Natural Entity Typing in Wikidata

Master's Thesis Presentation

Johannes Mannhardt

Examiner: Prof. Dr. Hannah Bast

2nd Examiner: Prof. Dr. Abhinav Valada

Adviser: Natalie Prange

March 11th, 2025

University of Freiburg, Chair for Algorithms and Data Structures

Introduction: Wikidata

Wikidata: Structured collaborative knowledge base.

Wikidata entities:

- **Items (QID)**: **Everything there is**, including people, places, concepts, etc. ($\sim$116 million items)
- **Properties (PID)**: Relationships between items ($\sim$12,400 properties)

Each entity usually has a label and a description.

Introduction: Wikidata

Example triples (statements):

- `Berlin (Q64) → country (P17) → Germany (Q183)`
- `iPhone (Q2766) → developer (P178) → Apple (Q312)`
- `Europe (Q46) → part of (P361) → Eurasia (Q5401)`

Wikidata entities form a Knowledge Graph:

- **Nodes:** items
- **Edges:** properties

Introduction: Wikidata

Ontological Properties:

- **P31 (instance of):** Assigns an entity to a class
  - "Germany" as instance of "Country"
- **P279 (subclass of):** Defines hierarchical relationships between classes
  - "non-coding RNA" as subclass of "RNA"
  - P279 is transitive

Introduction: Entity Typing

Entity Typing: Assign types to entity mentions, e.g., “Paris” as City or “Einstein” as Person.

Why Entity Typing?

- Enhances NLP tasks:
  - Named Entity Recognition
  - Search
  - Question Answering

Existing approaches: typically rely on context, crowd-sourcing, and smaller knowledge bases.

Problem Statement

Goal: Natural entity typing in Wikidata

- Assign the most natural, single type to each Wikidata entity

Challenges:

- What even is a “natural type”?
- Ambiguity from overlapping types
- Inconsistent/wrong use of P31 and P279
- Lack of clear ontological constraints for new entities

Problem Statement

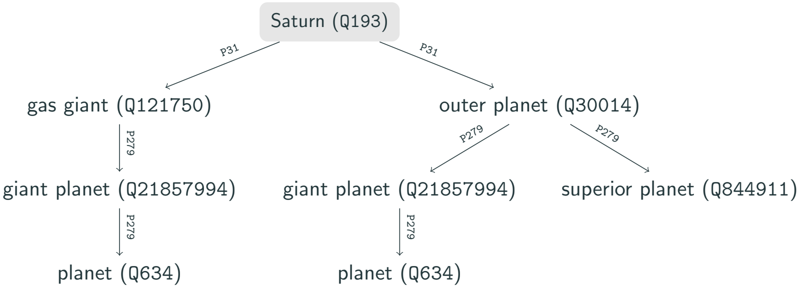

Figure 1: Multiple types assigned to *Saturn*, demonstrating varying levels of specificity.

Problem Statement

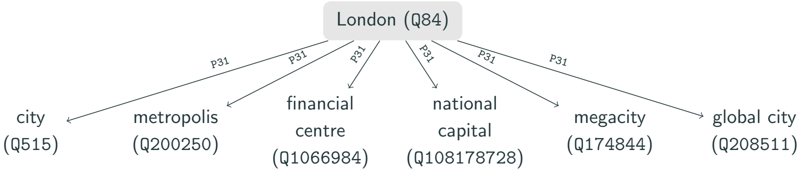

Figure 2: Multiple types directly assigned to the entity *London*, highlighting overlapping categories.

Methodology: Candidate Selection

Approach: Identify potential types from existing connections

- P31 (instance of)
- P279 (subclass of)

Benefit: Clear candidate sets ensure consistency, simplify evaluation, and directly provide types as labels with corresponding QIDs.

Methodology: Selection Criteria

Main Criteria:

- **Layer 1**: Select types reachable via `P31` (instance of).
- **Layer 2+**: Select types reachable via `P279` (subclass of).

Intuition behind criteria:

- If an entity is an instance of type A, and A is a subclass of B, the entity implicitly inherits B due to transitivity of `P279`.

Methodology: Selection Criteria

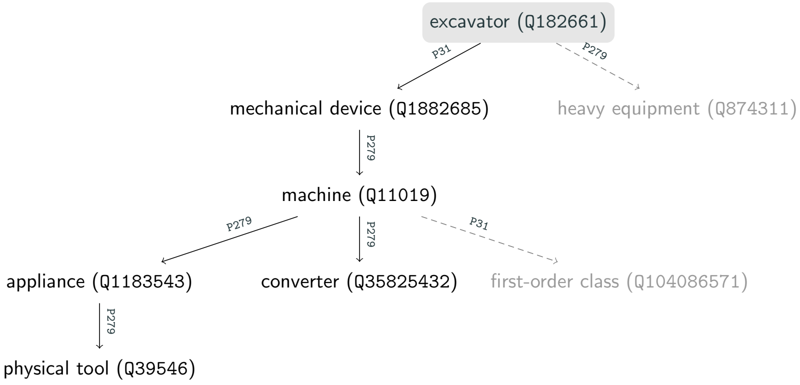

Figure 3: Systematic identification of candidate types based on Wikidata properties (P31 and P279).

