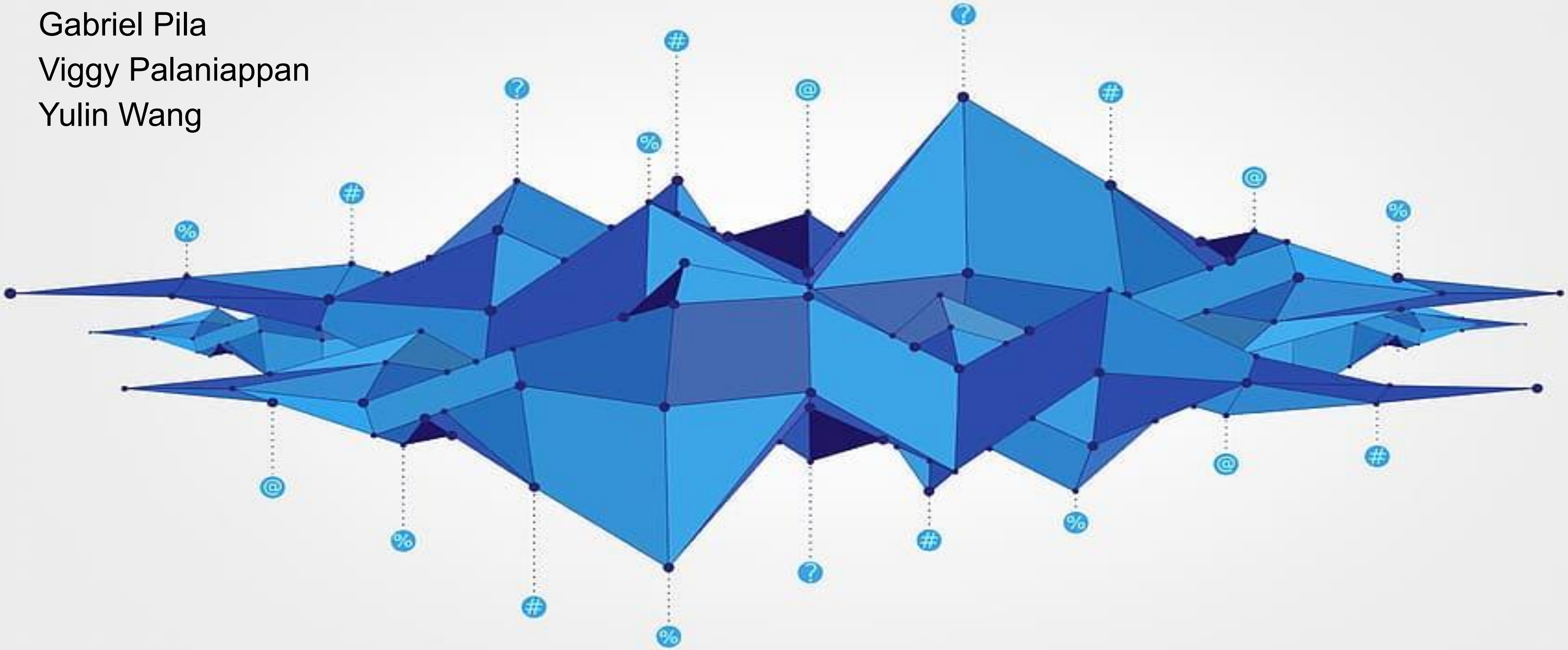
TEAM 8

Gabriel Pila
Viggy Palaniappan
Yulin Wang



Introduction

FOUNDERS FUND

intel Capital

SEQUOIA

andreessen. horowitz

khosla ventures

It's time to build

Context: Imagine we are a venture capital firm and have a large sum of money to invest somewhere…

We need to build an application that can help us decide what technology to invest money in and what resources to hire for it.

(Alternative Context): Imagine we are leaking money

We need to build an application that can help us decide what technology to stay invested in and what resources to keep.

Data Sources & Methodology

We made use of 5 data sources:

1) ***(Graph data)***
   Neo4J technologies data (provided)
2) ***(Graph data)***
   ACM Digital Library
3) ***(Relational data)***
   Postgres patents and awards (provided)
4) ***(Relational data)***
   Papers from UCSD Dimensions Data Nexus Portal
5) ***(Semi Structured data)***
   Wikipedia API

Role, Contents

1) ***List of technologies, and companies that have expertise in them.***
2) ***A computing classification system which has keywords organized by field within computing***
3) ***List of patents and grant awards including source of awards, dates, citations, status.***
4) ***2022 paper publications from universities (decided to select publications from each of our undergrad universities)***
5) ***Wiki page html, page content, infobox***

Application WorkFlow to build a decision helper report on what technology to invest money and what resources to hire for it.

1. Pre-select Top 50 technologies
From the patents' neo4j dataset select X technologies to explore

Tech: Machine Learning, Quantum computing

2. Define a set of keywords related to the technologies
Extract keywords from ACM Library for each technology.

Keyword_universe: *[*
'quantum computing',
'efficient computing',
'natural language processing',
'explainability',
'computer vision'
]

Note: *If we include more keywords, our co-occurrence graph will be denser*

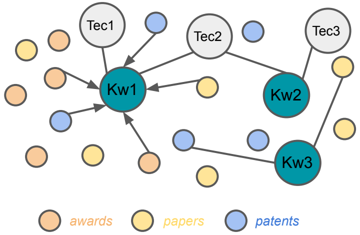
3. Detect keywords in from texts of relational datasets
Look for the existence of a list of keywords in the texts of the papers, awards, and patents datasets and add the found keywords as a new column

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

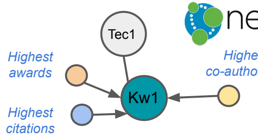
4. Load the ids, keywords and attributes to neo4j
Extract the ids, keywords and attributes from postgres and load them to neo4j (nodes and edges)

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

Co-occurrence graph connected by keywords



5. Find the paths that relate the keywords from the tech to the documents
Using the keywords as intermediate nodes, find which docs are connected and return the most important



Tech 1 → patent_id1, patent_id2, paper_id1, paper_id2, award_id1, award_id2

6. Get information about the selected document ids
From the chosen ids, get interesting data from the available datasets on postgres.

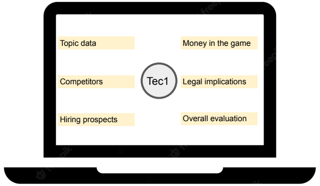
Select (important_data) From patentsdb Where id in (patent_id1, patent_id2)	Select (important_data) From awardsdb Where id in (award_id1, award_id2)	Select (important_data) From papersdb Where id in (paper_id1. paper_id2)

7. Get wikipedia documents from the selected ids
Look for summary or sidebar information about the companies, universities, technologies etc.

patent_id1 → 'General motors' (webpage)
Paper_id2 → 'UC San Diego' (webpage)

Companies
Universities
Topics

8. Build a report with the gathered data
Use information retrieved from 6 and 7 to build a report around a technology of interest.



