Chapter 17
Building Knowledge Graphs in the Biomedical Domain: Methods and Case Studies

Shahid Azim and Hazra Imran

Abstract This chapter provides a comprehensive guide to constructing knowledge graphs in the biomedical domain, covering data acquisition, knowledge representation, ontology development with a practical case study. It highlights the applications of biomedical KGs and discusses challenges and future directions for this field. By the end of the chapter, readers will have a solid understanding of how knowledge graphs can transform biomedical research and contribute to improved patient outcomes.

Keywords Knowledge graph construction · Named Entity Recognition (NER) · Relation extraction · Knowledge representation · Ontology development

Abbreviations

KG	Knowledge graph
DIS	Data integration system
OBDA	Ontology-based data access
OBDI	Ontology-based data integration
MDPI	Multidisciplinary digital publishing institute
PRISMA	Preferred reporting items for systematic reviews and meta-analyses
GSCs	Gold standard corpora
NLP	Natural language processing
NELL	Never ending language learning
NER	Named entity recognition
PMID	PubMed ID
RE	Relation extraction

S. Azim
School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India
e-mail: shahidazim.jmi@gmail.com

H. Imran (✉)
Northeastern University, Vancouver, Canada
e-mail: h.imran@northeastern.edu

A. Sharan et al. (eds.), *Text Mining Approaches for Biomedical Data*, Transactions on Computer Systems and Networks, https://doi.org/10.1007/978-981-97-3962-2_17

17.1 Introduction

Biomedical data's increasing volume and complexity have made it challenging to extract meaningful insights and knowledge from it. Knowledge Graphs (KGs) have emerged as a powerful tool for representing and integrating data from various sources, enabling advanced data analysis and discovery. KGs can transform how we approach biomedical research and improve patient outcomes. This chapter provides an overview of the methods for constructing KGs in the biomedical domain, including data acquisition, knowledge representation, and ontology development. We will also present a case study on building a KG in the context of biomedical documents.

This chapter aims to provide a practical guide for constructing KGs in the biomedical domain and highlight their potential to accelerate biomedical research and improve patient outcomes.

17.2 Data Sources for the Construction of Biomedical Knowledge Graphs

The data available worldwide combines structured, unstructured, and semi-structured data. When it comes to KGs, unstructured data is of no use, structured data in systematic order, like a tabular format, is converted into KGs using a semantic integration process (Futia 2020). Since there is a huge stack of data available for the construction of a KG, the Data Integration System (DIS) comes into use as it integrates all the available resources on a single platform for easy access (Chaves-Fraga 2021; Lenzerini 2002).

Ontology-Based Data Access (OBDA) and Ontology-Based Data Integration (OBDI) are a type of DIS where the central dogma of knowledge revolves around Ontologies of the concept (Poggi et al. 2008).

Another strategy for data collection is the subtle use of keywords specific to the area of interest, in which the KGs ought to be constructed. By referring to the collected articles and eliminating the irrelevant ones, the authors in Abu-Salih et al. (2022) narrowed their search to 101 articles from an initial record of 643. They used the keywords like 'biomedicine', 'healthcare', 'medicine', 'drug', 'drug-discovery', 'knowledge graph construction', etc. to retrieve relevant articles from renowned journals and sources like Elsevier, MDPI (Multidisciplinary Digital Publishing Institute), Google Scholar, and ACM Digital Library. Furthermore, they also considered the citations mentioned in the retrieved paper for further assistance in their designed work-frame. This is an enhanced way to collect relevant data and delimitate the unneeded ones.

