

DSC 202 Final Report: LinkedIn Unleashed

Group 14

(Abdullah Ashfaq, Aagaaz Sayed, Jacob Ryan)

December 12, 2023



Table of Contents

Introduction	4
The Data	4
PostGres	4
MongoDB	5
Neo4j	7
ChromaDB	10
How they all connect	10
Infrastructure Set up	10
Data Loading: SQL+Mongo+Neo4j	11
Data Loading: ChromaDB	12
Design Considerations	13
Data Redundancy	13
PGVector vs ChromaDB	14
Task 0: Finding a Domain	14
How it Works	14
Step 1	15
Step 2	15
Step 3	15
Task 1: Finding a Mentor	16
How it Works	16
Step 1	16
Step 2	17
Step 3	17
Task 2: Researching your Dream Position	18
How it Works	18
Step 1	18
Step 2	18
Step 3	18
Step 4A	19
Step 4B	19
Task 3: Finding a Connection Path to your Dream Job	20
How it Works	20
Step 1	20
Step 2	21

Step 3	21
Task 4: Know companies that hire you!	22
How it works	22
Step 1	22
Step 2	24
Step 3	25
Step 4	26
Task 5: Connect with similar people and keep an eye out for referrals.	28
How it Works	28
Step 1	28
Step 2	30
Step 3	30
Limitations and Future Considerations	31

Introduction

Applying for jobs is hard. These days, when you search for job postings, you're hit with entry-level jobs expecting 3+ years of experience or job posting just a few hours old with thousands of applicants.

Our tool, **LinkedIn Unleashed**, aims to alleviate these problems by optimizing your job search.

Built directly from data scraped from LinkedIn, this tool can help you find better connections, groups, and advice, all with the goal of getting the best job possible.

Github link: <https://github.com/AbdullahAshfaq/LinkedInUnleashed>

Extra files for infrastructure setup and notebooks for exploration and testing are in github.

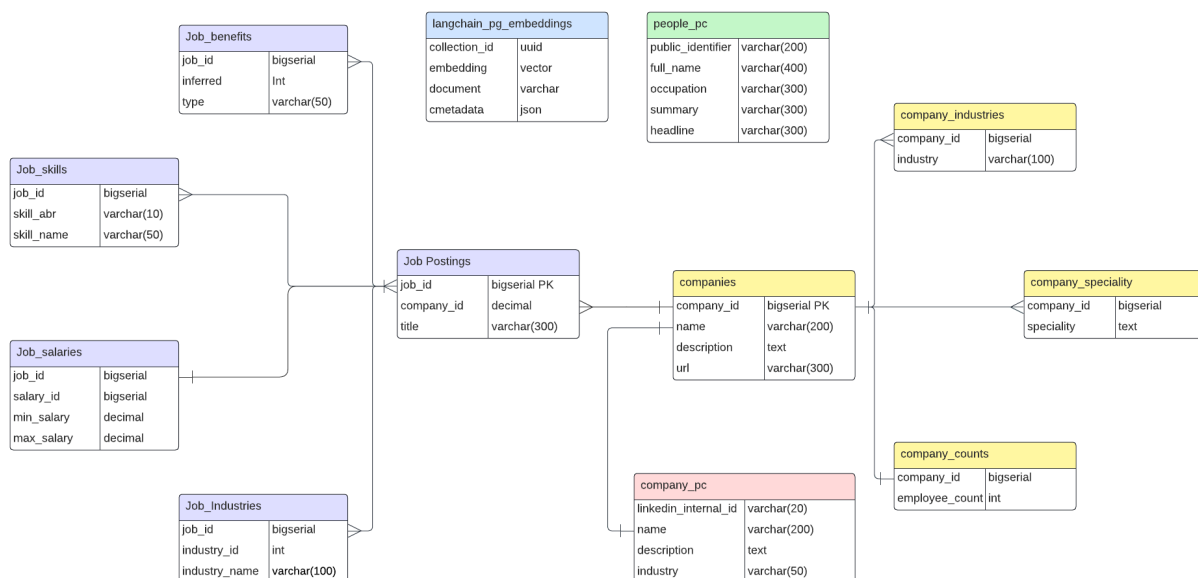
The Data

Our data initially comes from three sources.

1. [LinkedIn Companies Dataset](#)
2. [LinkedIn Users Dataset](#)
3. [LinkedIn Job Postings](#)

These datasets are processed and loaded into four databases as follows:

PostGres



The schema above shows the relational database created within PostGres for this tool.

There are three main types of objects.

1. **Job Postings**: these objects contain a variety of data. Most importantly, they contain a unique job id, a company id (foreign key for companies), and a job description. These job descriptions are further vectorized with the addition of pgVector. The embedding mechanism is discussed in Data Considerations
2. **Companies**: these objects contain typical information about a company: name, description, url, headquarter location, a unique id, industry. It also contains employee count and specialties.
3. **People**: these objects include name, occupation, summary, a unique id, and other various attributes commonly found on LinkedIn user profiles.

For clarity, we provide example tables of...

Job Postings

	job_id	company_id		title	description	formatted_experience_level
0	3757940104	553718.0		Hearing Care Provider	Overview\n\nHearingLife is a national hearing ...	Entry level
1	3757940025	2192142.0	Shipping & Receiving Associate 2nd shift (Beav...		Metalcraft of Mayville\n\nMetalcraft of Mayville...	None
2	3757938019	474443.0		Manager, Engineering	\n\nThe TSUBAKI name is synonymous with excellen...	None
3	3757938018	18213359.0		Cook	descriptionTitle\n\nLooking for a great oppor...	Entry level
4	3757937095	437225.0	Principal Cloud Security Architect (Remote)		Job Summary\n\nAt iHerb, we are on a mission to ...	Mid-Senior level
5	3757937037	13727.0		Territory Manager - New Haven	Location: Remote, CT, United States of America...	Mid-Senior level
6	3757937004	10515052.0		Auto Body Technician	Company: Gerber Collision & Glass\n\nWELCOME T...	Entry level
7	3757936167	2915.0	ACME D8- Asst Store Director (ASD) Sussex, NJ		The First Assistant Store Director is actively...	Mid-Senior level
8	3757936097	18213359.0		Dishwasher	descriptionTitle\n\n\$2,000 Sign-on Bonus Guar...	Entry level
9	3757932736	73013724.0		Sales Manager	Position Summary: Our Sales Manager has managi...	Mid-Senior level

Companies

	company_id		name	country	url
0	1009		IBM	US	https://www.linkedin.com/company/ibm
1	1016		GE HealthCare	US	https://www.linkedin.com/company/gehealthcare
2	1021		GE Power	US	https://www.linkedin.com/company/gepower
3	1025	Hewlett Packard Enterprise		US	https://www.linkedin.com/company/hewlett-packa...
4	1028		Oracle	US	https://www.linkedin.com/company/oracle
5	1033		Accenture	IE	https://www.linkedin.com/company/accenture
6	1038		Deloitte	OO	https://www.linkedin.com/company/deloitte
7	1043		Siemens	DE	https://www.linkedin.com/company/siemens
8	8296	Aerojet Rocketdyne		US	https://www.linkedin.com/company/aerojet-rocke...
9	1044		PwC	GB	https://www.linkedin.com/company/pwc

