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 24, 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
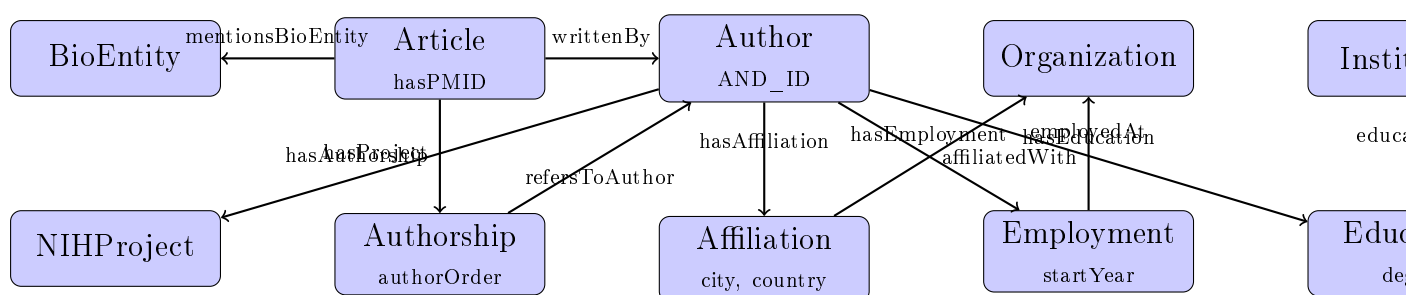
Figure 1: Conceptual Model of PKG2020 Ontology

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

```
1  class PublicationStatus(Thing):
2      """Enumeration of publication statuses"""
3      pass
4
5  published = PublicationStatus("Published")
6  preprint = PublicationStatus("Preprint")
7  retracted = PublicationStatus("Retracted")
8  in_review = PublicationStatus("InReview")
9
10 PublicationStatus.equivalent_to = [
11     OneOf([published, preprint, retracted, in_review])
12 ]
```

Listing 1: Enumeration Class Definition

5.3 Cardinality Restrictions

```
1  # Every Article must have at least 1 author
2  Article.is_a.append(writtenBy.min(1, Author))
3
4  # Every Article must have exactly 1 PMID
5  Article.is_a.append(hasPMID.exactly(1, str))
6
7  # Article may have at most 1 status
8  Article.is_a.append(hasStatus.max(1, PublicationStatus))
```

Listing 2: Cardinality Restrictions

5.4 Intersection, Union, and Complement Classes

```
1  # INTERSECTION: Author with known career start year
2  class ActiveAuthor(Author):
3      equivalent_to = [Author & careerStartYear.some(int)]
4
5  # UNION: Any research-related entity
6  class ResearchEntity(Thing):
7      equivalent_to = [Author | Article]
```

```
8
9  # COMPLEMENT: Author without career info
10 class AnonymousAuthor(Author):
11     equivalent_to = [Author & Not(ActiveAuthor)]
12
13 # Additional defined classes for reasoning
14 class ProlificAuthor(Author):
15     equivalent_to = [Author & writtenBy.min(5, Article)]
16
17 class SingleAuthorArticle(Article):
18     equivalent_to = [Article & writtenBy.exactly(1, Author)]
19
20 class MultiAuthorArticle(Article):
21     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

```python
# 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:

9

- **Organizations** → DBpedia resources

- **Institutions** → Wikidata entities

- **Authors** → ORCID (potential)

- **Articles** → PubMed (via PMID)

7.2 Linking Implementation

```python
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

```python
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

8.3 SWRL Rules (Bonus)

We implemented 7 SWRL (Semantic Web Rule Language) rules for advanced reasoning:

```
# Rule 2: Funded Author Inference
Author(?a) ^ hasProject(?a, ?p) ^ NIHProject(?p)
    -> FundedAuthor(?a)

# Rule 3: Established Researcher
Author(?a) ^ hasEmployment(?a, ?e) ^ hasEducation(?a, ?d)
    -> EstablishedResearcher(?a)

# Rule 4: Collaborative Article
Article(?art) ^ writtenBy(?art, ?a1) ^ writtenBy(?art, ?a2)
    ^ differentFrom(?a1, ?a2) -> CollaborativeArticle(?art)

# Rule 5: Gene-Disease Link Article
Article(?art) ^ mentionsBioEntity(?art, ?g) ^ Gene(?g)
    ^ mentionsBioEntity(?art, ?d) ^ Disease(?d)
    -> GeneDiseaseLinkArticle(?art)

# Rule 7: Alumni Peer Connection
Author(?a1) ^ Author(?a2) ^ hasEducation(?a1, ?e1)
    ^ hasEducation(?a2, ?e2) ^ educatedAt(?e1, ?inst)
    ^ educatedAt(?e2, ?inst) ^ differentFrom(?a1, ?a2)
    -> isAlumniPeerOf(?a1, ?a2)
```

Listing 8: SWRL Rule Examples

These rules are saved in `owl/pkg2020_with_swrl.owl` and can be executed using Protege's SWRL Tab plugin.

