Project Title: Knowledge Graph-Based Travel Destination Similarity and Recommendation

Name: Emirhan Kurtulus 12243493

Representations

(LO1) Understand and apply **Knowledge Graph Embeddings**	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
I applied the *TransE* using PyKEEN, trained and evaluated with multiple metrics (AUC, AP, Hits@K, MRR), and reflected on its strengths/weaknesses compared to GNNs	Page 14-17, section of Machine Learning, Results

(LO2) Understand and apply **logical knowledge** in KGs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Created Cypher-based logical rules	Logic section 11 and page 15

(LO3) Understand and apply **Graph Neural Networks**	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
Implemented GraphSAGE and NNConv using PyTorch Geometric, evaluated their performance, and compared them to TransE; NNConv showed strong results thanks to the edge attributes.	Pages 10 and 14

(LO4) Compare different Knowledge Graph **data models** from the database, semantic web, machine learning and data science communities.	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I used the Neo4j's property graph model and reflected its flexibility over alternatives like RDF/OWL by highlighting why Neo4j was more practical for ML pipelines	Page 15, in the data model reflection

Systems

(LO5) Design and implement **architectures** of a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I designed a full pipeline architecture: Neo4j for storage, PyKEEN for the embeddings, PyTorch Geometric for GNNs, CSV for the interoperability.	Pages 7-8 KG Construction and reflection in page 15

(LO6) Describe and apply **scalable reasoning** methods in Knowledge Graphs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Combined the logical inference with ML reasoining	Pages 7-10 and 13-14

(LO7) Apply a system to **create** a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I imported Eurostat datasets into Neo4j, constructed entities and relationships and enriched them via Cypher queries.	Page 7, KG creation part

(LO8) Apply a system to **evolve** a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Evolved the KG with inferred edges and logical labels	Pages 7, 9, and 14

Applications

(LO9) Describe and design **real-world applications** of Knowledge Graphs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I designed a KG-based travel recommendation system based on the similarities, focusing on traveler profiles(age, purpose) and destination.	Pages 5, and 15

(LO10) Describe **financial Knowledge Graph** applications	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I reflected on how methods used, these can be adapted to finance.	Page 5

(LO11) Apply a system to provide **services** through a Knowledge Graph	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
I built a service for recommending similar destinations based on traveler profiles; compared TransE, GraphSAGE, and NNConv	Pages 6, 13, and 15

(LO12) Describe the **connections** between Knowledge Graphs (KGs), Machine Learning (ML) and Artificial Intelligence (AI)	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
I reflected on how logical rules, embeddings, and GNNs complement eachother.	Page 16

Additional Information

HAS NO EFFECT ON MARKING!

(please fill it out honestly, even if it is less than what is suggested in the ECTS breakdown – you are not judged on time spent!)

How many hours did you spend on your **mini-project**? (the ECTS breakdown suggests **40** hours for this)	61-63 hours

* please exclude any hours you spent on parts reused from other courses

How many hours did you spend on your **portfolio document preparation (this PDF)**? (the ECTS breakdown suggests **15** hours for this)	12-13 hours

* please exclude any hours you spent on parts reused from other courses

Please indicate if you have **reused** parts of the **mini-project** from other courses	☐ I reused some parts: <XX>% of the mini-project
<If this is the case, please describe here what parts you reused and how you adapted them. Note that this is perfectly fine if you did so and has no effect on marking!>	

Please indicate if you have **reused** parts of the **portfolio document** from other courses	☐ I reused some parts: <XX>% of this document
<If this is the case, please describe here what parts you reused and how you adapted them. Note that this is perfectly fine if you did so and has no effect on marking! To avoid (self-)plagiarism, do not forget to also cite appropriately in this document.>	

Declaration

I have marked all parts generated by Generative AI (e.g., ChatGPT) and given any prompt I used either in a footnote or in an appendix making clear which parts are generated by which prompts or similar.	☒ I confirm this

1. Scenario

Tourism is an industry that grows fast and the travelers are facing with the huge number of choices when planning trips. When we look at the traditional recommendation systems we see that they rely on user reviews, popularity rankings, or simple filters such as budget. While useful, these methods often fail to capture deeper *semantics patterns*, for example user profiles such as age group, travel purpose align with destination characteristics.

