# PRIM Report : Knowledge Graphs Entity Type Prediction Leveraging large language models and graph neural networks

Denis Fouchard, supervised by Mehwish Alam

Télécom Paris – Polytechnique Institute of Paris, France

**Abstract.** The entity type information in Knowledge Graphs (KGs) such as DBpedia, Freebase, etc. is often incomplete due to automated generation or human curation. Entity typing is the task of assigning or inferring the semantic type of an entity in a KG. This PRIM projects aims at leveraging the outstanding semantic knowledge and understanding of new open source large language models (LLMs) such as Llama 3 8-b jointly with Graph Neural networks to perform automatic entity type prediction from given node and surrounding relationships with other entities in a Knowlege Graph.

**Keywords:** Entity Type Prediction · RDF2vec · Knowledge Graph Embedding · Large Language Models

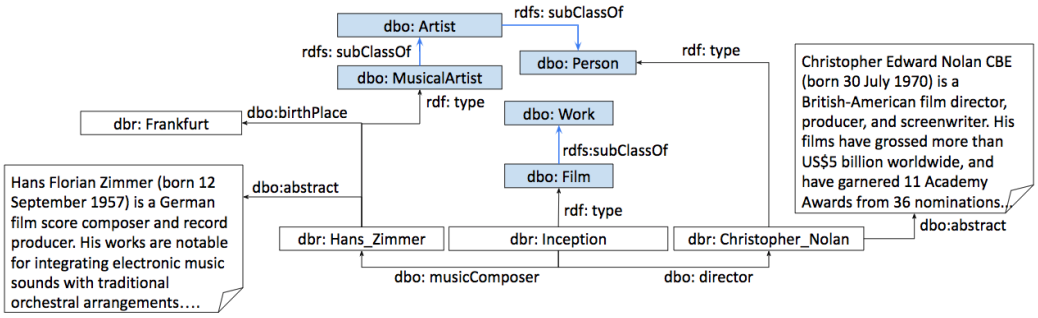
## 1 Introduction

Many efforts have been made towards the automated generation of Knowledge Graphs (KGs) from heterogeneous resources such as text or images. One such effort is the creation of cross-domain KGs such as DBpedia [2], Wikidata [16], Freebase [4], etc. which are either extracted automatically from structured data, generated using heuristics, or are human-curated. This leads to incomplete information in the KGs which can occur on factual level (e.g., missing entities and/or relations between the entities) or on schema level (e.g., the missing entity type information). For instance, DBpedia version 2016-10 consists of 48 subclasses of *dbo:Person*; however, only 36.6% of the total number of entities belonging to *dbo:Person* are assigned to its subclasses. Moreover, 307,164 entities in the entire DBpedia 2016-10 version are assigned to *owl:Thing*.

To address the KG incompleteness on the factual level, a lot of models [5, 6, 14], etc. have been proposed. These models focus mainly on predicting the missing entities and relations in the KGs but not the entity types. However, the entity type information in KGs plays a vital role in various Natural Language Processing based applications such as question answering [15], relation extraction [8], recommendation, or system [17]. Following these lines, this paper focuses on the problem of entity typing which is the task of assigning or inferring the semantic type of an entity in a KG. Figure 1 shows an excerpt from DBpedia where the class *dbo:MusicalArtist*is a subclass of *dbo:Artist* which is a subclass of *dbo:Person*. *dbo:Artist* and *dbo:MusicalArtist*, respectively, are the fine-grained entity types for *dbr:Hans_Zimmer* and *dbo:Artist* is the missing type information. *dbo:Person* is the coarse-grained type.

Recent years have witnessed a few studies on entity typing approaches in KGs using heuristics [13] and machine learning based classification models [12, 18, 9, 10, 3]. These models

Fig. 1: Excerpt from DBpedia




predict entity types using different KG features such as the anchor text mentions in the textual entity descriptions, relations between the entities, entity names, and Wikipedia categories. They learn the representation of the entities from their KG structure by using translational models [11], GCN-based models [10], neighborhood based attention models [20] followed by the correlation between the entities and its types. These models exploit the neighborhood information only by the entities directly connected, i.e., the triple information of the entities. However, the large amount of semantic information those nodes carry, which could be leveraged with the advent of open-source large language models such as the Llama family of models [1], are largely underexploited. Additionally, the textual entity descriptions in the KGs contain rich semantic information which is beneficial in predicting the missing entity types. For instance, as depicted in Figure 1, the textual entity descriptions of the entities clearly mentions that *dbr: Christopher_Nolan* is a *director*, *dbr: Hans_Zimmer* is a *music composer*, and *dbr: Inception* is a *film*. Some of the existing baseline models such as MuLR [19] use non-contextual Neural Language Models (NLMs), whereas the other uses GCN model [10] on the words extracted from the entity descriptions. Therefore, to capture the contextual information of the textual entity description contextual NLM, is used to generate entity representations.

