Utrecht University

Department of Information and Computing Science

Automated annotation of GIS workflows using knowledge graph embedding (KGE)

Author: Wiktoria Libera

Supervisor: Haiqi Xu

First examiner: Dr. Simon Scheider

Second Examiner: Dr. Judith Verstegen

Abstract

To solve the “indirect” question-answering problem, the core concepts of spatial information, which distinguish geo-semantics, are used to interpret both questions and workflows (answers). Geographical Information Systems (GIS) processes occurring in these workflows can then be annotated by the core concept data (CCD) types which combine data forms with their semantics. In QuAnGIS the annotations are currently done manually, however, it is a complex and time-consuming process. In this project, we check ‘*How do Knowledge Graph Embeddings (KGE) models help with automated annotation of GIS workflows?*’. We test RESCAL and ConvE models and how they behave with geo-specific data. The aim is to check if automatic annotation with use of KGE models is even possible, and if so, what method would be the best. Some of the results were positive in terms of tail prediction and evaluation metrics. Despite that, the automatic annotation is rather challenging, with the current state of the data used in QuAnGIS project.

1. Introduction

Geo-analytical tasks rely on interpretation, analysis and understanding of geospatial data to get meaningful insight. Those tasks analyse spatial patterns by implementing analytical techniques. One of the ways to present the process is with the use of workflows, that shows the subsequent steps and helps facilitate geo-analytical tasks. Therefore, their composition, as well as their retrieval, are important in this domain. For example, automatic workflow composition enables the delivery of a framework that can be used immediately to generate a geo-analytical task description that can then be executed (Kruiger et al., 2020). Moreover, in GIS workflows are used as answers for geo-analytical questions. For simple geographical questions like "Where is Amsterdam?" and "How deep is the Mariana Trench?" users can easily find answers in search engines (Xu et al., 2020) . The system captures them with the use of natural language interfaces and transforms them into queries to find the appropriate answer (Scheider et al., 2020). The information retrieved is based on factoid knowledge, such as existing documents and knowledge bases (Xu, Nyamsuren, Scheider, et al., 2022; Scheider et al., 2020). However, the situation becomes more difficult when the user wants to know the answer to previously mentioned geo-analytical questions. An example of such a problem can be: "Which neighbourhoods are within 100m from school in Amsterdam?". According to Scheider, Nyamsuren, et al. (2020) that can be defined as an ‘indirect’ question, as the answer to it is not straightforward and the queries cannot simply filter out the necessary information. In order to get the answer, we need to first gain an understanding of the situation and the intentions behind it (Xu et al., 2020). This usually calls for Geographical Information System (GIS) experts as well as proper input data (Xu, Nyamsuren, & Scheider, 2022; Xu, Nyamsuren, Scheider, et al., 2022). The process requires the transformation of the question and the use of geo-analytical tools (Scheider et al., 2020). This is when the workflow is applied to show the analytical answer (Xu et al., 2020). An easy way to understand this process is to look at the steps required to answer the question about Amsterdam's neighbourhoods in figure 1.

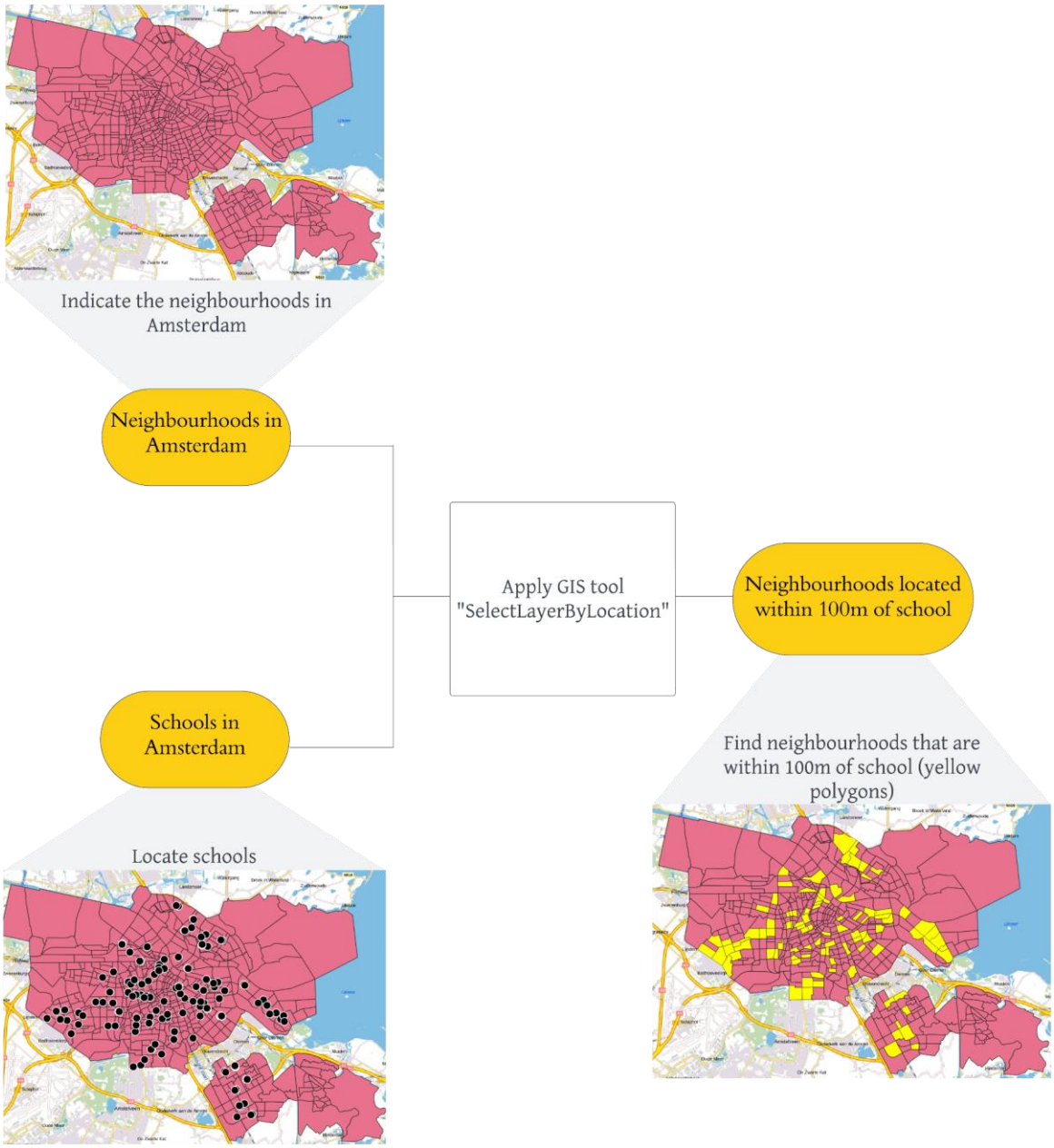

Figure 1. Steps needed for answering the question "Which neighbourhoods are within 100m from school in Amsterdam?"

Figure 1 shows that the question consists of 3 other separate tasks that constrain the solution. In order to produce a meaningful answer, we have to approach them in the correct order and use proper analytical tools (Xu, Nyamsuren, Scheider, et al., 2022). Therefore, the first step is to determine the neighbourhoods within Amsterdam. Then the schools within that city are identified. Lastly, by applying the GIS tool "SelectLayerByLocation" the neighbourhoods that have schools within 100m can be established (yellow polygons).

The example above demonstrates that geo-analytical question answering (QA) is far more complex (Scheider et al., 2018) than simply retrieving facts (Xu et al., 2020). There are numerous difficulties that we can encounter. To begin with, the questions are more flexible and

“indirect", which leads to their semantic ambiguity (Scheider et al., 2020). That also means that they have more diverse GIS objectives that can be achieved in multiple ways. Additionally, there are different semantic interpretations of words, for example, "distance" can mean "how far?" same as "how close?" (Scheider, Nyamsuren, et al., 2020). That causes multiple interpretations and answers that are equally valid. This is, however, strictly dependent on the expert's understanding and the tools they work with. Last, but not least, the process of geo-analytical QA consists of multiple steps. As previously stated, the question needs to be first interpreted and divided into sub-questions, and then the expert needs to recognise the potential of the data that is available, and finally apply GIS theory and tools to transform the data for the solution tools (Scheider, Nyamsuren, et al., 2020).

Geo-analytical QA is not as common with artificial intelligence, due to the challenges described above. Nonetheless, such an invention would be a beneficial addition for scientists in different domains and non-GIS users as they would be able to investigate those problems. Currently, some work is being done on QuAnGIS- an “indirect” question-answering system that attempts to fill that gap. It intends to generate workflows that indicate procedures to be used in GIS software to answer questions. The core concepts of spatial information proposed by Werner Kuhn (2012), as a conceptual and computational interface to GIS, play an important role in that system. The common concepts in geo-analytical tasks are field (quality surface measurable everywhere on a metric space), object (spatially bounded discrete entity with qualities), event (a temporally bounded discrete entity with qualities) and network (quantified relation between objects). Questions are annotated using those core concepts. Coming back to the example of Amsterdam, that would be “Which neighbourhoods (object) are within 100m (field) from school (object) in Amsterdam?” When it comes to the answer, the annotation of the workflow is done with CCD types. That can be visible in figure 2. Both of the annotations are necessary in order to match them together.

