Analysis of Paper Titles

20191449 강병무

As a senior college student, I began to contemplate my future career and developed an interest in pursuing graduate studies. The diversity of research fields in graduate programs intrigued me, leading me to wonder which academic disciplines were most active in research and what topics were currently trending. To address these questions, I embarked on a text mining project to analyze paper titles from various departments.

Research Objectives

The primary research objectives were to identify topics studied intensively over time and to determine which subjects were explored by multiple academic departments. To collect the necessary data, I utilized Arxiv, a free paper repository maintained by Cornell University, which covers a wide array of disciplines, including physics, mathematics, and computer science.

To get the data for the papers, I used site named Arxiv. Arxiv is free paper save site that is runned by Connel university. It has various papers with physics, math, computer science.

Data Collection

To begin, I decided to focus on my major, Electrical Engineering. Arxiv lumps Electrical Engineering and Systems Science together, providing access to papers from 2017 to 2023. I set out to gather 1000 paper titles from each of these years, using Python and BeautifulSoup to scrape the titles. The following code exemplifies how the titles were collected:

#Electrical Engineering and Systems Science

import requests

from bs4 import BeautifulSoup

# Initialize a variable to keep track of the number of titles to scrape

titles\_to\_scrape = 1000

titles\_scraped = 0

years = [23, 22, 21, 20, 19, 18, 17]

# Create separate lists for each year using globals()

for year in years:

    globals()[f'title\_list\_20{year}'] = []

for year in years:

    titles\_scraped = 0

    page\_number = 0

    while titles\_scraped < titles\_to\_scrape:

        # Construct the URL with the appropriate year and page number

        area\_url = f"https://arxiv.org/list/eess/{year}?skip={25 \* page\_number}&show=25"

        response = requests.get(area\_url)

        if response.status\_code == 200:

            soup = BeautifulSoup(response.text, 'html.parser')

            # Find all elements with the class "list-title mathjax"

            elements = soup.find\_all('div', class\_='list-title mathjax')

            # Iterate through the elements and extract the titles

            for element in elements:

                title = element.span.next\_sibling.strip()  # Removes the "Title:" prefix

                titles\_scraped += 1

                globals()[f'title\_list\_20{year}'].append(title)

                if titles\_scraped == titles\_to\_scrape:

                    break

            page\_number += 1

Data Preprocessing

To make the data more amenable to analysis, I performed several preprocessing steps. I converted all words to lowercase, removed stop words using the NLTK library, and applied the Porter Stemming algorithm to reduce words to their root form. The following code snippet illustrates the preprocessing process:

import nltk

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

stemmer = PorterStemmer()

years = [23, 22, 21, 20, 19, 18, 17]

for year in years:

    title\_list = globals()[f'title\_list\_20{year}']

    processed\_title\_list = []

    # Process titles in the original list

    for title in title\_list:

        # Convert to lowercase

        title = title.lower()

        # Tokenize the title

        tokens = word\_tokenize(title)

        # Perform stemming

        stemmed\_tokens = [stemmer.stem(token) for token in tokens]

        # Join the stemmed tokens to form the processed title

        processed\_title = ' '.join(stemmed\_tokens)

        processed\_title\_list.append(processed\_title)

    # Store the new list of processed titles

    globals()[f'processed\_title\_list\_20{year}'] = processed\_title\_list

Frequent Word Analysis

To gain insights from the paper titles, I conducted a frequent word analysis. This involved identifying the top 20 most common words in the titles and creating word clouds for visualization. Below are the results for the years 2023 and 2017:

from wordcloud import WordCloud

for year in years:

    processed\_title\_list = globals()[f'processed\_title\_list\_20{year}']

    # Combine processed titles into a single string

    all\_titles = ' '.join(processed\_title\_list)

    # Tokenize the combined text

    tokens = word\_tokenize(all\_titles)

    # Create a frequency distribution of words

    word\_freq = nltk.FreqDist(tokens)

    top\_20\_words = Counter(tokens).most\_common(20)

    # Print the top 20 words

    print(f"Top 20 Most Frequent Words for Year 20{year}:")

    for word, freq in top\_20\_words:

        print(f"{word}: {freq}")

    # Create a separate figure for the word frequency graph

    plt.figure(figsize=(12, 6))

    word\_freq.plot(30, cumulative=False)

    plt.show()

    # Create a separate figure for the word cloud

    wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(all\_titles)

    plt.figure(figsize=(12, 6))

    plt.imshow(wordcloud, interpolation='bilinear')

    plt.axis("off")

    plt.title(f"Word Cloud for Year 20{year}")

    plt.show()

Top 20 Most Frequent Words for Year 2023:

system: 142

learn: 123

network: 123

imag: 102

model: 102

use: 98

control: 85

deep: 77

neural: 60

data: 57

base: 56

detect: 54

commun: 51

optim: 51

analysi: 49

via: 41

predict: 41

gener: 40

estim: 39

power: 37

라인, 그래프, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 그래픽, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

Top 20 Most Frequent Words for Year 2017:

network: 173

use: 141

deep: 108

learn: 102

speech: 92

detect: 90

system: 84

base: 80

music: 77

analysi: 77

model: 75

imag: 75

audio: 69

neural: 67

eeg: 65

event: 58

gener: 57

classif: 55

approach: 52

estim: 51

그래프, 라인, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 타이포그래피, 디자인이(가) 표시된 사진

자동 생성된 설명

These results indicate that words such as "network," "system," "model," and "image" consistently appear in paper titles. Additionally, I observed the rise of terms related to deep learning over time. However, due to the limited time span of the data, significant changes in title keywords were not prominent.

