Ideas for Knowledge Graph Related Projects

## Knowledge Graph Infrastructure

The idea would be to construct an ecosystem of tools that is model agnostic (as much as possible) for people working on models that operate on top of knowledge graphs. It would address common pain points in the entire process from beginning to end:

1. Knowledge graph creation
2. Knowledge graph management
3. Model training
4. Model deployment
5. Sharing results / Creating end user application

If the ecosystem is cohesive enough, in the future this also makes it possible for some sort of TDC 2.0 for KG-based models to be built, such that GNNs, KGs and Explainability modules can be benchmarked and compared against each other.

### Easy deployment of Models operating on KGs (e.g., GNNs, LLMs)

1. With a couple lines of code, you could incorporate the different aspects of a model to the network, to be able to server results for end-user applications.
   1. Knowledge Graph
   2. GNN / LLM
   3. Explainability Module

### Efficient storage of KGs

1. Could enable functionality like graph variants and versioning
2. Being able to store KGs at a very small size, makes it practical to generate multiple variants of Prime KG,

### Easy displaying

1. Building a library in a popular framework, such a React, to visualize different aspects of knowledge graphs, such as the model outputs, score explanations and the graph itself, could be useful for building GRAVITY, but also for other end-user applications inside and outside the lab.
   1. Graph viz can be built on top of cytoscape.js (what is currently being used in Gravity)

### Easy sharing and collaboration (GRAVITY)

1. A very simple website using the components mentioned above could be integrated into the deployment library so that results from a new model are immediately shareable through a website and a researcher’s collaborators can interact with a new model without having to download or install anything. (See more info below)
2. Displaying the information of a big knowledge graph on the web could pose technical challenges, as internet download speeds and browser memory caps prevent large files from being downloaded and queried directly in a web app. Below are some ideas on how to overcome with problem:
   1. Providing a GraphQL endpoint that the client can query
      1. Pros:
         1. Can handle any size of graph
         2. Can provide a common query language for all clients
      2. Cons:
         1. Will require a lot of work to implement
         2. Will require a lot of work to maintain
         3. Will require a lot of work optimize
   2. Shipping the graph in a compressed format to the client for almost instant access to the data
      1. Pros:
         1. Minimal latency
         2. No need to pre-think about the types of queries, or a query language, because the clients can do whatever it wants with the data
      2. Cons:
         1. Will eventually not work for graphs of a certain size
         2. Will require reimplementation of common data queries in all clients (although this could be avoided by providing a common TypeScript package that performs these)
   3. Provide HTTP endpoints with predefined functionality and allow the person creating the server to make custom endpoints that answer a specific question
      1. Pros:
         1. Can handle any size of graph
         2. Can provide a common query language for all clients
         3. Can provide a way for the server to optimize queries
         4. The custom query will already need to be implemented, so it is an easy way to bypass this
      2. Cons:
         1. Will require implementing multiple separate endpoints for common queries

## PrimeKG

### Remaking a data pipeline for PrimeKG that provides reproducible results

1. This pipeline would query the data sources, adapt it to the PrimeKG schema, deduplicate nodes and edges, and store the data in a format that can be exported to common relational graph formats (e.g., RDF, SQL) and databases (e.g., Neo4j, PostgreSQL)
2. Triggers could be set up that are manual, cron-based (e.g., once a month), or event-based (e.g., a data source was updated, the pipeline code was updated)
3. I was not aware of BioCypher, but a lot of the work has been cut out for us, which means what remains would be resting adapters for new data sources not currently available, and adding triggers to the pipeline. RDF and CSV export have already been implemented. Neo4j (most widely used graph database) and Memgraph (fast in-memory LPG database - great for integrating with model inference results and LLM queries) would be great and we could collaborate on its implementation.

### Adding a self-constructing component

This component is able to ingest medical/clinical literature and build on top of the structured knowledge graph foundation to add more information to it. This could open up the door to a lot more information at a pace that is higher than other labs / companies / traditional databases.

1. Potential text source: FDA clinical approval documents processed in collaboration with CZI?
2. Aside from growing the graph, this system could also be used to highlight discrepancies between the recent medical literature and the existing knowledge graph, marking them for human review.
3. Potential implementation:
   1. We split unstructured sources into small chunks of text that fit in a context window.
   2. We use GraphRAG (Microsoft) to obtain the relevant subgraph of the graph currently being edited.
   3. We use Graph RAG to use the subgraph and combine it with the new data to suggest additions or edits to the subgraph.
   4. These edits are then merged into the larger graph using the algorithms form older LLM KG construction methods (e.g., Cosmo).
4. Alternative implementation:
   1. Transforming the starting graph into an index graph that complies with the GraphRAG requirements.
   2. Loading text by chunks and continuing constructing the graph index.
   3. Transforming the graph index back into a collection of triples that can be exported in standard graph format.

### Questions

1. What value can we provide given this context?
2. Could we potentially write a paper saying “We are creating PrimeKG, after hitting a structured data limit, we created this system for extended learning”? We could have a continuously updated KG built on top of BioCypher that outputs new versions regularly. We could then serve the multiple versions and variants in a compressed format to be easy to integrate into new research (Good DX - Developer Experience).

## Links

1. Knowledge Graphs + LLMs
   1. [GraphRAG](https://microsoft.github.io/graphrag/)
   2. Reddit Article
      1. [Knowledge Graphs for RAG](https://www.deeplearning.ai/short-courses/knowledge-graphs-rag/)
      2. [Structured Knowledge Graph Extraction with Llama Index](https://www.llamaindex.ai/blog/introducing-the-property-graph-index-a-powerful-new-way-to-build-knowledge-graphs-with-llms)
   3. [Adding Data to an Existing Graph/Index with GraphRAG](https://github.com/microsoft/graphrag/issues/360)
   4. [Building a knowledge graph to enable precision medicine](https://www.nature.com/articles/s41597-023-01960-3)
   5. [Implementing RAG from Scratch](https://www.youtube.com/watch?v=sVcwVQRHIc8&ab_channel=freeCodeCamp.org)
2. Knowledge Graphs + Bio
   1. BioCypher
      1. [Library website](https://biocypher.org/)
      2. [Nature article](https://www.nature.com/articles/s41587-023-01848-y)