NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

MSc THESIS

Geospatial Query Answering Using Knowledge Graph Embeddings

Markos K. Iliakis

Supervisor: **Manolis Koubarakis,** Professor

Co-Supervisor: **Eleni Tsalapati,** Associate Researcher

ATHENS

OCTOBER 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Απάντηση Γεωχωρικών Ερωτημάτων Χρησιμοποιώντας Ενσωματώσεις Γράφων Γνώσης

Μάρκος Κ. Ηλιάκης

Επιβλέπων: **Μανόλης Κουμπαράκης,** Καθηγητής

Συνεπιβλέπων: **Ελένη Τσαλαπάτη,** Συνεργαζόμενος Ερευνητής

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2022

MSc THESIS

Geospatial Query Answering Using Knowledge Graph Embeddings

Markos K. Iliakis
S.N.: ds1190006

SUPERVISOR: **Manolis Koubarakis,** Professor

COSUPERVISOR: **Eleni Tsalapati,** Associate Researcher

THESIS COMMITTEE: **Dimitrios Gounopoulos,** Professor
Manolis Koubarakis, Professor
Alexandros Ntoulas, Professor

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Απάντηση Γεωχωρικών Ερωτημάτων Χρησιμοποιώντας Ενσωματώσεις Γράφων Γνώσης

Μάρκος Κ. Ηλιάκης
Α.Μ.: ds1190006

ΕΠΙΒΛΕΠΩΝ: **Μανόλης Κουμπαράκης,** Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ: **Ελένη Τσαλαπάτη,** Συνεργαζόμενος Ερευνητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Δημήτριος Γουνόπουλος,** Καθηγητής
Μανόλης Κουμπαράκης, Καθηγητής
Αλέξανδρος Ντούλας, Καθηγητής

ABSTRACT

Geospatial knowledge graphs suffer from incompleteness which is due to the not-always-reliable data sources. This dramatically affects the results of geospatial query answering with traditional techniques which use standard query languages like stSPARQL or GeoSPARQL.An alternative method for query answering is by using KG embeddings. Embedding-based models project entities and relations of the posed query onto the continuous vector space, predicting, this way, the answers to the posed query. Hence, they can handle queries for which the data required for their answering is not explicitly stated in the knowledge graph. In this research work, we have developed the embedding-based geospatial query answering model, SQABo, which encodes the geospatial queries as boxes into the embedding space and returns the answers inside the box. We show that this approach performs better than existing work in the literature, which encodes the queries as points in the vector space. Additionally, we make freely available a query-answering dataset for YAGO2geo, one of the richest and most precise geospatial knowledge graphs, to the research community for future research.

SUBJECT AREA: Information systems Geographic information systems

KEYWORDS: Geospatial Query Answering, Geospatial Knowledge Graph Embeddings

ΠΕΡΙΛΗΨΗ

Τα γραφήματα γεωχωρικής γνώσης πάσχουν από ελλιπή στοιχεία, τα οποία οφείλονται στις όχι πάντα αξιόπιστες πηγές δεδομένων. Αυτό επηρεάζει δραματικά τα αποτελέσματα της απάντησης γεωχωρικών ερωτημάτων με τις παραδοσιακές τεχνικές που χρησιμοποιούν τυποποιημένες γλώσσες ερωτημάτων όπως η stSPARQL ή η GeoSPARQL. Τα μοντέλα που βασίζονται στην ενσωμάτωση προβάλλουν τις οντότητες και τις σχέσεις του ερωτήματος που τίθεται στον συνεχή διανυσματικό χώρο, προβλέποντας, με αυτόν τον τρόπο, τις απαντήσεις στο ερώτημα που τίθεται. Ως εκ τούτου, μπορούν να χειριστούν ερωτήματα για τα οποία τα δεδομένα που απαιτούνται για την απάντησή τους δεν δηλώνονται ρητά στον γράφο γνώσης. Στην παρούσα ερευνητική εργασία, αναπτύξαμε το μοντέλο απάντησης γεωχωρικών ερωτημάτων με βάση την ενσωμάτωση, SQABo, το οποίο κωδικοποιεί τα γεωχωρικά ερωτήματα ως κουτιά στον χώρο ενσωμάτωσης και επιστρέφει τις απαντήσεις εντός του κουτιού. Δείχνουμε ότι αυτή η προσέγγιση έχει καλύτερες επιδόσεις από τις υπάρχουσες εργασίες στη βιβλιογραφία, οι οποίες κωδικοποιούν τα ερωτήματα ως σημεία στο διανυσματικό χώρο. Επιπλέον, διαθέτουμε ελεύθερα στην ερευνητική κοινότητα ένα σύνολο δεδομένων για την απάντηση ερωτημάτων για το YAGO2geo, έναν από τους πλουσιότερους και ακριβέστερους γράφους γεωχωρικής γνώσης, για μελλοντική έρευνα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Πληροφοριακά Συστήματα, Γεωγραφικα πληροφοριακα συστηματα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Απάντηση Γεωχωρικών Ερωτημάτων, Ενσωματώσεις Γεωχωρικών Γράφων Γνώσης

To my family.

ACKNOWLEDGEMENTS

I would like to thank Professor Manolis Koubarakis for his guidance throughout the year, as well as for giving me the opportunity to become a member of the AI Research Group, and therefore meet and collaborate with great researchers. I would also like to thank Eleni Tsalapati for her continuous guidance and help during the development of this thesis.

The present work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the first Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Problem Statement

Geospatial data and knowledge have become ubiquitous in the Web today and in applications, such as navigation, smart cities, Earth observation, etc. To retrieve efficiently such geospatial knowledge, several geospatial knowledge graphs (KGs) have been proposed in the literature (e.g., YAGO2geo [16], WorldKG [6], KnowWhereGraph [13]). *Geospatial knowledge graphs* enable the representation of geospatial knowledge in a semantically enriched, formal and structured way using ontologies and the RDF data model.

The standard way to retrieve knowledge from such geospatial KGs is by using geospatial query answering systems, such as Strabo 2 [**?**] or GraphDB[1], which perform geospatial processing over RDF graphs, mainly targeting the GeoSPARQL [18] vocabulary and query language. A bottleneck of such approaches is that they require that the targeted KG is complete, which rarely happens. Incompleteness in knowledge graphs occur either from wrong or from incomplete data entries (e.g. missing relations, entities or attributes such as location) due to the different and often crowd-sourced data sources (e.g., Wikidata, DBPedia). Moreover, geospatial knowledge suffers from an intrinsic vagueness [27]: the shape of some geospatial features (e.g., forest, mountain) cannot be precisely defined (e.g., borders of a mountain and a valley), and, in cases that these can be defined, timeliness (e.g., the sizes of cities can change through time) and administrative definitions (e.g., boundaries of administrative regions can change from government authorities) often affect the query answering results.

One way to address these issues is by employing KG embedding techniques. Embedding-based models project entities and relations of the queries and the KGs onto a continuous vector space. This way, they are able to predict an embedding (representing the query), "close enough" to the desired answer entity embedding. Hence, they can handle queries for which the data required for their answering is not explicitly stated in the KG.

Current state-of-the-art geospatial query answering models that use KG embeddings were initially introduced by Nickel et al., 2012 [25] and later extended by Hamilton et al., 2018 [10] and Wang et al., 2018 [32]. The geospatial component of the geospatial KGs introduces the additional difficulty of having to encode, also, datatype properties (e.g., points, polygons) into the KG embedding space. One of the first attempts for geospatial encoding was Space2Vec [21], which was later exploited by the location-aware model for query answering SE-KGE [19]. SE-KGE, uses a feature and geospatial (Space2Vec) encoder to better capture all the aspects of each entity and returns the k-nearest neighbors to the target node. However, as in most such systems, the number k is rather arbitrarily defined, hence we may miss important answers or get some redundant ones. To overcome this issue, recently, Ren et al. [28], proposed the framework Query2Box, with which the queries are represented as trainable boxes in the vector space and the answers as sets of points within these boxes.