Note: Due to the extensive volume of results generated during the analysis, it was impractical to include the complete set of findings in this report. Therefore, only a subset of results for specific years is presented here as a representative sample. To access the complete set of results and a more detailed analysis, please refer to the attached IPython Notebook file (.ipynb) for a comprehensive view of the project's findings.

Interdisciplinary Comparison

To explore how title keywords differed across academic departments, I extended the analysis to include papers from the mathematics department. The mathematics papers spanned from 1992 to 2023, providing a more extensive timeframe for investigation. I collected 200 titles from each year and employed the same preprocessing and analysis techniques.

import requests

from bs4 import BeautifulSoup

import nltk

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from wordcloud import WordCloud

import matplotlib.pyplot as plt

from collections import Counter

import string

import re

nltk.download('punkt')

nltk.download('stopwords')  # Download NLTK stopwords

# Initialize a variable to keep track of the number of titles to scrape

titles\_to\_scrape = 200

titles\_scraped = 0

# Year range from 1992 to 2023

years = list(range(92, 100)) + [f"{i:02}" for i in range(24)] # Updated year range

# Create a single list to store titles

math\_title\_list = []

# Define stopwords to remove

stop\_words = set(stopwords.words('english'))

for year in years:

    titles\_scraped = 0

    page\_number = 0

    while titles\_scraped < titles\_to\_scrape:

        # Construct the URL with the appropriate year and page number

        area\_url = f"https://arxiv.org/list/math/{year}?skip={25 \* page\_number}&show=25"

        response = requests.get(area\_url)

        if response.status\_code == 200:

            soup = BeautifulSoup(response.text, 'html.parser')

            # Find all elements with the class "list-title mathjax"

            elements = soup.find\_all('div', class\_='list-title mathjax')

            # Iterate through the elements and extract the titles

            for element in elements:

                title = element.span.next\_sibling.strip()  # Removes the "Title:" prefix

                titles\_scraped += 1

                math\_title\_list.append(title)

                if titles\_scraped == titles\_to\_scrape:

                    break

            page\_number += 1

stemmer = PorterStemmer()

for year in years:

    title\_list = math\_title\_list

    processed\_title\_list = []

    # Process titles in the original list

    for title in title\_list:

        # Convert to lowercase

        title = title.lower()

        # Remove punctuation and tokenize the title

        tokens = word\_tokenize(title)

        tokens = [token for token in tokens if token not in string.punctuation]

        # Remove possessive forms like " 's "

        tokens = [re.sub(r"(\w+)'s", r"\1", token) for token in tokens]

        # Remove stopwords

        filtered\_tokens = [word for word in tokens if word not in stop\_words]

        # Perform stemming

        stemmed\_tokens = [stemmer.stem(token) for token in filtered\_tokens]

        # Join the stemmed tokens to form the processed title

        processed\_title = ' '.join(stemmed\_tokens)

        processed\_title\_list.append(processed\_title)

    # Store the new list of processed titles

    math\_title\_list = processed\_title\_list

for year in years:

    processed\_title\_list = math\_title\_list

    # Combine processed titles into a single string

    all\_titles = ' '.join(processed\_title\_list)

    # Tokenize the combined text

    tokens = word\_tokenize(all\_titles)

    # Create a frequency distribution of words

    word\_freq = nltk.FreqDist(tokens)

    top\_20\_words = Counter(tokens).most\_common(20)

    # Print the top 20 words

    print(f"Top 20 Most Frequent Words for {year}:")

    for word, freq in top\_20\_words:

        print(f"{word}: {freq}")

    # Create a separate figure for the word frequency graph

    plt.figure(figsize=(12, 6))

    word\_freq.plot(30, cumulative=False)

    plt.show()

    # Create a separate figure for the word cloud

    wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(all\_titles)

    plt.figure(figsize=(12, 6))

    plt.imshow(wordcloud, interpolation='bilinear')

    plt.axis("off")

    plt.title(f"Word Cloud for {year}")

    plt.show()

Top 20 Most Frequent Words for 92:

algebra: 535

group: 499

space: 404

equat: 350

function: 328

gener: 295

theori: 248

oper: 243

system: 242

quantum: 242

problem: 238

theorem: 212

polynomi: 196

model: 192

structur: 182

invari: 182

map: 180

surfac: 177

graph: 177

그래프, 라인, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 그래픽, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

Top 20 Most Frequent Words for 23:

algebra: 535

group: 499

space: 404

equat: 350

function: 328

gener: 295

theori: 248

oper: 243

system: 242

quantum: 242

problem: 238

theorem: 212

polynomi: 196

model: 192

structur: 182

invari: 182

map: 180

surfac: 177

graph: 177

그래프, 라인, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 그래픽, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

The results of this interdisciplinary comparison revealed that mathematical research maintained a certain level of consistency over time. The most frequent words in mathematics titles included "algebra," "function," "space," and "equation." This observation aligns with the formal and abstract nature of mathematical research.

Conclusion

In summary, this text mining project unveiled several insights into academic research trends based on paper titles. While certain keywords remained relatively consistent over time, such as "network" in electrical engineering and "algebra" in mathematics, the project provided a valuable perspective on the dynamism and stability of research in different fields.

This analysis serves as a foundation for making informed decisions about graduate school applications and future career paths. Moreover, it highlights the potential for text mining to reveal underlying patterns and trends within vast academic repositories like Arxiv.

The project showcases the potential for text mining to uncover hidden patterns within extensive academic datasets, helping researchers and academics make more informed decisions regarding their fields of study.