Selecting technologies from neo4j

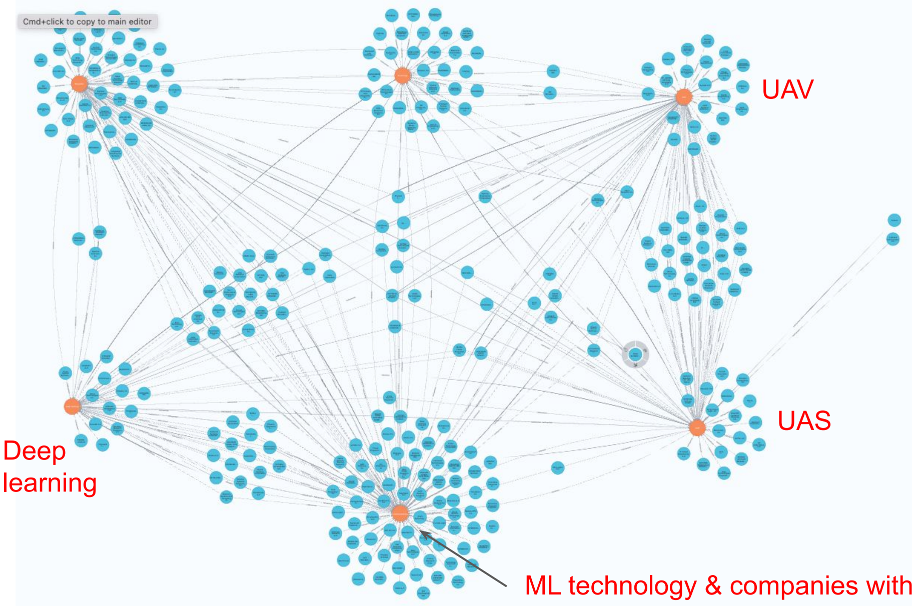
1. Pre-select Top 50 technologies
From the patents' neo4j dataset select X technologies to explore

neo4j

Tech: *Machine Learning, Quantum computing*

Visualize the data (from given neo4j database)

```
match p= (a:technology)-[r:hasExpertiseIn]-(b)
return p
limit 500
```



From python, query the technologies and filter Top 50

```
match p= (a:technology)-[r:hasExpertiseIn]-(b)
return a.name AS Technology, count(p) AS Degree
order by Degree desc
```

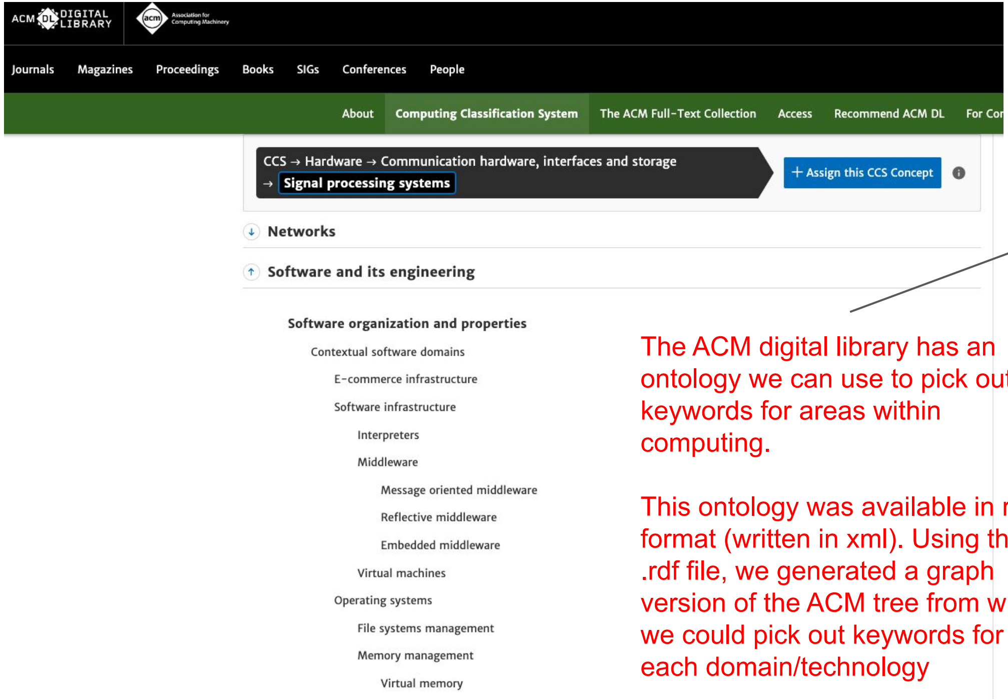
"Technology"	"Degree"
"machine learning"	128
"Simulation"	86
"UAV"	76
"Navigation"	75
"Infrared"	72
"Radar"	67
"artificial intelligence"	65
"UAS"	65
"Modeling"	64

Create a keyword universe for each technology

2. Define a set of keywords related to the technologies
Extract keywords from ACM Library for each technology.

Keyword_universe: *[*
'quantum computing',
'efficient computing',
'natural language processing',
'explainability',
'computer vision'
]

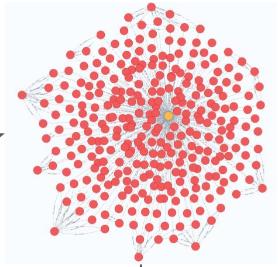
Note: *If we include more keywords, our co-occurrence graph will be denser*



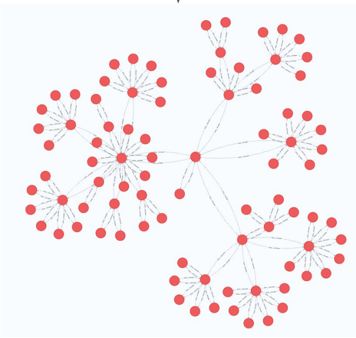
The ACM digital library has an ontology we can use to pick out keywords for areas within computing.

This ontology was available in rdf format (written in xml). Using this .rdf file, we generated a graph version of the ACM tree from which we could pick out keywords for each domain/technology

RDF upload into neo4j



Querying for 'Machine Learning'



Create a keyword universe for each technology

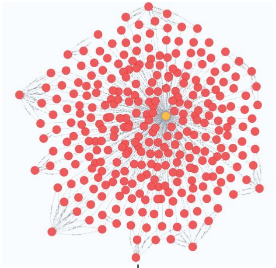
2. Define a set of keywords related to the technologies
Extract keywords from ACM Library for each technology.