To address this, I aim to build a **Knowledge Graph-Based Travel System**. A knowledge graph enables representing structured tourism data in a way that is both human and machine interpretable. By combining logical rules, young travelers often prefer city trips). This approach is not only beneficial for individuals but also for travel agencies, airlines, and tourism boards. Using publicly available **Eurostat tourism datasets**, in this project we aim to demonstrate how KGs can transform structured statistical data into a service.

In this project, I aimed to compute similarities between countries based on conditions such as traveler age range or purpose of the trip. To establish this, I used Eurostat datasets from past years and focused on two main tasks. The first task was identifying similar destinations for the same age range group, for example recommending destinations that young travelers (ages 15–24) have visited and enjoyed. The second task was identifying similar destinations for the same travel purpose, such as countries commonly visited for business or leisure by specific groups like older travelers. The results provide a recommendation framework that suggests destinations users are more likely to enjoy or find useful. For instance, for elderly travelers, the system can suggest countries where similar age groups have frequently taken trips. **(LO9)**

In this project, I did not aim to use any financial knowledge graph, as it does have the value column, it was not the first focus of the project. Instead, it centers on **travelers, trips, destinations, and travel purposes** extracted from Eurostat's tourism dataset. The focus is on modeling travel behavior and destination patterns rather than financial performance. Still, the same metods applied here- such as node classification (e.g., "Young Traveler", "Seasonal Destinations") and relationship prediction- could maybe have transferred to financial domains with some adjustments, for example in profiling or detecting market trends. **(LO10)**

In this project, I tried to design the knowledge graph to support several services:

- **Destination recommendation**, suggesting similar destinations based on traveler age groups and trip purposes.
- **Temporal insights**, identifying seasonal destinations by analyzing travel trends across multiple years.
- **Traveler profiling**, enriching nodes with logical labels such as Young Traveller, Adult Traveler, or Senior Traveller.

Beyond these, the KG can also serve as a **query able knowledge base**, enabling efficient exploration of questions like: *Which destinations are most common for certain age groups?*

Which purposes dominate trips to specific regions? Which traveler groups frequently visit the same destinations?
These services are powered by a combination of **logical inference**, **knowledge graph embeddings (TransE)**, and **graph neural network models (GraphSAGE, NNConv)**. **(LO11)**

2. Knowledge Graph Construction

2.1 Dataset

The data of the knowledge graph is based on the publicly available **Eurostat tourism datasets** containing the information about the countries based on some features that I focused on mostly the age range and the purpose of the trips. The Metadata of the dataset also can be found in the given **link**. The datasets are also separated below by providing their direct links. Additionally, they can be also found in the *github repository* as all together.

The datasets:

- **estat_tour_dem_extot.tsv** – This dataset provides statistics on total number of trips and expenditures by EU residents. It includes information such as the number of nights spent, the type of destination (eg. Domestic or abroad), and expenditure categories.
- **estat_tour_dem_tnage.tsv** – This dataset focuses on tourism demand by age group of the traveller. It allows analyzing how different age categories (e.g., 15–24, 25–34, 65+) engage in tourism activities and what destinations they prefer.
- **estat_tour_dem_ttsex.tsv** – This dataset provides insights into tourism demand by gender, capturing differences in the travel behavior of male and female travelers.

Together, these datasets provide a good view of tourism behavior across demographic factors (age, gender) and trip characteristics (destination, expenditure, purpose, and year). They are particularly well-suited for knowledge graph construction, as they allow linking travelers, trips, and destinations through meaningful relations.

