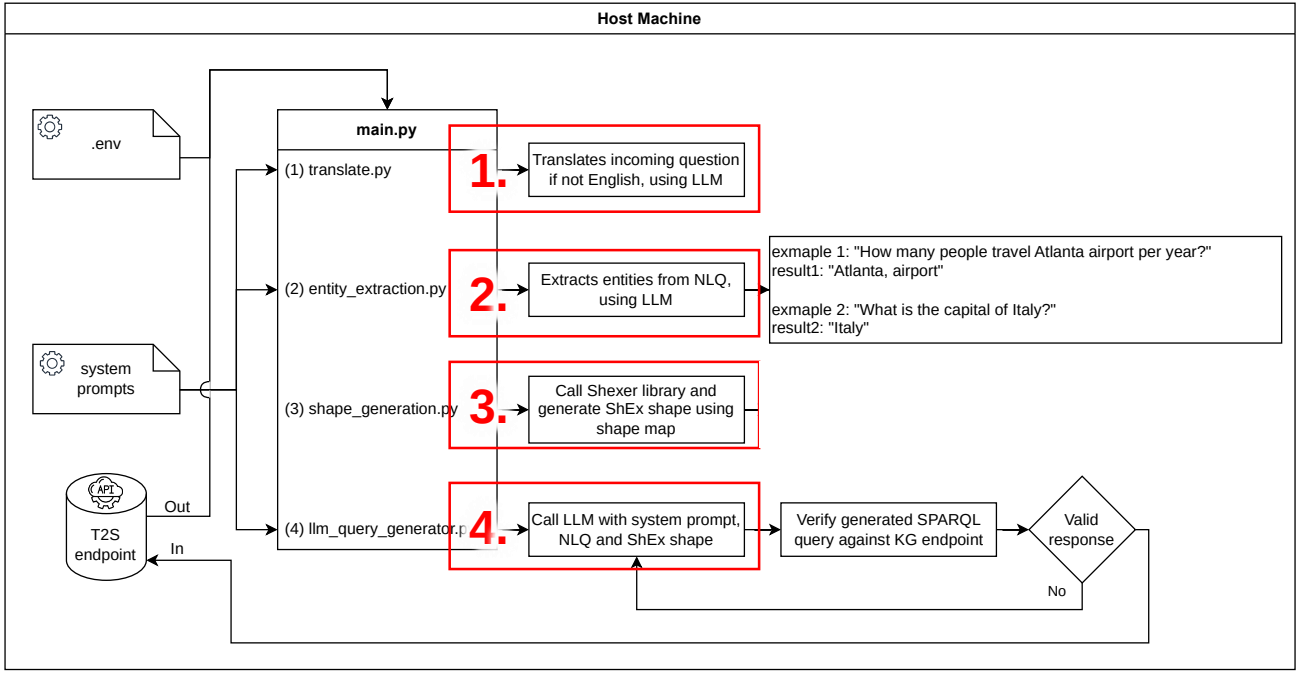
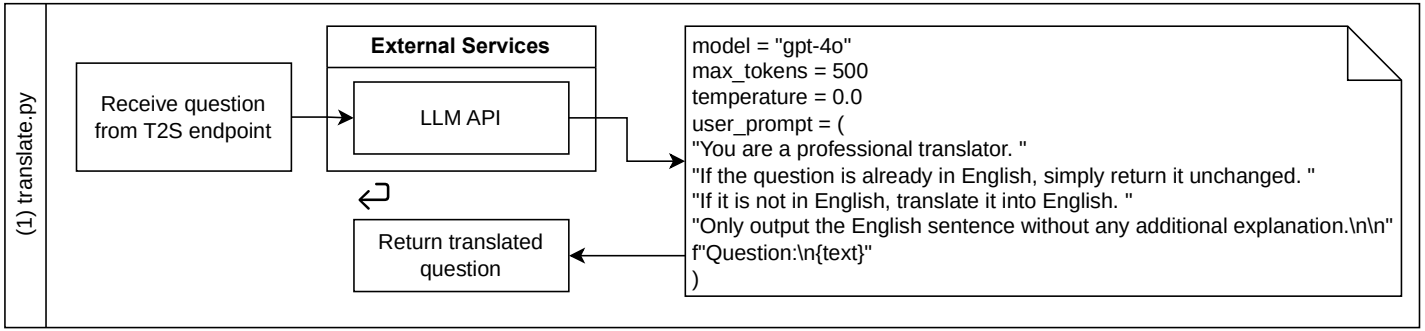
Challenge submission: AIFB

