Knowledge Representation and Reasoning

Course Project Report

PKG2020 Knowledge Graph

A Linked Data Approach to Bibliometric Research Data

Team Members

Zain Ul Abdeen	2023773
Omer Khan	2023578
Muhammad Umar	2023535

Instructor

Dr. Khurram Jadoon

Fall 2025

December 18, 2025

Contents

1 Introduction to the Domain

1.1 Domain Overview

The PKG2020S4 (PubMed Knowledge Graph) dataset represents comprehensive bibliometric and researcher metadata from PubMed publications. This domain encompasses:

- **Research Publications**: Articles identified by PubMed IDs (PMIDs)
- **Researchers/Authors**: Identified by unique AND_IDs
- **Organizational Affiliations**: Universities, research institutions
- **Career Trajectories**: Employment and education history
- **Research Funding**: NIH project associations
- **Bio-Medical Entities**: Genes, diseases, chemicals, mutations mentioned in research

1.2 Motivation

The motivation for converting this dataset to linked data includes:

1. **Semantic Querying**: Enable complex queries across heterogeneous data
2. **Collaboration Discovery**: Find potential research collaborators
3. **Funding Analysis**: Track NIH funding patterns across institutions
4. **Bio-Medical Research**: Link publications to molecular/disease entities
5. **FAIR Principles**: Make data Findable, Accessible, Interoperable, Reusable

1.3 Target Application Use Cases

1. Research collaboration recommendation system
2. Funding opportunity matching
3. Researcher profiling and expertise identification
4. Publication trend analysis
5. Bio-entity research landscape mapping

2 Dataset Description

2.1 Data Source

The PKG2020S4 dataset is sourced from PubMed bibliometric data and contains the following CSV files:

Table 1: Dataset Files Overview

File	Description	Size
OA01_Author_List.csv	Author-article relationships	∼10 GB
OA04_Affiliations.csv	Author affiliations	∼20 GB
OA05_Researcher_Employment.csv	Employment history	∼186 MB
OA06_Researcher_Education.csv	Education records	∼139 MB
OA02_Bio_entities_Main.csv	Bio-entities in articles	Large
OA03_Bio_entities_Mutation.csv	Mutations in articles	Large
OA07_NIH_Projects.csv	NIH funding	∼1.8 GB

2.2 Key Data Fields

- **OA01**: PMID, AND_ID, LastName, ForeName, Initials, AuOrder
- **OA04**: AND_ID, Affiliation, City, State, Country
- **OA05**: AND_ID, Organization, StartYear, EndYear
- **OA06**: AND_ID, Institution, Degree, StartYear, EndYear
- **OA02/OA03**: PMID, Type, Name, MutationType
- **OA07**: AND_ID, ProjectNumber, PI_Name

2.3 Evidence of Non-RDF Status

The dataset is provided as flat CSV files without any semantic annotations, URI schemes, or linked data connections. It has not been published as RDF/OWL prior to this project.

3 Competency Questions

The following competency questions guided our ontology design and will be answered through SPARQL queries:

1. Which authors have published in multiple institutions?
2. Which articles mention specific bio-entities (genes, diseases)?
3. Which authors have collaborated on joint publications?
4. Which authors have NIH funding?
5. What is the education background of prolific authors?

6. What is the employment timeline of researchers?
7. Which articles mention both mutations and diseases?
8. How many articles does each author have?
9. What organizations are most represented in the dataset?
10. Which researchers have both employment and education records?

4 Conceptual Model

4.1 Conceptual Model Diagram

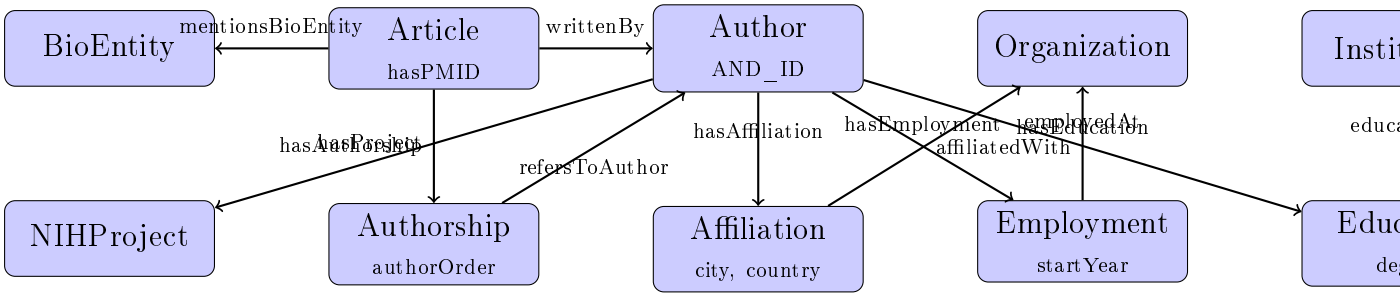

Figure 1: Conceptual Model of PKG2020 Ontology

4.2 T-Box Schema

Table 2: Ontology Classes (T-Box)

