Open Data Science – Semester Project

CONSTRUCT A KNOWLEDGE GRAPH WITH NLP.

Introduction: The biomedical field is a prime example where representing the data as a graph makes sense as you are often analyzing interactions and relations between genes, diseases, drugs, proteins, and more. For instance, In Figure 1 we have ascorbic acid, also known as vitamin C, and some of its relations to other concepts, showing that vitamin C could be used to treat chronic gastritis. Now, you could have a team of domain experts map all of those connections between drugs, diseases, and other biomedical concepts for you. But, unfortunately, not many of us can afford to hire a team of medical doctors to do the work for us. In that case, we can resort to using NLP techniques to extract those relationships automatically. The good part is that we can use an NLP pipeline to read all of the research papers out there, and the bad part is that not all obtained results will be perfect. However, given that we don't have a team of scientists ready at our side to extract relations manually, we will resort to using NLP techniques to construct a biomedical knowledge graph of our own.

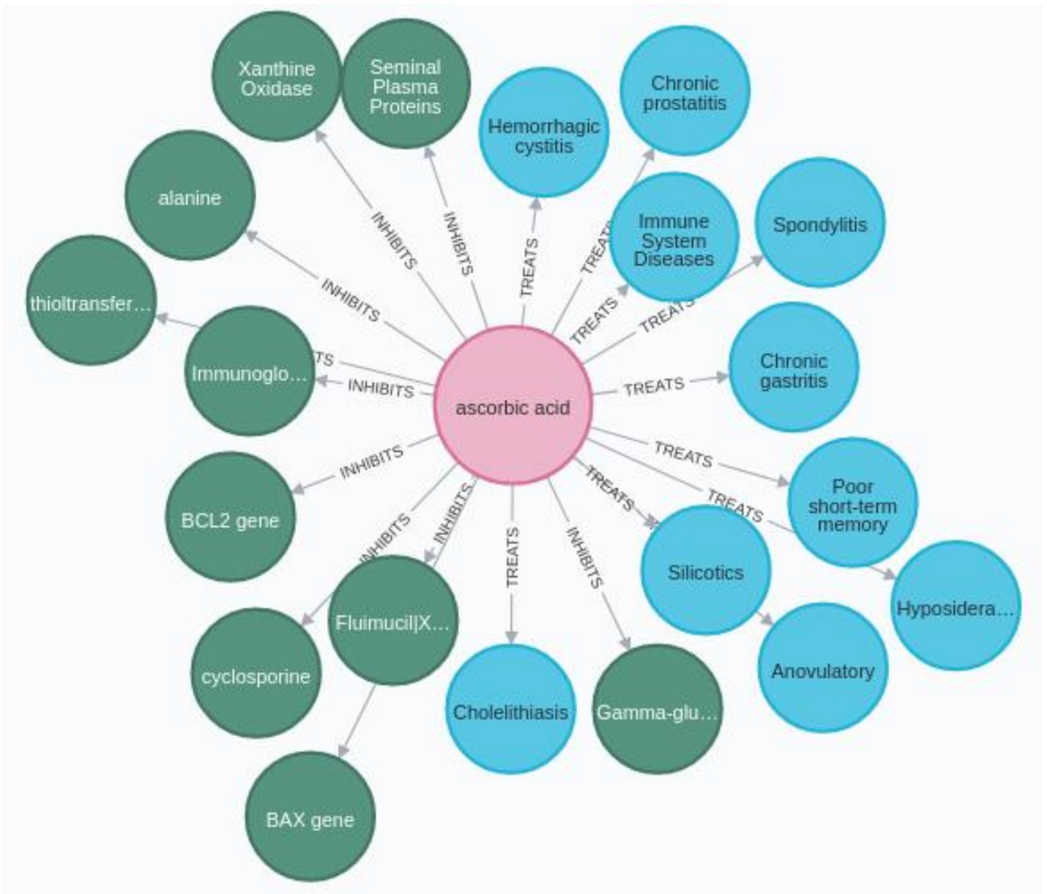


Figure 1. Example subgraph that shows ascorbic acid relationships to other biomedical concepts.

Objectives: Combine OCR, named entity linking, relation extraction and external enrichment databases to construct a knowledge graph. By the end of this semester project, on the basis of a PDF research article, you will have to construct a graph following the schema presented in Figure 2. Each article can have one or more authors. We will split the article content into sentences and use NLP to extract both medical entities and their relationships. It might be a bit counter-intuitive that we will store the relations between entities as intermediate nodes instead of relationships. The critical factor behind this decision is that we want to have an audit trail of the source text from which the relation was extracted. With the labeled property graph model, you can't have a relationship pointing to another relationship. For this reason, we refactor the connection between medical concepts into an intermediate node. This will also allow a domain expert to evaluate if a relation was correctly extracted or not.

