0.1 Mandatory Summary Section

This thesis describes various transformation and prediction tasks executed on the data generated by a 2023 paper [36] that parsed the medieval gazetteer Kitāb Mu'jam al-Buldān written by Yâqût al-Hamawî [14].

First, the various technologies and concepts that form the core of this thesis are introduced. These concepts include but are not limited to Knowledge Graphs, Graph Neural Networks, Wikidata [33], and various evaluation metrics such as Mean Rank and Hits@K.

Then, the next short section details the original gazetteer and its parsed version. Having established the base knowledge necessary to follow this thesis, the next sections analyze the initial parsed dataset represented as a graph. The graph, in its purest form, includes only **place** type nodes, **administrative hierarchical**, and **distance** edges. This analysis includes metrics such as network density, average node degree, and clustering.

Then, some initial link prediction experiments were performed using various knowledge graph embedding models, as implemented in Ampligraph [7]. These approaches are namely: TransE [3], DistMult [34], ComplEx [32], HolE [22] and RotatE [31]. However, these first attempts fell short of expectations regarding statistical performance and the ability to predict new, true positive links.

After analyzing the shortcomings of the initial models, the graph's structure was heavily modified. The graph on which the previous models were trained included various ancillary edges and nodes representing data scraped from Wikidata; these were removed. Moreover, previously, the graph's ontology did not differentiate between various distances and treated them equally. In later versions, these edges are binned according to predefined boundaries. Second, the hierarchical edges were explicitly defined across all levels. Third, reverse edges were added.

After some additional tests, it became apparent that there was a need to experiment with recent state-of-the-art models. Therefore, the rest of the thesis uses Neural Bellman-Ford Networks [39] as its basis (NBFNet). The model proposed in the NBFNet paper is currently SOTA [20] for link prediction using the FB15K-237 dataset.

The next section evaluates the results after training the NBFNet model on the gazetteer dataset. While the results are marginally better in some metrics, this thesis hypothesizes that it is possible to reach even better results by pretraining the model on a similar, but orders of magnitude larger, synthetic dataset. To create such a dataset, the thesis relies on WorldKG [9], a geographical knowledge graph constructed based on Open Street Maps [5] data.

The data found in the WorldKG dataset is used to create a synthetic dataset mimicking Yāqūt's Kitāb Mu'jam al-Buldān. This synthetic dataset also allowed

the introduction of specific biases that are not commonly found in the original data source but are of interest to the researchers.

The penultimate chapter discusses predicted new positive triplets and potential false positives generated by the original parsing strategy, the NBFNet interpretation of these triplets is also discussed. Within this section, there is also a dedicated part for triplet candidate discovery strategies, as it is a computationally expensive process. Various techniques, such as hop limiting and cluster triangles, are detailed.

The final section of the thesis reiterates the results and reflects on potential improvements, shortcomings, and future research possibilities.

Exploring Methods for Link Prediction on a Historical Geographic Knowledge Graph

Master's thesis

Balázs Márk Agárdi

Aalborg University
Department of Computer Science

Department of Computer Science
Selma Lagerløfs Vej 300
http://www.cs.aau.dk/

AALBORG UNIVERSITY

Title:
Exploring Methods for Link Prediction on a Historical Geographic Knowledge Graph

Thesis Period:
Spring Semester of 2024

Student:
Balázs Márk Agárdi

Supervisor:
Tomer Sagi

Page Count: 41

Date of Completion:
June 7, 2024

Abstract:

This thesis explores data parsed [36] from the medieval gazetteer, Kitāb Mu'jam al-Buldān written by Yâqût al-Hamawî [14].
The parsed dataset was created as part of The Middle East Heritage Data Integration Endeavour [28] The MEHDIE Project attempts to aggregate multilingual historical information in the domain of Medieval Middle Eastern History.
This thesis proposes an approach to transform the parsed Kitāb dataset into a knowledge graph to assist with MEHDIE's goal. To further assist the aggregation efforts, this thesis explores various link-prediction methods to predict new positive triplets and detect parsing errors.
Moreover, it proposes a potential approach for generating a large and dense synthetic pre-training dataset for geospatial link prediction models using a Knowledge Graph representation of OpenStreetMaps [5] called WorldKG [9].
Finally, it is demonstrated that a GNN model could act as an Error Detection model for the rule-based parser used to generate the original Kitāb dataset.

Contents

Chapter 1

Introduction

1.1 Problem Statement

A common task of historians is to digitize, parse, and categorize historical written records. One such project is The Middle East Heritage Data Integration Endeavour (MEHDIE) [28]. MEHDIE aims to aggregate and align multilingual data generated in the Medieval Middle East.

One such dataset is derived from Yaqut al-Hamawi's Dictionary of Countries [14]. This dataset was created by scanning a manuscript with OCR technology and then extracting parsed data from the text entries using a rule-based model [36]

However, this parsed dataset has some shortcomings. First, it is only available in a non-standard format, making it difficult for subsequent researchers to consume the data. Second, it is strictly based on the data parsed from the original manuscript. This limitation, however, hinders MEHDIE's data alignment initiatives. This thesis addresses the former issue by experimenting with various knowledge graph representations of the original dataset . It addresses the latter issue by using these new representations to create link prediction models to expand the available information.

1.2 Relevant Concepts and Technologies

1.2.1 Knowledge Graphs

One may think of Knowledge Graphs as directed heterogeneous graphs created with the intent of representing knowledge bases in a machine interpretable manner. Nodes may (but not bound to) be objects, events, situations, abstract concepts or locations, with the edges between the nodes representing conceptual relationships among the entities.

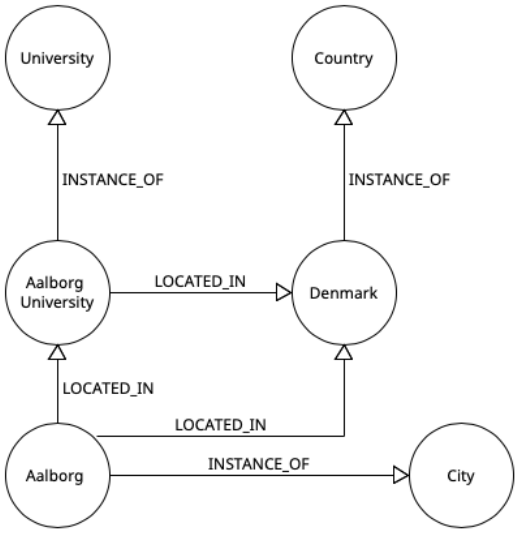

Figure 1.1: Example of a knowledge graph.

Knowledge graphs are often represented as lists of statements such that a statement is: $s = (h, r, t)$ where h refers to the head entity, t represents the tail entity and r represents the edge connecting the two entities. These statements have also been called triplets.

Knowledge graphs are often backed by predefined ontologies. Ontologies define entities and relationships referenced in the list of statements. serving as their explicit schema, making the parsing of and work with these graphs easier. One may also think of knowledge graphs as the combination of the statements, and the relevant ontologies.

Some use cases of knowledge graphs include Google's enhanced contextual response to search queries [30] (using their internal knowledge graph) or a more recent example: researchers have been experimenting with augmenting large language models with knowledge graphs [24], in order to ensure factual answers.

Examples of publicly available knowledge graphs are: Wikidata [33] a generic knowledge graph backing the rest of the Wikimedia ecosystem, or WN18 dataset parsed from WordNet introduced by [3] and is commonly used to evaluate the performance of various graph machine learning models.

1.2.2 Graph Representation Learning

Graph Representation Learning is a research field dedicated to creating various methods of embedding nodes of a graph into a low-dimensional vector space that may be used to perform various downstream tasks such as graph and node classification or link prediction.

These GRL models rely on an encoder-decoder model. "The intuition behind the encoder-decoder idea is the following: If we can learn to decode high-dimensional graph information—such as the global positions of nodes in the graph

or the structure of local graph neighborhoods—from encoded low-dimensional embeddings, then, in principle, these embeddings should contain all information necessary for downstream machine learning tasks" [16]

In other words, while downstream tasks consume the vector representation, the decoder is used to create a well-trained encoder model.

The encoder may create either shallow embeddings or deep embeddings. Shallow embedding techniques are generally simpler and faster to train but may struggle to capture highly complex patterns and hierarchical relationships within the graph.

Example of shallow embedding methods include: Node2Vec [12] or DeepWalk [26]

Deep embedding methods are commonly some variation of Graph Neural Networks detailed in section 1.2.3

1.2.3 Graph Neural Networks

The challenge in creating encoding models that capture deep insight into graph structures is that graph structures are inherently variable [15].

For example, if one was to build a model that categorizes social networks using classical tools such as Convolutional Neural Networks or Recurring Neural Networks, The model would be restricted to graphs with a set number of nodes.