Jan Wardenga and Tobias Käfer, TEXT2SPARQL@ESWC2025

Architecture Overview

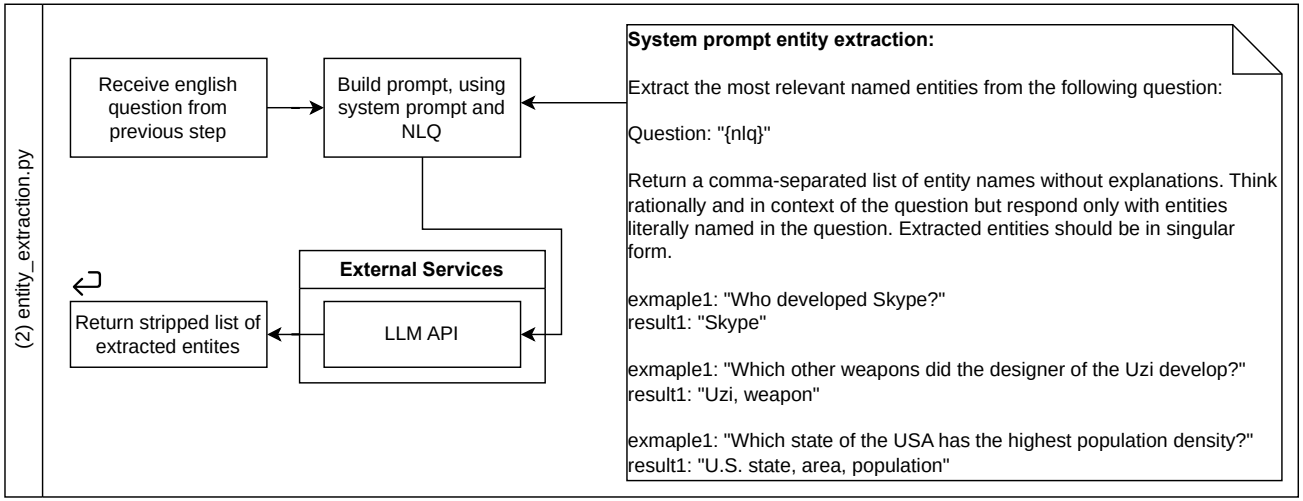


1. Natural Language Questions Translation



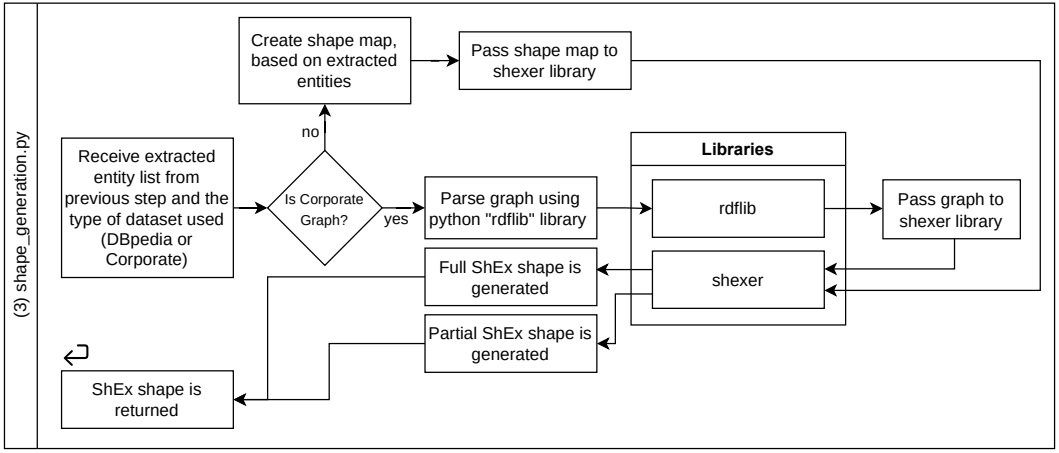
- This module translates the incoming natural language questions (NLQ's) into English, using a large language model (LLM).
- If the input is already in English, it is returned unchanged.
- English was selected as the target language for downstream processing, as it improved entity extraction performance.