ACM DL DIGITAL LIBRARY

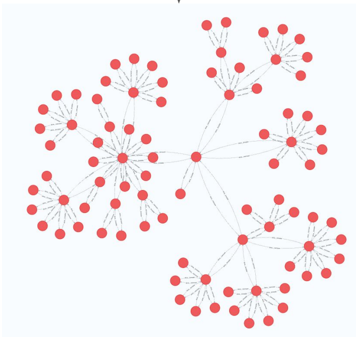
Keyword_universe: *[*
'quantum computing',
'efficient computing',
'natural language processing',
'explainability',
'computer vision'
]

Note: *If we include more keywords, our co-occurrence graph will be denser*

6-tier ontology means we search 6 levels deep



Querying for 'Machine Learning'



Querying for 'Machine Learning' using variable range relationship patterns

```
MATCH p=(j:skos__Concept {skos__prefLabel:'Machine learning' })-[r:skos__narrower*..6]->(b)
RETURN p
```

But we are interested in the keywords, not the path. This query flattens out the graph and gives all keywords under 'hardware'

```
MATCH p=(j:skos__Concept)-[r:skos__narrower*..6]->(b)
WHERE j.skos__prefLabel =~ '(?i)HARDWARE'
RETURN b.skos__prefLabel AS keyword
```

Used regexp with case insensitivity

"keyword"
"Hardware test"
"Fault models and test metrics"
"Memory test and repair"
"Hardware reliability screening"

Create a keyword universe for each technology

2. Define a set of keywords related to the technologies
Extract keywords from ACM Library for each technology.

ACM DL DIGITAL LIBRARY

Keyword_universe: *[*
'quantum computing',
'efficient computing',
'natural language processing',
'explainability',
'computer vision'
]

Note: *If we include more keywords, our co-occurrence graph will be denser*

```
#---(2) Search ACM Ontology for keywords-----------------------------------------

def neo4j_get_ontology_keywords(graphdb,technology):

    #Initialize keyword list and set initial keyword to be the name of the
    kw_list =[technology.lower()]

    #Initialize session in neo4j and run query
    #Query: retrieve up to 6 levels of the keyword ontology when looking up
    session = graphdb.session()
    q2="MATCH p=(j:skos__Concept)-[r:skos__narrower*..6]->(b) WHERE j.skos_
    nodes = session.run(q2)

    #Populate keyword list (all lower case)
    for node in nodes:
        keyword = node.value('keyword').lower()
        kw_list.append(keyword)

    #Close session and return tech list
    session.close()

    return kw_list

# #------------------------------------------------------------------------------
```

	Technology	Keywords
0	machine learning	[machine learning, learning paradigms, reinfor...
1	simulation	[simulation]
2	uav	[uav]
3	navigation	[navigation]
4	infrared	[infrared]
5	radar	[radar]
6	uas	[uas]
7	artificial intelligence	[artificial intelligence, planning and schedul...
8	modeling	[modeling]
9	guidance	[guidance]

Keyword Universe

E.g. this function in python to pick up keywords was called for every technology (~29,000) in the original neo4j dataset to find related ACM ontology keywords.

Where there were no matches, the technology itself was passed on as a keyword. Since we had a computing ontology, these are the ones that are 'useful'

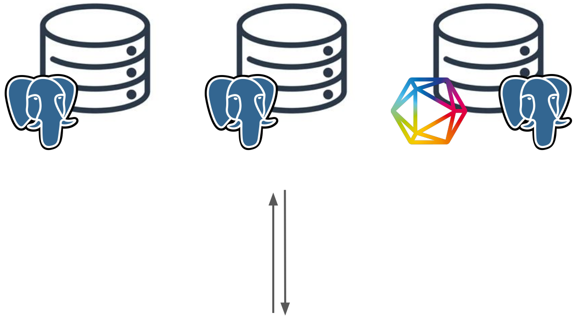
Matching keywords in datasets

3. Detect keywords in from texts of relational datasets
Look for the existence of a list of keywords in the texts of the papers, awards, and patents datasets and add the found keywords as a new column

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

Look for the existence of any keywords in the texts of the patents, awards, and papers datasets and add the found keywords as a new columns

Patents Awards Papers



	Technology	Keywords
0	machine learning	[machine learning, learning paradigms, reinfor...
1	simulation	[simulation]
2	uav	[uav]
3	navigation	[navigation]
4	infrared	[infrared]
5	radar	[radar]
6	uas	[uas]
7	artificial intelligence	[artificial intelligence, planning and schedul...
8	modeling	[modeling]
9	guidance	[guidance]

Keyword Universe

Code to retrieve all data to python

```
query_text_papers = '''
    select "Publication ID" publication_id,
       "Title" title,
       "Abstract" abstract,
       array_length(string_to_array("Authors", ';'), 1)  num_authors,
       "PubYear" pub_year
    from papersdb
'''

query_text_patent = '''
    SELECT patentid,
       title,
       abstract,
       coalesce(jsonb_array_length(cite -> 'backwardReferences'), 0) num_backward,
       coalesce(jsonb_array_length(cite -> 'forwardReferencesOrig'), 0) num_forward
    FROM patentdb
'''

query_text_awards = '''
    SELECT "RecordID" record_id,
           "Award Title" title,
           "abstract" abstract,
           "Award Year" award_year,
           "Award Amount" award_amount
    FROM sbir_award_data
'''
```

Detect keyword in from texts of relational datasets

3. Detect keywords in from texts of relational datasets
Look for the existence of a list of keywords in the texts of the papers, awards, and patents datasets and add the found keywords as a new column

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

```python
def get_kw_data_in_df(df, searched_keywords):
    '''
    Looks for the existence of keywords in a text
    '''
    chars_to_delete = string.punctuation + string.digits + '\n≥'

    found_kw_data = []
    id_col = list(df)[0]

    n = len(df)
    for i, row in tqdm(df.iterrows(), total=n, position=0, leave=True):
        try:
            id_row = row[id_col]
            total_text = str(row['abstract']) + ' ' + str(row['title'])
            total_text = total_text.lower()
            total_text = total_text.replace('\n', ' ')

            clean_text = re.sub(f'[{chars_to_delete}]',  ' ', total_text)
            clean_text = re.sub(' +', ' ', clean_text)

            kw_found = [kw for kw in searched_keywords if kw in clean_text]

            if len(kw_found)>0:
                kw_data = {
                    id_col: id_row,
                    'keywords': kw_found
                }
                found_kw_data.append(kw_data)
        except:
            pass
    return pd.DataFrame(found_kw_data)
```

In [90]:
```python
# Retrieve the keywords found in the texts of each type of documents
awards_keywords = get_kw_data_in_df(text_awards.sample(2000, random_state=2), searched_keywords)
patents_keywords = get_kw_data_in_df(text_patents.sample(2000, random_state=2), searched_keywords)
papers_keywords = get_kw_data_in_df(text_papers.sample(2000, random_state=2), searched_keywords)
```

```
100%|██████████████████████████████| 2000/2000 [00:02<00:00, 941.26it/
s]
100%|██████████████████████████████| 2000/2000 [00:01<00:00, 1268.98it/
s]
100%|██████████████████████████████| 2000/2000 [00:02<00:00, 852.89it/
s]
```

Now, we find whether the keyword exists in the text.