Graph neural networks, which use the concept of neural message passing, are used to address the above issue and leverage the structure of graphs. The recent state-of-the-art knowledge graph completion models are commonly based on some neural message passing framework [25] due to their inherent ability to capture deeper neighborhood structures.

Figure 1.2 illustrates the basic concept of message passing. In a message-passing layer, each neighboring node exchanges its features and encodings with others. For each node, these incoming messages are aggregated to form a new encoding for that node. Depending on the architecture, several message-passing layers may be stacked. By the end of this process, nodes will represent a combination of their initial features and the features of their local neighborhood.

1.2.4 Wikidata

"Wikidata is a free, collaborative, multilingual, secondary knowledge base, collecting structured data to provide support for Wikipedia, Wikimedia Commons, the other wikis of the Wikimedia movement, and to anyone in the world."

"The Wikidata repository consists mainly of items, each one having a label, a description and any number of aliases. Items are uniquely identified by a Q followed by a number, such as Douglas Adams (Q42). Statements describe detailed

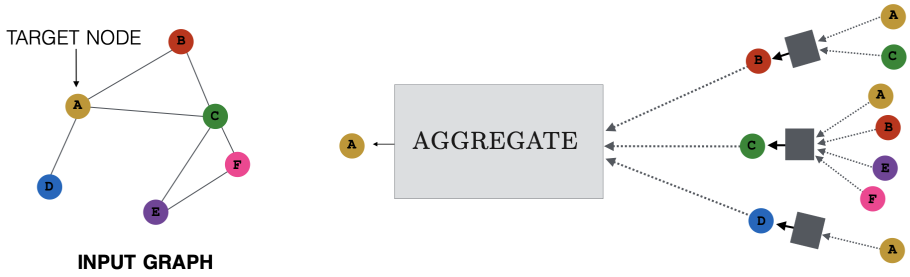

Figure 1.2: A figure from the book Graph Representation Learning [15] showcases the general intuition behind neural message passing.

characteristics of an Item and consist of a property and a value. Properties in Wikidata have a P followed by a number, such as with educated at (P69)." [33]

As detailed later in section 2.1, the initial dataset that forms the basis of this thesis contains some nodes that have inferred Wikidata IDs. Therefore, Wikidata will be a potential source for enriching the dataset and creating a denser network.

Moreover, a secondary object of this thesis is to represent Kitāb Mu'jam al-Buldān in a consumable knowledge graph format. Therefore, it is easy to argue that using Wikidata's ontology will streamline such efforts.

1.2.5 Evaluation Metrics

This section details the commonly used metrics in link prediction literature. The usual environment for model evaluation involves the generation of synthetic negative triplets. The need for negative is explained in subsection 3.1.4 These negative triplets, alongside the positives are then ranked based on some score

Notationally, in the next few sections Q refers to a set of triples $s = (h, r, t)$ where $s \in Q$ [7]

Rank Score

Simple integer value, the rank of a triplet in a vector of triplets.

Mean Rank

The average rank score of a vector of positive triplets in Q

$$MR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_{(h,r,t)_i}$$

Mean Reciprocal Rank

Similar to the mean rank, but instead of averaging triplet ranks, the reciprocal ranks are averaged. This should make the metric more robust against outliers [7].

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_{(h,r,t)_i}}$$

Hits@K

The ratio of how many positive triplets make it to the top K list of triplets when ranked against negative triplets.

$$Hits@K = \sum_{i=1}^{|Q|} 1 \text{ if } rank_{(h,r,t)_i} \leq K$$

In this thesis $K = 1, 3, 5$

1.3 Kitāb Mu'jam al-Buldān Dataset

Kitāb Mu'jam al-Buldān [14] "Dictionary of Countries" was written by Yāqūt Shihāb al-Dīn ibn-Abdullāh al-Rūmī al-áamawī between 1224 and 1228.

The Gazetteer is a "comprehensive index of places and places descriptions, mainly in the Muslim World ... he (Yāqūt) depicted a semi-anachronistic look at the Muslim Caliphate in the 7th-10th centuries " [36]

At the time of writing, there is no exact number for the places detailed in the book, but there are at least 12,400 entries. Some projects, however, distinguish multiple places if they are mentioned within a different entry, lacking their own dedicated one. [29]. Example entries are shown on figure 1.3

Translation	Yāqūt's syntax
and through all cities before Jayhon river	وما يتخلل ذلك من المدن التي دون نهر جيحون
and some people may add Khawarizem district to that (Khurāsān)	ومن الناس من يدخل أعمال خوارزم فيها
and (the people also say) it is a part of Mā Warā' al-Nahr ('beyond the river', transoxania), but that is not correct	ويعد ما وراء النهر منها وليس الأمر كذلك

Figure 1.3: original, Arabic entries from Kitāb Mu'jam al-Buldān with their corresponding English translations [36]

.

1.3.1 Parsed Dataset - Places

The layout of the gazetteer is highly structured. Therefore, researchers were able to create a rule-based method to parse and expose the data in a tabular database [36]. This database serves as the primary data source for this thesis.

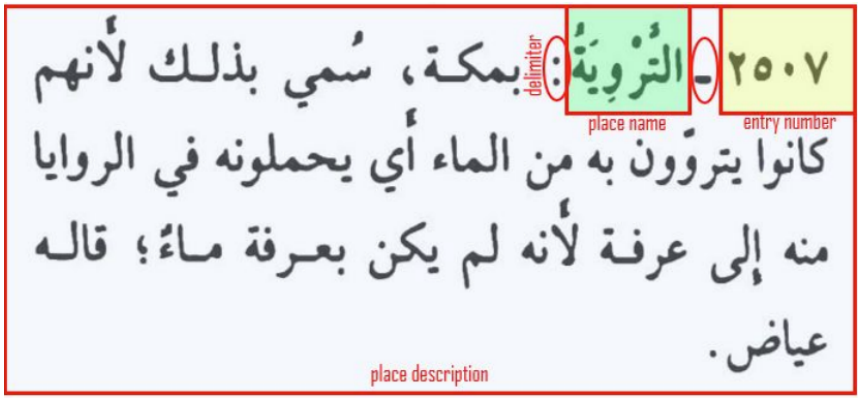

Figure 1.4: Example of a Mujam al-Buldān entry [36]

.

The primary, parsed datasource provided the following information:

- Latitude
- Longitude
- Wikidata ID
- al-Turayyā ID [29]
- Name (Arabic)
- Name (English)
- Type (lower hierarchy)
- Type (upper hierarchy)
- Metropolitan ID (reference to another place in the dataset)
- District ID (reference to another place in the dataset)
- Provincial ID (reference to another place in the dataset)

The only fields guaranteed to have data were the Arabic name and the unique identifier assigned based on the corresponding al-Turayyā gazetteer [29].

al-Turayyā ID

The IDs correspond to the database entries partially backing the al-Turayyā gazetteer project. Namely, al-Turayyā used the same IDs to identify the OCR scanned entries. A small caveat is that [36] occasionally parsed multiple place descriptions from the same entry, therefore, these entries have additional suffixes after the al-Turayyā ID to keep them unique.

Wikidata ID

Some more well-known and easily identifiable place entries, such as Baghdad, were already enriched with Wikidata [33] IDs. This information was initially used to build some graphs, but later iterations skipped its use due to their unreliability. As an example, the entry for Mecca, which is a highly central and important node, had the Wikidata ID of Q3289054 which refers to a city in the United States, not the Saudi Arabian city.

Categories

The rule-based parsing system also attempts to assign a category to each place entry. The categories are selected from a pre-defined hand-verified list.

The categories are split into a two-level hierarchy. For example, the level one category "town" has multiple sub-categories, like village, small town, neighboring villages, or abodes.

However, not all entries will necessarily have a secondary category. In total, there are 96 distinct categories; they are available in the Appendix 8.1.

1.3.2 Parsed Dataset - Distances

The other important block of data available in the parsed dataset are the distances. Over a thousand entries parsed from the original book express some spatial relationship between two places. The dataset this thesis works with already contains this information in kilometers. However, it is important to remember that the kilometer values provided vary significantly in accuracy. This is because the original entries defined distance in terms of various non-standard methods such as days of walking, traveling on camelback, and so on.

Chapter 2

Building the Initial Graph and its Analysis

2.1 Building the Graph

Within this thesis, two target edge categories are valuable to predict: distance and hierarchy. Therefore, creating an initial investigative graph containing only this information is reasonable. The graph will contain only place nodes, one node for each unique place ID. For simplicity, the distance edges will not yet be binned, and each hierarchical relationship will explicitly be defined instead of relying on meta-paths. The database engine storing the graph will be Neo4J [21], populated by a Python script, while the analysis will be mainly done with Gephi [1].

While building the graph, it became apparent that the number of disconnected nodes is exceedingly high (6289 nodes are orphans out of 14863 total nodes); therefore the decision was made to only consider connected nodes as part of the graph, especially since the link prediction methods require at least some connections.