2. Entity Extraction



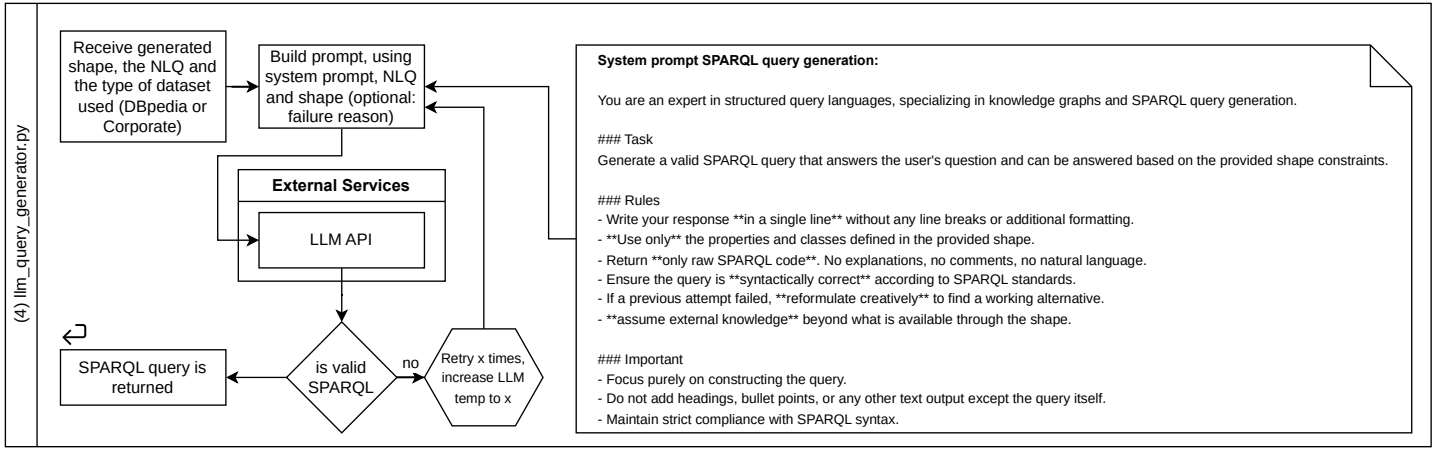
- This module identifies and extracts named entities from the English NLQ's.
- It constructs the LLM prompt by combining a predefined system prompt with the input NLQ.
- The returned list of entities is stripped of unneeded characters, formatted, and passed to downstream modules.

3. Shape Generation



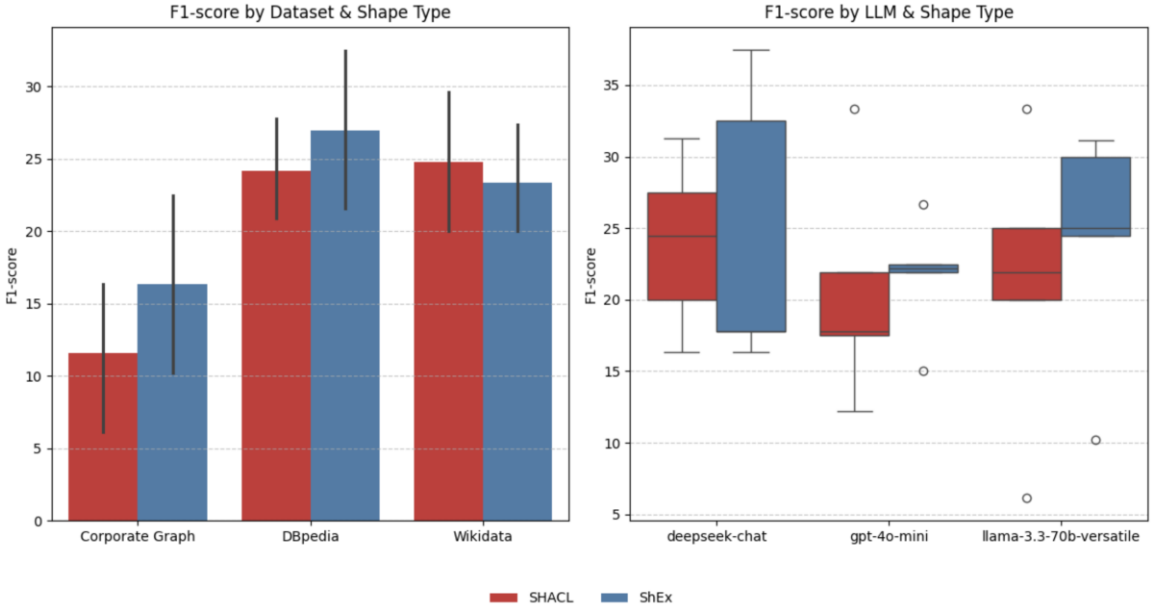
- This module receives the list of extracted entities and the dataset type (DBpedia or corporate graph).
- Based on the dataset type, it decides whether to parse a local corporate RDF graph or use the DBpedia endpoint to generate the shape.
- Two libraries are used:
  - **rdflib**: a standard Python library for working with RDF graphs.
  - **shexer**: a shape generation library that supports the use of shape maps and integrates well with rdflib.
- If a corporate graph is used, the entire graph is parsed and a full ShEx shape is generated.
- If a public dataset is used, a partial ShEx shape is generated
- The resulting ShEx shape is returned for further processing.