Figure 17.1 defines the strategy for selecting articles through the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) approach as data sources for developing KGs. Existing knowledge graphs can also serve as data sources (Ryen et al. 2022) by procuring a sub-graph out of it. This is achieved

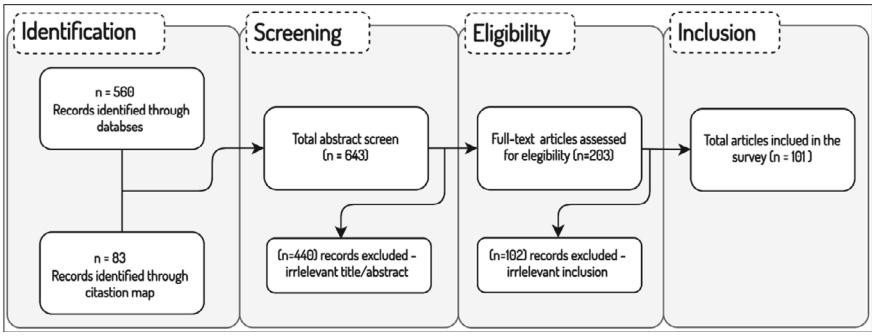

Fig. 17.1 Selecting articles through PRISMA (Abu-Salih et al. 2022)

by SPARQL CONSTRUCT, in which a CONSTRUCT query is used as a feature to get an RDF graph.

17.3 Construction of Knowledge Graphs

Constructing a KG is tricky as it involves several steps, each requiring experts' potential. The basics to begin a KG construction involves searching through databases. Databases like NCBI's Entrez, UniProt, and OMIM host information in various domains of biomedical research, which contain manually curated and annotated data with careful automation as well; the types of data might include species, genes, proteins, their sequences, taxonomy, inter-relation, etc. (Nicholson and Greene 2020). Some notable biomedical databases available online are mentioned in Table 17.1.

After construction of KG, knowledge extraction from different databases is the next big step. It could be achieved by three different approaches:

i. **Rule-Based Extraction**: Text-mining, has roles to play at this step. Extracting and establishing a relation between important data components requires a specialized knowledge so that salient keywords can be pinpointed and relationships set in. For this job, ontologies play a significant role by bringing up related keywords, while the identification of the structure of a sentence is done using various 'parse trees' (Nicholson and Greene 2020). Parse trees are derivation trees (GeeksforGeeks 2024) that serve two important purposes:

   a. Parsing: to thoroughly review a text, sentence, or string of words.
   b. Derivation of a tree: that represents an ordered or hierarchical arrangement of the parsed information.

A parse tree has a root from which branches are derived in the form of leaves which usually makes up the terminal nodes. The structure of a parse tree could be given as shown in Fig. 17.2.

Table 17.1 Online biomedical databases

Database	Hosted by	Description
Entrez (2024)	NCBI	NCBI's data retrieval tool; provides gene, protein structural and interaction data
OMIM (2024)	Johns Hopkins University	Online Mendelian inheritance in man; human genes and disorders
UniProt (The UniProt Consortium 2021)	European Bioinformatics Institute (EBI), Protein Information Resource (PIR), Swiss Institute of Bioinformatics (SIB) [44]	Protein-sequence and functions
PDB (Protein Data Bank 2024)	RCSB (Bank, n.d.)	Protein Data Bank; 3D protein structures
COSMIC (2024)	Sanger Institute	Catalogue of somatic mutations in cancer; extensive information about human cancers
CTD (Grondin et al. 2021)	NC State University	Comparative toxicogenomics database; environment–disease–human relations

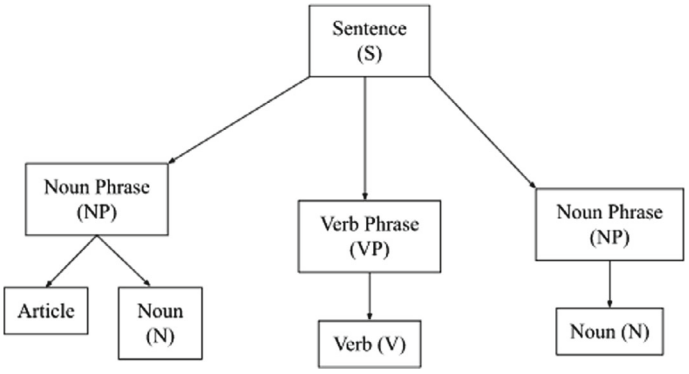

Fig. 17.2 Structure of a parse tree (CS 2112/ENGRD 2112 Fall 2018)

Suppose the input example is *Mycobacterium tuberculosis*, a causative agent of Tuberculosis. The outputs based on parse tree is shown in Fig. 17.3.