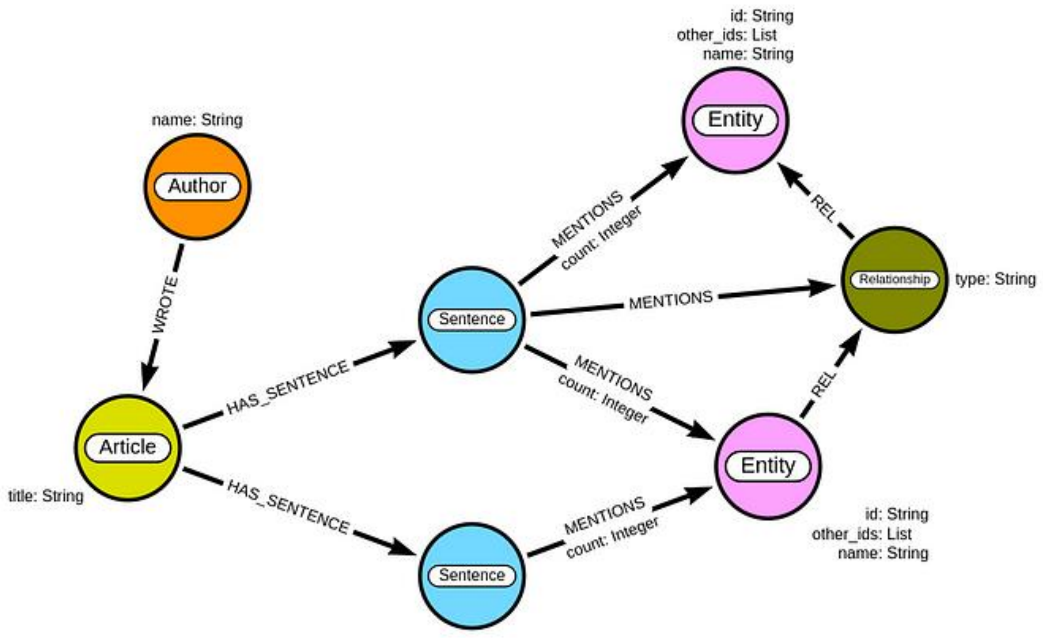


Figure 2. Biomedical graph schema.

Steps:
1. Setup your Python development environment
2. Read a PDF document with OCR
3. Text preprocessing
4. Biomedical named entity recognition and linking
5. Construct a knowledge graph
6. Relation extraction
7. External database enrichment

Deliverable: A **PDF report outlining and commenting on the case's resolution**, challenges encountered and potential use cases of your final knowledge graph.

More in details…

Setup your Python development environment

Install Python3.7 and the Anaconda distribution on your workstation. With the Anaconda Prompt, create a new virtual environment and activate it!

```
conda create --name semesterproject python=3.7
conda activate semesterproject
```

Still in the Anaconda Prompt install the following dependencies in your virtual environment.

```
conda install -c conda-forge tesseract
conda install -c conda-forge poppler
conda install -c anaconda nltk
conda install -c anaconda pandas
pip install pytesseract pdf2image zero-shot-re neo4j
```

Reading a PDF document with OCR

To start you will be using the *Tissue Engineering of Skin Regeneration and Hair Growth* paper written by *Mohammadreza Ahmadi*. The PDF version of the article is available under the CC0 1.0 license, which means you can easily download it with Python. The *pytesseract* library is one of the most popular libraries for OCR.

```
import requests
import pdf2image
import pytesseract

pdf = requests.get('https://arxiv.org/pdf/2110.03526.pdf')
doc = pdf2image.convert_from_bytes(pdf.content)

# Get the article text
article = []
for page_number, page_data in enumerate(doc):
     txt = pytesseract.image_to_string(page_data).encode("utf-8")
     # Sixth page are only references
     if page_number < 6:
       article.append(txt.decode("utf-8"))
article_txt = " ".join(article)
```

