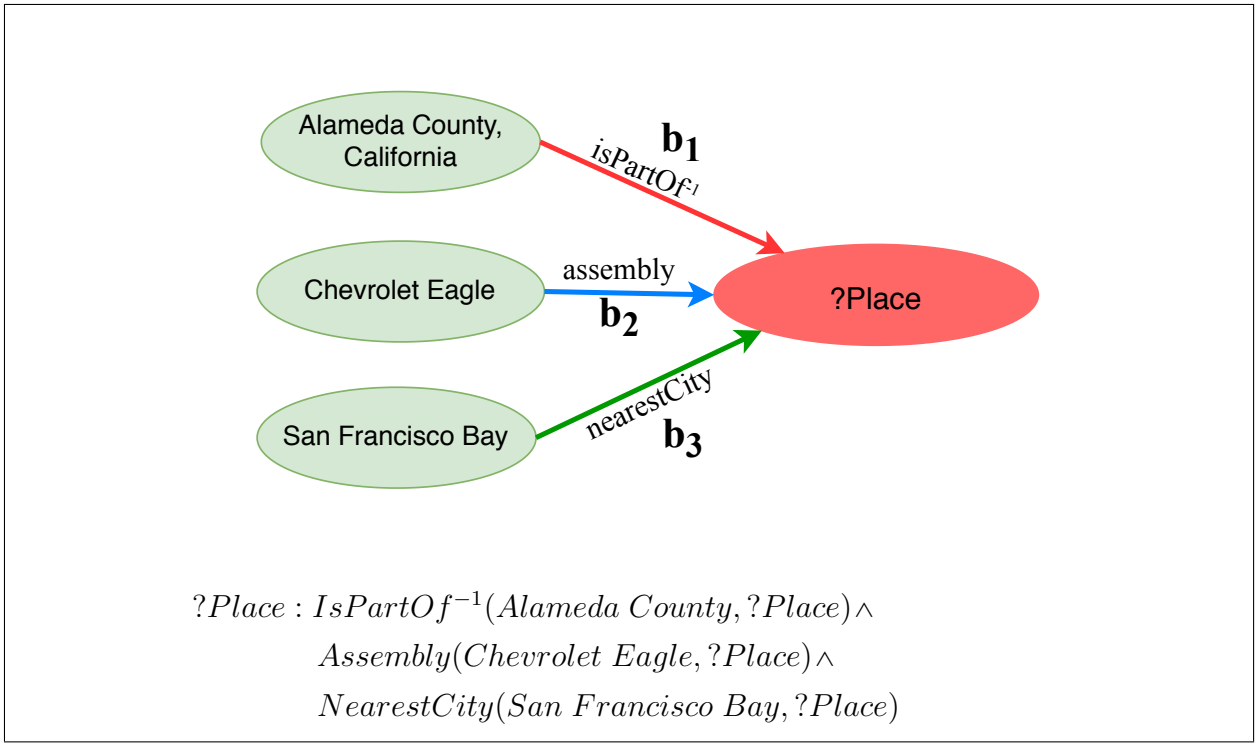
Here $\mathbf{e}_i = Enc(e_i) \in \mathbb{R}^d$ is the entity embedding of e_i produced by an embedding encoder $Enc()$. $\mathbf{\Omega}(\cdot)$ denotes the cosine similarity function:

$$\mathbf{\Omega}(\mathbf{q}, \mathbf{e}_i) = \frac{\mathbf{q} \cdot \mathbf{e}_i}{\| \mathbf{q} \| \| \mathbf{e}_i \|} \tag{3}$$

Note that q can be a geographic query or non-geographic query, i.e., $q \in (Q(\mathcal{G}) \backslash Q_{geo}(\mathcal{G})) \vee Q_{geo}(\mathcal{G})$. *Geographic logic query answering* indicates a logic query answering process over $Q_{geo}(\mathcal{G})$. The query embedding function $\mathbf{\Phi}_{\mathcal{G},\theta}(q)$ is constructed based on all three components of *SE-KGE* without any extra parameters: $Enc()$, $\mathcal{P}()$, and $\mathcal{I}()$, i.e., $\theta = \{\theta_{Enc}, \theta_{\mathcal{P}}, \theta_{\mathcal{I}}\}$.

Task 2 (Spatial Semantic Lifting)**.** *Given a geographic knowledge graph $\mathcal{G}$ and an arbitrary location $\mathbf{x} \in \mathcal{A} \subseteq \mathbb{R}^2$ from the current study area $\mathcal{A}$, and a relation $r \in \mathcal{R}$ such that $Domain(r) \subseteq \mathcal{V}_{pt}$, we define a spatial semantic lifting function $\mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}, r) : \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{R}^d$, which is parameterized by θ_{ssl}, to map $\mathbf{x}$ and r to a vector representation of d dimension, i.e., $\mathbf{s} = \mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}, r) \in \mathbb{R}^d$. A*

[4] `dbo:nearestCity` triples in DBpedia are triplified from the "Nearest major city" row of the info box in each entity's corresponding Wikipedia page which may contain several cities. See http://dbpedia.org/resource/San_Francisco_Bay.



$$?Place : IsPartOf^{-1}(Alameda\ County, ?Place) \wedge$$
$$Assembly(Chevrolet\ Eagle, ?Place) \wedge$$
$$NearestCity(San\ Francisco\ Bay, ?Place)$$

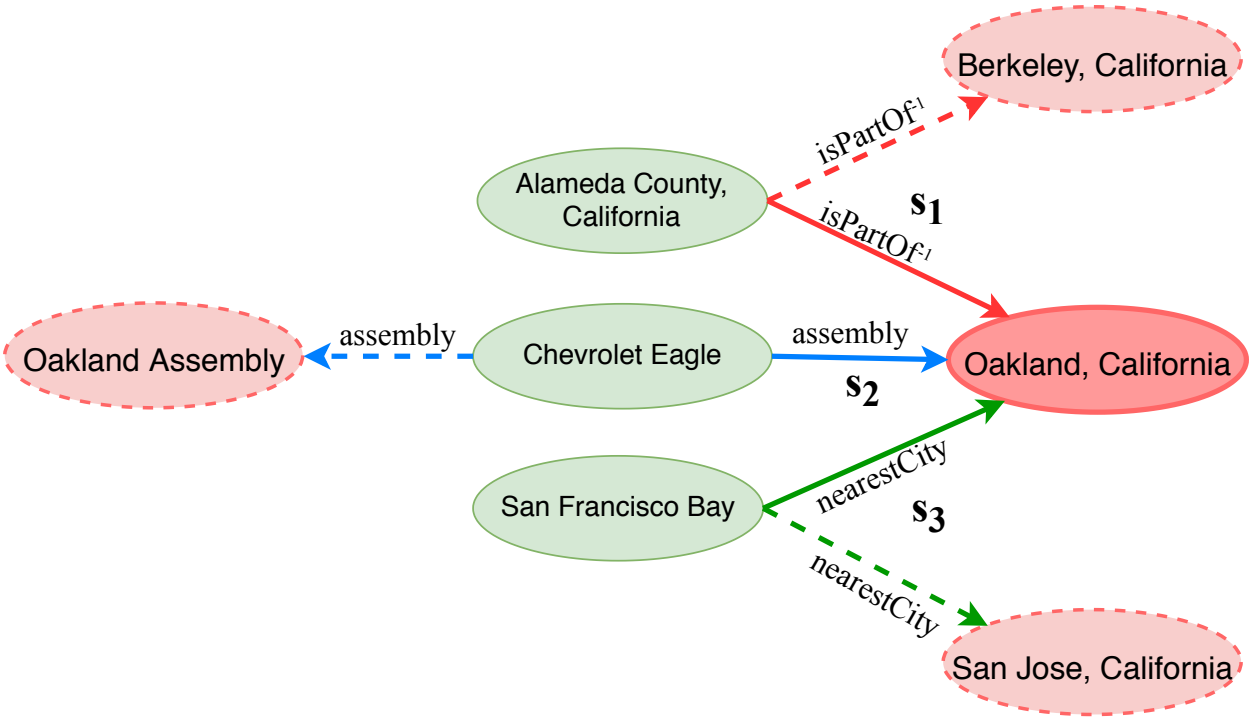


Figure 1: **Query q_C: Top box:** Conjunctive Graph Query and Directed Acyclic Graph of the query structure corresponding to the SPARQL query in Listing 3. b_1, b_2, and b_3 indicates three basic graph patterns in query q_C. `?Place` is the target variable indicated as the red node while three green nodes are anchor nodes. There is no bound variable in this query. **Below:** The matched underlining KG patterns represented by solid arrows. s_1, s_2, and s_3 indicates the matched triples for b_1, b_2, and b_3 respectively for query q_C.

nearest neighbor search is utilized to search for the most probable entity $e' \in \mathcal{V}_{pt}$ so that a virtual triple can be constructed between location $\mathbf{x}$ and e', i.e., $r(\mathbf{x}, e')$, where

$$e' = \arg\max_{e_i \in \mathcal{V}} \ \mathbf{\Omega}(\mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}, r), Enc(e_i)) = \arg\max_{e_i \in \mathcal{V}} \ \mathbf{\Omega}(\mathbf{s}, \mathbf{e}_i) \tag{4}$$

The spatial semantic lifting function $\mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}, r)$ consists of two components of *SE-KGE* without any extra parameter: $Enc()$ and $\mathcal{P}()$, i.e., $\theta_{ssl} = \{\theta_{Enc}, \theta_{\mathcal{P}}\}$. This spatial semantic lifting task is related to the link prediction task (Lao et al., 2011) which is commonly used in the knowledge graph embedding literature (Bordes et al., 2013; Nickel et al., 2016; Cai et al., 2019). The main difference is that instead of predicting links between entities in the original knowledge graph $\mathcal{G}$ as link prediction does, spatial semantic lifting links an arbitrary location $\mathbf{x}$ to $\mathcal{G}$. *Since none of the existing KG embedding models can directly encode locations, they cannot be used for spatial semantic lifting*.

5 Logic Query Answering Backgrounds

Before introducing our *SE-KGE* model, we will first give an overview of how previous work (Hamilton et al., 2018; Mai et al., 2019a) tackled the logic query answering task with KG embedding models. Generally speaking, a logic query answering model is composed of three major components: entity encoder $Enc()$, projection operator $\mathcal{P}()$, and intersection operator $\mathcal{I}()$.

1. **Entity encoder** $Enc()$: represents each entity as a high dimension vector (embedding);
2. **Projection operator** $\mathcal{P}()$: given a basic graph pattern $b = r(e_i, V_j)$ (or $b = r(V_i, V_j)$) in a CGQ q, while the subject embedding $\mathbf{e}_i$ (or $\mathbf{v}_i$) of entity e_i (or Variable V_i) is known beforehand, $\mathcal{P}()$ *projects* the subject embedding through a relation specific matrix to predict the embedding of V_j.
3. **Intersection operator** $\mathcal{I}()$: integrates different predicted embeddings of the same Variable (e.g., V_j) from different basic graph patterns into one single embedding to represent this variable.

Given these three neural network modules, any CGQ q can be encoded according by following their DAG query structures such that the embedding of the unique target variable $V_?$ for each query can be obtained - $\mathbf{v}_?$. We call it *query embedding* $\mathbf{q} = \mathbf{\Phi}_{\mathcal{G},\theta}(q) = \mathbf{v}_?$ for CGQ q. Then the most probable answer is obtained by a nearest neighbor search for $\mathbf{q}$ in the entity embedding space (See Equation 2). Our work will follow the same model component setup and query embedding computing process. However, neither Hamilton et al. (2018) nor Mai et al. (2019a) has considered encoding spatial information of geographic entities into the entity embedding space which is the core contribution of our work. Moreover, we extend the current model architecture such that it can also be applied to the spatial semantic lifting task. This new task cannot be handled by previous work. In the following, we will use $\cdot^{(GQE)}$ and $\cdot^{(CQA)}$ to indicate that these are model components used by Hamilton et al. (2018) and Mai et al. (2019a):

5.1 Entity Encoder

In general, an entity encoder aims at representing any entity in a KG as a high dimension embedding so that it can be fed into following neural network modules. The normal practice shared by most KG embedding models (Bordes et al., 2013; Nickel et al., 2016; Wang et al., 2014; Schlichtkrull et al., 2018; Mai et al., 2018, 2019b; Cai et al., 2019; Qiu et al., 2019) is to initialize an embedding matrix randomly where each column indicates an embedding for a specific entity. The entity encoding becomes an *embedding lookup* process and these embeddings will be updated during the neural network backpropagation during training time.

Previous work has demonstrated that most of the information captured by entity embeddings is type information (Hamilton et al., 2017b, 2018). So Hamilton et al. (2018) and Mai et al. (2019a) took a step further and used a type-specific embedding lookup approach. We call the resulting module entity feature encoder $Enc^{(c)}()$.

Definition 4 (Entity Feature Encoder: $Enc^{(c)}()$)**.** *Given any entity $e_i \in \mathcal{V}$ with type $c_i = \Gamma(e_i) \in \mathcal{C}$ from $\mathcal{G}$, entity feature encoder $Enc^{(c)}()$ computes the feature embedding $\mathbf{e}_i^{(c)} \in \mathbb{R}^{d^{(c)}}$ which captures the type information of entity e_i by using an embedding lookup approach:*

$$\mathbf{e}_i^{(c)} = Enc^{(c)}(e_i) = \frac{\mathbf{Z}_{c_i}\mathbf{h}_i^{(c)}}{\| \mathbf{Z}_{c_i}\mathbf{h}_i^{(c)} \|_{L2}} \tag{5}$$

Here $\mathbf{Z}_{c_i} \in \mathbb{R}^{d^{(c)} \times |\mathcal{C}|}$ is the type-specific embedding matrix for all entities with type $c_i = \Gamma(e_i) \in \mathcal{C}$. $\mathbf{h}_i^{(c)}$ is a one-hot vector such that $\mathbf{Z}_{c_i}\mathbf{h}_i^{(c)}$ will perform an embedding lookup operation which selects an entity feature embedding from the corresponding column. $\| \cdot \|_{L2}$ indicates the $L2$-norm.