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

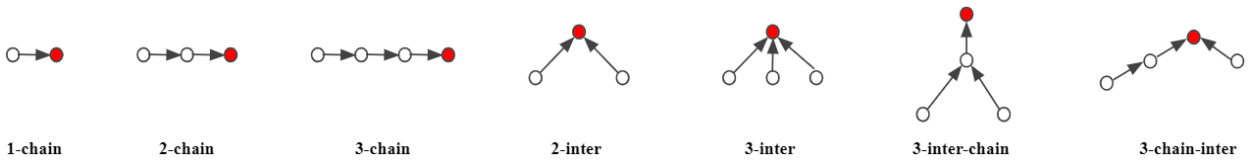

Figure 1.1: The DAG Structures we used in the sampling stage. In each conjunctive graph query, there is a target node (red nodes), one or more anchor nodes (starting white nodes) and zero or more variable nodes (intermediate white nodes). The target node represents the desired answer and the anchor nodes represent the locations given in the query.

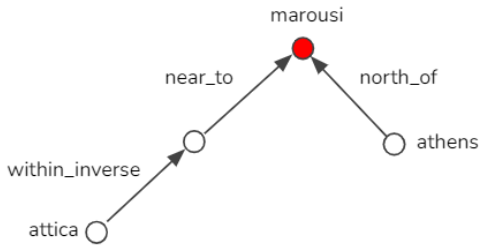

Figure 1.2: The DAG graph of the query $Q(x) = \exists y : north\_of(athens, x) \wedge near\_To(y, x) \wedge within\_inverse(attica, y)$, which corresponds to the question "Which places are north of Athens and near to another place which is within Attica?".

1.2 Project Scope

In this thesis, we will address the research question of whether the transformation of the encoded geospatial queries into boxes can optimize the results of SE-KGE.

Since SE-KGE is the starting point of our research, we focus on answering the same type of conjunctive queries as SE-KGE. Specifically, these types are presented in Figure 1.1, where each query has a query answer node (target node illustrated with red in the figure) and one (1-,2-,3-chain) or more instances of the KG. An example of such a query is illustrated in Figure 1.2.

We train and test our model by using two geospatial KGs: i) YAGO2geo[2], and ii) DBGeo[3]. We chose YAGO2geo as it is one of the richest and most precise geospatial KGs. DBGeo[4], was chosen because it was also used for the evaluation of SE-KGE in [19].

1.3 Aim and Objectives

The overarching aim of this thesis is to create a model for query answering over geospatial knowledge graphs that will be able to handle queries for which, the data required for their answering is not explicitly stated in the KG. More specifically, the objectives for this thesis are formed as follows:

- Objective O1: To study the state-of-the-art in QA over KGs with the use of KG embeddings and, then, examine further the model that focus on geospatial QA. This will lead to the identification of challenges that remain open in the literature.

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

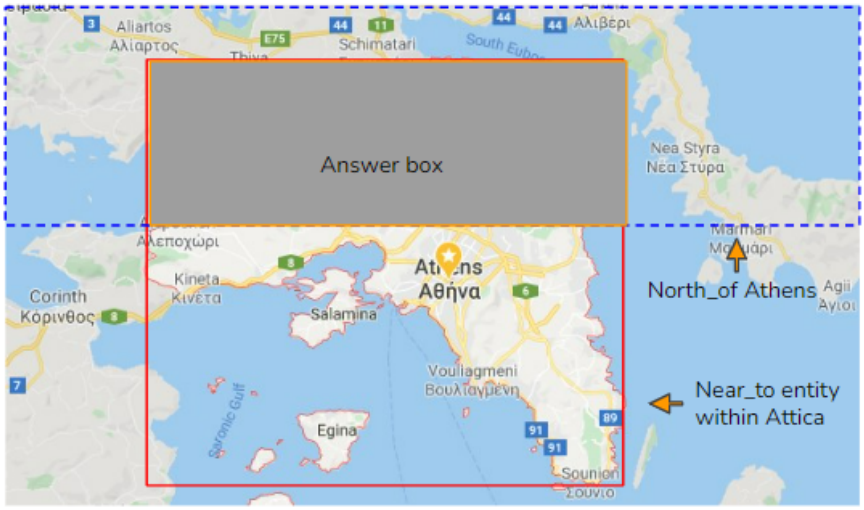

Figure 1.3: The output of SQABo for the query presented in Figure 1.2

.

- Objective O2: To design the architecture and implement geospatial QA model over KGs that will address some of the challenges identified in O1.
- Objective O3: To train and evaluate the model and fine-tune its hyper-parameters.

1.4 Methodology

In this thesis, we develop the novel geospatial query answering model SQABo (geoSpatial conjunctive graph Query Answering based on knowledge graph embeddings with Boxes), by leveraging the techniques developed for both SE-KGE and Query2Box. Particularly, SQABo initially encodes the entities and relations appearing in any incoming conjunctive graph query, utilizing the entity encoder developed in SE-KGE. Then, by using techniques from Query2Box, these vectors are gradually projected into boxes. The answer to the input conjunctive graph query is resulted by intersecting these boxes using contextual graph attention and returning all the entities that are inside this box. In this way, for instance, the answer to the question of Figure 1.2 will be included in the box depicted in Figure 1.3.

We compared our model against SE-KGE and we show that for both datasets (YAGO2geo, DBGeo) SQABo outperforms SE-KGE.

1.5 Thesis Layout

The rest of the thesis is structured as follows. First, for better comprehension of the text, we introduce the main terms mentioned in this thesis in Section 2. Then, to explore the different techniques of embedding knowledge graphs, an overview of the related work is presented in Section 3. In Section 4, we present in detail the architecture of SQABo. In Section 5 we describe the pre-processing methods used to create the dataset, the training method of the model and finally, the evaluation of the model. The conclusions and future work are outlined in Section 6.

2. PRELIMINARIES

In this section we will describe and define all the prerequisite knowledge needed for this thesis such as Geospatial Knowledge Graphs, Query Answering with Knowledge Graph Embeddings and Geospatial Data Modeling.

2.1 Geospatial Data Modeling

A *geographic feature* (or simply feature), is an abstraction of a real world phenomenon and can have various attributes that describe its thematic and spatial characteristics [26]. Geographic features, are essentially components of a planet like mountains, canyons, lakes, streams and can be referred to as locations, sites, areas, or regions. Knowledge about the spatial attributes of a feature can be *quantitative* or *qualitative*. For example, the fact that the distance between Athens and Patras is 250 km is *quantitative knowledge*, while the fact that river Rhine crosses Germany and France and forms their border is *qualitative knowledge*. *Quantitative geographic knowledge* is usually represented using geometries (e.g., points, lines and polygons on the Cartesian plane) while *qualitative geographic knowledge* is captured by qualitative binary relations between the geometries of features [26].

The geometry of every place and object is composed of *points*. *Points* are coordinates in a 2-, 3- or 4-dimensional space. All points in a geometry have the same dimensionality.

A *bounding box* is the rectangle that contains every point from a set that describes a place or an object. A minimum bounding box is the box with the smallest measure (area, volume, or hypervolume in higher dimensions) within which all the points lie.

2.2 Knowledge Graphs

A *Knowledge Graph* $\mathcal{G}\langle\mathcal{V},\mathcal{R}\rangle$ is a directed edge and node labeled multigraph that represents knowledge about world objects (the nodes $\mathbf{v} \in \mathcal{V}$ of the graph) and relationships among these world objects (the edges $\mathbf{r} \in \mathcal{R}$ of the graph).

Some well-known open source KGs are the DBpedia, Wikidata, YAGO. DBpedia is a project aiming to extract structured content from the information created in the Wikipedia project. Wikidata is a collaboratively edited multilingual knowledge graph hosted by the Wikimedia Foundation and is used by Wikipedia to get its data. YAGO which is created by Max Plank Institute is open source and it is automatically extracted from Wikipedia and other sources.

2.2.1 Geospatial Knowledge Graphs

For the purpose of this thesis we define geospatial KGs as follows:

A *Geospatial Knowledge Graph* $\mathcal{G}\langle\mathcal{V}\bigcup\mathcal{V}_{\mathcal{G}},\mathcal{R}\rangle$ is a directed edge and node labeled multigraph where $\mathcal{V}$ is a set of entities/nodes, $\mathcal{V}_{\mathcal{G}} \neq \emptyset$ represents the set of *(geo)spatial entities* and $\mathcal{R}$ is the set of directed edges. Each entity $v_i \in \mathcal{V}_{\mathcal{G}}$ can be mapped to either a point $\mathbf{x} \in \mathcal{A} \subseteq \mathbb{R}^2$ or to a bounding box $[\mathbf{x};\mathbf{y}] \in \mathbb{R}^4$, where $\mathbf{x},\mathbf{y} \in \mathcal{A} \subseteq \mathbb{R}^2$ and $\mathcal{A}$ denotes the

bounding box containing all geographic entities in $\mathcal{G}$. $\mathcal{V}_{\mathcal{G}}$ is formed from the union of $\mathcal{V}_{\mathcal{T}}$ which is the set of the geospatial entities represented by a point and $\mathcal{V}_{\mathcal{B}}$ which is the set of geospatial entities represented by a polygon.