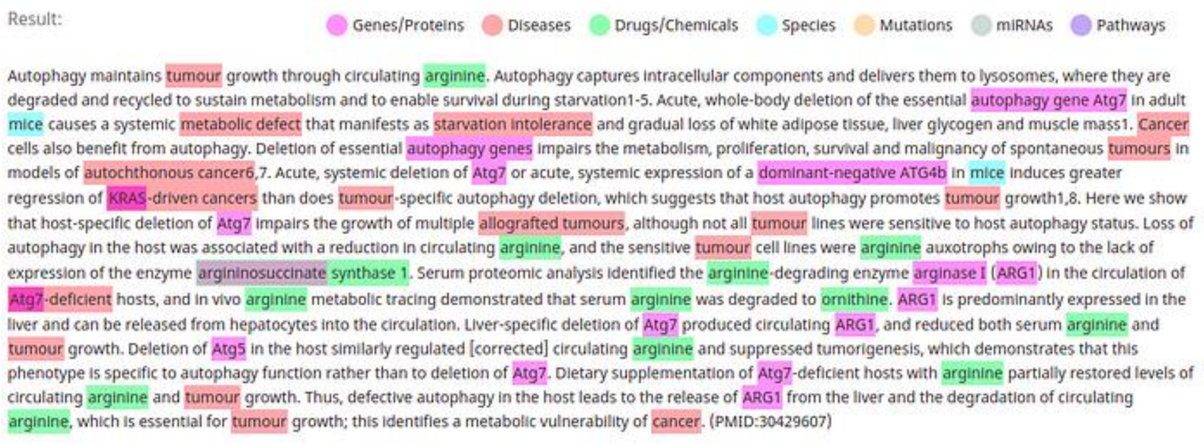
Text preprocessing

Now that you have the article content available, you will need to remove section titles and figure descriptions from the text. Here is an example on how to possibly do it in Python:

```
import nltk
nltk.download('punkt')

def clean_text(text):
  """Remove section titles and figure descriptions from text"""
  clean = "\n".join([row for row in text.split("\n") if
(len(row.split(" "))) > 3 and not (row.startswith("(a)"))
                              and not row.startswith("Figure")])
  return clean

text = article_txt.split("INTRODUCTION")[1]
ctext = clean_text(text)
sentences = nltk.tokenize.sent_tokenize(ctext)
```

Biomedical named entity linking

Now comes the exciting part. Named entity recognition techniques are used to detect relevant entities or concepts in the text. For example, in the biomedical domain, we want to identify various genes, drugs, diseases, and other concepts in the text.



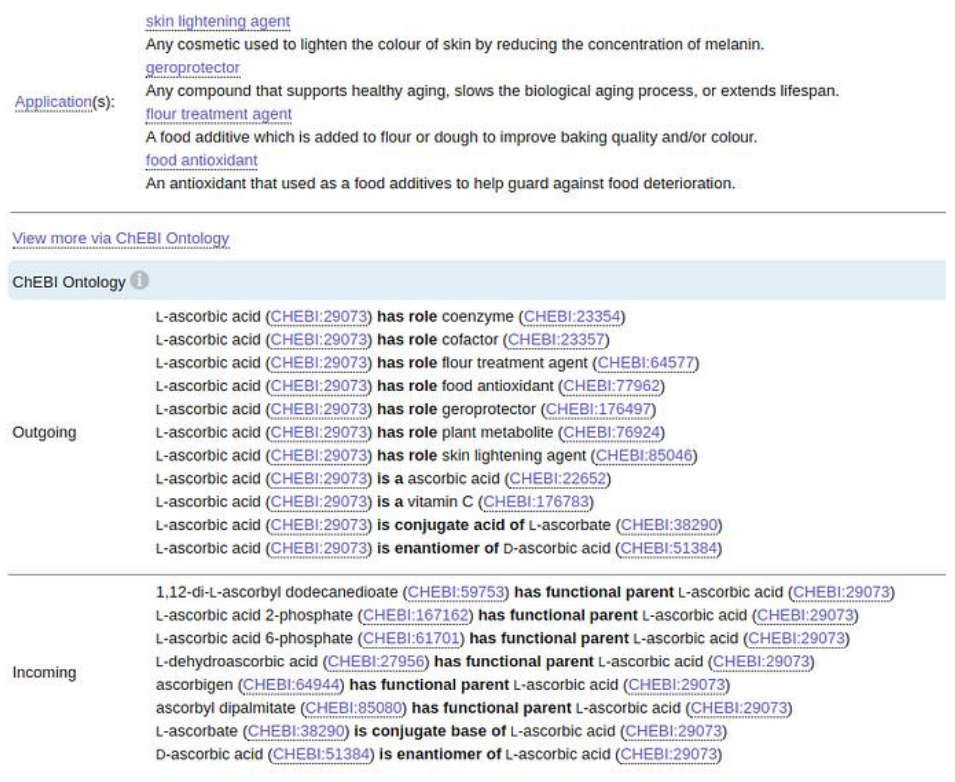
Autophagy maintains tumour growth through circulating arginine. Autophagy captures intracellular components and delivers them to lysosomes, where they are degraded and recycled to sustain metabolism and to enable survival during starvation1-5. Acute, whole-body deletion of the essential autophagy gene Atg7 in adult mice causes a systemic metabolic defect that manifests as starvation intolerance and gradual loss of white adipose tissue, liver glycogen and muscle mass1. Cancer cells also benefit from autophagy. Deletion of essential autophagy genes impairs the metabolism, proliferation, survival and malignancy of spontaneous tumours in models of autochthonous cancer6,7. Acute, systemic deletion of Atg7 or acute, systemic expression of a dominant-negative ATG4b in mice induces greater regression of KRAS-driven cancers than does tumour-specific autophagy deletion, which suggests that host autophagy promotes tumour growth1,8. Here we show that host-specific deletion of Atg7 impairs the growth of multiple allografted tumours, although not all tumour lines were sensitive to host autophagy status. Loss of autophagy in the host was associated with a reduction in circulating arginine, and the sensitive tumour cell lines were arginine auxotrophs owing to the lack of expression of the enzyme argininosuccinate synthase 1. Serum proteomic analysis identified the arginine-degrading enzyme arginase I (ARG1) in the circulation of Atg7-deficient hosts, and in vivo arginine metabolic tracing demonstrated that serum arginine was degraded to ornithine. ARG1 is predominantly expressed in the liver and can be released from hepatocytes into the circulation. Liver-specific deletion of Atg7 produced circulating ARG1, and reduced both serum arginine and tumour growth. Deletion of Atg5 in the host similarly regulated [corrected] circulating arginine and suppressed tumorigenesis, which demonstrates that this phenotype is specific to autophagy function rather than to deletion of Atg7. Dietary supplementation of Atg7-deficient hosts with arginine partially restored levels of circulating arginine and tumour growth. Thus, defective autophagy in the host leads to the release of ARG1 from the liver and the degradation of circulating arginine, which is essential for tumour growth; this identifies a metabolic vulnerability of cancer. (PMID:30429607)

Figure 3. Biomedical concepts extraction.

