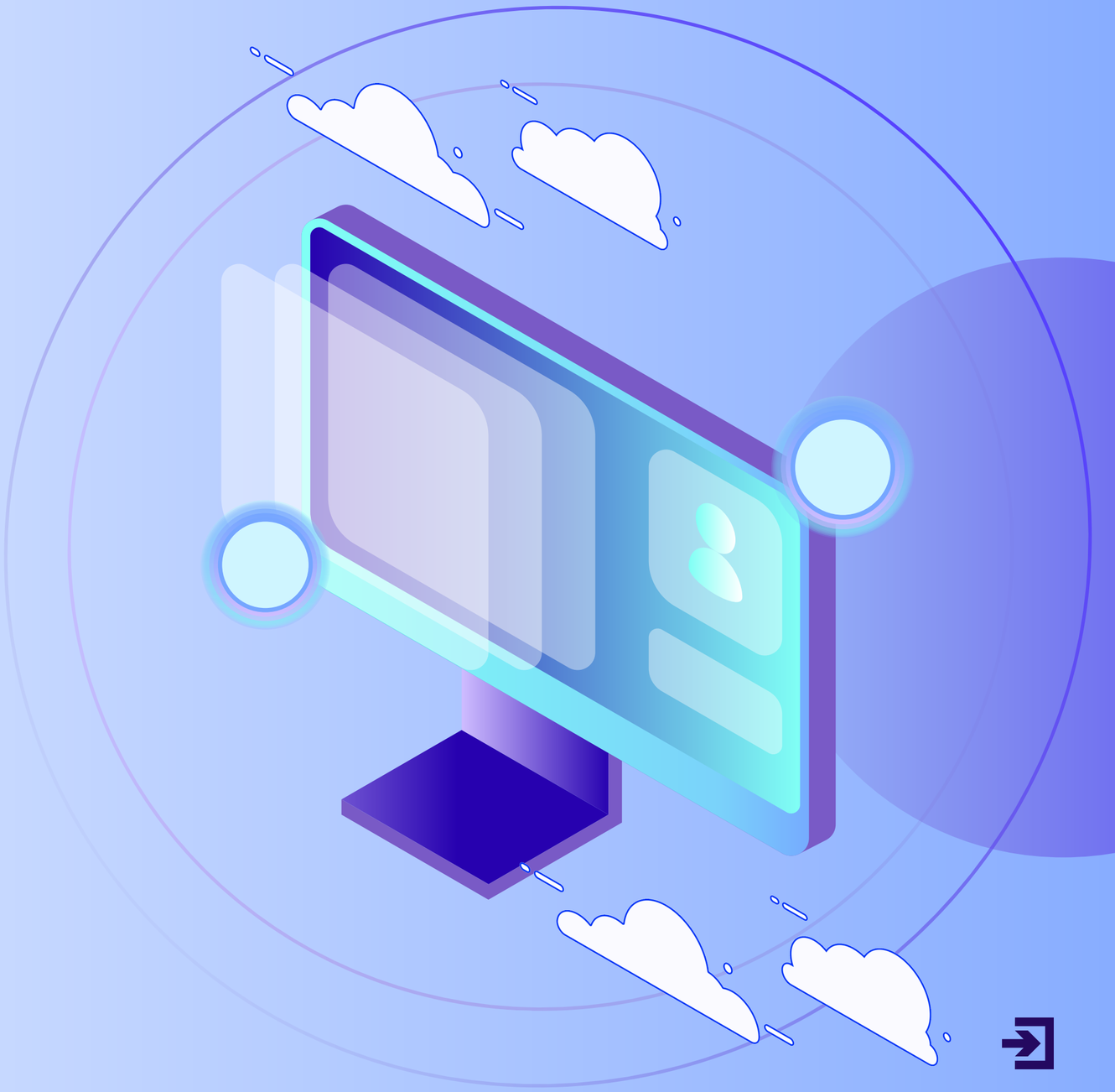
VERY DEMURE's

DATA SCIENCE PROJECT



VERY DEMURE'S TEAM

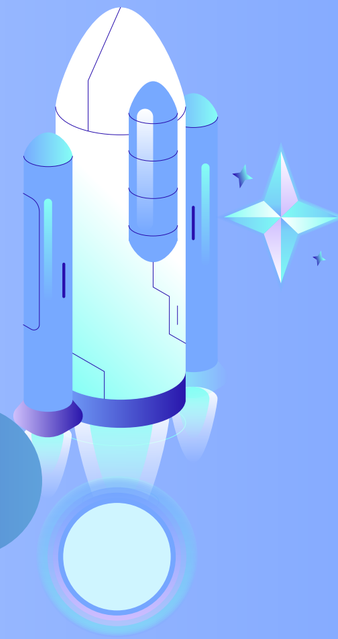
Panatchakorn Tirachuli
6638145521

Pimmada Aphiwetsa
6638166721

Sunatcha Kulaputana
6638252021

Anunya Tangrungruangchai
6638264521

OUR PROJECT FLOW



DATA Module

Scopus

Data

748

+

300

Beautiful Soup

SCOPUS APIs

arXiv

AI Module

classify and vectorize

TF - IDF

train + test

predict subject area

LOGISTIC REGRESSION MODEL

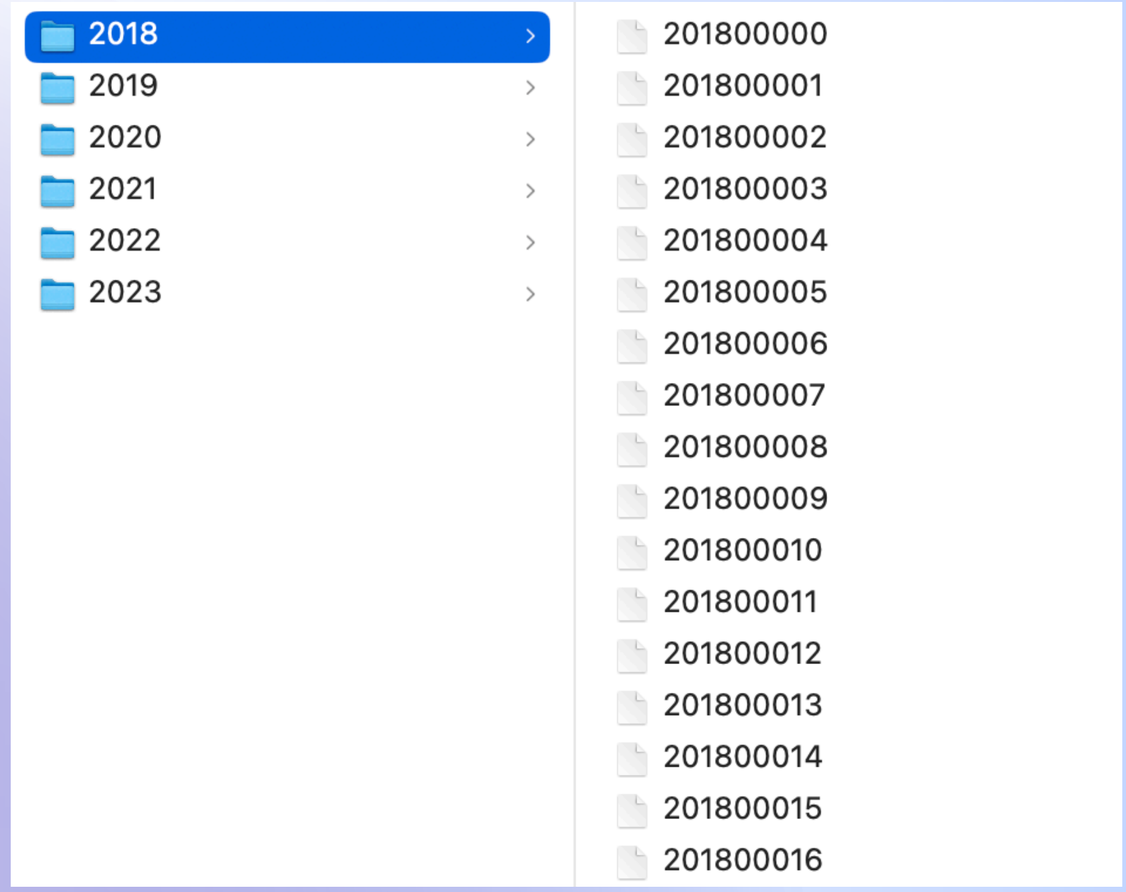
Visualization

Streamlit

plotly

GeoPy

EXTRACTING DATA



```python
def process_file(file_path):
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            data = json.load(file)
        return {
            "Title": title(data),
            "Cover Date": coverDate(data),
            "Aggregation Type": aggregationType(data),
            "Authors": ", ".join(author(data)) if author(data) else None,
            "Subject Code": ", ".join(subjectCode(data)) if subjectCode(data) else None,
            "Subject Areas": ", ".join(subjectArea(data)) if subjectArea(data) else None,
            "Author Keywords": ", ".join(authKeyword(data)) if authKeyword(data) else None,
            "Abstract": abstract(data),
            "Reference Count": refCount(data),
            "Publication Name": publicationName(data),
            "City": city(data),
            "Country": country(data),
            "Citation Count" : citedbyCount(data)
        }
    except KeyError as e:
        print(f"KeyError for file {file_path}: {e}")
    except json.JSONDecodeError as e:
        print(f"JSONDecodeError for file {file_path}: {e}")
    except Exception as e:
        print(f"Error processing file {file_path}: {e}")
    return None
```

Example Function

```python
def subjectCode(data):
    try:
        subject_area_list = data.get('abstracts-retrieval-response', {}).get('subject-areas', {}).get('subject-area', [])

        abbreviations = [area.get('@abbrev') for area in subject_area_list if '@abbrev' in area]

        return abbreviations
    except (TypeError, AttributeError):
        return None
```

