Evaluation Event II/III/Final - *BeigeCrackingEgg*

1st Daria Stetsenko (*daria.stetsenko@uzh.ch*)

2nd Zahraa Zaiour (*zahraa.zaiour@uzh.ch*)

October 30, 2025

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: (1) Factual/SPARQL Approach with pattern-based query analysis and dynamic SPARQL generation, and (2) Embedding Approach using TransE knowledge graph embeddings with semantic similarity search.

Key features include dual-mode operation for both factual and embedding-based query processing, a hybrid pipeline combining pattern recognition and entity extraction with LLM-based SPARQL generation, robust multi-strategy entity extraction with case-insensitive matching, security-first design with input validation and query sanitization, and production-ready deployment on the Speakeasy platform with real-time interaction.

2 Capabilities

Our agent employs a dual-pipeline architecture orchestrated by a central query classifier. The system routes questions to either a Factual Pipeline (SPARQL-based) or an Embedding Pipeline (TransE-based) depending on query characteristics. The Factual Pipeline performs: (1) Input Validation, (2) Pattern Analysis using DistilBERT, (3) Entity Extraction via spaCy and rules, (4) SPARQL Generation using LLM with templates, (5) Secure Execution with timeout and caching, and (6) Natural Language Formatting. The Embedding Pipeline executes: (1) Query Analysis, (2) Entity Extraction with URI lookup, (3) Embedding Arithmetic computing $h + r \approx t$, (4) Cosine Similarity Top-k with type filtering, and (5) Result Annotation with Q-codes. Both pipelines feed into a hybrid response selector with confidence gating.

Factual Questions: Our agent answers factual questions using DistilBERT-based pattern detection [**?**], followed by entity extraction from quoted text,

spaCy NER, and capitalization patterns. SPARQL queries are generated via LLM with template fallback, then executed securely with timeout protection, validation, and caching before formatting results in natural language. Accuracy is approximately 85–90%.

Embedding Questions: The agent uses TransE embeddings (100D) for entities and relations, computing $h + r \approx t$ to find answers through nearest-neighbor retrieval with cosine similarity and optional type filtering. Accuracy is approximately 60–70% with strong paraphrase robustness.

Multimedia Questions: Planned for future implementation using 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: Factual approach achieves 85–90% accuracy with 0.5–2s response time, excellent support for complex queries, and high robustness. Embedding approach achieves 60–70% accuracy with 0.2–1s response time, limited complex query support, and medium robustness.

3 Adopted Methods

Approach 1: Factual/SPARQL-Based

Core Methodology: Converts natural language questions into structured SPARQL using a hybrid pattern recognition and LLM pipeline.

Key Components: Fine-tuned DistilBERT [**?**] classifies query patterns including Forward, Reverse, Verification, Complex, and Superlative types. 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. A security layer validates inputs, blocks dangerous operations such as INSERT/DELETE/DROP, enforces complexity limits of 50 triples and depth 10, and applies 30-second timeouts with LRU caching.

Advantages: High accuracy (85–90%); supports complex queries including multi-constraint, aggregation, and multi-hop reasoning; explainable with human-readable SPARQL; deterministic behavior.

Limitations: Requires explicit pattern definitions; entity extraction vulnerable to misspellings; small LLM (1.3B) produces occasional errors; higher latency (0.7–1.5s).

Approach 2: Embedding-Based (TransE)

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

Key Components: Pre-trained TransE model with approximately 14,000 entities and 12 relations. Scoring function: $\text{score}(h, r, t) = \|h + r - t\|$ using L2 distance. Type filtering reduces search space from 14,000 to 1,500 for films. Optional natural language query embedding via sentence transformers with learned

projection.

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

Limitations: Lower accuracy (60–70%); no support for complex queries, aggregation, or multi-hop reasoning; less explainable "black box" similarity; coverage limited to training set; requires type annotation; quality depends on training data.

4 Examples

Simple Forward Query: Given the question "Who directed The Matrix?", our agent first uses DistilBERT to classify it as a forward pattern (movie to director). Entity extraction identifies "The Matrix" via quoted text matching. The LLM generates a SPARQL query with P57 (director) relation, which executes against the knowledge graph. The agent returns "Wachowski Brothers" with high confidence. This is correct as verified in Wikidata. The embedding approach also returns "Wachowski Brothers" with Q5 type annotation, though SPARQL is more reliable.