This PRIM project is based on both the work of the team on the GRAND Framework, and the G-Retriever model [7] . The objectives are the following :

- Understand the structure of Knowledge Graphs such as DBPedia, and how they are represented in data.
- Familiarise with state-of-the-art work on the subject, including models proposed by the team at Télécom Paris, and explore various architectures proposed by peers regarding closely-related tasks.
- Explore ideas leading to the design a Deep Neural Network architecture including a Large Language Model to perform entity type prediction
- Benchmark those ideas using the DBPedia Knowledge Graph.
- Deduce from those experiments the necessity or efficacy of using LLMs for that task.

# 2 Entity Type Prediction: starting from the G-Retriver Model

## 2.1 G-Retriever

As previously stated, our base for this project is the G-Retriever model, which is summarized by the figure below. This is a good base to work with, because this model has succeeded in capturing both the graph structure of the data and the semantic expressiveness of the entities. This model consists of :

- A node and edge indexer (Step 1)
- A node and edgge retriever (Step 2)
- A graph encoder with a projector (Graph Transformer)
- A large language model

We will try to tweak each part of this model to fit our specific task. Indeed, our goal is slightly different than that tackled by G-Retrievers. While G-Retriever uses the user prompt to select the most semantically relevant node and edges from the source entity $k$-hop subgraph, ours does not rely on a prompt. Our query is always the same: to find the type of the given entity. Moreover, there is no such thing as a target entity in our problem. While the initial model aims at retrieving a desired entity and generates an answer for the user from it, we simply need to generate a predicted entity type.

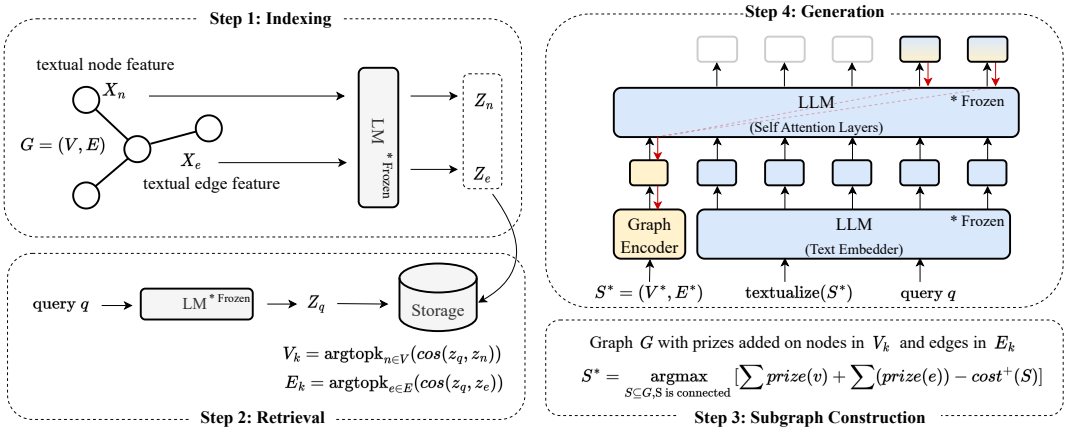



Fig. 2: Model G-Retriever

## 2.2 Entity Representation

A Knowledge Graph (KG) can be represented similarly to an oriented graph. $G = (V, E)$ where the node set $V$ represents the entities, and $E$ is a collection of triples $(s, r, t)$ where :

- $s$ is the source entity

- $t$ is the target entity
- $r$ is the relationship (or predicate) linking the source to the target.

The DBPedia ontology is a hierarchy of types and subtypes. It can be represented as a tree where each node is a type that can have multiple inheriting types, and the leaf types are types with the highest level of granularity. Granularity is a key component in the prediction of types. However, in most of the work, we will keep the smallest granularity with only a few classes. It would be interesting to challenge the granularity, explore incomplete typing, or try with other classes more semantic similarity, or less similarity but this surpasses the scope of this short PRIM.

Entities $V$ are simply represented as their DBpedia address (their url in the database). For example, Albert Einstein will be represented as

*http://dbpedia.org/resource/Albert_Einstein*

although a LLM will be slightly more performant with stripping the prefixes and only working with the natural language version

*Albert Einstein*

The knowledge graph is simply represented as $E$, a list of triples.

### 2.3 Generating Entity Embeddings

Since we will use both a GNN and a LLM in this project, we need to compute different embeddings for the entites.

