NatLangKG

AI-Enhanced Semantic Interfacing for Decentralized Knowledge Graphs

Document status: First Draft

Last updated: 2023-09-25

Introduction

This project aims to harness the capabilities of advanced language models to create a natural language interface for the OriginTrail Decentralized Knowledge Graph (DKG). We plan to empower non-technical users to query Knowledge Assets with ease, transforming natural language input into SPARQL and providing understandable natural language outputs. Our initiative promotes a broader adoption of the DKG by making it more user-friendly, while also contributing to the democratization and trustworthiness of AI by rooting the model in a verifiable data source.

To facilitate integration with other projects and maximize synergy, our project will utilize existing transformer-based text embeddings and, as a novel approach, leverage graph embeddings for a better understanding of the DKG structure. We will provide compensation to contributors for valuable contributions such as training data and model improvements.

Project requirements

List of personas

Data Creators: Individuals or entities that provide data to the DKG.

Questioners: Users who query the DKG using natural language.

Labellers: People who label natural language-to-SPARQL examples

User stories

Data Creators: "I want to upload my knowledge assets to the DKG so they can be easily queried. I'd also like the option to fine-tune a language model on this data to enhance the quality of answers returned."

Questioners: "I want to ask the DKG questions in my natural language and receive precise, structured answers. If the answers aren't satisfactory, I might be interested in funding further model training on specific knowledge assets."

Labellers: "I am familiar with SPARQL and can describe them in natural language and vice Versa. I am willing to create examples for free, perhaps because I am a

data creator, or for compensation, perhaps because I am a domain expert offering my service for a fee."

Functional requirements (FRs)

User Interface (UI): An intuitive and user-friendly UI that facilitates:

- Natural language input for querying the DKG.
- Selection and submission of datasets for model fine-tuning.
- Viewing of query results and related information.

Natural Language Processing (NLP) Engine: A system capable of:

- Converting user-provided natural language questions into their SPARQL equivalents, and converting SPARQL results into natural language.

DKG Interaction: The ability to:

- Execute SPARQL queries on the DKG.

Model Fine-Tuning Framework:

- Mechanisms to accept and process user-specific datasets.
- Infrastructure to perform model fine-tuning on the provided datasets.

Compensation Module:

- Mechanisms to reward contributors for providing training data, model improvements, or labeling services.
- Could be in the form of cryptocurrency payments, platform credits, or other incentives.

Non-functional requirements (NFRs)

Performance:

- Response Time: The system should return results quickly, aiming for near real-time responses for most queries.
- Scalability: As the number of users and the amount of data grows, the system should be capable of handling increased load.

Accuracy & Robustness:

- Query Conversion: High accuracy in converting natural language questions into SPARQL queries.

- Result Interpretation: Reliable conversion of SPARQL results back to natural language.

Security & Privacy:

- Data Storage: data, especially user-provided training datasets, should be stored securely.
- Data Transmission: Secure and encrypted communication between different system components and the user's device.
- Access Control: Restrictive measures to ensure only authorized personnel can access sensitive functionalities like model fine-tuning. Web3 enabled wallets would be useful for user auth.

Usability:

- Accessibility: The user interface should be usable by a wide range of users, including those with disabilities.
- Multi-Device Support: The system should be usable on various devices such as desktops, tablets, and mobiles.