2.2 Initial Analysis

2.2.1 Network Density and Connectivity

Even with the disconnected nodes removed, the graph appears to be extremely sparse. This is apparent from the density, which may be calculated as follows:

$$D = \frac{2E}{N(N-1)}$$

With 8574 nodes and 10790 edges, the density is 0.00029. Considering that a graph's density is such that: $(0 \leq density \leq 1)$, this graph is rather disconnected.

2.2.2 Average Degree and Degree Distribution

The average node degree is 1.258, reinforcing the perception that the graph is sparse. This low density may cause problems down the line, as link prediction methods commonly perform better on denser graphs

The degree distribution (figure 2.1) shows that there are a few nodes in the graph that are exceedingly central (upwards of 300 edges). After some investigation, it was found that these nodes generally correspond to large medieval cities or countries such as Baghdad, Alexandria or Yemen. Besides these highly central nodes, most nodes (6000+) only have a single edge.

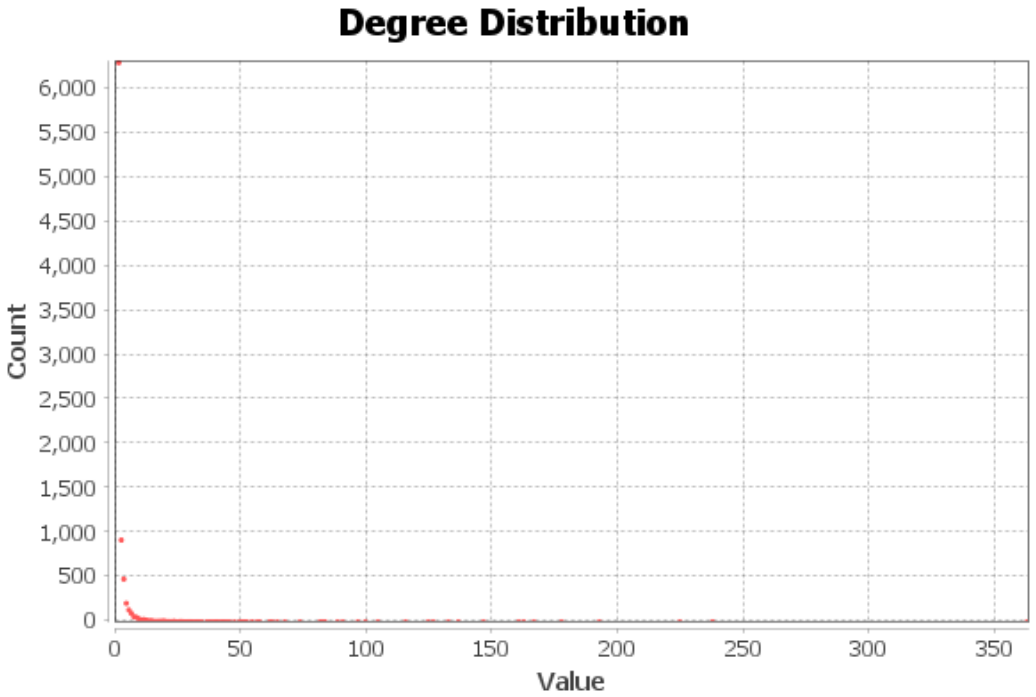

Figure 2.1: Degree Distribution of the initial graph.

2.2.3 Modularity

Using Gephi's [1] implementation of the Louvian method [2] with a resolution of 5.0, 217 communities were found. However, the high number of communities is only indicative of a highly disconnected graph, with many small, isolated islands. In practice, eight large groups account for 94.62% of the nodes. The modularity breakdown is shown on table: 2.1

Figure 2.2 shows that the detected, large communities are fairly easy to visually distinguish. Moreover, it shows that the highly central nodes (visualized by a larger

Name	Population	Color on Figure 2.2
Group 1	22.26%	Pink
Group 2	17.46%	Green
Group 3	15.13%	Blue
Group 4	9.41%	Black
Group 5	9%	Orange
Group 6	8.69%	Red
Group 7	6.85%	Yellow
Group 8	5.82%	Cyan

Table 2.1: Communities detected by the Louvain method

diameter) are spread around the communities fairly evenly.

.

2.2.4 Unexpected Patterns

While working with the parsed dataset, some odd patterns were found. First, distance triplets were found, where both the head and tail entities were the same. At the recommendation of the head, author of [36], these triplets were discarded.

Secondly, during manual observation of the data, "odd" hierarchical patterns were observed; that did not fit to the classical strict, unidirectional layout of other similar structures, shown on figure 2.3 and 2.4 .

These patterns were extracted and forwarded to the MEHDIE researchers using the following Neo4J query:

```
MATCH (n)-[:{r1}]->(m)-[:{r2}]->(n) RETURN n,m
```

Where r1 and r2 correspond to all potential pairwise combinations of the edge types: **wd_P131_METROPOLITAN**, **wd_P131_DISTRICT** and **wd_P131_PROVINCE**

2.3 Creating a Knowledge Graph

A secondary focus of this thesis is to transform the parsed dataset into an easily interpretable and information-rich knowledge graph to serve further downstream research as part of MEHDIE [28].

To create such a graph, it is beneficial to rely on existing ontologies, to increase inter-compatibility and standardization. Specifically, this new Kitāb KG's ontology extends Wikidata's ontology in terms of relationship and category types.

In terms of place node ontology, however, the KG was constructed by entirely using internal IDs.

Figure 2.2: Graph visualization showing the communities listed on table 2.1

As mentioned earlier in section 1.3.1, some Wikidata equality information was already available in the parsed dataset. This information was also used to expand the graph, and moreover, could potentially be used to help training better models for link prediction.

2.3.1 Hierarchical Edges

One could take two different approaches when constructing the hierarchical edges. Either the graph could explicitly contain all known hierarchical edges (figure 2.6), or just the next level up in the hierarchy (figure 2.5).

It seems that the larger, well-known graphs do not explicitly define all levels of hierarchy for any given place. Therefore, this knowledge graph was also constructed in a compositional manner.

Another caveat is that Wikidata's ontology does not define various hierarhical edge types and instead relies on the aggregating **P131** edge type (located in the administrative territorial entity).

For the purposes of this KG, it is more beneficial to split this type into three distinct types, namely: **wd_P131_METROPOLITAN**, **wd_P131_DISTRICT** and **wd_P131_PROVINCE**

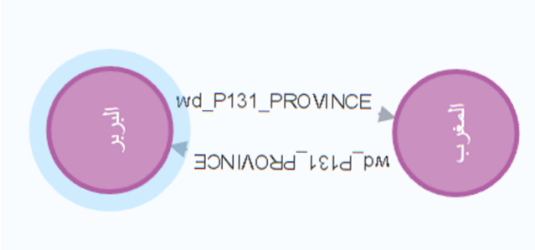

Figure 2.3: Unusual hierarchical pattern where two nodes reference each other as their province.

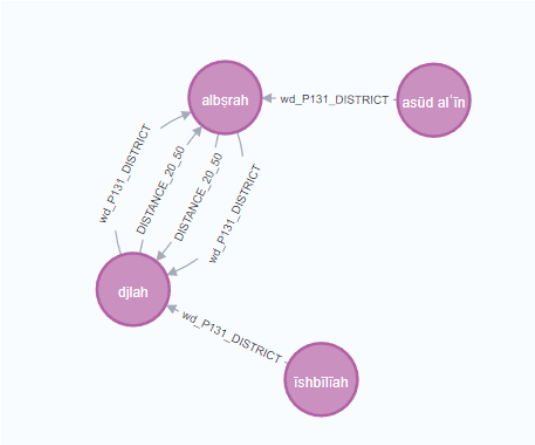

Figure 2.4: Unusual hierarchical pattern where two nodes reference each other as their district.

2.3.2 Distance Edges

In the parsed Kitāb dataset, there is a large amount of distance information between various place pairs. However, by simply inserting these edges into the KG, without the numerical distance values, the informational value of the graph would be significantly diminished.

However, the simple triplet structure of knowledge graphs does not allow the definition of features on edges or nodes. Normally, the place nodes would have a corresponding point literal in the triplet store, and the distance between the two points could be extracted using some geospatial calculation.

Unfortunately, in this case, there are very few nodes with known coordinates. As a compromise, the **Distance** edge types were split into multiple edge types, each type representing a specific range.

These binned edge types are also shown on table: 2.2

Finally, as the distance is obviously symmetric, these triplets were defined twice by switching the head and tail entities.

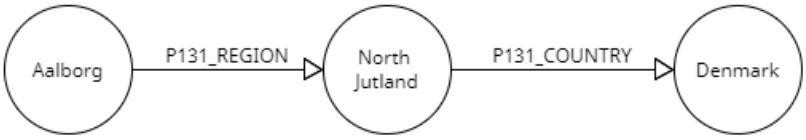

Figure 2.5: Small example of a graph where not all known hierarchical relationships are defined.