Both Hamilton et al. (2018) and Mai et al. (2019a) use $Enc^{(c)}()$ as their entity encoder (See Equation 6). Figure 2 is an illustration of their approach. Note that this encoder does not consider the spatial information (e.g., coordinates and spatial extents) of geographic entities which causes a lower performance for answering geographic logic queries. As for our *SE-KGE* model, we add an additional entity space encoder $Enc^{(x)}()$ to handle this (See Definition 7).

$$Enc^{(GQE)}(e_i) = Enc^{(CGA)}(e_i) = Enc^{(c)}(e_i) \tag{6}$$

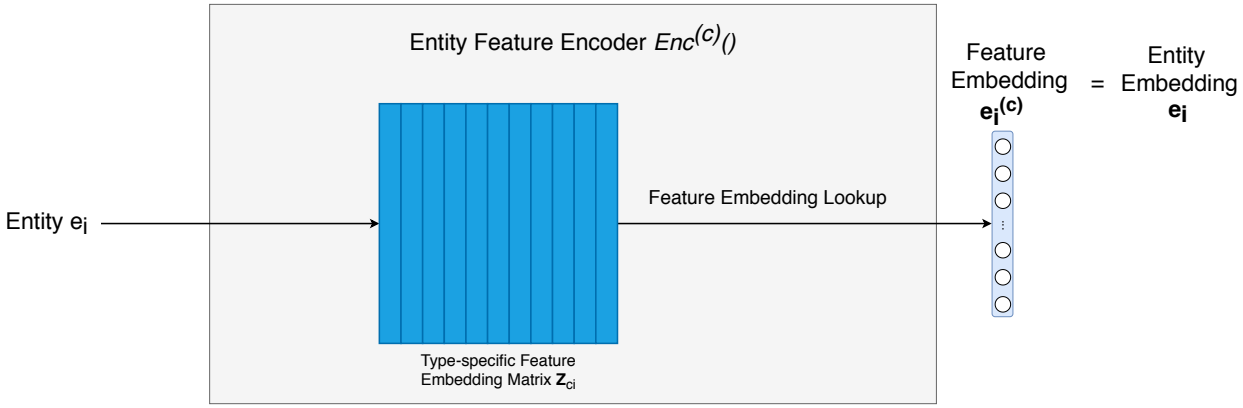


Figure 2: The entity encoder used by Hamilton et al. (2018) and Mai et al. (2019a).

5.2 Projection Operator

The projection operator is utilized to do link prediction: given a basic graph pattern $b = r(h_i, V_j)$ in a conjunctive graph query q with relation r in which h_i is either an entity e_i (an anchor node in q) or an existentially quantified bound variable V_i, the projection operator $\mathcal{P}()$ predicts the embedding $\mathbf{e}'_i \in \mathbb{R}^{d^{(c)}}$ for Variable V_j. Here, the embedding of h_i can be either the entity embedding $\mathbf{e}_i = Enc^{(c)}(e_i)$ or the computed embedding $\mathbf{v}_i$ for V_i which is known beforehand. Both Hamilton et al. (2018) and Mai et al. (2019a) share the same projection operator $\mathcal{P}^{(GQE)} = \mathcal{P}^{(CGA)}$ (See Equation 7) by using a bilinear matrix $\mathbf{R}_r \in \mathbb{R}^{d^{(c)} \times d^{(c)}}$. $\mathbf{R}_r$ can also be a bilinear diagonal matrix as DisMult (Yang et al., 2015) whose corresponding projection operator is indicated as $\mathcal{P}^{(GQE_{diag})}$.

$$\mathbf{e}'_i = \begin{cases} \mathcal{P}^{(GQE)}(e_i, r) = \mathcal{P}^{(CGA)}(e_i, r) = \mathbf{R}_r Enc^{(c)}(e_i) = \mathbf{R}_r \mathbf{e}_i & if\ input\ = (e_i, r) \\ \mathcal{P}^{(GQE)}(V_i, r) = \mathcal{P}^{(CGA)}(V_i, r) = \mathbf{R}_r \mathbf{v}_i & if\ input\ = (V_i, r) \end{cases} \tag{7}$$

In *SE-KGE*, we extend projection operator $\mathcal{P}()$ so that it can be used in the spatial semantic lifting task (See Definition 8).

Figure 3 uses the basic graph pattern $b_2 = Assembly(Chevrolet\ Eagle, ?Place)$ in Figure 1 as an example to demonstrate how to do link prediction with $\mathcal{P}^{(GQE)}() = \mathcal{P}^{(CGA)}()$. The result embedding $\mathbf{e}_{?2}$ can be treated as the prediction of the embedding of Variable `?Place`. By following the same process, we can predict the embedding of the variable `?Place` from the other two basic graph patterns b_1 and b_3 - $\mathbf{e}_{?1}$ and $\mathbf{e}_{?3}$.

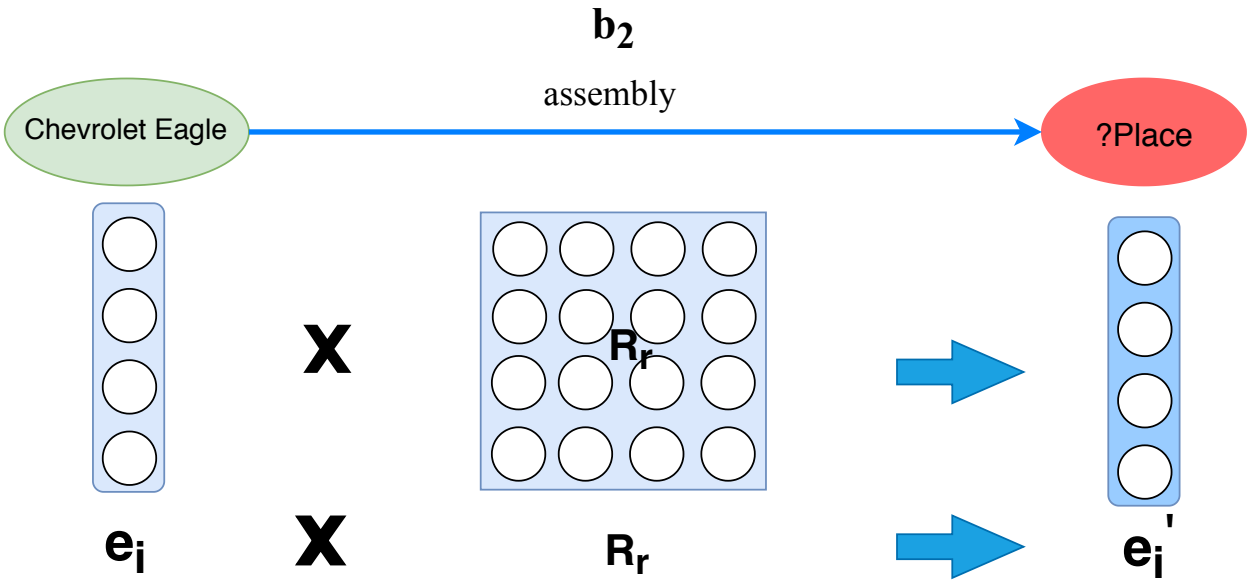


Figure 3: An illustration of projection operator $\mathcal{P}^{(GQE)}() = \mathcal{P}^{(CGA)}()$ used by Hamilton et al. (2018) and Mai et al. (2019a).

5.3 Intersection Operator

The intersection operator $\mathcal{I}()$ is used to *integrate* multiple embeddings $\mathbf{e}_{1?}$, $\mathbf{e}_{2?}$, ..., $\mathbf{e}_{i?}$,..., $\mathbf{e}_{n?}$ which represent the same (bound or target) variable $V_?$ in a CGQ q to produce one single embedding $e_?$ to represent this variable. Figure 4 illustrates this idea by using CGQ q_C in Figure 1 as an example where $\mathbf{e}_{?1}$, $\mathbf{e}_{?2}$ and $\mathbf{e}_{?3}$ indicates the predicted embedding of `?Place` from three different basic graph pattern b_1, b_2, and b_3. The intersection operator *integrates* them into one single embedding $\mathbf{e}_?$ to represent `?Place`. Since `?Place` is the target variable of q, $\mathbf{e}_?$ is the final *query embedding* we use to do nearest neighbor search to obtain the most probable answer (See Task 1). More formally,

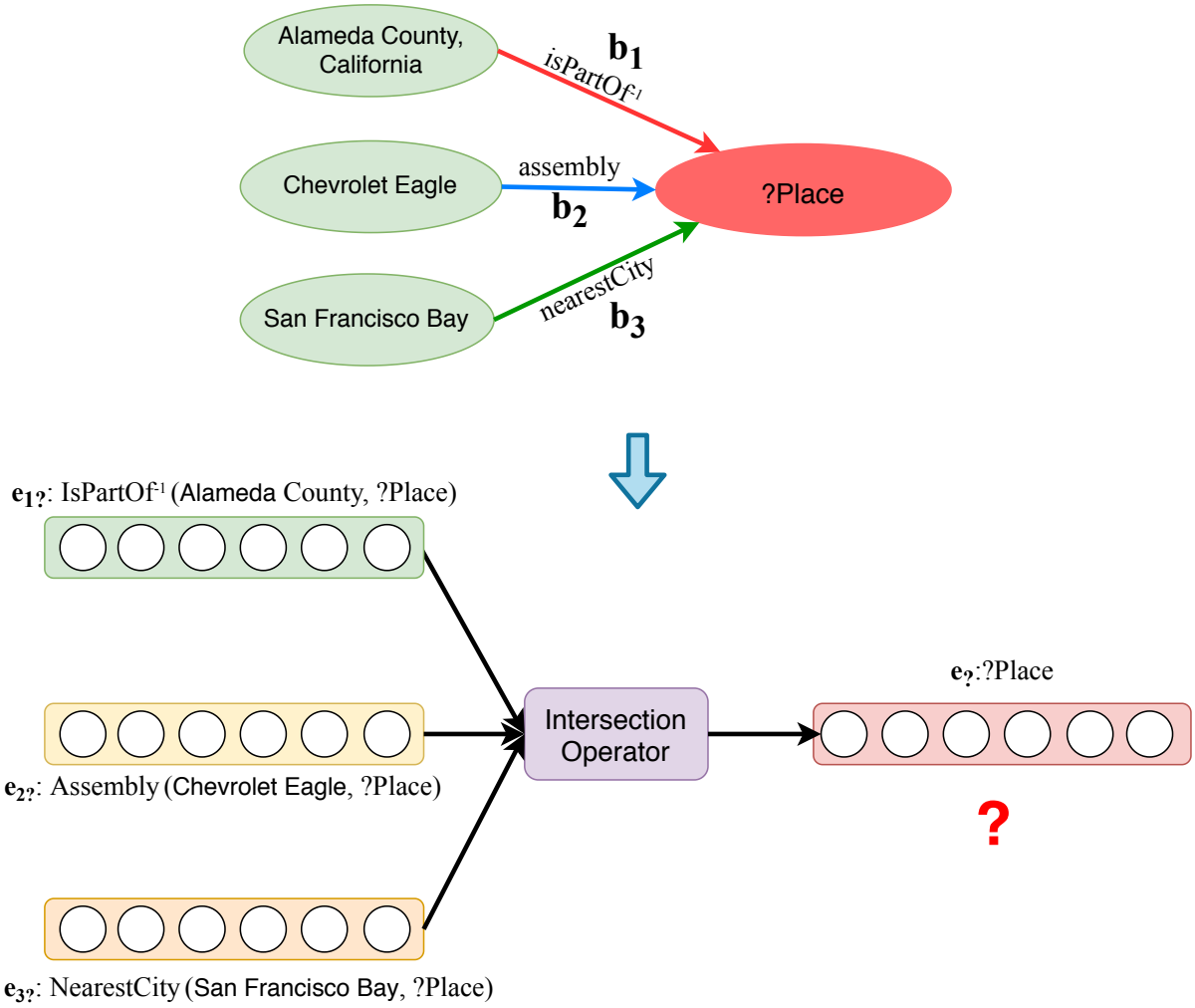


Figure 4: An illustration of intersection operator $\mathcal{I}()$.

Definition 5 (Intersection Operator $\mathcal{I}()$)**.** *Given a set of n different input embeddings $\mathbf{e}_{1?}$, $\mathbf{e}_{2?}$, ..., $\mathbf{e}_{j?}$,..., $\mathbf{e}_{n?}$, intersection operator $\mathcal{I}()$ produces one single embedding $\mathbf{e}_?$:*

$$\mathbf{e}_? = \mathcal{I}(\{\mathbf{e}_{1?}, \mathbf{e}_{2?}, ..., \mathbf{e}_{j?}, ..., \mathbf{e}_{n?}\}) \tag{8}$$

Intersection operator $\mathcal{I}()$ represents the logical conjunction in the embedding space. Any permutation invariant function can be used here as a conjunction such as elementwise mean, maximum, and minimum. We can also use any permutation invariant neural network architecture

(Zaheer et al., 2017) such as Deep Sets (Zaheer et al., 2017). GQE (Hamilton et al., 2018) used an elementwise minimum plus a feed forward network as the intersection operator which we indicate as $\mathcal{I}^{(GQE)}()$. Mai et al. (2019a) showed that their CGA model with a self-attention based intersection operator $\mathcal{I}^{(CGA)}()$ can outperform GQE. So in this work, we use $\mathcal{I}^{(CGA)}()$ as the intersection operator $\mathcal{I}()$. Readers that are interested in this technique are suggested to check Mai et al. (2019a) for more details.

5.4 Query Embedding Computing

Hamilton et al. (2018) proposed a way to compute the query embedding of a CGQ q based on these three components. Given a CGQ q, we can encode all its anchor nodes (entities) into entity embedding space using $Enc()$. Then we recursively apply the projection operator $\mathcal{P}()$ and intersection operator $\mathcal{I}()$ by following the DAG of q until we get an embedding for the target node (variable $V_?$), i.e., $\mathbf{q} = \mathbf{\Phi}_{\mathcal{G},\theta}(q) = \mathbf{v}_?$. Then we use the nearest neighbor search in the entity embedding space to find the *closest* embedding, whose corresponding entity will be the predicted answer to Query q. For details of the query embedding algorithm, please refer to Hamilton et al. (2018).

Figure 5 gives an illustration of the query embedding computation process in the embedding space by using Query q_C as an example. We first use $Enc()$ to get the embeddings of three anchor nodes (see the dash green box in Figure 5.). Then $\mathcal{P}()$ (three green arrows) is applied to each basic graph pattern to get three embeddings $\mathbf{e}_{1?}$, $\mathbf{e}_{2?}$, and $\mathbf{e}_{3?}$. $\mathcal{I}()$ (red arrows) is used later on to integrate them into one single embedding $\mathbf{e}_?$ or $\mathbf{q}$ for the target variable `?Place`.