In this example, the NLP model identified genes, diseases, drugs, species, mutations, and pathways in the text. As mentioned, this process is called named entity recognition. An upgrade to the named entity recognition is the so-called named entity linking. The named entity linking technique detects relevant concepts in the text and tries to map them to the target knowledge base. In the biomedical domain, some of the target knowledge bases are:

- MESH
- CHEBI
- OMIM
- ENSEMBL

Why would we want to link medical entities to a target knowledge base? The primary reason is that it helps us deal with entity disambiguation. For example, we don't want separate entities in the graph representing ascorbic acid and vitamin C as domain experts can tell you those are the same thing. The secondary reason is that by mapping concepts to a target knowledge base, we can enrich our graph model by fetching information about the mapped concepts from the target knowledge base. If we use the ascorbic acid example again, we could easily fetch additional information from the CHEBI database if we already know its CHEBI id.



Application(s):

skin lightening agent
Any cosmetic used to lighten the colour of skin by reducing the concentration of melanin.
geroprotector
Any compound that supports healthy aging, slows the biological aging process, or extends lifespan.
flour treatment agent
A food additive which is added to flour or dough to improve baking quality and/or colour.
food antioxidant
An antioxidant that used as a food additives to help guard against food deterioration.

View more via ChEBI Ontology

ChEBI Ontology

Outgoing

L-ascorbic acid (CHEBI:29073) **has role** coenzyme (CHEBI:23354)
L-ascorbic acid (CHEBI:29073) **has role** cofactor (CHEBI:23357)
L-ascorbic acid (CHEBI:29073) **has role** flour treatment agent (CHEBI:64577)
L-ascorbic acid (CHEBI:29073) **has role** food antioxidant (CHEBI:77962)
L-ascorbic acid (CHEBI:29073) **has role** geroprotector (CHEBI:176497)
L-ascorbic acid (CHEBI:29073) **has role** plant metabolite (CHEBI:76924)
L-ascorbic acid (CHEBI:29073) **has role** skin lightening agent (CHEBI:85046)
L-ascorbic acid (CHEBI:29073) **is a** ascorbic acid (CHEBI:22652)
L-ascorbic acid (CHEBI:29073) **is a** vitamin C (CHEBI:176783)
L-ascorbic acid (CHEBI:29073) **is conjugate acid of** L-ascorbate (CHEBI:38290)
L-ascorbic acid (CHEBI:29073) **is enantiomer of** D-ascorbic acid (CHEBI:51384)

Incoming

1,12-di-L-ascorbyl dodecanedioate (CHEBI:59753) **has functional parent** L-ascorbic acid (CHEBI:29073)
L-ascorbic acid 2-phosphate (CHEBI:167162) **has functional parent** L-ascorbic acid (CHEBI:29073)
L-ascorbic acid 6-phosphate (CHEBI:61701) **has functional parent** L-ascorbic acid (CHEBI:29073)
L-dehydroascorbic acid (CHEBI:27956) **has functional parent** L-ascorbic acid (CHEBI:29073)
ascorbigen (CHEBI:64944) **has functional parent** L-ascorbic acid (CHEBI:29073)
ascorbyl dipalmitate (CHEBI:85080) **has functional parent** L-ascorbic acid (CHEBI:29073)
L-ascorbate (CHEBI:38290) **is conjugate base of** L-ascorbic acid (CHEBI:29073)
D-ascorbic acid (CHEBI:51384) **is enantiomer of** L-ascorbic acid (CHEBI:29073)

Figure 4. Enrichment data available about ascorbic acid on CHEBI website. All the content on the website is available under CC BY 4.0 license.

Lots of NLP models focus on extracting only a specific subset of medical concepts like genes or diseases. It is even rarer to find a model that detects most medical concepts and links them to a target knowledge base. Luckily, we can rely on BERN2[1], a neural biomedical entity recognition and multi-type normalization tool. It is a fine-tuned BioBert model with various named entity linking models integrated for mapping concepts to biomedical target knowledge bases. Not only that, but they also provide a free REST endpoint, so we don't have to deal with the headache of getting the dependencies and the model to work. The biomedical named entity recognition visualization used above was created using the BERN model, so we know it detects genes, diseases, drugs, species, mutations, and pathways in the text. Unfortunately, the BERN model does not assign target knowledge base ids to all concepts. So, the following script will help you looking if a distinct id is given for a concept, and if it is not, it will use the entity name as the id. It will also compute the sha256 of the text of sentences to identify specific sentences easier later when we will be doing relation extraction.