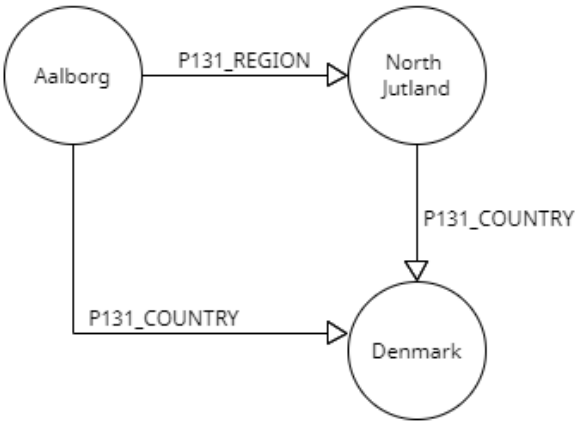

Figure 2.6: Small example of a graph where all known hierarchical relationships are explicitly defined.

2.3.3 Categories

There is a significant overlap between the categories defined in the parsed Kitāb dataset and the available conceptual entities in Wikidata, wherever available, the Wikidata Ontology ID was used.

For example, the city category was mapped to the Wikidata entity *Q515*. However, there were some very specific categories in the parsed dataset, such as kasbah (city center), which is distinct from just a generic kasbah *Q89468*

These special category entities were defined in the Mehdie ontology. E.g., kasbah (city center) became *mehdie_11*

Since the categories are hierarchical, this hierarchy was captured using the **P279** (subclass of) edge type, with the place nodes referencing the lowest available category level. Between Place and Category entities, the **P31** edge type was used.

2.3.4 Scraping Wikidata

As mentioned earlier, in the parsed Kitāb dataset, a number of places had a Wikidata ID defined. This introduced the opportunity to enrich the knowledge graph with extra Wikidata information.

The full list of scraped edge types are listed on table 2.2

One could make the argument that if a place node has a corresponding Wikidata entity, then it is bad practice to keep both nodes in the graph.

However, considering that these ID assignments may be incorrect, it is safer to link the corresponding. MEHDIE and Wikidata entities with a **P460** - "said to be the same as".

Table 2.2: Edge types used in the Kitāb Knowledge Graph

Name	Description
wd_P131_METROPOLITAN	*head* is in the metropolitan area of *tail*
wd_P131_DISTRICT	*head* is in the district area of *tail*
wd_P131_PROVINCE	*head* is in the provincial area of *tail*
DISTANCE_0_5	*head* and *tail* are closer than 5km to each other
DISTANCE_5_10	*head* and *tail* are between 5 and 10 kilometers from each other
DISTANCE_10_20	*head* and *tail* are between 10 and 20 kilometers from each other
DISTANCE_20_50	*head* and *tail* are between 20 and 50 kilometers from each other
DISTANCE_100_200	*head* and *tail* are between 100 and 200 kilometers from each other
DISTANCE_200_500	*head* and *tail* are between 200 and 500 kilometers from each other
DISTANCE_500_1000	*head* and *tail* are between 500 and 1000 kilometers from each other
DISTANCE_1000_inf	*head* and *tail* are over 1000 km away from each other
P17	Wikidata type: "Country"
P206	Wikidata type: "located in or next to body of water"
P706	Wikidata type: "located in/on physical feature"
P361	Wikidata type: "part of"
P402	Wikidata type: "OpenStreetMap relation ID"
P11693	Wikidata type: "OpenStreetMap node ID"
P625	Wikidata type: "coordinate location"
P460	Wikidata type: "said to be the same as"

Chapter 3

Knowledge Graph Embedding Methods

The first broad classification of link prediction methods explored in this thesis are the knowledge graph embedding methods (KGE). Since graph neural networks generate node embeddings as well, the terminology is somewhat ambiguous. The following few paragraphs will attempt to distinguish the class of methods detailed in this chapter.

KGE methods attempt to position each node in a vector space (figure 3.1). In a well-trained model, nodes from the same neighborhood will be placed near each other within the vector space. Moreover, these positions are strictly unique, i.e., two distinct nodes cannot occupy the same place. From this, it naturally follows that KGE methods are strictly transductive and cannot be generalized.

Fortunately, a transductive setting is not necessarily limiting for the purposes of this thesis. as all the link-predictions tasks are limited to the domain of places mentioned in Kitāb Mu'jam al-Buldān.

KGE methods are generally more efficient and easily more scalable than their GNN counterparts.

.

3.1 General Architecture

Generally, KGE models, at least within the context of this thesis, fit into the same generic architecture (figure 3.2).

.

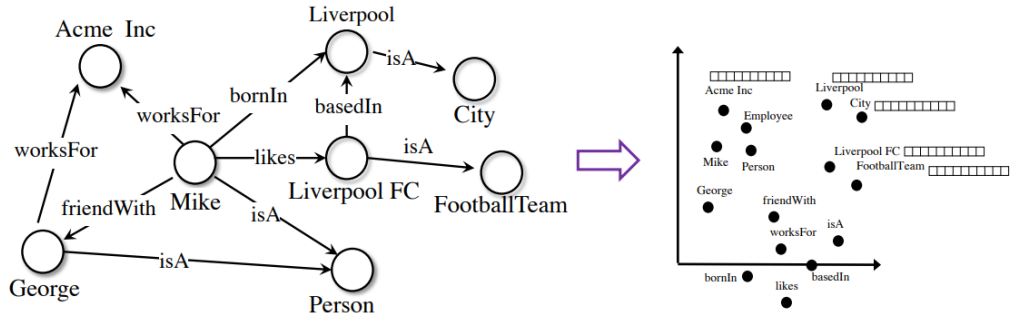

Figure 3.1: Visualization of Vector Space Embedding [8]

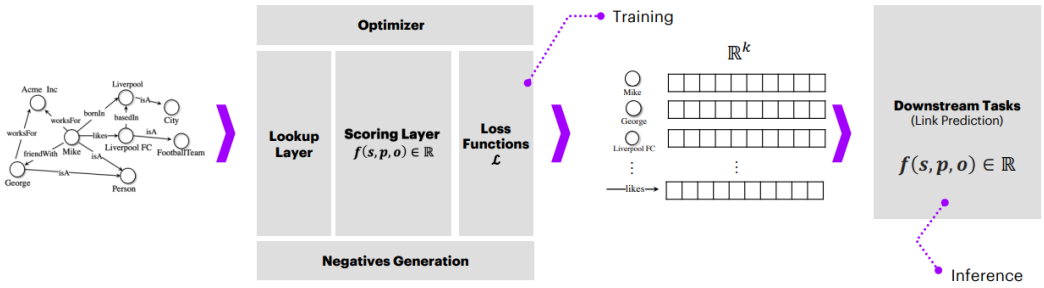

Figure 3.2: Generic KGE architecture [8]

3.1.1 Lookup Layer

The Lookup layer just simply functions as a dictionary between the individual nodes and relationships and their respective embeddings.

3.1.2 Scoring Layer

Colloquially, the heart of the architecture, the scoring layer takes the encoding of each member of the triplet and assigns a score to the whole statement. The higher the score, the more likely it is that the statement is a true statement. The specific scoring functions are detailed in section 3.2

3.1.3 Loss Function

As with all other neural models; the generic KGE architecture relies on a loss function, which is necessary to optimize the model. In the KGE section of this thesis, two loss functions were used.

Pairwise, Max-margin Loss

This function penalizes the model if the score assigned by scoring function f_{model} to a positive triple t^+ selected from the set of positives $\mathcal{G}$, is lower than the score assigned to a negative triple t^- selected from the set of corruptions $\mathcal{C}$ by margin γ

$$\mathcal{L}(\Theta) = \sum_{t^+ \in \mathcal{G}} \sum_{t^- \in \mathcal{C}} max(0, [\gamma + f_{model}(t^-; \Theta) - f_{model}(t^+; \Theta)])$$

Negative Log-Likelihood Loss

Another commonly used loss function, $y \in -1, 1$ dependent on whether the statement is positive or negative.

$$\mathcal{L}(\Theta) = \sum_{t \in \mathcal{G} \cup \mathcal{C}} log(1 + exp(-y\, f_{model}(t; \Theta)))$$

3.1.4 Negatives Generation

A very distinctive feature of the domain of knowledge graph link prediction is the (usual) lack of true negative data points. Let's consider the domain of image recognition. If someone were to train a model to detect the presence of a cat in a photo, the negative data points could be any photos that do not contain a cat. However, in the case of knowledge graphs, there are no negative facts. A knowledge graph may contain the triplet <*London,CapitalOf,UnitedKingdom*>

And even though for a human reader, this automatically means that <*London, CapitalOf, Denmark*> is a false statement, a normal knowledge graph will not contain such information explicitly. Therefore, link prediction methods commonly rely on some form of synthetic negative triplet generation.