MongoDB

Within MongoDB, there are two important collections: People and Companies

An example Person record:

```
_id: ObjectId('656cd1dcfd7ade15c982e2d6')
public_identifier: "a-arone"
full_name: "Aaron Jones"
▼ experiences: Array (2)
  ▼ 0: Object
    ▶ starts_at: Object
    ends_at: null
    company: "Dish Network"
    company_linkedin_profile_url: "https://www.linkedin.com/company/dish-network"
    title: "Retention Specialist"
    description: null
    location: "Roseland NJ"
    logo_url: "https://media-exp1.licdn.com/dms/image/C560BAQE0KxtLULI5lQ/company-log..."
  ▶ 1: Object
▼ education: Array (1)
  ▼ 0: Object
    ▶ starts_at: Object
    ▶ ends_at: Object
    field_of_study: "Marketing"
    degree_name: "Bachelor of Science - BS"
    school: "Cheyney University of Pennsylvania"
    school_linkedin_profile_url: "https://www.linkedin.com/school/cheyney-university-of-pennsylvania/"
    description: null
    logo_url: "https://media-exp1.licdn.com/dms/image/C510BAQFqhBmXswrzww/company-log..."
▶ languages: Array (empty)
```

As can be seen, our LinkedIn user Aaron Jones currently works at Dish Network. He has prior employment experiences, an education at the institution Cheyney University of Pennsylvania, and other, less important, features.

An example Company record:

```

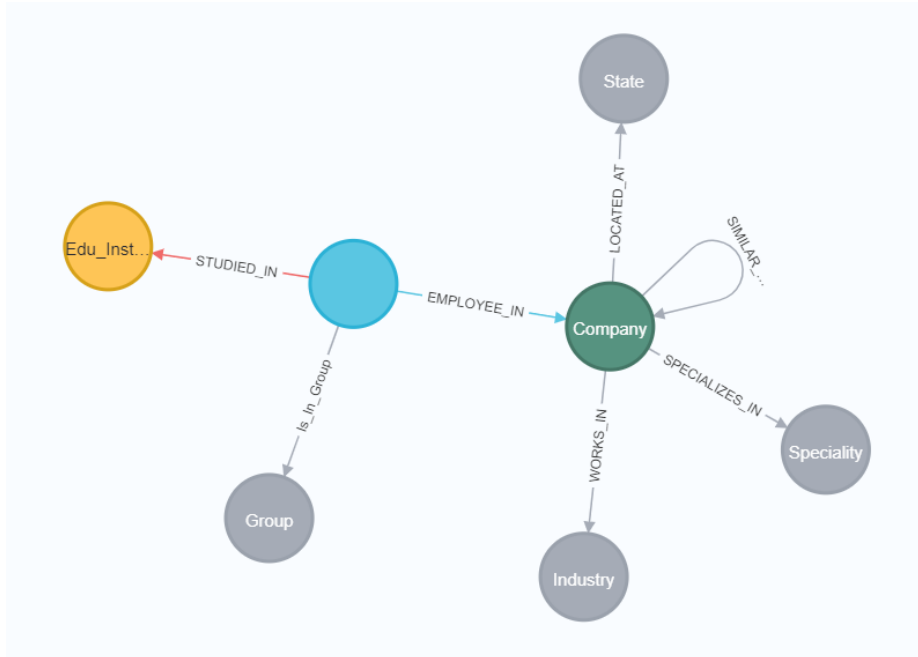
_id: ObjectId('656cd1e1fd7ade15c983099b')
linkedin_internal_id: 3057029
name: "(twenty)2 films"
website: "http://www.twenty2films.com"
industry: "Motion Pictures and Film"
▼ hq: Object
  country: "US"
  city: "Brooklyn"
  postal_code: "11201"
  line_1: "155 Water Street #2-28C"
  is_hq: true
  state: "NY"
▼ specialities: Array (3)
  0: "Film Production"
  1: "Development"
  2: "Post Production"
▼ locations: Array (2)
  ▼ 0: Object
    country: "US"
    city: "Brooklyn"
    postal_code: "11201"
    line_1: "155 Water Street #2-28C"
    is_hq: true
    state: "NY"
  ▶ 1: Object
▼ similar_companies: Array (10)
  ▼ 0: Object
    name: "HouseTwelve Media"
    link: "https://www.linkedin.com/company/housetwelve-media"
    industry: "Media Production"
    location: "Rochester, NY"
  ▶ 1: Object
  ▶ 2: Object
  ▶ 3: Object
  ▶ 4: Object
  ▶ 5: Object
  ▶ 6: Object
  ▶ 7: Object
  ▶ 8: Object
  ▶ 9: Object
▼ updates: Array (empty)

```

This company twenty2 films is in the industry of Motion Pictures and Film, has a headquarter in the US, specializes in Film Production, Development, and Post Production, and has similar companies like HouseTwelve Media.

Neo4j

The general schema of our data in Neo4j is as follows



Overview

Node labels

* (7) Industry (1) Group (1) Company (1)
 Speciality (1) State (1) Edu_Institution (1)
 People (1)

Relationship Types

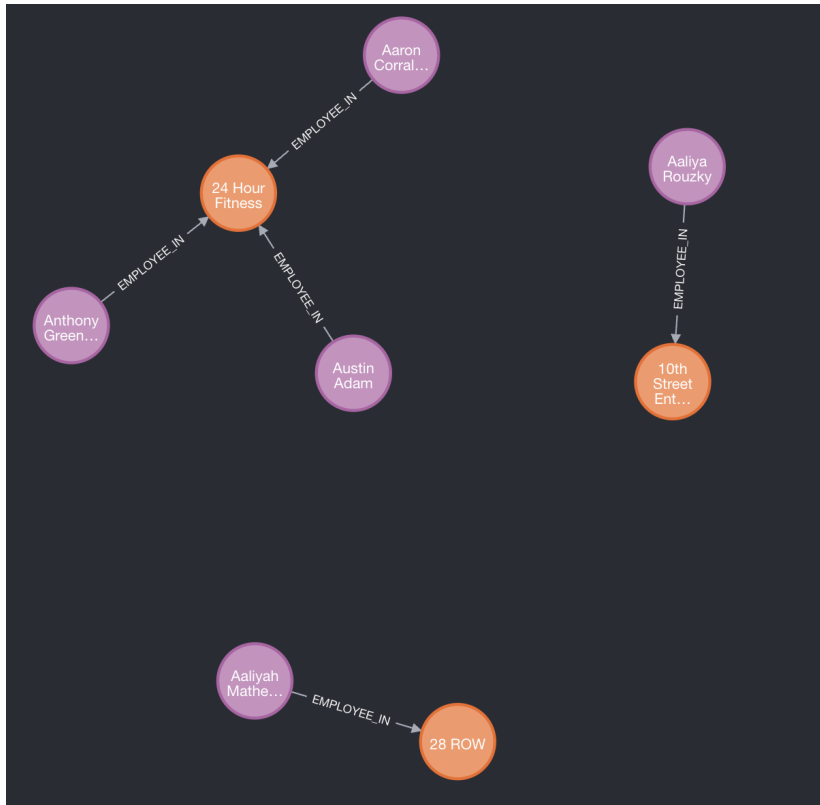
* (7) SIMILAR_TO (1) STUDIED_IN (1)
 LOCATED_AT (1) WORKS_IN (1)
 EMPLOYEE_IN (1) SPECIALIZES_IN (1)
 Is_In_Group (1)

Displaying 7 nodes, 7 relationships.