2.2 Technologies used

In this project, firstly I was using the Jupyter notebook to check the dataset and if it requires to do the preprocessing steps, to get some data insights. Then, the raw files were preprocessed and stored as .csv files to make it easier to create the knowledge graphs. And then the datasets were also committed to GitHub repository to be able to run the queries to make the importing process easier this part is also explained with its details in the README.txt file in this folder. The preprocessed files are also stored in the folder as preprocessed and there is an extra dataset that was created in a case I would came across with any problem to merge in Neo4j, which contains the three datasets as merged. Furthermore, Neo4j AuraDB was used as the main graph database to store and query the knowledge graph. Cypher queries were applied for graph construction, enrichment, and logical reasoning. There was no need to create a predefined schema, or merging the three datasets, that is why I did not use the merged dataset, all nodes, edges, and the relations could have been added by the Cypher queries, as it is provided in the "Graph Construction-Enrichment" folder and text file named the same. Neo4j AuraDB was also used to create the logical knowledge that will be explained in the following sections. Python was my as the core programming language for data processing(Jupyter), and the machine learning tasks for example to train the models, and to evaluate the metrics to be able to compare them. **(LO5)**

These given tools were used for creating the knowledge graphs and evolving them. The required libraries are also provided in the requirements.txt file in the machine learning folder with their versions. **(L7, LO8)**

2.3 Knowledge Graph Creation

In the project that I worked with namely Travel Destination KG a straightforward structure, consisting of main node types: TravellerAge, Destination, Purpose, and Year. The TravellerAge node represents the age group of travelers (e.g., Y15-24, Y25-34, Y65+), while the Destination node represent a country or a region, uniquely identified its geographic code (geo). The Purpose node illustrates the aim of the trip, and the Year node indicates the time dimension of the data. **(LO6, LO7)**

The nodes are connected through several relationships see *Figure 1.1*. The most central one is Took_Trip, which links a TravellerAge group to a Trip, and from there to a Destination via To_Destination. Trips are further connected to their temporal dimension with the In_Year relationship and to travel reasons with the Has_Purpose edge. Additional logical enrichment introduced higher-level relationships, such as Traveled_With, which links different TravelerAge groups that visited the same destination. This structure captures both the demographic and temporal dimensions of travel, while also representing purpose and patterns. This exact design was chosen to enable analyzing how different age groups and travel purposes relate to over time.

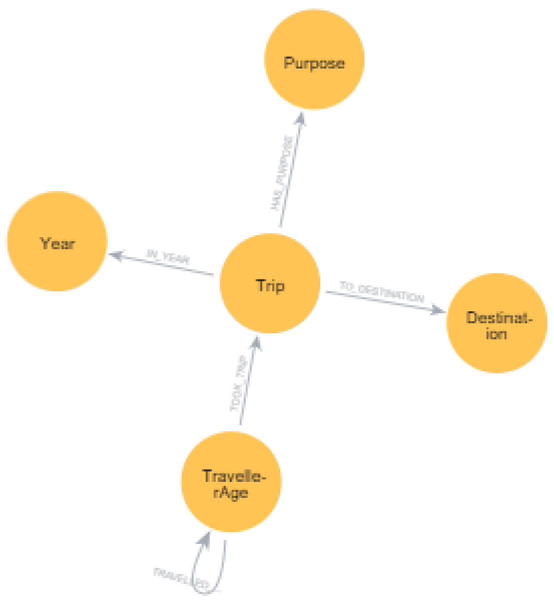


Figure 1.1 Simple Schema of the graph structure

By structuring the KG this way, it becomes possible to query not only simple counts (e.g., “How many trips were taken to Italy(“IT”) in 2019?) but also higher-level insights (e.g., “Which destinations are most frequently visited together?”, although this is a bit tough to distinguish because of the dataset size or “Which age groups prefer to take what kind of purposes such as Personal?).

In Neo4j, the data was imported from the preprocessed Eurostat CSV files, using Cypher queries stored in the *Graph Construction Enrichment* folder. Each dataset was uploaded separately and linked by shared attributes, such as *geo* code for destinations or *year* for temporal alignment **(LO5, LO6)** After the enrichment, the final version of the KG contained approximately *3,600 nodes and over 50,000 relationships*, creating a rich basis for both logical reasoning and machine learning tasks A small sample of the graph that was created I nNoe4J can be seen in the *Figure 1.2.*