While there have been many strategies proposed [17], in this thesis, a very simple method was chosen. Given a true statement $s = (h, r, t)$, a negative sample will be generated by corrupting t by randomly replacing it with another node from the graph. Of course, corruption is done with consideration of the ground truth triplets.

This has a negative effect under the open world assumption of potentially labeling unknown true positives as negative data points [35]. However, due to its simplicity, this approach has the benefit of avoiding any potential bias in the training and being computationally efficient.

3.1.5 Optimizer

Same concept as in other machine learning architectures, in this thesis two optimizers were used: Adam [18] and AdaGrad [10]

3.2 Embedding Methods

3.2.1 Translating Embeddings (TransE)

Commonly used benchmark, and one of the first KGE models. This approach attempts to model relationships as translations between two points (nodes) in a vector space

Given a statement $s = (h, r, t)$, ideally h and t should be close to each other in the vector space, with the difference being the translation of the relation r [3].

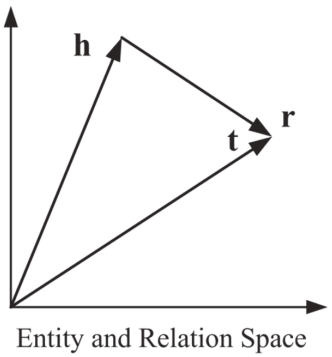

Figure 3.3: Visualization of TransE [13]

.

3.2.2 Relational Rotation Embeddings (RotatE)

"The RotatE model maps the entities and relations to the vector space and defines each relation as a rotation from the source entity to the target entity." [31]

The benefit of RotatE over TransE lies in expressiveness. It is able to model more relationship patterns, such as symmetry, antisymmetry, inversion and composition.

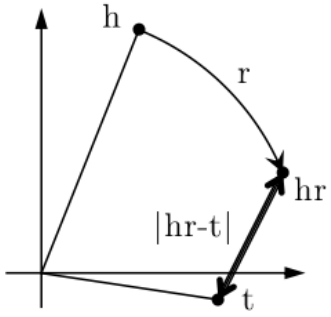

Figure 3.4: Visualization of RotatE [31]

.

3.2.3 Holographic Embeddings (HolE)

Holographic Embedding [22] is a model for learning representations of entities and relationships in a knowledge graph. It uses the principles of holography and circular correlation to combine entity embeddings in a way that captures rich interactions between entities and relations.

The main benefit of HolE over other approaches is the increased modeling power.

3.2.4 The Bilinear Diagonal DistMult Model (DistMult)

DistMult is a tensor factorization based model introduced in [34] that uses, trilinear dot product as its scoring function.

However, a big limitation of DistMult is that it cannot model asymmetric relationships.

3.2.5 Complex Embeddings (ComplEx)

ComplEx [32] is similar to DistMult in using the dot product of two vectors to calculate the score. However, it uses the Hermitian dot product, which is able to handle asymmetrical relationships.

Moreover, instead of using real-valued vectors, DistMult uses a complex vector space for both entities and relations. Allowing to capture richer information about the graph at the cost of increased computational cost.

3.2.6 Benchmark Comparison of KGE Methods

Table 3.1 shows the performance of the above-detailed KGE methods on the FB15k benchmark dataset.

	MRR	Hits@1	Hits@10
TransE	0.463	0.29	0.75
RotatE	0.797	**0.746**	0.884
HolE	0.524	0.4	0.73
DistMult	**0.841**		**0.914**
ComplEx	0.84	0.692	0.75

Table 3.1: Performance of Discussed KGE Methods on the FB15k Benchmark Dataset. Datasources [4, 22]

3.3 Methodology

In the initial KGE experiments, the dataset generated in section 2.3 was used as data for the models. The dataset was split into train, test and validation dataset in a 80%, 10% and 10% ratio. The split was done in a transductive manner, such that there are no, previously unseen nodes in the test and validation dataset.

The implementation of the models were done using the Python library—Ampligraph [7]

To find the best performing model and the most optimal hyperparameters, a grid search was performed, with the MRR score being the basis of comparison.

For each method detailed in section 3.2, the following hyperparameters were tuned:

- **Batch size:** 128, 256, 512, 1024, 2048
- **Epochs:** 5, 25, 50, 100, 200, 250
- **Eta:** 5, 10, 15 (*eta specifies the number of corruptions to generate per each positive*)
- **Vector embedding size:** 50, 100, 150, 200
- **Loss function:** pairwise, nll
- **Optimizer:** AdaGrad, adam

The best performing model and hyperparameter combinations are shown on table 3.2.

	batch size	**epochs**	**k**	**eta**	**loss**	**optimizer**
TransE	2048	200	150	5	pairwise	adam
RotatE	1024	90	150	15	nll	adam
HolE	128	80	200	5	pairwise	adam
DistMult	128	60	200	5	pairwise	adam
ComplEx	128	50	200	5	pairwise	adam

Table 3.2: Best performing KGE hyperparameters

3.4 Results

Unfortunately, even with the tuned hyperparameters, the KGE models produced subpar results (table 3.3). With the highest MRR being 0.18 produced by HolE, DistMult and ComplEx with DistMult slightly under-performing in hits@10 compared to the other two.

	MRR	Hits@10	Hits@5	Hits@1
TransE	0.11	0.24	0.14	0.04
RotatE	0.16	0.21	0.17	0.13
HolE	0.18	0.22	0.19	0.16
DistMult	0.18	0.21	0.18	0.16
ComplEx	0.18	0.22	0.19	0.16

Table 3.3: Performance of the hyperparameter tuned KGE models on a previously unseen test dataset

After some investigation, two apparent issues were found. First, the kilometer values of distance edges range from 1 kilometer to over 1000 kilometers. Therefore, all the distance edges were binned into ranges.

Second, the model was incorrectly predicting not explicitly defined hierarchical information. Since this information was available in the source dataset, it made sense to modify the training data to include all available hierarchical information instead of expecting the model to learn the hierarchical meta-paths.

Finally, reverse edge types were introduced to increase the graph density. These edges were only inserted after the train / test / validation split to avoid leakage.

The final edge composition in the data is shown on table 3.4 (excluding the reverse edge counts as they are obviously equal to their counterparts.).

With the new training dataset, the hyperparameter tuning was repeated for each method. Unfortunately it appears that the new dataset did not result in increased model performance. In fact all across the board, the performance of the models decreased (table 3.5).

However, it is important to point out that some of the reduction in performance may be attributed to the increased difficulty in distance prediction.

Edge Type	**Count**
DISTANCE_0_5	50
DISTANCE_5_10	48
DISTANCE_10_20	254
DISTANCE_20_50	690
DISTANCE_50_100	340
DISTANCE_100_200	294
DISTANCE_200_500	142
DISTANCE_500_1000	6
DISTANCE_1000_inf	8
wd_P17	3507
wd_P31	13707
wd_P131_DISTRICT	3124
wd_P131_METROPOLITAN	4695
wd_P131_PROVINCE	4252
wd_P279	66
wd_P206	223
wd_P361	287
wd_P706	57

Table 3.4: Edge Count of the Second Iteration of the KGE Experiments

	MRR	Hits@10	Hits@5	Hits@1
TransE	0.09	0.24	0.12	0.037
RotatE	0.07	0.14	0.09	0.039
HolE	0.12	0.19	0.16	0.09
DistMult	0.16	0.22	0.19	0.12
ComplEx	0.16	0.22	0.2	0.13

Table 3.5: Performance of the hyperparameter tuned KGE models on a previously unseen test dataset

Chapter 4

Experiments with Neural Bellman-Ford Networks

4.1 Introduction

After the lackluster result from the translation methods detailed in chapter 3, the need for experimentation with other approaches became apparent.

At the time of writing, the state-of-the art multi-relational link prediction methods [25] are generally based on, and expand on the message passing, Graph Neural network architecture.

Specifically, the top performing model, according to the Papers with Code benchmark on FB15k-237 link prediction [20] are the Neural Bellman-Ford Networks (NBFNet).

Therefore, from now on, this thesis will focus on training an NBFNet model on the Kitāb Mu'jam al-Buldān dataset.

4.2 Brief Overview of Neural Bellman-Ford Networks

4.2.1 Generalization of Graph Heuristics

The first intuition behind NBFNet is that many of the traditional graph heuristics such as Katz-Index, Personalized PageRank [23] or Graph Distance can be generalized into a *multiplication* and a *summation* step.

For example, to find the shortest Graph Distance between two nodes, first we count the number of hops (generalized multiplication step by count) and then we select the path with the fewest hop (generalized summation via the min function)

The second intuition behind NBFNet is that traditional link prediction methods that rely on encapsulating local neighborhoods [19, 11] perform random walks between the source and the target node.