In this work, we follow the same query embedding computation process. Furthermore, we extent the current model architecture to do spatial semantic lifting.

6 *SE-KGE* Model

Since many geographic questions highly rely on spatial information (e.g., coordinates) and spatial reasoning, a spatially-explicit model is desired for the geographic logic query answering task. Moreover, the spatial semantic lifting task (Task 2) is only possible if we have an entity encoder which can encode the spatial information of geographic entities as well as a specially designed projection operator. To solve these problem, we propose a new entity encoder $Enc()$ (See Section 6.1) and a new projection operator (See Section 6.2) for our *SE-KGE* model. Next, Task 1 and 2 require different training processes which will be discussed in Section 6.3 and 6.4. *SE-KGE* extends the general logic query answering framework of GQE (Hamilton et al., 2018) and CGA (Mai et al., 2019a) with explicit spatial embedding representations.

6.1 Entity Encoder

Definition 6 (Entity Encoder: $Enc()$)**.** *Given a geographic knowledge graph $\mathcal{G}$, entity encoder $Enc() : \mathcal{V} \rightarrow \mathbb{R}^d$ is defined as a function parameterized by θ_{Enc}, which maps any entity $e_i \in \mathcal{V}$ to a vector representation of d dimension, so called entity embedding $\mathbf{e}_i \in \mathbb{R}^d$. $Enc()$ consists of two parts – the entity feature encoder $Enc^{(c)}() : \mathcal{V} \rightarrow \mathbb{R}^{d^{(c)}}$ and the entity space encoder $Enc^{(x)}() : \mathcal{V} \rightarrow \mathbb{R}^{d^{(x)}}$. These two encoders map any entity $e_i \in \mathcal{V}$ to a feature embedding $\mathbf{e}_i^{(c)} \in \mathbb{R}^{d^{(c)}}$ and*

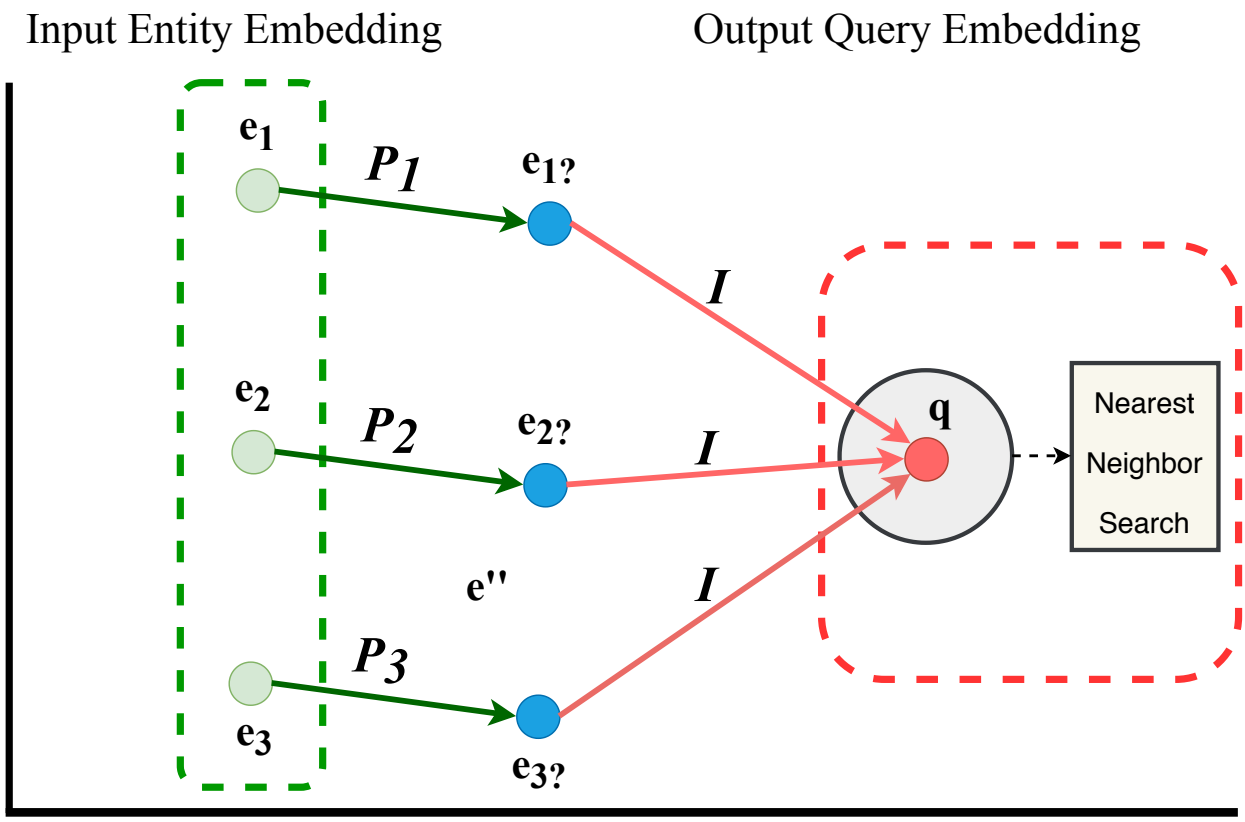


Figure 5: An illustration of (geographic) logic query answering in the embedding space

space embedding $\mathbf{e}_i^{(x)} \in \mathbb{R}^{d^{(x)}}$*, respectively. The final entity embedding* $\mathbf{e}_i$ *is the concatenation of* $\mathbf{e}_i^{(c)}$ *and* $\mathbf{e}_i^{(x)}$*, i.e., :*

$$\mathbf{e}_i = Enc(e_i) = [Enc^{(c)}(e_i); Enc^{(x)}(e_i)] = [\mathbf{e}_i^{(c)}; \mathbf{e}_i^{(x)}] \tag{9}$$

Here $[;]$ denotes vector concatenation of two column vectors and $d = d^{(c)} + d^{(x)}$. $Enc^{(c)}()$ has been defined in Definition 4.

6.1.1 Entity Space Encoder

In our work, and instead of calling them *location encoder* and *location embedding* (Mac Aodha et al., 2019), we use the term *space encoder* to refer to the neural network model that encodes the spatial information of an entity and call the encoding results *space embeddings*. While location encoder focus on encoding one single point location, our space encoder $Enc^{(x)}()$ aims at handling spatial information of geographic entities at different scales:

1. For a small geographic entity $e_i \in \mathcal{V}_{pt} \backslash \mathcal{V}_{pn}$ such as radio stations or restaurants, we use its location $\mathbf{x}_i = \mathcal{PT}(e_i)$ as the input to $Enc^{(x)}()$.

2. For an geographic entity with a large extent $e_i \in \mathcal{V}_{pn}$ such as countries and states, at each encoding time, we randomly generate a point $\mathbf{x}_i^{(t)}$ as the input for $Enc^{(x)}()$ based on the 2D

uniform distribution defined on its spatial extent (bounding box) $\mathcal{PN}(e_i) = [\mathbf{x}_i^{min}; \mathbf{x}_i^{max}]$, i.e., $\mathbf{x}_i^{(t)} \sim \mathcal{U}(\mathbf{x}_i^{min}, \mathbf{x}_i^{max})$. Since during training $Enc^{(x)}()$ will be called multiple times, it will at the end learn a uniform distribution over e_i's bounding box. In practice, one can sample using any process, such as stratified random sampling, or vary the sampling density by expected variation.

3. For non-geographic entity $e_i \in \mathcal{V} \backslash \mathcal{V}_{pt}$, we randomly initialize its space embedding. One benefit of this approach is that during the KG embedding training process, these embeddings will be updated based on back propagation in neural networks so that the spatial information of its connected entities in $\mathcal{G}$ will propagate to this embedding as its pseudo space footprint. For example, a person's spatial embedding will be close to the embedding of his/her birthplace or hometown.

The entity space encoder $Enc^{(x)}()$ is formally defined as follow:

Definition 7 (Entity Space Encoder: $Enc^{(x)}()$)**.** *Given any entity $e_i \in \mathcal{V}$ from $\mathcal{G}$, $Enc^{(x)}()$ computes the space embedding $\mathbf{e}_i^{(x)} = Enc^{(x)}(e_i) \in \mathbb{R}^{d^{(x)}}$ by*

$$\mathbf{e}_i^{(x)} = \begin{cases} LocEnc^{(x)}(\mathbf{x}_i)\ , where\ \mathbf{x}_i = \mathcal{PT}(e_i)\ , & if\ e_i \in \mathcal{V}_{pt} \backslash \mathcal{V}_{pn} \\ LocEnc^{(x)}(\mathbf{x}_i^{(t)})\ , where\ \mathbf{x}_i^{(t)} \sim \mathcal{U}(\mathbf{x}_i^{min}, \mathbf{x}_i^{max})\ ,\ \mathcal{PN}(e_i) = [\mathbf{x}_i^{min}; \mathbf{x}_i^{max}]\ , & if\ e_i \in \mathcal{V}_{pn} \\ \dfrac{\mathbf{Z}_x \mathbf{h}_i^{(x)}}{\parallel \mathbf{Z}_x \mathbf{h}_i^{(x)} \parallel_{L2}}, & if\ e_i \in \mathcal{V} \backslash \mathcal{V}_{pt} \end{cases} \tag{10}$$

Here $\mathbf{Z}_x$ and $\mathbf{h}_i^{(x)}$ are the embedding matrix and one-hot vector for non-geographic entities in entity space encoder $Enc^{(x)}()$ similar to Equation 5. $LocEnc^{(x)}()$ denotes a location encoder module (See Equation 1). Figure 6 illustrates the architecture of entity encoder $Enc()$. Compared with GQE's entity encoder $Enc^{(GQE)}()$ shown in Figure 2, the proposed entity encoder of *SE-KGE* adds the entity space encoder $Enc^{(x)}()$ which leverages a multi-scale grid cell representation to capture the spatial information of geographic entities.

As far as using a bounding box as approximation is concerned, one reason to use bounding boxes instead of the real geometries is that doing point-in-polygon operation in real time during ML model training is very expensive and not efficient. Many spatial databases use bounding boxes as approximations of the real geometries to avoid intensive computation. We adopt the same strategy here. Moreover, the detailed spatial footprint of e_i is expected to be captured through the training process of the entity embedding. For example, even if the model is only aware of the bounding box of California, by using the `dbo:isPartOf` relations between California and its subdivisions, the model will be informed of all the spatial extents of its subdivisions.

6.2 Projection Operator

Definition 8 (Projection Operator $\mathcal{P}()$)**.** *Given a geographic knowledge graph $\mathcal{G}$, a projection operator $\mathcal{P}() : \mathcal{V} \cup \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{R}^d$ maps a pair of (e_i, r), (V_i, r), or $(\mathbf{x}_i, r)$, to an embedding $\mathbf{e}'_i$. According to the input, $\mathcal{P}()$ can be treated as:* ***(1) link prediction*** *$\mathcal{P}^{(e)}(e_i, r)$**:** given a triple's head entity e_i and relation r, predicting the tail; (2)* ***link prediction*** *$\mathcal{P}^{(e)}(V_i, r)$**:** given a basic graph pattern $b = r(V_i, V_j)$ and $\mathbf{v}_i$ which is the computed embedding for the existentially quantified bound*

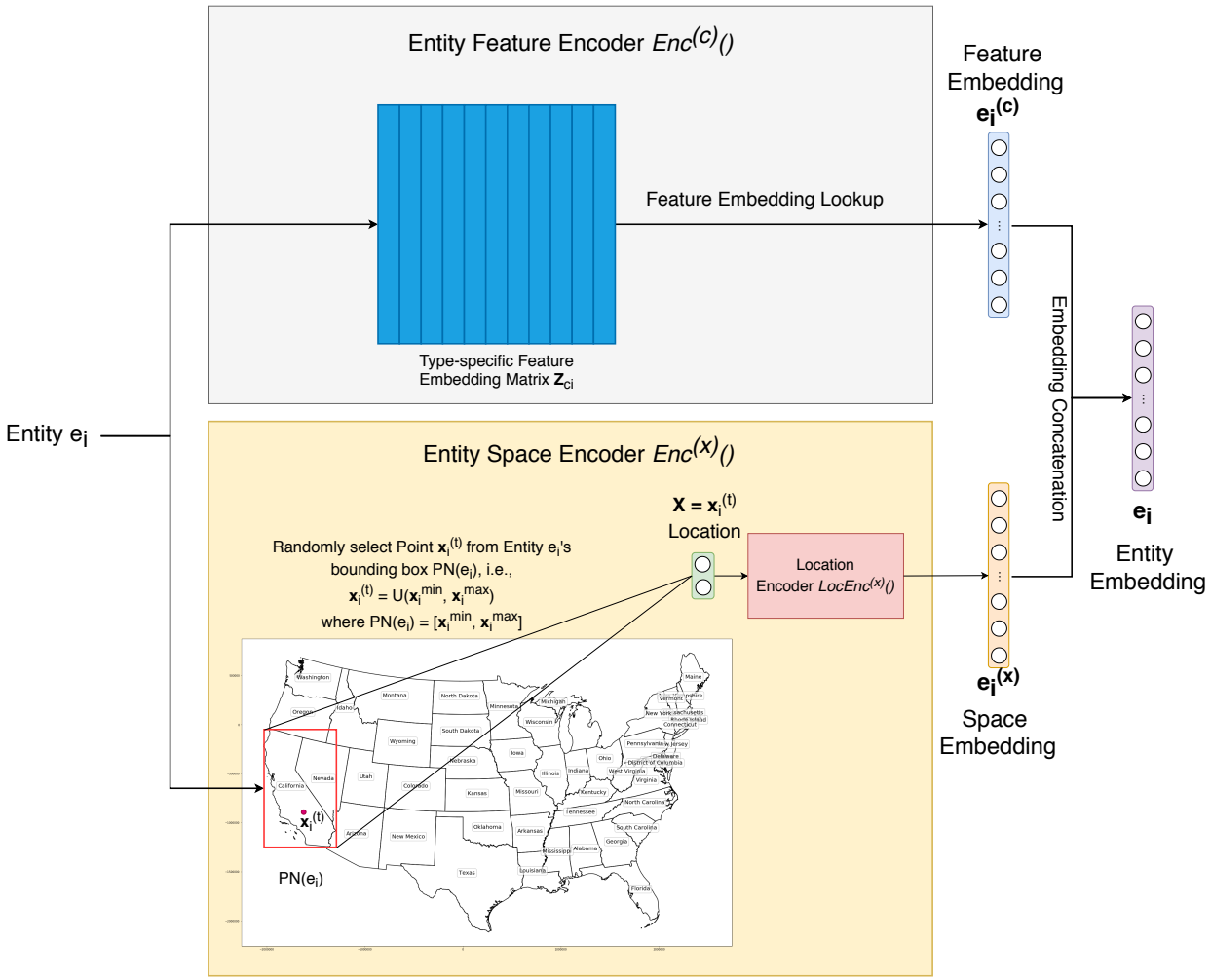


Figure 6: The entity encoder $Enc()$ of *SE-KGE*. Compared with previous work (Figure 2) an entity space encoder component $Enc^{(x)}()$ is added to capture the spatial information of geographic entities.