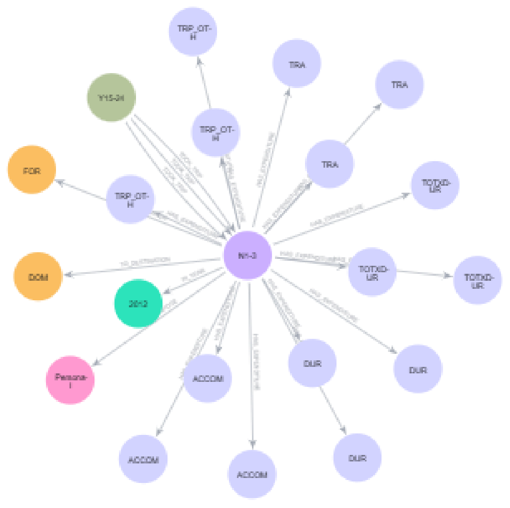


Figure 1.2. Small sample of the dataset as graph

3. Machine Learning

3.1 Knowledge Graph Embeddings

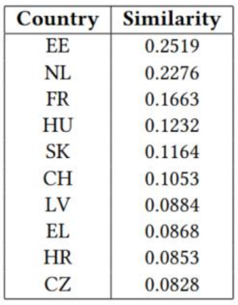
For the first embedding approach, I applied the TransE model by using the PyKEEN library. TransE is most widely used knowledge graph embedding model. It is a simple but effective approach, which works by minimizing the distance between the head and tail entities in the embedding space. Despite its efficiency, it is limited to model pairwise relationships and struggles to capture more complex patterns in the data[1].

It represents entities (nodes) and relations (edges) as vectors in continuous n-dimensional space. The idea is simple but for some of the cases it is really powerful: for a given triple (head, relation, tail), the model enforces that the embedding of the head plus the relation vector should be close to the embedding of the tail (h +r ~ t). This makes TransE suitable for graphs with relatively simple and interpretable relationships, such as those present in my KG; *Trip -> To_Destination -> Destination* or *Traveler -> Took_Trip ->Trip*. **(LO1)**

The main motivation for using the TransE in this project as I said before was to generate embedding that allow us to go beyond logical reasoning and compute similarities between destinations. [1] For example, we could identify which destinations are frequently visited by the similar age groups, or which purposes of travel are related across different countries. Unlike logical queries, Embeddings can capture latent patterns that are not explicitly encoded in the data.

To train TransE, all triples from Neo4j Aura database were first exported into a simple CSV format (head, relation, tail). These included relationships such as To_Destination, Has_Purpose, Took_Trip, In_Year, together with enriched labels like *YoungTraveller or SeasonalDestination.* The final dataset contained several thousand triples. Inside the *train_transe.py* script, these triples were loaded, processed, and split into train/validation/test subsets (80-10-10) using PyKEEN's TriplesFactory.

Evaluation was performed using the *evaluate_transe.py script,* which computed standard link prediction metrics such as **AUC, Average Precision, Hits@K (1,3,5,10), and MRR**. Results are also saved in "*evaluation_results.txt*". Additionally, the metric_transe.py script was used to parse training logs and format the metrics for easier interpretation. Finally, the similarity_transe.py script leveraged the trained embeddings to compute **cosine similarity** between entities, e.g., between two destinations or between traveler groups, providing a ranking of the most similar nodes. From the *Table 1*, we can see that EE(Estonia) has the highest similarity to Austria, and it was followed by the Netherland.



Country	Similarity
EE	0.2519
NL	0.2276
FR	0.1663
HU	0.1232
SK	0.1164
CH	0.1053
LV	0.0884
EL	0.0868
HR	0.0853
CZ	0.0828

Table 1. Top 10 most similar countries to AT(Austria)

Although TransE gave a good **baseline**, it has known limitations. It cannot handle continuous attributes directly, since it only supports symbolic triples. This reduces its ability to fully capture the numerical aspects of tourism data. **(LO6)** However, even with this limitation, the Embeddings were useful for exploring hidden patterns in the KG and as a baseline for comparison with more advanced graph neural network models. **(LO8)**

3.2. Graph Neural Networks – GraphSAGE

Besides embeddings with TransE, I implemented a **Graph Neural Network (GNN)** model, specifically **GraphSAGE (Graph Sample and Aggregate)**, using PyTorch Geometric.