**Graph Embedding**
To provide context on the surrounding of an entity $e \in E$, we will extract from the entire DBPedia Knowledge graph a $k$ hop subgraph surrounding $e$. In practice, this subgraph $G_k(e)$ will be the graph induced by the nodes that can be reached with $k$ "hops" or steps in a graph walk. We need to balance between a rich context and too much noise and irrelevant information. As we move further in the environment, the nodes are less connected to the entity considered $e$ and can increase in generality. We opt for $k = 2$, and limit the subgraphs to 100 triples.

A Knowledge Graph Neural Network, such as a graph transformer will provide an embedding $g_e$ for a subgraph $G_k(e)$

**LLM embedding**
We chose an off-the-shelf pre-trained open-souce LLM - Llama3-8b [1]. We will be attempting several text representations of the source entity, such as giving the whole DBPedia URL, or simply the human readable name, or even a text description of the entity. We will denote such text representation query $q$.

### 2.4 Entity Description Representation

In some setups (such as using SentenceBert instead of a LLM), we will need to provide our model with a text description of the entity. For that, we will use the description provided by

DBPedia, and strip it to use only the first sentence, as it is the one that generally reflects the nature of the entity. For instance, the description for *Howard McFarland* :

*Howard McFarland Hall was an American early-era racecar driver. Hall competed in the inaugural 1911 Indianapolis 500 in a Velie.*

### 2.5 Graph Encoder

For all experiments, we employ a Graph Transformer as the graph encoding model. The architecture consists of 4 attention layers, each with 4 attention heads. The input and hidden feature dimensions are set to 1024, ensuring a rich representation capacity. To enhance generalization and prevent overfitting, a dropout rate of 0.2 is applied to the output of each attention module.

### 2.6 Entity Type Prediction

We have a set of entities $X = \{s_1, .., s_n\}$ and a set of corresponding labels $Y = \{y_1, .., y_n\}$ belonging to the set of classes $C$. The original G-Retriever frameworks aims at generating a text answer to a text prompt. This is a classification task, not a typical LMM inference. The desired output from the model is a class prediction $\hat{y} \in C$.

**Class Representation**

To start simple, we selected a subset of classes $C$, and chose samples from those classes. There are $|C|$ classes in total so we need to output a softmax distribution.

**Classification Head**

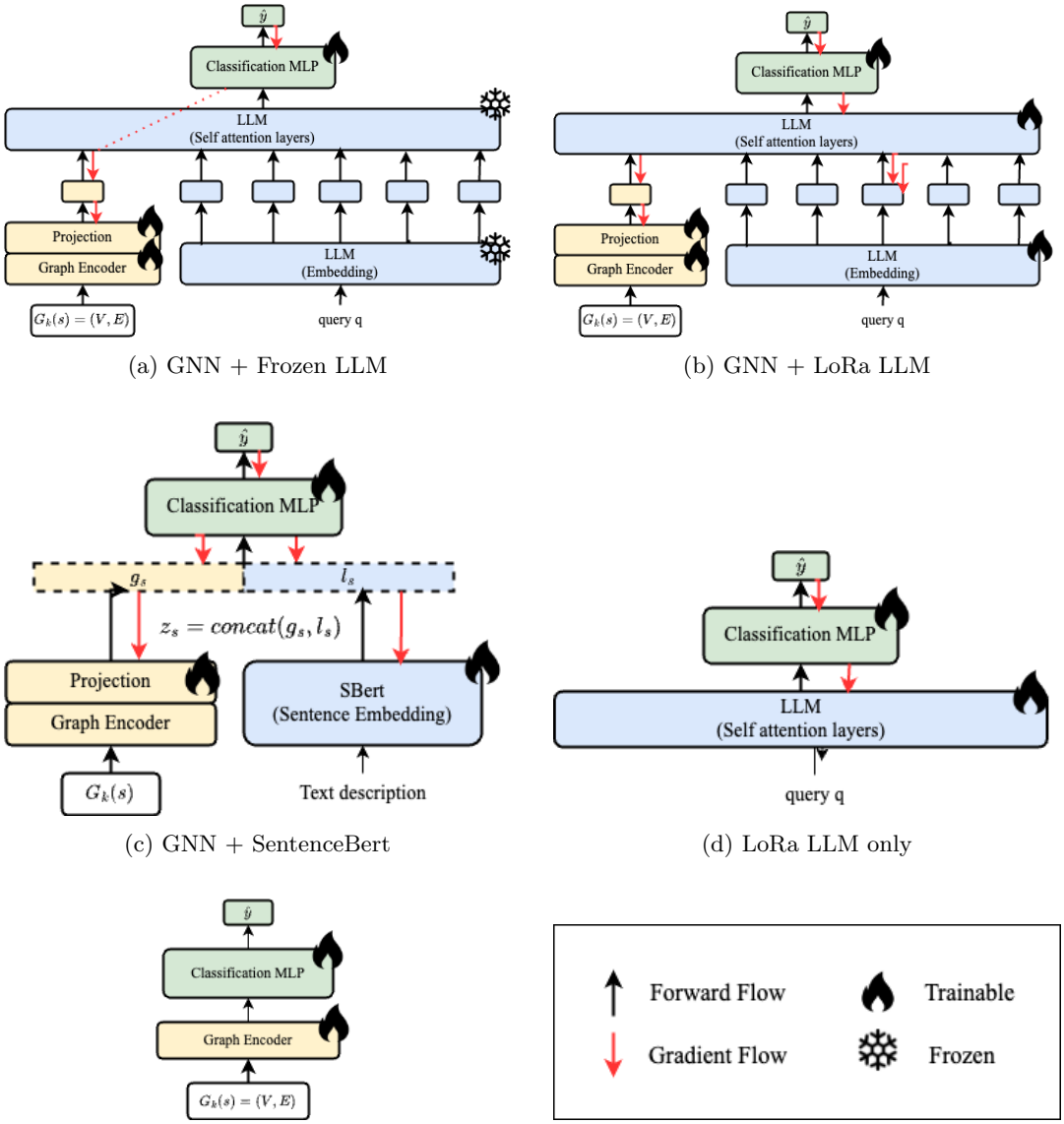
The challenge was replacing the LLM inference head (which aims at predicting the next token of the sequence), by a classification head. What we did was taking the last hidden layer of the LLM, and feeding it into a fully connected multi-layer perceptron. The idea is to capture the semantic understanding of the LLM through the forward pass in the attention layers, and then use it to learn a classification of entity. The head consists of two fully connected layers, the first of input dimension 4096 (i.e.e. the dimension of the hidden representation output), with output dimension 1024, followed by a dropout of 0.1, and batch normalization and a ReLU activation function, into a second layer with output size $|C|$ the number of classes, followed by softmax.