An example of a parse tree could be a 'Dependency Parse Tree', where the dependency of the tree relies upon the grammar the sentence holds. Another example, a 'Constituency Parse Tree', breaks a sentence into its constituent elements to evaluate connections (Nicholson and Greene 2020).

ii. **Supervised Relationship Extraction**: This unique approach separates affirmative parts of a sentence from the counter ones that are extracted from a previously

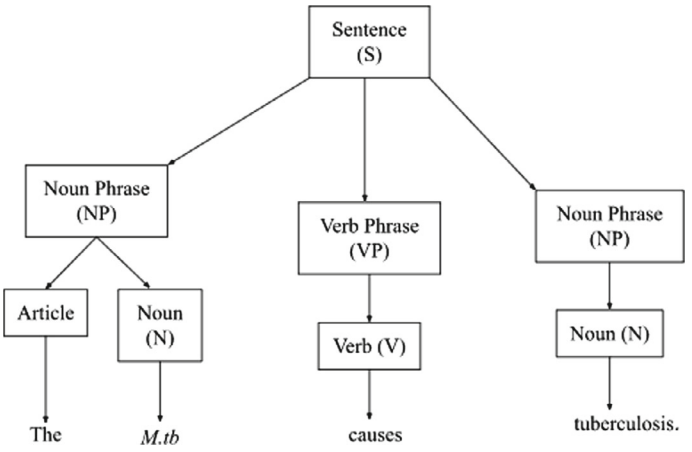

Fig. 17.3 Parse tree with rule-based relationship extraction

laid set of curated data. The sentences supporting a particular objective are segregated from those that deviate from the same cause (Nicholson and Greene 2020). An example that uses a supervised relationship algorithm is the one stated in the paper of Bhasuran and Natarajan (2018), in which their goal revolved around various genes responsible for causing versatile diseases. They achieved it by thoroughly checking their model through the Gold Standard Corpora (GSCs), which are databases needed to monitor the projects that make use of NLPs (Natural Language Processing) (Mitrofan et al. 2018). The steps that follow a supervised relationship extraction process are presented in Fig. 17.4.

iii. **Extraction Without Labels**: This approach is called 'Unsupervised' extraction as it does not depend on the subject's particularity. Instead, it handles gigantic amounts of data simultaneously independent of linear specificities (Mintz et al. 2009). This approach attracts imprecise information as it does not involve proper curation via experts, but this method contains many differing patterns containing similar data.

Furthermore, to extract data, DeepDive is a software that could extract structured data and helps build a knowledge base, which would enhance the quality of knowledge graphs by picking relationships out of it (Hao et al. 2021). DeepDive supports many databases like PubMed and is compatible with Natural Language Processing.

To express the acquired data in the form of *Triples* in knowledge graphs, a software named Neo4j is used. MySQL is another software which initially recognizes relationships between entities and probable ontologies. The Neo4j comes later, which is helpful in directly displaying the entered queries in the form of graphs.

However, creating a knowledge graph does not come with a pre-defined methodology. Although manual curation is still thought to be best, nowadays, with the busiest schedule of all time, technology has taken over manual curation. As an example,