GraphSAGE is designed to learn node embeddings by **aggregating information from a node's local neighborhood**, which makes it particularly powerful for large graphs where explicit embeddings for unseen nodes are not practical. **(LO3)**

In this project, GraphSAGE was used, which has gained attention for its scalability and inductive capabilities, making it well-suited for applications where new nodes frequently appear [2], this model was applied to the travel KG to better incorporate **contextual attributes**, such as the year of the trip or expenditure values, which TransE could not capture directly. The *train_graphsage.py* script handled the preprocessing of the Neo4j-exported graph (nodes, edges, and features) and trained the GraphSAGE model. The workflow

Evaluation was made with the *evaluate_graphsage.py* script, which we computed the same metrics as we did for the TransE. Additionally, the similarity_graphsage.py script allowed for cosine similarity computations, e.g., to identify the most similar destinations or age groups.

GraphSAGE proved useful as it can generalize the training set and leverage structural patterns in the graph, rather than relying on symbolic triples as TransE does. However, it also requires more training time and careful tuning of hyper parameters, such as neighborhood size and number of layers. **(LO6, LO8)**

3.3. Graph Neural Networks – NNConv

Finally, I experimented with a more advanced GNN architecture, the **NNConv (Neural Network Convolutional Operator)**. Unlike GraphSAGE, it dynamically generates edge-specific filter using neural networks, allowing it to incorporate edge attributes directly into the learning process. **(LO3)** This was particularly relevant for my KG, since edges such as ***Took_Trip*** or ***Has_Expenditure*** carry meaningful values.

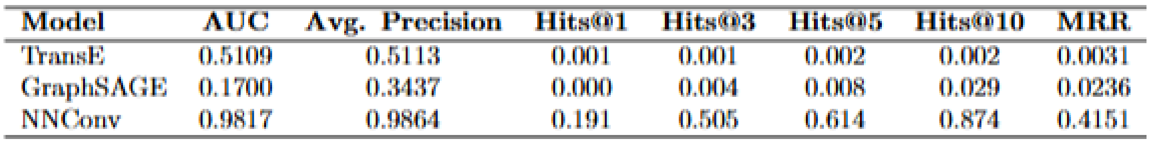
The model was implemented in the train_nnconv.py script. After training, the embeddings and mappings were saved for downstream tasks. The evaluate_nnconv.py script was used to evaluate link prediction performance, again using **AUC, AP, Hits@K, and MRR**. The final script, similarity_results.py, was used to analyze node similarities based on the trained embeddings.

As a summary in the light of the information provided above, among three approaches (TransE, GraphSAGE, NNConv) NNConv achieved the best performance that we will discuss the results in the next chapters. Its ability to incorporate edge features allowed for a richer representation of the travel KG, leading to more meaningful similarity results between destinations and traveler profiles. **(LO6, LO8)**

3.4. Comparison of Models' Results

The **core** task of this project was not to design a recommendation system in the classical sense, but rather to *discover similarities between travel destinations* based on traveler demographics (e.g., age groups) and purposes of trips. By computing embeddings with models such as TransE, GraphSAGE, and NNConv, we were able to capture latent patterns in the tourism data and measure how closely different destinations are related.

Each model was evaluated using link prediction metrics as we mentioned before, which indicate how well the embeddings preserve the observed structure of the graph. The NNConv model achieved the strongest results, with high AUC and Hits@K scores, meaning it was particularly effective at capturing meaningful patterns in destination relationships, see Table 2. TransE provided a simple but interpretable baseline, while GraphSAGE underperformed on this dataset, likely due to its limited capacity to model heterogeneous relations.



Model	AUC	Avg. Precision	Hits@1	Hits@3	Hits@5	Hits@10	MRR
TransE	0.5109	0.5113	0.001	0.001	0.002	0.002	0.0031
GraphSAGE	0.1700	0.3437	0.000	0.004	0.008	0.029	0.0236
NNConv	0.9817	0.9864	0.191	0.505	0.614	0.874	0.4151

Table 2. Comparison of models on AUC, Average Precision, Hits@K, and MRR.