As with the trend, this schema shows a reformatting of the data seen before. People are nodes that have studied at Education Institution nodes. People are employed at Company nodes, who specialize in Specialty nodes, and work in Industry nodes. Companies can be similar to other Company nodes.

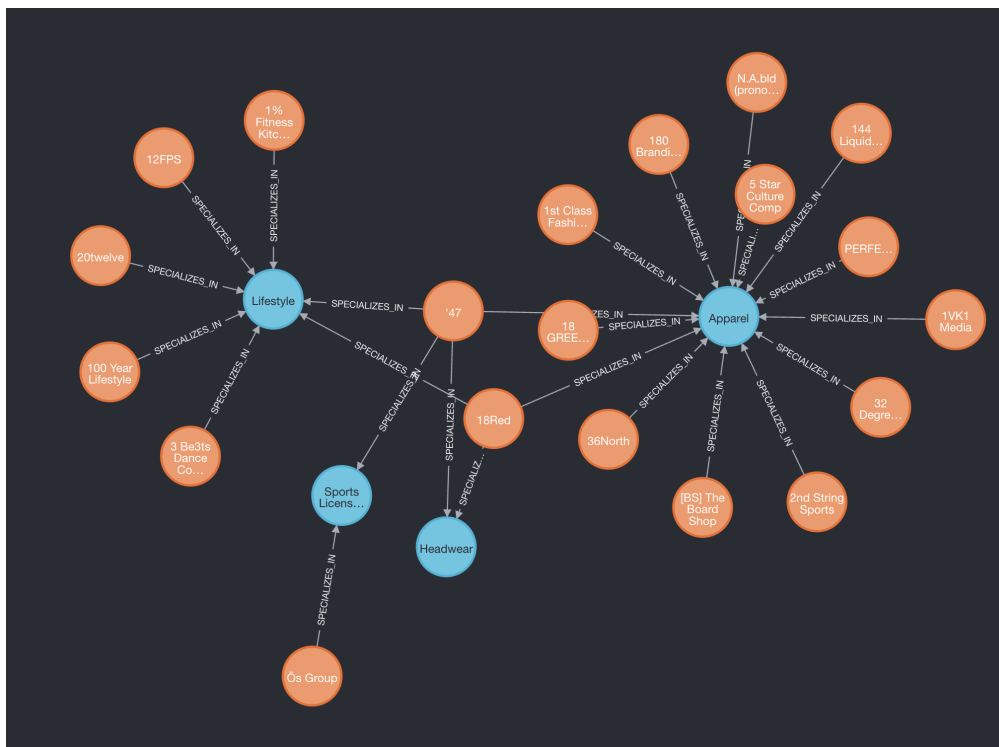
To emphasize two important relationships seen above, we visualize examples of ...

People Employed by Companies



Here we can see several people employed at 24 Hour Fitness.

Companies Specializes in Industry



Many companies specialize in Apparel, and the overlap of companies the specialize in Apparel and Lifestyle is evident.

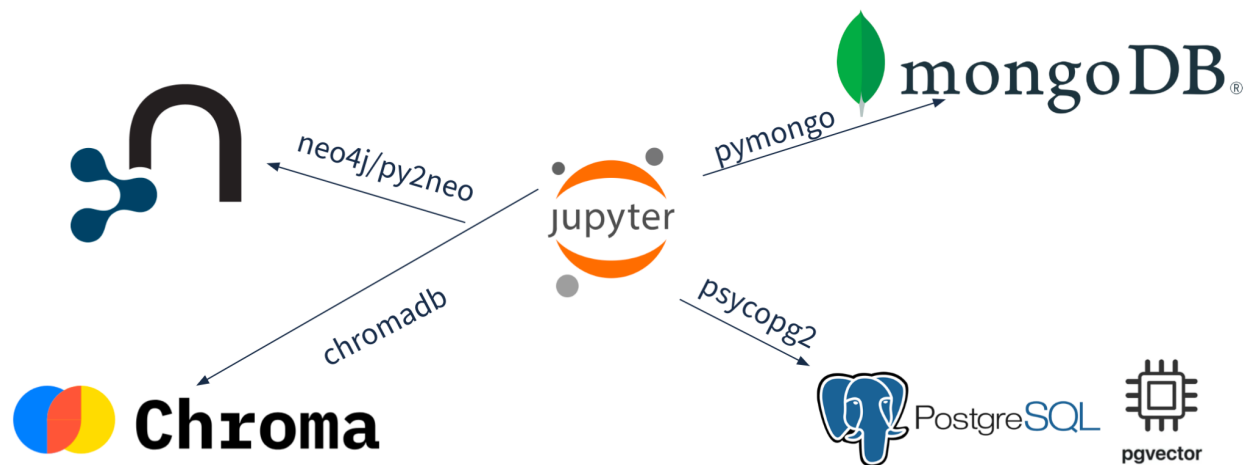
ChromaDB

ChromaDB is a vector database we experimented with (we also utilized vector embeddings in PostGres using pgVector).

We vectorized two key pieces of data: Job Posting descriptions and User descriptions.

We experimented with multiple methods of embedding data: OpenAI Embedding Function, multiple public HuggingFace transformers, and upon the consensus of reddit research decided upon utilizing [paraphrase-MiniLM-L6-v2](#).








How they all connect



All analysis was done within Jupyter notebooks. We utilized chromadb api for ChromaDB, pymongo api for MongoDB, neo4j and py2neo apis for Neo4j, and psycpg2 api for PostGres and its associated pgVector component.

Infrastructure Set up

All the databases are in docker.