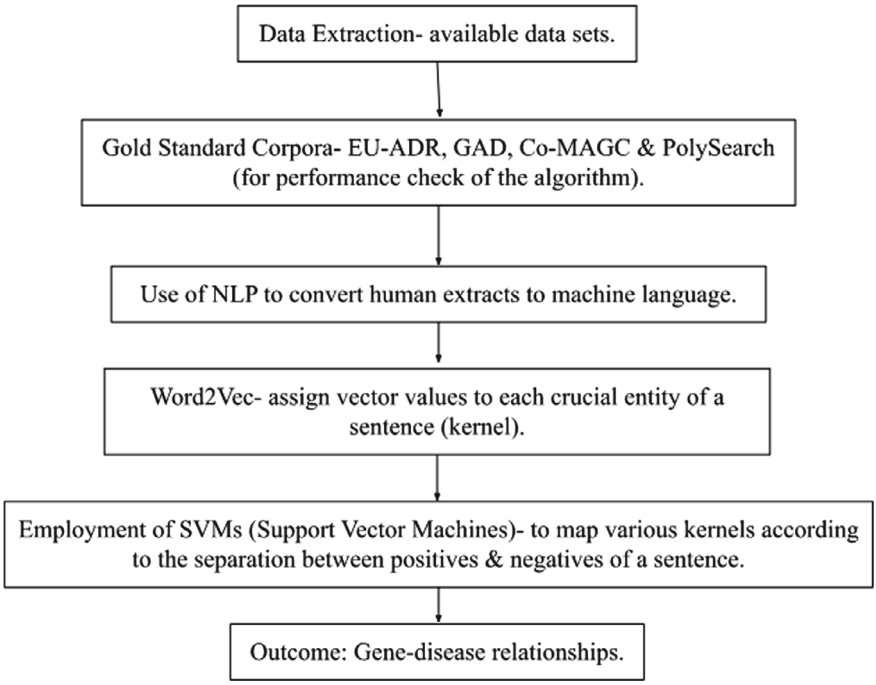

Fig. 17.4 Flowchart showing supervised relationship extraction process

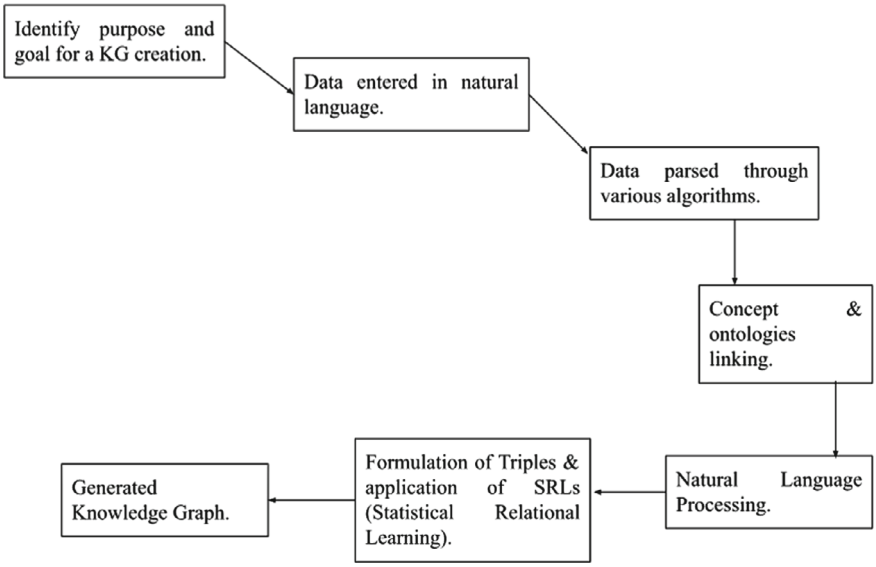

Fig. 17.5 Flow diagram representing the construction of a KG without supervision

Ebeid (2020) indicate that Cyc is a manually curated knowledge graph, Freebase and Wikidata were a product of technology. Moreover, these can also be a result of both automatic and manual curation, e.g. NELL (Never Ending Language Learning) is an example of such knowledge graph. The flow diagram for construction of KG is presented in Fig. 17.5.