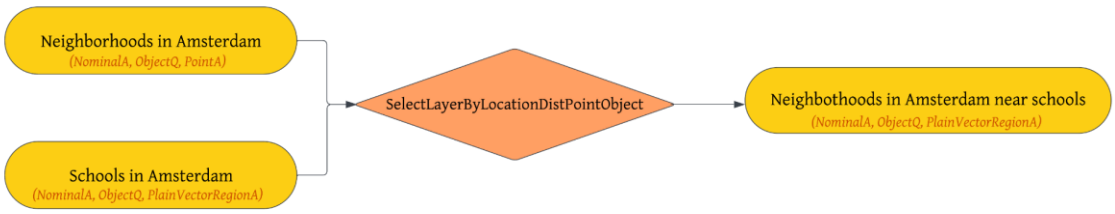

Figure 2. Annotated workflow with core concepts data types for question ““Which neighbourhoods are within 100m from school in Amsterdam?”

Annotation with CCD types significantly improves the precision of workflow synthesis and captures important distinctions necessary for applying geospatial analysis tools (Kruiger et al., 2020). However, in QuAnGIS, this process is currently done manually. Because it takes a long time, it is inefficient if the system is intended to work on a larger scale. Therefore, machine learning and deep learning techniques can be utilised to automate this process. Since a GIS workflow can be considered as a graph consisting of data as nodes and tools as edges, using Knowledge Graphs Embedding (KGE) models could be one option. This method is based on Knowledge Graphs (KG), which are multi-relational graphs with two components: entities that

represent nodes, and relations which are various types of edges (Q. Wang et al., 2017). Each graph is represented as a triple (head entity (h), relation (r), tail entity (t)). KGE models project those KG elements into low-dimensional vector space with preserved properties (Hubert et al., 2022). That brings us to the project's goal, which is to train the models using KGE on existing annotated workflows in order to see if it is possible to predict the correct tail entity given a head entity and a relation. By doing this, it would automate the annotation of feature workflows in the QuAnGIS system. Therefore, the research question is:

How do Knowledge Graph Embeddings help with automated annotation of GIS workflows?

Additionally, three sub-questions were created to guide the process of this project.

Sub-question 1: Which of the KGE models could be suited for this task?
Sub-question 2: Which of the KGE models performs best?
Sub-question 3: Does the triple creation in terms of indicating the inputs affect the tail prediction in the better model?

To answer these questions, we will compare two types of KGE models and two ways of triple creation in this thesis. The rest of this paper is divided into several sections. In order to gain an understanding of the techniques and decisions made in the analysis, a theoretical background is provided at the outset. The methodology then describes how the data was prepared and how the models were applied to it. Later, the models' evaluations for both cases are compared to see which KGE model performs better. Finally, the discussion and conclusion summarise the work while highlighting potential limitations and future research on this topic.

2. Literature Review

2.1. Core concepts of spatial information

Transdisciplinary research is a challenging task; however, it is an important part of science. To produce satisfactory output, extensive knowledge of the systems and possible tools to use is required. Geographical Information System (GIS) is no exception in that (Scheider, Nyamsuren, et al., 2020). Kuhn (2012) proposed the idea of core concepts of spatial information in order to bridge the gap between spatial information scientists and those from other domains. Those core concepts include field, object, event, network, location, neighbourhood, granularity, accuracy, meaning and value. Because their names are universal, they can be viewed as conceptual "lenses" that allow for the interpretation of spatial features, as well as the interpretation of spatial information in general (Scheider, Nyamsuren, et al., 2020).

2.1.2. Core concept data type CCD

There is a difference between the core concept and the data types of spatial information (Scheider, Meerlo, et al., 2020). The first is more concerned with the expert's interpretation, whereas the second is concerned with the concept's indirect representation. Both of them are important in analysis. Core concept data types (CCD) combine geometric data formats, such as polygons, lines and points, with core concept types (field, object etc.) and measurement levels (nominal, ordinal, interval, ratio, count). Furthermore, CCD types add necessary geo-analytical constraints and capture functionality in terms of transformations in semantic dimensions, namely: geometric layer types, core concepts of spatial information, attribute measurement level, and extensiveness (Kruiger et al., 2020). The first dimension mentioned generalises the spatial geometry of the entire layer or data. We can use it to determine whether the geometric properties of a layer are appropriate for representing geo-analytical concepts within its spatial extent. The core concepts have already been explained, but their task is to capture what the layer represents. When it comes to attribute measurement level, it restricts the types of operations that can be performed on values at a given level of measurement. The extensiveness then determines whether the attributes are extensive or intensive. The former are dependent on the size of the region and behave additively, whereas the latter are completely independent in size.

The combination of all four dimensions captures both syntactic and semantic aspects of geodata, as they were explicitly formalised at the attribute level by the introduction of corresponding attribute types for each of the aforementioned layer types, denoted by the suffix "A." (Kruiger et al., 2020). For example, the annotation of neighbourhoods in Amsterdam, is "Nominal A, ObjectQ, PointA". This means we have an attribute layer type "point", with a measurement level "nominal" and "ObjectQ" indicating that the attribute represents the quality values of an object.

2.1.3 Core concepts and CCD in terms of geo-analytical QA

Core concepts and data types of spatial information allow for the broader use of GIS. They can be seen as an "internal representation" of the GIS, as they capture the concepts and needed tools for approaching goe-analytical QA (Scheider, Meerlo, et al., 2020; Scheider, Nyamsuren, et al., 2020). Core concepts' properties automatically indicate some of the operations that can be applied to them (Xu, Nyamsuren, Scheider, et al., 2022).

In terms of QuAnGIS, core concepts and CCD types help in matching questions to workflows. As stated in the introduction, they are used to annotate workflows (Xu, H., Nyamsuren, E., & Scheider, S., 2022). Furthermore, they can be used as input for knowledge graph embeddings. This can then be used for automatic workflow annotation based on link prediction. That is because we can teach the model that certain CCD types combined with particular GIS tools will produce a specific output.

2.2. Knowledge Graph Embeddings

As indicated before, geo-analytical QA is challenging, especially because the answers can be in the form of workflows that describe necessary steps taken in GIS. Those answers are no different from graphs that have nodes and edges. Those respectively correspond to entities and relations, in terms of knowledge graphs. With that, we are able to create triples, where we present nodes as head and tail entities and edges as relation. That denoted as (h, r, t). However, there are some drawbacks to the simplicity of knowledge graphs. They are incomplete, spare and difficult to manipulate (Mai et al., 2020; Q. Wang et al., 2017). Furthermore, those graphs are easy to understand for humans, which is not the case for computers (Teja, 2023). Those limitations affect the performance of the QA process, as some triples might be missing or the fact that they might cause questions to be unanswerable (Mai et al., 2020). All of that made researchers use KGE models more enthusiastically. This technique embeds the components of KG, both entities and relations, into low-dimensional vector spaces (Hubert et al., 2022; Q. Wang et al., 2017). This way the graph structure and properties are preserved as much as possible. Additionally, it provides a scalable way of automating GIS workflow annotation. This can then be used to infer knowledge in workflows, both for QA or for workflow retrieval or synthesis.

Working with KGE models requires a few important parts. Firstly, we need to represent the entities and relations as vectors in the vector space. The tails and heads can be for example points, and the relations are presented as matrices, tensors or multivariate Gaussian distributions (Q. Wang et al., 2017). Then, when there is time to train the model, negative samples are created, to teach the model what is not correct (Hubert et al., 2022). This step replaces either the head or tail entity with another entity to create negative triples. Additionally, we need to define the scoring function, which will assign the higher scores to the "true triples" rather than the negative ones. Lastly, there is an optimization algorithm that maximises the plausibility of the observed positive triples (Q. Wang et al., 2017).

2.2.1. Knowledge Graph Embedding types

Knowledge Graph embedding models can be differentiated by some of their characteristics. Translational distance-based ones, for example, employ scoring functions based on the distance between two entities, implying that the probability of something is also measured in that term (Q. Wang et al., 2017). That is preceded by a specific type of translation carried out on the triple. Additionally, some models can be considered semantic matching models, also known as factorization-based (Q. Wang et al., 2017; Teja, 2023). They compare the underlying meanings of entities and relationships using similarity-based methods and semantic information for scoring functions. Lastly, some models utilise neural networks, and are becoming increasingly popular, and can be found in a variety of fields other than KGE models. They yield better results and enable more efficient and computational work (M. Wang et al., 2021).

2.2.2. Knowledge Graph Embeddings Usage

There are multiple implications of KGE. Some of them are called in-KG applications (Q. Wang et al., 2017). They are concerned with learning the entity and relation embeddings and analysing them. Examples of such employment are entity resolution and classification as well as triple classification and link prediction. The last one is particularly significant for this project. This method can be utilised in two ways. One is entity prediction, which occurs when there is a missing tail or head entity, and the other occurs when there is a missing relation and two entities are provided (M. Wang et al., 2021). In summary, the method uses the learned relationships between entities to find whether the missing entity or relation can be seen as a fact together with the other parts of the triple that are already established (Teja, 2023). This method grows more in popularity in many domains. It is good for forecasting and looking for the missing part (Rossi et al., 2021).

Another category of KGE application is known as out-of-KG, which goes beyond the KG input and scales up to other domains (Q. Wang et al., 2017). Furthermore, those applications assess the quality of the embeddings, and whether they are relevant to the task. Those include recommender systems, relation extraction or question answering. In terms of the last one, there has been some implementations made in the field of geography. One example is the location-aware question-answering system SE-KGE (Mai et al., 2020). This model embeds the bounding boxes of geographical entities or point coordinates into vector space in the embedding space. Unlike QuAnGIS, however, the project focuses on logic query answering, specifically conjunctive query answering (CGQ). There is one more example that is similar to the approach that is taken in this paper. The project is called "GeoQA". For now, there are not many papers published by them, however, on their website, they state that they aim to use knowledge graph embeddings to "answer complex non-factoid questions" (*GeoQA*, n.d.).