We also define $PT()$ which is the function that returns the coordinates of an entity if it is represented as a point and $PN()$ which is the uniform sampling function over the bounding box of an entity's polygon.

Although DBpedia and Wikidata have some geospatial knowledge, some other well known sources like YAGO2 and GeoNames gazeteer contain more specific geospatial data or are entirely dedicated to them. YAGO2 [12], the second version of YAGO, introduces geospatial and temporal information to the YAGO KG. Geopatial information in YAGO2 comes not only from Wikipedia but also from the gazeteer GeoNames. Temporal information is represented using dates as time points. GeoNames gazeteer is a freely available geographical database with 7M geographical names in various languages. As topological information it contains partonomic relations (e.g., Berlin is located in Germany is located in Europe) and neighboring countries for each location.

YAGO2geo [15] is an extention of YAGO2 [12] with geospatial information represented by geometries (e.g., lines, polygons, multipolygons, etc.) encoded by Open Geospatial Consortium standards. The added geospatial information comes from official sources such as the administrative divisions of countries but also from volunteered open data of OpenStreetMap. YAGO2geo contains the geometries of places and objects as well as the geospatial relationships between them inside the UK, Ireland, Greece and USA.

DBGeo [19] is a subgraph of DBpedia containing the mainland of United States which has relatively richer geographic coverage than other regions in DBpedia. It contains 176.671 triples with 25.980 entities and 227 geospatial and non-geospatial relations.

The most recent geospatial knowledge graphs are the KnowWhereGraph [14] and WorldKG [7]. KnowWhereGraph contains also data about extreme events, administrative boundaries, soils, crops, climate, transportation giving this way rapid access to information such as the wildfires that have affected an area, the major transportation axis crossing a certain region, and the type of crops and soils present in a given region. WorldKG data come from OpenStreetMap containing more than 100M geographic entities from 188 countries and more than 800M triples where the geographic objects are represented as points.

2.3 Graph Queries

Conjunctive Graph Queries (CGQ) [10] are a subclass of the first-order logical queries that use only existential ($\exists$) quantifier and conjunction ($\wedge$) logical operator. They are formally defined as follows:

$$q[V_?] = V_?.\exists V_1, \ldots, V_k : e_1 \wedge e_2 \wedge \cdots \wedge e_n,$$
$$\text{where } e_i = r(v_a, V), V \in \{V_?, V_1, ..., V_k\},\ v_a \in V,\ r \in R, \text{ or}$$
$$e_i = r(V, V'),\ V, V' \in \{V_?, V_1, ..., V_k\}, V \neq V', r \in R,$$

where v_a represents a constant, $V_1, \ldots V_k$ are existentially quantified variables, $e_1, \ldots, e_n$ are the directed edges and $V_?$ is the target variable (i.e., the queried node).

The *dependency graph* of a conjunctive query q, is a graphical representation query q, where the nodes can be either the variables or the anchor entities, and the edges are the relations appearing in q. According to [10], for a CGQ to be *valid* (i.e., there are no

contradictions or redundancies), its dependency graph must be a *directed acyclic graph (DAG)*, having the anchor node as the source node and the query target as the unique sink node.

2.4 Knowledge Graph Embeddings

KG embedding systems are systems which learn a low or high, depending on the task and the implementation, dimensional representation of a KG's entities and relations while preserving their semantic meaning. They can be used for various tasks such as link prediction, triple classification, recommender systems and enrichment based embeddings.

The link prediction task focuses on finding an entity that can be represented as a fact (edge) together with a given relation and entity i.e., (entity, relation, ?), where ? refers to the missing entity. Enrichment based embeddings are essentially contextualized embeddings created by enriching the entity's embedding using information from its neighbourhood. Triple classification is the problem of identifying whether a given triple is correct and give a yes or no answer. Finally, recommender systems assists the user in an environment where multiple options are available by providing a certain ordering of choices that the recommendation algorithm infers. This inference can be based on the similarity of the choices and behaviour pattern of different users.

2.4.1 Conjunctive Query Answering with Knowledge Graph Embeddings

Query Answering with knowledge graph embeddings is an evolution of simple edge prediction. Hamilton et al., [10] first introduced more complex DAG structures (Figure 1.1) sampled from the knowledge graph as conjunctive queries. He also created the architecture to answer them by adding a sophisticated intersection module to the existing link prediction methods. This module, which is, also, incorporated a simple attention mechanism [2], was used to combine the simple queries (answered with the classical link prediction techniques) from which the complex conjunctive one was made. At the end, the complex query was represented as a point in the vector space and the answer to it is the closest entity.

2.5 Summary

In this chapter we introduced the definitions of knowledge graphs, knowledge graph queries, geographical features, geographical points, and bounding boxes. We also presented some large, open-source geospatial and non-geospatial knowledge graphs. Finally, we introduced the notion of knowledge graph embeddings and how they can be used for query answering.

3. RELATED WORK

In this section, we initially present the related work in the domain of knowledge graph embeddings and the various categories of knowledge graph embedding models, and then we present in detail the models developed by Mai et al. [19] and Ren et al. [28] which are utilized by our model.

3.1 Semantic Matching Models

Semantic Matching Models use similarity-based functions to calculate the similarity between the different entities and relations. The first model using low-dimensional representations belonging also in the Semantic Matching Models category was RESCAL [25]. It followed the Statistical Relational Learning Approach which is based on a Tensor Factorization model that takes into account the inherent structure of relational data. Tensor Factorization is basically the expression of a Tensor as a sequence of elementary operations acting on other, often simpler Tensors. Using tensor factorization, which was a similar method to decomposition into directional components developed by Harshman et al. [11], RESCAL authors were able to derive better models. They achieved higher quality and significant runtime improvements over models using decomposition into directional components.

The main disadvantage of RESCAL is that it is a three-way model which performs fairly good for relationships which occur frequently but it performs poor for the rare relationships and leads to major over-fitting. The issue of major over-fitting for rare relationships can be controlled by regularizing or reducing the expressivity of the model. The second method for reducing the expressivity is Two-way interaction. Two-way interaction approaches overperform the Three-way approaches on many datasets and specifically on those that have more rare relationships. The problem with the two-way interaction is that they are limited and are not able to represent all kinds of relations with entities. TATEC [9] is a latent factor model which is capable of incorporating the high capacity Three-way model with well-controlled two-way interactions and take the advantage of both of them.

Another problem that we face on KG applications that predict missing relations or entities is that while dot product of vector embedding of KG triplets is being successfully used for symmetric, reflexive, anti-reflexive and even transitive relations, it can't be used for anti-symmetric relations. Complex [30] embedding facilitates joint learning of subject and object entities while preserving the asymmetry of the relation. It uses Hermitian dot product of embedding of subject entities and object entities. The eigen vector decomposition is used to identify a low rank diagonal matrix W which is later used to predict missing relations.

3.2 Translational Models

Later on, translational models like TransE [4] and TransH [33] appeared. They belonged in this category as they were using distance-based scoring functions to calculate the similarity between the different entities and relations. They were based on the concept of encoding the relations as geometric transformations between the head and the tail of a fact. To compute the embedding of the tail, it was necessary to apply a transformation to the head embedding. Then, the distance function was used to evaluate the embedding or

to score the reliability of the fact.

TransE was the first translational model proposed and specifically used a scoring function that forces the embeddings to satisfy a simple vector sum equation in each fact in which they appear ($head + relation = tail$). To learn the embeddings, minimization of ranking based loss function over the training set was used. Although its novelty at that time, this model fails in case of one-to-many and many-to-many relations.

To overcome this deficit a new model TransH was proposed. TransH was an evolution of TransE that introduced a hyperplane as geometric space to solve the problem of correctly representing the types of relations. This model also enabled an entity to have distributed representation based on their involvement in the relation. Although these systems made fundamental contributions, they lack the ability to generalize and, also, to encode geospatial data.

These approaches also fail to capture the semantic hierarchies. In HakE [36], the authors have proposed a method to model the hierarchy in the entities as concentric circles in polar coordinate. The entity with smaller radius belongs to higher level up in the hierarchy and the angle between them represents the variation in the meaning.