Error Handling

SAVE TO CSV

Title	Cover Date	Aggregation Type	Authors	Subject Code	Subject Areas	Author Keywords	Abstract
Recent developments in bif	2018-11-15	Journal	Hunsom, Kaewsai, Kannan	ENER, ENER, PHYS, ENER	Renewable Energy, Sustainability and th	Bifunctional air electrode, Catalyst suppor	© 2018 Hydrogen Energy Pul
The benefit of punishment	2018-06-01	Journal	Harry Manley, Stuart Beattie, F	PSYC	Social Psychology	anxiety, defensive distance, performance,	© 2017 Wiley Periodicals, Inc
MAGIC biomarkers predict	2018-06-21	Journal	Hannah Major-Monfried, Anne	BIOC, IMMU, MEDI, BIOC	Biochemistry, Immunology, Hematology,	Cell Biology	© 2018 by The American Soc
Are endocrine disrupting co	2018-05-01	Journal	Amer Moosa, Henry Shu, Tewa	BIOC, NEUR, NEUR	Endocrinology, Endocrine and Autonom	Autism, Endocrine disrupting compounds,	© 2017 Elsevier Inc.Recent re
Probabilistic regular gramn	2018-01-01	Conference Proceeding	Torsak Penpinun, Athasit Surar	COMP, COMP, COMP	Information Systems, Computer Science	Grammar inference, Incremental technique	© Proceedings of 2018 the 8
Optimization of cinnamon c	2018-03-15	Journal	Piyanan Chuesiang, Ubonrat S	MATE, MATE, MATE, CENG	Electronic, Optical and Magnetic Materi	Cinnamon oil, Nanoemulsions, Natural pro	© 2017 Elsevier Inc.Essential
Identifying factors associat	2018-11-01	Journal	Kazumi Taguchi, Jonathan D. H	MEDI	Urology	Durability, Reusable flexible ureteroscope,	© 2017, Springer-Verlag Gmb
Serum tryptophan, tryptoph	2018-01-01	Journal	Heidi Ormstad, Vesna Bryn, Rc	NEUR, PHAR	Neuroscience (all), Pharmacology	Autism spectrum disorders, Brain-derived	© 2018 Bentham Science Pu
Media perspectives on the	2018-01-01	Book Series	Yifan Zhang	SOCI, SOCI, SOCI, SOCI	Sociology and Political Science, Development, Communication, Cultural Studies		
Delta-like ligand 4 in hepat	2018-09-14	Journal	Areerat Kunanopparat, Jiraphc	MEDI	Gastroenterology	Delta-like ligand 4, Hepatocellular carcinor	© The Author(s) 2018. Publish
Cyanobacterial UV sunscre	2018-01-01	Book	Hakuto Kageyama, Rungaroon	BIOC	Biochemistry, Genetics and Molecular B	Carotenoids, Cyanobacteria, Glycoside, M	© 2010 by Nova Science Pub
A new 4-arylflavan from the	2018-11-17	Journal	Rico Ramadhan, Irawan Wijaya	CHEM, BIOC, AGRI, CHEM	Analytical Chemistry, Biochemistry, Plan	antioxidant, diabetes, flavan, Horsfieldia m	© 2017, © 2017 Informa UK L
WS2 and C-TiO2 Nanorods	2018-12-11	Journal	Chengwu Yang, Jiaqian Qin, S	ENVI, CENG, MATE, ENER	Environmental Chemistry, Chemical Eng	charge separators, g-C3N4 composite, hy	© 2018 Wiley-VCH Verlag Gn
Lower bound limit analysis	2018-06-10	Journal	Boonchai Ukritchon, Suraparb	ENGI, MATE, EART, ENGI	Computational Mechanics, Materials Sc	anisotropic strength, finite element limit an	Copyright © 2018 John Wiley
The association of PNPLA3	2018-08-01	Journal	Rattanaporn Kiatbumrung, Nat	MEDI, MEDI, MEDI, BIOC	Epidemiology, Oncology, Public Health,	Cirrhosis, Fibrosis, HCV, HIV, Polymorphisı	© 2018 Asian Pacific Organiz
Ontology-Based Framewor	2018-01-01	Book Series	Marut Buranarach, Thanyalak I	ENGI, COMP	Control and Systems Engineering, Comp	Decision table, Knowledge engineering, Re	© Springer International Publ
Insight into a conformation	2018-04-16	Journal	Amphawan Maitarad, Nattawe	PHYS, CHEM	Physics and Astronomy (all), Physical ar	Binding energy, Molecular dynamics simul	© 2018 Elsevier B.V.Suitable
Constraints on the double-(	2018-02-01	Journal	Sirunyan, Tumasyan, Adam, Ar	PHYS	Nuclear and High Energy Physics		© 2018, The Author(s).A first
The effects of an education	2018-08-03	Journal	Saowanee Thongnopakun, Tep	MEDI, MEDI	Health Policy, Public Health, Environmer	Condom and emergency contraceptive pil	© Saowanee Thongnopakun,
Drift-flux correlation for ga	2018-02-01	Journal	Somboon Rassame, Takashi H	PHYS, ENGI, CENG	Condensed Matter Physics, Mechanical	Drift-flux model, Gas-liquid flow, Horizonta	© 2017 Elsevier Inc.A drift-flu
Rare zooxanthellate Nanipo	2018-12-01	Journal	Pau Urgell Plaza, Rahul Mehro	EART, AGRI, AGRI	Oceanography, Ecology, Evolution, Beha	Aragonite skeleton, COI, Koh Tao, Thailanc	© 2017, Senckenberg Gesells
Rational design and synthe	2018-01-01	Journal	Chenyang Li, Jiaqian Qin, Mon	MATE, CENG, MATE, CHEM, MATE, MATE, MATE	Electronic, Optical and Magnetic Materials, Catalysis, Ceramics and Composites, C		© The Royal Society of Chem