17.4 Case Study: Automatic Knowledge Graph Construction from Biomedical Abstracts

Representing unstructured data, e.g. text to a more structured format, has many important benefits like querying, inferencing, question answering, visualization, integration, and many more. Knowledge graphs are one such structure that allows real-world knowledge available in an unstructured format to be represented in a structured format. A typical knowledge graph consists of a set of entities represented as nodes and the relationships between entities represented as labeled edges, which together (a pair of entities and the relation, also known as Triple) represent a real-world fact. Knowledge graphs can be created manually, automatically, or semi-automatically. A typical process of knowledge graph construction involves Named Entity Recognition (NER) and relation extraction. Recent progress in deep learning-based NER has significantly improved the automatic identification of entities and the relationship between them. Once we have identified the entities and relationship, we can push this information to a graph database like Neo4J for further querying, visualization, inferencing, and integration with other knowledge sources. Advancement in graph database technologies has enabled the efficient implementation of graph algorithms on larger graph-structured data and pushed the knowledge of graph-based analysis and research to the mainstream.

In this case study, we will demonstrate how to construct a knowledge graph from the abstract text of biomedical literature available in PubMed. We will represent each biomedical literature as a node labeled with its PubMed ID (PMID). An article may consist of entities of various kinds like genes, diseases, chemicals, species, mutation, DNA, RNA, etc. To keep our knowledge graph simple, we will be extracting only disease and gene mentions in the abstract text of the article. Each PMID node will be connected to disease and gene nodes using 'MENTIONS_DISEASE' and 'MENTIONS_GENE' relations. We will use BERN2 (Sung et al. 2022) as an NER tool and Neo4J a scalable graph database management system for this. We present some generic steps in automatic construction of knowledge graph.

Step-1: Corpus Curation

The knowledge graph can be constructed from structured, semi-structured, and unstructured data. Here we use unstructured sources like an abstract of a biomedical article from PubMed. We can include or exclude certain articles depending on the purpose of knowledge graph construction. For example, if one wants to get some

insight into treatment for pediatric oncology, he/she may exclude articles unrelated to pediatric oncology and include only relevant articles to create the knowledge graph.

Step-2: Named Entity Recognition (NER)

The next step toward knowledge graph construction is NER. It is a task in NLP, in which a fragment of free-flowing text is annotated or labeled with its corresponding entity type. For example, all mentions of 'cancer' or 'neoplasm' can be labeled '*Disease*'. We can use our custom NER tool or the best freely available tool. We are using BERN2 (Sung et al. 2022), an advanced neural network-based Named Entity Recognition and normalization tool, which we found to be freely available and most efficient, as demonstrated by García del Valle et al. (2021). Setup instructions can be found on their web page.[1] Once set up and deployed as REST API, we can use Python '*requests*' library to make an HTTP post request with the text to be annotated to the BERN URL and get an annotated result as a response.

Step-3: Relation Extraction (RE)

As we construct a knowledge graph to represent a biomedical research article, we use 'MENTIONS_DISEASE' and 'MENTIONS_GENE' as relation labels to connect an article represented as a PMID node to disease and gene nodes, respectively.

Step-4: Graph Database Population

Once we have identified the triples, we can insert these triples to graph databases for efficient storage, analysis, and visualization. We use one of the most popular graph database management systems, Neo4J,[2] with native graph storage and processing capabilities. Installation instructions can be found on the Neo4J website. We have installed Neo4J Desktop locally and installed the '*neo4j*' Python library, an official Neo4j driver for Python.

17.5 Demo

1. Start Jupyter Note Book and start executing the below code; first, install the neo4j driver for Python.

```
#install dependencies
!pip install neo4j
```

2. Import required libraries.

```
#import libraries
import requests
from neo4j import GraphDatabase
```

[1] https://github.com/dmis-lab/BERN2.

[2] https://neo4j.com/.