3. Methodology

3.1 Data collection

168 turtle files containing “raw” workflows were provided by the QuAnGIS system. They were automatically generated based on a smaller subset of twenty manually annotated workflows taken from GIS tutorials. Then using the workflow synthesis approach described by Kruiger et al. (2020) these workflows were generated for combinations of input and output types. They are a visual representation of the Resource Description Framework (RDF), which means that they provide a universal model to represent data. The graphs consist of objects (nodes or entities) and properties (edges or relations).

```
<https://example.com/#solution1> a ns1:Workflow ;
  ns1:edge [ ns1:applicationOf
<https://quangis.github.io/tool/abstract#IntersectDissolveField2Object> ;
      ns1:input1 _:"ObjectQ, VectorTessellationA, NominalA";
```

```
        ns1:input2 _:"FieldQ, VectorTessellationA, PlainNominalA";
        ns1:output _:"ObjectQ, VectorRegionA, ERA" ],
[ ns1:applicationOf <https://quangis.github.io/tool/abstract#SpatialJoinSumTessRatio> ;
        ns1:input1 _:"ObjectQ, VectorTessellationA, NominalA" ;
        ns1:input2 _:"ObjectQ, VectorRegionA, ERA";
        ns1:output [ "ObjectQ, VectorTessellationA, ERA" ] ;
```

Workflow 1. Modified example of the workflow from the file "solution_1.ttl".

Workflow 1 shows an example of one of the files used. It displays the RDF graph, which is based on the RDF triple of subject, predicate, object, and represents the fact or statement about the relationship between nodes. The preceding example defines a workflow with two application steps, each of which uses a specific tool on a specific input to produce an output. Two edges enclosed in square brackets define the workflow. They form a complete workflow, with the first serving as an input to the second. Figure 3 depicts this workflow structure to help understand it better.

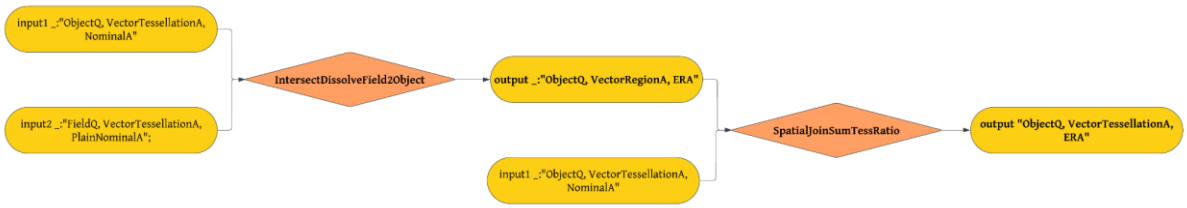

Figure 3. Visual representation of RDF file presented in Workflow 1.

3.2 Triple creation

In the previous section RDF triple was introduced. The following terminology is used in the knowledge graph: head for the subject, relation for the predicate, and tail for the object. Furthermore, in the case of data used in this project, the head corresponds to the input, relation to the transformation tool, and tail to output. As visible in figure 3, the composition of only three elements does not exist in the files that were used. The data was adapted for the purpose of using it in the KGE models. Therefore, data was processed in two ways:

a) Multiple input triple

The workflow depicted in figure 3 can be divided into two distinct sub-workflows, as shown in figure 4.

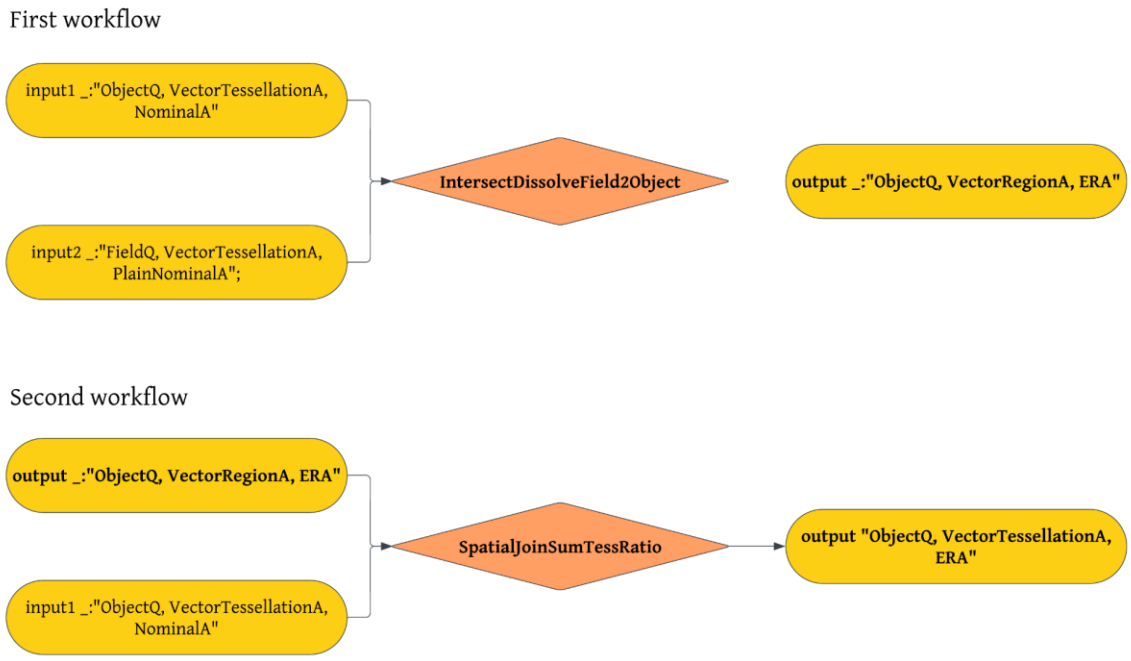

Figure 4. Sub-workflows of workflow 1.

However, there is another issue, namely that the graphs have two inputs, and each triple requires only one head. As a result, the triple produced by this method has "unique" names that combine two inputs into one. This way, the graph's structure can be preserved to the greatest extent possible. Based on figure 4, the triples will look as following:

Triples for the first workflow:
*Head:*ObjectQ, VectorTessellationA, NominalA, FieldQ, VectorTessellationA, PlainNominalA
Relation: IntersectDissolveField2Object
Tail: ObjectQ, VectorRegionA, ERA

Triples for the second workflow:
Head: ObjectQ, VectorRegionA, ERA, ObjectQ, VectorTessellationA, NominalA
Relation: IntersectDissolveField2Object
Tail: ObjectQ, VectorRegionA, ERA

This method resulted in the creation of 125 triples that are not duplicates.

b) Single input triple

To obtain the final output, this method disregards the multiple input. Rather, each input is treated as its own head, which is made up of the tool and the output. As a result, triples for the first workflow in figure 4 that uses the "IntersectDissolveField2Object" tool will be:

Triple 1.
Head: ObjectQ, VectorTessellationA, NominalA,
Relation: IntersectDissolveField2Object

Tail: ObjectQ, VectorRegionA, ERA

Triple 2
Head: FieldQ, VectorTessellationA, PlainNominalA
Relation: IntersectDissolveField2Object
Tail: ObjectQ, VectorRegionA, ERA

Therefore, 161 non duplicated triples were created in this data set.

3.3 Knowledge Graph Embeddings models

The data used in this project has two important characteristics. To begin, it is multi-relational, which means that there are different types of relationships between entities (Bordes et al., 2013). The second important property is that workflow relations are asymmetrical. That means that they go only in one direction, which is because the graphs present the dependency between the core concepts and CCD and the GIS tools used on them, and they have a specific one-way flow direction.

Therefore, the chosen model needs to take those two properties into account. From different KGE models described in the literature review, semantic matching and neural network-based models seem the most appropriate, especially RESCAL and ConvE. When it comes to the first one, it belongs to the category of models that manage semantic information. RASCAL is based on the three-way tensor factorization "where two modes are identically formed by the concatenated entities of the domain and the third mode holds the relations' as explained by Nickel et al. (2011). RESCAL is good for modelling multidimensional tensors, as well as antisymmetric triples, as it learns from the latent components. Figure 5. shows the structure of it.

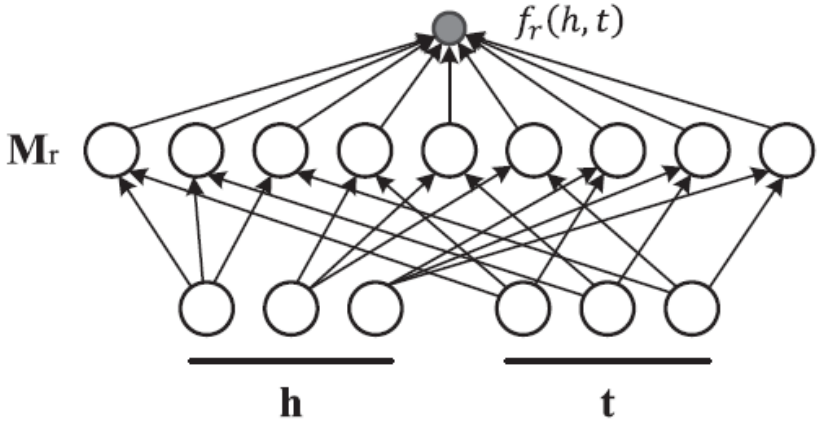

Figure 5. Structure of RESCAL[1].

Mr is a symbol that indicates that RESCAL can handle the relations between each head(h) and tail(t) as a matrix. Then fr(h,t) function computes a plausibility score for each triple.