3.3 Deep Neural Network Models

The models TransGCN [5], R-GCN [29] and SE-KGE use deep neural networks to learn the embeddings of the KGs. Despite their less-efficient training, these models have the ability to generalise well and achieve good predictive performance, especially when they are pre-trained. TransGCN and R-GCN use convolutional neural networks, which convolve the input data, applying a low-dimensional filter capable of embedding complex structures with few parameters by learning nonlinear features and dealing with the high multi-relational data characteristic of realistic knowledge graphs. More specifically, Relational Graph Convolutional Network (R-GCN) uses a convolution-based entity encoder which maps each entity to a real-valued vector and a decoder which reconstructs the edges of the graph based on vertex representations. Unlike R-GCN in which entity embedding learning was done through a convolution-based encoder and relation embedding learning was in the decoder, TransGCN trains relation and entity embeddings simultaneously during graph convolution operation. It uses fewer parameters compared to R-GCN by using relation as transformation operator between head and tail entity in a triple.

Hamilton et al. [10] developed a method that goes beyond simple edge prediction and handles more complex logical queries, which might involve multiple unobserved edges, entities, and variables. This end-to-end logic query answering model can answer conjunctive graph queries. Wang et al. [32] used an entity-context-preserving translational embedding model which is specially designed for SPARQL queries and can compute approximate answers for SPARQL queries that return an empty set. It does so by leveraging the RDF embeddings and the translation mechanism. Mai et al. [20] expanded the system of Hamilton et al. by dealing with the variability of contributions from different query paths. To do so, they created a multi-headed contextual graph attention mechanism that incorporated into an end-to-end logical query answering model.

Another recent work on logical query embedding models is CQD [1]. CQD is an embedding-based KG completion model and trained it to create missing edges during inference and merge entity rankings using t-norms and t-conorms [17]. CQD uses beam search for inference and although having severe scalability issues, it has demonstrated strong capability

of generalizing from KG edges to arbitrary EPFO queries. The scalability issues are mostly due to scoring every entity for every atomic query.

Another very recently presented work from Mai et al. [22] shows promising results on embedding geospatial data (points, polygons etc.) using convolution neural networks like ResNet1D and NUFT (Non-Uniform Fourier Transformation). Using these methods, they are able to capture local and global structures of polygons, while at the same time achieving loop origin invariance, trivial vertex invariance, part permutation invariance and topology awareness. We plan to utilize this method in our future work.

3.4 The SE-KGE Model

All aforementioned KG embedding systems, before the work of Mai et al. [19], did not take into account triples containing objects with values of datatype properties like dates, texts, numbers, and geometries. Mai et al. [19] encode and use such information to achieve better representations employing the conjunctive graph query answering system of Hamilton et al. [10]. They, also, use their previous work [20], and train the whole model jointly on sampled query-answer pairs from the original KG. To use the system for geospatial query answering, the geospatial representation learning technique Space2Vec [21] was utilized. Space2Vec is a representation learning model which encodes the absolute positions and geospatial relationships of places inspired by biological grid cells [8]. The main idea of Space2Vec is to use sinusoidal functions with different frequencies to encode these positions. Another recently proposed location encoding module by the same authors, is Sphere2Vec [23]. This framework has a location encoding module which instead of mapping coordinates from manifolds, like spherical surfaces, to Euclidean spaces, it directly encodes spherical coordinates, preserving spherical distances. This method leads to increased performance in geospatial entity encoding as it overcomes the map projection distortion problem [34].

3.5 The Query2Box Model

Another recent work on KG-embedding-based query answering has been the Query2Box [28]. This architecture models the query more naturally by using box embeddings instead of point embeddings in the vector space. This way the query boxes enclose the sets of answer entities, and operations like intersection of query boxes, have also a geometric meaning as in Venn diagrams. Also, executing logical operators over boxes result in new boxes, which means that the operations are closed and thus, logical reasoning can be effectively performed by iteratively updating boxes according to the query computation graph. Boxes also solves the problematic part of how to effectively model a set of answer entities using a single point in the vector space as answer. Techniques like searching for the k closest entities (nearest neighbours) are not so clear as to the definition of the number k. Instead, using boxes, every embedded knowledge graph entity located inside the box is considered as part of the set of answer entities.

3.6 Summary

In this chapter we discuss related work about knowledge graph embedding systems as well as query answering using these embeddings. We also see that most recent models of knowledge graph embeddings have incorporated neural networks with more sophisticated architectures and modules, achieving better results than older algorithms.

4. THE SQABO MODEL

In this section we present the architecture of the model.

To provide an intuitive understanding of the funtionality of SQABo we illustrate in Figure 4.1 how the query of the running example (Figure 1.2) will be processed. Initially, the anchor nodes ($attica$, $athens$) are encoded using the entity encoder of $SE - KGE$. Each generated embedding is used as a box center embedding by the box decoder. The box decoder takes also as input the relations ($within\_inverse$, $north\_of$) and creates one box for each of the 2 branches. The branch that has as leaf $attica$ has one more relation so the previously generated box is now the input to the box decoder along with the relation ($near\_to$). The two resulting boxes (red and blue in Figure 1.3) are intersected to get the final answer box embedding (orange box).

Next, we will describe the architecture of SQABo model in detail. SQABo is composed of the following four modules:

- Entity encoder
- Box decoder
- Box center intersection
- Box offset intersection

Each one of these components is described below.

4.1 Entity Encoder

The task of an entity encoder is to create high dimension embeddings in order to represent each entity of the KG. This embedding is then fed into following neural network modules. It is a common practice for entity encoders to initialize an embedding matrix randomly and then train it in order to learn the correct representations through the neural network back-propagation. Each column of this matrix essentially indicates an embedding for a specific entity.

The *Entity Encoder* [19], $Enc()$, presented graphically in Figure 4.2, takes as input an entity, e_i, and outputs the embedding of this entity, $\mathbf{e}_i$ (i.e., $Enc(e_i) = \mathbf{e}_i$), based on the geospatial and non-geospatial (class) features of this entity. It consists of the entity feature encoder and the entity space encoder. The entity feature encoder consists of type-specific feature embedding matrices, which are learned over the training time. This part helps the model to learn more general information about each entity based on their type (e.g., country, city).

The entity space encoder utilizes Space2Vec [21] to enrich the final representation with the geospatial information of the entity. For entities that have point representations, the geographic input to the module is each entity's co-ordinations. If an entity is of a larger geographic extent, the input is a uniformly random point inside the bounding box of the entity. The intuition behind this is that over the training time the encoder will be called many times and will learn a uniform distribution over the entity's bounding box. After constructing the input, the space representation will pass through a feed-forward neural network to get the geospatial embedding. We see the space is calculated as follows:

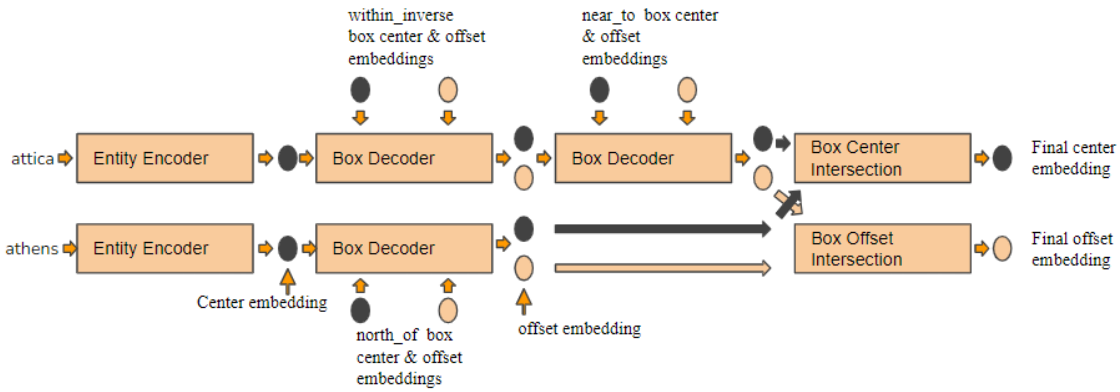

Figure 4.1: Processing of the query $Q(x) = \exists y :$ $north\_of(athens, x) \wedge near\_To(y, x) \wedge within\_inverse(attica, y)$ by SQABo.