4. SPARQL Query Generation



- This module builds a prompt that includes the system prompt, the natural language question (NLQ), the shape constraints, and optionally a failure reason from a previous attempt.
- The prompt is sent to the LLM API for SPARQL query generation.
- The returned output is validated:
  - If the query is valid, it is returned.
  - If invalid, the system retries up to *x* times, incrementally increasing the LLM's temperature to promote variation in reformulations.
- The process ensures strict syntactic compliance and attempts to creatively reformulate failed queries while respecting the shape constraints.

ShEx vs SHACL

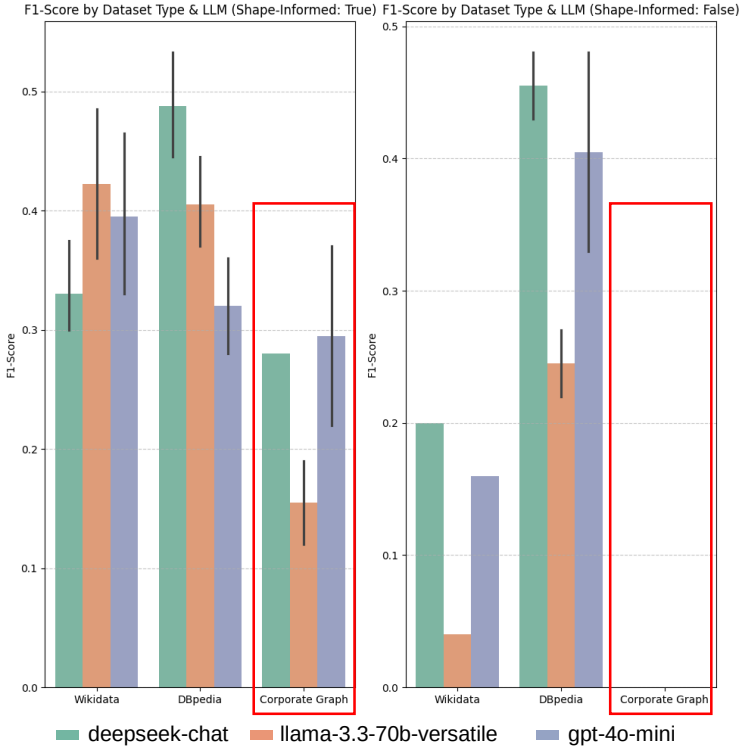


A seperate study was conducted to find out which shape type yields better results for KGQA tasks

- 42 experiments conducted:
  - 30 with shape-informed setups
  - 12 baseline (no shapes)
- Each experiment:
  - 50 questions from a Wikidata, DBpedia and T2S corporate dataset
  - Unique combination of LLM, dataset, shape type, and annotation setting

Key insight:
ShEx outperforms SHACL across most KGQA configurations

Our Results for T2S Corporate Graph



- Reported: F1 scores
- Configurations:
  - **Shape-informed** (using ShEx/SHACL constraints)
  - **Baseline** (no shape information)
- In the **baseline setting**, none of the LLMs were able to generate valid SPARQL queries for the corporate graph
- In contrast, the **shape-informed configuration** enabled LLMs to generate valid and executable SPARQL queries
- Datasets used in the study:
  - Wikidata → QALD-9-plus
  - DBpedia → QALD-9-plus & T2S Challenge
  - Corporate Graph → T2S Challenge