[1] Source: Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017, December 1). *Knowledge Graph Embedding: A Survey of Approaches and Applications*. IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/8047276

The ConvE model, which uses a neural network structure, is the second option. It reshapes the numerical representations of triples using a 2D convolutional neural network. Then, in order to learn the embeddings, it extracts features using convolution filters (Teja, 2023). Figure 6 depicts the model's four-step process.

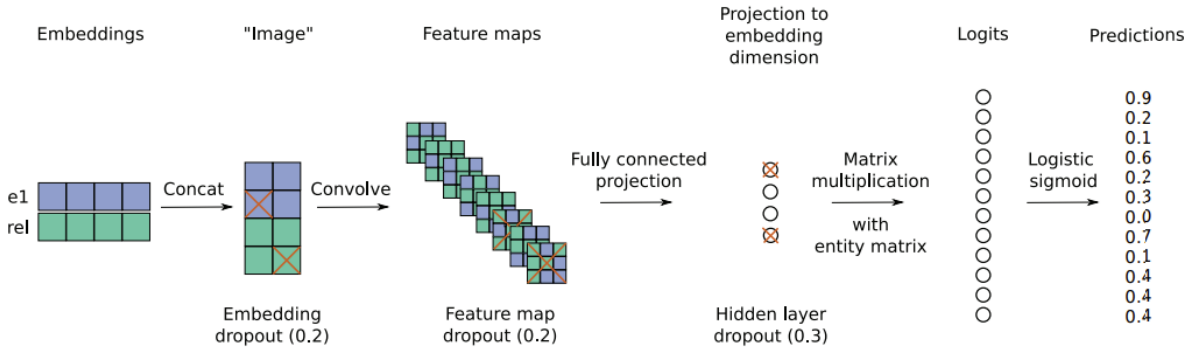

Figure 6. Process of ConvE model[2].

First, the embeddings of entities and relations are reshaped and concentrated. The matrix is then fed into the convolutional layer. The feature map tensor is vectorised and projected into k-dimensional space as a result of the fourth step. Finally, it is compared to all candidate embeddings. Overall, this model captures structural and semantic data patterns and handles asymmetric triples (Dettmers et al., 2018).

3.4. Implementation

Automatic annotation of GIS workflows can be done with link prediction. This method relies on learning about the relationship between two entities, to then try to predict either the head or tail based on the two other elements of the triple. In this paper, the tail prediction is done where we know the head entity and relation and we try to predict the tail, which can be written as (h, r, ?). That was possible with the use of the PyKEEN library[3] in Python. This package contains a number of functions that allow the use not only well-known KGE models, but also an optimisation algorithm that runs through all possible hyperparameter combinations to find the best one. PyKEEN is easy to use and does the majority of the job for modelling, optimization and evaluation automatically. However, before that data needs to be prepared to be suitable for the models. This step is a creation of entities, mapping them and creating so-called "factories" for training, testing and validation. Then that data can be fed to the models. The process after that consisted of four steps: determination of parameters, creating a basic model with them, fine-tuning mentioned parameters, and evaluating the tail prediction. All of the code can be found on github: https://github.com/wlibera/Automated-annotation-of-GIS-workflows-using-knowledge-graph-embedding-KGE-

[2] Source: Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D Knowledge Graph Embeddings. *Proceedings of the . . . AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.11573

[3] https://pykeen.readthedocs.io/en/stable/

3.4.1 Determination of parameters and basic modelling

The package offers a quick implementation of the model with elements such as negative sampling, scoring function and even evaluation in single function. Moreover, depending on the model, there is a range of parameters that can be set. Nonetheless, the determination of which parameters to use is dependent on the data and subjective decisions. As mentioned before, the data modelling is "basic". It means that the parameters included in it were chosen at the start and not determined by the hyperparameter tuning. For both RESCAL and ConvE, those included training assumption (training_loop = "slcwa"), negative sampling (negative _sampler = "basic") and learning rate (lr_scheduer = "exponential"). “Slcwa” refers to Stochastic Local World Assumptions (SLCWA). The reason for that is that Open World Assumption may lead to under-fitting of the data and Closed World Assumption is not a good choice either as it may lead to overfitting (*Training — Pykeen 1.10.1 Documentation*, n.d.)[4]. Therefore, PyKEEN implements Local Closed World Assumption (LCWA), and SLCWA. The reason why the SLCWA was chosen is that the model considered a random subset of the union of entities (head and tail) generated from LCWA as negative triples, which limits both of the problems mentioned before. Additionally, the use of SLCAWA, a basic negative sampling technique is automatically implemented. It uniformly randomly generates corrupted triples by using a corrupt head or tail operation. Lastly, for the learning rate, I decided to use exponential decay because it gradually decreases the learning rate over time allowing the model to make more significant updates at the beginning of training.

Then all of the parameters were implemented in the final model function to see how the data behaves in terms of the loss which shows the training process and embedding learning.

3.4.2 Fine tuning of parameters

As mentioned before, PyKEEN provides a pipeline for hyperparameter tuning[5]. When implemented, it goes through possible parameters n times and then lists the best combination. In this paper n = 30. On top of the parameters chosen before, the function for tuning goes through some default parameters for the models, which results in attributes of the function that were not specified before. For example, in the case of ConvE, the model presented the loss function “BCEAfterSigmoidLoss”[6] that is often used with the neural network models (Hentschke, 2021). Additionally, this tuning allows to determine the gamma for the learning rate, the batch size or the number of epochs. After the parameters are determined, the best pipeline configuration is saved to the specified directory and then implemented in the final model. All the parameters can be found in Appendix 1.

3.4.3 Results and evaluation of the models

After the final models were created, their ability to predict the correct tail was assessed. This was done for both multiple-input triples and single-input triples. For this step, all of the test

[4] https://pykeen.readthedocs.io/en/stable/reference/training.html

[5] https://pykeen.readthedocs.io/en/stable/tutorial/running_hpo.html

[6] https://pykeen.readthedocs.io/en/v1.10.1/api/pykeen.losses.BCEAfterSigmoidLoss.html#pykeen.losses.BCEAfterSigmoidLoss

triples were used and iterated over. The prediction follows the workflow's "natural" path, where the input (head) and transformation tool (relation) are indicated, and the model seeks the best-fitting output (tail). A score is assigned to each predicted tail, and they are ranked based on them. Higher score will be given to the tail that the model believes is the correct one based on head and relation. The prediction's results are then compared to the gold standard triple, which is specified in the test set.

Both RESCAL and ConvE are evaluated separately based on their loss curve and various ranking metrics that are automatically calculated by PyKEEN. The ranking metrics summarise model performance and how well it predicts and ranks the tail entities. A few of the ranking metrics are taken into consideration in this paper. The most common cases to evaluate link prediction include Hits@K, Mean Rank (MR) and Mean Reciprocal Rank (MMR) (Rossi et al., 2021). However, PyKEEN presents some alternatives for some of them. All of the metrics used and their descriptions are presented in Table 1

Name of the measure metric	**Description**
Adjusted Arithmetic Mean Rank (AAMR)	This metric is the equivalent of simple MR. However, since MR is heavily dependent on a number of triples and, therefore, makes comparing different datasets more difficult, the adjustment of it accounts for it (Ali et al., 2021). AAMR employs the arithmetic mean and considers the ranks of the correct entities or relationships predicted by the model. Therefore, the lower the value of AAMR, the better the outcome since then the correct prediction is ranked higher on average.
Adjusted Inverse Harmonic Mean Rank (AIHMR)	Inverse Harmonic Mean Rank (IHMR) was introduced by Hoyt et al., (2022), and it works similarly to the AAMR, but it also takes the inverse values into consideration of the rankings of correct predictions. Again, in PyKEEN the adjusted version is used. Similarly to AAMR, the lower value indicates better performance of the model, where correct triple elements are ranked higher on average.
Hits@K	Hits@K[7] shows the proportion of true triples in the top K positions of the ranked list. In this case, we want the number to be as close to 1 as possible, as it being 1, indicates that the correct entity is predicated on the top-ranked position among K candidates. In this paper, the following K values are taken into consideration {1, 3, 5, 10}.

Table 1. Measure metrics used for the evaluation for the models.

[7] https://pykeen.readthedocs.io/en/v1.10.1/api/pykeen.metrics.ranking.HitsAtK.html

3.4.4. Comparison of the models

In section 3.2 two datasets were created. The first one contains triples, where one head can have multiple inputs. This data is used as the core for this thesis, as it preserves better the original structure of the QuAnGIS workflows. Therefore, after the hyperparameter tuning two models: RESCAL and ConvE are analysed and compared on two bases. Firstly, the prediction of the correct tail is carried out on the test set for each model. This allows us to see how many triples are predicted correctly and, therefore, determine a better model "in practice". Afterwards, the loss functions and measuring metrics are carried out. This step allows for a more detailed comparison of each model in terms of how the model "should" perform (based on the metrics) and how it actually performs (tail prediction). The contrast shows the overall performance of each model.

The second data contains the triples with single inputs. The aim of it is to see whether KGE models can be fed less structured data, where the relationships between the input and output of the workflows are not fully correct (as the output cannot be achieved without one of the inputs). This type of data, however, is the most similar to the "real" structure of triples in KG. Those triples will be fed only to the better-performing model "in practice" in terms of the multiple input triples.

4. Results

This section contains the results of the tail prediction for all of the models. First, the multiple-input triples are presented, and then the separate section is for the single-input ones. Before, both of the data were divided into train, validation, and test sets. For multiple-input heads, those consisted of 100, 13, and 12 triples respectively. For the single-input data, the number of triples were 128, 17, 16.