SCOPUS API

query = “Data Science”

```
<entry>
  <link ref="self" href="https://api.elsevier.com/content/abstract/scopus_id/85202345424"/>
  <link ref="author-affiliation" href="https://api.elsevier.com/content/abstract/scopus_id/85202345424?
  field=author,affiliation"/>
  <link ref="scopus" href="https://www.scopus.com/inward/record.uri?partnerID=HzOxMe3b&scp=85202345424&origin=
  <link ref="scopus-citedby" href="https://www.scopus.com/inward/citedby.uri?
  partnerID=HzOxMe3b&scp=85202345424&origin=inward"/>
  <prism:url>https://api.elsevier.com/content/abstract/scopus_id/85202345424</prism:url>
  <dc:identifier>SCOPUS_ID:85202345424</dc:identifier>
  <eid>2-s2.0-85202345424</eid>
  <dc:title>Hacking into Datascience</dc:title>
  <dc:creator>Mahmud F.</dc:creator>
  <prism:publicationName>EAGE Workshop on Data Science - From Fundamentals to Opportunities</prism:publicationN
  <prism:isbn>[9789462824898]</prism:isbn>
  <prism:pageRange/>
  <prism:coverDate>2023-01-01</prism:coverDate>
  <prism:coverDisplayDate>2023</prism:coverDisplayDate>
  <prism:doi>10.3997/2214-4609.202377035</prism:doi>
  <citedby-count>0</citedby-count>
▼<affiliation>
    <affilname>Brunei Shell Petroleum</affilname>
    <affiliation-city>Seria</affiliation-city>
    <affiliation-country>Brunei Darussalam</affiliation-country>
  </affiliation>
  <prism:aggregationType>Conference Proceeding</prism:aggregationType>
  <subtype>cp</subtype>
  <subtypeDescription>Conference Paper</subtypeDescription>
  <source-id>21101244727</source-id>
  <openaccess>0</openaccess>
  <openaccessFlag>false</openaccessFlag>
</entry>
```

No abstract!

FIRST TRIAL “PLAYWRIGHT”

```python
async def run(playwright: Playwright):
    args = ["--disable-blink-features=AutomationControlled"]
    browser = await playwright.chromium.launch(headless=True, args=args)
    page = await browser.new_page()

    # Visit the webpage
    await page.goto("https://www.scopus.com/record/display.uri?eid=2-s2.0-85208099811&origin=
    await page.wait_for_load_state("networkidle")

    # Simulate random mouse movement
    await page.mouse.move(random.randint(50, 200), random.randint(50, 200))
    await asyncio.sleep(random.uniform(1, 3))  # Random sleep to simulate natural behavior

    # Extract the content of the #abstractSection
    try:
        abstract = await page.locator('#abstractSection').text_content()
        print(abstract)  # Print the abstract
    except Exception as e:
        print(f"Error fetching abstract: {e}")
        abstract = None

    await browser.close()  # Close the browser
    return abstract
```

BEAUTIFULSOUP

Beautiful Soup

```python
def fetch_abstract(link,delay):
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    }
    try:
        # Request the page
        response = requests.get(link, headers=headers)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')

            # Find the section with id="abstractSection"
            abstract_section = soup.find('section', id='abstractSection')
            if abstract_section:
                # Find all <p> tags within this section
                paragraphs = abstract_section.find_all('p')
                # Combine all text from the <p> tags
                return " ".join(p.get_text(strip=True) for p in paragraphs)
            else:
                return "Abstract section not found"
        elif response.status_code == 403:
            return "Access forbidden - Possible login required"
        elif response.status_code == 429:
            print("Rate limit hit. Retrying ")
            time.sleep(delay)
            delay *= 2
```

```
<section id="abstractSection" class="row"> == $0
  ::before
  <h3 class="h4">…</h3>
  <p>
    "With the rapid advancement of technology and the proliferation of data science globally, there is an opportunity to leverage the trend to deliver further business value. Hackathon provides an excellent platform to not only upskilling individuals but also allows the creation of a natural ecosystem to tackle business problems worth solving. © EAGE Workshop on Data Science - From Fundamentals to Opportunities 2023."
  </p>
```