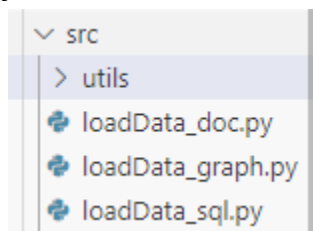
<input type="checkbox"/>	 linkedinunleas	Running (3/3)	0.88%	2 days ago
<input type="checkbox"/>	 neo4j-1 00118f134c7b neo4j:3.5	Running	0.53% 7474:7474  Show all ports (2)	2 days ago
<input type="checkbox"/>	 mongodb-1 cd1cac2bda6c mongo	Running	0.35% 27017:27017 	2 days ago
<input type="checkbox"/>	 postgres-1 0c5a24ce2069 ankane/pgvector	Running	0% 5439:5432 	2 days ago

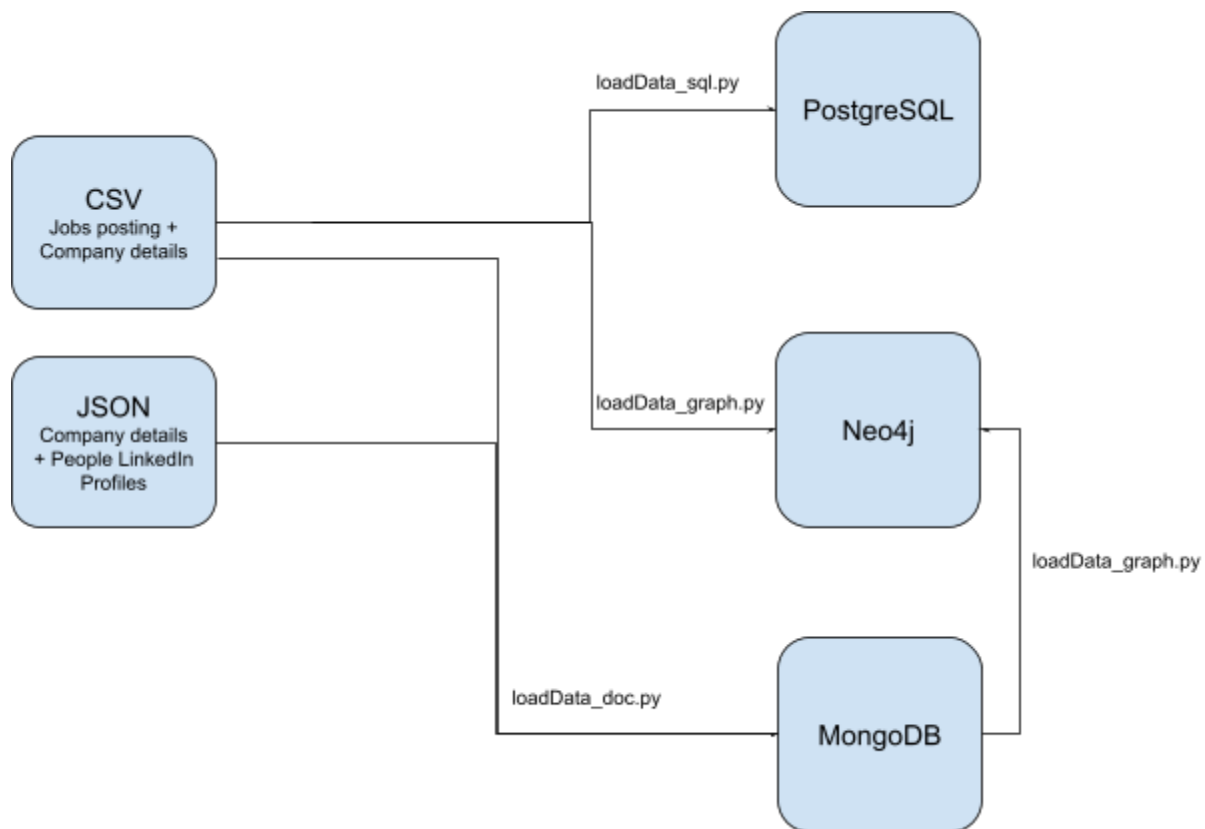
To set up the entire infrastructure, we just need to run command
`docker-compose up -d`
 The 3 database containers in docker must be running to proceed.

Data Loading: SQL+Mongo+Neo4j

Extract the zipped data into `./data` directory of the project. There should be a `./data/for_neo4j` directory which is used to share data with Neo4j container.

Our raw data is in csv and txt/json formats. We created individual scripts to load from csv and json into the databases.





If you use our docker-compose to set up the infrastructure, the DB credentials and ports will be the same so you can just run these commands in this order to populate the various tables.

1. `loadData_sql.py`: This file will create a schema using the commands in `models/postgresql_tables.sql` file. Then it will fetch csv data from `./data/` directory into these tables.
2. `loadData_doc.py`: This will fetch data from csv and txt files in `./data/` directory and load into `People` and `Company` collections in MongoDB
3. `loadData_graph.py`: This uses the csv files and MongoDB as source. First, it runs some Mongo queries and stores results in json files in `./data/for_neo4j` directory. This is shared with the neo4j container. Then it runs the commands in neo4j to load data from these csv and json files.

The design of these scripts is such that it is very easy to load additional sources of data.

Note: There are credentials near the top of these scripts which you might need to change if you are facing authentication issues while connecting to DBs.

Data Loading: ChromaDB

We store two kinds of embeddings in ChromaDB:

1. Description of a job posting
2. Summary of people's profiles

The model that we used for embeddings is called “paraphrase-MiniLM-L6-v2”. paraphrase-MiniLM-L6-v2 is a sentence-transformers model: It maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search.

One of the challenges we faced in this project was choosing the right model for creating vector embeddings. After trying out multiple models like all-MiniLM-L6-v2, OpenAI Embedding model and paraphrase-MiniLM-L6-v2, we found out that OpenAI Embedding model gives the best results closely followed by paraphrase-MiniLM-L6-v2. But since OpenAI Embedding model is paid, we chose to go ahead with paraphrase-MiniLM-L6-v2 and didn't observe much loss in performance.

1. Description of a job posting.

We make the text to feed into the embeddings function by concatenating the job title and the actual description. We do this because both of these columns carry information to capture in the embeddings. We make a composite id by combining the job id and the company id as we use both of them for PostgreSQL queries later.

2. Summary of people's profiles.

We use the summary column as text to feed to the embeddings function and we use person name as the id.

Design Considerations

Data Redundancy

We initially stored most data in SQL and MongoDB and stored only the necessary data in neo4j to create network. But to run a complete query, we had to fetch data from multiple DBs resulting in more communication cost.

We realized that if we can store more data in neo4j relations and nodes, we can minimize the need to join with additional data from other sources. So we achieve simpler queries and less communication cost by trading it for increased data redundancy i.e. same data in multiple sources.

PGVector vs ChromaDB

Currently, vector databases are in-demand due to the advancements in LLM. But only a few may stand the test of time. We considered PGVector and ChromaDB for our purposes as both are open-source. Here is a comparison of these

PGVector	ChromDB
Simpler Setup if you already have PostgreSQL. It is an extension in PostgreSQL that adds vector functionality	DB has to be installed and configured separately
If companies already have data in PostgreSQL, they can just add this extension and make minor changes in processes	More cost of migration of processes
No separate maintenance cost and enjoy the proven capabilities of PostgreSQL	Extra maintenance cost due to managing a separate DB and relatively new software
Need to use langchain embedding functions to encode the text/image to vectors. We tried HuggingFace 'Instructor Embeddings' with pgvector.	Comes with its built-in embedding functions
	Open source, free, good for quick experimentation

Task 0: Finding a Domain

If you're young and inexperienced, there's a good chance you do not know what you want to be doing. Maybe any job is the right job for you. This tool can help you find companies and domain areas that are hiring the most to increase chances of employment!