3. Below is a list of dictionaries with '*pmid*' and abstract as '*key*'. '*pmid*' is the PubMed ID of an article, and 'abstract' contains the abstract text of the article, which can be accessed using URL https://pubmed.ncbi.nlm.nih.gov/22554099/ for PMID 22554099. Replace the 'abstract' value with actual abstract text using the above URL for different PMIDs.

```
#our corpus; we can initialize this from a CSV also
article_list = [
           {          'pmid':'22554099',
           #https://pubmed.ncbi.nlm.nih.gov/22554099/
             'abstract':"'Abstract of PMID-22554099"'
            },
           {          'pmid':'32476297',
          #https://pubmed.ncbi.nlm.nih.gov/32476297/
             'abstract':"'Abstract of PMID-32476297"'
            },
            {           'pmid':'31466300', #https://pubmed.ncbi.
nlm.nih.gov/31466300/
             'abstract':"'Abstract of PMID-31466300"'
            }
        ]
```

4. Initialize connection to Neo4J database instance; make sure you have kept your database opened, i.e. Neo4J instance up and running.

```
bolt_url = 'bolt://localhost:7687'
user_id = 'neo4j'
pwd = 'myPassword'
gdbConnection    =    GraphDatabase.driver(bolt_url,auth=(user_
id,pwd))
```

5. Open Neo4J Browser Shell and Run the following commands; this adds a constraint to the Neo4J database to avoid duplicate nodes. **Note**: This is a cypher query to be run in Neo4J.

```
CREATE CONSTRAINT for (n: BioMedLiterature) require n.PMID IS
UNIQUE
CREATE CONSTRAINT for (n:Disease) require n.Name IS UNIQUE
CREATE CONSTRAINT for (n:Gene) require n.Name IS UNIQUE
```

6. This step method gets the NER response from the BERN2 server.

```
def query_bern_plain(text, url="http://localhost:8888/plain"):
  return requests.post(url, json={'text': text}).json()
```

7. We can see the NER response for the first article using the below code.

```
json_response = query_bern_plain(article_list[0]['abstract'])
#showing for first article abstract
print(json_response)
```

8. This step creates a node in Neo4J.

```
def create_node(node_type,node_properties):
   query ="CREATE (n: {} {})".format(node_type,node_properties)
   print(query)
   with gdbConnection.session() as session:
      result = session.run(query)
```

9. This method checks if an edge already exists to avoid duplicate edges.

```
def      doc_rel_already_exists(pmid,rel_name,node2_type,node2_
name):
   exists = False
    query = ''MATCH c= (:BioMedLiterature {{PMID: '{}'}})-[:{}]->
(:{}{{Name :'{}'}})
         RETURN count(c)"'.format(pmid,rel_name,node2_type,node2_
name)
   #print(query)
   with gdbConnection.session() as session:
      result = session.run(query)
      if(int(result.single()["count(c)"]) > 0):
         exists = True
   return exists
```

10. This method adds an edge/relationship between a PMID node and disease or gene nodes.

```
#add edges
def create_doc_relation(pmid,rel_name,node2_type,node2_name):
    if(not oc_rel_already_exists(pmid,rel_name,node2_type,node2_
name)):
      query = ''MATCH
           (a:BioMedLiterature {{PMID : '{}'}}),
           (b:{} {{Name:'{}'}})
           CREATE (a)-[r:{}]->(b)
           RETURN
b"'.format(pmid,node2_type,node2_name,rel_name)
      with gdbConnection.session() as session:
         result = session.run(query)
```

11. This step code acts as the main method. It iterates over each article in the list and gets PMID and text; then, it calls the BERN server for annotated JSON response. Then, we extract disease and gene mentions from the JSON file and create nodes and corresponding edges with our pre-defined relationships in Neo4J database.

```
for article in article_list:
   pmid = article['pmid']
   #print(pmid)
   abstract_text = article['abstract']
   #for NER
   json_response = query_bern_plain(abstract_text)
   annotations = json_response['annotations']
  disease_mentions = [x['mention'].lower() for x in annotations if
x['obj'].lower() == 'disease' ]
    genes = [x['mention'] for x in annotations if "NCBIGene" in
x['id'][0]]
   #create article node..representing the text
   try:
    create_node('BioMedLiterature',"{{PMID:'{}'}}".format(pmid))
   except:
     pass
   #create disease nodes
   for disease_name in set(disease_mentions):
     try:
create_node('Disease',"{{Name:'{}'}}".format(disease_name))
     except:
       pass
     try:
                              create_doc_relation(pmid,'MENTIONS_
DISEASE','Disease',disease_name)
     except :
       pass
  for gene in set(genes):
     try:
       create_node('Gene',"{{Name:'{}'}}".format(gene))
     except:
       pass
     try:
       create_doc_relation(pmid,'MENTIONS_GENE','Gene',gene)
     except :
       pass
```

12. Figure 17.6 shows the final knowledge graph generated in Neo4J Browser.

Red nodes represent an article labeled by its PMID, Yellow nodes represent genes, and Green nodes represent disease nodes.