For each row, we get the title and abstract. We cleaned the text and then looked for keywords. If we find a match, we return them.

The run time was good for this since we limited the length of the abstract in the cleaning process to 3,000 words max.

Detect keyword in from texts of relational datasets

3. Detect keywords in from texts of relational datasets
Look for the existence of a list of keywords in the texts of the papers, awards, and patents datasets and add the found keywords as a new column

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

Our first approach was to do the previous operations in postgres but then we had issues with data cleaning so we did this in python instead.

The main challenge we faced doing it like below was that the dataset (abstract) was too big.

Original postgres query

```
select *
from (select b.abstract, b.patentid, array_agg(b.keywords) as keywords
     from (select *,
                 case
                     when position(a.keywords in a.abstract) > 0 then 1
                     else 0 end tag
           from (select patentid, abstract, unnest(array ['NLP', 'text classification']) as
keywords
                 from patentdb) a) b
     where b.tag = 1
     group by b.abstract, b.patentid) d
        join
    patentdb e
    on d.patentid = e.patentid
```

Upload the edges & relationships

publication_id	keywords
pub.1149276597	[modeling, simulation]
pub.1149467428	[rf, detection, sensor]
pub.1144471504	[isr]
pub.1150097364	[training]
pub.1151482954	[simulation]

Uploads the elements of a DataFrame as nodes. It will upload all the content as attributes if not specified.

```
df = data[features].copy() if features != [] else data.copy()

querys = []
ln = len(df)
for i, row in tqdm(df.iterrows(), total=ln, position=0, leave=True):
 for i, row in df.iterrows():
    drow=row.to_dict()
    query = f'create ({label[0].lower()}:{label} {{'

    n = len(drow)
    # Creating the query
    for i, key in enumerate(drow):
        val = drow[key]
        if isinstance(val, str):
            qry = f"{key}:'{val}'"
        elif isinstance(val, int) or isinstance(val, float):
            qry = f"{key}:{val}"
        else:
            qry = f"{key}:'{val}'"
        query += f'{qry}, ' if i < n-1 else qry
    query += '})'
    #print(query)
    session.run(query)
    querys.append(query)
```

Uploads the relationship between elements as edges.

```
col_a, col_b = list(data)
la = label_a[0].lower()
lb = label_b[0].lower()
lr = label_rel[0].lower()
df = data.copy()

querys = []
ln = len(df)
for i, row in tqdm(df.iterrows(), total=ln, position=0, leave=True):
 for i, row in df.iterrows():

    drow=row.to_dict()

    n = len(drow)
    keys = list(drow)

    query = f'''
    match ({la}:{label_a}), ({lb}:{label_b})
    where {la}.{col_a} = '{drow[col_a]}' AND {lb}.{col_b} = '{drow[col_b]}'
    create ({la})-[{lr}:{label_rel}]->({lb})
    '''
    #print(query)
    session.run(query)

querys.append(query)
```

Creates a dataframe unnesting a column of type str[ ]

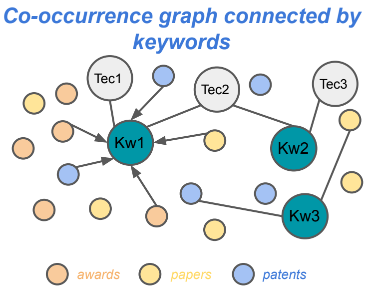
```
rels = []
id_col, agg_col = list(df)
for i, row in df.iterrows():
    for elm in row[agg_col]:
        id_val = row[id_col]
        rels.append([id_val, elm])
df_rels = pd.DataFrame(rels, columns=[id_col, agg_col])
return df_rels
```

Upload the edges & relationships

4. Load the ids, keywords and attributes to neo4j
Extract the ids, keywords and attributes from postgres and load them to neo4j (nodes and edges)

Patentsdb → ['explainability', 'quant computing]
Awardsdb → ['nlp', efficient computing']
Papersdb → ['computer vision', 'nlp"]

	publication_id	keywords
	pub.1149276597	[modeling, simulation]
	pub.1149467428	[rf, detection, sensor]
	pub.1144471504	[isr]
	pub.1150097364	[training]
	pub.1151482954	[simulation]



	record_id	award_year	award_amount
84	21082	2019	24977.0
338	21335	2019	745925.0
350	21350	2019	49999.0

	record_id	keywords
0	105385	monitoring
1	4096	autonomous
2	39142	rf
3	39142	detection

	publication_id	keywords
0	pub.1149276597	modeling
1	pub.1149276597	simulation
2	pub.1149467428	rf
3	pub.1149467428	detection

	publication_id	num_authors	pub_year
30	pub.1152483797	3	2022
76	pub.1152985106	5	2022
120	pub.1152927198	5	2022

	keywords
0	supervised learning by classification
1	simulation languages
2	approximate dynamic programming methods

	patentid	num_backward	num_forward
26	US7342497B2	56	68
102	US10437335B2	49	0
294	JP2010177687A	0	0

	patentid	keywords
0	EP1395385B1	laser
1	US9961675B1	rf
2	CN209948036U	rf
3	CN209948036U	antenna