How it Works

Source file: ./tasks/t0.ipynb

Step 1

Using PostGresSql, we query to find job openings per company and divide that number by the number of employees at that company.

1.1 Write postgres query as a string

```
query = '''
with no_of_postings as (
select distinct company_id, count(*) as no_p
from job_postings
where company_id is not null
group by company_id
order by no_p desc
LIMIT 10),
new_emp_count as (
select max(employee_count) as emp_count, company_id
from company_counts
group by company_id
)

select A.company_id, C.name, CAST(B.no_p as float) * 100 / A.emp_count as ratio
from new_emp_count A
inner join no_of_postings B inner join companies C on B.company_id = C.company_id
on A.company_id = B.company_id
order by CAST(B.no_p as float) * 100 / A.emp_count desc
'''
```

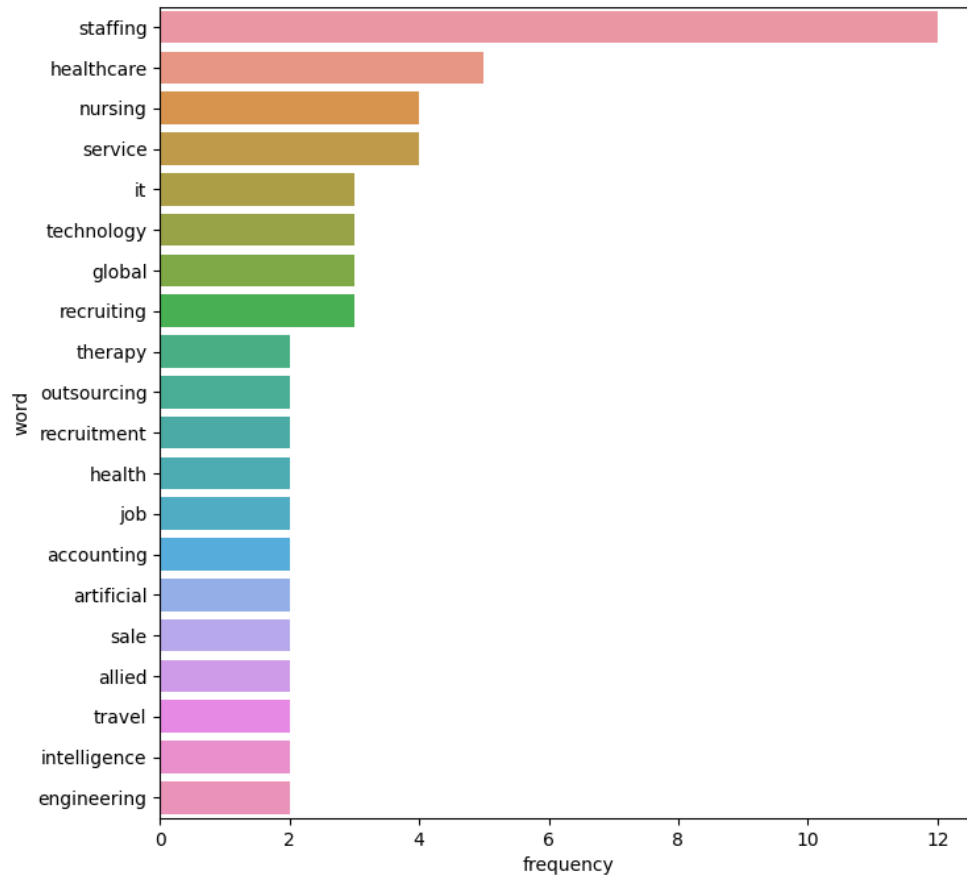
Step 2

Using MongoDB, retrieve Top 10 Companies with the best ratio.

```
_id: ObjectId('6568e3be5db40a39d8023a51')
index: 13
linkedin_internal_id: 1103
name: "Verizon"
description: "You want more out of a career. A place to share your ideas freely – ev..."
▸ specialties: Array (4)
▸ locations: Object
```

Step 3

We pull the specialties attribute from the company objects and visualize the frequency of specialties occurring in these hiring companies.



One can see that healthcare, nursing, IT, and technology are important domains. Focusing skill development in one of these domains would likely increase chances of employment in the future.

Task 1: Finding a Mentor

Source file: ./tasks/t1.ipynb

Now that you have a domain and you're becoming more familiar with the field, you want to take the next step in getting acquainted with the status quo of the industry. However, you don't know where to get started! **LinkedIn Unleashed** can help you find a mentor to guide you through the ins-and-outs of the industry.

How it Works

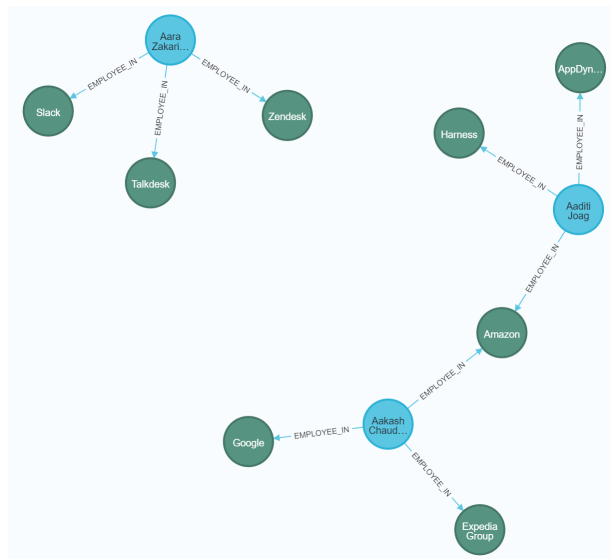
Step 1

User inputs the target domain into the system, e.g. 'Software'

Step 2

Using Neo4j, find the Top 3 people who have the most experience with companies in that industry.

```
domain = 'Software'
query = f'''
match (c:Company)-[:WORKS_IN]->(i:Industry)
where i.name =~ '(?i).*{domain}.*'
match (p:People)-[e:EMPLOYEE_IN]->(c)
return p, collect(e) as exp, collect(c) as comps
order by size(exp) desc limit 3;
'''
```



Step 3

Using Postgres, output a table containing these individuals information to allow for personalized LinkedIn connection messaging.

```
# From PostgreSQL
sql_query = f'''
select full_name, occupation, headline, summary, city, country from uspeople_pc where public_identifier in ({','.join(public_identifiers)});
'''

pgsql = sql_utils.PostgreSQL_Connector(creds['sql_creds'])
people_details = pgsql.runQuery(sql_query)
people_details
```

	full_name	occupation	headline	summary	city	country
0	Aaditi Joag	Software Development Engineer II at Amazon	Software Development Engineer at Amazon	None	San Francisco Bay Area	US
1	Aakash Chaudhary	Software Development Engineer 3 at Expedia Group	SDE 3 at Expedia Group Former SDE at Amazon ...	Expert Software Development professional with ...	Seattle	US
2	Aara Zakariaei	Healthcare Account Executive at Slack	Account Executive at Slack NASM Health Coach	None	Denver	US

Task 2: Researching your Dream Position

Source file: ./tasks/t2_vec1.ipynb

You've listened to your mentor about what the industry is like. You know what work you want to be doing and which company you want to be doing it for. Unfortunately, you're not sure if you really have all the skills that this position requires. This tool will find out the skills truly required for the position of your dreams!