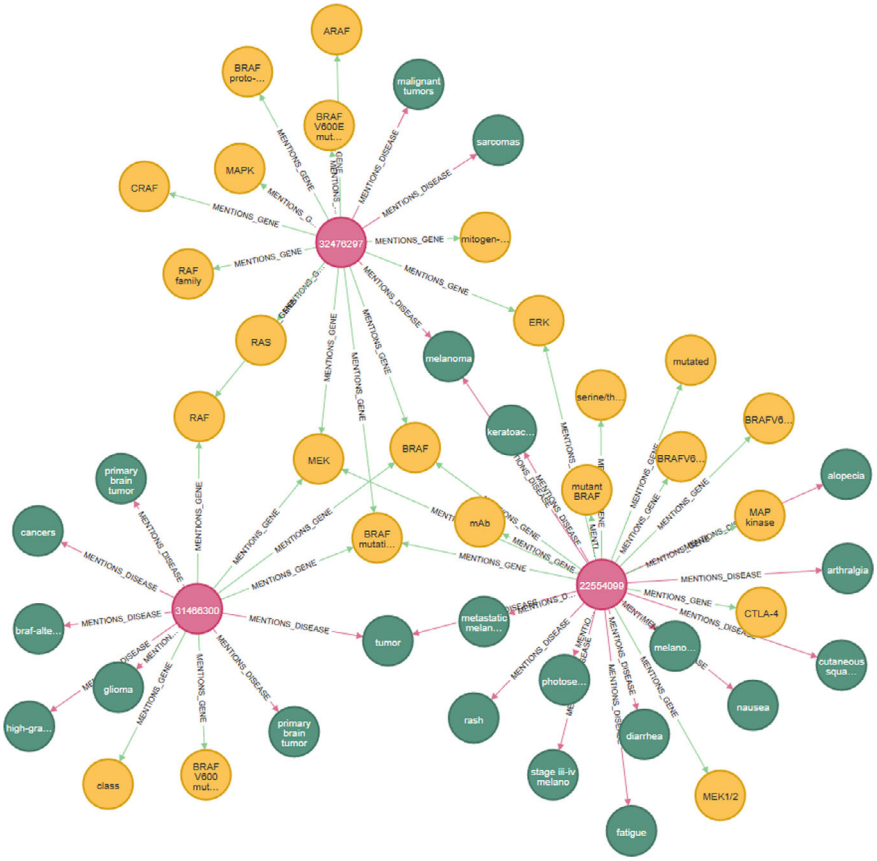

Fig. 17.6 Final knowledge graph generated

17.6 Summary

By providing a comprehensive overview of the methods for constructing knowledge graphs in the biomedical domain, this chapter aims to serve as a practical guide for researchers and practitioners interested in leveraging knowledge graphs to accelerate biomedical research and improve patient outcomes, the typical process of knowledge graph construction, which involves NER and relation extraction. By connecting the knowledge graph with external knowledge bases or datasets, researchers and practitioners can leverage a broader range of information to gain deeper insights and make more informed decisions.

