

What Are The Top Ten Keywords NY Times Used When Reporting AI at Different Periods?

Introduction

The release of ChatGPT marks a significant shift in the relationship between AI and humanity. Its powerful capabilities have rapidly permeated various aspects of our lives, production processes, and other societal activities. Concurrently, the rapid development of AI has shifted our focus and perceptions of it.

This leads to my question: How has the New York Times covered AI over specific periods?

Specifically, what are the ten most frequent keywords associated with AI in New York Times articles during different periods? To explore this, my hypothesis examines three time frames: 2018–2020, 2020–2022, and 2022–2024.

For early prediction, I predict that from 2018 to 2020, AI will be primarily associated with keywords related to its developmental trajectory. From 2020 to 2022, the focus will shift to its implications for society and law. However, from 2022 to 2024, the association will likely pivot toward software and companies with the rise of ChatGPT and other advanced AI models.

Considerations

For this analysis, only articles were considered and sorted by relevance. To maintain efficiency and avoid long run times, the analysis was limited to the first ten pages of search results for each period unless fewer pages were available.

It is important to note that the analysis is confined to three specific time periods:

- **2018/01/01 to 2020/01/01**
- **2020/01/01 to 2022/01/01**
- **2022/01/01 to 2024/01/01**


After completing the analysis for the first period, ensure that the chart window is closed before proceeding to the next period.

ETL Process

The following section outlines the code process and logic used to analyze the data and draw conclusions.

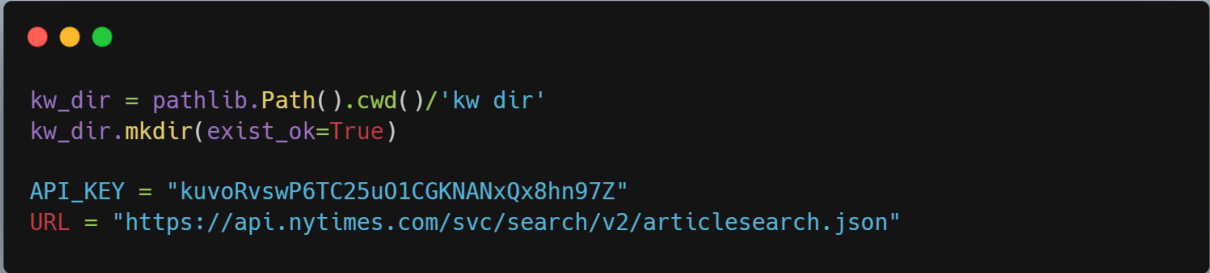
Extracting the Data

This project is designed for both online access to information and local file processing. To support these functionalities, libraries for file handling, data processing, and client-side operations were utilized. For more details, the code snippet below lists the imported libraries. Notably, matplotlib.pyplot was included to generate a bar chart as the final output.



```
import requests,pickle,pathlib,time
from math import ceil
import pandas as pd
import matplotlib.pyplot as plt
```

After importing the necessary libraries, a directory is created to store the subsequent files. Additionally, the URL and API key are defined as constants to ensure they can be accessed easily throughout the code.



```
kw_dir = pathlib.Path().cwd()/'kw_dir'
kw_dir.mkdir(exist_ok=True)

API_KEY = "kuvoRvswP6TC25u01CGKNANxQx8hn97Z"
URL = "https://api.nytimes.com/svc/search/v2/articlesearch.json"
```

