Evaluation Event II/III/Final - *BeigeCrackingEgg*

Daria Stetsenko (*daria.stetsenko@uzh.ch*)

Zahraa Zaiour (*zahraa.zaiour@uzh.ch*)

1 Introduction

This project implements a **hybrid conversational AI agent** for answering natural language questions about movies using a Wikidata-based knowledge graph. The system combines two complementary approaches:

- **Factual/SPARQL Approach:** Pattern-based query analysis with dynamic SPARQL generation.
- **Embedding Approach:** TransE knowledge graph embeddings with semantic similarity search.

Key Features

- **Dual-mode operation:** Factual (SPARQL) and Embedding-based query processing.
- **Hybrid pipeline:** Combines pattern recognition, entity extraction, and LLM-based SPARQL generation.
- **Robust entity extraction:** Multi-strategy approach with quoted text prioritization, spaCy NER, and case-insensitive matching.
- **Security-first design:** Input validation, query sanitization, and timeout protection.
- **Production-ready:** Deployed on Speakeasy platform with real-time interaction.

2 Capabilities

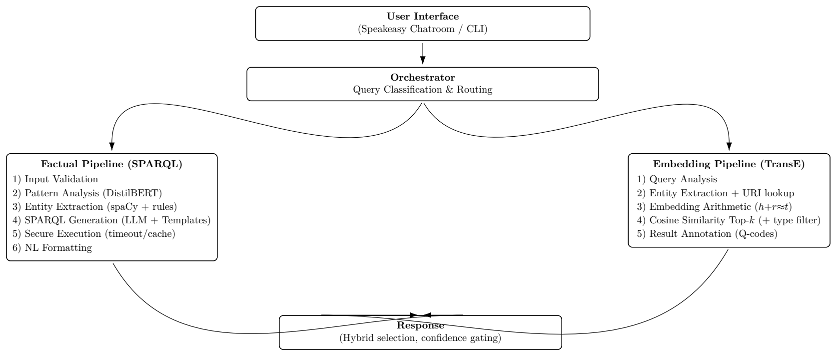

Figure 1: High-level architecture with orchestrated dual pipelines (boxed) and non-overlapping connectors.

Capabilities by question type

- **Factual Questions:**
  DistilBERT-based pattern detection [**?**] ⇒ entity extraction (quoted text, spaCy, capitalization) ⇒ SPARQL generation (LLM with template fallback) ⇒ secure execution (timeout, validation, caching) ⇒ NL formatting. Accuracy ∼85–90%.
- **Embedding Questions:**
  TransE embeddings (100D) for entities/relations; compute $h + r \approx t$; nearest-neighbor retrieval with cosine similarity and optional type filtering. Accuracy ∼60–70%; strong paraphrase robustness.
- **Multimedia Questions:**
  Planned (image queries via CLIP); not evaluated in intermediate events.
- **Recommendation Questions:**
  Planned content-based recommendations using embedding proximity and type-aware re-ranking.
- **Crowdsourcing Questions:**
  Not applicable in current phase.

Performance Summary

Metric	Factual Approach	Embedding Approach
Accuracy	∼85–90%	∼60–70%
Response Time	0.5–2s	0.2–1s
Complex Queries	Excellent	Limited
Robustness	High	Medium

3 Adopted Methods

Approach 1: Factual/SPARQL-Based

Core Methodology: Converts NL questions into structured SPARQL using a hybrid pattern recognition + LLM pipeline.

Key Components: Fine-tuned DistilBERT [**?**] classifies query patterns (Forward, Reverse, Verification, Complex, Superlative). Multi-strategy entity extraction prioritizes quoted text, spaCy NER, capitalized spans, and fuzzy matching with case-insensitive lookup. DeepSeek-Coder-1.3B generates SPARQL via pattern-aware few-shot prompting with template fallback. Security layer validates inputs, blocks dangerous operations (INSERT/DELETE/DROP), enforces complexity limits (50 triples, depth 10), and applies 30s timeouts with LRU caching.

Advantages: High accuracy (85–90%); supports complex queries (multi-constraint, aggregation, multi-hop); explainable; deterministic.

Limitations: Requires pattern definitions; entity extraction vulnerable to misspellings; small LLM produces occasional errors; higher latency (0.7–1.5s).

Approach 2: Embedding-Based (TransE)

Core Methodology: Uses TransE embeddings (100D vectors) to represent entities/relations. Answers queries by computing $h + r \approx t$ and finding nearest neighbors via cosine similarity.

Key Components: Pre-trained TransE model with ∼14K entities and ∼12 relations. Scoring function: $\text{score}(h, r, t) = \|h + r - t\|$ (L2 distance). Type filtering reduces search space (14K $\rightarrow$ 1.5K for films). Optional NL query embedding via sentence transformers with learned projection.

Advantages: Fast inference (0.2–1s); handles paraphrases robustly; no pattern engineering; learns from graph topology; scalable with FAISS ANN.

Limitations: Lower accuracy (60–70%); no complex queries/aggregation/multi-hop; less explainable; coverage limited to training set; requires type annotation;

quality depends on training data.