References

Abu-Salih B Al-Qurishi M, Alweshah M, Al-Smadi M, Alfayez R, Saadeh H (2022) Healthcare knowledge graph construction: state-of-the-art, open issues, and opportunities. arXiv https://doi.org/10.48550/arXiv.2207.03771

Bhasuran B, Natarajan J (2018) Automatic extraction of gene-disease associations from literature using joint ensemble learning. PLoS ONE. Accessed 19 Jan 2024. [Online]. Available: https://doi.org/10.1371/journal.pone.0200699

Chaves-Fraga D (2021) Knowledge graph construction from heterogeneous data sources exploiting declarative mapping rules. https://doi.org/10.20868/UPM.thesis.67890

COSMIC: somatic cancer genetics at high-resolution. Nucleic Acids Res. Oxford Academic. Accessed 19 Jan 2024. [Online]. Available: https://academic.oup.com/nar/article/45/D1/D777/2605743

CS 2112/ENGRD 2112 Fall 2018. Accessed 19 Jan 2024. [Online]. Available: https://www.cs.cornell.edu/courses/cs2112/2018fa/lectures/lecture.html?id=parsing

Ebeid I (2020) Knowledge graph mining: a survey of methods, approaches, and applications. https://doi.org/10.13140/RG.2.2.21187.48168/2

Entrez molecular sequence database system. Accessed 19 Jan 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html

Futia G (2020) Building knowledge graphs from structured sources. Medium. Accessed 19 Jan 2024. [Online]. Available: https://towardsdatascience.com/building-knowledge-graphs-from-structured-sources-346c56c9d40e

García del Valle EP, Lagunes García G, Prieto Santamaría L, Zanin M, Menasalvas Ruiz E, Rodríguez-González A (2021) Leveraging network analysis to evaluate biomedical named entity recognition tools. Sci Rep 11(1). Art. No. 1. https://doi.org/10.1038/s41598-021-93018-w

GeeksforGeeks. Parse tree in compiler design. Accessed 19 Jan 2024. [Online]. Available: https://www.geeksforgeeks.org/parse-tree-in-compiler-design/

Grondin CJ et al (2021) Predicting molecular mechanisms, pathways, and health outcomes induced by Juul e-cigarette aerosol chemicals using the Comparative Toxicogenomics Database. Curr Res Toxicol 2:272–281. https://doi.org/10.1016/j.crtox.2021.08.001

Hao X et al (2021) Construction and application of a knowledge graph. Remote Sens 13:2511. https://doi.org/10.3390/rs13132511

Lenzerini M (2002) Data integration: a theoretical perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, PODS'02. Association for Computing Machinery, New York, NY, pp 233–246. https://doi.org/10.1145/543613.543644

Mintz M, Bills S, Snow R, Jurafsky D (2009) Distant supervision for relation extraction without labeled data. In: Su K-Y, Su J, Wiebe J, Li H (eds) Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, Aug 2009. Association for Computational Linguistics, Suntec, Singapore, pp 1003–1011. Accessed 19 Jan 2024. [Online]. Available: https://aclanthology.org/P09-1113

Mitrofan M, Barbu Mititelu V, Mitrofan G (2018) Towards the construction of a gold standard biomedical corpus for the Romanian language. Data 3(4). Art. No. 4. https://doi.org/10.3390/data3040053

Nicholson DN, Greene CS (2020) Constructing knowledge graphs and their biomedical applications. PubMed. Accessed 19 Jan 2024. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32637040/

OMIM. Home. NCBI. Accessed 19 Jan 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/omim

Poggi A, Lembo D, Calvanese D, De Giacomo G, Lenzerini M, Rosati R (2008) Linking data to ontologies. J Data Semant 10:133–173. https://doi.org/10.1007/978-3-540-77688-8_5

Protein Data Bank. Nucleic Acids Res. Oxford Academic. Accessed 19 Jan 2024. [Online]. Available: https://academic.oup.com/nar/article/28/1/235/2384399

Ryen V, Soylu A, Roman D (2022) Building semantic knowledge graphs from (semi-)structured data: a review. Future Internet 14:129. https://doi.org/10.3390/fi14050129

Sung M, Jeong M, Choi Y, Kim D, Lee J, Kang J (2022) BERN2: an advanced neural biomedical named entity recognition and normalization tool. Bioinformatics 38(20):4837–4839. https://doi.org/10.1093/bioinformatics/btac598

The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Res 49(D1):D480–D489. https://doi.org/10.1093/nar/gkaa1100