Beyond metrics, the real value comes from cosine similarity queries, where embeddings were used to identify the top-10 most similar destinations for a given target. For example, Austria (geo:AT) was found to be closely linked to Italy and Slovakia in the "Young Traveller" group, reflecting realistic travel patterns. These insights demonstrate that embeddings allow us to uncover hidden similarities between countries, which can then be used as the basis for recommendation services if desired.

In summary, the models were compared not as competing recommender systems, but as different approaches to similarity discovery in travel data. The similarity-based results can directly support future recommendation systems, but the project's primary contribution is in showing how Embeddings can enrich the understanding of travel patterns.

4. Logical-Based Representation

4.1. Logical Rules and Inference

Logical knowledge was used to enrich the Knowledge Graph with meaningful categories that may help to create the similarities and the recommendations, and new relationships. **(LO2)** These rules go beyond the raw statistical data, enabling the KG to provide insights into traveler behavior, destination popularity, and purpose-specific patterns.

The first set of the rules was that I focused on **travelers**. I categorized the AgeGroup**s** into labels such as *YoungTraveler, AdultTraveler, and SeniorTraveler*, providing a clearer segmentation of traveler profiles, for example, YoungTraveler and the SeniorTraveler may have different kind of interest of different countries. Additionaly, connections were inferred between travelers through the creation of a *Traveled_With* relationship whenever two different age groups were observed to visit the same destination. With this, which adds a layer of social or behavioural similarity to the graph structure.

Next, I worked on the **destinations** to apply the logical rules. Nodes that appeared across multiple years of trips were marked as *SeasonalDestination*, while others were labelled as either

DomesticDestination or InternationalDestination depending on their geo attributes. Additionally, *PopularDestination* labels could be assigned based on the number of distinct visitors, highlighting destinations with high attractiveness.

The third set of the rules focused on **purpose of the travel**. Business-related trips led to a BusinessPurpose label corresponding nodes, and in future iterations, this can be extended with edges such as *Commonly_Associated_With* between the purposes and destinations, indicating patterns like "Business trips are often linked to capital countries instead of touristic countries."

As a final for the logical phase, I applied the logical inference to the trip metadata. For instance, trips taken in recent years were labelled as "*RecentTrip",* while older ones were classified as *OldTrips,* another approach could be in this phase to split the years as *BeforeCovid and AfterCovid,* to see maybe the changes. Furthermore, Frequent travelers could also be marked with a "*FrequentTraveler*" if their number of trips exceeded a threshold, but because of the dataset size it is a bit tough to apply this setting to the dataset. For instance, to give the basic idea behind some of the logical rules used for creating these labels were:

- **YoungTraveler** ; travelers in the age group 15-24 years old
- **AdultTraveler** ; travelers in the age groups 25-34 and 35-44 years old
- **SeniorTraveler** ; traveler in the age groups 45-54, 55-64 and 65+ years old
- **SeasonalDestination** ; destinations visited in two or more distinct years
- **BusinessPurpose** ; country nodes whose has more "Business" labels

To summarize the logical enrichment steps allowed the Travel KG to evolve from a raw dataset into a semantically meaningful graph. By combining structural patterns with domain-specific rules, the KG became better suited for downstream tasks.

4.2. Knowledge Graph Evaluation

The traveler and destination labels representing their main roles and characteristics, as well as the inferred Traveled_With relationship, were used to enrich and evolve the knowledge graph. The traveler labels (e.g. YoungTraveler, AdultTraveler, SeniorTraveler) were especially useful for improving the NNConv model by incorporating more semantic information that helped in the task of finding similar travel groups and destinations.

As we discussed already, the Traveled_with edge did not exist in the raw data. Its inclusion added richness and complexity to the graph by connecting travelers who visited the same destination. This does not only evolve the graph structure, it also helps to have more meaningful queries, for instance we can identify travel comparison by age group, or discovering hidden patterns of destinations often co-visited by different age ranges.