## 3 Experiments

This section provides details on the benchmark datasets, experimental setup, analysis of the results obtained, and the ablation study.

### 3.1 Model architectures

I started from the whole G-Retriever model without touching anything, and then modified parts progressively. 3




(a) GNN + Frozen LLM

(b) GNN + LoRa LLM

(c) GNN + SentenceBert

(d) LoRa LLM only

(e) GNN-Only

(f) Legend

Fig. 3: Comparison of the different test model architectures.

### 3.2 Dataset

DBpedia630k consists of 630,000 entities and 14 non-overlapping classes. The entities of the extended DBpedia630k dataset are split equally into three parts DB-1, DB-2, and DB-3, each containing 210,000 entities. Each DBpedia split is divided into a train, test and validation

Table 1: Statistics of the datasets

| Parameters | DB-1 | DB-2 | DB-3 |
|---|---|---|---|
| #Entities | 210,000 | 210,000 | 210,000 |
| #Entities train | 105,000 | 105,000 | 105,000 |
| #Entities test | 63,000 | 63,000 | 63,000 |
| #Entities validation | 42,000 | 42,000 | 42,000 |

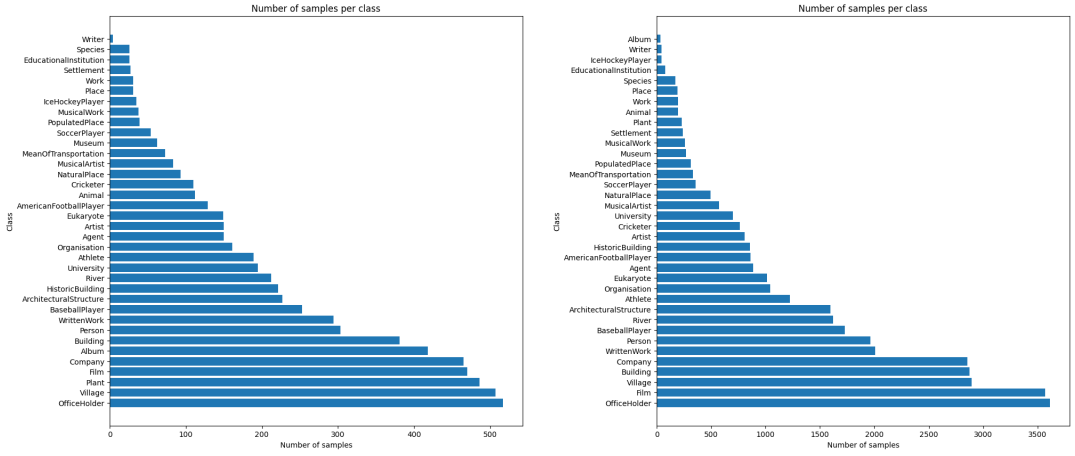
Table 2: Statistics of the two sampled datasets

| Parameters | DBPedia6k | DBPedia36k |
|---|---|---|
| #Entities | 6,721 | 36,891 |
| #Entities train | 4,704 | 25,823 |
| #Entities test | 1,009 | 5,534 |
| #Entities validation | 1,008 | 5,534 |

set with 50%, 30%, and 20% of the total entities respectively [10] as well as to 48 classes in the class hierarchy. There are no shared entities between the train, test, and validation sets for all the DBpedia630k splits. The statistics is provided in Table 1. The code, and data are publicly available[1].