CHALLENGES

```
# yoo = fetch_abstract(df['link'][0])
# print(yoo)
Tabnine | Edit | Test | Explain | Document | Ask
def extract_abstracts_from_df(df):
    abstracts = []
    for i in range(1000):
        abstract = fetch_abstract(df['link'][i])
        abstracts.append(abstract)
    df['abstract'] = abstracts
    return df

# Extract abstracts
df_with_abstracts = extract_abstracts_from_df(df)

# Display the updated DataFrame
print(df_with_abstracts)
```

122m 58.4s Python

more than 2 hr run time!

unable to access the page

```
Tabnine | Edit | Test | Explain | Document | Ask
def extract_abstracts_from_df(df):
    abstracts = []
    for i in range(11):
        abstract = fetch_abstract(df['link'][i])
        abstracts.append(abstract)
        print(abstract)
    # df['abstract'] = abstracts
    return  abstracts#df

# Extract abstracts
df_with_abstracts = extract_abstracts_from_df(df)

# Display the updated DataFrame
print(df_with_abstracts)
```

✓ 3.6s Python

```
Failed to fetch page (status 429)
Failed to fetch page (status 429)
Failed to fetch page (status 429)
Failed to fetch page (status 429)
```

CHALLENGES

more than 700 abstract is retrieved

```
abstract_count = 0
for i in range(len(batch_extract)):
    if batch_extract[i] != None and batch_extract[i] != "[No abstract available]":
        abstract_count+=1
print(abstract_count)
print(df['abstract'].head())
```

✓ 0.0s

```
754
0    Urban computing with a data science approaches...
1    Objectives: In 2016, the Department of Veteran...
2    This paper explores regional planning as an ap...
3    Model Predictive Control (MPC) is an establish...
4    Purpose The relationship between the height of...
Name: abstract, dtype: object
```

403 error

```
Tabnine | Edit | Test | Explain | Document | Ask
def extract_abstracts_from_df(df):
    delay = 50
    for i in range(800,1000):
        abstract = fetch_abstract(df['link'][i],delay)
        batch_extract.append(abstract)
        print(abstract)
    return  batch_extract
extract_abstracts_from_df(df)
```

⊗ 10.6s

```
Access forbidden - Possible login required
Access forbidden - Possible login required
Access forbidden - Possible login required
Access forbidden - Possible login required
```

ARXIV API

```
  <entry>
    <id>http://arxiv.org/abs/2403.00776v1</id>
    <updated>2024-02-14T15:55:40Z</updated>
    <published>2024-02-14T15:55:40Z</published>
    <title>A framework for understanding data science</title>
    <summary>  The objective of this research is to provide a framework with which the data
science community can understand, define, and develop data science as a field
of inquiry. The framework is based on the classical reference framework
(axiology, ontology, epistemology, methodology) used for 200 years to define
knowledge discovery paradigms and disciplines in the humanities, sciences,
algorithms, and now data science. I augmented it for automated problem-solving
with (methods, technology, community). The resulting data science reference
framework is used to define the data science knowledge discovery paradigm in
terms of the philosophy of data science addressed in previous papers and the
data science problem-solving paradigm, i.e., the data science method, and the
data science problem-solving workflow, both addressed in this paper. The
framework is a much called for unifying framework for data science as it
contains the components required to define data science. For insights to better
understand data science, this paper uses the framework to define the emerging,
often enigmatic, data science problem-solving paradigm and workflow, and to
compare them with their well-understood scientific counterparts, scientific
problem-solving paradigm and workflow.
</summary>
    <author>
      <name>Michael L Brodie</name>
    </author>
    <arxiv:comment xmlns:arxiv="http://arxiv.org/schemas/atom">28 pages, 10 figures</arxiv:com
    <link href="http://arxiv.org/abs/2403.00776v1" rel="alternate" type="text/html"/>
    <link title="pdf" href="http://arxiv.org/pdf/2403.00776v1" rel="related" type="application
    <arxiv:primary_category xmlns:arxiv="http://arxiv.org/schemas/atom" term="stat.OT" scheme=
    <category term="stat.OT" scheme="http://arxiv.org/schemas/atom"/>
    <category term="stat.ME" scheme="http://arxiv.org/schemas/atom"/>
    <category term="I.2.0; I.2.8; E.0" scheme="http://arxiv.org/schemas/atom"/>
  </entry>
```

```python
def fetch_arxiv_data(query, start_index, max_results):
    url = "http://export.arxiv.org/api/query"
    params = {
        "search_query": query,
        "start": start_index,
        "max_results": max_results
    }

    response = requests.get(url, params=params)
    if response.status_code == 200:
        data_dict = xmltodict.parse(response.content)
        arxiv_data = data_dict.get('feed', {}).get('entry', [])
        return arxiv_data if isinstance(arxiv_data, list) else [arxiv_data]
```

additional 300 papers

```python
all_entries = []
batch_size = 100  # ArXiv API limits to 100 results per request
for start in range(0, 300, 100):
    batch_entries = fetch_arxiv_data("data science", start, batch_size)
    if not batch_entries:
        break  # Stop if no more data
    all_entries.extend(process_arxiv_data(batch_entries))
```