variable V_i, predicting the embedding for Variable V_j; ***(2) spatial semantic lifting*** *$\mathcal{P}^{(x)}(\mathbf{x}_i, r)$: given an arbitrary location $\mathbf{x}_i$ and relation r, predicting the most probable linked entity. Formally, $\mathcal{P}()$ is defined as:*

$$\mathbf{e}'_i = \begin{cases} \mathcal{P}^{(e)}(e_i, r) = diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)}) Enc(e_i) = diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)})\mathbf{e}_i & if\ input\ = (e_i, r) \\ \mathcal{P}^{(e)}(V_i, r) = diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)})\mathbf{v}_i & if\ input\ = (V_i, r) \\ \mathcal{P}^{(x)}(\mathbf{x}_i, r) = diag(\mathbf{R}_r^{(xc)}, \mathbf{R}_r^{(x)})[LocEnc^{(x)}(\mathbf{x}_i); LocEnc^{(x)}(\mathbf{x}_i)] & if\ input\ = (\mathbf{x}_i, r) \end{cases} \tag{11}$$

where $\mathbf{R}_r^{(c)} \in \mathbb{R}^{d^{(c)} \times d^{(c)}}$, $\mathbf{R}_r^{(x)} \in \mathbb{R}^{d^{(x)} \times d^{(x)}}$, and $\mathbf{R}_r^{(xc)} \in \mathbb{R}^{d^{(c)} \times d^{(x)}}$ are three trainable and relation-specific matrices. $\mathbf{R}_r^{(c)}$ and $\mathbf{R}_r^{(x)}$ focus on the feature embedding and space embedding. $\mathbf{R}_r^{(xc)}$ transforms the space embedding $\mathbf{e}_i^{(x)}$ to its correspondence in feature embedding space. $diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)}) \in \mathbb{R}^{d \times d}$ and $diag(\mathbf{R}_r^{(xc)}, \mathbf{R}_r^{(x)}) \in \mathbb{R}^{d \times 2d^{(x)}}$ indicate two block diagonal matrices based on $\mathbf{R}_r^{(c)}$, $\mathbf{R}_r^{(x)}$, and $\mathbf{R}_r^{(xc)}$. $[LocEnc^{(x)}(\mathbf{x}_i); LocEnc^{(x)}(\mathbf{x}_i)]$ indicates the concatenation of

two identical space embedding $LocEnc^{(x)}(\mathbf{x}_i)$. *Here, we use the same* $\mathcal{P}^{(e)}()$ *for the first two cases to indicate they share the same neural network architecture. This is because both of them are link prediction tasks with different inputs.*

Link Prediction: Figure 7 illustrates the idea of projection operator $\mathcal{P}^{(e)}()$ by using the basic graph pattern b_2 in q_C (See Figure 1) as an example (the first case). Given the embedding of `dbr:Chevrolet_Eagle` and the relation-specific matrix $diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)})$ for relation `dbo:assembly`, we can predict the embedding of the variable `?Place` - $\mathbf{e}_{?2}$.

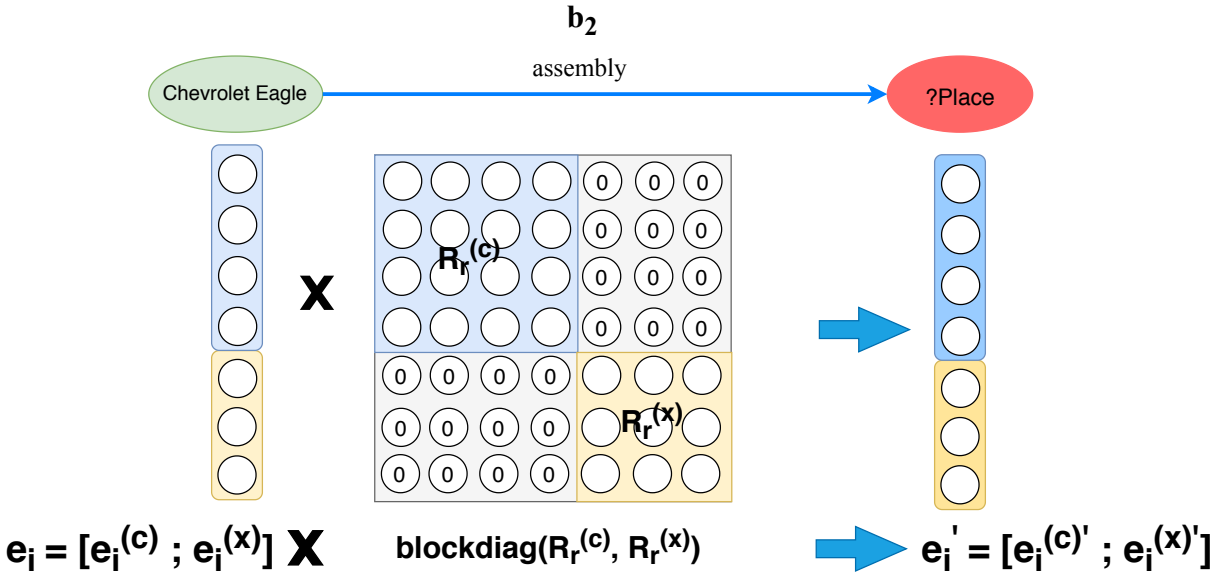


Figure 7: An illustration of projection operator $\mathcal{P}^{(e)}()$ of *SE-KGE* with the input (e_i, r).

Spatial Semantic Lifting: Figure 8 shows how to use $\mathcal{P}^{(x)}()$ in the semantic lifting task. See Section 6.4 for detail description. Note that "$\times$" in Figure 7 and 8 indicates $diag(\mathbf{R}_r^{(c)}, \mathbf{R}_r^{(x)})\mathbf{e}_i$ and $diag(\mathbf{R}_r^{(xc)}, \mathbf{R}_r^{(x)})[LocEnc^{(x)}(\mathbf{x}_i); LocEnc^{(x)}(\mathbf{x}_i)]$.

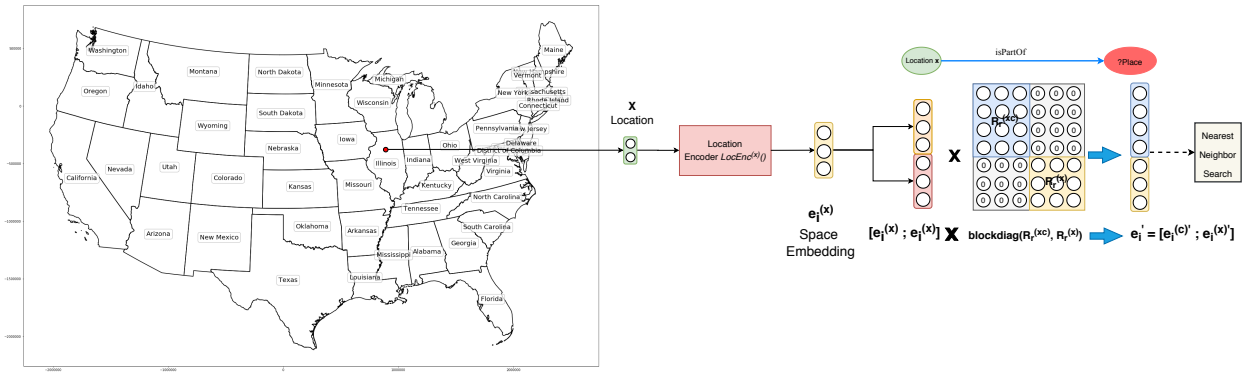


Figure 8: Spatial semantic lifting in the embedding space by using $Enc()$ and $\mathcal{P}^{(x)}()$

6.3 Geographic Logic Query Answering $\Phi_{\mathcal{G},\theta}(q)$ Model Training

We train the *SE-KGE* on both the original knowledge graph structure with an unsupervised objective $\mathcal{L}_{KG}$ and the query-answer pairs with a supervised objective $\mathcal{L}_{QA}$ (See Equation 12):

$$\mathcal{L}^{(QA)} = \mathcal{L}_{KG} + \mathcal{L}_{QA} \tag{12}$$

Unsupervised KG Training Phase In this phase, we train *SE-KGE* components based on the local KG structure. In $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for every entity $e_i \in \mathcal{V}$, we first obtain its 1-degree neighborhood $N(e_i) = \{(r_{ui}, e_{ui}) | r_{ui}(e_{ui}, e_i) \in \mathcal{G}\} \cup \{(r_{oi}^{-1}, e_{oi}) | r_{oi}(e_i, e_{oi}) \in \mathcal{G}\}$. We sample n tuples from $N(e_i)$ to form a sampled neighborhood $N_n(e_i) \subseteq N(e_i)$ and $|N_n(e_i)| = n$. We treat this subgraph as a conjunctive graph query with n basic graph patterns, in which entity e_i holds the target variable position. The model predicts the embedding of e_i such that the correct embedding $\mathbf{e}_i$ is the closest one to the predicted embedding $\mathbf{e}_i''$ against all embeddings $\mathbf{e}_i^-$ in negative sample set $Neg(e_i)$:

$$\mathcal{L}_{KG} = \sum_{e_i \in \mathcal{V}} \sum_{e_i^- \in Neg(e_i)} max(0, \Delta - \mathbf{\Omega}(\mathbf{H}_{KG}(e_i), \mathbf{e}_i) + \mathbf{\Omega}(\mathbf{H}_{KG}(e_i), \mathbf{e}_i^-)) \tag{13}$$

where

$$\mathbf{e}_i'' = \mathbf{H}_{KG}(e_i) = \mathcal{I}(\{\mathcal{P}^{(e)}(e_{ci}, r_{ci}) | (r_{ci}, e_{ci}) \in N_n(e_i)\}) \tag{14}$$

Here $\mathcal{L}_{KG}$ is a max-margin loss and Δ is the margin.

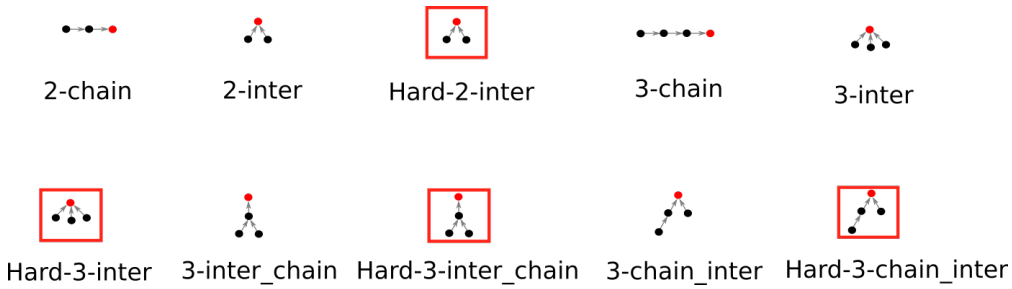


Figure 9: The DAG structures of the conjunctive graph queries we sampled from $\mathcal{G}$. Nodes indicates entities or variables and edges indicate basic graph patterns. The red node is the target variable of the corresponding query. the DAG structures surrounded by red boxes indicate queries sampled with hard negative sampling method.

Supervised Query-Answer Pair Training Phase We train *SE-KGE* by using conjunctive query-answer pairs. We first sample X different conjunctive graph query (logical query)-answer pairs $S = \{(q_i, a_i)\}$ from $\mathcal{G}$. We treat each entity as the target variable of a CQG and sample K queries for each DAG structure. All DAG structures we considered in this work are shown in Figure 9. The way to do query sampling is to sort the nodes in a DAG in a topological order and sample one basic graph pattern at one time by following this order and navigating on the $\mathcal{G}$ (Hamilton et al., 2018). **In order to generate geographic conjunctive graph query, we have the restriction $e_i \in \mathcal{V}_{pt}$.**

The training objective is to make the correct answer entity embedding $\mathbf{a}_i$ be the closest one to the predicted query embedding $\mathbf{q}_i = \mathbf{\Phi}_{\mathcal{G},\theta}(q_i)$ against all the negative answers' embeddings $\mathbf{a}_i^-$ in negative answer set $Neg(q_i, a_i)$. We also use a max-margin loss:

$$\mathcal{L}_{QA} = \sum_{(q_i,a_i)\in S} \sum_{a_i^- \in Neg(q_i,a_i)} max(0, \Delta - \mathbf{\Omega}(\mathbf{q}_i, \mathbf{a}_i) + \mathbf{\Omega}(\mathbf{q}_i, \mathbf{a}_i^-)) \tag{15}$$

For $Neg(q_i, a_i)$ we compared two negative sampling strategies : 1) *negative sampling*: $Neg(q_i, a_i) \subseteq \mathcal{V}$ is a fixed-size set of entities such that $\forall e_i^- \in Neg(q_i, a_i)$, $\Gamma(e_i^-) = \Gamma(e_i)$ *and* $e_i^- \neq e_i$; 2) *hard negative sampling*: $Neg(q_i, a_i)$ is a fixed-size set of entities which satisfy some of the basic graph patterns b_{ij} (See Definition 2) in q_i but not all of them.