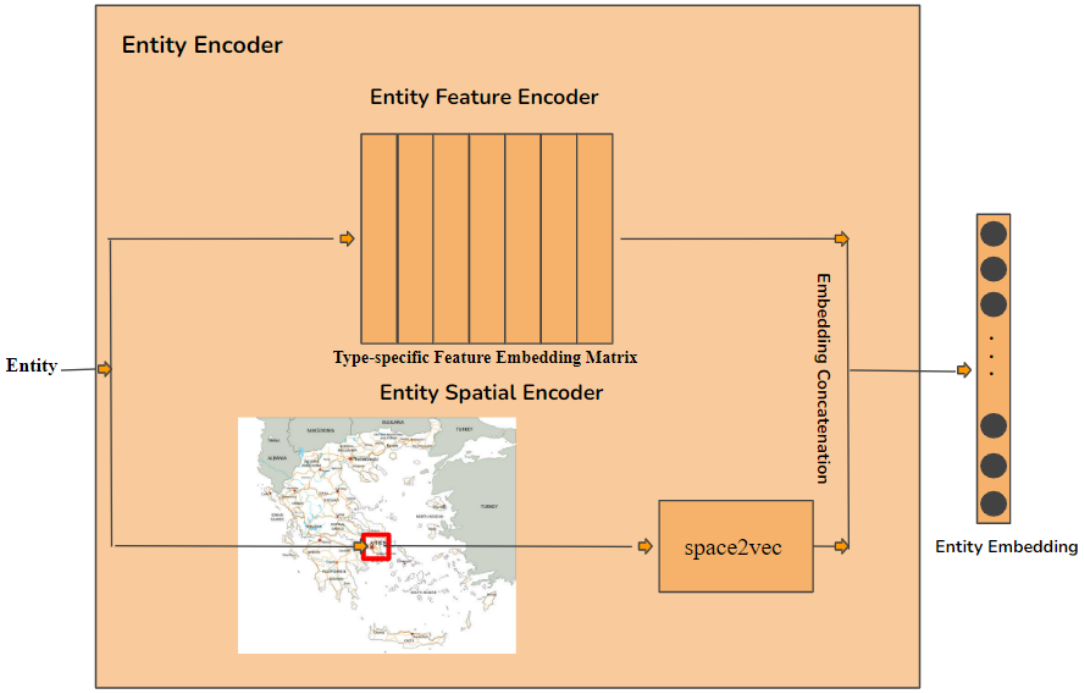

Figure 4.2: The Entity Encoder as developed by Mai et al.[19]

$$\mathbf{e}_i^s = Enc^s(\mathbf{x}_i) = \begin{cases} NN(Space2Vec(\mathbf{x}_i)), \ \textit{where } \mathbf{x}_i = PT(e_i), & \textit{if } e_i \in \mathbf{V}_T \\ NN(Space2Vec(\mathbf{x}_i)), \ \textit{where } \mathbf{x}_i = PN(e_i) = U(x_i^{min}, x_i^{max}), & \textit{if } e_i \in \mathbf{V}_B \end{cases}$$

where $\mathbf{e}_i^s$ is the final space embedding, $\mathbf{x}_i$ the point coordinates and x_i^{min}, x_i^{max} the corners of the bounding box of the entity's polygon. $NN()$ is the feed forward neural network and $Space2Vec()$ the Space2Vec system.

In the end, the feature embedding ($\mathbf{e}_i^f$) and the geospatial embedding ($\mathbf{e}_i^s$) will be concatenated ($[\mathbf{e}_i^f; \mathbf{e}_i^s]$) resulting in the entity embedding ($\mathbf{e}_i$) which will have both general and geospatial information embedded and will be passed to the next module.

$$\mathbf{e}_i = Enc(e_i) = [Enc^f(e_i); Enc^s(e_i)] = [\mathbf{e}_i^f; \mathbf{e}_i^s]$$

4.2 Box Decoder

In most cases of KG embedding systems, a projection operator is utilized in order to predict the embedding of the answer given a query (one or more entities and relations).

The second module, the Box Decoder (Figure 4.3), goes a step beyond simple edge prediction and incorporates the idea of box embeddings, proposed by Ren et al. [28]. The intuition behind the box embeddings' approach is that instead of dealing with points in the vector space (simple embeddings), to embed the queries as boxes and get the answers as the entities (vector points) which are inside the box. For this purpose, we use a center embedding (of the box of the query) and and an offset (from the center of the box) embedding.

The role of the Box Decoder is to create, move and enlarge the query box embedding by changing the center and offset embeddings. For this purpose, the Box Decoder takes as input the embedding of the non-variable anchor node v_a (for simplicity consider for now that there is only a single anchor node in the query) of the DAG query $\mathcal{Q}$ (where $\mathcal{Q} = r_1(v_a, v_1) \wedge r_2(v_2, v_3) \ldots \wedge r_m(v_m, v_{m+1})$) from the Entity Encoder, and considers it as the center of the initial box with zero offset. It continues by encoding the relation r_1 with which the anchor node is connected to, and by combining the embeddings with the previous zero-size initial boxes, it creates actual boxes, i.e. with offset > 0. Continuing to the next relation, r_2, in the path, the box decoder takes as input the box previously created, and combines it with the embedding of r_2. This process repeats until all relations in the path from the non-variable anchor node to the target node are processed.

The trainable matrices Relation Feature, $\mathbf{r}^f$, and Geospatial Embedding, $\mathbf{r}^s$, are used to encode each relation and focus on the feature and geospatial entity embedding, respectively. The concatenation of these matrices ($[\mathbf{r}^f, \mathbf{r}^s]$) is the Relation Box Center Embedding. Also, the trainable Relation Box Offset Embedding matrix is used, which when trained, it represents the correct size of the box (i.e., the distance from the center).

Formally, let $\mathcal{B} = \langle e, r_1, r_2, \ldots, r_n \rangle$ be a branch of a DAG query $\mathcal{Q}$. We operate on $\mathbb{R}^d$, and define the decoding function $Dec() : \mathcal{V}_\mathcal{G} \times \mathcal{R} \longrightarrow \mathbb{R}^{2d}$, which computes the box embedding of $\mathcal{B}$, as $\mathbf{p} = (Cen(\mathbf{p}), Off(\mathbf{p})) \in \mathbb{R}^{2d}$. Initially, the box embedding is calculated as follows:

$$Cen(\mathbf{p}_1) = Enc(e)$$
$$Off(\mathbf{p}_1) = \mathbf{0}$$

where $Enc(e) \in \mathbb{R}^d$ is calculated by the Entity Encoder, $Enc()$, and $\mathbf{0}$ is a d-dimensional all-zero vector. Then, given an input box embedding $\mathbf{p}_i$, the center $Cen(\mathbf{p}_{i+1})$, and offset $Off(\mathbf{p}_{i+1})$ of the new box $\mathbf{p}_{i+1}$, generated by the projection of $\mathbf{p}_i$ onto r_i, are defined as:

$$Cen(\mathbf{p}_{i+1}) = Cen(\mathbf{p}_i) + Cen(\mathbf{r}_i)$$
$$Off(\mathbf{p}_{i+1}) = Off(\mathbf{p}_i) + sigmoid(Off(\mathbf{r}_i))$$

where $\mathbf{r}_i = (Cen(\mathbf{r}_i), Off(\mathbf{r}_i))$, with:

$$Cen(\mathbf{r}_i) = [\mathbf{r}_i{}^f; \mathbf{r}_i{}^s] \tag{4.1}$$

and $Off(\mathbf{r}_i)$ is randomly initialized.

If the query contains multiple anchor nodes (i.e., multiple branches), then the Box Decoder generates multiple boxes. For this case, as it is described below, the answer is retrieved by intersecting these boxes.