DATA PREP AND ML

```python
import pandas as pd
from collections import Counter
import os

# Path relative to the script's directory
script_dir = os.getcwd()
path = os.path.join(script_dir, "../Data-Sci-Project/localresults/extracted_data.csv")

# Load CSV into DataFrame
df = pd.read_csv(path)

def findsa(list):

    count = Counter(list)
    max_count = 0
    most_frequent_item = None

    # Iterate through the list and find the item with the highest count
    for item in list:
        if count[item] > max_count:
            max_count = count[item]
            most_frequent_item = item
        # If there's a tie in count, no need to update since we return the first one
    return most_frequent_item

df['Subject Areas'] = df['Subject Code'].str.split(', ').apply(findsa)

subject_map = {
    "AGRI": "Agricultural and Biological Sciences", "ARTS": "Arts and Humanities", "BIOC": "Biochemistry",
    "BUSI": "Business", "CENG": "Chemical Engineering",
    "CHEM": "Chemistry", "COMP": "Computer Science", "DECI": "Decision Sciences", "DENT": "Dentistry",
    "EART": "Earth and Planetary Sciences", "ECON": "Econometrics and Finance", "ENER": "Energy",
    "ENGI": "Engineering", "ENVI": "Environmental Science", "HEAL": "Health Professions",
    "IMMU": "Immunology and Microbiology", "MATE": "Materials science", "MATH": "Mathematics",
    "MEDI": "Medicine", "NEUR": "Neuroscience", "NURS": "Nursing",
    "PHAR": "Toxicology and Pharmaceutics", "PHYS": "Physics and Astronomy", "PSYC": "Psychology",
    "SOCI": "Social Sciences", "VETE": "Veterinary", "MULT": "Multidisciplinary"
}


df['Subject Areas'] = df['Subject Areas'].map(subject_map)
df.drop('Processed Words', axis=1, inplace=True)
print (df["Subject Areas"])
df.to_csv('C:/Users/PangSunatcha/OneDrive - Chulalongkorn University/Documents/Y2S1 files/Data Sci/proj/Data-Sci-project/results/formatted_subjectAreas2.csv', index=False)
```

format subject area to focus only on the first most relevant subject area of each book for simpler classification and use map to convert the subject codes into actual subject name

DATA PREP AND ML

```python
import pandas as pd
import os

# Path relative to the script's directory
script_dir = os.getcwd()
print (script_dir)
path = os.path.join(script_dir, "../results/data_formatted_subjectAreas.csv")

df = pd.read_csv(path)

df.dropna(subset=['Title', 'Abstract', 'Publication Name'], inplace=True)
```

use AI to predict Subject Area by training model with the given data

- drop rows with missing values needed for ml
- used “TF-IDF” (Term Frequency-Inverse Document Frequency) to classify words in Title, Abstract, Publication Name and remove English Stop Words.
- fit and predict with Logistic Regression
- yield final accuracy of **0.88**

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from scipy.sparse import hstack

# Encode the target variable
label_encoder = LabelEncoder()
df['subject_area_encoded'] = label_encoder.fit_transform(df['Subject Areas'])

# Split data into features and target
X = df[['Title', 'Abstract', 'Publication Name']]
y = df['subject_area_encoded']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define vectorizers for each column
title_vectorizer = TfidfVectorizer(max_features=20000, stop_words='english', max_df=0.1)
abstract_vectorizer = TfidfVectorizer(max_features=50000, stop_words='english', max_df=0.1)
publication_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english', max_df=0.1)

# Fit and transform each column separately for the training set
X_train_title = title_vectorizer.fit_transform(X_train['Title'])
X_train_abstract = abstract_vectorizer.fit_transform(X_train['Abstract'])
X_train_publication = publication_vectorizer.fit_transform(X_train['Publication Name'])

# Transform the test set using the same vectorizers
X_test_title = title_vectorizer.transform(X_test['Title'])
X_test_abstract = abstract_vectorizer.transform(X_test['Abstract'])
X_test_publication = publication_vectorizer.transform(X_test['Publication Name'])

# Combine the transformed columns using hstack
X_train_combined = hstack([X_train_title, X_train_abstract,  X_train_publication])
X_test_combined = hstack([X_test_title, X_test_abstract,  X_test_publication])

# Define and train the classifier
classifier = LogisticRegression(C=100, max_iter=500, penalty='l2', solver='saga', random_state=50)
classifier.fit(X_train_combined, y_train)