**DBPedia6k and DBpedia36k** For the sake of computing time in our experiments, we did not use the whole dataset, but two sampled version from DB-3. Both contain 37 non-overlapping classes. Each DBpedia split is divided into a train, test and validation set with 70%, 15%, and 15% of the total entities respectively as well as to 48 classes in the class hierarchy. There are no shared entities between the train, test, and validation sets. The statistics is provided in Table 2. The code, and data are publicly available[2].

Here are some statistics about class representation in each dataset.




(a) DBPedia6k (b) DBPedia36k

[1] shorturl.at/abJRW [2] shorturl.at/abJRW

### 3.3 Experimental Setup

Training was performed on 5 epochs with Early Stopping, using the Adam optimizer and learning rate scheduling. The loss is a classical Binary Cross Entropy Loss. When LLM Fine-Tunning is considered, we used a LoRA adapter. See 3 for the full LoRA configuration.

| Parameter | Value |
|---|---|
| Matrix rank | 8 |
| $\alpha$ | 16 |
| Dropout | 0.05 |
| Target modules | { "q_proj", "v_proj" } |

Table 3: LoRA Configuration Parameters

We used accuracy, F1-Macro and F1-Micro as performance metrics.

## 4 Results

### 4.1 Impact of retrieval

A question is whether retrieval is relevant for our task. As previously said : Our query is always the same - finding the type of the given entity. Moreover, there is no such thing as a target entity in our problem. While the initial model aims at retrieving a desired entity and generate an answer for the user from it, we simply need to generate a predicted entity type.

The retriever 2 acts as a filter on a subgraph $G_k(v) = (V_k, E_k)$, thus reducing the size of the graph. We chose the unmodified model (see 8c), feeding in the query the entity as raw text. We ran a training on the GNN weights and the head classifier with DBPedia6k.

| Retrieval | Validation Accuracy |
|---|---|
| Yes | 0.662 |
| No | **0.764** |

Table 4: Retrieval vs Accuracy results - GNN + Frozen Llama3-8B - DBPedia6k

Retrieval was, as expected, not quite relevant.

### 4.2 Impact of Entity Text Description

As explained in 2.2, we tried two different entity text description. On DBPedia6k, still with the with LLM Llama 3-8B (LoRA finetuning) + GNN (see 4b).

| Language Model | Entity Text | Validation Accuracy |
|---|---|---|
| Llama3-8B (LoRA) | Raw | 0.764 |
| Llama3-8B (LoRA) | Natural | 0.816 |

Table 5: Accuracy results for Llama3-8B (8bit LoRA) with different Entity Text formats.

### 4.3 Impact of Fine-Tuning

We assess the relevance of fine-tuning our Llama model. In this experiment, fine-tuning is performed using a Low-Rank adaptation with 8-bit quantization. Specifically, we set the rank to $r = 8$, using a scaling factor $\alpha = 16$ and a dropout factor of 0.05. From now on, each time a fine-tuning is mentioned in experiments, those settings are used. DBPedia6k

| Language Model | Entity Encoding | Validation Accuracy |
|---|---|---|
| Llama3-8B (Frozen) | Natural | 0.808 |
| Llama3-8B (LoRA) | Natural | **0.816** |

Table 6: Validation Accuracy for a fine-tuned LLM (LoRA) vs. Frozen LLM

### 4.4 Accuracy per class

As expected, the representation of each class in the training dataset being unequal, the classification accuracy for rare classes is lower than that of the most represented classes.

