Evaluation Event II - *RedFlickeringCandle*

Haizhou Zheng (*haizhou.zheng@uzh.ch*)
Zhenmei Hao (*zhenmei.hao@uzh.ch*)

November 3, 2025

1 Introduction

We build a lightweight but reliable QA agent over a given knowledge graph (KG) that supports two answering routes: a **factual** route that compiles a SPARQL-style lookup, and an **embedding** route that retrieves answers in the vector space. If a question explicitly requires a factual (or embedding) approach, the agent only executes that route; otherwise it returns both.

2 Capabilities

Overview.

```
User Query → NLQ Parser → {EntityLinker, RelationMapper} →
    ┌──────────────────────────────────────┐
    │      Factual Executor (RDFlib)       │
    │                  or                  │ → Answer Composer
    │ Embedding Executor (TransE-style)    │
    └──────────────────────────────────────┘
                            (verbalization)
```

Factual questions. (1) The NLQ parser normalizes quotes and extracts entity mentions and relation triggers. (2) The EntityLinker builds a label index from the KG and uses RapidFuzz to return top-k entity candidates. (3) The RelationMapper maps trigger tokens to a KG predicate IRI. (4) The GraphExecutor then reads $(s, p, ?o)$ triples via RDFlib and resolves objects to readable labels from `rdfs:label`, `skos:prefLabel/altLabel`, `schema:name`, and `wdt:P1476`, followed by de-duplication and verbalization.

Embedding. This path answers the form $(subject, predicate, ?object)$. We pick the first subject candidate that has an embedding vector. Candidate tails are *first* collected from the 1-hop neighborhood $(subject, predicate, ?o)$; if empty, we *fallback* to all tails of that predicate across the graph with downsampling (`MAX_TAILS`). We compute a TransE-style target $t = \mathrm{norm}(s + r)$ and score candidates by cosine similarity. For the top hit we attach a short type from `rdf:type/wdt:P31`; we also estimate the predicate's majority object type as an *expected type* for display consistency.

3 Adopted Methods

- **RDFlib**[1]: used for parsing the KG and querying triples (s, p, o) in Python with stable APIs and serializers.
- **RapidFuzz**[2]: fast fuzzy string matching for entity labels to construct top-k candidates robust to typos and variants.
- **Translation-based scoring** [1]: we adopt $s + r \approx o$ and $o - r \approx s$ in a TransE-style setup, using cosine similarity.

4 Examples

- **Factual.** Given "*Please answer this question with a factual approach: Who is the director of 'Good Will Hunting'?* " The factual executor yields *Gus Van Sant*, which matches the KG.
- **Embedding.** Given "*What is the genre of 'Shoplifters'?*" The top hit is *drama film* with type *film genre* (e.g., Q201658).

5 Additional Features

- **Direction-aware candidates for embeddings.** For tail prediction we first gather 1-hop tails $(s, p, ?o)$ via graph lookup and only then fall back to global predicate tails; for head prediction we symmetrically use $(?s, p, o)$. This subject-/object-first policy keeps candidates semantically tight.
- **Efficient retrieval.** We downsample large global pools with `MAX_TAILS` and compute Top-k using vectorized `argpartition`, yielding low latency even on bigger KGs.
- **Readable labels and types.** We resolve labels from multiple RDF properties (RDFS/SKOS/Schema.org/Wikidata title) and extract types from `rdf:type/wdt:P31`. For tail prediction, we also estimate a predicate-level *expected object type* to satisfy the task's "return the entity type" requirement.

6 Conclusions

Our agent reliably answers KG questions via factual and embedding routes. Next steps include multimedia, recommendation question types.

Contributions. H. Zheng implemented the service skeleton, entity linker and NLQ parsing; Z. Hao implemented the intent routing and report. Both authors co-designed the other source code of agent.

[1] `https://rdflib.readthedocs.io/`

[2] `https://maxbachmann.github.io/RapidFuzz/`

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.