# Make predictions
y_pred = classifier.predict(X_test_combined)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

DATA PREP AND ML

we dropped the scopus api data where abstract cannot be fetched and predicted the subject areas for each dataset separately using the same trained model

```
# Preprocess the required columns from the scopus DataFrame
scopus_transformed_title = title_vectorizer.transform(scopus['title'].fillna(''))
scopus_transformed_abstract = abstract_vectorizer.transform(scopus['abstract'].fillna(''))
scopus_transformed_publication = publication_vectorizer.transform(scopus['publicationName'].fillna(''))

# Combine the features using hstack
scopus_combined_features = hstack([
    scopus_transformed_title,
    scopus_transformed_abstract,
    scopus_transformed_publication
])

# Use the trained model to predict subject areas
scopus_predictions = classifier.predict(scopus_combined_features)

# Decode the predicted labels to the original subject area names
scopus['predicted_subject_area'] = label_encoder.inverse_transform(scopus_predictions)

# Display the results
scopus[['title', 'predicted_subject_area']].head(10)
```

	title	predicted_subject_area
	Analysing trends of computational urban scienc...	Computer Science
	A landmark federal interagency collaboration t...	Social Sciences
	Regional planning: A failed or flawed project ...	Health Professions
	Data Science and Model Predictive Control:: A ...	Biochemistry
	Assessment of the relationship between central...	Medicine

scopus

```
# Preprocess the required columns from the scopus DataFrame
scopus_transformed_title = title_vectorizer.transform(arxiv['Title'].fillna(''))
scopus_transformed_abstract = abstract_vectorizer.transform(arxiv['Abstract'].fillna(''))
scopus_transformed_publication = publication_vectorizer.transform(arxiv['publicationName'].fillna(''))

# Combine the features using hstack
arxiv_combined_features = hstack([
    scopus_transformed_title,
    scopus_transformed_abstract,
    scopus_transformed_publication
])

# Use the trained model to predict subject areas
arxiv_predictions = classifier.predict(arxiv_combined_features)

# Decode the predicted labels to the original subject area names
arxiv['predicted_subject_area'] = label_encoder.inverse_transform(arxiv_predictions)

# Display the results
arxiv[['Title', 'predicted_subject_area']].tail(10)
```

0.0s

	Title	predicted_subject_area
0	Data Mining on Crash Simulation Data	Computer Science
1	The Value of Using Big Data Technologies in Co...	Computer Science
2	The LIGO Open Science Center	Social Sciences
3	Enabling Interactive Analytics of Secure Data ...	Computer Science
4	The Automated Data Extraction, Processing, and...	Medicine

arxiv

DATA PREP AND ML

```
import pandas as pd
data = pd.read_csv(os.path.join(script_dir, "../results/formatted_subjectAreas2.csv"))
scopus = pd.read_csv(os.path.join(script_dir, "../results/predicted_scopus.csv"))
arxiv = pd.read_csv(os.path.join(script_dir, "../results/predicted_arxiv.csv"))
```

1.4s

```
import json

Tabnine | Edit | Test | Fix | Explain | Document | Ask
def extract_cities(affiliations):
    try:
        affiliations = clean_json_string(affiliations)
        if isinstance(affiliations, list):
            return ', '.join([affil.get('city', '') for affil in affiliations])
    except (TypeError, AttributeError):
        return None
    return None

Tabnine | Edit | Test | Fix | Explain | Document | Ask
def extract_countries(affiliations):
    try:
        affiliations = clean_json_string(affiliations)
        if isinstance(affiliations, list):
            return ', '.join([affil.get('country', '') for affil in affiliations])
    except (TypeError, AttributeError):
        return None
    return None
```

0.0s

formatted City and Country by extracting from affiliations column in scopus api dataset

```
Tabnine | Edit | Test | Explain | Document | Ask
def clean_json_string(json_str):
    try:
        # Replace single quotes with double quotes for JSON compatibility
        json_str = json_str.replace("'", '"')
        # Parse the JSON string
        return json.loads(json_str)
    except (json.JSONDecodeError, TypeError):
        return None  # Return None if parsing fails
```

0.0s

```
scopus['City'] = scopus['affiliations'].apply(extract_cities)
scopus['Country'] = scopus['affiliations'].apply(extract_countries)
scopus
```

final data preparation for visualization

```
print(data.columns)
print(scopus.columns)
print(arxiv.columns)
```