9 Hand-Annotated Individuals

As per the rubric requirement, we created 10+ hand-annotated individuals in a separate file (`pkg2020_hand_annotated.owl`) to test our defined classes and reasoning:

Table 6: Hand-Annotated Individuals

#	Individual	Purpose
1	Article_HAND_001	SingleAuthorArticle test
2	Article_HAND_002	MultiAuthorArticle test
3	Author_HAND_001	ActiveAuthor (has careerStartYear)
4	Author_HAND_002	AnonymousAuthor (no career info)
5	Author_HAND_003	ActiveAuthor for collaboration
6	Org_HAND_Harvard	Organization with DBpedia link
7	Inst_HAND_MIT	Institution with Wikidata link
8	Aff_HAND_001	Affiliation with location
9	Gene_HAND_BRCA1	Gene bioentity
10	Disease_HAND_Cancer	Disease bioentity

9.1 Reasoning Verification

After running the HermiT reasoner on the hand-annotated individuals:

- **ActiveAuthor**: Author_HAND_001 and Author_HAND_003 classified (have careerStartYear)

- **AnonymousAuthor**: Author_HAND_002 classified (no careerStartYear)

- **SingleAuthorArticle**: Article_HAND_001 classified (1 author)

- **MultiAuthorArticle**: Article_HAND_002 classified (3 authors)

- **FundedAuthor**: Author_HAND_001 classified (has NIH project)

10   SPARQL Queries

10.1   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 9: Multiple Affiliations Query

10.2   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 10: BioEntity Query

10.3   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 .
```

```
 7      ?article pkg:writtenBy ?author2 .
 8      FILTER (?author1 < ?author2)
 9  }
10  GROUP BY ?author1 ?author2
11  ORDER BY DESC(?collaborations)
12  LIMIT 50
```

Listing 11: Collaboration Query

10.4 Query 4: Federated Query to DBpedia

```
 1  PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>
 2  PREFIX dbo: <http://dbpedia.org/ontology/>
 3
 4  SELECT ?org ?dbpediaLink ?abstract
 5  WHERE {
 6      ?org a pkg:Organization .
 7      ?org pkg:dbpediaLink ?dbpediaLink .
 8
 9      SERVICE <http://dbpedia.org/sparql> {
10          OPTIONAL { ?dbpediaLink dbo:abstract ?abstract }
11          FILTER (lang(?abstract) = 'en')
12      }
13  }
14  LIMIT 10
```

Listing 12: Federated Query

11 Web Application (BONUS)

11.1 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

11.2 Entity Types Searchable

1. Authors

2. Articles

3. Organizations

4. Affiliations

5. Employment

6. Education

7. BioEntities

8. NIH Projects

9. Institutions

11.3 Running the Application

```
1 cd scripts
2 pip install flask owlready2
3 python webapp.py
4 # Open http://localhost:5000
```
Listing 13: Running the Web App

12 Results and Statistics

12.1 Generated Data

Table 7: 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

12.2 OWL Files Generated

Table 8: Generated OWL Files

File	Size	Description
pkg2020_tbox_only.owl	14 KB	T-Box only (no individuals)
pkg2020_hand_annotated.owl	19 KB	10+ hand-annotated individuals
pkg2020_with_swrl.owl	27 KB	T-Box with SWRL rules
pkg2020_core.owl	4.2 KB	Core ontology
pkg2020_constrained.owl	8.4 KB	With OWL axioms
pkg2020_populated_authors.owl	35 MB	Authors and articles
pkg2020_step4_affiliations.owl	69 MB	With affiliations
pkg2020_final.owl	119 MB	Complete populated ontology
pkg2020_final.ttl	291 MB	Turtle format for GraphDB

12.3 Ontology Files Summary

As per the rubric requirement, we submit:

1. **T-Box Only**: `pkg2020_tbox_only.owl` - Contains all classes, properties, and axioms without any individuals

2. **With Individuals**: `pkg2020_hand_annotated.owl` - Contains 10+ hand-annotated individuals for testing

3. **Full Population**: `pkg2020_final.owl` - Complete dataset populated via Python scripts

13 Visualization

13.1 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

13.2 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

13.3 GraphDB Setup and SPARQL Queries

To run SPARQL queries via GraphDB:

1. Download GraphDB Free from `https://www.ontotext.com/products/graphdb/download/`

2. Create a new repository (e.g., "pkg2020")

3. Import `owl/pkg2020_sample.ttl` or `pkg2020_final.ttl`

4. Navigate to SPARQL → Query

5. Run the competency queries from Section 9

Alternatively, use GraphDB Sandbox (cloud-hosted):

1. Go to `https://graphdb.ontotext.com/`

2. Create a free account and new repository

3. Upload the TTL file via Import → RDF

4. Access the SPARQL endpoint

13.4 Sample GraphDB Query Session

```
1  # Count all entities by type
2  PREFIX pkg: <http://example.org/pkg2020/ontology.owl#>
3
4  SELECT ?type (COUNT(?entity) AS ?count)
5  WHERE {
6      ?entity a ?type .
7      FILTER(STRSTARTS(STR(?type),
8              "http://example.org/pkg2020/"))
9  }
10 GROUP BY ?type
11 ORDER BY DESC(?count)
```

Listing 14: GraphDB Query Example

14 Reflection

14.1 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

14.2 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

16

14.3 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

15 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.

16 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.

17 Appendix: Project Repository

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

17.1 Repository Structure

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