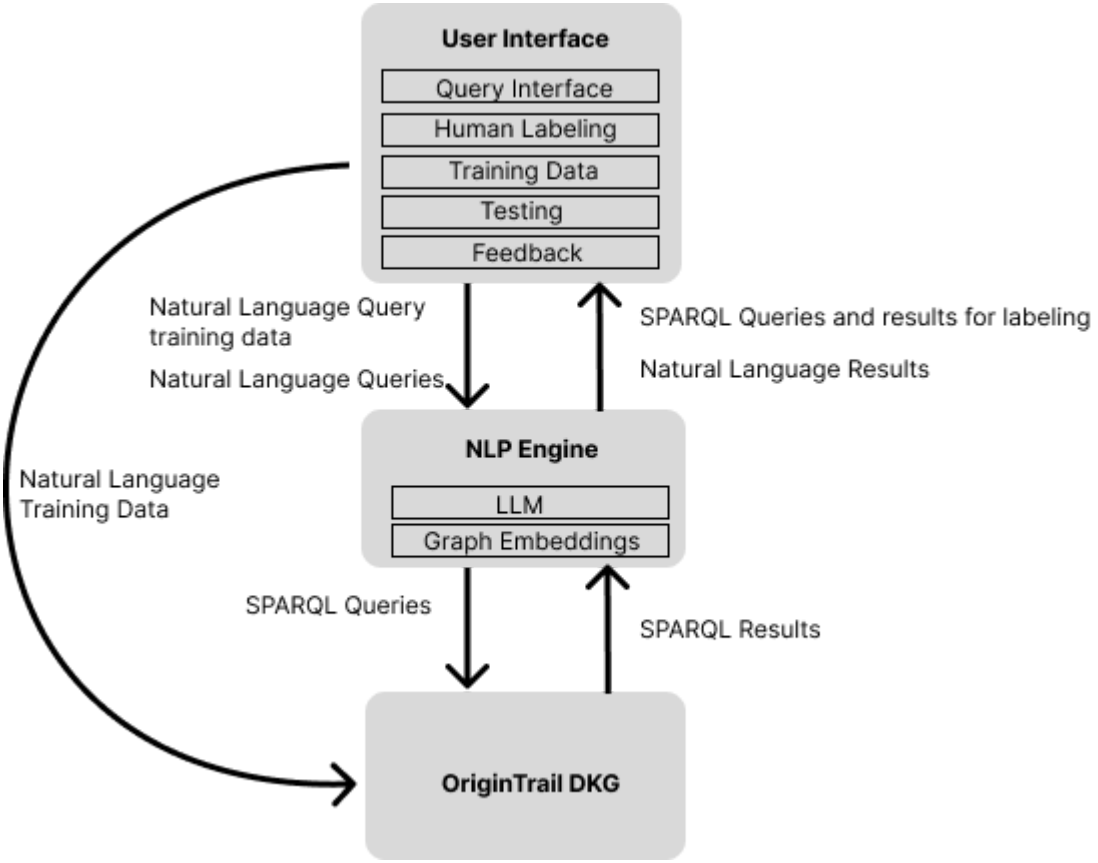
Maintainability & Extensibility:

- The system should be designed in a modular way, allowing for easy updates and the addition of new features.
- Regular updates and patches to address any issues or vulnerabilities.

Cost Efficiency:

- Keeping operational costs in check, especially when scaling the system for a larger user base.