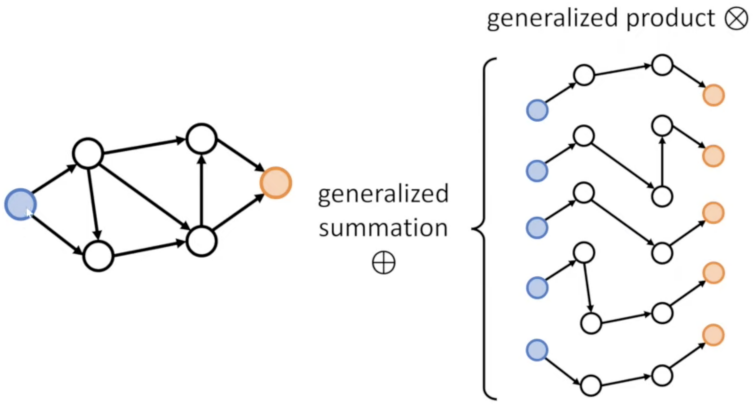

Figure 4.1: Generalized Graph Heuristic [38]

Instead of these random walks, the pair representation can be formulated as "a generalized sum of path representations between u and v with a commutative summation operator $\bigoplus$" [39].

Within this summation, each path "is defined as a generalized product of the edge representations in the path with the multiplication operator" [39].

4.2.2 Generalized Bellman-Ford Algorithm

Calculating such metrics for each node pair is computationally expensive due to its exponential nature. As a solution, the NBFNet performs these calculations as part of a generalized Bellman-Ford algorithm, as that algorithm is highly parallelizable (figure 4.2).

$$\boldsymbol{h}_q^{(0)}(u,v) \leftarrow \mathbb{1}_q(u=v)$$
$$\boldsymbol{h}_q^{(t)}(u,v) \leftarrow \left(\bigoplus_{(x,r,v)\in\mathcal{E}(v)} \boldsymbol{h}_q^{(t-1)}(u,x) \otimes \boldsymbol{w}_q(x,r,v) \right) \oplus \boldsymbol{h}_q^{(0)}(u,v)$$

Figure 4.2: Generalized Bellman-Ford Algorithm [39]

4.2.3 Neural Bellman-Ford Networks

Finally, to increase the potential of the models and detach from classical, hand-crafted methods, the creators of NBFNet replaced the generalized *summation* and *multiplication* operators with neural functions.

A Neural Bellman-Ford Network has three neural functions.

The INDICATOR function

"The INDICATOR function initializes a representation on each node, which is taken as the boundary condition of the generalized Bellman-Ford algorithm." [39]

It replaces the $\mathbf{h}_q^{(0)}(u,v) \leftarrow \mathbb{1}_q(u=v)$ step, seen on figure 4.2

The MESSAGE function

"The MESSAGE function replaces the binary multiplication operator $\otimes$" [39]

The AGGREGATE function

"The AGGREGATE function is a permutation invariant function over sets that replaces the n-ary summation operator $\bigoplus$ ... one may alternatively define AGGREGATE as the commutative binary operator $\bigoplus$ and apply it to a sequence of messages." [39]

The combination of the three neural function are shown on figure 4.3.

$$\boldsymbol{h}_v^{(0)} \leftarrow \text{Indicator}(u,v,q)$$
$$\boldsymbol{h}_v^{(t)} \leftarrow \text{Aggregate}\left(\left\{\text{Message}\left(\boldsymbol{h}_x^{(t-1)}, \boldsymbol{w}_q(x,r,v)\right) \middle| (x,r,v) \in \mathcal{E}(v)\right\} \cup \left\{\boldsymbol{h}_v^{(0)}\right\}\right)$$

Figure 4.3: Neural Bellman-Ford Network Architecture [38]

Finally, the learned vector representation is then fed into a multi-layer perceptron to predict the probability of a tail node given a head node and a query relationship (figure 4.4)

$$p(v|u,q) = \sigma\left(f\left(\boldsymbol{h}_q(u,v)\right)\right)$$

Figure 4.4: Final Prediction Step in NBFNet [38]

4.3 Methodology

4.3.1 NBFNet Function Selection

The training of the model was mainly done using the Pyg implementation of NBFNet, a codebase written by the original authors of [37]. For the AGGREGATE function, the principal neighborhood aggregation (PNA) [6] architecture was used, while for the MESSAGE function, the DistMult [34] function was used. This combination was picked as it was the highest-performing setup in the original paper. The model was trained with an Adam optimizer, with a learning rate of 5.0e-3 using 64 item batches for 20 epochs.

4.3.2 Experiment Data

For this experiment, a simplified view of the knowledge graph was used. This approach was selected due to the relatively low number of secondary edges derived from Wikidata which may have led to the introduction of damaging biases. Therefore, in the NBFNet experiment, only place nodes were used, with dense hierarchical edges and binned distance edges. In total, there were 25,982 triplets, which were split in a 0.8/0.1/0.1 for train, test, and validation datasets in a transductive manner.

4.4 Results

Unfortunately, the performance of the trained NBFNet model (table 4.1) on the previously unseen test dataset was comparable to the performance of the initial KGE models. It performed slightly worse in terms of MRR, but it achieved the highest score on Hits@10 and Hits@5

	MRR	Hits@10	Hits@5	Hits@1
TransE	0.11	0.24	0.14	0.04
RotatE	0.16	0.21	0.17	0.13
HolE	0.18	0.22	0.19	0.16
DistMult	0.18	0.21	0.18	0.16
ComplEx	0.18	0.22	0.19	0.16
NBFNet	0.17	**0.27**	**0.21**	0.11

Table 4.1: Performance of the KGE and NBFNet models

A potential approach to improve the model's performance is discussed in the next chapter.

Chapter 5

Pretrained WorldKG NBFNet Model

The previous experiments in chapter 4 and chapter 3 show that regardless of the class of approach used, there is a fundamental barrier when working with this thesis' knowledge graph. Namely, there is a high chance that the sparsity of the graph prevents any potential model from properly being able to generalize.

As a solution, this thesis proposes an alternate approach. Instead of trying additional models, the previously used NBFNet model could be pre-trained on a denser graph. Since the Kitāb KG, ultimately, is just the graph representation of a geospatial area, it is relatively easy to create a synthetic dataset that mimics it. Moreover, creating such a dataset allows for the introduction of specific distance edge biases, not generally present in Kitāb, that could provide valuable information to researchers. Pre-training the NBFNet model on such a dataset could allow it to generalize significantly better.

A perfect source of data for creating such a synthetic dataset is WorldKG [9] - a research project that parsed OpenStreetMaps [5] data into triplestore data. The desired synthetic dataset could be created by sampling WorldKG and converting the relevant triples to use this thesis' ontology.

5.1 Constructing The Dataset

To construct the synthetic dataset, useful patterns need to be created first. While it is entirely possible to create a fully connected graph, it would not represent the biases found Kitāb well. Nodes like Alexandria are central because Yâqût found it important to define places in relation to big, central places. Second, the local structures in Kitāb KG exist because Yâqût also found it important to define places in relation to their surroundings. These ideas may be boiled down into the following simple bullet points:

- A nodes's local neighborhood should be well-connected.

- Each node should be connected to some central entity.
- Central entities should be connected to each other.

Following these simple rules, one could create a Kitāb KG-like synthetic dataset with higher density. In theory, the entire WorldKG dataset could be used to create such a synthetic dataset, for practical purposes, in reality, the data was heavily down-sampled.

5.1.1 Selecting Relevant Nodes

In WorldKG, there are over one thousand node types, most of them irrelevant to this thesis' purpose since, for example, there were exceedingly few airports in the medieval arab world. Instead, the categories discussed in subsection 2.3.3 were mapped to their OSM equivalent.

Then, with Baghdad selected as the center point, all WorldKG place nodes (of relevant type) were selected in a 2000 km radius, using the following SPARQL query:

```
SELECT ?subject ?type ?label ?wikidata ?subjectWKT ?distance
WHERE {
    # wkg:21034458 = Baghdad
    wkg:21034458 wkgs:spatialObject ?spatialObject .
   ?spatialObject geo:asWKT ?referenceWKT .

    ?subject rdf:type ?type .
    ?subject rdfs:label ?label .
    ?subject wkgs:spatialObject ?subjectSpatialObject .
    ?subjectSpatialObject geo:asWKT ?subjectWKT .
    OPTIONAL {{ ?subject wkgs:wikidata ?wikidata . }}

    FILTER (?type IN ({types}))
    BIND(geof:distance(?referenceWKT, ?subjectWKT, uom:metre) AS ?distance) .
    FILTER(?distance < 2000000).
}
```

This query returned 238135 distinct nodes. These nodes were then randomly down-sampled to 47627 entities (20%).