	technology
0	machine learning
1	simulation
2	uav

	technology	keywords
0	machine learning	machine learning
1	machine learning	learning paradigms
2	machine learning	reinforcement learning
3	machine learning	apprenticeship learning

```
# UPLOADING NODES

# Upload Nodes: Awards, Papers, Patents
upload_nodes_df_to_neo4j(label='AWARDS', data=text_awards_data, session=session_neo4j)
upload_nodes_df_to_neo4j(label='PAPERS', data=text_papers_data, session=session_neo4j)
upload_nodes_df_to_neo4j(label='PATENTS', data=text_patents_data, session=session_neo4j)

# Upload Nodes: Keywords, Technology
upload_nodes_df_to_neo4j(label='KEYWORDS', data=keywords_data, session=session_neo4j)
upload_nodes_df_to_neo4j(label='TECHNOLOGY', data=technology_data, session=session_neo4j)
```

```
100%| | 1480/1480 [00:08<00:00, 182.06it/s]
100%| | 1212/1212 [00:04<00:00, 254.61it/s]
100%| | 1031/1031 [00:03<00:00, 264.13it/s]
100%| | 257/257 [00:00<00:00, 291.27it/s]
100%| | 48/48 [00:00<00:00, 247.58it/s]
```

```
# UPLOADING EDGES

# Upload Edges: Awards, Papers, Patents - Keywords
upload_edges_df_to_neo4j(label_a='AWARDS', label_b='KEYWORDS', label_rel='HAS_KEYWORD', data=rels_awards, session=session_neo4j)
upload_edges_df_to_neo4j(label_a='PAPERS', label_b='KEYWORDS', label_rel='HAS_KEYWORD', data=rels_papers, session=session_neo4j)
upload_edges_df_to_neo4j(label_a='PATENTS', label_b='KEYWORDS', label_rel='HAS_KEYWORD', data=rels_patents, session=session_neo4j)

# Upload Nodes: Technology - Keywords
upload_edges_df_to_neo4j(label_a='TECHNOLOGY', label_b='KEYWORDS', label_rel='HAS_KEYWORD', data=rels_tech, session=session_neo4j)
```

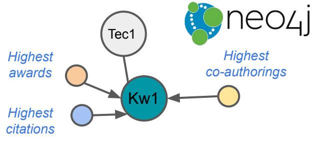
```
100%| | 3660/3660 [00:27<00:00, 134.52it/s]
100%| | 2008/2008 [00:16<00:00, 121.05it/s]
100%| | 1588/1588 [00:12<00:00, 131.88it/s]
100%| | 266/266 [00:01<00:00, 144.73it/s]
```

Output Neo4j network

Once we upload nodes and relationships into neo4j, we can query for everything related to one technology.

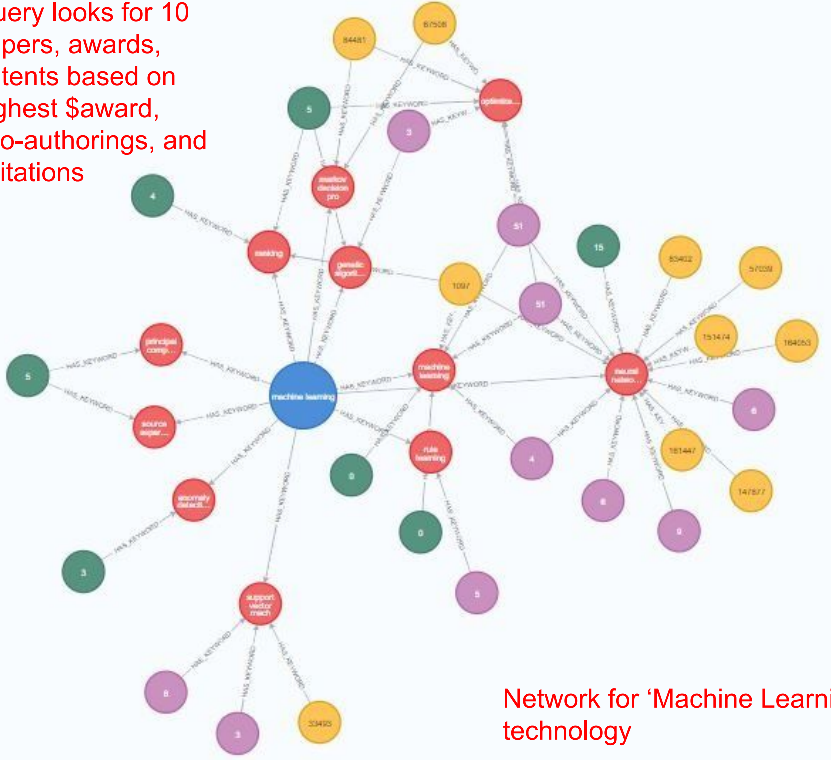
```
MATCH
(a:PAPERS)-[h:HAS_KEYWORD]->(k:KEYWORDS)<-[m:HAS_KEYWORD]-(t:TECHNO
LOGY)
WHERE t.technology='machine learning'
WITH a, h, k, m, t ORDER BY a.num_backward DESC, a.num_forward DESC
RETURN a, h, k, m, t
LIMIT 10

UNION ALL

MATCH
(a:AWARDS)-[h:HAS_KEYWORD]->(k:KEYWORDS)<-[m:HAS_KEYWORD]-(t:TECHNO
LOGY)
WHERE t.technology='machine learning'
WITH a, h, k, m, t ORDER BY a.num_backward DESC, a.num_forward DESC
RETURN a, h, k, m, t
LIMIT 10

UNION ALL

MATCH
(a:PATENTS)-[h:HAS_KEYWORD]->(k:KEYWORDS)<-[m:HAS_KEYWORD]-(t:TECHN
OLOGY)
WHERE t.technology='machine learning'
WITH a, h, k, m, t ORDER BY a.num_backward DESC, a.num_forward DESC
RETURN a, h, k, m, t
LIMIT 10
```

5. Find the paths that relate the keywords from the tech to the documents
Using the keywords as intermediate nodes, find which docs are connected and return the most important



Tech 1 → patent_id1, patent_id2, paper_id1, paper_id2, award_id1, award_id2

Query looks for 10 papers, awards, patents based on highest $award, #co-authorings, and #citations



Network for ‘Machine Learning’ technology

Postgres Data Gathering

6. Get information about the selected document ids
From the chosen ids, get interesting data from the available datasets on postgres.

Select (important_data) From patentsdb Where id in (patent_id1, patent_id2)

Select (important_data) From awardsdb Where id in (award_id1, award_id2)

Select (important_data) From papersdb Where id in (paper_id1. paper_id2)

We will retrieve information from postgres for the key ids obtained from the keyword co-occurrence graph.