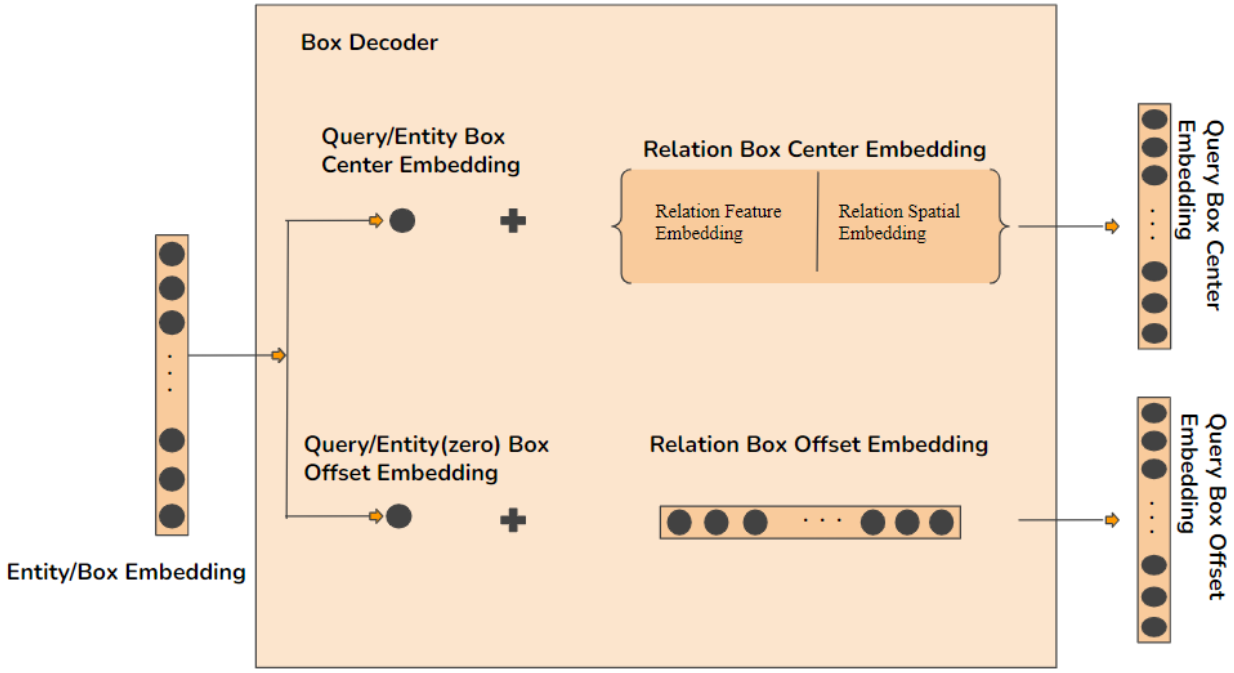

Figure 4.3: The Box Decoder of SQABo. Compared to [28], the relation space encoder component (Relation Spatial Embedding) is added to capture the geospatial relations and sigmoid function is used for the calculation of the offsets.

4.3 Box Center Intersection Attention

As we can have multiple branches in a query (i.e. 3-inter or 3-inter-chain DAG structure) and each branch will produce a box with a center and an offset, multiple boxes will occur. These box centers and eventually boxes will have to be combined to get a final answer box. To combine the n different output box embeddings from the Box Decoder to a single final answer box embedding, the *Box Center Intersection Attention module* is used to create the final box center and the *Box Offset Intersection module* to create the final box offset.

A very simple way to combine these box centers would be to take the mean of the centers but this would assume that each branch has an equal contribution to the final intersection embedding which is not necessarily the case in real settings. Ren et al. [28] showed that using a simple Attention mechanism [2] outperforms other techniques like average and DEEPSETS [35]. More complex and graph-oriented attention mechanisms like Graph Attention Networks [31] has shown that using an attention mechanism on graph-structured data also gives better results than other simpler approaches.

Mai et al. [20], following the idea of Graph Attention Networks, proposed an attention-based geometric intersection operator which uses multi-head self attention layer. This method represents the logical conjunction in the embedding space and it has shown better results than GQE [10] which used an element-wise mean or minimum approach. The novelty here was that instead of using the original multi-headed self attention, used an entity-type-specific trainable attention vector for each attention head in place of the original attention vector.

The Box Center Intersection Attention module applies the graph self-attention mechanism (Figure 4.4) introduced in CGA model [20]. It is also implemented by using a multi-head attention layer and a feed-forward neural network layer having normalization layers in between. In the end, using this method, the attention-weighted box center embedding is

computed as the weighted average of different input box center embeddings, while the weights are automatically learned by the multi-head attention mechanism. This leads to better representations incorporating the uneven participation of the neighboring box center embeddings to the final box center embedding.

Formally, if $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(n)}$ is the set of all branches appearing in $\mathcal{Q}$, and $Dec(\mathcal{B}^{(i)}) = (Cen(\mathbf{p}^{(i)}), Off(\mathbf{p}^{(i)}))$ the embedded box for branch $\mathcal{B}^{(i)}$, then the center of $\mathcal{Q}$ is calculated as follows:

$$Cen(\mathcal{Q}) = CGA(Cen(\mathbf{p}^{(1)}), \ldots, Cen(\mathbf{p}^{(n)})) = LayerNorm_2(W_\gamma e_{ln1} + B_\gamma + e_{ln1})$$

where $LayerNorm_2()$ is a normalization layer and $W_\gamma \in \mathbb{R}^{dxd}$ and $B_\gamma \in \mathbb{R}^d$ are trainable entity type γ specific weight matrix and bias vector, respectively, in a feed forward neural network. e_{ln1} is defined as :

$$e_{ln1} = LayerNorm_1(e_{attn} + e_{init})$$

where $LayerNorm_1()$ is a normalization layer, e_{init} is a permutation invariant transformation of the initial box center embeddings and

$$e_{attn} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{n} a_{ik} e_i)$$

where K is the number of attention heads, $\sigma()$ is the sigmoid activation function and n is the number of all the answer box centers to be intersected. a_{ik} is defined as:

$$a_{ik} = \frac{exp(LeakyReLU(a_{\gamma k}^T [e_{init}; e_i]))}{\sum_{j=1}^{n} exp(LeakyReLU(a_{\gamma k} [e_{init}; e_j]))}$$

where $a_{\gamma k} \in \mathbb{R}^{dx2}$ is the γ-type-specific trainable attention vector for the k^{th} attention head.

4.4 Box Offset Intersection

The second module is based on the offset intersection operator proposed by Ren et al. [28] and works as a box shrinking mechanism to get the intersection of the boxes. This module uses the permutation-invariant deep architecture Deepsets of Zaheer et al. [35]. Deepsets is modeled as:

$$DeepSets(x_1, \ldots, x_N) = MLP((\frac{1}{N}) \cdot \sum_{i=1}^{N} MLP(x_i))$$

where $MLP()$ is the Multi-Layer Perceptron. The final intersection offset is given by

$$Off(p_1, \ldots, p_n) = Min(Off(p_1), \ldots, Off(p_n)) \cdot \sigma(DeepSets(p_1, \ldots, p_n))$$

where the input are the offsets of the boxes generated for each branch of $\mathcal{Q}$.