Methodology: Selection Criteria

Exceptions:

- **Exception 1:** If no `P31` at Layer 1, allow `P279` at Layer 1.
  (Reason: Wikidata does not strictly distinguish between classes and instances.)
- **Exception 2:** If Layer 2 has no `P279` types and only one valid Layer 1 connection, reuse `P31` at Layer 2.
  (Reason: Occasionally an entity is treated as a class without itself being a subclass of another class.)

Methodology: Training Data Generation

Challenges:

- Ensuring diversity in data coverage
- Manual labeling is slow and expensive

Approach: Automated labeling using LLM (Gemini Flash 1.5)

- Provide entity label, description, and candidate types
- LLM selects best-fitting type based on detailed system-string

Methodology: Training Data Generation

Ensuring Structural Diversity:

1. Sample millions of entities from Wikidata
2. Filter out entities without labels/descriptions
3. Keep up to 3 entities per unique ontological position

Outcome:

- $\sim$169,000 diverse labeled entities ($\sim$24 hours for generation)

Methodology: Feature Extraction

Graph-based Features:

- Node degrees (log in/out) based on different properties

Semantic Embeddings:

- **RDF2Vec** – (Random walks + SkipGram) for knowledge graph embeddings [1]
- **Universal Sentence Encoder** – for description-based embeddings [2]

Methodology: Model Selection

Architectures:

- Feedforward Neural Network (FNN)
- Graph Neural Networks (GNNs)
  - GraphSAGE [3]
  - Graph Attention Network (GAT) [4]
  - Relational Graph Convolutional Network (R-GCN) [5] with additional properties/edge types

Training: Models trained using cross-entropy loss, regularization, hyperparameter tuning, class weights, and candidate masking.

Methodology: Candidate Masking

Problem: Large output space (> 4000 types)

Solution: Candidate masking to restrict predictions to valid types.

Benefits:

- Ensures ontological consistency
- Reduces complexity for the models
- Significantly improves accuracy

Methodology: Candidate Masking

Logits:	0.11	0.84	0.19	-0.23	0.42	1.10	-0.20	0.99	0.01	-0.75
Mask:	1	0	0	0	1	0	0	1	0	0

$\downarrow$ $\textbf{Logits} + \log(\textbf{Mask} + 10^{-9})$

Masked Logits:	0.11	-19.88	-20.53	-20.95	0.42	-19.62	-20.92	0.99	-20.71	-21.47

$\downarrow$ **Softmax**

Probabilities:	0.209	0.000	0.000	0.000	0.285	0.000	0.000	0.505	0.000	0.000
Types:	Work	Dog	Lake	Cat	Artwork	Print	Wall	Painting	Fruit	Car

Figure 4: Candidate masking restricts model predictions to candidate types.

Methodology: Benchmark Datasets

Two Benchmarks: Total of 800 human-annotated entities

- 500 sampled entities
- 300 hand-picked entities

Metrics:

- Top-1 accuracy
- Mean Reciprocal Rank (MRR)
- Sometimes allow > 1 types for an entity to avoid penalizing small differences in specificity/focus

Results

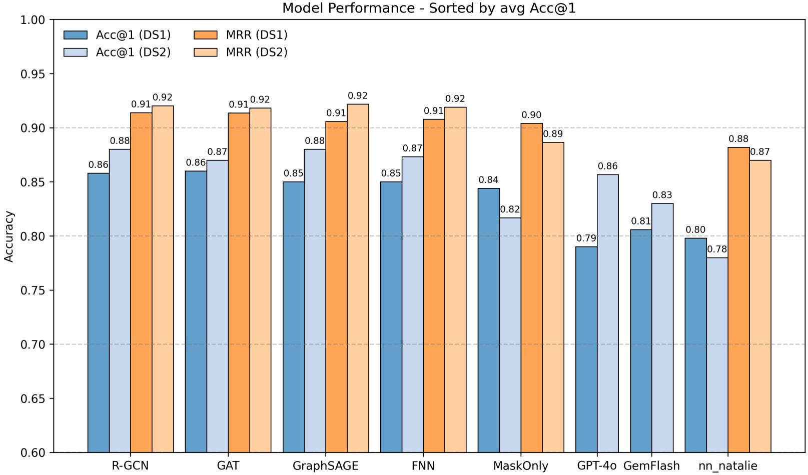

Figure 5: Model performance comparison on human-annotated benchmarks with candidate masking.

Results

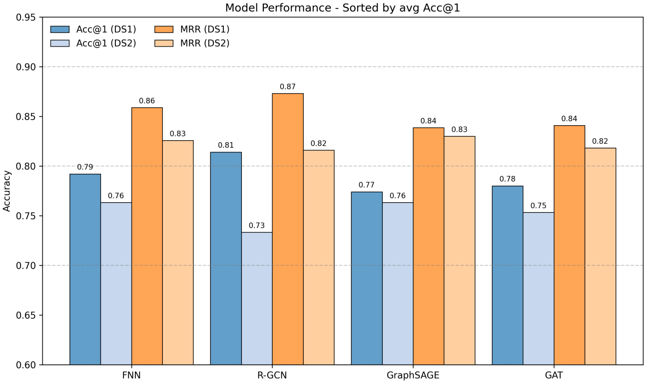

Figure 6: Model performance comparison on human-annotated benchmarks without candidate masking.