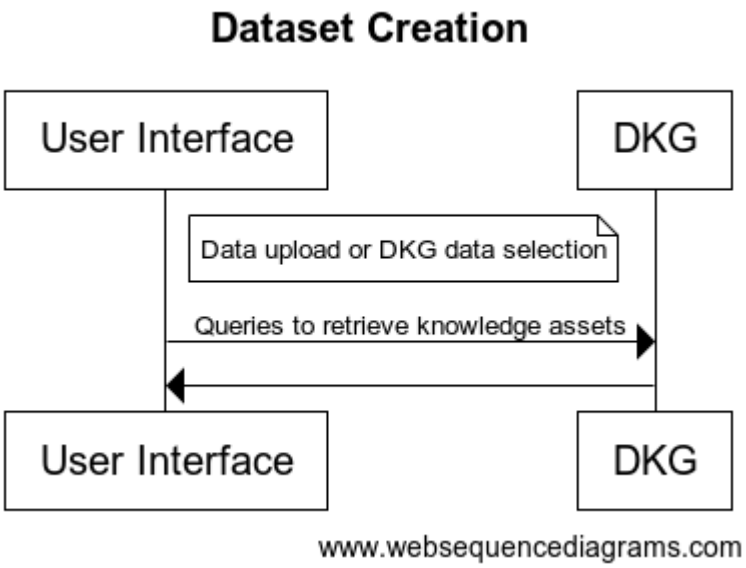
System architecture



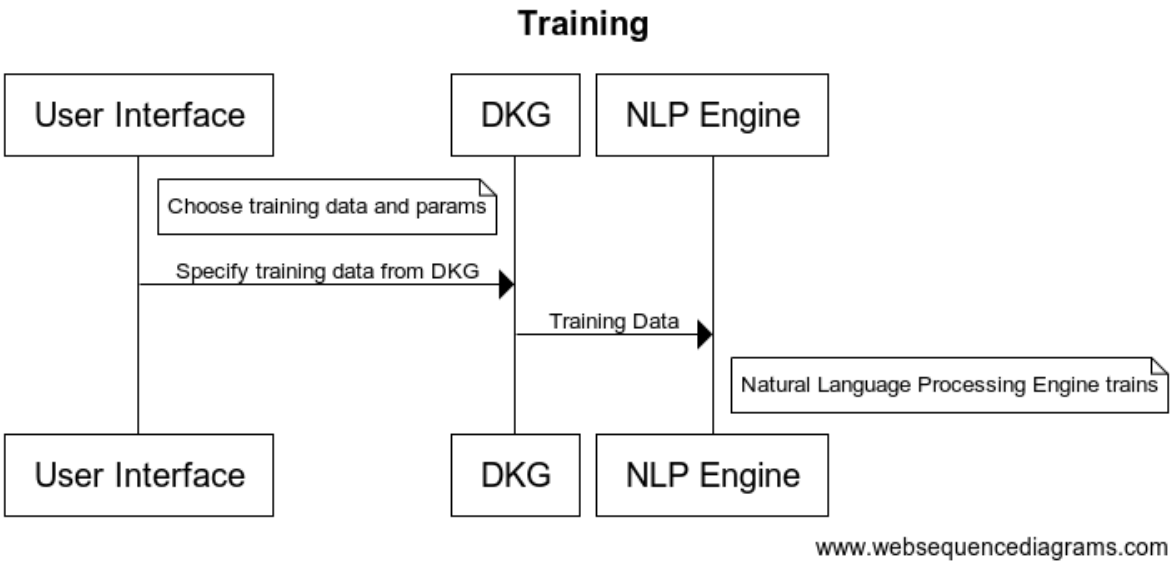
This diagram merges the training, testing, validation, and production processes. The user interface is where users can interact with the DKG in natural language. The interface can also be where users choose DKG assets to be used to train a model or generate graph embeddings. We will explore storing the training data in the DKG, but may use another data store if necessary.

The NLP Engine is where the LLM and Graph embeddings are maintained. This interfaces with the DKG to run queries.

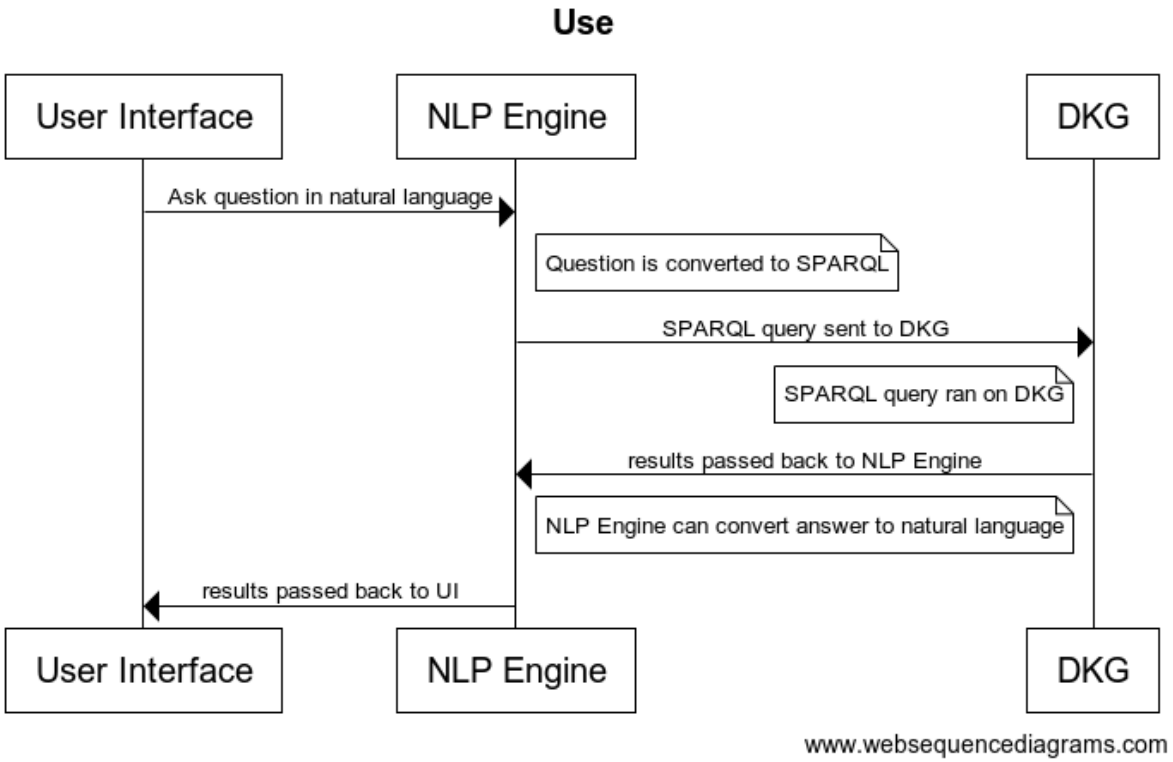
Sequence diagrams



Link



Link



Link

Knowledge modelling

- Data Type: Our proof of concept will be based on the metacrisis witha focus on regenerative finance. ReFiDAO, a regenerative finance web3 project, has provided data that we will use as a start.
- Source Format: Various - could be tabular data, text documents, geo-data, etc. Transformation to knowledge assets would depend on the data type. ReFIDAO has provided tabular data which we can export as CSVs. We will turn this data into graph data before training.
- Ontologies: A bespoke ontology tailored to the specific challenges of the metacrisis, regenerative finance, and bioregionalism, with potential for supporting others.
- Dominant Queries: We will begin with queries related to web3 regenerative finance organisations, people, events, etc. With a broader focus on environmental impact, socio-economic implications, potential solutions, and interconnected challenges.