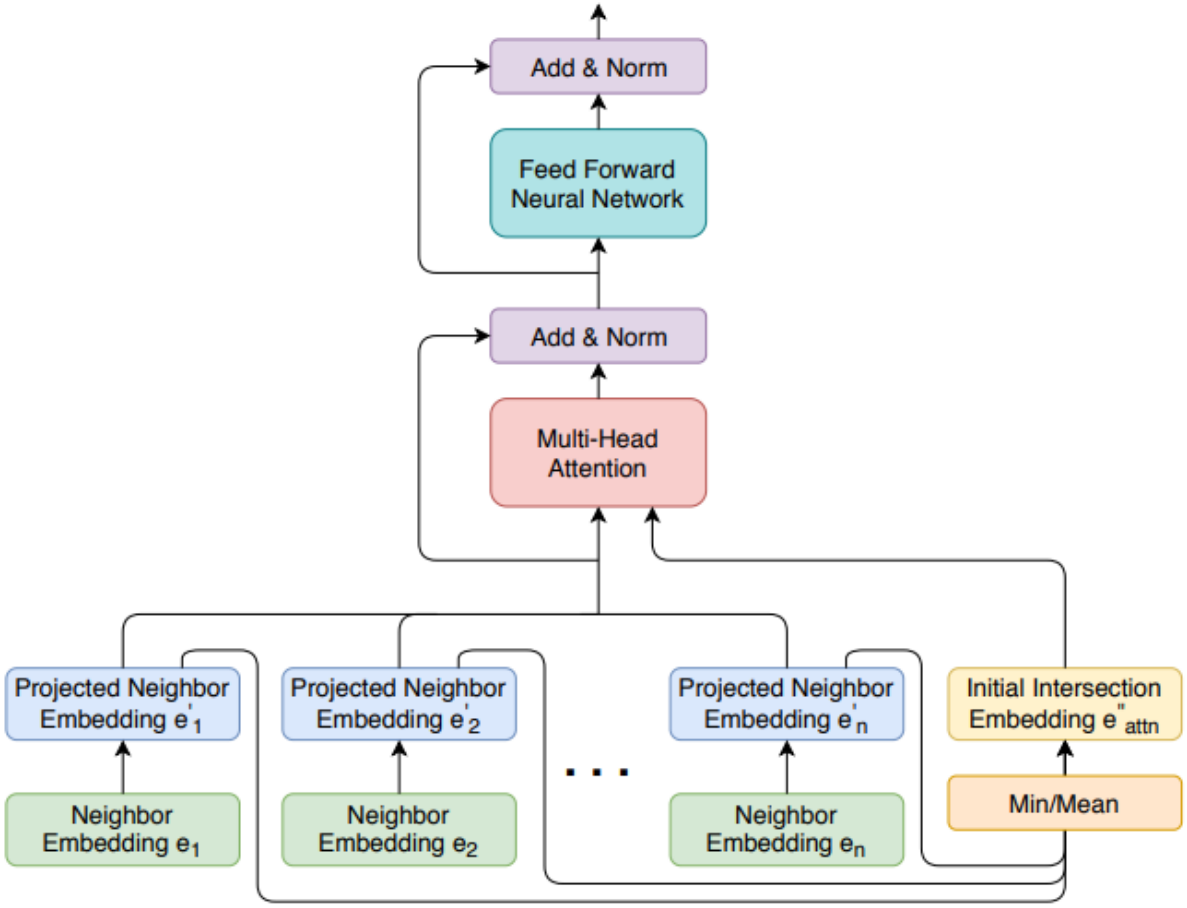

Figure 4.4: The Contextual Graph Attention [20] mechanism used from Box Center Intersection module

4.5 Summary

In this chapter we introduced the architecture of SQABo. We described the entity encoder, which is the same as the one developed for SE-KGE [19] and encodes the geospatial entities. We analyzed the Box Decoder which works as a projection operator and the Intersection modules that combine the different branches of the queries. In order to achieve better results we incorporated the Contextual Graph Attention [20] mechanism.

5. EXPERIMENTS

SQABo[1] is trained in two KGs: DBGeo and YAGO2geo. The statistics of both KGs and the respectively generated QA datasets are presented in Table 5.1.

	YAGO2geo	DBGeo$_{\mathcal{G}}$
Triples	17,353,031	176,671
Relations	9	227
Entities	772,143	25,980
Queries	1,000,000	1,000,000
Places	UK / Ireland / Greece	United States (DBpedia)

Table 5.1: Statistics of Knowledge Graphs used in SQABo

5.1 Preprocessing

DBGeo did not require any preprocessing, as it was already used for the evaluation of the SE-KGE model, and SQABo takes the input in exactly the same form with SE-KGE.

YAGO2geo data is in the form of RDF triples, representing the properties of each entity (id, type, geometry, label, population, etc) and the relations between these entities. From this data we produced mappings of entities, relations, and geometries to ids, the reverses of these files (ids to entities) and custom structures represented by their entities' ids and classes. The geometries of the entities in YAGO2geo are given in points and polygons. To encode the entities described by polygons we created the respective bounding boxes, so as to decrease the computational complexity and follow the entity encoder architecture of SE-KGE [19].

For the generation of the training and validation queries, we selected 10% of the edges uniformly random and removed them from the graph and then we performed sampling on this down-sampled training graph, taking n samples of the 10 different DAG structures presented in Figure 1.1. To make the test queries, we sampled them from the original graph, but we ensured that the test query samples are not directly answerable in the training graph. This means, that at least the answer nodes of these testing sub-graphs (queries) should not be part of the training graph and, therefore, the model should not have used them in the training phase. This method for the generation of the dataset was first introduced by [10] and, then, used by [19] for the generation of DBGeo QA dataset. The QA dataset for YAGO2geo is openly available.[2]

5.2 Model Training

The training of the model is supervised, as we sample the query-answer pairs from the graph. In the training phase we sample n conjunctive query-answer pairs, m for each DAG structure (i.e., $n = 7 * m$) and k negative answers for each query. An example of a sampled query is the one presented in Figure 1.2.

[1] https://github.com/markos-iliakis/GeospatialKGEmbeddings

[2] https://github.com/markos-iliakis/GeospatialKGEmbeddings/tree/master/Data

The objective is to make the correct answer entity embedding, $\mathbf{v} \in \mathbb{R}^d$, be inside the query box, $\mathbf{q} \in \mathbb{R}^{2d}$, generated by SQABo and, in particular to be the closest one to the predicted query box center embedding. The negative answers should be away from the box. To measure the distance between **v** and **q**, we follow the approach of Ren et al. [28], which is briefly described below.

The answer entity distance from the predicted query box embedding ($(Cen(\mathbf{q}), Off(\mathbf{q}))$) is measured by adding the L1 distance $\|Cen(\mathbf{q}) - \mathbf{v}\|_1$, of the entity to the perimeter of the box and the box offset embedding. If the answer entity is inside the box then the distance is only calculated by the L1 distance of the entity from the box center embedding. The calculation of the model loss is based on Query2Box [28]:

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

where γ represents a fixed scalar margin, v is an answer to the query q (positive entity), v' is the i-th negative entity (non-answer to the query q), and k is the number of negative entities. $dist_{box}$ is the distance of the answer entity from the predicted query box and is calculated as:

$$dist_{box}(\mathbf{v}; \mathbf{q}) = dist_{outside}(\mathbf{v}; \mathbf{q}) + a * dist_{inside}(\mathbf{v}; \mathbf{q})$$

where a is a fixed scalar and

$$dist_{outside}(\mathbf{v}; \mathbf{q}) = \|Max(\mathbf{v} - \mathbf{q}_{max}, \mathbf{0}) + Max(\mathbf{q}_{min} - \mathbf{v}, \mathbf{0})\|_1$$
$$dist_{inside}(\mathbf{v}; \mathbf{q}) = \|Cen(\mathbf{q}) - Min(\mathbf{q}_{max}, Max(\mathbf{q}_{min}, \mathbf{v}))\|_1$$

and

$$\mathbf{q}_{max} = Cen(\mathbf{q}) + Off(\mathbf{q})$$
$$\mathbf{q}_{min} = Cen(\mathbf{q}) - Off(\mathbf{q})$$

5.3 Evaluation Results

To measure the performance of $SQABo$, i.e., how representative are the final embeddings of the target nodes (answers), we use Average Percentile Rank (APR). The percentile rank of a given score is the percentage of scores in its frequency distribution that are less than that score. APR in our model is calculated for each query by getting the average percentile rank of the correct answer among all negative answers based on the prediction of the model.

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

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

Parameter	Value
Entity Encoder Feature Embedding dimension	128
Entity Encoder Spatial Embedding dimension	128
Feed Forward Hidden size	512
Feed Forward Dropout	0.5
Feed Forward skip connections	True
Space2Vec Max radius	5400000
Space2Vec Min radius	50
Space2Vec frequency	16
Graph Attention heads	2

Table 5.2: Best Hyperparameters for each module of the architecture

Due to the fact that APR uses all negative samples for each query, as opposed to the AUC (Area Under Roc Curve) which uses only one negative sample per query, as an evaluation metric, it is more robust.

The hyperparameters that returned the optimal results are presented in Table 5.2. We implemented all models in PyTorch and trained/evaluated each model on a Ubuntu machine with 1 GeForce GTX 1080 Nvidia GPU core, which has 10GB memory.

The evaluation results of SQABo against SE-KGE are presented in Table B.1. We compare the two models with two experiments per model plus one for the original SE-KGE with all of its data as a reference point. Also, in order to test the contribution of Contextual Graph Attention we run an experiment using only simple attention from Bahdanau et al., [3]. For the first experiment, we used the geospatial fragment of the DBGeo, DBGeo$_{\mathcal{G}}$, i.e. the fragment of DBGeo KG that represents knowledge only about geospatial entities and the subset of DBGeo QA dataset that is only about geospatial entities. For every single query structure, except for 2-inter (where SE-KGE oupterformed SQABo only by 1.15%), SQABo demonstrates better APR, resulting in an APR score difference of 4.5%, when macro averaged. For the second experiment, we used YAGO2Geo. The results in this dataset were even better, with SQABo being better in every query structure and having a macro averaged APR score difference of 5.6%. Lastly, the model without the contextual graph attention still performed better than SE-KGE with an average APR of 89.29 but worse than the model with contextual graph attention showing the importance of its use.