This section aims to show the performance of the model in terms of tail prediction when the head and relation are provided. This is done based on the test set, as those triples were not presented during training. The scores are shown in the tables, where they are sorted from the highest to the lowest score for top 10 predictions. The reason why the top 10 were chosen, is that this is the highest value of Hits@K taken into consideration, and will be beneficial for further comparison. The results are compared to the golden standard triple.

4.1. Tail prediction for multiple input triples

4.1.1. Tail prediction for RESCAL

There were 12 triples in the test set for RESCAL with multiple input heads. For all of them, the prediction was carried out, and an example of the result for triple (*ObjectQVectorTessellationAPlainRatioAObjectQVectorTessellationACountA', 'LoadCountAmounts', 'ObjectQVectorTessellationACountA')* is visible in table 2.

	Tail id	score	tail_label
69	69	3888,66	ObjectQVectorTessellationACountA
126	126	3598,87	ObjectQVectorTessellationAPlainOrdinalAObjectQVectorTessellationAERA
111	111	3559,33	ObjectQVectorTessellationANominalAObjectQVectorTessellationAPlainIntervalA
87	87	3557,71	ObjectQVectorTessellationACyclicAObjectQPointAPlainOrdinalA
48	48	3538,36	ObjectQPlainVectorRegionAPlainRatioAObjectQVectorTessellationAERA
84	84	3531,25	ObjectQVectorTessellationACyclicAObjectQPointAIRA
36	36	3519,13	ObjectQPlainVectorRegionAPlainIntervalAObjectQVectorTessellationAERA
56	56	3518,72	ObjectQPointAPlainRatioAObjectQPlainVectorRegionACountA
113	113	3507,20	ObjectQVectorTessellationAOrdinalAObjectQVectorRegionAERA
13	13	3500,03	ObjectQPlainVectorRegionACountAFieldQVectorTessellationAPlainOrdinalA

Table 2. Results of the tail prediction for triple ('ObjectQVectorTessellationAPlainRatioAObjectQVectorTessellationACountA', 'LoadCountAmounts', 'ObjectQVectorTessellationACountA')

Table 2 presents the possible tails given the '*ObjectQVectorTessellationAPlainRatioAObjectQVectorTessellationACountA'* and *'LoadCountAmounts'* for a tool. It indicates the tail IDs, their label and the score. The results show that based on those two pieces of information, the correct tail should be 'ObjectQVectorTessellationACountA', which when compared with the gold standard triple is in fact true.

Such prediction was carried out for all of the 12 test triples. Figure 7 presents the distribution of the results for all of them.

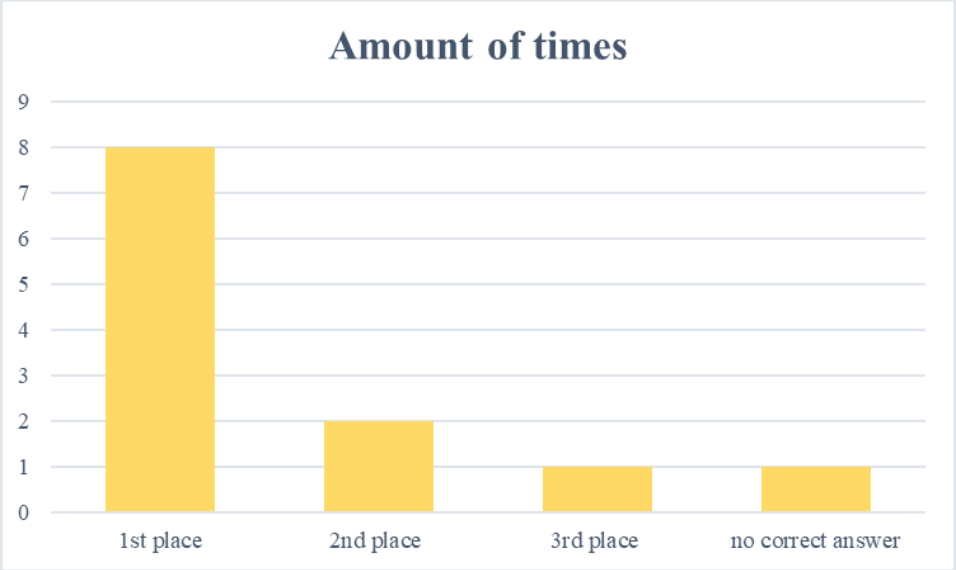

Figure 7. How often the correct tail was at a specific place in the ranking for RESCAL with multiple-input triples.

As presented in figure 7, for the majority of the cases (8 out of 12) the correct tail was predicted with the highest score. There were only 2 triples where the correct tail was scored the second highest and one where there was no correct answer. That was the situation for the following triple:

Head: 'ObjectQVectorTessellationAPlainIntervalAObjectQVectorTessellationAERA'
Relation: 'SelectLayerByLocationPlainRegionObjects'
Tail: 'ObjectQVectorTessellationAERA'

What is interesting is that there was previously a triple with the same relation and tail but a different head. The previous prediction was correct, but not in this case. This situation indicates that the model may need to learn more about the dependencies between triples.

4.1.2. Tail prediction for ConvE

Table 3 displays the prediction for the same triple as RESCAL. The triple looked like this**:**

Head: 'ObjectQVectorTessellationAPlainRatioAObjectQVectorTessellationACountA'
Relation: 'LoadCountAmounts'
Tail: 'ObjectQVectorTessellationACountA'

However, this yielded the most optimistic result because it was the only time the predicted tail scored the highest and was the first one.

	tail id	**score**	**tail_label**
69	69	-0,433	ObjectQVectorTessellationACountA
120	120	-0,437	ObjectQVectorTessellationAPlainNominalA

25	25	-1,015	ObjectQPlainVectorRegionAIRA
123	123	-1,188	ObjectQVectorTessellationAPlainNominalAObjectQVectorTessellationABooleanA
85	85	-1,188	ObjectQVectorTessellationACyclicAObjectQPointAPlainIntervalA
9	9	-1,258	ObjectQPlainVectorRegionACountA
37	37	-1,302	ObjectQPlainVectorRegionAPlainNominalA
22	22	-1,447	ObjectQPlainVectorRegionAERA
91	91	-1,504	ObjectQVectorTessellationAERA
113	113	-1,524	ObjectQVectorTessellationAOrdinalAObjectQVectorRegionAERA

Table 3. Results of the tail prediction for triple ('ObjectQVectorTessellationAPlainRatioAObjectQVectorTessellationACountA', 'LoadCountAmounts', 'ObjectQVectorTessellationACountA')

When it comes to the other triples in the testing set, the situation differs a lot depending on a triple, which can be seen on figure 8.

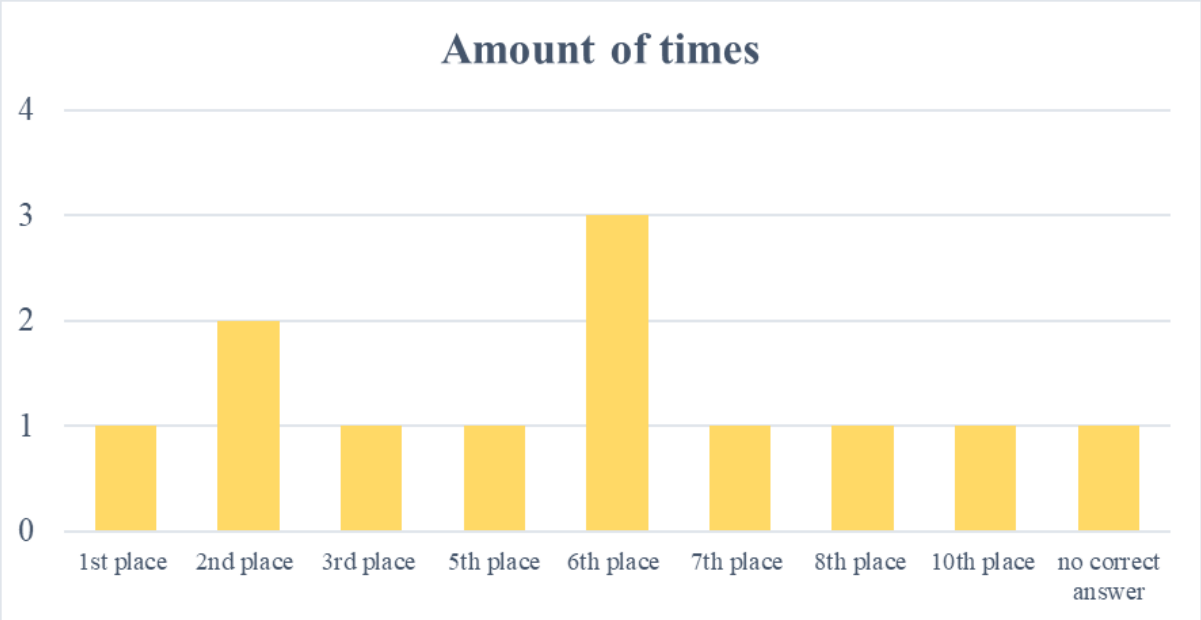

Figure 8. How often the correct tail was at a specific place in the ranking for ConvE.

On the bright side, two triples showed up in second place, and they did not have any part of a triple in common. Then the correct tail prediction was 3 times on the 6th position, and other than that we could observe the true tail in the 3rd, 5th, 7th, 8th and 10th placements. Similarly, to the RESCAL model, the correct triple was not predicted at all in one case. The triple was as follows:

Head: *'ObjectQVectorTessellationACyclicAObjectQPointACountA'*
Relation: *'SpatialJoinCountTess'*
Tail: *'ObjectQVectorTessellationACountA'*

Furthermore, the combination of relation and tail was correctly predicted previously, and the tail was on the sixth predicted tail. Again, the difference was in the head.