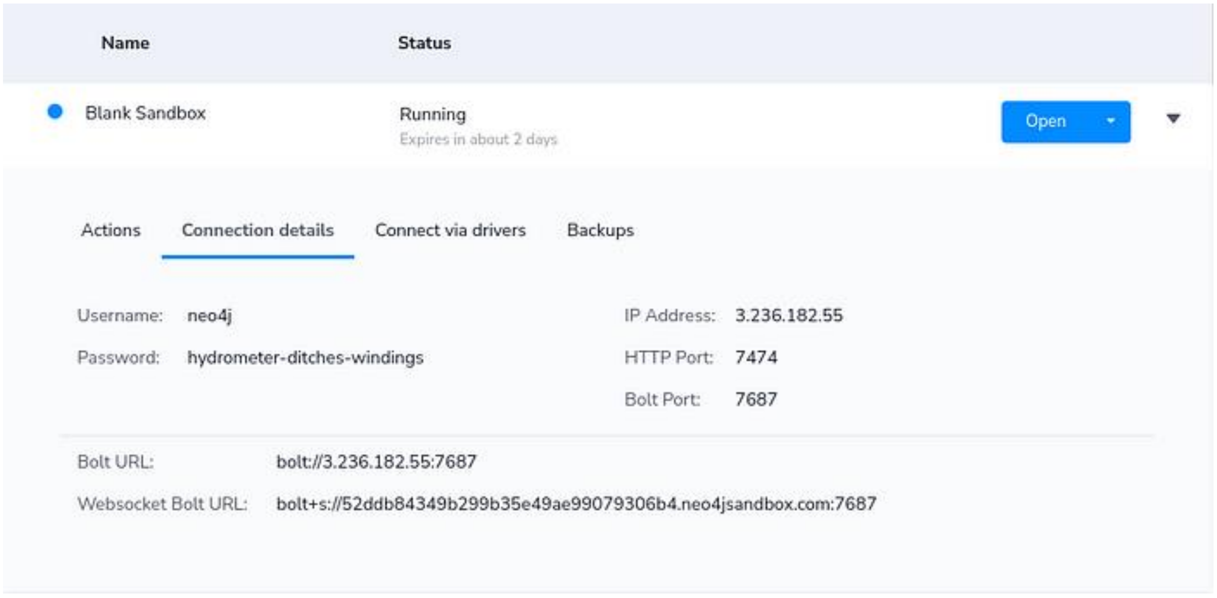
```
import hashlib

def query_plain(text, url="http://bern2.korea.ac.kr/plain"):
  """Biomedical entity linking API"""
  return requests.post(url, json={'text':str(text)}).json()

entity_list = []
# The last sentence is invalid
for s in sentences[:-1]:
  entity_list.append(query_plain(s))

parsed_entities = []
for entities in entity_list:
  e = []
  # If there are not entities in the text
  if not entities.get('annotations'):
      parsed_entities.append({'text':entities['text'], 'text_sha256':
hashlib.sha256(entities['text'].encode('utf-8')).hexdigest()})
      continue
  for entity in entities['annotations']:
      other_ids = [id for id in entity['id'] if not
id.startswith("BERN")]
      entity_type = entity['obj']
      entity_name =
entities['text'][entity['span']['begin']:entity['span']['end']]
      try:
        entity_id = [id for id in entity['id'] if
id.startswith("BERN")][0]
      except IndexError:
        entity_id = entity_name
      e.append({'entity_id': entity_id, 'other_ids': other_ids,
'entity_type': entity_type, 'entity': entity_name})
  parsed_entities.append({'entities':e, 'text':entities['text'],
'text_sha256': hashlib.sha256(entities['text'].encode('utf-
8')).hexdigest()})
```

Construct a knowledge graph

Before looking at relation extraction techniques, we will construct a biomedical knowledge graph using only entities and examine the possible applications. You don't have to deal with preparing a local Neo4j environment. Instead, you can use a free Neo4j Sandbox instance.

Neo4j Sandbox

Start the Blank project in the sandbox and use your connection details into the Python code.



Blank Sandbox — Running, Expires in about 2 days

Username: neo4j
Password: hydrometer-ditches-windings
IP Address: 3.236.182.55
HTTP Port: 7474
Bolt Port: 7687

Bolt URL: bolt://3.236.182.55:7687
Websocket Bolt URL: bolt+s://52ddb84349b299b35e49ae99079306b4.neo4jsandbox.com:7687

Figure 5. Neo4j Sandbox connection details.