| Dataset | Retrieval | GNN Encoding | Lang. Mod. | Entity Encoding | Test Acc. | MicroF1 | MacroF1 |
|---|---|---|---|---|---|---|---|
| DBPedia6k | Yes | Yes (GT) | Llama3-8B (LoRA) | Raw | 0.662 | - | - |
| DBPedia6k | No | Yes (GT) | Llama3-8B (LoRA) | Raw | 0.764 | - | - |
| DBPedia6k | No | Yes (GT) | SentenceBERT (Frozen) | Raw | 0.581 | 0.581 | 0.365 |
| DBPedia6k | No | Yes (GT) | Llama3-8B (LoRA) | Natural | 0.816 | - | - |
| DBPedia6k | No | Yes (GT) | Llama3-8B (Frozen) | Natural | 0.808 | 0.808 | 0.739 |
| DBPedia35k | No | None | Llama3-8B (Frozen) | Natural | 0.241 | 0.24 | 0.05 |
| DBPedia35k | No | Yes (GT) | SentenceBERT (FT) | Sentence desc. | **0.89** | **0.89** | **0.8** |
| DBPedia35k | No | Yes (GT) | Llama3-8B (LoRA) | Natural | 0.798 | 0.798 | 0.557 |

Table 7: Performance comparison of different models on DBPedia datasets.

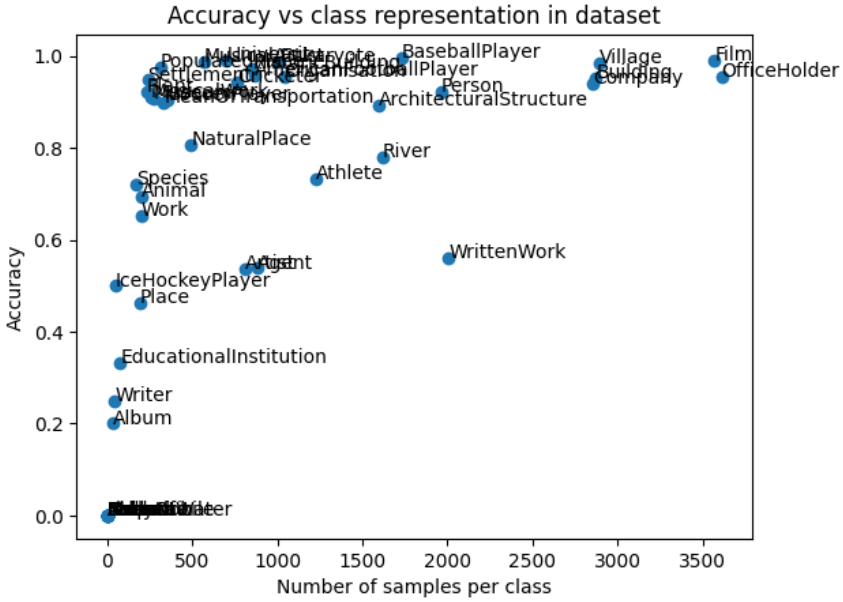



Fig. 5: Prediction accuracy of each class according to their frequency in the dataset (Model GNN + Frozen LLM on DBPedia35k)

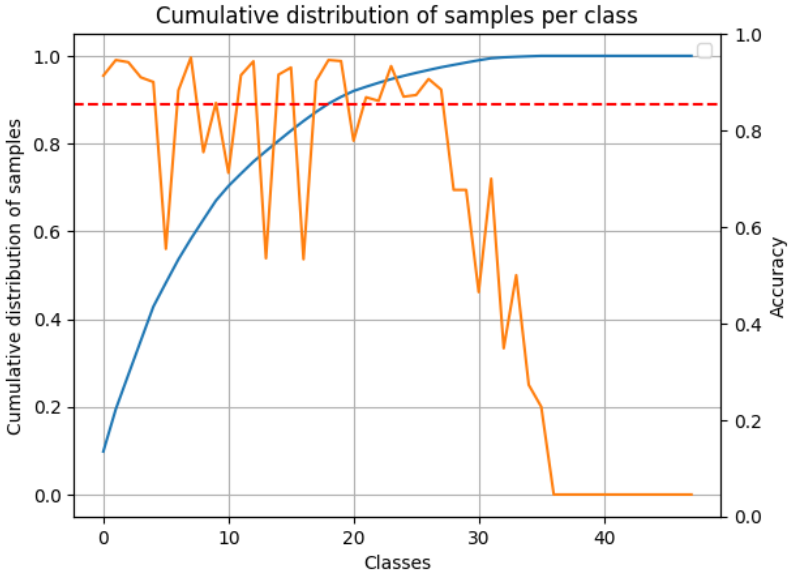



Fig. 6: Cummulative proportions of classes in the whole dataset (sorted by highest occurence) and respective prediction accuracy. The red line represents the global predictiona accuracy