Reverse Query: For "What films did Christopher Nolan direct?", the pattern classifier identifies a reverse query (person to movies). Entity extraction finds "Christopher Nolan" via spaCy NER. SPARQL generation creates a query traversing the inverse director relation. The factual approach returns the complete filmography including Inception, Interstellar, Dunkirk, etc. The embedding approach returns only nearest neighbors, missing some films due to approximate similarity limitations.

Country of Origin: When asked "From what country is 'Aro Tolbukhin. En la mente del asesino'?", the agent detects a forward country-of-origin pattern. Entity extraction prioritizes the quoted title. The LLM generates a P495 (country of origin) SPARQL query, returning "Spain" correctly. The embedding approach fails as this entity is missing from the training set. For "The Bridge on the River Kwai", the factual approach correctly returns UK/US, while embeddings return similar but incorrect results due to approximate matching.

Complex Multi-Constraint: For "Which movie from South Korea won Academy Award for Best Picture?", the pattern classifier identifies a complex query requiring multiple constraints (country AND award). SPARQL generation creates a query with both P495 (country) and P166 (award) predicates, correctly returning "Parasite". The embedding approach cannot handle multi-constraint queries and fails.

Superlative Query: When asked "Which movie has the highest user rating?", the agent recognizes a superlative pattern. SPARQL generation uses ORDER BY DESC with LIMIT 1 on the rating property, applying guards to filter valid ratings between 1.0 and 9.5. The factual approach succeeds. The embedding approach cannot perform aggregation operations and fails.

Verification: For "Did Christopher Nolan direct Inception?", the pattern

classifier identifies a verification query. SPARQL generation creates an ASK query checking the existence of the director relationship, returning true or false. The factual approach correctly returns "yes". The embedding approach cannot perform boolean verification and fails.

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 without requiring a hallucinating LLM. Context-aware phrasing adapts per relation type (directed by, starring, written by, produced by). Superlative understanding processes questions like "Which movie has the highest rating?" using ORDER BY plus LIMIT logic with rating value formatting.

Timeliness and responsiveness: LRU caching with size 256 reduces latency on repeated queries for frequent lookups. Hard timeouts of 30 seconds for SPARQL execution via a POSIX alarm guard prevent system stalls. Lightweight input normalization avoids heavy pre-processing, keeping latency low.

Robustness to user input: Case-insensitive matching for labels uses regex with "i" flag and LCASE equality checks. Label "snap-back" normalizes to graph canonical capitalization when possible. Multi-strategy entity extraction prioritizes quoted titles, then spaCy NER, capitalized spans, and fuzzy whole-word matching.

Safety and stability: Input validation detects SQL, script, and command injection attempts along with suspicious sequences. SPARQL validation rejects modifying queries including INSERT, DELETE, and LOAD operations, checks query complexity, and prevents excessive nesting. Post-processing of LLM-generated SPARQL fixes smart quotes, ensures proper periods in triples, and anchors regex patterns to exact titles.

Code examples: Case-insensitive matching uses FILTER(LCASE(STR(?movieLabel)) = LCASE("The Bridge on the River Kwai")) or FILTER(regex(str(?movieLabel), "^Inception$", "i")). Input normalization strips instruction-like prefixes using regular expressions while preserving meaning. Superlative queries apply guards like FILTER(?rating >= 1.0 && ?rating <= 9.5) with ORDER BY DESC(?rating) LIMIT 1. SPARQL security blocks dangerous operations and enforces 30-second timeouts on graph queries.

6 Conclusions

Lessons Learned: Pattern recognition using Transformer-based models significantly outperforms rule-based approaches. Entity extraction remains the primary bottleneck, with most failures stemming from incorrect identification. LLM post-processing is essential for correcting raw outputs. Embeddings com-

plement 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 plans include entity disambiguation via Wikidata descriptions, larger LLMs (7B+) with self-correction, custom NER training for the movie domain, and fuzzy matching with Levenshtein distance. Medium-term goals involve multi-hop SPARQL chaining, conversational context with pronoun resolution, and ComplEx/RotatE embeddings. Long-term objectives include neural seq2seq NL-to-SPARQL translation, multimodal support using CLIP, content-based recommendations with embedding proximity, and knowledge graph expansion integrating IMDb and Rotten Tomatoes. Planned technical work includes FAISS HNSW/IVF indexing, type-aware reranking, NL-to-TransE projection training, confidence gating, out-of-vocabulary fallbacks, and periodic index rebuilds for freshness.