The logical labels on traveler, destinations, and purposes serve the same purpose when querying the graph, helping to find relevant groups of travelers or destinations based on key characteristics, without the immediate need for applying ML models. **(LO8)**

4.3. Context and Limitations

While these rules add more knowledge and richness to the travel knowledge graph, this approach still has some limitations. First of all, all of the rules were manually defined and executed as Cypher queries, relying on fixed thresholds (e.g., distinguishing YoungTravelers as ages 15-24, or defining SeasonalDestinations as destinations visited in more than two years). While this works reasonably well on the Eurostat dataset that I used for the project, it may not scale easily up to larger or more diverse datasets where patterns are less rigid, even I have been struggling with the size of the dataset it may be a big limitation that I have come across.

Second, the rules cover only a limited number of attributes, such as traveler age, trip purpose, or year of travel, and chosen labels are relatively straightforward. If we wanted to define more specific categories (e.g., "Budger Traveler"based on expenditure, or "Business-Frequent Traveler for the Senior Age Range" based on purpose and frequency), it would be require significantly more manual effort and increasingly complex rules.

Another limitation is that the current rules lack *temporal* and *contextual sensitivity*. For example, a traveler classified as a *YoungTraveler* today will age into another group, and destinatinations labeled a *Seasonal* might only hold that property for a small specific time window. Our current setup does not account such changes in real-world dynamics.

In terms of scalability, adding more destinations, travelers, and trip records would work as long as the attributes remain consistent. However, applying this to other datasets from other sources (e.g., from airlines, tourism boards, or different countries for example to increase the size for the countries from other continents and so on) would require adapting or maybe recreating the logical rules.

As a summary of this part, we can say that an avenue for improving the scalability and reducing the manual effort would be to integrate rule mining or probabilistic logic techniques. This would allow the system to automatically discover patterns instead of relying solely on hard-coded thresholds, providing more flexibility and robustness. **(LO6)**

5. Results/Reflections

5.1. Service Outcome

When I started for the project at the beginning, my main goal was to build a knowledge graph-based system that could recommend or identify similar travel destinations by considering different traveler profiles such as age group or travel purpose. **(LO11)** To achieve this, three different models were implemented and compared as we discussed before: TransE, GraphSAGE, and NNConv.

The TransE model provided a baseline using simple triples extracted from knowledge graph that we created on Neo4J Aura. For example, when querying for the most similar countries to *Austria (AT),* see below *Table 3. TransE* returned countries like *Estonia (EE), Netherlands (NL),* and *France (FR)*. While these results showed some meaningful connections, the overall performance across evaluation metrics remained very, as seen in the *Table 4.* **(LO1)**

Rank	Country	Similarity
1	EE	0.2519
2	NL	0.2276
3	FR	0.1663
4	HU	0.1232
5	SK	0.1164
6	CH	0.1053
7	LV	0.0884
8	EL	0.0868
9	HR	0.0853
10	CZ	0.0828

***Table 3**. Top 10 most similar countries to "AT", for the TransE model results*

On the other hand, The GraphSAGE model provided attempted to incorporate additional contextual attributes, for instance, as the year of the trip or expenditure. Although it improved slightly in capturing structural similarities, its performance metrics were still weak compared to expectations, with especially low values on *Hits@K and MRR*, see *Table 4*. This suggests that while GraphSAGE has inductive capabilities, it struggled with the scale and nature of the Eurostat dataset used in this project. **(LO3, LO6)**

Model	AUC	Average Precision	Hits@1	Hits@3	Hits@5	Hits@10	MRR
TransE	0.5109	0.5113	0.001	0.001	0.002	0.002	0.0031
GraphSAGE	0.1700	0.3437	0.000	0.004	0.008	0.029	0.0236
NNConv	0.9817	0.9864	0.191	0.505	0.614	0.874	0.4151

***Table 4**. Comparison of model performance across evaluation metrics*

Lastly, the NNConv model, which work edge attributes directly, significantly outperformed the other two. It achieved the highest results across all evaluation metrics, with an *AUC of 0.98, AP of 0.99, and Hit@10 close to 0.87,* more detailed information see Table 4. These results demonstrate that incorporation with the edge-level information such as trip characteristics allowed for a richer representation of the travel KG, leading to more meaningful similarity results between *destinations and traveler profiles*. **(LO3, LO8)**