It is worth noting that, for 3-chain queries over DBGeo$_{\mathcal{G}}$, SQABo outperformed SE-KGE by 13.46%. While the maximum difference between the two models for YAGO2geo KG was in 3-inter queries, by 8.72%.

5.4 Summary

In this chapter we compared our model with SE-KGE using the Average Percentile Rank as a metric. We showed that SQABo achieved better results on both DBGeo (geospatial part) and YAGO2geo. We also replaced the contextual graph attention mechanism with simple attention and showed its important contribution.

6. CONCLUSIONS-FUTURE WORK

In this research work, we present the novel geospatial query answering model SQABo. SQABo encodes the geospatial and non-geospatial features of the entities and relations appearing in incoming conjunctive graph queries. Then, these encodings are gradually projected into boxes in the vector space. The answer to an input conjunctive graph query is computed by intersecting these boxes using contextual graph attention and returning the entities inside the boxes. Experimental results on the two geospatial KGs YAGO2geo and DBGeo, demonstrate that SQABo outperforms the existing relevant work.

As a future research work, we plan to increase the accuracy of our results by employing geospatial encoding techniques that appeared very recently in the literature [22]. These techniques capture local and global structures of polygons and not only of bounding boxes. We plan to extent the expressivity of the queries that SQABo can support, as in [28], by exploiting the special feature of boxes, that are essentially the Venn Diagrams in vector space. Another plan is to replace the bounding boxes and the polygons with spherical data. Using boxes rises the problem of projection distortion due to the mapping of real-world gps data (spherical surface) to 2D euclidean surfaces. A very recent and promising work (Sphere2Vec [24]) solves this problem by encoding the point coordinates on a spherical surface. This idea could also be incorporated in our system. Finally, we plan to make more comparisons with traditional systems such as Strabo 2 in order to determine any computation time advantages as well as the advantages on incomplete data.

APPENDIX A. ACRONYMS

AUC	Area Under ROC Curve
APR	Average Percentile Rank
CGA	Contextual Graph Attention
CF	Cumulative Frequency
DAG	Directed Acyclic Graph
GQE	Graph Query Embedding
KG	Knowledge Graph
NUFT	Non-Uniform Fourier Transform
SQABo	geoSpatial Query Answering using Boxes
RDF	Resource Description Framework
SE-KGE	Spatially-Explicit Knowledge Graph Embedding
TATEC	Two And Three-way Embeddings Combination
TransGCN	Translational Graph Convolutional Network
R-GCN	Relational Graph Convolutional Network
ResNet	Residual Network

APPENDIX B. RESULTS TABLE

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

Table B.1: Results (APR score) of SE-KGE model versus SQABo model evaluated on DBGeo and YAGO2geo

10pt 10pt

BIBLIOGRAPHY

[1] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *International Conference on Learning Representations*, 2021.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[5] Ling Cai, Bo Yan, Gengchen Mai, Krzysztof Janowicz, and Rui Zhu. Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 131–138, 2019.

[6] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. WorldKG: A world-scale geographic knowledge graph. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 4475–4484. ACM, 2021.

[7] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. WorldKG: A world-scale geographic knowledge graph. In *Proceedings of the 30th ACM International Conference on Information &amp Knowledge Management*. ACM, oct 2021.

[8] Ruiqi Gao, Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Learning grid cells as vector representation of self-position coupled with matrix representation of self-motion, 2018.

[9] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Effective blending of two and three-way interactions for modeling multi-relational data. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 434–449, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[10] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[11] Richard A Harshman. Models for analysis of asymmetrical relationships among n objects or stimuli. In *First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton, Ontario, 1978*, 1978.

[12] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[13] Krzysztof Janowicz, Pascal Hitzler, Wenwen Li, Dean Rehberger, Mark Schildhauer, Rui Zhu, Cogan Shimizu, Colby K. Fisher, Ling Cai, Gengchen Mai, Joseph Zalewski, Lu Zhou, Shirly Stephen, Seila Gonzalez Estrecha, Bryce D. Mecum, Anna Lopez-Carr, Andrew Schroeder, Dave Smith, Dawn J. Wright, Sizhe Wang, Yuanyuan Tian, Zilong Liu, Meilin Shi, Anthony D'Onofrio, Zhining Gu, and Kitty Currier. Know, know where, knowwheregraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Mag.*, 43(1):30–39, 2022.

[14] Krzysztof Janowicz, Pascal Hitzler, Wenwen Li, Dean Rehberger, Mark Schildhauer, Rui Zhu, Cogan Shimizu, Colby K. Fisher, Ling Cai, Gengchen Mai, Joseph Zalewski, Lu Zhou, Shirly Stephen, Seila Gonzalez, Bryce Mecum, Anna Lopez-Carr, Andrew Schroeder, David Smith, Dawn Wright, Sizhe Wang, Yuanyuan Tian, Zilong Liu, Meilin Shi, Anthony D'onofrio, Zhining Gu, and Kitty Currier. Know, know where, know where graph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Magazine*, 43(1):30–39, March 2022. Funding Information: The authors acknowledge support by the National Science Foundation under Grant 2033521 A1: KnowWhere-Graph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Publisher Copyright: © 2022 The Authors.

[15] Nikolaos Karalis, Georgios Mandilaras, and Manolis Koubarakis. Extending the yago2 knowledge graph with precise geospatial knowledge. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, pages 181–197, Cham, 2019. Springer International Publishing.

[16] Nikolaos Karalis, Georgios M. Mandilaras, and Manolis Koubarakis. Extending the YAGO2 knowledge graph with precise geospatial knowledge. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2019.

[17] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular norms*. Kluwer Academic Publishers, 2000.

[18] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: A semantic geospatial DBMS. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, volume 7649 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2012.

[19] Gengchen Mai, Krzysztof Janowicz, Ling Cai, Rui Zhu, Blake Regalia, Bo Yan, Meilin Shi, and Ni Lao. Se-kge: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. *Transactions in GIS*, 24(3):623–655, 2020.

[20] Gengchen Mai, Krzysztof Janowicz, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. Contextual graph attention for answering logical queries over incomplete knowledge graphs. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 171–178, 2019.

[21] Gengchen Mai, Krzysztof Janowicz, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. Multi-scale representation learning for spatial feature distributions using grid cells. In *The Eighth International Conference on Learning Representations*. openreview, 2020.

[22] Gengchen Mai, Chiyu Jiang, Weiwei Sun, Rui Zhu, Yao Xuan, Ling Cai, Krzysztof Janowicz, Stefano Ermon, and Ni Lao. Towards general-purpose representation learning of polygonal geometries, 2022.

[23] Gengchen Mai, Yao Xuan, Wenyun Zuo, Yutong He, Stefano Ermon, Jiaming Song, Krzysztof Janowicz, and Ni Lao. Sphere2vec: Self-supervised location representation learning on spherical surfaces, 2022.

[24] Gengchen Mai, Yao Xuan, Wenyun Zuo, Krzysztof Janowicz, and Ni Lao. Sphere2vec: Multi-scale representation learning over a spherical surface for geospatial predictions, 2022.

[25] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280, 2012.

[26] Dharmen Punjani, Markos Iliakis, Theodoros Stefou, Kuldeep Singh, Andreas Both, Manolis Koubarakis, Iosif Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, Theofilos Ioannidis, Nikolaos Karalis, Christoph Lange, Despina-Athanasia Pantazi, Christos Papaloukas, and Georgios Stamoulis. Template-based question answering over linked geospatial data, 2020.

[27] Blake Regalia, Krzysztof Janowicz, and Grant McKenzie. Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data. *Transactions in GIS*, 23(3):601–619, 2019.

[28] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. 2020.

[29] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[30] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction, 2016.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[32] Meng Wang, Ruijie Wang, Jun Liu, Yihe Chen, Lei Zhang, and Guilin Qi. Towards empty answers in sparql: approximating querying with rdf embedding. In *International semantic web conference*, pages 513–529. Springer, 2018.

[33] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[34] David L. Williamson and Gerald L. Browning. Comparison of Grids and Difference Approximations for Numerical Weather Prediction Over a Sphere. *Journal of Applied Meteorology*, 12(2):264–274, March 1973.

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2017.

[36] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction, 2019.