5.1.2 Generating Hierarchy Triplets

The only piece of information that was not readily available in WorldKG was the hierarchical information. Fortunately, for each node selected, there is a correspond-

ing geometric point literal. This point can be used to query an OSM Overpass API, such as [27] to get all administrative objects in which the point is contained.

```
is_in({x},{y})->.a;
rel(pivot.a)[boundary=administrative];
out tags center;
```

The above query, written in Overpass API Query Language, among other things, gets the ID, administrative level, and the geographical center of all enclosing administrative boundaries overlapping the parameter point. In OpenStreetMaps, there are 11 levels of administrative hierarchy that try to represent every nation's system. To construct the hierarchical data for the synthetic dataset, the levels were split as follows:

1. $[4, 3, 2] \mapsto$ **wd_P131_PROVINCE**
2. $[7, 6, 5] \mapsto$ **wd_P131_DISTRICT**
3. $[11, 10, 9, 8] \mapsto$ **wd_P131_METROPOLITAN**

The logic was written such that it always tries to pick the level with the highest integer value; this was done to increase granularity in the data.

5.1.3 Generating Distance Edges

In line with the above-detailed rules, for each node, its close neighborhood should be as dense as possible. Therefore, for each node pair, a corresponding great circle distance was calculated. Then, for each node, these distance edges were sampled, such that the further away the nodes were from each other, the less likely to be sampled.

5.1.4 Generating Distance Edges for Hierarchical Nodes

To generate the distance edges between nodes that are higher in the hierarchy (i.e., more important places) all possible pair combinations were selected. These pairs then, without sampling, were written into triplets with the corresponding distance edge type, selected based on the great circle distance.

5.1.5 Generating Category Triplets

In the original query, where all the nodes were selected, their types were also queried. These can be mapped onto the Kitāb KG categories.

Finally, every node is then connected to the initial starting point, Baghdad, based on the great circle distance.

Chapter 6

Select Predictions and Interpretations

Although the performance metrics of the trained models fell short of the expectations. The trained models could still potentially be used to generate valuable information, especially in cases where the model is highly certain about the truth value of a triplet.

Moreover, due to how NBFNet works, the reasonings behind the model's predictions are human-interpretable. Understanding these reasonings may also be used to create hand-made rule-based link prediction algorithms.

Some of these results were also analyzed and evaluated by the first author of [36], the explanations provided also offer an interesting insight into the blind spots of the original rule-based model, and why the NBFNet model struggled in some situations.

6.1 Generating New, Potentially Positive Triplets

To generate previously unknown positive triplets, one needs to generate a set of candidates. A generation strategy is necessary as trying all possible (h, r, t) combinations is computationally expensive with its cost increasing exponentially with the number of nodes and edge types.

In the case of hierarchical edges, first, the nodes with missing hierarchical information were selected as the head entities. Second, all place nodes within 2-hops of the head entity were selected as tail entity candidates.

For the distance-type edges, such a hop limitation would not be beneficial as the kilometer value of the edges ranges from One kilometer to over a thousand kilometers.

Therefore, it was decided to randomly sample nodes with low centrality measurements. This approach has the benefit of potentially providing the highest value information.

Predicting a new edge for an entity that only has 1–2 connections to the rest of the graph increases the average node degree at a larger ratio than the prediction of a new edge for Baghdad, which is already an extremely central node in the graph.

The Ampligraph [7] library exposes several strategies for finding such nodes. These strategies are:

- Entity Frequency
- **Entity connectedness:** Graph Degree, Cluster Coefficient
- **Local Graph Structures:** Cluster Triangles, Cluster Squares

The Local Graph Structures Strategies rely on the idea that well-connected structures are less likely to have missing facts. In this thesis, the Cluster Triangles approach was used to generate triplet candidates.

6.1.1 Hierarchical Triplet Candidates

The general feedback on the predicted hierarchical candidates was that the model is seemingly able to recognize hierarchical ordering between two places but struggles to correctly identify the exact level of hierarchy. A false positive example highlighted in the evaluation was the prediction that المراكب (almrākb), a city in Tunisia was in the province of Tunisia. However, Tunisia is not actually a province; instead, both Tunisia and المراكب are in the province of North Africa. Upon reviewing the graph, it was found that no node has a **wd_P131_PROVINCE** edge with Tunisia as the tail.

When making this prediction, the model's decision could be followed as:

```
weight: 3.33629:
<almrākb, wd_P131_DISTRICT, Tunisia>
```

This is to be interpreted such that the model's main reason for predicting this triplet to be true is the known **wd_P131_DISTRICT** edge between the two entities. This is a rather odd finding, as no similar patterns were found in the training dataset, where the district and the province tail entity of a head node is the same.

6.1.2 Distance Triplet Candidates

The Distance predictions fail in a similar manner, the general feedback was that the model seems to be able to capture some form of semantic relation between places. For example, the highest rated distance predictions are consistently for places that are found in the same administrative area. However, the types of distance edge chosen are often wildly inaccurate. For example, a highly rated false positive triplet is between the Spanish city of Córdoba and the Spanish municipality of Toledo. The model, with high certainty predicted that these two entities should be within 5 kilometers of each other, when in reality they are roughly 230 kilometers away from each other.

The main contributor to this prediction is as follows:

```
weight: 3.27499
<Córdoba, DISTANCE_200_500, Toledo>
```

This is once again an unusual result, as the model predicted a different distance edge based on the existence of another distance edge between the two places. It also highlights a bug in the candidate selection logic.

6.2 False Positive Triplet Detection

Besides generating new potential positive triplets, the training data was fed through the model once again as well. The training triplets with the lowest certainty were then evaluated to check whether they were actually true statements. Surprisingly, unlike in the domain of new triplet prediction, the model appeared to perform much better at recognizing false negative triplets.

For example, the model flagged the **DISTANCE_0_5** relationship between armīah and al-Buheirah as a potential false positive. After reviewing the text snippet, from which these places were parsed from, it was found that there is another place called al-Buheirah.

Another interesting example was the triplet of حقيل, **wd_P131_METROPOLITAN** نهر المعلى. In this case, the text from which these places and relationships were parsed contained a story told by a shepherd. And because the shepherd mentioned one of the places, the rule-based parser picked up on it and generated a

false positive triplet. The model apparently picked up the issue as the two nodes were in a different province.

The general feedback on the highlighted potential false positives was that the model marked a lot of triplets where one of the members had a very generic name like mountain or lake.

Chapter 7

Conclusion

7.1 Potential Areas of Improvement

There are many potential areas for improvement. The largest area is the fact the pre-trained NBFNet model discussed in chapter 5 only hypothetically outperforms the other models, but this hypothesis was not tested.

Second, the section on false positives highlighted many potential issues in the training dataset. Perhaps better results all across the board could have been reached by aggressively filtering out these triplets. In the same manner, the true positive prediction section 6.1 showed how restrictive the distance bins are on the smaller end.

It could also have been interesting to explore rule-based methods or spend more time interpreting the NBFNet results to generate such rules.

7.2 Final Conclusion

This thesis explored various subjects, including knowledge graph construction 2.3, experiments with Knowledge Graph Embedding models (chapter 3), and Graph Neural Networks (chapter 4). The research focused on two main areas of improvement: firstly, modifying or replacing the experimental models, and secondly, continually experimenting with the methods of feeding the generated knowledge graph into the model. Another area of the thesis was the construction of a second, larger synthetic knowledge graph mimicking the original Kitāb KG, to train better NBFNet models (chapter 5).