0.0s

```
lex(['Title', 'Cover Date', 'Aggregation Type', 'Authors', 'Subject Code',
     'Subject Areas', 'Author Keywords', 'Abstract', 'Reference Count',
     'Publication Name', 'City', 'Country', 'Citation Count', 'Year'],
    dtype='object')
lex(['Unnamed: 0', 'title', 'author', 'publicationName', 'cover_date',
     'scopus_id', 'cited_by_count', 'open_access', 'eid', 'aggregationType',
     'affiliations', 'link', 'abstract', 'predicted_subject_area',
     'author_keywords', 'City', 'Country'],
    dtype='object')
lex(['Unnamed: 0', 'ID', 'Title', 'Abstract', 'Authors', 'Published Date',
     'Updated Date', 'Comments', 'Primary Category', 'PDF Link', 'Language',
     'author_keywords', 'publicationName', 'predicted_subject_area'],
    dtype='object')
```

```
data = data.rename(columns={'Subject Areas':'Subject Area'}).drop(['Subject Code','Year'],axis=1)
df2 = scopus.rename(columns={'title': 'Title', 'author': 'Authors',
                             'publicationName':'Publication Name','cover_date':'Cover Date',
                             'cited_by_count':'Citation Count','aggregationType':'Aggregation Type',
                             'abstract':'Abstract','predicted_subject_area':'Subject Area',
                             'author_keywords':'Author Keywords'
                             }).drop(['Unnamed: 0','scopus_id','open_access','eid','affiliations','link',], axis=1)

df3 = arxiv.rename(columns={ 'Published Date':'Cover Date','author_keywords':'Author Keywords',
                            'publicationName':'Publication Name', 'predicted_subject_area':'Subject Area'
                            }).drop(['ID','Unnamed: 0', 'PDF Link', 'Language', 'Updated Date', 'Comments', 'Primary Category'],axis =1)

# Identify the union of columns
all_columns = sorted(set(data.columns) | set(df2.columns) | set(df3.columns))

# Align columns for each DataFrame to the union of columns
df1 = data.reindex(columns=all_columns)
df2 = df2.reindex(columns=all_columns)
df3 = df2.reindex(columns=all_columns)

# Concatenate the DataFrames
combined_df = pd.concat([df1, df2, df3], axis=0, ignore_index=True)

# Check the shape of the combined DataFrame to confirm
print(combined_df.shape)

# Display the first few rows of the combined DataFrame
combined_df.columns
```

rename columns from the given data, scopus api data, and arxiv api data to match and drop unused columns **then join all datasets together**

VISUALIZE

```python
def parse_city_column(city_value):
    try:
        return ast.literal_eval(city_value) if isinstance(city_value, str) else []
    except (ValueError, SyntaxError):
        return []
```

convert city into geographical coordinates

```python
def parallel_geocode(cities):
    results = []
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        futures = {executor.submit(geocode_city, city): city for city in cities}
        for future in concurrent.futures.as_completed(futures):
            city = futures[future]
            try:
                result = future.result()
                results.append((city, *result))
            except Exception as e:
                results.append((city, None, None))
    return results
```

```python
    fig = px.scatter_mapbox(city_counts,
                            lat='latitude',
                            lon='longitude',
                            size='Count',
                            hover_name='City',
                            hover_data=['Count'],
                            mapbox_style="open-street-map",
                            title=f"Map of {selected_subject_code}")


    fig.update_layout(mapbox_accesstoken=mapbox_access_token)


    st.plotly_chart(fig)
```

using Plotly and Streamlit to display map and data

```python
with col2:

    city_counts = city_counts[['City', 'Count']]
    city_counts.set_index('City', inplace=True)

    st.write(f"City Counts for Subject: {selected_subject_code}")
    st.dataframe(city_counts)
```

MAP

Choose a page:

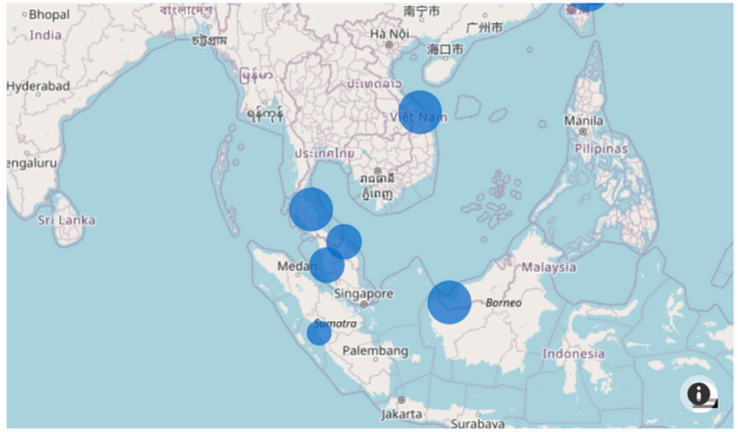
Subject Areas Popularity

Select Subject Area

Agricultural and Biological Scien…

- Agricultural and Biological Sciences
- Arts and Humanities
- Biochemistry
- Business
- Chemical Engineering
- Chemistry
- Computer Science
- Decision Sciences

Map of Agricultural and Biological Sciences



City Counts for Subject: Agricultural and Biological Sciences

City	Count
Nakhon si Thammarat	3
Da Nang	3
Taoyuan	3
Kota Samarahan	3
North Bangkok	2
Nottingham	2
Mumbai	2
Minneapolis	2
Kawaguchi	2
Keelung	2

THANK YOU!