1. Introduction

This document outlines a plan to execute a series of tasks related to processing and querying a knowledge graph derived from the FlexiFusion dump. The primary objectives include creating a sample from the dump, implementing various partitioning schemes, defining SPARQL queries of different complexity levels, and measuring metrics for constructing a sub-knowledge graph to fulfill a specific query. The focus is on ensuring efficient data handling, optimal query performance, and scalability for subsequent testing and analysis. This plan details the methodologies and timelines to achieve these goals within a structured framework.

2. Task Breakdown and Execution Plan

2.1 Create a Representative Sample from the FlexiFusion Dump

The first task is loading the dataset from the FlexiFusion Dump into an Apache Spark instance running locally in Docker. The process begins with an exploration of the dump to understand its structure, size, and content distribution, including entities, predicates, and sources. Tools such as RDFlib will be used to facilitate queries against the data reproducibly. A stratified sampling strategy will be used to ensure the sample reflects the diversity of entity types, predicates, and sources, targeting a manageable size of approximately 100GB dataset, representing geopolitical data. The output will be a sampled RDF dataset in .nt format, stored locally. A potential challenge is ensuring that the subgraph contains all necessary information and does not lose key information, reducing query quality. When that phase is completed, a performance analysis will be executed. This is to quantitively evaluate the varying partitioning schemes in terms of speed, completeness and size of precessed data.

2.2 Implement Pre-Fusion Partitioning Schemes

The second task involves partitioning the FlexiFusion dump into sub-graphs based on specified criteria to enhance querying and processing efficiency. This task is divided into four sub-tasks, each addressing a different partitioning scheme. Which are to be compared later.

2.2.1 By Predicate

The predicate-based partitioning has already been completed. The focus will be on reviewing the existing partitioning code to ensure compatibility with other schemes.

2.2.2 By RDF Type

Partitioning by RDF type involves identifying all class types in the dataset using a SPARQL query to select distinct types where a resource is defined as a specific class. Triples associated with each type, such as Person, Country, or Location, will be extracted using RDFlib and stored in separate files. Depending on the actual provided dataset, the types will be adjusted accordingly.

2.2.3 By Source

Partitioning by source requires identifying source metadata, such as provenance annotations or dataset origins, within the FlexiFusion dump. Triples will be grouped by source using a script to parse provenance or source-related predicates, with results stored in separate RDF files or database tables. This task will utilize Python with RDFlib for processing, SPARQL for source identification. The output will be RDF files.

2.2.4 Combination of Type and Predicate

The combined type and predicate partitioning scheme builds on the previous approaches by creating hybrid partitions, such as all triples with a specific predicate. A nested partitioning strategy will be implemented, first grouping by type and then by predicate within each type.

2.3 Define SPARQL Queries of Varying Complexity

The third task is to create three SPARQL queries of easy, medium, and difficult complexity to extract information from the sampled knowledge graph, such as a medium one to retrieve all U.S. presidents. The easy query will retrieve all entities of a specific type, such as selecting all resources defined as Person with a limit of 100 results, to test basic type-based retrieval and dataset structure. The medium query will retrieve entities with a specific predicate and apply a filter to test predicate-based filtering. The difficult query will retrieve the latest U.S. president, assuming predicates like positionHeld and startDate exist in the dataset, using a query that selects the president with the most recent start date, ordered descending and limited to one result. The queries will be executed on the sampled dataset using a RDFlib with results validated against known data, such as the current U.S. president as of May 29, 2025. The output will be three documented SPARQL queries with expected result formats.

2.4 Measure Metrics for Creating a Sub-Knowledge Graph

The last task is to validate metrics to quantify the quality of previously generated partitioning schemes on our defined queries. The goal was to create a sub-knowledge graph that is as small as possible but contains all necessary information to answer the SPARQL query. Therefore, execution time is a relevant metric on which the sub-graphs are to be evaluated on. The execution time is measured by the time the database needs to respond to a query: From the start of the request until a response arrives. Since our database runs locally, network influences can be ruled out. This time is then to be compared to the execution time on the whole graph. Additionally, the size of the subgraph also matters. The subgraph needs to be storage efficient even though there might be no effect on query response time because disk space is a limited resource. Download time might also be a relevant metric to analyze, however, to validate it, a query is needed that returns a sizable amount of response data. It is at this point unclear

if the formulated queries return significant data so that download times become noticeable.

3. Resource Requirements

The tests and operations outlined previously, are done running locally and providing as many resources as possible without exhausting the host machine. 700GB of disk space can be allocated as well as a maximum 56GB of memory and 11 cores.

4. Timeline Summary

The implementation starts at the beginning of June and is projected to finish around mid-July so that the results can be presented on the specified date in August.

6. Conclusion

This plan provides a structured and efficient approach to execute the specified knowledge graph tasks within the timeline given by the schedule of the course. By leveraging appropriate tools, sampling strategies, and partitioning schemes, the project is expected to deliver a representative sample, optimized partitions, well-defined SPARQL queries, and detailed metrics for sub-knowledge graph creation.

Delivered by Valentina Gruber and Arne Steffensen