The main program begins here. In this section, I first define the time periods for the search and store them in a list. Then, a for loop is used to iterate through the dates, placing them into the parameters to complete the API setup. Next, the program checks whether a corresponding file already exists in the folder for the given time period (indicating that the data extraction has already been completed). If the file exists, it is analyzed locally (see page 5). Otherwise, the API is used to extract the data before proceeding with the analysis (see page 5).

```

begin_end_dates = [[20180101,20200101],[20200101,20220101],[20220101,20240101]]

for period in begin_end_dates:
    begin_date = period[0]
    end_date = period[1]

    params = {
        'q': "AI",
        'begin_date': begin_date,
        'end_date': end_date,
        'api-key': API_KEY,
        'sort': 'relevance',
        'fq' : 'document_type:(\"article\")',
    }

    print()
    print(f"Now viewing the period of {begin_date} ~ {end_date}.")

    file_path = kw_dir / f"{begin_date}_{end_date}_keywords.pkl"
    if file_path.exists():
        data_analysis(begin_date, end_date)
    else :
        get_docs_between_dates(URL,params)
        data_analysis(begin_date, end_date)

```

After I get the correct result from the previous image. When this function runs and the API responds correctly, it will get the data in 'docs' according to the page. Otherwise, return the error code.

```

def get_hits(URL,params):
    response = requests.get(URL, params=params)
    if response.status_code == 200:
        data = response.json()
        return data['response']['meta']['hits']
    else:
        return "API request failed with status code " + str(response.status_code)

```

After obtaining the correct result from the previous step, this function retrieves the data from the 'docs' section of the API response based on the specified page, provided the API responds correctly. If the API encounters an error, the function returns the corresponding error code.

```
def get_docs(page, params, URL):
    params['page'] = page
    response = requests.get(URL, params=params)
    if response.status_code == 200:
        data = response.json()
        return data['response']['docs']
    else:
        return "API request failed with status code " + str(response.status_code)
```

Transforming the Data

As shown in the image, retrieving data from the API involves two key functions. The `get_hits` function (see page 3) is used to calculate the total number of pages in the search results, while the `get_docs_between_dates` function (see page 3) retrieves all the data from the 'docs' section on a specific page.

After obtaining the data, the program begins parsing the 'docs' section to extract keywords, which are stored in a list. Once all pages have been processed, the list contains all keywords from the search results. Finally, `pathlib` is used to save the list into a `.pkl` file, completing the local data storage process.

```
def get_docs_between_dates(URL, parmas):
    max_page = min(10, ceil(get_hits(URL, parmas)/10))
    keyword_list = []
    for page in range(max_page):
        docs = get_docs(page, params, URL)
        for doc in docs:
            article_keywords = doc['keywords']
            for keyword in article_keywords:
                value = keyword['value']
                keyword_list.append(value)

        time.sleep(15)
        print(f"Page completed search: {page+1}")

    file_name = f"{begin_date}_{end_date}_keywords.pkl"
    file_path = kw_dir/file_name
    with file_path.open(mode='wb') as file:
        pickle.dump(keyword_list, file)
```

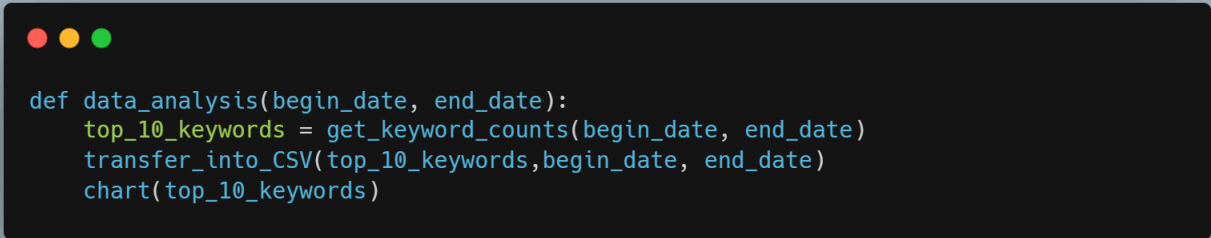
After I store data locally, this function transforms data in the PKL file into human-readable content for people to analyze it. It includes three functions: "`get_keyword_counts`", "`transfer_into_CSV`", and

“chart”. The first one filters out the top 10 keywords. The second one transfers the dictionary into a CSV file. The third one draws a chart.