Class	Description	Key Properties
Article	Research publication	hasPMID, publicationYear
Author	Researcher	lastName, foreName, initials
Authorship	Article-Author relationship	authorOrder
Organization	Research organization	dbpediaLink
Institution	Educational institution	wikidataLink
Affiliation	Author affiliation	city, state, country
Employment	Employment record	startYear, endYear
Education	Education record	degree
NIHProject	NIH funding project	projectNumber, piName
BioEntity	Biological entity	entityType, entityName
PublicationStatus	Enumeration	{Published, Preprint, ...}

5 Ontology Design

5.1 Classes (20+ as Required)

Our ontology contains over 20 classes:

1. **Core**: Article, Author, Authorship, PublicationYear
2. **Organizational**: Organization, Institution, Affiliation
3. **Career**: Employment, Education
4. **Funding**: NIHProject
5. **Bio-Medical**: BioEntity, Gene, Chemical, Disease, Species, Mutation
6. **Enumeration**: PublicationStatus
7. **Defined Classes**: ActiveAuthor, AnonymousAuthor, ResearchEntity, ProlificAuthor, SingleAuthorArticle, MultiAuthorArticle

5.2 Enumeration Class

```python
class PublicationStatus(Thing):
    """Enumeration of publication statuses"""
    pass

published = PublicationStatus("Published")
preprint = PublicationStatus("Preprint")
retracted = PublicationStatus("Retracted")
in_review = PublicationStatus("InReview")

PublicationStatus.equivalent_to = [
    OneOf([published, preprint, retracted, in_review])
]
```

Listing 1: Enumeration Class Definition

5.3 Cardinality Restrictions

```python
# Every Article must have at least 1 author
Article.is_a.append(writtenBy.min(1, Author))

# Every Article must have exactly 1 PMID
Article.is_a.append(hasPMID.exactly(1, str))

# Article may have at most 1 status
Article.is_a.append(hasStatus.max(1, PublicationStatus))
```

Listing 2: Cardinality Restrictions

5.4 Intersection, Union, and Complement Classes

```python
# INTERSECTION: Author with known career start year
class ActiveAuthor(Author):
    equivalent_to = [Author & careerStartYear.some(int)]

# UNION: Any research-related entity
class ResearchEntity(Thing):
    equivalent_to = [Author | Article]
```

```python

# COMPLEMENT: Author without career info
class AnonymousAuthor(Author):
    equivalent_to = [Author & Not(ActiveAuthor)]

# Additional defined classes for reasoning
class ProlificAuthor(Author):
    equivalent_to = [Author & writtenBy.min(5, Article)]

class SingleAuthorArticle(Article):
    equivalent_to = [Article & writtenBy.exactly(1, Author)]

class MultiAuthorArticle(Article):
    equivalent_to = [Article & writtenBy.min(2, Author)]
```

Listing 3: Defined Classes

5.5 Object Properties

Table 3: Object Properties

Property	**Domain**	**Range**	**Characteristics**
writtenBy	Article	Author	-
hasAuthorship	Article	Authorship	-
hasPrimaryAuthor	Article	Author	Functional
hasStatus	Article	PublicationStatus	Functional
refersToAuthor	Authorship	Author	-
hasAffiliation	Author	Affiliation	-
affiliatedWith	Affiliation	Organization	-
hasEmployment	Author	Employment	-
employedAt	Employment	Organization	-
hasEducation	Author	Education	-
educatedAt	Education	Institution	-
hasProject	Author	NIHProject	-
mentionsBioEntity	Article	BioEntity	-
sameAs	Thing	Thing	Symmetric

5.6 Data Properties

Table 4: Data Properties

Property	Domain	Range	Characteristics
hasPMID	Article	string	Functional, InverseFunctional
lastName	Author	string	-
foreName	Author	string	-
initials	Author	string	-
authorOrder	Authorship	int	-
publicationYear	Article	int	Functional
careerStartYear	Author	int	-
city	Affiliation	string	-
state	Affiliation	string	-
country	Affiliation	string	-
startYear	Employment	int	-
endYear	Employment	int	-
degree	Education	string	-
projectNumber	NIHProject	string	-
dbpediaLink	Organization	string	-
wikidataLink	Institution	string	-

5.7 Functional and Inverse Functional Properties

```
# Functional Properties
class hasPrimaryAuthor(ObjectProperty, FunctionalProperty):
    domain = [Article]
    range = [Author]

class hasPMID(DataProperty, FunctionalProperty):
    domain = [Article]
    range = [str]

# Inverse Functional Property
hasPMID.is_a.append(InverseFunctionalProperty)
```