Now you can go ahead and prepare the Neo4j connection in the notebook.

```
from neo4j import GraphDatabase
import pandas as pd

host = 'bolt://34.203.247.205:7687'
user = 'neo4j'
password = 'strobe-mail-atmospheres'
driver = GraphDatabase.driver(host,auth=(user, password))

def neo4j_query(query, params=None):
     with driver.session() as session:
          result = session.run(query, params)
          return pd.DataFrame([r.values() for r in result],
columns=result.keys())
```

Start by importing the author and the article into the graph. The article node will contain only the title. If you open the Neo4j Browser, you should see a graph that looks like the one here below. In this example, Mahammadrez Ahmadi is the author and "Tissue Engineering and Regeneration of Skin and Hair Follicle Growth" is the title of the article:

```
# import the   and the article into the grap
author = article_txt.split("\n")[0]
title = " ".join(article_txt.split("\n")[2:4])

neo4j_query("""
MERGE (a:Author{name:$author})
MERGE (b:Article{title:$title})
MERGE (a)-[:WROTE]->(b)
""", {'title':title, 'author':author})
```

If you open the Neo4j Browser, you should see the following graph.

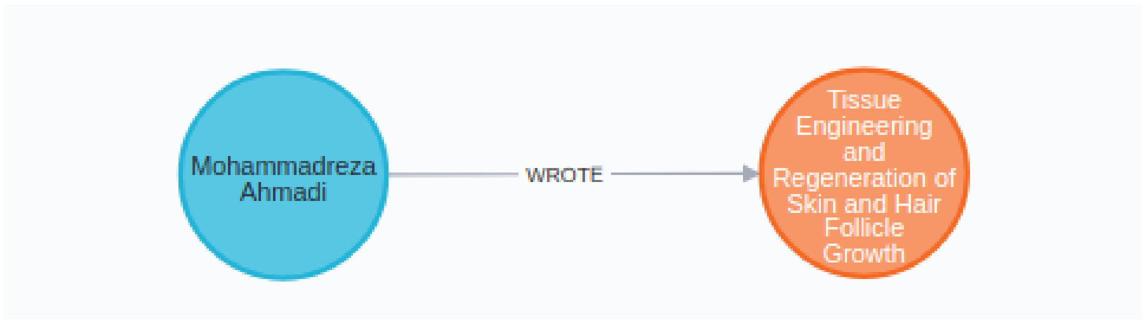

Mohammadreza Ahmadi
WROTE
Tissue Engineering and Regeneration of Skin and Hair Follicle Growth

Figure 6. Author - Article relation in Neo4j

You can import the sentences and mentioned entities by executing the following Cypher query:

```
# import the sentences and mentioned entities
neo4j_query("""
MATCH (a:Article)
UNWIND $data as row
MERGE (s:Sentence{id:row.text_sha256})
SET s.text = row.text
MERGE (a)-[:HAS_SENTENCE]->(s)
WITH s, row.entities as entities
UNWIND entities as entity
MERGE (e:Entity{id:entity.entity_id})
ON CREATE SET e.other_ids = entity.other_ids,
                  e.name = entity.entity,
                  e.type = entity.entity_type
MERGE (s)-[m:MENTIONS]->(e)
ON CREATE SET m.count = 1
ON MATCH SET m.count = m.count + 1
""", {'data': parsed_entities})
```

You can execute the following Cypher queries to inspect the constructed graph:

```
# example application 1 : search engine
neo4j_query("""
MATCH (e:Entity)<-[:MENTIONS]-(s:Sentence)
WHERE e.name = "autoimmune diseases"
RETURN s.text as result
""")

# example application 2 : co-occurrence
neo4j_query("""
MATCH (e1:Entity)<-[:MENTIONS]-()-[:MENTIONS]->(e2:Entity)
WHERE id(e1) < id(e2)
RETURN e1.name as entity1, e2.name as entity2, count(*) as
cooccurrence
ORDER BY cooccurrence
DESC LIMIT 3
""")

# example application 3 : author expertise
neo4j_query("""
MATCH (a:Author)-[:WROTE]->()-[:HAS_SENTENCE]->()-[:MENTIONS]-
>(e:Entity)
RETURN a.name as author, e.name as entity, count(*) as count
ORDER BY count DESC
LIMIT 5
""")
```

If you have correctly imported the data, you should potentially see a visualization looking like this:

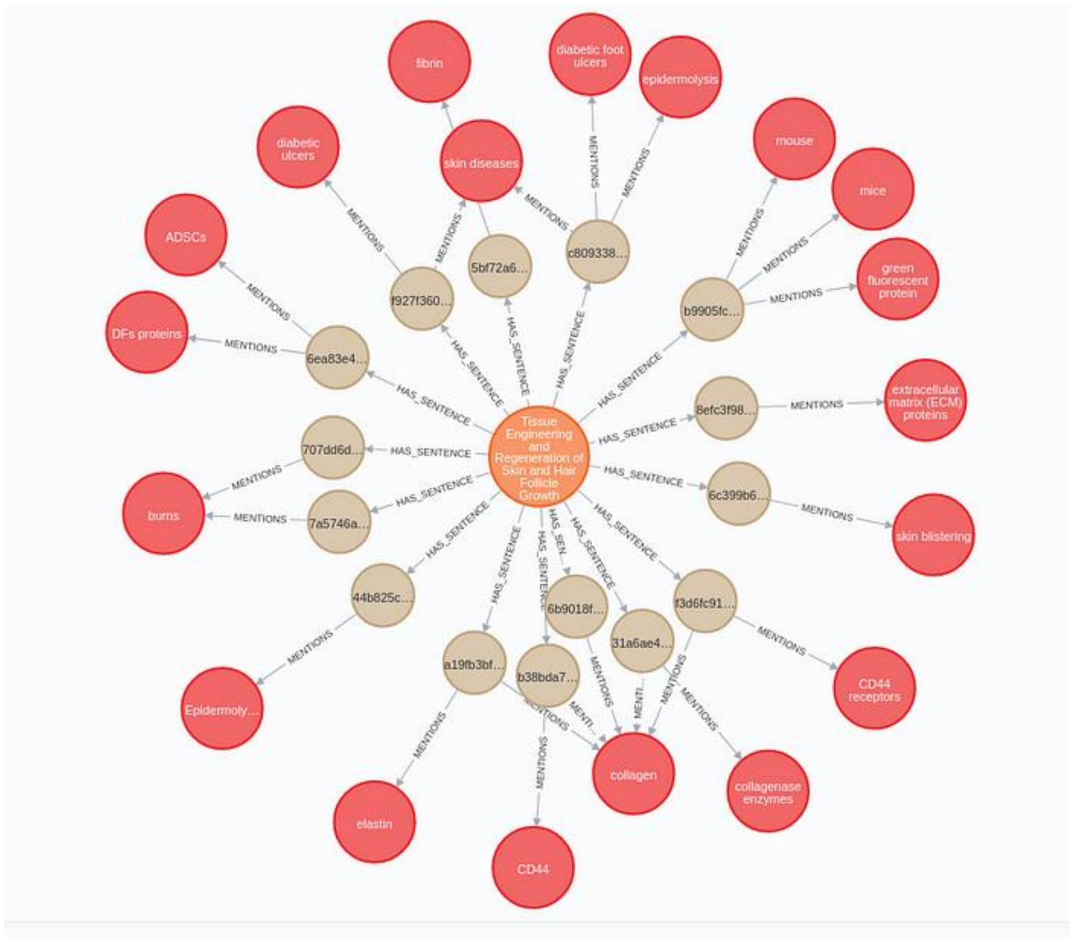


Figure 7. Entity extraction stored as a graph.

Relation extraction

Now you will try to extract relations between medical concepts. You can use the *zero-shot relation* extractor based on the paper *Exploring the zero-shot limit of FewRel*[2]. While it wouldn't be recommendable to put this model into production, it is good enough for the sake of this project. The model is available on HuggingFace, so we don't have to deal with training or setting up the model.

```
from transformers import AutoTokenizer
from zero_shot_re import RelTaggerModel, RelationExtractor

model = RelTaggerModel.from_pretrained("fractalego/fewrel-zero-
shot")
tokenizer = AutoTokenizer.from_pretrained("bert-large-uncased-whole-
word-masking-finetuned-squad")
relations = ['associated', 'interacts']
extractor = RelationExtractor(model, tokenizer, relations)
```

With the zero-shot relation extractor, you can define which relations you would like to detect. In this example, I've used the associated and interacts relationships. I've also tried more specific relationship types such as treats, causes, and others, but the results were not great.

With this model, you have to define between which pairs of entities you would like to detect relationships. We will use the results of the named entity linking as an input to the relation extraction process. First, we find all the sentences where two or more entities are mentioned and then run them through the relation extraction model to extract any connections. Here below the threshold value have been defined to 0.85, meaning that if a model predicts a link between entities with a probability lower than 0.85, we'll ignore the prediction.

```
# Candidate sentence where there is more than a single entity
present
candidates = [s for s in parsed_entities if (s.get('entities')) and
(len(s['entities']) > 1)]
predicted_rels = []
for c in candidates:
  combinations = itertools.combinations([{'name':x['entity'],
'id':x['entity_id']} for x in c['entities']], 2)
  for combination in list(combinations):
      try:
        ranked_rels = extractor.rank(text=c['text'].replace(",", "
"), head=combination[0]['name'], tail=combination[1]['name'])
        # Define threshold for the most probable relation
        if ranked_rels[0][1] > 0.85:
           predicted_rels.append({'head': combination[0]['id'],
'tail': combination[1]['id'], 'type':ranked_rels[0][0], 'source':
c['text_sha256']})
      except:
        pass
```

You can examine the extracted relationships between entities and the source text with the following Cypher query:

```
neo4j_query("""
UNWIND $data as row
MATCH (source:Entity {id: row.head})
MATCH (target:Entity {id: row.tail})
MATCH (text:Sentence {id: row.source})
MERGE (source)-[:REL]->(r:Relation {type: row.type})-[:REL]-
>(target)
MERGE (text)-[:MENTIONS]->(r)
""", {'data': predicted_rels})

# examine the extracted relationships
neo4j_query("""
MATCH (s:Entity)-[:REL]->(r:Relation)-[:REL]->(t:Entity), (r)<-
[:MENTIONS]-(st:Sentence)
RETURN s.name as source_entity, t.name as target_entity, r.type as
type, st.text as source_text
""")
```