After storing the data locally, this function transforms the data from the .pkl file into human-readable content for analysis. It utilizes three key functions:

- **get_keyword_counts:** Filters and identifies the top 10 keywords.
- **transfer_into_CSV:** Converts the keyword dictionary into a CSV file.
- **chart:** Generates a chart based on the data.

These steps ensure the original data is transferred into data that is easily accessible and visually interpretable for further analysis.



```
def data_analysis(begin_date, end_date):  
    top_10_keywords = get_keyword_counts(begin_date, end_date)  
    transfer_into_CSV(top_10_keywords, begin_date, end_date)  
    chart(top_10_keywords)
```

Loading The Data

The primary objective of this function is to load the data and filter out the top ten keywords. To achieve this, the process involves three main steps:

- **Loading the PKL file:** The .pkl file is read and its data is loaded into a list.
- **Creating a dictionary:** Using the list, a dictionary is generated to count the occurrences of each keyword.
- **Sorting and filtering:** The dictionary is sorted in descending order based on keyword counts. Finally, the top ten keywords are selected and stored in a new dictionary.

In summary, the data transformation follows this flow:

- **PKL file → List → Dictionary → Sorted Dictionary → Dictionary with Top Ten Keywords.**

```

def get_keyword_counts(begin_date, end_date):
    keyword_values = {}
    kw_file = kw_dir / f"{begin_date}_{end_date}_keywords.pkl"
    with kw_file.open(mode='rb') as file:
        kw_list = pickle.load(file)
    for keyword in kw_list:
        if keyword in keyword_values:
            keyword_values[keyword] += 1
        else:
            keyword_values[keyword] = 1

    sort_keyword_values = {k:v for k,v in sorted(keyword_values.items(), key = lambda
kv:kv[1], reverse= True)}

    top_10_keywords = dict(list(sort_keyword_values.items())[:10])

    return top_10_keywords

```

After obtaining the top ten keywords, this function is executed. Its primary purpose is to use Pandas to load data from the top ten keywords dictionary into a human-readable CSV file.

The process involves the following steps:

- **Creating a DataFrame:** The `pd.DataFrame` function is used to create a DataFrame containing the keywords from the `top_10_keywords` dictionary, their corresponding counts, and column names.
- **Naming the CSV file:** A name is assigned to the CSV file.
- **Saving the DataFrame:** The DataFrame is saved as a CSV file, completing the process.

```

def transfer_into_CSV(top_10_keywords, begin_date, end_date):
    word_count_df = pd.DataFrame(list(top_10_keywords.items()), columns=['Keyword', 'Count'])
    final_result = kw_dir / f"{begin_date}_{end_date}_word_counts.csv"
    word_count_df.to_csv(final_result, index=False)
    print(word_count_df)

```

After generating the CSV file, the chart function provides a visual representation of the data by drawing a bar chart based on the top 10 keywords in the dictionary.

The process involves the following steps:

- **Extracting data:** Two lists are created to store the dictionary's content—one for keys (keywords) and the other for values (counts).

- **Setting axis labels:** The x-axis is labeled using the keys list, and the y-axis is labeled using the values list.
- **Optimizing the display:** The `xticks()` and `tight_layout()` functions are applied to enhance the chart's display and readability when rendered.

This ensures the data is both visually appealing and easy to interpret.

```
def chart (top_10_keywords):  
    key = list(top_10_keywords.keys())  
    values = list(top_10_keywords.values())  
  
    plt.bar(key, values, color='skyblue')  
  
    plt.title(f'On Page {0} ~ {9}, Top Ten Common Keywords In AI Articles During {begin_date} ~  
{end_date}')  
    plt.xlabel('keywords')  
    plt.ylabel('Amount')  
  
    plt.xticks(rotation=45, ha='right')  
    plt.tight_layout()  
    plt.show()
```

Finding

By analyzing the generated charts and CSV files, my assumptions are largely validated.