Rank	Country	Similarity
1	AT	0.9992
2	AT	0.9967
3	BG	0.7860
4	MT	0.7704
5	SK	0.7606
6	SK	0.7602
7	IT	0.7422
8	BG	0.7197
9	SK	0.7097
10	IT	0.7081

***Table 5**. Top 10 most similar countries to "AT" for NNConv model results*

In the light of the information and results given above, the comparison shows that while TransE and GraphSAGE provided a foundation for similarity-based recommendations, the **NNConv** model best supported the intended service outcome of recommending similar destinations. It can be seen as a proof-of-concept that the travel KG, when combined with edge aware GNN, can serve as the basis for the meaningful recommendation in tourism. **(LO9, LO11)**

5.2. Data Model Reflection

For this project, I chose to represent the travel knowledge graph as a property graph in Neo4j. This model allowed me to flexibly represent different entities—such as traveler age groups, travel purposes, destinations, years, and expenditures—as nodes, and connect them with relationships like Took_Trip, To_Destionation, In_Year, or Has_Expenditure. Each of these nodes and edges could also carry attributes, which made it possible to capture richer contextual information.

The property graph structure was especially beneficial because of its schema-free nature. With Cypher queries, I could directly import the data from the Eurostat tourism datasets that I commited on GitHub, define node labels, set relationships, and enrich the graph with logical rules. This flexibility made Neo4j AuraDB the ideal choice for building and evolving the KG. Also, I want to mention it to work with the cloud base Neo4j Aura is used for the project instead of the local Neo4J, because it may be important that if someone want to run the queries and may not work on the local Neo4j. From a practical perspective, Neo4j also supported exporting parts of the graph into CSV triples, which were later used in machine learning tasks (TransE, GraphSAGE, NNConv). This tight integration between Neo4j and the ML pipelines in PyTorch Geometric and PyKEEN was crucial, because it ensured that the data model could be directly connected to embedding models and graph neural networks.

While other graph data models such as RDF/OWL provide more formal ontological reasoning capabilities, they would have been less practical in this case. Preparing the data for ML would require additional transformations, making the workflow more complex. By using Neo4j's property graph model, I could quickly move from raw datasets to an ML-ready KG. This shows how the choice of data model directly supports the scalability and practicality of ML-driven KG applications. **(LO4, LO5)**

5.3. Connections

Last of all, I want to mention that on the connections between the different components used in this project. The travel knowledge graph was first enriched with logical queries, for example by labeling traveler age groups (*YoungTaveler, AdultTraveler, SeniorTraveler)* or defining *BusinessPurpose and SeasonalDestination* nodes. These enrichments provided higher-level **semantic information** on top of the Eurostat data, making the graph more interpretable.**(LO2)**

It is worth to mention that important connection was the **interplay between embedding models (TransE) and GNNs (GraphSAGE, NNConv)**. While TransE offered a lightweight baseline focusing on triples, GNNs leveraged the enriched property graph structure more effectively, particularly NNConv, which incorporated edge features. Together, these models demonstrated how different KG representation techniques complement one another, and how combining symbolic (logic-based) and sub-symbolic (ML-based) reasoning can improve the overall system. **(LO12)**

To summarize, in this project I tried to show that reflect on the main connection between logical knowledge and ML methods, and graph neural networks can work together to evolve a knowledge graph into a service-oriented system, in my case enabling recommendations of similar travel destinations. These connections highlight the broader vision of knowledge graph as a bridge between logic, machine learning, and real-world applications. **(LO12)**

References

[1] S. M. Asmara, N. A. Sahabudin, N. S. N. Ismail and I. A. A. Sabri, "A Review of Knowledge Graph Embedding Methods of TransE, TransH and TransR for Missing Links," 2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS), Penang, Malaysia, 2023, pp. 470-475, doi: 10.1109/ICSECS58457.2023.10256354.

[2] Gomes, Tomás Pereira Matos. "GraphSage for Link Prediction." (2025).