4.2. Tail prediction for single input triples

Since the RASCAL model performs better “in practice” for multiple-input triples data, it was additionally applied for the case when heads have single input. The test set consisted of 16 triples for single input triples. Table 4 shows the outcomes of one of the triples, namely:

Head: 'ObjectQPointAERA'
Relation: 'SelectNeighborhoodsByLocationDistPointObject'
Tail: ‘ObjectQPlainVectorRegionAPlainNominalA’

	Tail id	**score**	**tail_label**
15	15	3730,77	ObjectQPlainVectorRegionAPlainNominalA
4	4	3327,98	FieldQRasterAPlainRatioA
12	12	3256,79	ObjectQPlainVectorRegionANominalA
10	10	3211,38	ObjectQPlainVectorRegionAERA
0	0	3194,99	FieldQPlainVectorRegionAPlainNominalA
27	27	3189,99	ObjectQVectorTessellationACountA
34	34	3186,85	ObjectQVectorTessellationAPlainNominalA
11	11	3183,83	ObjectQPlainVectorRegionAIRA
14	14	3176,05	ObjectQPlainVectorRegionAPlainIntervalA
23	23	3160,03	ObjectQPointAPlainOrdinalA

Table 4. Results for tail prediction for triple ('ObjectQPointAERA', 'SelectNeighborhoodsByLocationDistPointObject',‘ObjectQPlainVectorRegionAPlainNominalA’)

The result of this prediction compared with the golden standard triple is correct. What is interesting, is that this is not the only case. When the prediction was applied to all 16 triples in the test set, all of the results came back as accurate as this one. For all of them, the model predicted the correct tail with the highest score.

5. Evaluation

Evaluation of the models consists of the analysis of loss function, as well as the measure metrics. Again, first RESCAL and ConvE models with multi-input triples are presented and then RESCAL with single-input triples.

5.1. Results of model training on multiple input triples

5.1.1 Loss function

The loss curve is a popular plot for graph embeddings. It presents the training process and how well the model is learning (Jose, 2021).

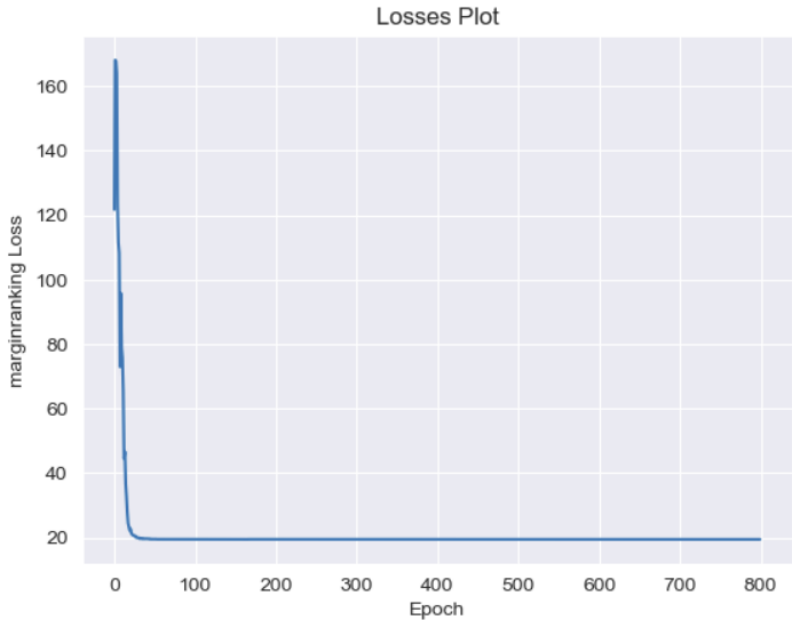

Figure 9. Loss function for RESCAL model with multiple input triples after fine tuning the parameters.

The loss vs. epoch diagram for RESCAL with multiple inputs is shown in figure 9. The model employs margin-ranking loss, and its value is rapidly decreasing. The minimum, indicating that the model can distinguish between positive and negative triples, is at 20 and around 25 epochs. That is also when the model stops improving, which means that the scores for positive triples are higher than for the corrupted ones.

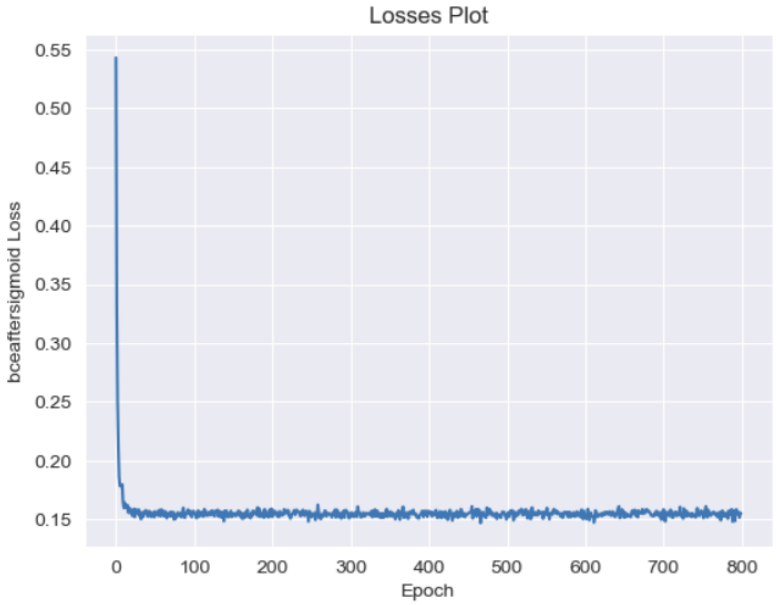

Figure 10. Loss function for ConvE model with multiple input triples after fine tuning parameters.

ConvE, unlike RESCAL, employs different loss functions. Figure 10 shows that the model uses Binary Cross Entropy After Sigmoid. This is specific to tasks such as binary classification, where the model predicts whether or not a relationship between entities exists. As visible on the chart, when the value decreases, the prediction of the true label is better. However, what is interesting is the fact that the loss function never stops fluctuating. This could be due to the SLCWA assumption used when training the model.

5.1.2. Measure metrics

The measure metrics listed in section 3.4.3 were calculated for each model, and the results are shown in Tables 5, 6 and 7. They contain three perspectives, "both", "head" and "tail". The first one combines the performance of the model for both tail and head. It is more used to compare the models to each other, and therefore the focus on that will be in the discussion section. Nonetheless, some comments are made about them. However, the "tail" and "head" values are used to make a comparison within the model, and to determine if it is better in terms of head to tail prediction.

5.1.2.1. Measure Metrics for RESCAL

Table 5 displays all of the metrics available for evaluating the RESCAL model. Overall, the results for "both," conclude that the model is fairly accurate. The value for AIHMR should be as low as possible. Because the value for RESCAL is 0.51, we can conclude that the model ranks the correct triples higher than the negative ones on average. The situation is slightly better for AAMR, because the value is 0.4, therefore, more triples are correctly scored. In terms of Hits@K rank, the situation appears to be promising, as the metric was greater than 0.5 for every value of K. The best-case scenario is when k=10 is predicted as the correct tail entity and is included in the top 10 predictions for 75% of the test triples.

Measure Metric	Both	Head	Tail
Adjusted Arithmetic Mean Rank (AAMR)	0.412	0.885	**0.052**
Adjusted Inverse Harmonic Mean Rank (AIHMR)	0.516	**0.012**	1.0
Hits@K	**0.624**	0.167	**1.0**
Hits@1	**0.531**	0.006	**1.0**
Hits@10	**0.75**	0.5	**1.0**
Hits@3	**0.531**	0.062	**1.0**
Hits@5	**0.562**	0.125	**1.0**

Table 5. Realistic measure metrics for RESCAL model with multiple input triples.

The difference between head and tail prediction in the RESCAL is significant, especially in terms of Hits@K. When this rank equals 1, it means that the ground truth triple is among the top K elements. That is true for all of the k values {1,3,5,10} that are considered . A positive score indicates that regardless of the head and relation, it should correctly predict the tail every time. Similarly, the prediction for the tail appears to be better in terms of AAMR. However, AIHMR is the only instance in which the model appears to perform better for the head prediction. This, as will be seen in the other cases, produced results that are opposite to the indication of other metrics. Beside that, in terms of tail prediction, RESCAL is better at distinguishing negative samples from correct triples when all metrics are considered.

5.1.2.2. Measure Metric for ConvE

When we look at Table 6 in the "both" column, the ConvE model seems to perform satisfactorily in terms of AAMR and AIHMR. For both of those ranks, the value is low. Therefore, it means that the correct triples get higher scores. In terms of Hits@K, the situation is slightly worse. From it, it looks like that correct answer is in the top 10 predictions for 58% of test triples. That score is slightly above the average. Moreover, the probability that the predicted triple is true and, it is the very first prediction, is only 8.3%, on average.

Measure Metric	Both	Head	Tail
Adjusted Arithmetic Mean Rank (AAMR)	**0.339**	0.624	**0.076**
Adjusted Inverse Harmonic Mean Rank (AIHMR)	**0.201**	**0.109**	0.293
Hits@K	0.547	0.182	**0.910**
Hits@1	0.083	0.083	0.083

Hits@10	0.583	0.25	**0.916**
Hits@3	0.291	0.166	**0.416**
Hits@5	0.333	0.166	**0.5**

Table 6. Realistic measure metrics for ConvE model with multiple input triples.