How it Works

Step 1

Input type of role for a particular company, e.g. something data related at IBM.

Step 2

Using Neo4j, search for employees in that domain at that company.

```
domain = 'data'
company = 'IBM'
f'''
match (c:Company)-[r:EMPLOYEE_IN]-(p)
where r.title =~ '(?i).*{domain}.*' and c.name='{company}'
return *;
'''
```



Step 3

Still using Neo4j, retrieve the descriptions from the users.

```

description_lst
✓ 0.0s

['Designed and implemented a new Watson Cloud offering that uses speech recognition to automatically caption video content using
'07/2019 - Current: Global Sales Incentives - E2E Data Integration Expert (Global NY - Remote MTL)\n- Data integration expert
'',
 '* Develop Watson health cognitive framework for image analytics to support various advisors such as breast advisor, heart advisor
'Contributed in co-creation projects to develop cognitive solutions and services for the Cognitive Enterprise Data Platform (C
'',
'Database development and Administration for all Microsoft SQL Server instances owned by Ogilvy Commonwealth World Wide.\nDevelop

```

Step 4A

Using these descriptions, we can use an LLM to ask the technologies being utilized within this role.

```

CONNECTION_STRING = f"postgresql+psycopg2://{creds['sql_creds']['user']}:{creds['sql_creds']['password']}@{creds['sql_creds']['host']}:{creds['sql_creds']['port']}/{creds['sql_creds']['db']}"
COLLECTION_NAME = 'ds_descri'

vectorstore = get_vectorstores(desc_chunks, COLLECTION_NAME, CONNECTION_STRING)
conversation = get_conversation_chain(vectorstore)

load INSTRUCTOR_Transformer
max_seq_length 512
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, and simply means that the 'legacy' (previous) behavior will be used.
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

user_question = "List the technologies used in these roles?"
response = conversation({'question': user_question})

response['answer']

'SQL, JS and DataStage as well as the development of new web based tools in Bluemix using Python'

```

Step 4B

Alternatively, we can utilize a statistical NLP approach to analyze the descriptions and determine important skills present within them.

```
nlp = spacy.load('en_core_web_sm')
text = ';'.join(description_lst)
doc = nlp(text)
doc.ents[:10]
```

```
(Watson Cloud,
Node.js.;07/2019,
Incentives,
SQL,
JS,
DataStage,
Bluemix,
Office - Transformation & Operations - Senior Data Analyst,
Global - Remote,
IBM)
```

Task 3: Finding a Connection Path to your Dream Job

Source file: ./tasks/t3.ipynb

You're confident you're qualified for the position of your dreams. You're ready to apply, but you want the best chances possible. In this day and age, a referral can be the difference maker. How do you find the right person to get connected with? People typically ignore random 4th degree connection requests. This tool finds the most sensible connection and the steps that should be taken to connect with them.

How it Works

Step 1

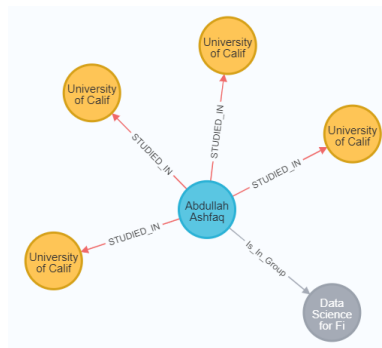
Create your own profile and insert it into the Neo4j graph.

```
name = 'Abdullah Ashfaq'
public_identifier = 'aashfaq11'
educational_institution = '.*University of California.*San Diego.*'
groups = ['Data Science for Finance and Economics']
target_company = 'Google'

# Starting points are my groups and universities
starting_groups = []
starting_uni = ['.*University of California.*San Diego.*']
```

```
# Inserting myself in the network
insert_query = f'''
Merge (a:People {{full_name: "{name}", public_identifier: "{public_identifier}"}})
'''
neoCon.runQuery(insert_query)

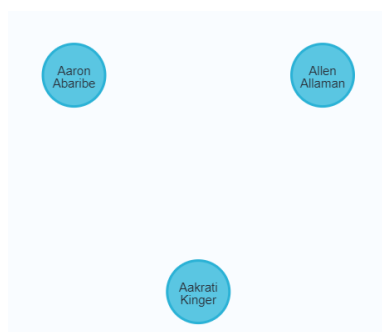
insert_query = f'''
Match (a:People {{full_name: "{name}", public_identifier: "{public_identifier}"}}), (u:Edu_Institution), (g:Group)
where u.name =~ '(?i){educational_institution}' and g.name = '{groups[0]}'
merge (a)-[:STUDIED_IN]->(u)
merge (g)-[:Is_In_Group]->(a)
'''
print(f"Running query: \n {insert_query}")
neoCon.runQuery(insert_query)
```



Step 2

Find the shortest path in Neo4j between the new profile and employees at the dream job, say Google. Of course, we want to skip connections that involve company nodes.

```
query = f'''
MATCH (end:People)-[:EMPLOYEE_IN]->(c:Company)
WHERE c.name =~ '{target_company}'
WITH collect(end) as endPersons
Match (start:People {{full_name: "{name}", public_identifier: "{public_identifier}"}})
CALL apoc.path.subgraphNodes(start, {{relationshipFilter:'', endNodes:endPersons, limit:3, labelFilter:'-Company'}}) YIELD node
RETURN node
'''
print(f"Running query: \n {query}")
values = neoCon.runQuery(query)
```



Step 3

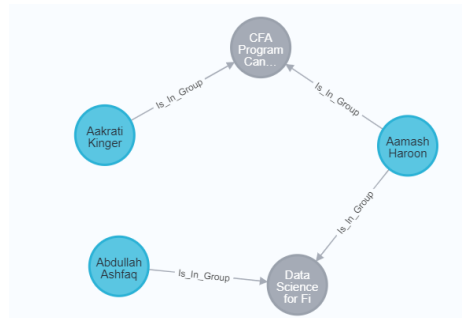
For each of the returned employees, find the most sensible, shortest path to connecting with them. Choose the most appropriate and start connecting!

```

all_paths = []
for p in pub_identif:
    query = f'''
    Match p=shortestPath((a:People {{full_name: "{name}", public_identifier: "{public_identifier}"}})-[*]-(p1:People {{public_identifier:'{p}'}}))
    return p;
    '''
    print(f"Running query: \n {query}")
    values = neoCon.runQuery(query)
    all_paths.append(values)

all_paths

```



Task 4: Know companies that hire you!

Source file: ./tasks/t4_vec2.ipynb

If you are at a stage in your career where you want to apply for jobs. You have some experience in and know what kind of work you want to do and have a profile summary statement prepared. This tool will help you find relevant job openings, perform some analysis on them and display a network of companies that you should keep an eye on for potential job openings in the future.