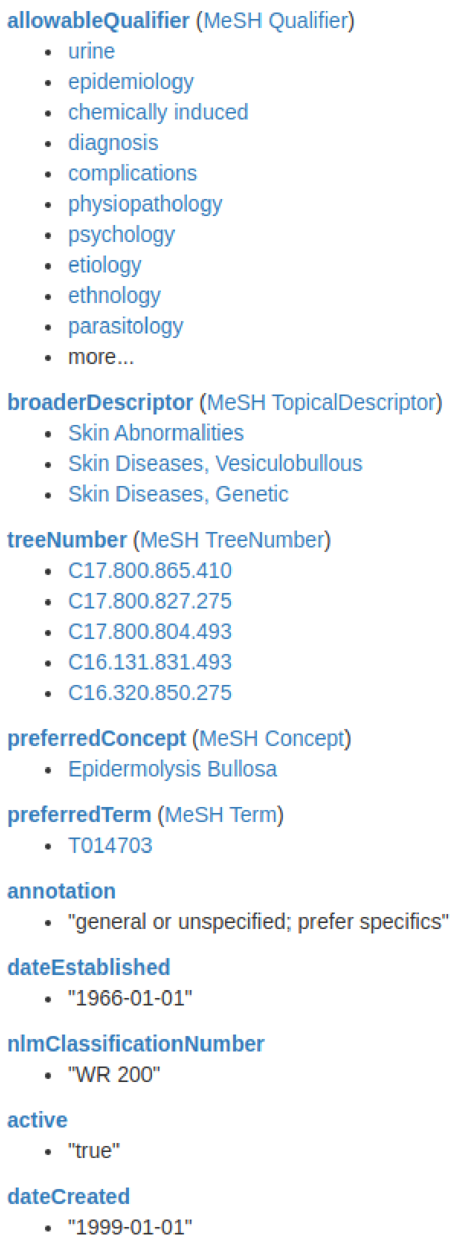
External database enrichment

You can use the external databases like CHEBI or MESH to enrich your graph. For example, your graph contains a medical entity Epidermolysis bullosa and we also know its MeSH id.

You can retrieve the MeSH id of Epidermolysis bullosa with the following query:

```
MATCH (e:Entity)
WHERE e.name = "Epidermolysis bullosa"
RETURN e.name as entity, e.other_ids as other_ids
```

You could go ahead and inspect the MeSH to find available information:



allowableQualifier (MeSH Qualifier)

- urine
- epidemiology
- chemically induced
- diagnosis
- complications
- physiopathology
- psychology
- etiology
- ethnology
- parasitology
- more...

broaderDescriptor (MeSH TopicalDescriptor)

- Skin Abnormalities
- Skin Diseases, Vesiculobullous
- Skin Diseases, Genetic

treeNumber (MeSH TreeNumber)

- C17.800.865.410
- C17.800.827.275
- C17.800.804.493
- C16.131.831.493
- C16.320.850.275

preferredConcept (MeSH Concept)

- Epidermolysis Bullosa

preferredTerm (MeSH Term)

- T014703

annotation

- "general or unspecified; prefer specifics"

dateEstablished

- "1966-01-01"

nlmClassificationNumber

- "WR 200"

active

- "true"

dateCreated

- "1999-01-01"

Figure 8. MeSH example data

Here is a screenshot of the available information on the MeSH website for Epidermolysis bullosa. We are not a medical doctor, so we don't know exactly what the best way would be to model this information in a graph. However, you can retrieve this information in Neo4j using the *apoc.load.json* procedure to fetch the information from the MeSH REST endpoint. And then, you can ask a domain expert to help you model this information.

The Cypher query to fetch the information from MeSH REST endpoint is:

```
MATCH (e:Entity)
WHERE e.name = "Epidermolysis bullosa"
WITH e,
    [id in e.other_ids WHERE id contains "MESH" |
split(id,":")[1]][0] as meshId
CALL
apoc.load.json("https://id.nlm.nih.gov/mesh/lookup/details?descripto
r=" + meshId) YIELD value
RETURN value
```

Last but not least

Try to further enrich your knowledge graph by:

1. Adding a new paper to your knowledge graph.

Using the PubMed library, search and select a paper that may be related to the one of *Mohammadreza Ahmadi*. Following the example of what has been done so far, import the phrases and entities of this new article into your knowledge graph.

2. Try to further enrich your knowledge graph by using SNOMED-CT and Wikidata.
3. Layout a **report** documenting all the work above mentioned, try to conclude by provide a couple of use-cases in which your graph could be useful.

References

This project is largely based on:
https://towardsdatascience.com/construct-a-biomedical-knowledge-graph-with-nlp-1f25eddc54a0

[1] D. Kim *et al.*, "A Neural Named Entity Recognition and Multi-Type Normalization Tool for Biomedical Text Mining," in *IEEE Access*, vol. 7, pp. 73729–73740, 2019, doi: 10.1109/ACCESS.2019.2920708.

[2] Cetoli, A. (2020). Exploring the zero-shot limit of FewRel. In *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 1447–1451). International Committee on Computational Linguistics.