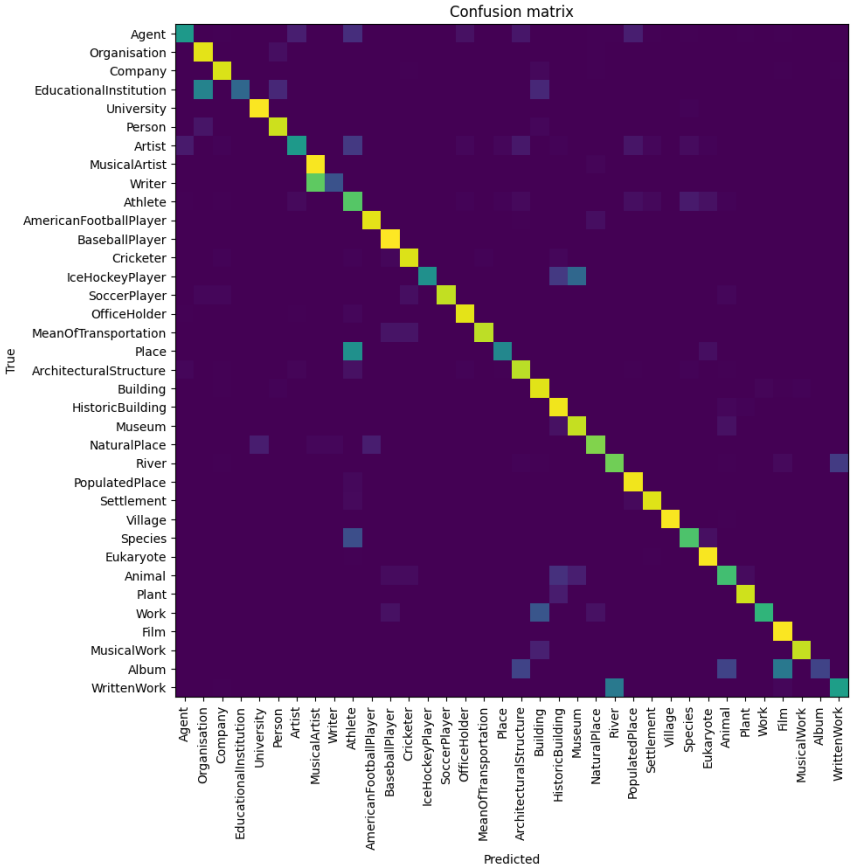
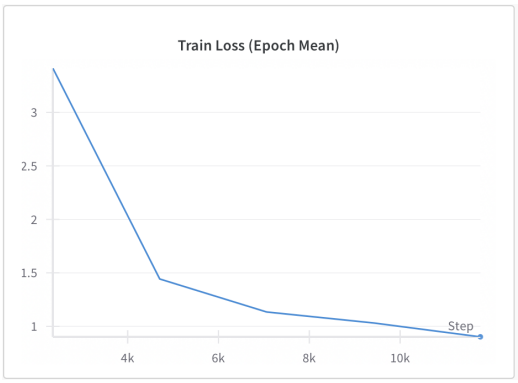
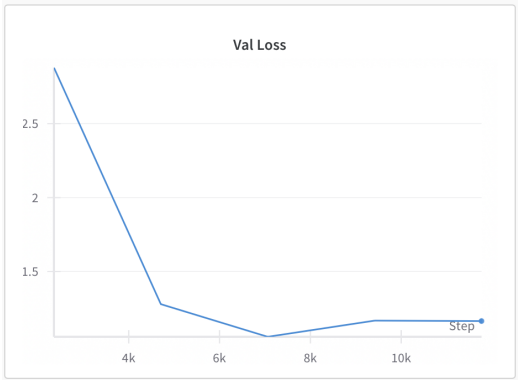


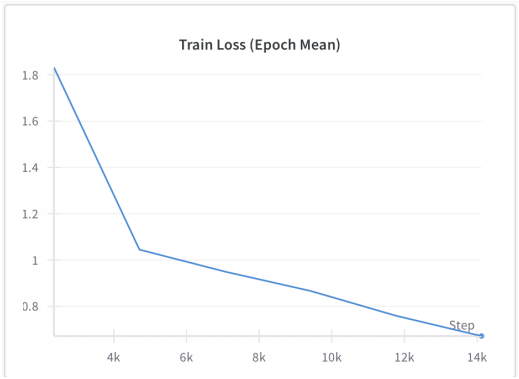

Fig. 7: Confusion matrix of predicted vs true labels



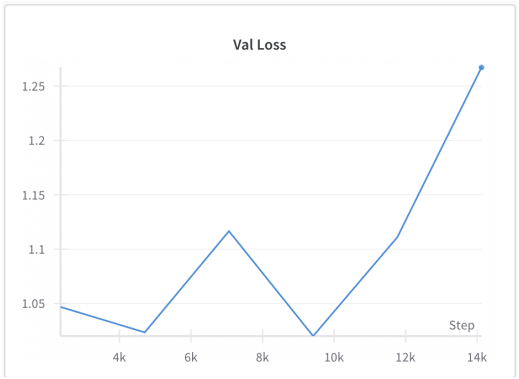

(a) Train Loss - GNN + Llama3 8b (Frozen)