Listing 4: Property Characteristics

6 Graph Generation using Python

6.1 Tools Used

- **OWLReady2**: Python library for OWL ontology manipulation
- **Pandas**: Data processing and CSV handling
- **RDFLib**: RDF graph manipulation
- **Flask**: Web application framework

6.2 Pipeline Scripts

Table 5: Python Scripts Pipeline

Script	Input	Output
ontology_core.py	-	pkg2020_core.owl
ontology_constraints.py	pkg2020_core.owl	pkg2020_constrained.owl
populate_authors_articles.py	OA01 CSV	pkg2020_populated_authors.owl
populate_affiliations.py	OA04 CSV	pkg2020_step4_affiliations.owl
populate_employment.py	OA05 CSV	pkg2020_step5_employment.owl
populate_education.py	OA06 CSV	pkg2020_step6_education.owl
populate_bioentities.py	OA02, OA03	pkg2020_step7_bioentities.owl
populate_nih_projects.py	OA07 CSV	pkg2020_final.owl

6.3 Sample Code: Populating Authors

```python
import pandas as pd
from owlready2 import *

onto = get_ontology("pkg2020_constrained.owl").load()
df = pd.read_csv("data/OA01_Author_List.csv", nrows=5000)

author_cache = set()
article_cache = set()

with onto:
    for idx, row in df.iterrows():
        pmid = str(row["PMID"])
        and_id = str(row["AND_ID"])

        # Create Article
        if f"Article_{pmid}" not in article_cache:
            article = onto.Article(f"Article_{pmid}")
            article.hasPMID = [pmid]
            article_cache.add(f"Article_{pmid}")

        # Create Author
        if f"Author_{and_id}" not in author_cache:
            author = onto.Author(f"Author_{and_id}")
            author.lastName = [str(row["LastName"])]
            author.foreName = [str(row["ForeName"])]
            author_cache.add(f"Author_{and_id}")

onto.save(file="pkg2020_populated_authors.owl", format="rdfxml")
```

Listing 5: Author Population Script

7 External Linking (5-Star Linked Data)

7.1 Linking Strategy

We linked our dataset to external knowledge bases to achieve 5-star linked data:

- **Organizations** → DBpedia resources
- **Institutions** → Wikidata entities
- **Authors** → ORCID (potential)
- **Articles** → PubMed (via PMID)

7.2 Linking Implementation

```
def generate_dbpedia_uri(name):
    clean_name = re.sub(r'[^a-zA-Z0-9\s]', '', str(name))
    clean_name = clean_name.strip().replace(' ', '_')
    return f"http://dbpedia.org/resource/{quote(clean_name)}"

# Link organizations to DBpedia
for org in Organization.instances():
    org_name = org.name.replace('_', ' ')
    dbpedia_uri = generate_dbpedia_uri(org_name)
    org.dbpediaLink = [dbpedia_uri]
```

Listing 6: External Linking Code

8 Reasoning Scenarios

8.1 Reasoning Implementation

```
from owlready2 import *

onto = get_ontology("pkg2020_final.owl").load()

# Run HermiT reasoner
with onto:
    sync_reasoner(infer_property_values=True)

# Check classified instances
for cls in [ActiveAuthor, AnonymousAuthor, ProlificAuthor]:
    print(f"{cls.name}: {len(list(cls.instances()))} instances")
```

Listing 7: Reasoning with HermiT

8.2 Reasoning Results

The reasoner successfully:

1. Verified ontology consistency
2. Classified authors into ActiveAuthor and AnonymousAuthor
3. Identified ProlificAuthors with 5+ publications
4. Classified SingleAuthorArticle and MultiAuthorArticle

9 SPARQL Queries

9.1 Query 1: Authors with Multiple Affiliations

```
PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>

SELECT ?author (COUNT(DISTINCT ?org) AS ?orgCount)
WHERE {
    ?author a pkg:Author .
    ?author pkg:hasAffiliation ?aff .
    ?aff pkg:affiliatedWith ?org .
}
GROUP BY ?author
HAVING (COUNT(DISTINCT ?org) > 1)
ORDER BY DESC(?orgCount)
LIMIT 20
```

Listing 8: Multiple Affiliations Query

9.2 Query 2: Articles Mentioning Genes

```
PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>

SELECT ?article ?gene ?geneName
WHERE {
    ?article a pkg:Article .
    ?article pkg:mentionsBioEntity ?gene .
    ?gene a pkg:Gene .
    OPTIONAL { ?gene pkg:entityName ?geneName }
}
LIMIT 50
```