Examples of Correct Predictions

Entity (Wikidata ID)	Predicted Type (Wikidata ID)	Probability (%)
Mona Lisa (Q12418)	painting (Q3305213)	98.4
	work of art (Q838948)	0.6
	drawing (Q93184)	0.3
Costa Concordia (Q190542)	shipwreck (Q852190)	67.8
	ship (Q11446)	5.9
	boat (Q35872)	4.8
Mount St. Helens (Q4675)	volcano (Q8072)	66.9
	mountain (Q8502)	23.4
	landform (Q271669)	1.3
baseball cap (Q639686)	clothing (Q11460)	24.6
	headgear (Q14952)	6.5
	hat (Q80151)	6.0

Table 1: Examples of correct predictions made by the masked FNN model.

Examples of Incorrect Predictions

Entity (Wikidata QID)	Predicted Type (Wikidata QID)	Probability (%)
Kreuzberg (Q308928)	locality of berlin (Q35034452)	72.6
	populated place (Q123964505)	12.8
	neighborhood (Q123705)	9.7
Wilhelma (Q679067)	garden (Q1107656)	80.2
	botanical garden (Q167346)	17.1
	park (Q22698)	0.4
cinnamon (Q28165)	substance (Q378078)	12.3
	material (Q214609)	11.6
	fiber (Q161)	11.3
quadrate bone (Q589072)	class of anatomical entity (Q112826905)	98.1
	class (Q5127848)	1.5
	entity (Q35120)	0.1

Table 2: Examples of incorrect predictions made by the masked FNN model.

Limitations

Key Limitations:

- **Subjectivity:** Ambiguity in natural type selection
- **Noisy Training Data:** LLM-generated labels are inconsistent
- Wikidata's characteristics complicate candidate selection

Future Work

Areas for Future Work:

- Crowdsourced benchmarks
- Refine training-data with human-in-the-loop feedback
- Incorporate more sophisticated LLMs
- Propose new Wikidata property for natural types

Acknowledgments & References

Acknowledgments:

- Prof. Dr. Hannah Bast
- Prof. Dr. Abhinav Valada
- Natalie Prange

Thank you for your attention!
Questions?

LLM System String i

```
Objective:
From a pre-selected list, choose the most natural, everyday-language type
for a Wikidata item based on its label and description.

Rules:
- Your choice **must** be one of the provided pre-selected types.
- Generally, choose the broadest category that still represents a natural
  and commonly understood everyday term (e.g., choose 'Disease' over
  'Infectious Disease', 'RNA' over 'Non-coding RNA', 'Star' over
  'Variable Star', etc.).
- However, if a more specific category is a **very commonly recognized
  and understood** everyday category, choose it. Think about what a
  typical person would call it (e.g., 'Lake' rather than 'Body of Water',
  'Village' rather than 'Human Settlement', etc.).
- Again, avoid too much specificity (e.g., choose 'Surname' over
  'Japanese Surname', 'Monument' over 'Heritage Monument', etc.).
- Generally speaking, a good type is short and intuitive, while a bad
  type is long and overly specific.
- Return only the type (with label and QID).
- Do not output JSON.

Examples:
- Berlin -> City
- Albert Einstein -> Person
```

LLM System String ii

```
- T-Shirt -> Clothing
- Germany -> Country
- Carbon Dioxide -> type of chemical entity
- Breaking Bad -> Television Series
- Jazz -> Musical Genre
- Sagrada Família -> Church
- Green Tea -> Drink
- FC Bayern Munich -> Sports Club (Football Club would be too specific)

Important: A type as long and specific as e.g. 'civil parish in Ireland'
will **almost never** be a good choice (just 'civil parish' would be much
better). Remember, a type should be short, intuitive, and represent a
commonly understood category.
```

Related Work

Comparison of Entity Typing Approaches:

Approach	Knowledge Base	Context?	Manual Data	Scale
Tipalo	DBpedia	No	Yes	Medium
TRank/TRank++	Multiple	Yes	Yes	Small
ManyEnt	Wikidata	Yes	Yes	Medium
RL-TRank	Multiple	Yes	Yes	Medium
Our Approach	Wikidata	No	No	Large

References i

P. Ristoski and H. Paulheim, “RDF2Vec: RDF graph embeddings for data mining,” in *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Proceedings, Part I*, (Berlin, Heidelberg), pp. 498–514, Springer, 2016.

D. Cer, Y. Yang, S.-Y. Kong, and et al., “Universal sentence encoder for english,” in *Proc. EMNLP 2018: Sys. Demos*, (Brussels, Belgium), pp. 169–174, ACL, 2018.

W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *CoRR*, vol. abs/1706.02216, 2017.

References ii

- P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.
- M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," 2017.