Operations used: *window functions, subqueries, groups, unnest, dates, jsonb, order,*

PATENTS

```sql
WITH AWARDS AS (
    SELECT "RecordID" record_id
         , "Company" company
         , "Award Title" award_title
         , "Agency" agency
         , "Branch" branch
         , "Phase" phase
         , TO_DATE("Proposal Award Date", 'MM/DD/YYYY')::timestamp proposal_date
         , TO_DATE("Contract End Date", 'MM/DD/YYYY')::timestamp contract_end_date
         , "Award Year" award_year
         , replace("Award Amount",',','')::float award_amount
         , "Number Employees" num_employees
         , "City" city
         , case when length("abstract") < 3000 then "abstract" else substr("abstract", 1, 3000)
         , "Contact Name" contact_name
         , "Contact Phone" contact_phone
         , "Contact Email" contact_email
    FROM sbir_award_data
)
, COMPANY_DATA AS (
    SELECT company
        , count(1) num_awards_company
        , max(award_amount) max_award_amount_company
        , sum(award_amount) total_award_amount_company
        , round(avg(award_amount)) avg_award_amount_company
    FROM AWARDS
    GROUP BY 1
)
, DATA_AWARDS AS (
    SELECT A.record_id
        , A.award_title
        , A.abstract
        , A.agency
        , A.phase
        , A.num_employees
        , A.company
        , B.num_awards_company
        , A.award_amount
        , B.max_award_amount_company
        , B.total_award_amount_company
        , B.avg_award_amount_company
        , A.proposal_date::date proposal_date
        , A.contract_end_date::date contract_end_date
        , abs(extract(days from A.contract_end_date - A.proposal_date)) award_days_duration
        , A.contact_name
        , A.contact_phone
        , A.contact_email
    FROM AWARDS A
    LEFT JOIN COMPANY_DATA B ON A.company = B.company
)
SELECT *
FROM DATA_AWARDS
WHERE record_id IN CHOSEN_IDS
  LIMIT 300
```

Select (important_data) From patentsdb Where id in (patent_id1, patent_id2)

AWARDS

```sql
WITH
papers as (
    SELECT "Rank" rank
        , "Publication ID" publication_id
        , "DOI" doi
        , "PMID" pmid
        , "PMCID" pmcid
        , "Title" title
        , "Abstract" abstract
        , "Acknowledgements" acknowledgements
        , "Source title" source_title
        , "Publisher" publisher
        , "RCDC Categories" rcdc_categories
        , string_to_array("Authors", '; ') authors_array
    FROM papersdb
)
, AUTHORS AS (
    SELECT publication_id
        , unnest(authors_array) authors
    FROM papers
)
, COUNTER_PUBLICATIONS AS (
    SELECT authors
        , count(1) num_publications
    FROM AUTHORS
    GROUP BY 1
)
, PUBLICATION_COUNTER AS (
    SELECT A.publication_id, A.authors, B.num_publications
    FROM AUTHORS A
    LEFT JOIN COUNTER_PUBLICATIONS B ON A.authors = B.authors
)
, STATS_CITATIONS AS (
    SELECT publication_id
        , MAX(num_publications) max_publications_per_author
        , round(AVG(num_publications), 2) avg_publications_per_author
        , SUM(num_publications) total_publications_authors
        , COUNT(num_publications) num_authors
    FROM PUBLICATION_COUNTER
    GROUP BY 1
)
, DATA_PAPERS AS (
    SELECT A.publication_id
        , A.rank
        , A.title
        , case when length(A.abstract) < 3000 then A.abstract else substr(A.abstract, 1, 3000) end abstract
        , A.publisher
        , A.mesh_terms
        , A.publication_date
        , A.open_access
        , A.authors
        , B.avg_publications_per_author
        , B.total_publications_authors
        , B.max_publications_per_author
        , B.num_authors
        , A.research_organizations
        , A.dimensions_url
    FROM PAPERS A
    LEFT JOIN STATS_CITATIONS B ON A.publication_id=B.publication_id
)
SELECT *
FROM DATA_PAPERS
WHERE PUBLICATION_ID IN CHOSEN_IDS
```

Select (important_data) From awardsdb Where id in (award_id1, award_id2)

PAPERS

```sql
WITH
EVENT_DATASET AS (
    WITH
    TIME_EVENTS AS (
        SELECT p.patentid
            , arr.value->>'title' as event
            , arr.value->>'date' as date
            , arr.value
        FROM patentdb p, jsonb_array_elements(time_events) as arr
        WHERE jsonb_typeof(time_events) = 'array'
    )
    , RANKED_EVENTS AS (
        SELECT
            *
           , row_number() over (partition by patentid order by date asc) rown_asc
           , row_number() over (partition by patentid order by date desc) rown_desc
        FROM TIME_EVENTS
    )
    , AGG_REGISTRIES AS (
        SELECT patentid
    -- Events: publication, external-priority, reassignment, priority,
    -- filed, legal-status, litigation, granted
            , sum(CASE WHEN event = 'publication' then 1 else 0 end) num_publications
            , sum(CASE WHEN event = 'granted' then 1 else 0 end) num_grants
            , sum(CASE WHEN event = 'litigation' then 1 else 0 end) num_litigations
            , min(CASE WHEN event = 'publication' then date end) first_publication
            , max(CASE WHEN event = 'publication' then date end) last_publication
            , min(CASE WHEN event = 'granted' then date end) first_grant
            , max(CASE WHEN event = 'granted' then date end) last_grant
        FROM TIME_EVENTS
        GROUP BY 1
    )
    , FIRST_REGISTRY AS (
        SELECT *
        FROM RANKED_EVENTS
        WHERE rown_asc = 1
        --    AND  patentid='AU1769501A'
    )
    , LAST_REGISTRY AS (
        SELECT *
        FROM RANKED_EVENTS
        WHERE rown_desc = 1
        --    AND  patentid='AU1769501A'
    )
    SELECT ar.*
        , fr.event first_status
        , fr.date first_status_date
        , lr.event last_status
        , lr.date last_status_date
        --, round(abs(extract(days from ar.first_publication::timestamp - fr.date::timestamp))/30.4)
months_from_first_activity_to_first_publication
    FROM AGG_REGISTRIES ar
    LEFT JOIN FIRST_REGISTRY fr ON ar.patentid=fr.patentid
    LEFT JOIN LAST_REGISTRY lr ON ar.patentid=lr.patentid
)
, INVENTOR_DATASET AS (
    WITH
    INVENTORS AS (
        SELECT patentid
            , unnest(inventor_name) inventor
        FROM patentdb
    )
    , COUNTER_PATENTS AS (
```

Select (important_data) From papersdb Where id in (paper_id1. paper_id2)

PATENTS

	record_id	award_title	abstract	agency	phase	num_employees	company	num_awards_company	award_amount	max_award_amount_company	total_award_amount_company	avg_award_amount_compa
0	88708	Achieving a High Level of Scalability in Feder...	A federated search engine that is capable of s...	Department of Energy	Phase II	14.0	Deep Web Technologies, Llc	22	499078.0	749826.0	5791368.0	26324
1	89892	Adaptive Toolkit for Pattern Discovery (ATPD)	This project will design, configure, and demon...	Department of Defense	Phase II	80.0	KNOWLEDGE BASED SYSTEMS, INC.	253	1749910.0	7433422.0	102220746.0	40403
2	97535	Wireless Fidelity from the Sea (WFFTS)	The Navy requires development of enabling tech...	Department of Defense	Phase I	300.0	OCEAN SYSTEMS ENGINEERING CORP.(OSEC)	10	18000.0	750000.0	2625896.0	26259