6.4 Spatial Semantic Lifting $\mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}, r)$ Model Training

We randomly select a point $\mathbf{x}_i \in \mathcal{A} \subseteq \mathbb{R}^2$ from the study area, and use location encoder $LocEnc^{(x)}$ to encode its location embedding $\mathbf{e}_i^{(x)} \in \mathbb{R}^{d^{(x)}}$. Since we do not have the feature embedding for this location, to make the whole model as an inductive learning one, we use $\mathcal{P}^{(x)}()$ to predict the tail embedding $\mathbf{e}' = \mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}_i, r)$ of this virtual triple $r(\mathbf{x}_i, e')$. This is equivalent to ask a query $r(\mathbf{x}_i, ?e)$ to $\mathcal{G}$. A nearest neighbor search in the entity embedding space will produce the predicted entity who can link to location $\mathbf{x}_i$ with relation r. Since given any location $\mathbf{x}_i$ from the study area, $\mathbf{\Psi}_{\mathcal{G},\theta_{ssl}}(\mathbf{x}_i, r)$ can predict the entity embedding that $\mathbf{x}_i$ can link to given relation r, this is a fully inductive learning based model. This model does not require location $\mathbf{x}_i$ to be selected from a predefined set of locations which is a requirement for transductive learning based models such as Kejriwal and Szekely (2017). Figure 8 shows the idea of spatial semantic lifting.

We train the spatial semantic lifting model *SE-KGE$_{ssl}$* with $Enc()$, $\mathcal{P}^{(e)}()$, and $\mathcal{P}^{(x)}()$ by using two objectives: link prediction objective $\mathcal{L}_{LP}$ and spatial semantic lifting objective $\mathcal{L}_{SSL}$.

$$\mathcal{L}^{(SSL)} = \mathcal{L}_{LP} + \mathcal{L}_{SSL} \tag{16}$$

Link Prediction Training Phase The link prediction training phase aims at training the feature embeddings of each entity. For each triple $s_i = (h_i, r_i, t_i) \in \mathcal{T}$, we can use $Enc()$ and $\mathcal{P}^{(e)}()$ to predict the tail entity embedding given the head and relation - $\mathcal{P}^{(e)}(h_i, r_i)$ - or predict the head entity embedding given the tail and relation - $\mathcal{P}^{(e)}(t_i, r_i^{-1})$. Note that we have two separate $\mathcal{P}^{(e)}()$ for r_i and r_i^{-1}. Equation 17 shows the loss function where $Neg_t(e_i)$ is the set of negative entities who share the same type with entity e_i.

$$\begin{aligned}\mathcal{L}_{LP} = &\sum_{s_i=(h_i,r_i,t_i)\in\mathcal{T}} \sum_{t_i^- \in Neg_t(t_i)} max(0, \Delta - \mathbf{\Omega}(\mathcal{P}^{(e)}(h_i, r_i), \mathbf{t}_i) + \mathbf{\Omega}(\mathcal{P}^{(e)}(h_i, r_i), \mathbf{t}_i^-)) \\ + &\sum_{s_i=(h_i,r_i,t_i)\in\mathcal{T}} \sum_{h_i^- \in Neg_t(h_i)} max(0, \Delta - \mathbf{\Omega}(\mathcal{P}^{(e)}(t_i, r_i^{-1}), \mathbf{h}_i) + \mathbf{\Omega}(\mathcal{P}^{(e)}(t_i, r_i^{-1}), \mathbf{h}_i^-))\end{aligned} \tag{17}$$

Spatial Semantic Lifting Training Phase We also directly optimize our model on the spatial semantic lifting objective. We denote $\mathcal{T}_s$ and $\mathcal{T}_o$ as sets of triples whose head (or tail) entities are

geographic entities, i.e., $\mathcal{T}_s = \{s_i | s_i = (h_i, r_i, t_i) \in \mathcal{T} \wedge h_i \in \mathcal{V}_{pt}\}$ and $\mathcal{T}_o = \{s_i | s_i = (h_i, r_i, t_i) \in \mathcal{T} \wedge t_i \in \mathcal{V}_{pt}\}$. The training objective is to make the tail entity embedding $\mathbf{t}_i$ to be the closest one to the predicted embedding $\mathcal{P}^{(x)}(\mathcal{X}(h_i), r_i)$ against all negative entity embeddings $\mathbf{t}_i^-$. We do the same for the inverse triple (t_i, r_i^{-1}, h_i). The loss function is shown in Equation 18.

$$\begin{aligned}\mathcal{L}_{SSL} = &\sum_{s_i=(h_i,r_i,t_i)\in\mathcal{T}_s} \sum_{t_i^- \in Neg_t(t_i)} max(0, \Delta - \mathbf{\Omega}(\mathcal{P}^{(x)}(\mathcal{X}(h_i), r_i), \mathbf{t}_i) + \mathbf{\Omega}(\mathcal{P}^{(x)}(\mathcal{X}(h_i), r_i), \mathbf{t}_i^-)) \\ + &\sum_{s_i=(h_i,r_i,t_i)\in\mathcal{T}_o} \sum_{h_i^- \in Neg_t(h_i)} max(0, \Delta - \mathbf{\Omega}(\mathcal{P}^{(x)}(\mathcal{X}(t_i), r_i^{-1}), \mathbf{h}_i) + \mathbf{\Omega}(\mathcal{P}^{(x)}(\mathcal{X}(t_i), r_i^{-1}), \mathbf{h}_i^-))\end{aligned} \tag{18}$$

where

$$\mathcal{X}(e_i) = \begin{cases} \mathbf{x}_i = \mathcal{PT}(e_i)\ , & if\ e_i \in \mathcal{V}_{pt} \backslash \mathcal{V}_{pn} \\ \mathbf{x}_i^{(t)} \sim \mathcal{U}(\mathbf{x}_i^{min}, \mathbf{x}_i^{max})\ ,\ \mathcal{PN}(e_i) = [\mathbf{x}_i^{min}; \mathbf{x}_i^{max}]\ , & if\ e_i \in \mathcal{V}_{pn} \end{cases} \tag{19}$$

7 Experiment

To demonstrate how *SE-KGE* incorporates spatial information of geographic entities such as locations and spatial extents we experimented with two tasks – geographic logic query answering and spatial semantic lifting. To demonstrates the effectiveness of spatially explicit models and the importance to considering the scale effect in location encoding we select multiple baselines on the geographic logic query answering task. To show that *SE-KGE* is able to link a randomly selected location to entities in the existing KG with some relation, which none of the existing KG embedding models can solve, we proposed a new task - spatial semantic lifting.

7.1 *DBGeo* Dataset Generation

In order to evaluate our proposed location-aware knowledge graph embedding model *SE-KGE*, we first build a geographic knowledge graph which is a subgraph of DBpedia by following the common practice in KG embedding research (Bordes et al., 2013; Wang et al., 2014; Mai et al., 2019b). We select the mainland of United States as the study area $\mathcal{A}$ since previous research (Janowicz et al., 2016) has shown that DBpedia has relatively richer geographic coverage in United States. The KG construction process is as follows:

1. We collect all the geographic entities within the mainland of United States as the seed entity set $\mathcal{V}_{seed}$ which accounts for 18,780 geographic entities[5]; We then collect their 1- and 2-degree object property triples with `dbo:` prefix predicates/relations[6];

2. We compute the degree of each entity in the collected KG and delete any entity, together with its corresponding triples, if its node degree is less than a threshold η. We use $\eta = 10$ for

[5]We treat an entity as a geographic entity if its has a `geo:geometry` triple in DBpedia
[6]http://dbpedia.org/sparql?help=nsdecl

non-geographic entities and $\eta = 5$ for geographic entities, because many geographic entities, such as radio stations, have fewer object type property triples and a smaller threshold ensures that a relative large number of geographic entities can be extracted from the KG;

3. We further filter out those geographic entities that are newly added from Step 2 and are outside of the mainland of United States. The resulting triples form our KG, and we denote the geographic entity set as $\mathcal{V}_{pt}$.

4. We split $\mathcal{G}$ into training, validation, and testing triples with a radio of 90:1:9 so that every entity and relation appear in the training set. We denote the knowledge graph formed by the training triples as $\mathcal{G}_{train}$ while denoting the whole KG as $\mathcal{G}$.

5. We generate K conjunctive graph query-answer pairs from $\mathcal{G}$ for each DAG structure shown in Figure 9 based on the query-answer generation process we described in Section 6.3. $Q(\mathcal{G})$ and $Q(\mathcal{G})_{geo}$ indicate the resulting QA set while $Q_{geo}(\mathcal{G})$ indicates the geographic QA set. For each query q_i in training QA set, we make sure that each query is answerable based on $\mathcal{G}_{train}$, i.e., $\varphi(\mathcal{G}_{train}, q_i) \neq \varnothing$. As for query q_i in validation and testing QA set, we make sure each query q_i satisfies $\varphi(\mathcal{G}_{train}, q_i) = \varnothing$ and $\varphi(\mathcal{G}, q_i) \neq \varnothing$.

6. For each geographic entity $e \in \mathcal{V}_{pt}$, we obtain its location/coordinates by extracting its `geo:geometry` triple from DBpedia. We project the locations of geographic entities into US National Atlas Equal Area projection coordinate system (epsg:2163) $\mathcal{XY}$. $\mathcal{PT}(e) = \mathbf{x}$ indicates the location of e in the projection coordinate system $\mathcal{XY}$.

7. For each geographic entity $e \in \mathcal{V}_{pt}$, we get its spatial extent (bounding box) $\mathcal{PN}(e)$ in $\mathcal{XY}$ by using ArcGIS Geocoding API[7] and OpenStreetMap API. 80.6% of geographic entities are obtained. We denote them as $\mathcal{V}_{pn}$.

8. For each entity $e_i \in \mathcal{V}$, we obtain its types by using `rdf:type` triples. Note that there are entities having multiple types. We look up the DBpedia Ontology (class hierachy) to get their level-1 superclass. We find out that every entity in $\mathcal{G}$ has only one level-1 superclass type. Table 2 shows statistics of entities in different types.

9. To build the training/validation/testing datasets for spatial semantic lifting, we obtain $\mathcal{T}_s, \mathcal{T}_o \subseteq \mathcal{T}$ (See Section 6.4), each triple of which is composed of geographic entities as its head or tail. We denote $\mathcal{R}_{ssl} = \{r_i | s_i = (h_i, r_i, t_i) \in \mathcal{T}_s \cap \mathcal{T}_o\}$

We denote $Q^{(2)}(\mathcal{G})$, $Q^{(3)}(\mathcal{G})$ as the general QA sets which contain 2 and 3 basic graph patterns, and similarly for $Q^{(2)}_{geo}(\mathcal{G})$, $Q^{(3)}_{geo}(\mathcal{G})$. Table 1 shows the statistics of the constructed $\mathcal{G}$, the generated QA sets, and the spatial semantic lifting dataset in *DBGeo*. Figure 10 shows the spatial distribution of all geographic entities $\mathcal{V}_{pt}$ in $\mathcal{G}$.

[7] https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer/find

Table 1: Statistics for our dataset in *DBGeo* (Section 7.1). "XXXX/QT" indicates the number of QA pairs per query type.

		DBGeo		
		Training	Validation	Testing
Knowledge Graph	$\lvert\mathcal{T}\rvert$	214,064	2,378	21,406
	$\lvert\mathcal{R}\rvert$	318	-	-
	$\lvert\mathcal{V}\rvert$	25,980	-	-
	$\lvert\mathcal{V}_{pt}\rvert$	18,323	-	-
	$\lvert\mathcal{V}_{pn}\rvert$	14,769	-	-
Geographic Question Answering	$\lvert Q^{(2)}(\mathcal{G})\rvert$	1,000,000	-	-
	$\lvert Q^{(3)}(\mathcal{G})\rvert$	1,000,000	-	-
	$\lvert Q^{(2)}_{geo}(\mathcal{G})\rvert$	1,000,000	1000/QT	10000/QT
	$\lvert Q^{(3)}_{geo}(\mathcal{G})\rvert$	1,000,000	1000/QT	10000/QT
Spatial Semantic Lifting	$\lvert\mathcal{T}_s \cap \mathcal{T}_o\rvert$	138,193	1,884	17,152
	$\lvert\mathcal{R}_{ssl}\rvert$	227	71	135

Table 2: Number of entities for each entity type in *DBGeo*

Entity Type	Number of Entities
dbo:Place	16,527
dbo:Agent	8,371
dbo:Work	594
dbo:Thing	179
dbo:TopicalConcept	134
dbo:MeanOfTransportation	104
dbo:Event	71

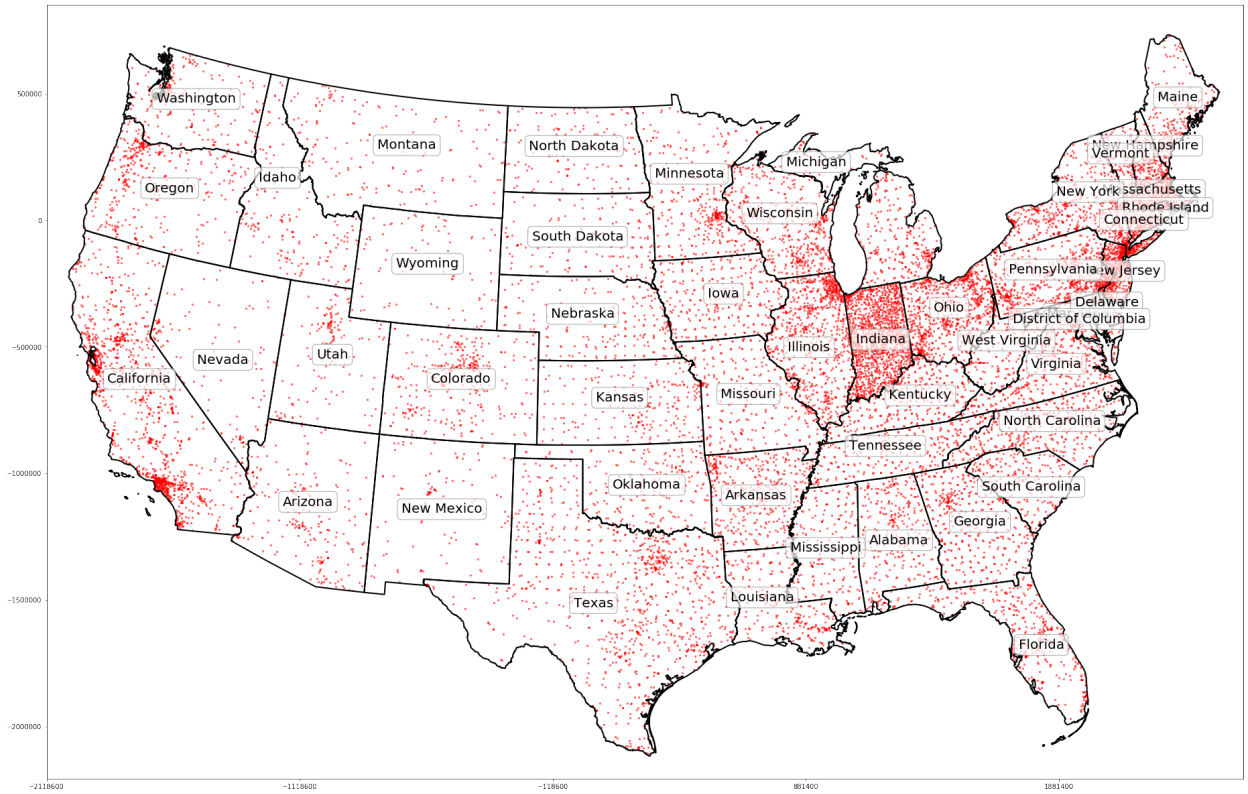


Figure 10: Spatial distribution of all geographic entities in $\mathcal{G}$

7.2 Evaluation on the Geographic Logic Query Answering Task

7.2.1 Baselines

In order to quantitatively evaluate *SE-KGE* on geographic QA task, we train *SE-KGE*$_{full}$ and multiple baselines on $\mathcal{G}$ in *DBGeo*. Compared to previous work (Hamilton et al., 2018; Mai et al., 2019a), the most important contribution of this work is the entity space encoder $Enc^{(x)}()$ which makes our model spatially explicit. So we carefully select four baselines to test the contribution of $Enc^{(x)}()$ on the geographic logic QA task. We have selected four baselines:

1. GQE_{diag} and GQE: two versions of the logic query answering model proposed by Hamilton et al. (2018) which have been discussed in detail in Section 5. The main different between GQE_{diag} and GQE is the projection operator they use: $\mathcal{P}^{(GQE_{diag})}$ and $\mathcal{P}^{(GQE)}$ accordingly. Compared with *SE-KGE*$_{full}$, both GQE_{diag} and GQE only use entity feature encoder $Enc^{(c)}()$ as the entity encoder and $\mathcal{I}^{(GQE)}$ as the intersection operator. Both methods only use $\mathcal{L}_{QA}$ in Equation 12 as the training objective. There two baselines are implemented based on the original code repository[8] of Hamilton et al. (2018).

2. CGA: a logic query answering model proposed by Mai et al. (2019a) (See Section 5). Compared with *SE-KGE*$_{full}$, CGA uses different entity encoder ($Enc^{(CGA)}$) and projection operator ($\mathcal{P}^{(CGA)}$) such that the spatial information of each geographic entity is not considered.

[8] https://github.com/williamleif/graphqembed

This baseline is used to test whether designing spatially explicit logic query answering model can outperform general models on the geographic query answering task.

3. *SE-KGE*$_{direct}$: a simpler version of *SE-KGE*$_{full}$ which uses a **single scale location encoder** in the entity encoder instead of the multi-scale periodic location encoder as shown in Equation 1 in Section 2.3. Instead of first decomposing input $\mathbf{x}$ into a multi-scale periodic representation by using sinusoidal functions with different frequencies (Mai et al., 2020), the location encoder of *SE-KGE*$_{direct}$ directly inputs $\mathbf{x}$ into a feed forward network. This single-scale location encoder is proposed in Mai et al. (2020) as one baseline model - $direct$. Moreover, its entity space encoder does not consider the spatial extent of each geographic entity either and just uses its coordinates to do location encoding. This baseline is used to test the effectiveness of using multi-scale periodical representation learning in our *SE-KGE* framework.

4. *SE-KGE*$_{pt}$: a simpler version of *SE-KGE*$_{full}$ whose entity space encoder does not consider the spatial extents of geographic entities. The only different between *SE-KGE*$_{pt}$ and *SE-KGE*$_{direct}$ is that *SE-KGE*$_{pt}$ uses $Space2Vec$ (Mai et al., 2020) as the location encoder while *SE-KGE*$_{direct}$ utilizes the single scale $direct$ model as the location encoder. This baseline is used to test the necessity to consider the spatial extent of geographic entities in our *SE-KGE* framework. In other words, it uses Equation 20 for its space encoder:

$$\mathbf{e}_i^{(x)} = \begin{cases} LocEnc^{(x)}(\mathbf{x}_i)\ , where\ \mathbf{x}_i = \mathcal{PT}(e_i)\ , & if\ e_i \in \mathcal{V}_{pt} \\ \dfrac{\mathbf{Z}_x \mathbf{h}_i^{(x)}}{\parallel \mathbf{Z}_x \mathbf{h}_i^{(x)} \parallel_{L2}}, & if\ e_i \in \mathcal{V} \backslash \mathcal{V}_{pt} \end{cases} \tag{20}$$

5. *SE-KGE*$_{space}$: a simpler version of *SE-KGE*$_{full}$ whose entity encoder does not have the feature encoder component. This baseline is used to understand how the space encoder $Enc^{(x)}()$ captures the connectivity information of $\mathcal{G}$.

7.2.2 Training Details

We train our model *SE-KGE*$_{full}$ and six baselines on *DBGeo* dataset. GQE_{diag} and GQE are trained on the general QA pairs and geographic QA pairs as Hamilton et al. (2018) did. The other models are additionally trained on the original KG structure. Gird search is used for hyperparameter tuning: $d = [32, 64, 128]$, $d^{(c)} = [16, 32, 64]$, $d^{(x)} = [16, 32, 64]$, $S = [8, 16, 32, 64]$, $\lambda_{min} = [10, 50, 200, 1000]$. The best performance is obtained when $d = 128$, $d^{(c)} = 64$, $d^{(x)} = 64$, $S = 16$, $\lambda_{min} = 50$. $\lambda_{max} = 5400000$ is determined by the study area $\mathcal{A}$. We also try different activation functions (i.e., Sigmoid, ReLU, LeakyReLU) for the full connected layers $\mathbf{NN}()$ of location encoder $LocEnc^{(x)}()$. We find out that *SE-KGE*$_{space}$ achieves the best performance with LeakyReLU as the activation function together with L2 normalization on the location embedding. *SE-KGE*$_{direct}$, *SE-KGE*$_{pt}$, and *SE-KGE*$_{full}$ obtain the best performance with Sigmoid activation function without L2 normalization on the location embedding. We implement all models in PyTorch and train/evaluate each model on a Ubuntu machine with 2 GeForce GTX Nvidia GPU cores, each of which has 10GB memory. The *DBGeo* dataset and related codes will be opensourced.

7.2.3 Evaluation Results

We evaluate *SE-KGE*$_{full}$ and six baselines on the validation and testing QA datasets of *DBGeo*. Each model produces a cosine similarity score between the predicted query embedding $\mathbf{q}$ and the correct answer embedding $\mathbf{a}$ (as well as the embedding of negative answers). The objective is to rank the correct answer top 1 among itself and all negative answers given their cosine similarity to $\mathbf{q}$. Two evaluation metrics are computed: Area Under ROC curve (AUC) and Average Percentile Rank (APR). AUC compares the correct answer with one random sampled negative answer for each query. An ROC curve is computed based on model performance on all queries and the area under this curve is obtained. As for APR, the percentile rank of the correct answer among all negative answers is obtained for each query based on the prediction of a QA model. Then APR is computed as the average of the percentile ranks of all queries. Since AUC only uses one negative sample per query while APR uses all negative samples for each query. We consider APR as a more robust evaluation metric.

Table 3 shows the evaluation results of *SE-KGE*$_{full}$ as well as six baselines on the validation and testing QA dataset of *DBGeo*. We split each dataset into different categories based on their DAG structures (See Figure 9). Note that logic query answering is a very challenging task. As for the two works which share a similar set up as ours, Hamilton et al. (2018) show that their GQE model outperforms TransE baseline by 1.6% of APR on Bio dataset. Similarly, Mai et al. (2019a) demonstrate that their CGA model outperfroms GQE model by 1.39% and 1.65% of APR on DB18 and WikiGeo19 dataset. In this work, we show that our *SE-KGE*$_{full}$ outperforms the current state-of-the-art CGA model by 2.17% and 1.31% in terms of APR on the validation and testing dataset of *DBGeo* respectively. We regard it as a sufficient signal to show the effective of *SE-KGE*$_{full}$ on the geographic QA task. Some interesting conclusions can be drawn from Table 3:

1. CGA has a significant performance improvement over GQE_{diag} and GQE on *DBGeo*. This result is consistent with that of Mai et al. (2019a) which demonstrates the advantage of the self-attention mechanism in $\mathcal{I}^{(CGA)}$.

2. The performance of *SE-KGE*$_{direct}$ and CGA are similar, which shows that a simple single-scale location encoder (*SE-KGE*$_{direct}$) is not sufficient to capture the spatial information of geographic entities.

3. *SE-KGE*$_{full}$ performs better than *SE-KGE*$_{pt}$ which only considers the location information of geographic entities. This illustrates that scale effect is beneficial for the geographic logic QA task.

4. The performance of *SE-KGE*$_{space}$ is the worst among all models. This indicates that it is not enough to only consider spatial information as the input features for entity encoder $Enc()$. This makes sense because each entity in $\mathcal{G}$ has a lot of semantic information other than their spatial information, and only using spatial information for entity embedding learning is insufficient. However, *SE-KGE*$_{space}$ is a fully inductive learning model which enables us to do spatial semantic lifting.

5. Compared *SE-KGE*$_{full}$ with CGA, we can see that *SE-KGE*$_{full}$ outperforms CGA for almost all DAG structures on testing dataset except "Hard-3-chain_inter" (-0.58%) while top 2 DAG

structures with the largest margin are "3-inter_chain" (2.15%) and "3-chain_inter" (2.08%). On the validation dataset, ***SE-KGE***$_{full}$ gets higher ΔAPR compared to CGA on "Hard-3-inter_chain" (7.42%) and "3-inter_chain" (6.08%). GQE_{diag} shows the best performance on "Hard-3-chain_inter" query structure.

In order to demonstrate how the intersection operator $\mathcal{I}()$ helps to improve the model performance on the geographic QA task, we show ***SE-KGE***$_{full}$'s predicted ranking list of entities on Query q_C as well as its three basic graph patterns in Table 4. These 12 entities in this table represent the hard negative sampling set of Query q_C. `dbr:Oakland,_California` is the correct answer for Query q_C. We can see that the top ranked four entities of b_1: $IsPartOf^{-1}(Alameda\ County, ?Place)$ are all subdivisions of Alameda County. The top ranked 5 entities of b_2: $Assembly(Chevrolet\ Eagle, ?Place)$ are all assembly places of Chevrolet Eagle. Similarly, the top ranked entities of b_3: $NearestCity(San\ Francisco\ Bay, ?Place)$ are close to San Francisco Bay. The full query q_C yield the best rank of the correct answer. This indicates that each basic graph pattern contributes to the query embedding prediction of ***SE-KGE***$_{full}$. Moreover, to compare performances of different models on Query q_C, the percentile rank given by CGA, ***SE-KGE***$_{pt}$, and ***SE-KGE***$_{full}$ are 53.9%, 61.5%, and 77.0%, respectively.

We also test how well the location encoder $LocEnc^{(x)}()$ in ***SE-KGE*** can capture the global position information and how $LocEnc^{(x)}()$ interacts with other components of ***SE-KGE***. We use ***SE-KGE***$_{space}$ as an example. Since $LocEnc^{(x)}()$ is an inductive learning model, we divide the study area $\mathcal{A}$ into $20km \times 20km$ grids and take the location of each grid center as the input of $LocEnc^{(x)}()$. Each grid will get a $d^{(x)}$ dimension location embedding after location encoding. We apply hierarchical clustering on these embeddings. Figure 11a shows the clustering result. We compare it with the widely used USA Census Bureau-designated regions[9] (See Figure 11b). We can see that Figure 11a and 11b look very similar to each other. We use two clustering evaluation metrics - Normalized Mutual Information (NMI) and Rand Index - to measure the degree of similarity which yield 0.62 on NMI and 0.63 on Rand Index. To take a closer look at Figure 11a, we can also see that the clusters are divided on the state borders. We hypothesize that this is because $LocEnc^{(x)}()$ is informed of the connectivity of different geographic entities in $\mathcal{G}$ during model training, resulting in that locations which are connected in original $\mathcal{G}$ are also clustered after training.

To validate this hypothesis, we apply Louvain community detection algorithm with a shuffled node sequence[10] on the original $\mathcal{G}$ by treating $\mathcal{G}$ as an undirected and unlabeled graph. Figure 11c shows the community structure with the best modularity which contains 32 communities. Some interesting observations can be made by comparing these three figures:

1. Most communities in Figure 11c are separated at state borders, which is an evidence of our hypothesis;

2. Some communities contain locations at different states, which are far away from each other. For example, the red community which contains locations from Utah, Colorado, and Alabama. This indicates that some locations are very similar purely based on the graph structure of $\mathcal{G}$. As $LocEnc^{(x)}()$ imposes spatial constraints on entities, spatially coherent clusters in Figure 11a are presented.