- Data Validation: Given the importance of the data, some form of validation is likely necessary, especially for human labelled training data.
- Knowledge Ownership: The knowledge assets used to train a model could be owned by anyone. The asset owners may choose to pay for a model to be trained on their assets, the resulting fine tuned model of which could also be owned and managed by them. Any public data could be used for training.

AI capability

- A Large Language Model (LLM): fine-tuned on domain-specific data, capable of converting between natural language and SPARQL.
- Text embeddings: provide the semantic understanding needed to interpret the query
- Graph embedding: provide the structural understanding needed to map the query onto the knowledge graph

Quality assurance

- Regular model evaluations to ensure conversion accuracy.
- Continuous integration and deployment for software updates as well as dataset evolution.
- User feedback mechanisms to capture issues and improve system quality.
- Clear and comprehensive documentation of solution. Tutorials and examples could help usability.
- Cultivating a commons around the solution, where the community can take ownership of the tools from the bottom up with democratic and prosocial governance.

Technical implementation

Environments

Both testnet and mainnet deployments. Allow for local setups for developers and potentially specific DKG supported blockchains.

Versioning

Semantic versioning for the software with clear changelogs.

Deployment

Will explore easy deployment methods such as containerization (e.g., Docker).

API reference

Generating an OpenAPI spec would be useful for third-party developers. We will explore this and other API options.

Open questions and research

The feasibility and scalability of decentralizing the NLP Engine and overall solution, including architecture and stewardship of the system.

The cost implications of fine-tuning and maintaining the LLM, especially as the DKG grows.

Dataset Collection and Creation Strategy:

- Crowdsourcing: This could involve creating a platform where community members can submit natural language queries and their corresponding SPARQL queries. To incentivize participation, we could offer compensation. However, this would require careful validation to ensure the quality of the submitted data.

    - strategy to use DKG knowledge assets:
        - Data Extraction: Extract data from the DKG and use this data to create a variety of SPARQL queries. This could involve selecting a subset of the DKG, and writing a script to generate different SPARQL queries that extract various types of information from this subset.
        - Human Labeling: Then, we could pay people to label these queries by writing the corresponding natural language question or statement for each query.
        - Machine Labeling: We can explore using a LLM to label queries with corresponding natural language, and use humans to validate the labels.
        - Open Source Datasets: There may already exist datasets that map natural language queries to SPARQL queries. One notable dataset is the LC-QuAD, which includes around 30,000 SPARQL queries and their corresponding natural language questions. These datasets could potentially be used as a starting point for the model, but they might not perfectly fit the context of the OriginTrail DKG, so fine tuning on the DKG will likely be a useful step.
- We will explore if it makes sense economically to store the natural language-to-SPARQL training data on the DKG.

Strategy for learning natural language presentation of query results:

- Similar to training on queries, we can train the LLM to provide a natural language description of results returned from a query. We can explore existing models, and fine tune them with examples where DKG query results are labelled with a natural language counterpart. The labelling could be done by humans or possibly a LLM with humans in the loop to verify or tweak the results.

Potential open-source datasets and integration possibilities:

- graph embeddings There are several popular methods for learning graph embeddings, including:
    - DeepWalk and Node2Vec: These methods work by performing random walks on the graph to generate sequences of nodes, similar to sentences in a text corpus. These sequences are then fed into a Word2Vec-like model to learn embeddings for each node.

    - Graph Convolutional Networks (GCNs): These methods work by iteratively aggregating information from each node's neighbors to update its embedding.
    - GraphSAGE: This method extends GCNs by allowing nodes to sample a fixed number of neighbors at each layer of the model, which can make it more scalable for large graphs. These are just a few examples, and the potential applications could be even broader depending on the specific characteristics of the OriginTrail Decentralized Knowledge Graph. The combination of text embeddings and graph embeddings allows you to leverage both the semantic content of the data and its structural relationships, which can provide a more holistic view of the information in the graph.
- training data
    - There may already exist datasets that map natural language queries to SPARQL queries. One notable dataset is the LC-QuAD, which includes around 30,000 SPARQL queries and their corresponding natural language questions. These datasets could potentially be used as a starting point for the model, but they might not perfectly fit the context of the OriginTrail DKG, so fine tuning on the DKG will likely be a useful step.