AWARDS

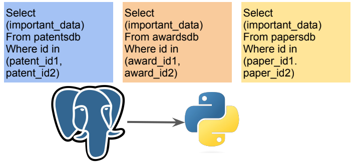
	publication_id	rank	title	abstract	publisher	mesh_terms	publication_date	open_access	authors	avg_publications_per_author	total_publications_authors	max_publications_per_author	num_authors
0	pub.1152720431	500	Fast rule switching and slow rule updating in ...	To adapt to a changing world, we must be able ...	eLife	None	2022-11-14	All OA; Gold	Bouchacourt, Flora; Tafazoli, Sina; Mattar, Ma...	1.40	7.0	3	5
1	pub.1152680845	100	Automated Detection of Posterior Vitreous Deta...	Objective To develop automated algorithms for ...	Elsevier	None	2022-11	All OA; Gold	Li, Alexa L.; Feng, Moira; Wang, Zixi; Baxter,...	3.73	41.0	12	11
2	pub.1152356379	500	Combining OCT and OCT-Angiography Longitudinal...	PURPOSE: To use longitudinal optical coherence...	Elsevier	None	2022-10-31	Closed	Kamalipour, Alireza; Moghimi, Sasan; Khosravi,...	13.15	171.0	32	13

PAPERS

	patentid	inventor_name	assignee_name_origin	assignee_name_current	countrycode	title	abstract	num_publications	num_grants	num_litigations	first_publication	last_publication
0	JP6592755B1	[雄一 佐々木 太一 小井土 路威 重松]	[ニューラルポケット 株式会社]	[]	[JP, JP]		\nA system, method, and program for more appro...	1	1	0	2019-10-23	2019-10-23
1	US6735703B1	[Douglas Kilpatrick, Mark Lee Badger, Calvin Ko]	[Networks Associates Technology Inc]	[McAfee LLC]	[US, US]	System and method for providing a network host...	\nA real-time sequence-based anomaly detection...	1	1	0	2004-05-11	2004-05-11
2	RU2701843C2	[Владимир Сергеевич Пахомов]	[Федеральное государственное казённое военное ...	[]	[RU, RU]	DEVICE FOR SELECTING A VARIANT OF ACCOMPLISHED...	\nFIELD: calculating; counting.\nSUBSTANCE: in...	3	1	0	2019-04-23	2019-10-01

Postgres Data Gathering

6. Get information about the selected document ids
From the chosen ids, get interesting data from the available datasets on postgres.



The following are some of the operations we used to process the data

window functions, subqueries, groups, unnest, dates, jsonb, order,

Unnest

Group by

Merge

```
)
, AUTHORS AS (
    SELECT publication_id
         , unnest(authors_array) authors
    FROM papers
)
, COUNTER_PUBLICATIONS AS (
    SELECT authors
         , count(1) num_publications
    FROM AUTHORS
    GROUP BY 1
)
, PUBLICATION_COUNTER AS (
    SELECT A.publication_id, A.authors, B.num_publications
    FROM AUTHORS A
    LEFT JOIN COUNTER_PUBLICATIONS B ON A.authors = B.authors
)
, STATS_CITATIONS AS (
    SELECT publication_id
         , MAX(num_publications) max_publications_per_author
         , round(AVG(num_publications), 2) avg_publications_per_author
         , SUM(num_publications) total_publications_authors
         , COUNT(num_publications) num_authors
    FROM PUBLICATION_COUNTER
    GROUP BY 1
)
```

JSONB

Window functions

Group by

```
WITH
EVENT_DATASET AS (
    WITH
    TIME_EVENTS AS (
        SELECT p.patentid
             , arr.value->>'title' as event
             , arr.value->>'date' as date
             , arr.value
        FROM patentdb p, jsonb_array_elements(time_events) as arr
        WHERE jsonb_typeof(time_events) = 'array'
    )
    , RANKED_EVENTS AS (
        SELECT
             *
             , row_number() over (partition by patentid order by date asc) rown_asc
             , row_number() over (partition by patentid order by date desc) rown_desc
        FROM TIME_EVENTS
    )
    , AGG_REGISTRIES AS (
        SELECT patentid
    -- Events: publication, external-priority, reassignment, priority,
    -- filed, legal-status, litigation, granted
             , sum(CASE WHEN event = 'publication' then 1 else 0 end) num_publications
             , sum(CASE WHEN event = 'granted' then 1 else 0 end) num_grants
             , sum(CASE WHEN event = 'litigation' then 1 else 0 end) num_litigations
             , min(CASE WHEN event = 'publication' then date end) first_publication
             , max(CASE WHEN event = 'publication' then date end) last_publication
             , min(CASE WHEN event = 'granted' then date end) first_grant
             , max(CASE WHEN event = 'granted' then date end) last_grant
        FROM TIME_EVENTS
        GROUP BY 1
    )
```

Retrieve info from Wikipedia and upload to MongoDB

7. Get wikipedia documents from the selected ids
Look for summary or sidebar information about the companies, keywords, authors, universities, etc. (technologies list)

patent_id1 → 'General motors' (webpage)
Paper_id2 → 'UC San Diego' (webpage)

Companies
Universities
Topics

mongoDB.

Mongo DB records created for [companies, universities, other]

```
_id: ObjectId('639114cd2d6bad708e8fcdf6')
parse: Object
  title: "University of California, San Diego"
  pageid: 31927
  > redirects: Array
  > html: Object
  abstract: " The University of California, San Diego[10] (UC San Diego or colloqu…"
Title: "University of California, San Diego"
Grouping: "university"
Motto: "Fiat lux  ( Latin )"
Motto in English: "" Let there be light ""
Type: "Public  land-grant  research university"
Established: "November 18, 1960 ; 62 years ago  ( 1960-11-18 ) [1]"
Parent institution: "University of California"
Accreditation: "WSCUC"
Academic affiliations: "AAU APRU URA Sea-grant Space-grant"
Endowment: "$2.6 billion (2021) [2]"
Chancellor: "Pradeep Khosla [3]"
Academic staff: "10,915 (October 2020) [4]"
Administrative staff: "23,461 (October 2020) [4]"
Students: "42,968 (Fall 2022) [5]"
Undergraduates: "33,096 (Fall 2022) [5]"
Postgraduates: "9,872 (Fall 2022) [5]"
Location: "La Jolla ,  San Diego ,  California ,  United States 32°52′48″N  117°…"
Campus: "Large City, [6]  2,178 acres (881 ha) [7]"
Newspaper: "UCSD Guardian"
Colors: "UC San Diego blue and gold [8]      "
Nickname: "Tritons"
Sporting affiliations: "NCAA Division I  –  Big West MPSF WWPA WIRA"
Mascot: "King Triton [9]"
Website: "ucsd.edu"
```