- **2018/01/01 to 2020/01/01:** Several keywords indicate areas where AI was expected to develop, supporting the idea that this period focused on the future direction of AI.
- **2020/01/01 to 2022/01/01:** Keywords related to society and law ranked high in frequency, suggesting that AI became increasingly associated with societal and legal aspects during this time.
- **2022/01/01 to 2024/01/01:** Keywords shifted toward companies, researchers, and entrepreneurs, highlighting a big change in focus. This suggests that the development of AI began to emphasize corporate products and innovations, reflecting its transition into profitability and widespread application.

Observations and Insights:

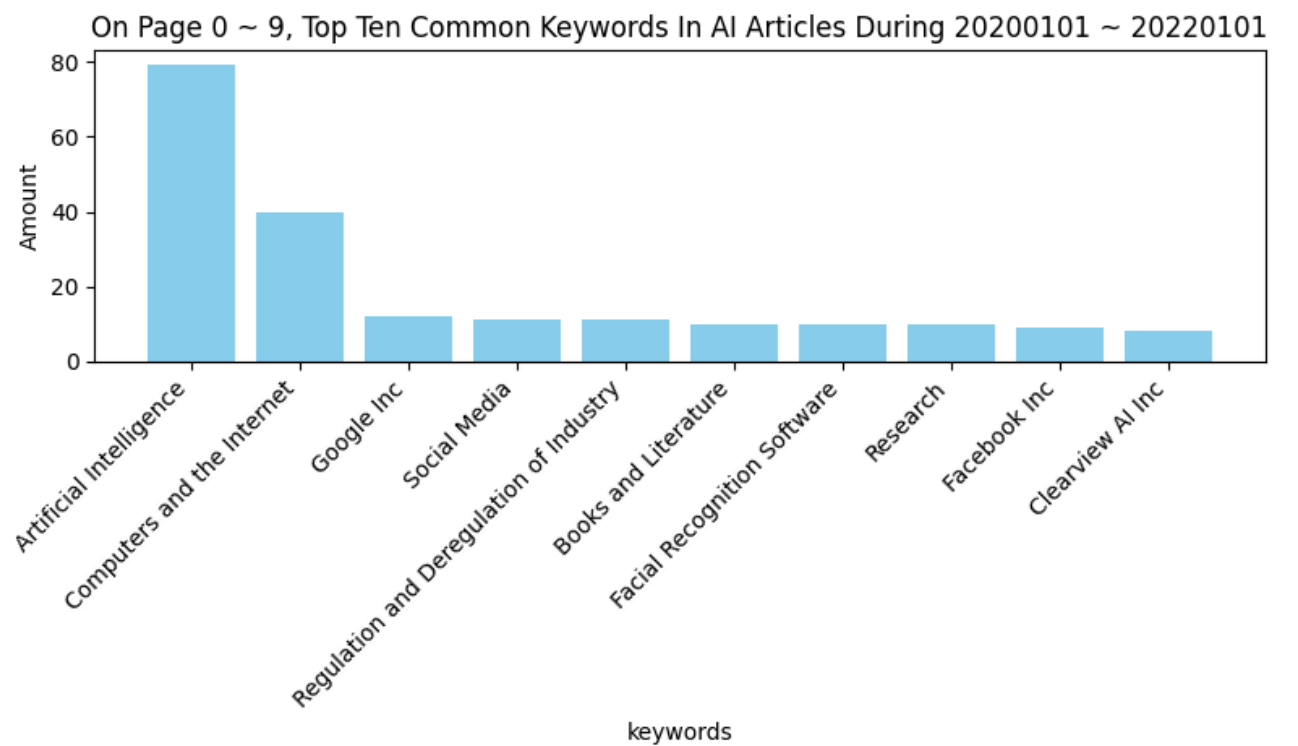
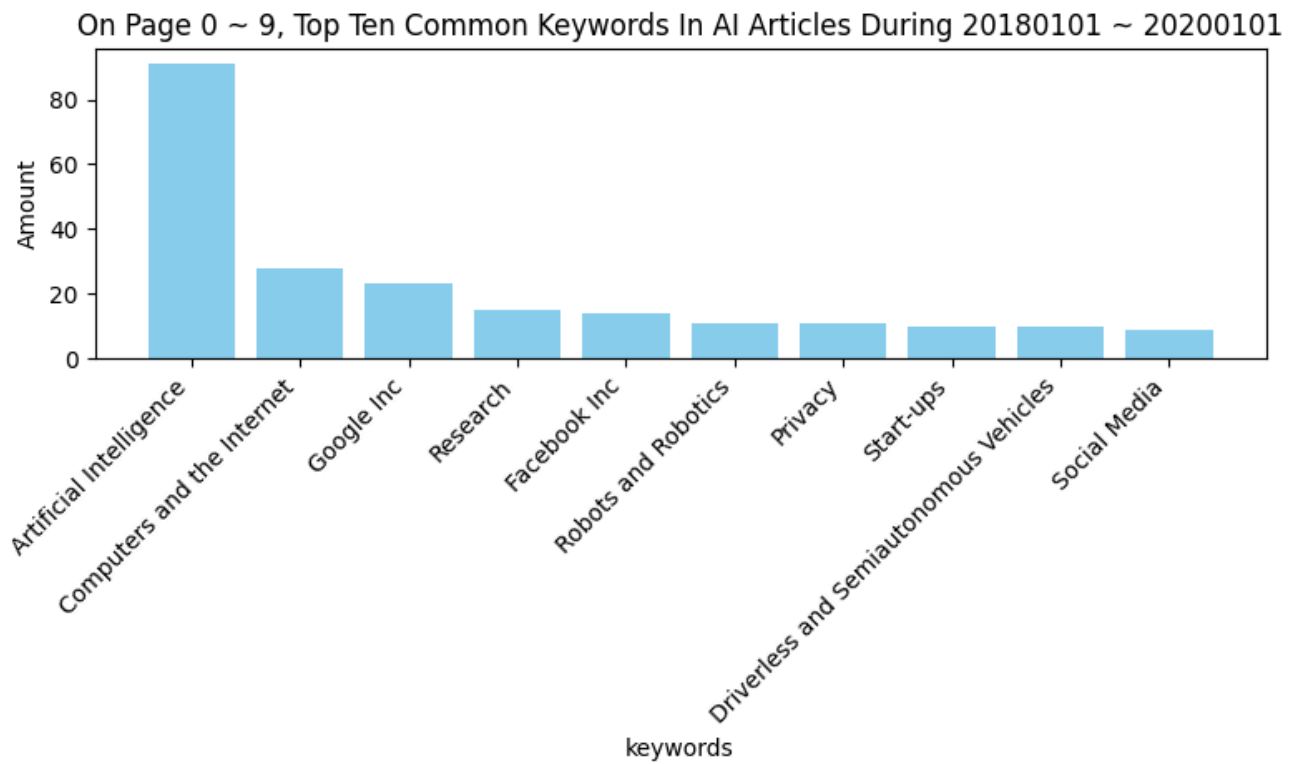
By comparing keywords across these three periods, we can observe a clear progression in AI development:

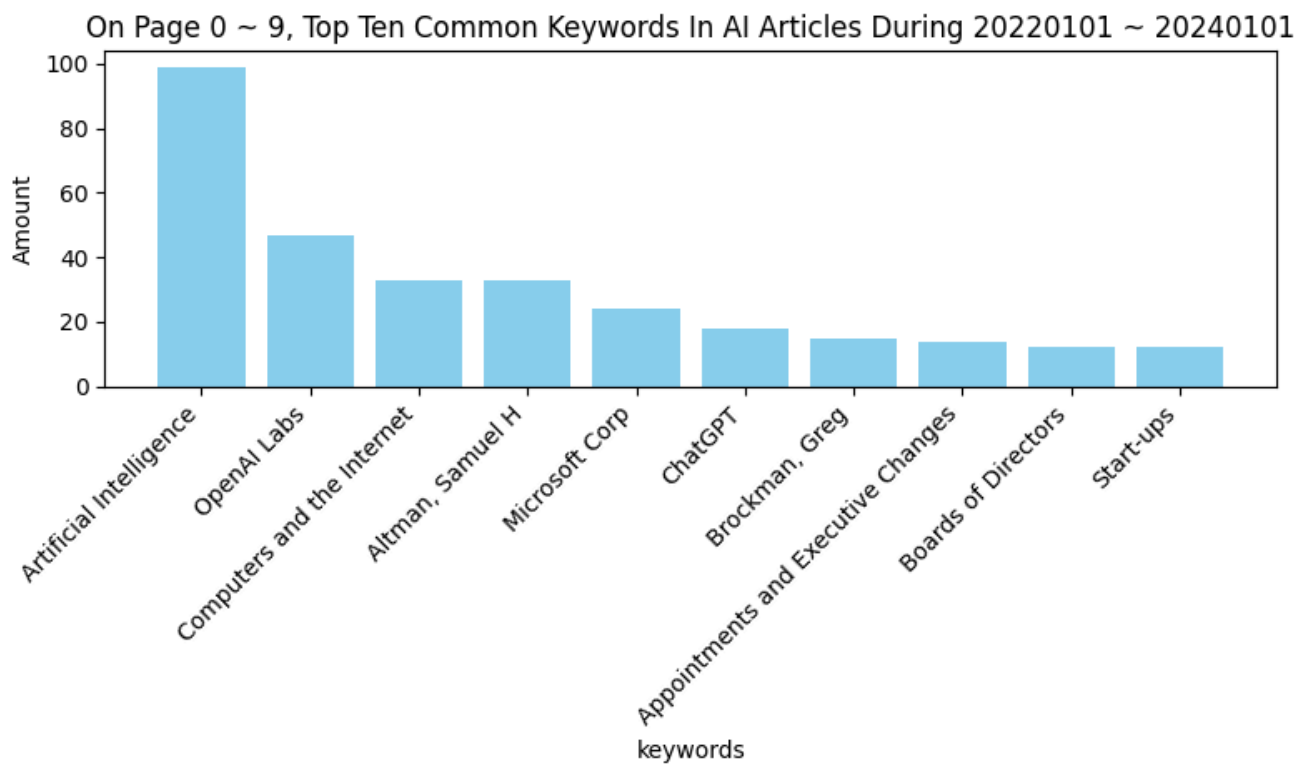
- **2018–2020:** Exploration of AI's future development directions.
- **2020–2022:** Discussions about integrating AI into human society.
- **2022–2024:** A shift from research to profitable AI products led by companies.

An Additional Insight:

An interesting trend emerges when examining Facebook’s ranking over these periods. Its consistent decline, eventually dropping out of the rankings altogether, reflects the company’s lagging position in the AI development race.

Below are the charts illustrating the top keywords for each period.





Next Step

The core of this report is centered on identifying the top ten keywords associated with AI articles within a given time period. The next phase will involve conducting a secondary search based on these keywords. Specifically, the search will analyze articles where both “AI” and the respective keyword appear, focusing on fields such as the article’s abstract, snippet, or lead paragraph. This textual analysis aims to extract more valuable insights about AI and its connection to each keyword, including but not limited to identifying development trajectories, usage trends, or conducting sentiment analysis. Thereby, additional libraries and third-party packages could be added for the next step, such as TextBlob and large language models, which will be employed to enhance the analytical operations and extract more profound insights.