How it works

Step 1

Find 25 job postings most similar to your profile using similarity search in chromaDB using the following query.

```

query_string = """Data Science grad with 2 years of
experience in the entire data science life cycle.
Have multiple internships and projects in computer
vision and natural language processing.
Proficient with Python, R and querying
languages like SQL."""

```

```

results = collection.query(
    query_texts=[query_string],
    n_results=25
)
|
results

```

Results of Step 1:

Ids of corresponding jobs:

```

{'ids': [['3699097574:2012779.0',
'3699087065:1403.0',
'3749360141:2203697.0',
'3697390531:1403.0',
'3697387964:1403.0',
'3757459741:53864.0',
'3757725777:157356.0',
'3757471229:3749364.0',
'3699077628:1441.0',
'3697388794:1403.0',
'3693048844:8117.0',
'3697391430:1403.0',
'3757756157:5140.0',
'3757496203:2385278.0',
'3756116362:77869403.0',
'3757725778:157356.0',
'3757727698:157356.0',
'3757454394:1199482.0',
'3693586591:45346.0',

```

Their distances from the input query string:

```

'distances': [[14.12105655670166,
14.57985782623291,
15.007060050964355,
15.131120681762695,
15.138842582702637,
15.30224609375,
15.412276268005371,
15.722932815551758,
15.806909561157227,
15.818198204040527,
15.830878257751465,
15.923686027526855,
15.930133819580078,
15.943193435668945,
16.098758697509766,
16.104883193969727,
16.104883193969727,
16.212535858154297,

```

Example of a document:

```
'documents': [['Sr. Data Engineer Title: Sr. Data EngineerLocation: Austin, TxDuration: 6 MonthsPay Rate: 65HR 70HR on W2 Re
quirements:Bachelors degree in Computer Science or related field plus 4 years of relevant work experience or a Masters degree
plus 2 years of relevant work experience or a PhD plus 01 years of relevant experienceIn lieu of a degree, qualified candidat
es would require 8 years of relevant professional experienceExcellent with SQLExcellent programming skills in Python and idea
lly demonstrating an aptitude via experience with multiple languagesExcellent understanding of patterns for data ingest into
data warehouse, ingest, cleansing, standardizing, etc., in addition to different data structures like normalized, denormalize
d, starExcellent experience supporting Snowflake Data Warehouse, including Snowpipe including streams, tasks, transformation
s, views, dynamic tables. This should include advanced skills in ensuring efficient utilization of Snowflake compute and the
ability to optimize workloads and warehouseBroad experience with Cloud PaaS capabilities, ideally AWS CloudWatch, Lambda, Ste
p Functions, SNS, SQS, DynamoDB, etc.Experience utilizing reporting and data insights toolsExperience in supporting analytics t
eams data needs, in addition to customer and business reporting. What you'll be doing:Provides Data Engineering support for ar
eas such as Finance, Sales, Business Intelligence, Product Development and/or other business usersWorks with data consumers an
d Project Managers to determine logical and physical database designs for analytics modelsCreates and maintains optimal data
pipeline architecturesEnsures architectures are aligned with and support business requirements;Ensures that required data is
available, can be trusted and is readily accessible by those who need itInfluences the data infrastructure roadmapIdentifies,
```

Step 2

Location Analysis using PostgreSQL

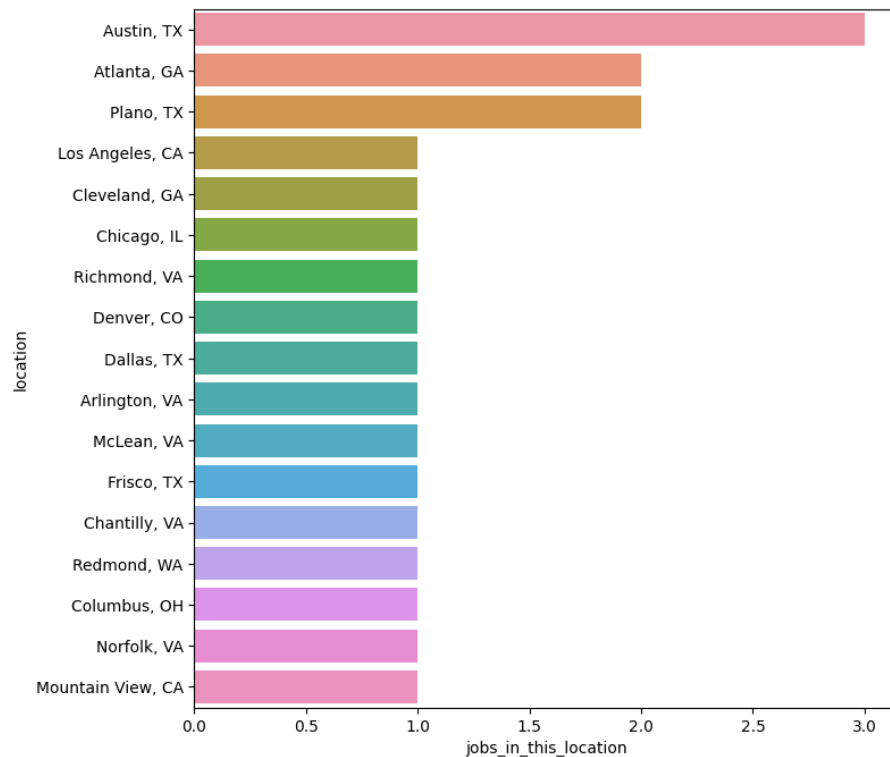
We use the job ids obtained from the vector query to obtain other details about the job using PostgreSQL and perform location(US state) analysis.

```
location_query = """
with main_query as (select job_id, c.name, title, formatted_work_type as work_type, location
from job_postings t1, unnest(%) j_id, companies c
where job_id::text = j_id::text
and c.company_id = t1.company_id)
select location, count(*) as jobs_in_this_location
from main_query
where location != 'United States'
group by location
order by count(*) desc
"""
```

```
locations = runQuery(location_query, (jobs,))
```

Here, jobs represent the array of job ids obtained from the vector query in the first step.

We visualize the result of this query as a frequency table.



