TP: Link Prediction in a Knowledge Graph

lamine.diop [at] epita.fr

December 2, 2024

Learning Objectives

- To explore and implement state-of-the-art methods for knowledge graph embeddings.
- To evaluate embedding quality using performance metrics.
- To analyze the results and understand the strengths and limitations of each applied method.
- To utilize the ontology to infer predicted facts and assess their logical validity.

Introduction

You will work with a knowledge graph in the Turtle format stored in the file 'human.ttl' representing entities and their relationships, as well as an additional file 'humanrdf.ttl' containing the ontology associated with the graph. These files contain triples of the form `(subject, predicate, object)`, such as `(Person_A, a, Researcher)`. The objective of this assignment is to implement and analyze link prediction methods using embedding models and verify if certain predicted facts can be inferred using the ontology.

Part 1: Data Preparation and Exploration

1. Loading the Turtle File

Use a Python library (e.g., `rdflib`) to load both `human.ttl` and `humanrdf.ttl`. Iterate through the triples and identify the unique entities, relations, and number of triples in `human.ttl`.

1. How many unique entities and relations are in `human.ttl`?
2. What is the size of the graph in terms of the number of triples?
3. Identify some entities and relationships that could illustrate interesting link cases.

2. Preparing Data for Learning

- Split the triples into a training set (80%) and a test set (20%) for model evaluation.
- Ensure the presence of "negative" triples for evaluation using a negative sampling technique.

Part 2: Implementation of Embedding Models

Use the following state-of-the-art methods: **TransE**, **DistMult**, **ComplEx**, or **RotatE**. Libraries like `PyKEEN`, `Pytorch-Geometric`, `OpenKE`, or `Ampligraph`(may not be up to date) may be used for implementation. Some tutorials are available here:

- With `PyKEEN` (`https://colab.research.google.com/drive/1Eu9mm_F7a7kcp-fZaL5qOF6ZksQTYeDI#scrollTo=VbvrIsPEvFIg`)
- With `Pytorch-Geometric` (`https://medium.com/stanford-cs224w/introducing-distmult-and-complex-for-pytorch-geometri`
- With `Ampligraph` (`https://colab.research.google.com/drive/1Fcf8vkuaO6VCOB3MAZlpDebCAgyUnMBj#scrollTo=4dcvCAfyDc5S`)

1. Train Models

Train each model on the training set with default parameters, then tune embedding parameters (e.g., embedding dimension).

1. What are the main differences in entity representations between TransE and ComplEx?
2. Analyze the computational complexity and training time impact of each model.

2. Embedding Visualization

Use dimensionality reduction (e.g., PCA or t-SNE) to visualize entity embeddings.

1. Are entities with similar relationships clustered in embedding space?
2. Identify and analyze clusters of entities. Do these clusters correspond to coherent entity types?

Part 3: Model Evaluation and Result Analysis

1. Evaluation Metrics

Evaluate models using **Mean Rank**, **Mean Reciprocal Rank (MRR)**, and **Hits@K** (for $K = 1, 3, 10$).

1. Define each metric and explain how they assess link prediction quality.
2. Interpret the results for each model and identify the top-performing models for this dataset ((a) with the train dataset, (b) with the test dataset).

2. Comparative Analysis

Compare the performance of the models in terms of MRR and Hits@K.

1. Which model provides more accurate predictions (i.e., better Hits@1)?
2. What is the distribution of scores for "false positive" links?

3. Ontology Verification

Using `humanrdf.ttl`, try to infer some of the links predicted by the models.

1. Which predicted links are also inferred by the ontology?
2. Are there any link predictions that contradict the ontology? If so, why?

Part 4: Advanced - Self-Adversarial Negative Sampling

1. Self-Adversarial Negative Sampling

Implement self-adversarial negative sampling to generate harder negative examples based on current embeddings.

1. Does self-adversarial sampling improve metric scores compared to standard negative sampling?
2. Analyze why self-adversarial sampling generates "harder" negatives and its impact on model training.

Part 5: Link prediction in UniProt KG

To ground this exercise in real-world data, you will use the UniProt[1] (The Universal Protein Resource) knowledge base, a rich resource for protein information. The data will be retrieved using the UniProt SPARQL endpoint, which supports semantic queries over RDF (Resource Description Framework) data. Students are encouraged to formulate their own SPARQL queries to extract RDF triples that align with their specific interests, such as (Protein, participatesIn, BiologicalProcess) or other relevant patterns. Alternatively, they can select and adapt a query from the example queries provided on the UniProt SPARQL endpoint documentation (`https://sparql.uniprot.org/sparql`) to retrieve a sample of RDF data that suits their focus area.

This exercise will demonstrate the application of advanced techniques in data mining and machine learning to biological data, enabling insights into complex biological systems.

Part 6: Report and Conclusions

1. Synthesis of Results

Write a report summarizing metric results for each model, discussing the relative performance of each approach.

1. Which models and adjustments are most suitable for large for the provided graph?
2. Suggest potential improvements to enhance link prediction performance.

2. Future Perspectives

Discuss the utility of link prediction in knowledge graphs. What other types of models or techniques could improve predictions?

3. Course Feedback and Improvement Suggestions

How did you feel about the course overall, and what additional topics or improvements would you suggest if it were to be redesigned or offered again?

[1] `https://www.ebi.ac.uk/`

Appendix: Code and Documentation

Provide the Colab page for your implementations along with brief documentation explaining the main steps and functionalities of the primary functions.

Why don't skeletons fight each other?
Because they don't have the guts!

Now that we've warmed up with a bit of humor, remember this: Just like in coding (or life), sometimes things might seem confusing at first, but with a bit of patience and the right mindset, you can unlock the answers! Keep pushing forward, and don't hesitate to ask questions – we're all here to learn and grow together. You've got this!