Using the API, we can retrieve the boxed data with the page's html

The **University of California, San Diego**[10] (**UC San Diego** or colloquially, **UCSD**) is a public land-grant research university in San Diego, California.[11] Established in 1960 near the pre-existing Scripps Institution of Oceanography, UC San Diego is the southernmost of the ten campuses of

University of California, San Diego

Motto	*Fiat lux* (Latin)
Motto in English	"Let there be light"
Type	Public land-grant research university
Established	November 18, 1960; 62 years ago[1]
Parent institution	University of California
Accreditation	WSCUC
Academic affiliations	AAU · APRU · URA · Sea-grant · Space-grant
Endowment	$2.6 billion (2021)[2]
Chancellor	Pradeep Khosla[3]

Next, using Beautiful soup, we can parse the html data to extract (1) the abstract paragraphs, and (2) the infobox from the wikipedia page.

The MongoDB collection can be queried to pull out key information e.g. $Endowment, #Academic staff etc.

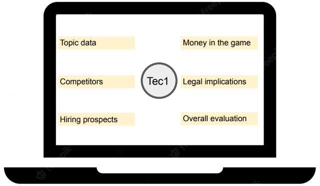
MongoDB function to retrieve items from the Collection

```python
def get_mongo_info_university(universities):
    df = pd.DataFrame()
    for university in universities:
        cursor = collection_page.find( {'Title':f"{university}"},
                                       {'Title': 1,
                                        'Type': 1,
                                        'Location':1,
                                        'Endowment': 1,
                                        'Academic staff':1,
                                        'Total staff':1,
                                        'Undergraduates': 1,
                                        'Postgraduates': 1,
                                        'Nickname':1,
                                        'Mascot': 1,
                                        '_id': 0 } )


        df_i =  pd.DataFrame(list(cursor)).fillna("-")
        df = df.append(df_i)
    df = df.drop_duplicates().reset_index()
    return df
```

Reporting

Now we are able to create a report for each one of the technologies once we select them

8. Build a report with the gathered data providing recommendations
Use information retrieved from 6 and 7 to build a report for a client on who to hire, which competitors we have, among others



FINAL PROJECT - DSC 202 Data management

Technology Studied: MACHINE LEARNING

```
Note: This report contains information about papers, patents and awards associated with Machine Learning
```

a) Key Awards in Machine Learning

	record_id	award_title	abstract	agency	phase	num_employees	company	num_awards_company	award_an
0	88708	Achieving a High Level of	A federated search engine that	Department	Phase	14.0	Deep Web Technologies,	22	49

b) Key Patents in Machine Learning

	patentid	inventor_name	assignee_name_origin	assignee_name_current	countrycode	title
0	JP6592755B1	[雄一 佐々木, 太一 小井土, 路威 重松]	[ニューラルポケット株式会社]	[]	[JP, JP]	an
		[Douglas				System and

c) Key Papers in Machine Learning

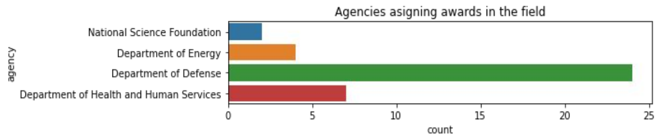
publication_id	rank	title	abstract	publisher	mesh_terms	publication_date	open_access	autho
pub.1152720431	500	Fast rule switching and slow rule updating in ...	To adapt to a changing world, we must be able ...	eLife	None	2022-11-14	All OA; Gold	Bouchacou Flor Tafazo Sina; Matta

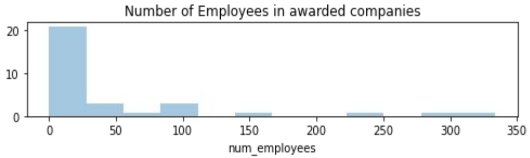
d) Companies in the field of Machine Learning

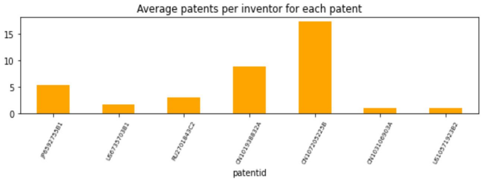
	index	Title	Type	Revenue	Number of employees
0	0	Technology Service Corporation	Private	US$37.2 million	400

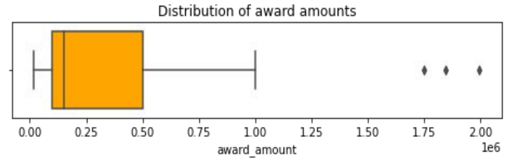
e) Universities in the field of Machine Learning

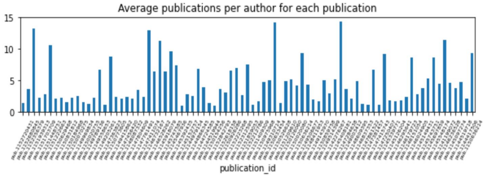
	index	Title	Type	Endowment	Academic staff	Undergraduates	Post
0	0	West Virginia University	Public land-grant research university	$611.3 million (2020) [2]	1,870	21,086 (Morgantown) [6] \n 1,300 (Keyser) [7] ...	(Morga
1	0	Princeton University	Private research	$37.7 billion	1,289 [3]	5,422 (Fall 2019) [5]	2,997











Next Steps 

1. *Expand keyword universe list by incorporating new sources besides the ACM ontology. Using the ACM website only picked up computing related technologies*

2. *We could look for approximate matches when keyword matching e.g. ‘machine learning’ vs. ‘learning machines’. (‘fuzzy wuzzy’ library)*

3. *We only uploaded ~2000 nodes (papers, patents, awards) of each, we could try to do this for all the data and get a denser data.*

4. *Wikipedia information retrieved through the API needs significant cleaning for the data to be more operable. Ideally this should be done before the upload. If this was the case, we could do more complex queries through the mongo connection rather than in python with the retrieved data.*

5. *Parallelize the code to use multiple cores on our machines or upload all data to the cloud to take advantage of elastic resource capability.*

Happy Holidays!