Step 3

Job Title Analysis

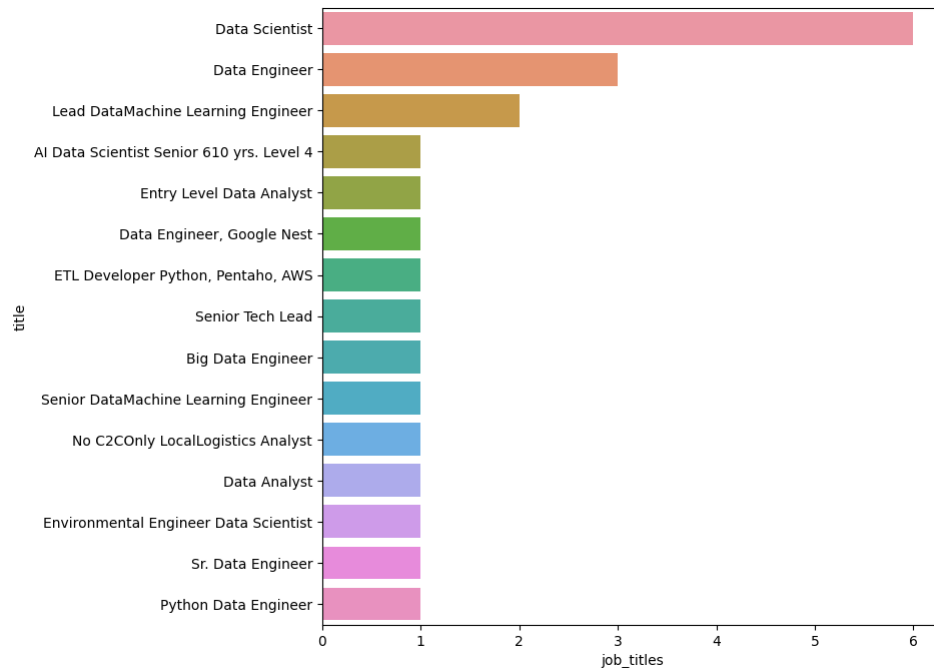
We use the job ids obtained from the vector query to obtain other details about the job using postgresSQL and perform job-title analysis.

```
titles_query = """
with main_query as (select job_id, c.name, title, formatted_work_type as work_type, location
from job_postings t1, unnest(%s) j_id, companies c
where job_id::text = j_id::text
and c.company_id = t1.company_id)
select title, count(*) as job_titles
from main_query
group by title
order by count(*) desc
"""
```

```
titles = runQuery(titles_query, (jobs,))
```

Here, jobs represent the array of job ids obtained from the vector query in the first step.

We visualize the result of this query as a frequency table.



Step 4

Network Graph of similar companies for potential applications

All the above 3 steps were conducted on current job postings. This step is conducted to make a network of similar companies to keep an eye on.

Cypher Query:

```
graph_query = """match p=(n1)-[:SIMILAR_TO]-(d)
where n1.name IN $array
return p"""
```

```
result = graph.run(graph_query, parameters={"array": company_names})
```

Results of the cypher query:

p

```
(Thoughtworks)<-[:SIMILAR_TO {}]-(3IS Solutions Inc)
(Virtusa)<-[:SIMILAR_TO {}]-(Auxenta)
(Booz Allen Hamilton)<-[:SIMILAR_TO {}]-(1901 Group)
```

p

```
(Virtusa)<-[:SIMILAR_TO {}]-(Auxenta)
(Booz Allen Hamilton)<-[:SIMILAR_TO {}]-(1901 Group)
(Google)<-[:SIMILAR_TO {}]-(1863 Ventures)
```

p

```
(Booz Allen Hamilton)<-[:SIMILAR_TO {}]-(1901 Group)
(Google)<-[:SIMILAR_TO {}]-(1863 Ventures)
(Google)<-[:SIMILAR_TO {}]-(2mrw)
```

p

```
(Google)<-[:SIMILAR_TO {}]-(1863 Ventures)
(Google)<-[:SIMILAR_TO {}]-(2mrw)
(Google)<-[:SIMILAR_TO {}]-(3 Little Birds)
```

p

```
(Google)<-[:SIMILAR_TO {}]-(2mrw)
(Google)<-[:SIMILAR_TO {}]-(3 Little Birds)
(Flexton Inc.)<-[:SIMILAR_TO {}]-(4 Consulting Inc.)
```

p

```
(Google)<-[:SIMILAR_TO {}]-(3 Little Birds)
(Flexton Inc.)<-[:SIMILAR_TO {}]-(4 Consulting Inc.)
(The Intersect Group)<-[:SIMILAR_TO {}]-(6 Degrees Group)
```

Graph Visualization:



Task 5: Connect with similar people and keep an eye out for referrals.

Source file: ./tasks/t4_vec2.ipynb

If you are at a stage in your career where you want to apply for a new job or want to switch jobs, you need to know what companies people who have similar interests as you work in. After connecting with such people, you can ask for a referral from them and make a network of companies that you can get referred to. We also present the top-skills required for such jobs based on the skills of similar people working in the industry.

```
query_string = """Data Science grad with 2 years of
experience in the entire data science life cycle.
Have multiple internships and projects in computer
vision and natural language processing.
Proficient with Python, R and querying
languages like SQL."""
```

How it Works

Step 1

Find others with similar profile using ChromaDB vector search

```

results = collection.query(
    query_texts=[query_string],
    n_results=50
)

results

```

Results:

Names:

```

{'ids': [['Aadit Vyas',
'Aakriti Gupta',
'Abdullah Aburomeh',
'Adam R.',
'Azaan Barlas',
'Aabhashree Lamichhane',
'Aalap M',
'Aaditya Bhat',
'Aarohi Mehta',
'Aakanksha Patil',
'Aanchal Gosain',
'Aakanksha Bharde',
'Aabir Abubaker Kar',
'Aamir Goriawala',
'Aaditya Mohapatra',
'Aaron Carney',
'Ahmed Jaafar',
'Aadithya Viswanath Ramasubramaniam',
'Aafaz Ilahi',

```

Distances:

```

'distances': [[11.632522583007812,
11.83469009399414,
12.008356094360352,
12.191132545471191,
12.510778427124023,
12.657659530639648,
13.151001930236816,
13.15983772277832,
13.318685531616211,
13.436904907226562,
13.474906921386719,
13.501951217651367,
13.68099594116211,
13.831799507141113,
13.864422798156738,
13.871238708496094,
14.15813159942627,
14.240290641784668,
14.483783721923828,

```

Top 3 matched of profile statement:


```
(Amazon)<-[:SIMILAR_TO {}]-{1STWEST Background Due Diligence LLC}
(LexisNexis)<-[:SIMILAR_TO {}]-{21Development - Local SEO}
(Amazon)<-[:SIMILAR_TO {}]-{1369 Coffeehouse}
```

```
(LexisNexis)<-[:SIMILAR_TO {}]-(21Development - Local SEO)
(Amazon)<-[:SIMILAR_TO {}]-(1369 Coffeehouse)
(Amazon)<-[:SIMILAR_TO {}]-(180 South Solar)
```

```
(Amazon)<-[:SIMILAR_TO {}]-(1369 Coffeehouse)
(Amazon)<-[:SIMILAR_TO {}]-(180 South Solar)
(Amazon)<-[:SIMILAR_TO {}]-(24 Seven Talent)
```

[illegible]

A clear limitation of this work is the size of the data we have present. It is a sample of the massive amount of data present on LinkedIn, and thus the results may not be as exemplary as they could be. The sparseness of connections between people and companies this sample offers also likely means bugs that may arise with less sparsity.

31

access to data like connections between people could improve upon some of the tasks offered (like Task 3) as lower-degree connections would be more frequent.

LinkedIn API costs \$50/month to get data for 500 profiles and \$500/month to get data for 10,000 people. If we had access to such API, we can create an end-to-end product by creating an automated ingestion pipeline that processes batches as new data arrives.