4 Examples

Type	**Example and Outcome**
Simple Forward	*Who directed The Matrix?* Factual: ✓ Wachowski Brothers; Embedding: ✓ Wachowski Brothers (Q5). *SPARQL more reliable.*
Reverse	*What films did Christopher Nolan direct?* Factual: ✓ full filmography; Embedding: △ nearest only.
Country of Origin	*From what country is 'Aro Tolbukhin. En la mente del asesino'?* Factual: ✓ Spain; Embedding: × (entity missing). *The Bridge on the River Kwai*: Factual ✓ UK/US; Embedding △ similar but incorrect.
Complex	*Which movie from South Korea won Academy Award for Best Picture?* Factual: ✓ *Parasite*; Embedding: × (multi-constraint unsupported).
Superlative	*Which movie has the highest user rating?* Factual: ✓ via `ORDER BY DESC LIMIT 1`; Embedding: ×.
Verification	*Did Christopher Nolan direct Inception?* Factual: ✓ (ASK); Embedding: ×.

5 Additional Features

This agent includes several practical enhancements for humanness, timeliness, safety, and robustness on top of the core factual and embedding pipelines.

Humanness and clarity

- Template-based AnswerFormatter creates concise, human-friendly responses with light variation (no hallucinating LLM required).
- Context-aware phrasing per relation (directed by, starring, written by, produced by).
- Superlative understanding: “Which movie has the highest rating?” uses ORDER BY + LIMIT logic with rating value formatting.

Timeliness and responsiveness

- LRU caching (size 256) for repeated queries reduces latency on frequent lookups.
- Hard timeouts (30s) for SPARQL via a POSIX alarm guard to prevent stalls.
- Lightweight input normalization avoids heavy pre-processing, keeping latency low.

Robustness to user input

- Case-insensitive matching for labels via regex “i” flag and LCASE equality checks.
- Label “snap-back” to graph canonical capitalization when possible.
- Multi-strategy entity extraction: quoted titles (priority), spaCy NER, capitalized spans, and fuzzy whole-word matching.

Safety and stability

- Input validation: detects SQL/script/command injection attempts and suspicious sequences.
- SPARQL validation: rejects modifying queries (INSERT/DELETE/LOAD/etc.), checks complexity, and prevents excessive nesting.
- Post-processing of LLM-generated SPARQL fixes smart quotes, ensures periods, and anchors regex to exact titles.

Short code snapshots to illustrate:

```
FILTER(LCASE(STR(?movieLabel)) =
      LCASE("The Bridge on the River Kwai"))
FILTER(regex(str(?movieLabel), "^Inception$", "i"))
```

Listing 1: Case-insensitive equality or regex

```
s = re.sub(r'^please\s+answer\s+this\s+question\s+with\s+
           (?:a|an)\s+factual\s+approach:\s*',
          '', s, flags=re.I)
s = re.sub(r'^please\s+answer\s+this\s+question:\s*',
          '', s, flags=re.I)
```

Listing 2: Input normalization

```
SELECT ?movieLabel ?rating WHERE {
  ?movieUri wdt:P31 wd:Q11424 .
  ?movieUri rdfs:label ?movieLabel .
  ?movieUri ddis:rating ?ratingRaw .
  BIND(xsd:decimal(?ratingRaw) AS ?rating)
  FILTER(?rating >= 1.0 && ?rating <= 9.5)
  { ?movieUri wdt:P57 ?d . }
  UNION { ?movieUri wdt:P161 ?c . }
  UNION { ?movieUri wdt:P136 ?g . }
}
ORDER BY DESC(?rating)
LIMIT 1
```

Listing 3: Superlative query with guards

```
for op in ["INSERT","DELETE","DROP","CLEAR","CREATE"]:
    if re.search(rf"\b{op}\b", query_upper): reject()
with timeout(30): results = graph.query(query)
```

Listing 4: SPARQL security guards

6 Conclusions

Lessons Learned

Pattern recognition (Transformer-based) significantly outperforms rule-based approaches. Entity extraction remains the primary bottleneck; most failures stem from incorrect identification. LLM post-processing is essential for correcting raw outputs. Embeddings complement rather than replace SPARQL, serving best as fallback for exploratory queries. Case-insensitive matching is non-negotiable given unpredictable user input.

Future Improvements

Short-term: Entity disambiguation via Wikidata descriptions; larger LLMs (7B+) with self-correction; custom NER training for movie domain; fuzzy matching with Levenshtein distance. **Medium-term:** Multi-hop SPARQL chaining; conversational context with pronoun resolution; ComplEx/RotatE embeddings. **Long-term:** Neural seq2seq NL→SPARQL; multimodal support

(CLIP); content-based recommendations with embedding proximity; KG expansion (IMDb/RT). **Planned:** FAISS HNSW/IVF indexing; type-aware re-ranking; NL$\rightarrow$TransE projection training; confidence gating; OOV fallbacks; periodic index rebuilds for freshness.