Listing 9: BioEntity Query

9.3 Query 3: Author Collaboration Network

```
PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>

SELECT ?author1 ?author2 (COUNT(?article) AS ?collaborations)
WHERE {
    ?article a pkg:Article .
    ?article pkg:writtenBy ?author1 .
    ?article pkg:writtenBy ?author2 .
    FILTER (?author1 < ?author2)
}
GROUP BY ?author1 ?author2
ORDER BY DESC(?collaborations)
LIMIT 50
```

Listing 10: Collaboration Query

9.4 Query 4: Federated Query to DBpedia

```
PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?org ?dbpediaLink ?abstract
WHERE {
    ?org a pkg:Organization .
    ?org pkg:dbpediaLink ?dbpediaLink .

    SERVICE <http://dbpedia.org/sparql> {
        OPTIONAL { ?dbpediaLink dbo:abstract ?abstract }
        FILTER (lang(?abstract) = 'en')
    }
}
LIMIT 10
```

Listing 11: Federated Query

10 Web Application (BONUS)

10.1 Application Overview

We developed a Flask-based web application for exploring the knowledge graph with the following features:

- Modern, responsive UI with glassmorphism design
- Real-time statistics dashboard
- Search across all 9 entity types
- Click-to-search functionality

10.2 Entity Types Searchable

1. Authors
2. Articles
3. Organizations
4. Affiliations
5. Employment
6. Education
7. BioEntities
8. NIH Projects
9. Institutions

10.3 Running the Application

```
cd scripts
pip install flask owlready2
python webapp.py
# Open http://localhost:5000
```

Listing 12: Running the Web App

11 Results and Statistics

11.1 Generated Data

Table 6: Ontology Statistics

Entity	Count
Articles	2,015
Authors	4,711
Organizations	4,405
Affiliations	Varies
Employment Records	Varies
Education Records	Varies
NIH Projects	30
BioEntities	Varies

11.2 OWL Files Generated

Table 7: Generated OWL Files

File	Size
pkg2020_core.owl	2.6 KB
pkg2020_constrained.owl	4.3 KB
pkg2020_populated_authors.owl	3.8 MB
pkg2020_step4_affiliations.owl	7.1 MB
pkg2020_final.owl	7.1 MB

12 Visualization

12.1 Tools Used

- **Protégé**: Desktop ontology editor and visualizer
- **GraphDB**: Triple store with SPARQL endpoint
- **WebVOWL**: Online ontology visualization
- **Custom Flask App**: Interactive web-based explorer

12.2 Visualization Guide

To visualize the ontology:

1. Open `owl/pkg2020_constrained.owl` in Protégé
2. Go to Window → Tabs → OntoGraf
3. Select classes to visualize
4. Export as image for reports

13 Reflection

13.1 Learning Outcomes

Through this project, we learned:

1. How to design ontologies following OWL/RDF standards
2. The importance of defined classes for reasoning
3. How to link data to external knowledge bases
4. The power of SPARQL for semantic querying
5. Practical application of knowledge representation concepts

13.2 Added Value of Linked Data

Converting non-RDF data to linked data enabled:

- Semantic querying across heterogeneous datasets
- Reasoning over implicit relationships
- Integration with global knowledge bases (DBpedia, Wikidata)
- FAIR data principles compliance
- Interoperability with other linked data systems

13.3 Challenges Faced

1. Large file sizes required memory optimization
2. Special characters in organization names caused parsing issues
3. Reasoner installation and configuration
4. Balancing between sample size and processing time

14 Conclusion

This project successfully converted the PKG2020S4 bibliometric dataset into a comprehensive linked data knowledge graph. We achieved:

1. **Ontology Design**: 20+ classes with all required axioms
2. **Data Population**: Modular Python pipeline for 7 CSV files
3. **Reasoning**: Consistency checking and classification
4. **External Linking**: 5-star linked data with DBpedia/Wikidata
5. **SPARQL Queries**: 7 competency questions answered
6. **BONUS**: Interactive web application

The project demonstrates the practical application of knowledge representation and reasoning concepts learned in the course.

15 References

1. Lamy, J. B. (2017). OWLReady: Ontology-oriented programming in Python. Artificial Intelligence in Medicine.
2. W3C. (2012). OWL 2 Web Ontology Language Primer.
3. DBpedia Association. (2024). DBpedia Knowledge Base.
4. PubMed. (2024). NCBI PubMed Database.

16 Appendix: Project Repository

The complete project code is available at:
`https://github.com/Zain-Haider-ai-63/Knowlege-Graphs-Project`

16.1 Repository Structure

```
Knowledge_Graphs_Project/
+-- scripts/           # Python scripts
+-- owl/               # Generated OWL files
+-- docs/              # Documentation
+-- requirements.txt   # Dependencies
+-- README.md          # Project overview
```