[9] https://en.wikipedia.org/wiki/List_of_regions_of_the_United_States
[10] https://github.com/tsakim/Shuffled_Louvain

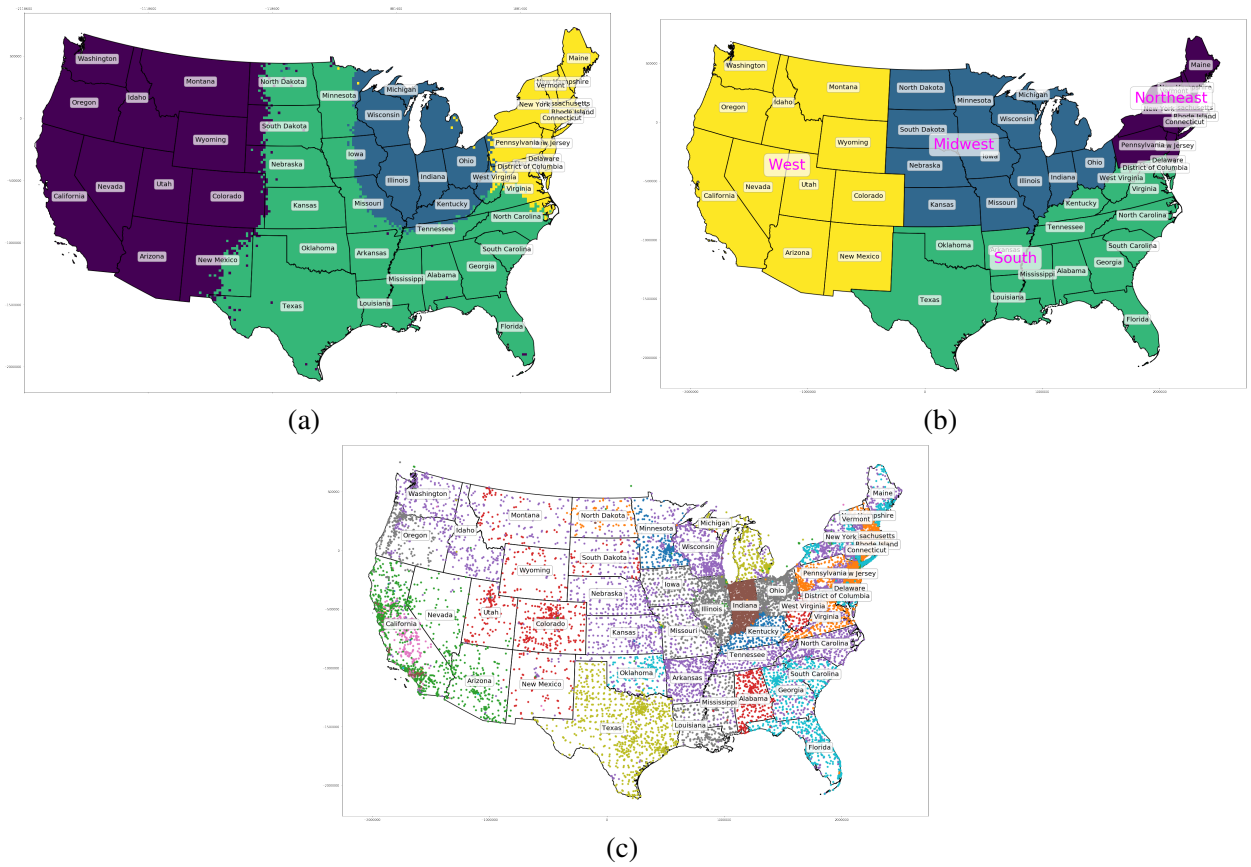


Figure 11: (a) Clustering result of location embeddings produced by the location encoder $LocEnc^{(x)}()$ in *SE-KGE*$_{space}$. It illustrates spatial coherence and semantics (b) Census Bureau-designated regions of United States, and (c) the community detection (Shuffled Louvain) results of knowledge graph $\mathcal{G}$ by treating $\mathcal{G}$ as a undirected unlabeled multigraph. It lacks spatial coherence.

One hypothesis why Figure 11a and 11b look similar is that in the KG, the number of connections between entities within one Bureau-designated region is more than the number of connections among entities in different regions. This may be due to the fact that DBpedia uses census data as one of the data sources while census data is organized in a way which reflects Bureau-designated regions of the US. More research is needed to validate this hypothesis in the future.

7.3 Evaluation on Spatial Semantic Lifting Task

7.3.1 Baselines

The spatial semantic lifting model is composed of $Enc()$, $\mathcal{P}^{(e)}()$, and $\mathcal{P}^{(x)}()$ which is indicated as *SE-KGE*$_{ssl}$. In order to study the contribution of feature encoder and location encoder, we create a baseline *SE-KGE*$'_{space}$ whose entity encoder does not have the feature encoder component, similar to *SE-KGE*$_{space}$. The difference is that they are trained on different objectives. These are the only two models that can do spatial semantic lifting task, since they are fully inductive learning models directly using locations as the only input features.

7.3.2 Training Detail

We train *SE-KGE$_{ssl}$* and *SE-KGE$'_{space}$* based on $\mathcal{L}^{(SSL)}$. To quantitatively evaluate them on spatial semantic lifting task, we use $\mathcal{T}_s \cap \mathcal{T}_o$ in the validation and testing dataset with different relations (See Table 1). For each triple $s_i = (h_i, r_i, t_i) \in \mathcal{T}_s$, given the head entity's location and r_i, we use $\mathcal{P}^{(x)}(\mathcal{X}(h_i), r_i)$ (See Equation 19) to predict the tail entity embedding. Similar process can be done for $s_j = (h_j, r_j, t_j) \in \mathcal{T}_o$ but from the reverse direction. We also use AUC and APR as the evaluation metrics. Note that since $\mathcal{X}(h_i) = \mathbf{x}_i^{(t)} \sim \mathcal{U}(\mathbf{x}_i^{min}, \mathbf{x}_i^{max})$, $\mathcal{PN}(h_i) = [\mathbf{x}_i^{min}; \mathbf{x}_i^{max}]$ if $h_i \in \mathcal{V}_{pn}$, the location of head entity is randomly generated, which can be treated as unseen in the training process. We use the same hyperparameter configuration as *SE-KGE$_{full}$*.

7.3.3 Evaluation Results

Table 5 shows the overall evaluation results. We can see that *SE-KGE$_{ssl}$* outperforms *SE-KGE$'_{space}$* with a significant margin (ΔAUC = 9.86% and ΔAPR = 9.59% on the testing dataset) which clearly shows the strength of considering both feature embedding and space embedding in spatial semantic lifting task.

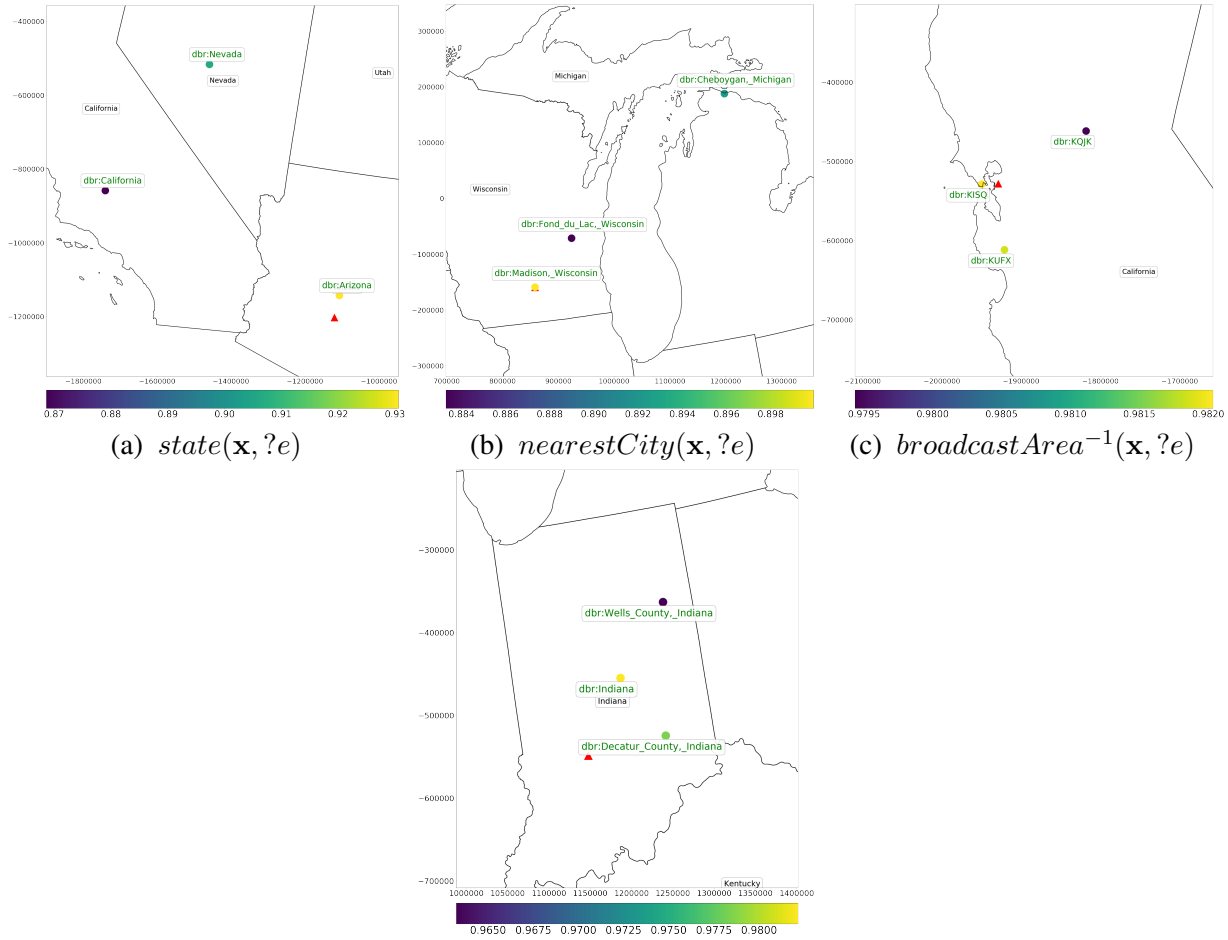
Next, among all validation and testing triples with different relations, we select a few relations and report APR of two models on these triples with specific relations. The results are shown in Table 6. These relations are selected since they are interesting from spatial reasoning perspective. We can see that *SE-KGE$_{ssl}$* outperforms *SE-KGE$'_{space}$* on all these triple sets with different relations.

In order to know how well *SE-KGE$_{ssl}$* understands the semantics of different types of (spatial) relations, we visualize the spatial semantic lifting results in Figure 12 for four spatial relations: `dbo:state`, `dbo:nearestCity`, , `dbo:broadcastArea`$^{-1}$, and `dbo:isPartOf`. `dbo:state`, `dbo:isPartOf`, and `dbo:broadcastArea`$^{-1}$ are about partonomy relations while `dbo:nearestCity` represents an example of point-wise metric spatial relations. Some interesting observations can be made:

1. *SE-KGE$_{ssl}$* is capable of capturing the spatial proximity such that the top 1 geographic entity (yellow point) in each case is the closest to location $\mathbf{x}$ (red triangle). We also treat this as an indicator for the capability of *SE-KGE$_{ssl}$* to handle partonomy relations and point-wise metric spatial relations.

2. *SE-KGE$_{ssl}$* can capture the semantics of relations, e.g., the domain and range of each relation/predicate. All top ranked entities are within the range of the corresponding relation. For example, in Figure 12a with query $state(\mathbf{x}, ?e)$, the top 3 entities are all states spatially close to $\mathbf{x}$. In Figure 12c with query $broadcastArea^{-1}(\mathbf{x}, ?e)$, all top 3 entities are nearby radio stations. In Figure 12d (d) with query $isPartOf(\mathbf{x}, ?e)$, all top 3 entities are states (`dbo:Indiana`) and counties.

3. We notice that the result of query $nearestCity(\mathbf{x}, ?e)$ in Figure 12b is not good enough since the second result - `dbo:Cheboygan,_Michigan` - is outside of Wisconsin. After investigating the triples with `dbo:nearestCity` as the relation, we find out `dbo:nearestCity` usually links a natural resource entity (e.g., lakes, national parks) to a city. These natural resource entities usually cover large area and complex geometries. So `dbo:nearestCity` is not a purely point-wise distance base relation but a complex distance base relation based on their real geometries. Since our model only takes the bounding



(a) $state(\mathbf{x}, ?e)$ (b) $nearestCity(\mathbf{x}, ?e)$ (c) $broadcastArea^{-1}(\mathbf{x}, ?e)$

(d) $isPartOf(\mathbf{x}, ?e)$

Figure 12: The visualization of spatial semantic lifting of *SE-KGE$_{ssl}$*. Figure (a), (b), (c), and (d) shows the top 3 geographic entities which can answer query $r(\mathbf{x}, ?e)$ where r is the relation we pick. **Red triangle**: the select location $\mathbf{x}$. **Circles**: top 3 geographic entities ranked by our model, and their colors indicates cosine similarity between the geographic entities and the predicted query embedding.

box of each entity and there are usually no subdivions of these nature resource entities, it is hard for our model to learn the semantics of `dbo:nearestCity`.

Based on the evaluation results and model analysis, we can see that given a relation r, *SE-KGE$_{ssl}$* is able to link a location $\mathbf{x}$ to an entity e in $\mathcal{G}$ by considering the semantics of r and spatial proximity.

8 Conclusion

In this work, we propose a location-aware knowledge graph embedding model called *SE-KGE* which enables spatial reasoning in the embedding space for its three major components - entity embedding encoder $Enc()$, projection operator $\mathcal{P}()$, and intersection operator $\mathcal{I}()$. We demonstrate how to incorporate spatial information of geographic entities such as locations and spatial extents into $Enc()$ such that *SE-KGE* can handle different types of spatial relations such as point-wise

metric spatial relations and partonomy relations. To the best of our knowledge, this is the first KG embedding model which incorporates location encoding into the model architecture instead of relying on some form of distance measure among entities while capturing the scale effect of different geographic entities. Two tasks have been used to evaluate the performance of *SE-KGE* - geographic logic query answering and spatial semantic lifting. Results show that *SE-KGE*$_{full}$ can outperform multiple baselines on the geographic logic query answering task which indicates the effectiveness of spatially explicit models. It also demonstrates the importance to considering the scale effect in location encoding. Also we proposed a new task - spatial semantic lifting, aiming at linking a randomly selected location to entities in the existing KG with some relation. None of the existing KG embedding models can solve this task except our model. We have shown that *SE-KGE*$_{ssl}$ can significantly outperform the baseline *SE-KGE*$'_{space}$ (ΔAUC = 9.86% and ΔAPR = 9.59% on the testing dataset). Visualizations show that *SE-KGE*$_{ssl}$ can successfully capture the spatial proximity information as well as the semantics of relations. In the future, we hope to explore a more concise way to encode the spatial footprints of geographic entities in a KG. Moreover, we want to explore more varieties of the spatial semantic lifting task.

References

Abbott, A., Callaway, E., 2014. Nobel prize for decoding brain's sense of place. Nature 514 (7521).

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al., 2018. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 .

Berant, J., Chou, A., Frostig, R., Liang, P., 2013. Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 conference on empirical methods in natural language processing. pp. 1533–1544.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795.