When relation and head are provided, the model performs better for tail entities. In terms of AAMR, the value is very low, indicating excellent prediction performance. Similarly, higher Hits@k values indicate better tail performance. The only exception is when k = 1, in which case the rank value (0.083) is the same for both the head and tail. The probability of prediction being in the top 10 of test triples is the highest, this time 91.6%. Again, contrary to what might be expected based on the other metrics, the value of AIHMR for head is lower. That indicates that the model performs better for head prediction in that case. Nonetheless, by looking at all the metrics provided, ConvE is better at distinguishing negative samples from correct triples in terms of tail prediction.

5.2. Results of model training on single input triples

5.2.1. Loss Function

When it comes to the loss function for RESCAL with single input, the results are presented in figure 11. What is interesting, is that no matter whether we consider the single or the multiple input the model learns equivalently quickly and finishes on the same loss value (20). Moreover, the model learns at the similar rate as the multiple input one.

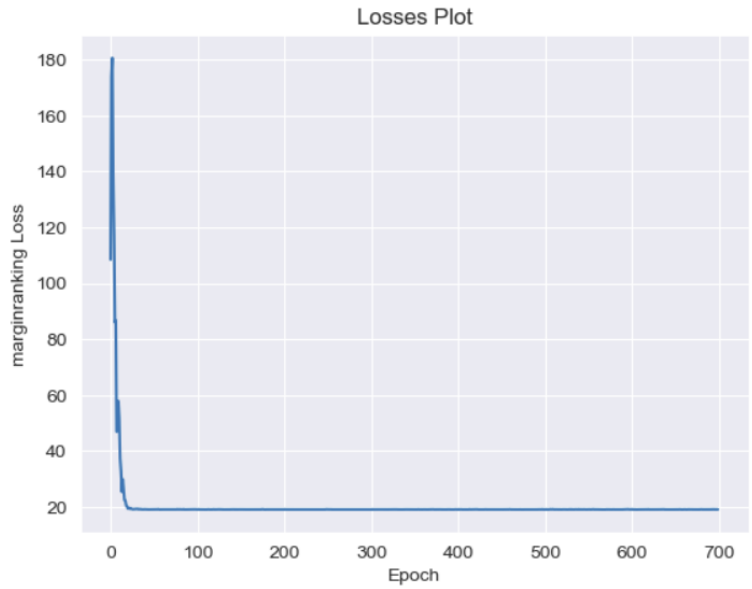

Figure 11 Loss function for RESCAL model with single input triples after fine tuning parameters.

5.2.2. Measure Metrics

The RESCAL model based on the single input triple has relatively good scores in terms of the metrics when looking at the “both” column. For every single case of Hits@K, the value does not drop down below 0.593. That shows that on average, this model should predict the correct part of the triple as the first score in 59.3% of the test triples. Additionally, AAMR is low, which indicates good performance.

Measure Metric	**Both**	**Head**	**Tail**
Adjusted Arithmetic Mean Rank (AAMR)	0.281	0.581	**0.526**
Adjusted Inverse Harmonic Mean Rank (AIHMR)	0.595	**0.173**	1.0
Hits@K	0.765	0.479	**1.0**
Hits@1	0.563	0.125	**1.0**
Hits@10	0.844	0.687	**1.0**
Hits@3	0.688	0.375	**1.0**
Hits@5	0.719	0.438	**1.0**

Table 7. Realistic measure metrics for RESCAL with single input triples.

Looking closely at “tail” vs “head” metrics is clear that the tail prediction is better. The Hits@K metric values are always equal to 1, even when k=1. That indicates that the correct triple should be in the first position of the possible predicted triples. Similarly to other models, the AIHMR is better for the head, rather than the tail.

6. Discussion and conclusion

The analysis and comparison of the models and their performance was done in three steps. First, we looked at the prediction in practice, and then we evaluated the models with the loss functions and the measure metrics. All of that was done for the data with multiple inputs in heads. Lastly, the better model "in practice", RESCAL, was analysed with the single input triples data.

When looking at the results for tail prediction for both RESCAL and ConvE (on multiple-input triples), it is clear that the former performs significantly better, as visible in Figures 7 and 8. There were more cases where the "ground truth" tail was the same as the highest-scored tail by the model. Moreover, RESCAL stayed more consistent when it came to ranking tails, compared to ConvE, which placed them in different positions almost every time.

When we look at the loss curve, the situation does not differ significantly. For both RESCAL and ConvE the model seems to learn efficiently about the relations that occur.

Table 5 presented us with excellent results when it comes to the comparison between head and tail prediction for RESCAL with multiple-input data. As mentioned before, Hits@K indicated that it should predict the tail correctly all the time, which we proved to be wrong on test data, as described above. The reason for this distinction can be that, in theory, for the model it should be easier to predict the tail entity due to the natural flow of the workflow. However, the ConvE model was not far off from the results of measure metrics (Table 6). For Hits@K the values seem to be more accurate, as the highest value for tail prediction was at k=10. When the model was applied on the test triples, the "correct" tail was within the top 10 predictions, except once. Based on the different values of K, for the Hits@K metric, both of the models should perform better in tail prediction. However, Hits@K is not the best metric for comparison between the two models. That is because this metric ignores triples with a rank higher than K (Hubert et al., 2022). As a result, the model in which the ground truth is located at position K+1 is regarded as equally satisfactory as one in which it is located at position K+d, where d is significantly greater than 1.

In order to compare RESCAL and Conve, we can use "both" metrics. They showed that the ConvE model assigned higher scores to the correct triples, indicating better performance, when looking at AARM and AIHMR. For both of them, the value for ConvE was lower than for RESCAL, which is the desired result.

What is described above, points out that the RESCAL model is better in practice, while ConvE got better results in terms of evaluation. There are multiple reasons for such a distinction. First of all, both AAMR and AIHMR are rank-based metrics. They assess the relative position of entities among others (Hubert et al., 2022). Since they are considering the overall performance, some generalisations may occur. Additionally, the reason may lie in the difference between how the evaluation metrics and predictions work. The metrics have different objectives and criteria compared to the model prediction in practice. Lastly, RESCAL is the factorization-based model that aims to capture the interaction between the entities and relations in the best way possible (Nickel et al., 2011). Moreover, it does so on a more global scale. Contrary to this, ConvE captures more local patterns and relationships and uses that as an input for convolutional layers (Dettmers et al., 2018).

Since RESCAL was established to perform better in practice on multiple input data, it was also applied to the single input one. The outcomes were unexpected. Despite the fact that the AAMR and AIHMR did not perform well, the model's performance on test data was excellent. The model predicted the correct triple, every single time in the first place. This was unexpected given the way the data was generated, and the workflow structure was disrupted. Technically, the model should not have learned the relationships, as well as it did if there was no interaction function to show the model that multiple inputs are required to predict the correct output. The reason for such result could be that we separated inputs into different triples, which causes more triples to have the same output, increasing the likelihood of predicting correct tails. It

could be a good outcome because we can assume that if we have more triples that indicate one output with its possible inputs, the accuracy of the models will improve, especially if we implement it in multiple input triples. It is also possible that separating multiple inputs and using KGE models is not appropriate for our workflows.

Although some of the results appear promising, and RESCAL performed better on the test set, there are numerous limitations and potential future work. We could start with additional evaluation methods, such as Sem@K, which was introduced in a paper by Hubert et al. (2022), because it captures the semantic profile of the relations and thus may be better to use to compare the models. It was, however, not available in the Python library used in this project. PyKEEN is a user-friendly package, but it has some drawbacks on its own. The functions are typically written for pre-existing datasets that are specific to knowledge graphs. When it comes to custom data, such as the one used in this project, the use of some of them becomes more complicated. Inductive learning link prediction is one of the examples. The goal is to predict links between known and unknown entities or relations (Galkin, 2022). The model is trained in such a way that it can generalise its knowledge to new, previously unseen entities or relationships. It can predict links that were not observed during training. Even though this method is included in the PyKEEN, it is done on their own data and is not appropriate for the custom one. Furthermore, increasing the amount of data could be one of the improvements. The situation with RESCAL with single input triple and multiple input triple may indicate that the model learns better when there are more triples available. Last but not least, the data on its own is a limitation. As presented in section 3.2 the normal triples of KG were adapted for this project, to preserve the properties. Even though the single excellent results, we cannot be sure that it actually learned all the inputs to receive some of the outputs. Because the data was not appropriate, in some of the predictions of the test set, the proposed tails by the model consisted of multiple outputs merged together. That is not something that should happen, as there is only one possible output when it comes to geospatial data. That, however, can be due to the fact that when the model is trained, it puts all the entities (heads and tails) into one.

Taking all of these factors into account, the answer to the research question is that Knowledge Graph embedding is not the best method for automatic annotation of GIS workflows. We determined that the factorization-based models and neural network ones are the most suitable. Despite the fact that RESCAL has some promising results, the data fed into it is not suitable. That was further proven by the performance of ESCAL in single-input data. Where the triples were as the KGE models expect them to be, but we cannot trust the results, as they divide the workflows into too many sub-steps.

Acknowledgments

Sincere gratitude goes to Haiqi Xu for the guidance and mentorship that enabled me to complete that project, which taught me so many valuable skills and values. I also want to thank Dr. Simon Scheider for his encouragement and support.