(b) Test Loss - GNN + Llama3 8b (Frozen)




(c) Train Loss - GNN + Llama3 8b (LoRA FT)




(d) Test Loss - GNN + Llama3 8b (LoRA FT)

Fig. 8: Plots of Train and Validation Loss for a task with LoRA finetuning and the other with frozen LLM weights

## 5 Summary & Future Directions

In conclusion, this project highlighted both the importance of a message passing model such as a Graph Transformer to take into account the graph structure of the input data, but also the sub-optimal use of a LLM for classification. This could be due to several factors :

- Not using the full potential of the LLM
- Not using a Knowledge-Graph specific graph network to create embeddings for the graph nodes.
- Combining representations : we are using the concatenation operator, maybe there would be a smarter way to combine the expressiveness from the GNN embedding and the LLM / Bert hidden representation for the nodes. Some joint training, knowledge distillation from LLM to GNN where considered at the begining of the project, but not retained going forward. Maybe there is a path to investigate there.

An idea that would be worth investigating in the future (that was suggested to me by Cristian Santini, would be to do a per-class probability estimation. For each class, using the final representation of an entity as well as a vector representation of the class (ie using the hidden representation from a BERT-like Model) to compute class-entity similarity.

For a given entity $s$, and an encoded vector representation $z_s$ and a given class $c$ encoded as $z_c$:

$$P(s \in c) = f(z_c, z_s)$$

.

For example

$$P(s \in c) = proj[z_s]^T z_c$$

where $proj$ is a fully connected layer projecting $z_s$ in the embedding space of $z_c$

## References


1. et al., A.G.: The llama 3 herd of models (2024), https://arxiv.org/abs/2407.21783
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K., Noy, N.F., Allemang, D., Lee, K., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, Lecture Notes in Computer Science, vol. 4825, pp. 722–735. Springer (2007). https://doi.org/10.1007/978-3-540-76298-0_52, https://doi.org/10.1007/978-3-540-76298-0_52
3. Biswas, R., Sofronova, R., Sack, H., Alam, M.: Cat2type: Wikipedia category embeddings for entity typing in knowledge graphs. In: Gentile, A.L., Gonçalves, R. (eds.) K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021. pp. 81–88. ACM (2021). https://doi.org/10.1145/3460210.3493575, https://doi.org/10.1145/3460210.3493575
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM SIGMOD international conference on Management of data (2008)
5. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
6. Dettmers, T., Pasquale, M., Pontus, S., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the 32th AAAI Conference on Artificial Intelligence (2018)
7. He, X., Tian, Y., Sun, Y., Chawla, N.V., Laurent, T., LeCun, Y., Bresson, X., Hooi, B.: G-retriever: Retrieval-augmented generation for textual graph understanding and question answering (2024), https://arxiv.org/abs/2402.07630
8. Jain, P., Kumar, P., Chakrabarti, S., et al.: Type-sensitive knowledge base inference without explicit type supervision. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 75–80 (2018)
9. Jin, H., Hou, L., Li, J., Dong, T.: Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In: 27th International Conference on Computational Linguistics (2018)
10. Jin, H., Hou, L., Li, J., Dong, T.: Fine-grained entity typing via hierarchical multi graph convolutional networks. In: Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (2019)
11. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conference on artificial intelligence (2015)

12. Melo, A., Paulheim, H., Völker, J.: Type Prediction in RDF Knowledge Bases Using Hierarchical Multilabel Classification. In: WIMS (2016)
13. Paulheim, H., Bizer, C.: Type Inference on Noisy RDF Data. In: ISWC (2013)
14. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Proceedings of the European semantic web conference (2018)
15. Tong, P., Zhang, Q., Yao, J.: Leveraging domain context for question answering over knowledge graph. Data Science and Engineering (2019)
16. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM (2014)
17. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.S.: Explainable reasoning over knowledge graphs for recommendation. In: Proceedings of the AAAI conference on artificial intelligence (2019)
18. Yaghoobzadeh, Y., Adel, H., Schütze, H.: Corpus-level fine-grained entity typing. J. Artif. Intell. Res. (2018)
19. Yaghoobzadeh, Y., Schütze, H.: Multi-level representations for fine-grained typing of knowledge base entities. In: 15th Conference of the European Chapter of the Association for Computational Linguistics (2017)
20. Zhuo, J., Zhu, Q., Yue, Y., Zhao, Y., Han, W.: A neighborhood-attention fine-grained entity typing for knowledge graph completion. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. pp. 1525–1533 (2022)