Cai, L., Yan, B., Mai, G., Janowicz, K., Zhu, R., 2019. Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In: K-CAP. ACM.

Chu, G., Potetz, B., Wang, W., Howard, A., Song, Y., Brucher, F., Leung, T., Adam, H., 2019. Geo-aware networks for fine-grained recognition. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 0–0.

De Nicola, A., Di Mascio, T., Lezoche, M., Tagliano, F., 2008. Semantic lifting of business process models. In: 2008 12th Enterprise Distributed Object Computing Conference Workshops. IEEE, pp. 120–126.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W., 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 601–610.

Hamilton, W., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J., 2018. Embedding logical queries on knowledge graphs. In: Advances in Neural Information Processing Systems. pp. 2026–2037.

Hamilton, W., Ying, Z., Leskovec, J., 2017a. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. pp. 1024–1034.

Hamilton, W. L., Ying, R., Leskovec, J., 2017b. Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 .

Hoffart, J., Suchanek, F. M., Berberich, K., Weikum, G., 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence 194, 28–61.

Janowicz, K., Hu, Y., McKenzie, G., Gao, S., Regalia, B., Mai, G., Zhu, R., Adams, B., Taylor, K., 2016. Moon landing or safari? a study of systematic errors and their causes in geographic linked data. In: The Annual International Conference on Geographic Information Science. Springer, pp. 275–290.

Janowicz, K., Scheider, S., Pehle, T., Hart, G., 2012. Geospatial semantics and linked spatiotemporal data–past, present, and future. Semantic Web 3 (4), 321–332.

Kejriwal, M., Szekely, P., 2017. Neural embeddings for populated geonames locations. In: International Semantic Web Conference. Springer, pp. 139–146.

Kristiadi, A., Khan, M. A., Lukovnikov, D., Lehmann, J., Fischer, A., 2019. Incorporating literals into knowledge graph embeddings. In: International Semantic Web Conference. Springer, pp. 347–363.

Lao, N., Mitchell, T., Cohen, W. W., Jul. 2011. Random walk inference and learning in a large scale knowledge base. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Edinburgh, Scotland, UK., pp. 529–539.

Liang, C., Berant, J., Le, Q., Forbus, K., Lao, N., 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 23–33.

Mac Aodha, O., Cole, E., Perona, P., 2019. Presence-only geographical priors for fine-grained image classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9596–9606.

Mai, G., Janowicz, K., Yan, B., 2018. Support and centrality: Learning weights for knowledge graph embedding models. In: EKAW. Springer, pp. 212–227.

Mai, G., Janowicz, K., Yan, B., Zhu, R., Cai, L., Lao, N., 2019a. Contextual graph attention for answering logical queries over incomplete knowledge graphs. In: K-CAP. ACM.

Mai, G., Janowicz, K., Yan, B., Zhu, R., Cai, L., Lao, N., 2020. Multi-scale representation learning for spatial feature distributions using grid cells. In: The Eighth International Conference on Learning Representations. openreview.

Mai, G., Yan, B., Janowicz, K., Zhu, R., 2019b. Relaxing unanswerable geographic questions using a spatially explicit knowledge graph embedding model. In: AGILE 2019. Springer, pp. 21–39.

Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E., 2015. A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104 (1), 11–33.

Nickel, M., Rosasco, L., Poggio, T. A., et al., 2016. Holographic embeddings of knowledge graphs. In: AAAI. pp. 1955–1961.

Nickel, M., Tresp, V., Kriegel, H.-P., 2012. Factorizing yago: scalable machine learning for linked data. In: WWW. ACM, pp. 271–280.

Pennington, J., Socher, R., Manning, C. D., 2014. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543.

Pezeshkpour, P., Chen, L., Singh, S., 2018. Embedding multimodal relational data for knowledge base completion. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3208–3218.

Qiu, P., Gao, J., Yu, L., Lu, F., 2019. Knowledge embedding with geospatial distance restriction for geographic knowledge graph completion. ISPRS International Journal of Geo-Information 8 (6), 254.

Rajpurkar, P., Jia, R., Liang, P., 2018. Know what you dont know: Unanswerable questions for squad. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 784–789.

Regalia, B., Janowicz, K., McKenzie, G., 2019. Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data. Transactions in GIS 23 (3), 601–619.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M., 2018. Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. Springer, pp. 593–607.

Schölkopf, B., 2001. The kernel trick for distances. In: Advances in neural information processing systems. pp. 301–307.

Scholkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V., 1997. Comparing support vector machines with gaussian kernels to radial basis function classifiers. IEEE transactions on Signal Processing 45 (11), 2758–2765.

Trisedya, B. D., Qi, J., Zhang, R., 2019. Entity alignment between knowledge graphs using attribute embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. pp. 297–304.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008.

Wang, M., Wang, R., Liu, J., Chen, Y., Zhang, L., Qi, G., 2018. Towards empty answers in sparql: Approximating querying with rdf embedding. In: International Semantic Web Conference. Springer, pp. 513–529.

Wang, Q., Mao, Z., Wang, B., Guo, L., 2017. Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering 29 (12), 2724–2743.

Wang, Z., Zhang, J., Feng, J., Chen, Z., 2014. Knowledge graph embedding by translating on hyperplanes. In: AAAI. Vol. 14. pp. 1112–1119.

Yan, B., Janowicz, K., Mai, G., Zhu, R., 2019. A spatially explicit reinforcement learning model for geographic knowledge graph summarization. Transactions in GIS 23 (3), 620–640.

Yang, B., Yih, W.-t., He, X., Gao, J., Deng, L., 2015. Embedding entities and relations for learning and inference in knowledge bases. In: The Third International Conference on Learning Representations.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., Smola, A. J., 2017. Deep sets. In: Advances in neural information processing systems. pp. 3391–3401.

Zhu, R., Hu, Y., Janowicz, K., McKenzie, G., 2016. Spatial signatures for geographic feature types: Examining gazetteer ontologies using spatial statistics. Transactions in GIS 20 (3), 333–355.

Table 3: The evaluation of geographic logic query answering on *DBGeo* (using AUC (%) and APR (%) as evaluation metric)

	DAG Type	GQE_{diag}		GQE		CGA		*SE-KGE*$_{direct}$		*SE-KGE*$_{pt}$		*SE-KGE*$_{space}$		*SE-KGE*$_{full}$	
		AUC	APR	AUC	APR	AUC	APR	AUC	APR	AUC	APR	AUC	APR	AUC	APR
Valid	2-chain	63.37	64.89	84.23	**88.68**	84.56	86.8	83.12	84.79	**85.97**	84.9	76.81	67.07	85.26	87.25
	2-inter	97.23	97.86	96.00	97.02	98.87	98.58	98.98	98.28	98.95	98.52	85.51	87.13	**99.04**	**98.95**
	Hard-2-inter	70.99	73.55	66.04	73.83	73.43	79.98	73.27	76.36	**74.38**	82.16	63.15	62.91	73.42	**82.52**
	3-chain	61.42	67.94	79.65	79.45	79.11	80.93	77.92	79.26	79.38	83.97	70.09	60.8	**80.9**	**85.02**
	3-inter	98.01	99.21	96.24	98.17	99.18	**99.62**	**99.28**	99.41	99.1	99.56	87.62	89	99.27	99.59
	Hard-3-inter	78.29	85	68.26	77.55	79.59	86.06	79.5	84.28	**80.48**	**87.4**	63.37	67.17	78.86	85.2
	3-inter_chain	90.56	94.08	93.39	91.52	94.59	90.71	95.99	95.11	95.86	94.41	81.16	83.01	**96.7**	**96.79**
	Hard-3-inter_chain	74.19	**83.79**	70.64	74.54	73.97	76.28	74.81	78.9	**76.45**	75.95	65.54	68.21	76.33	83.7
	3-chain_inter	**98.01**	97.45	92.69	93.31	96.72	97.61	97.31	98.67	97.79	**98.76**	83.7	84.42	97.7	98.65
	Hard-3-chain_inter	**83.59**	**88.12**	66.86	74.06	72.12	77.53	73.23	79.24	74.74	80.47	65.13	69.29	74.72	78.11
	Full Valid	81.57	85.19	81.4	84.81	85.21	87.41	85.34	87.43	**86.31**	88.61	74.21	73.9	86.22	**89.58**
Test	2-chain	64.88	65.61	85	87.41	84.91	86.74	83.61	85.97	86.08	88.08	75.46	73.38	**86.35**	**88.12**
	2-inter	96.98	97.99	95.86	97.18	98.79	98.71	98.98	98.94	**98.98**	**99.08**	87.01	85.78	98.93	99.01
	Hard-2-inter	70.39	76.19	64.5	71.86	72.15	79.26	72.04	79.11	**73.72**	**81.78**	61.22	62.97	72.62	81.04
	3-chain	62.3	62.29	79.19	80.19	78.93	80.17	77.53	78.86	79.43	**81.28**	70.55	68.04	**80.49**	80.63
	3-inter	98.09	99.12	96.54	97.94	99.33	99.56	**99.45**	99.47	99.41	**99.63**	88.05	87.63	99.39	99.59
	Hard-3-inter	77.27	83.92	68.69	75.42	78.93	83.52	78.58	84.14	**80.11**	84.87	64.44	64.53	78.76	**84.89**
	3-inter_chain	90.39	91.96	92.54	93.13	93.46	94.36	95.23	95.92	95.02	95.78	81.52	79.61	**95.92**	**96.51**
	Hard-3-inter_chain	72.89	79.12	70.67	75.55	73.47	79.61	73.93	80.21	74.88	79.36	64.99	65.52	**75.36**	**80.72**
	3-chain_inter	97.35	98.27	92.22	94.08	96.55	96.67	97.29	98.39	**97.79**	98.68	85.28	84.08	97.64	**98.75**
	Hard-3-chain_inter	**83.33**	**86.24**	66.77	72.1	72.31	77.89	73.55	77.08	75.19	77.42	65.07	65.41	74.62	77.31
	Full Test	81.39	84.07	81.2	84.49	84.88	87.65	85.02	87.81	**86.06**	88.2	74.36	73.7	86.01	**88.96**

Table 4: The rank of entities in the hard negative sample set of Query q_C based on *SE-KGE*$_{full}$'s prediction for different queries: 1) b_1: $IsPartOf^{-1}(Alameda\ \ County, ?Place)$; 2) b_2: $Assembly(Chevrolet\ \ Eagle, ?Place)$; 3) b_3: $NearestCity(San\ Francisco\ Bay, ?Place)$; 4) The Full Query q_C. The correct answer is highlighted as bold.

	b_1	b_2	b_3	Query q_C
1	dbr:Emeryville,_California	dbr:Flint_Truck_Assembly	dbr:Alameda,_California	dbr:San_Jose,_California
2	dbr:Castro_Valley,_California	dbr:Norwood,_Ohio	dbr:San_Jose,_California	**dbr:Oakland,_California**
3	dbr:Alameda,_California	dbr:Flint,_Michigan	dbr:Berkeley,_California	dbr:Berkeley,_California
4	dbr:Berkeley,_California	dbr:Tarrytown,_New_York	**dbr:Oakland,_California**	dbr:Alameda,_California
5	dbr:San_Jose,_California	**dbr:Oakland,_California**	dbr:Emeryville,_California	dbr:San_Francisco
6	dbr:Fremont,_California	dbr:Alameda,_California	dbr:Fremont,_California	dbr:Fremont,_California
7	**dbr:Oakland,_California**	dbr:San_Jose,_California	dbr:Norwood,_Ohio	dbr:Emeryville,_California
8	dbr:San_Francisco	dbr:San_Francisco	dbr:San_Francisco	dbr:Castro_Valley,_California
9	dbr:Norwood,_Ohio	dbr:Berkeley,_California	dbr:Flint_Truck_Assembly	dbr:Flint,_Michigan
10	dbr:Flint_Truck_Assembly	dbr:Fremont,_California	dbr:Flint,_Michigan	dbr:Norwood,_Ohio
11	dbr:Tarrytown,_New_York	dbr:Emeryville,_California	dbr:Castro_Valley,_California	dbr:Flint_Truck_Assembly
12	dbr:Flint,_Michigan	dbr:Castro_Valley,_California	dbr:Tarrytown,_New_York	dbr:Tarrytown,_New_York

Table 5: The evaluation of spatial semantic lifting on *DBGeo* over all validation/testing triples

	SE-KGE$_{space}$		*SE-KGE*$_{ssl}$		*SE-KGE*$_{ssl}$ - *SE-KGE*$_{space}$	
	AUC	APR	AUC	APR	ΔAUC	ΔAPR
Valid	72.85	75.49	**82.74**	**85.51**	9.89	10.02
Test	73.41	75.77	**83.27**	**85.36**	9.86	9.59

Table 6: The evaluation of *SE-KGE*$_{ssl}$ and *SE-KGE*$'_{space}$ on *DBGeo* for a few selected relation r (using APR (%) as evaluation metric).

	Query Type	*SE-KGE*$'_{space}$	*SE-KGE*$_{ssl}$	ΔAPR
Valid	$state(\mathbf{x}, ?e)$	92.00	**99.94**	7.94
	$nearestCity(\mathbf{x}, ?e)$	84.00	**94.00**	10.00
	$broadcastArea^{-1}(\mathbf{x}, ?e)$	91.60	**95.60**	4.00
	$isPartOf(\mathbf{x}, ?e)$	88.56	**98.88**	10.32
	$locationCity(\mathbf{x}, ?e)$	83.50	**99.00**	15.50
	$residence^{-1}(\mathbf{x}, ?e)$	90.50	**93.50**	3.00
	$hometown^{-1}(\mathbf{x}, ?e)$	61.14	**74.86**	13.71
Test	$state(\mathbf{x}, ?e)$	89.06	**99.97**	10.91
	$nearestCity(\mathbf{x}, ?e)$	87.60	**99.80**	12.20
	$broadcastArea^{-1}(\mathbf{x}, ?e)$	90.81	**96.63**	5.82
	$isPartOf(\mathbf{x}, ?e)$	87.66	**98.87**	11.21
	$locationCity(\mathbf{x}, ?e)$	84.80	**99.10**	14.30
	$residence^{-1}(\mathbf{x}, ?e)$	61.21	**77.68**	16.47
	$hometown^{-1}(\mathbf{x}, ?e)$	61.44	**76.83**	15.39