References

A Unified Framework for Rank-based Evaluation Metrics for Link Prediction in Knowledge Graphs. (n.d.). Benjamin M. Gyori, Ph.D. https://scholar.harvard.edu/bgyori/publications/unified-framework-rank-based-evaluation-metrics-link-prediction-knowledge-graphs

Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., & Lehmann, J. (2021). Bringing Light Into the Dark: A Large-Scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(12), 8825–8845. https://doi.org/10.1109/tpami.2021.3124805

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-relational Data. In *HAL (Le Centre pour la Communication Scientifique Directe)*. French National Centre for Scientific Research. https://hal.archives-ouvertes.fr/hal-00920777

Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D Knowledge Graph Embeddings. *Proceedings of the . . . AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.11573

Galkin, M. (2022, March 26). Inductive Link Prediction in Knowledge Graphs - Towards Data Science. *Medium*. https://towardsdatascience.com/inductive-link-prediction-in-knowledge-graphs-23f249c31961

GeeksforGeeks. (2023). Downloading PDFs with Python using Requests and BeautifulSoup. *GeeksforGeeks*. https://www.geeksforgeeks.org/downloading-pdfs-with-python-using-requests-and-beautifulsoup/

GeoQA. (n.d.). GeoQA. https://geoqa.di.uoa.gr/index.html

Hentschke, H. (2021, December 7). Sigmoid Activation and Binary Crossentropy —A Less Than Perfect Match? *Medium*. https://towardsdatascience.com/sigmoid-activation-and-binary-crossentropy-a-less-than-perfect-match-b801e130e31

Hoyt, C. T., Berrendorf, M., Gaklin, M., & Gyori, B. M. (2022). A Unified Framework for Rank-based Evaluation Metrics for Link Prediction in Knowledge Graphs. *ResearchGate*. https://www.researchgate.net/publication/359254469_A_Unified_Framework_for_Rank-based_Evaluation_Metrics_for_Link_Prediction_in_Knowledge_Graphs

Hubert, N., Monnin, P., Brun, A., & Monticolo, D. (2022). Knowledge Graph Embeddings for Link Prediction: Beware of Semantics! *ResearchGate*.

https://www.researchgate.net/publication/363834073_Knowledge_Graph_Embeddings_for_Link_Prediction_Beware_of_Semantics

Introduction to Knowledge Graph Embedding — dglke 0.1.0 documentation. (n.d.). https://aws-dglke.readthedocs.io/en/latest/kg.html

Jose, G. V. (2021, December 12). Useful Plots to Diagnose your Neural Network - Towards Data Science. *Medium*. https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45

Kamakoti, B. (2021, December 14). Introduction to Knowledge Graph Embeddings - Towards Data Science. *Medium*. https://towardsdatascience.com/introduction-to-knowledge-graph-embedding-with-dgl-ke-77ace6fb60ef

Kruiger, J. F., Kasalica, V., Meerlo, R. J., Lamprecht, A., Nyamsuren, E., & Scheider, S. (2020). Loose programming of GIS workflows with geo-analytical concepts. *Transactions in GIS*, *25*(1), 424–449. https://doi.org/10.1111/tgis.12692

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, *26*(12), 2267–2276. https://doi.org/10.1080/13658816.2012.722637

Mai, G., Janowicz, K., Cai, L., Zhu, R., Regalia, B., Yan, B., Shi, M., & Lao, N. (2020). *SE-KGE* : A location-aware Knowledge Graph Embedding model for Geographic Question Answering and Spatial Semantic Lifting. *Transactions in Gis*, *24*(3), 623–655. https://doi.org/10.1111/tgis.12629

Metrics — pykeen 1.10.1 documentation. (n.d.). https://pykeen.readthedocs.io/en/stable/reference/metrics.html

Newlon, B. (n.d.). *Getting Started with RDF & SPARQL The basics of RDF graphs and the SPARQL query language* [Slide show]. stardog academy. https://info.stardog.com/hubfs/Stardog%20Academy%20-%20Stage%203%20Fundamentals_Slide%20Decks/Stardog%20Academy%20Video%203_%20Getting%20Started%20with%20RDF%20and%20SPARQL.pdf

Nickel, M., Tresp, V., & Kriegel, H. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data. In *International Conference on Machine Learning* (pp. 809–816). https://icml.cc/2011/papers/438_icmlpaper.pdf

Rossi, A., Barbosa, D., Firmani, D., Matinata, A., & Merialdo, P. (2021). Knowledge Graph Embedding for Link Prediction. *ACM Transactions on Knowledge Discovery From Data*, *15*(2), 14. https://doi.org/10.1145/3424672

Scheider, S., Meerlo, R., Kasalica, V., & Lamprecht, A.-L. (2020, June 30). *Ontology of core concept data types for answering geo-analytical questions*. https://josis.org/index.php/josis/article/view/125

Scheider, S., Nyamsuren, E., Kruiger, H., & Xu, H. (2020). Geo-analytical question-answering with GIS. *International Journal of Digital Earth*, *14*(1), 1–14. https://doi.org/10.1080/17538947.2020.1738568

Sudhahar, S. (2021, December 7). Knowledge Graph Embeddings for Entity, Link Prediction — The Basics. *Medium*. https://medium.com/@saatviga/knowledge-graph-embeddings-for-entity-link-prediction-the-basics-4d433e048c0a

Teja, R. (2023, January 26). Knowledge Graph Embedding — A Simplified Version - Towards Data Science. *Medium*. https://towardsdatascience.com/knowledge-graph-embedding-a-simplified-version-e6b0a03d373d

Training — pykeen 1.10.1 documentation. (n.d.). https://pykeen.readthedocs.io/en/stable/reference/training.html

Wang, M., Qiu, L., & Wang, X. (2021). A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry*, *13*(3), 485. https://doi.org/10.3390/sym13030485

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017, December 1). *Knowledge Graph Embedding: A Survey of Approaches and Applications*. IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/8047276

Xu, H., Hamzei, E., Nyamsuren, E., Kruiger, H., Winter, S., Tomko, M., & Scheider, S. (2020). Extracting interrogative intents and concepts from geo-analytic questions. *AGILE-GISS*, *1*, 1–21. https://doi.org/10.5194/agile-giss-1-23-2020

Xu, H., Nyamsuren, E., & Scheider, S. (2022). Assemble geo-analytical questions through a Blockly-based natural language interface. *AGILE: GIScience Series*, *3*, 1–5. https://doi.org/10.5194/agile-giss-3-69-2022

Xu, H., Nyamsuren, E., Scheider, S., & Top, E. J. (2022). A grammar for interpreting geo-analytical questions as concept transformations. *International Journal of Geographical Information Science*, *37*(2), 276–306. https://doi.org/10.1080/13658816.2022.2077947

Appendix

Appendix1. Parameters for RASCAL and ConvE.

This appendix contains all the parameters (after tuning) for all the models.

1. **Parameters for RESCAL with multiple input data:**

```
final_model_RESCAL = pipeline(
   evaluator= "rankbased",
   filter_validation_when_testing = True,
   loss= "marginranking",
   loss_kwargs= dict(margin = 0.44486940248176415),
   lr_scheduler= "exponential",
   lr_scheduler_kwargs =dict(
      gamma =0.9444785118171812),
   model= "rescal",
   model_kwargs= dict(
      embedding_dim = 240),
   negative_sampler= "basic",
   negative_sampler_kwargs= dict(
      filtered = True,
      num_negs_per_pos = 82),
   optimizer= "adam",
   optimizer_kwargs= dict(
      lr= 0.037833961058865585),
   regularizer= "lp",
   regularizer_kwargs= dict(
      weight= 0.3932254214904536),
   testing = testing_factory,
   training= training_factory,
   training_kwargs= dict(
      batch_size = 256,
      num_epochs = 800),
   training_loop ="slcwa",
   validation= validation_factory,
   use_testing_data = True,
   random_seed=42
)
```

2. **Parameters for ConvE with multiple input data:**

```
final_model_multiple_conve = pipeline(
```

```
    evaluator = "rankbased",
    filter_validation_when_testing= True,
    loss= "bceaftersigmoid",
    lr_scheduler= "exponential",
    lr_scheduler_kwargs= dict(
      gamma = 0.8514039201039556
    ),
    model = "conve",
    model_kwargs= dict(
      feature_map_dropout = 0.30000000000000004,
      input_dropout = 0.4,
      output_channels = 32,
      output_dropout = 0.4
    ),
    negative_sampler = "basic",
    negative_sampler_kwargs= dict(
      filtered =  True,
      num_negs_per_pos = 21
    ),
    optimizer = "adam",
    optimizer_kwargs = dict(
      lr = 0.040051093486250015
    ),
    testing=  testing_factory,
    training = training_factory,
    training_kwargs= dict(
      batch_size = 32,
      drop_last = True,
      num_epochs= 800
    ),
    training_loop= "slcwa",
    validation=  validation_factory,
    use_testing_data = True,
    random_seed=42
)
```

3. Parameters for RESCAL with single input triples

```
final_model_RESCAL_single = pipeline(
    evaluator= "rankbased",
    filter_validation_when_testing = True,
    loss= "marginranking",
    loss_kwargs= dict(margin = 2.021353165839844),
    lr_scheduler= "exponential",
```

```
    lr_scheduler_kwargs =dict(
        gamma =0.9215220344613282),
    model= "rescal",
    model_kwargs= dict(
        embedding_dim = 192),
    negative_sampler= "basic",
    negative_sampler_kwargs= dict(
        filtered = True,
        num_negs_per_pos = 9),
    optimizer= "adam",
    optimizer_kwargs= dict(
        lr= 0.0725126122335267),
    regularizer= "lp",
    regularizer_kwargs= dict(
        weight= 0.4341880607711006),
    testing = testing_factory,
    training= training_factory,
    training_kwargs= dict(
        batch_size = 2048,
        num_epochs = 700),
    training_loop ="slcwa",
    validation= validation_factory,
    use_testing_data = True,
    random_seed=42
)
```