However, the most promising results came from false positive detection for Kitāb KG's source dataset. It could be argued that the ultimate result of this thesis is an Error Detection model for the rule-based parser introduced in [36]

Chapter 8

Appendix

8.1 Complete List Kitāb Mu'jam al-Buldān Knowledge Graph Categories

```
categories_dict = {
    'mehdie_01': 'islamic longitude',
    'Q3024290': 'descent',
    'mehdie_02': 'two mountains',
    'Q149621': 'district',
    'Q8085493': 'borders',
    'Q39715': 'lighthouse',
    'Q23442': 'island',
    'Q23397': 'lake',
    'Q32815': 'mosque',
    'mehdie_03': 'palaces',
    'Q16560': 'palace',
    'mehdie_04': 'neighboring villages',
    'Q34442': 'road',
    'mehdie_05': 'neighboring castles',
    'Q12819564': 'station',
    'Q47499118': 'date palm orchard',
    'Q8514': 'desert',
    'mehdie_06': 'small abodes',
    'Q131401': "caliphate's country",
    'mehdie_07': 'neighboring forts',
    'Q2472587': 'people',
    'mehdie_08': 'cultivated low terrain',
    'Q133311': 'tribe',
    'Q3957': 'town',
```

```
'Q34876': 'province',
'mehdie_09': 'two rivers',
'mehdie_10': 'small town',
'mehdie_11': 'kasbah (city center)',
'Q34770': 'language',
'Q54050': 'hill',
'Q1907114': 'metropolitan area',
'Q8384057': "yemeni districts",
'Q236371': 'orchard',
'Q580112': 'military district',
'mehdie_12': 'two valleys',
'Q4026570': 'countries',
'Q98929991': 'place',
'Q811979': 'structure',
'mehdie_13': 'inhabited wide valley',
'mehdie_14': 'frontier town',
'mehdie_15': 'neighboring towns',
'mehdie_16': 'neighboring places',
'Q44782': 'port',
'Q133346': 'border',
'Q515': 'city',
'Q30892871': 'inhabited region',
'mehdie_17': 'pole',
'mehdie_18': 'fortified round tower',
'Q1355821': 'frontier',
'Q80018988': 'idol',
'Q330284': 'market',
'Q75520': 'plateau',
'Q1785071': 'fort',
'Q6256': 'country',
'mehdie_19': 'frontier district',
'mehdie_20': 'border sign',
'mehdie_21': 'travelers station',
'mehdie_22': 'roof shed',
'mehdie_23': 'low place',
'Q6617100': 'yemeni district',
'Q5620504': 'provinces',
'Q8502': 'mountain',
'Q4022': 'river',
'Q33829': 'population',
'Q6493590': 'castles',
```

```
    'Q11081619': 'land',
    'mehdie_24': 'non-arab province',
    'mehdie_25': 'inhabited valley',
    'Q12518': 'tower',
    'Q165': 'sea',
    'Q532': 'village',
    'mehdie_26': 'cultivated populated lands',
    'mehdie_27': 'cultivated populated land',
    'Q39816': 'valley',
    'Q6394918': 'islands',
    'Q7163803': 'valleys',
    'Q12280': 'bridge',
    'mehdie_28': 'abodes',
    'Q124714': 'spring',
    'Q43742': 'oasis',
    'Q5461006': 'villages',
    'Q82794': 'region',
    'mehdie_29': 'suburbs',
    'Q188509': 'suburb',
    'Q9126476': 'cities',
    'mehdie_30': 'small village',
    'Q5741923': 'mountains',
    'mehdie_31': 'khan',
    'Q89468': 'kasbah',
    'Q23413': 'castle',
    'Q7481476': 'places',
    'Q93352': 'coast',
    'Q43483': 'well',
    'Q27862014': 'forts',
    'Q15324': 'water',
    'mehdie_32': 'sand (desert)'
}
```

Bibliography

[1] Mathieu Bastian et al. *Gephi - The Open Graph Viz Platform*. Version 0.10.0. Accessed: 2024-05-27. 2008. URL: https://gephi.org/.

[2] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/p10008. URL: http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

[3] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[4] Papers With Code. *FB15k Benchmark (Link Prediction)*. Accessed: 2024-06-05. 2024. URL: https://paperswithcode.com/sota/link-prediction-on-fb15k.

[5] OpenStreetMap contributors. *OpenStreetMap*. https://www.openstreetmap.org. 2024.

[6] Gabriele Corso et al. *Principal Neighbourhood Aggregation for Graph Nets*. 2020. arXiv: 2004.05718 [cs.LG].

[7] Luca Costabello et al. *AmpliGraph: a Library for Representation Learning on Knowledge Graphs*. Mar. 2019. DOI: 10.5281/zenodo.2595043. URL: https://doi.org/10.5281/zenodo.2595043.

[8] Luca Costabello et al. *Knowledge Graph Embeddings Tutorial: From Theory to Practice*. https://kge-tutorial-ecai2020.github.io/. Sept. 2020. DOI: 10.5281/zenodo.4268208. URL: https://doi.org/10.5281/zenodo.4268208.

[9] Alishiba Dsouza et al. "WorldKG: A World-Scale Geographic Knowledge Graph". In: *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*. CIKM '21. ACM, Oct. 2021. DOI: 10.1145/3459637.3482023. URL: http://dx.doi.org/10.1145/3459637.3482023.

[10] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2121–2159.

[11] Matt Gardner and Tom Mitchell. "Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction". In: Jan. 2015, pp. 1488–1498. DOI: 10.18653/v1/D15-1173.

[12] Aditya Grover and Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. 2016. arXiv: 1607.00653 [cs.SI].

[13] Arthur Gusmão and São Paulo. "Interpreting Embedding Models of Knowledge Bases". PhD thesis. Jan. 2018.

[14] Yāqūt Shihāb al-Dīn ibn-Abdullāh al-Rūmī al-áamawī. *Kitāb Mu'jam al-Buldān*. 1224 - 1228.

[15] William L. Hamilton. "Graph Representation Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (), pp. 1–159.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. *Representation Learning on Graphs: Methods and Applications*. 2018. arXiv: 1709.05584 [cs.SI].

[17] Md Kamrul Islam, Sabeur Aridhi, and Malika Smail-Tabbone. "Simple Negative Sampling for Link Prediction in Knowledge Graphs". In: *Complex Networks & Their Applications X*. Ed. by Rosa Maria Benito et al. Cham: Springer International Publishing, 2022, pp. 549–562.

[18] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[19] Ni Lao and William Cohen. "Relational Retrieval Using a Combination of Path-Constrained Random Walks". In: *Machine Learning* 81 (Oct. 2010), pp. 53–67. DOI: 10.1007/s10994-010-5205-8.

[20] *Link Prediction Leaderboard on FB15k-237*. Accessed: 2024.05.21. URL: https://paperswithcode.com/sota/link-prediction-on-fb15k-237.

[21] Neo4j, Inc. *Neo4j: Neo4j Graph Database Analytics*. Version 5.0, Accessed: 2024-05-21. 2024. URL: https://neo4j.com.

[22] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. *Holographic Embeddings of Knowledge Graphs*. 2015. arXiv: 1510.04935 [cs.AI].

[23] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web*. Tech. rep. Stanford Digital Library Technologies Project, 1998. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768.

[24] Shirui Pan et al. "Unifying Large Language Models and Knowledge Graphs: A Roadmap". In: *IEEE Transactions on Knowledge and Data Engineering* (2024), pp. 1–20. ISSN: 2326-3865. DOI: 10.1109/tkde.2024.3352100. URL: http://dx.doi.org/10.1109/TKDE.2024.3352100.

[25] Papers With Code. *Link Prediction*. https://paperswithcode.com/task/link-prediction. Accessed: 2024-05-21. 2024.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '14. ACM, Aug. 2014. DOI: 10.1145/2623330.2623732. URL: http://dx.doi.org/10.1145/2623330.2623732.

[27] Martin Raifer. *Overpass Turbo*. https://overpass-turbo.eu/. Accessed: 2024-06-07.

[28] Tomer Sagi et al. *The Middle East Heritage Data Integration Endeavour*. URL: mehdie.org.

[29] Masoumeh Seydi and Maxim Romanov. *al-Turayyā Project: a gazetteer and a geospatial model of the early Islamic World. (Based on: Georgette Cornu's Atlas du monde arabo-islamique à l'époque classique: IXe-Xe siècles, Leiden: Brill, 1983)*. 2022 -. URL: https://althurayya.github.io/.

[30] Amit Singhal. *Introducing the Knowledge Graph: Things, Not Strings*. Accessed: 2024-05-21. 2012. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[31] Zhiqing Sun et al. *RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space*. 2019. arXiv: 1902.10197 [cs.LG].

[32] Théo Trouillon et al. *Complex Embeddings for Simple Link Prediction*. 2016. arXiv: 1606.06357 [cs.AI].

[33] Wikidata contributors. *Wikidata: A Free Collaborative Knowledge Base*. Accessed: 2024-05-21. 2024. URL: https://www.wikidata.org.

[34] Bishan Yang et al. *Embedding Entities and Relations for Learning and Inference in Knowledge Bases*. 2015. arXiv: 1412.6575 [cs.CL].

[35] Haotong Yang, Zhouchen Lin, and Muhan Zhang. *Rethinking Knowledge Graph Evaluation Under the Open-World Assumption*. 2022. arXiv: 2209.08858 [cs.AI].

[36] Moran Zaga et al. *A Rule-Based Approach in GeoHistorical Analysis: The Case of Medieval Muslim Geographies*. 2023.

[37] Kiddo Zhu. *NBFNet-PyG*. Accessed: 2024-05-30. 2023. URL: https://github.com/KiddoZhu/NBFNet-PyG.

[38] Zhaocheng Zhu. *Neural Bellman-Ford Networks*. YouTube. Available at: https://youtu.be/x8S0e56o70I?RLBFy9Kfr2D, Accessed: 2024-05-27. 2022.

[39] Zhaocheng Zhu et al. *Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction*. 2022. arXiv: 2106.06935 [cs.LG].