New York Times Archives Data Analysis and Visualization using Prefect, DuckDB, and Streamlit

By Katie Hutchinson
DS 3022: Data Engineering

Project Summary

This project is an end-to-end data pipeline that analyzes 10 years of New York Times headlines from 2015–2025 in over 400,000 records to uncover shifts in public discourse.

By processing unstructured text, the system identifies how dominant cultural topics, like "COVID-19," "Elections," and "AI, "have risen and fallen over the decade.


"All the News That's Fit to Print"

The New York Times

LATE CITY EDITION

VOL. CXIX ... No. 40,097 NEW YORK, THURSDAY, APRIL 30, 1970 10 CENTS

PRICES CLIMB 0.4% BUT RATE OF RISE APPEARS TO SLOW

Millions Join Earth Day Observances Across the Nation

CAMBODIAN CRISIS GROWS AS TROOPS SEEM TO FALTER

U.S. CONCERN SUED WITH 2 IN JAPAN

Mood Is Joyful as City Gives Its Support

Activity Ranges From Oratory to Legislation

BAKERS OF BOJAS THREATEN REVOLT

U.S. Plane Flies In Arms As Trinidad Fights Mutiny

Technologies Used

- ★ **Prefect**: manages the workflow, ensuring resilient data processing and error handling.
- ★ **Pandas**: for efficient data manipulation.
- ★ **BERTopic**: clusters semantic meaning from headlines to discover topics
- ★ **DuckDB**: provides a high-performance, serverless SQL database optimized for analytical queries
- ★ **Streamlit and Plotly**: power an interactive frontend dashboard for real-time trend exploration.
- ★ **Conda/Mamba:** Manages the project's virtual environment and ensures reproducible dependency handling.


Project Steps

1. **Ingestion.py**: Obtain & Process Data from New York Times Archives from 2015 - 2025 using API Key
2. **Pipeline.py**: Transform raw text into structured insights
    a. Load the raw CSV into a Pandas DataFrame, verifying file integrity.
    b. Trained the BERTopic model on a randomized subset (20,000 articles).
    c. The trained model then predicts and assigns a Topic ID to the remaining 380,000+ articles.
3. **App.py**: Visualize the data using Plotly
    a. Using Streamlit, create the app with a drop-down menu and trend chart
    b. Use Plotly to plot individual lines
    c. View the key insights for each topic

Step 1: Obtain & Process Data from Archives

```python
import requests
import pandas as pd
import time
from datetime import datetime
import os

# --- CONFIGURATION ---
API_KEY = "MKyesAaNHHea0smCGlNCQ9x45gvfxLpr"
BASE_URL = "https://api.nytimes.com/svc/archive/v1/{year}/{month}.json"

# Fetches the entire archive for a specific month
def fetch_month_archive(year, month):

    url = BASE_URL.format(year=year, month=month)
    params = {'api-key': API_KEY}

    print(f"Fetching Archive: {year}-{month:02d} ...")

    try:
        response = requests.get(url, params=params)

        if response.status_code == 429:
            print("Rate limit hit. Waiting 60 seconds...")
            time.sleep(60)
            response = requests.get(url, params=params)

        response.raise_for_status()
        data = response.json()

        docs = data['response']['docs']
        print(f"  -> Found {len(docs)} articles.")
        return docs

    except Exception as e:
        print(f"Error fetching {year}-{month}: {e}")
        return []
```

First, I fetched all of the archives from the year 2020-2025 from the NYT Archives API. In order to comply with the request limit, I slept 60 seconds when the limit was hit.

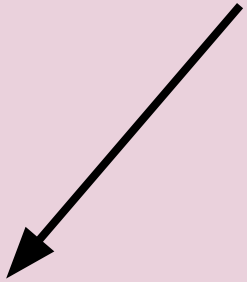
Next, I cleaned the data by filtering for just news articles. Then I added each cleaned line with the following data (date, headline, etc.) to a list of dictionaries.

```python
def clean_archive_data(docs):
    cleaned = []
    for doc in docs:
        if doc.get('type_of_material') not in ['News', 'Article', 'Review', 'Op-Ed']: # Filter for just "News" type material
            continue

        cleaned.append({
            'date': doc.get('pub_date'),
            'headline': doc.get('headline', {}).get('main', ''),
            'abstract': doc.get('abstract'),
            'keywords': [k['value'] for k in doc.get('keywords', [])],
            'section': doc.get('section_name')
        })
    return cleaned
```

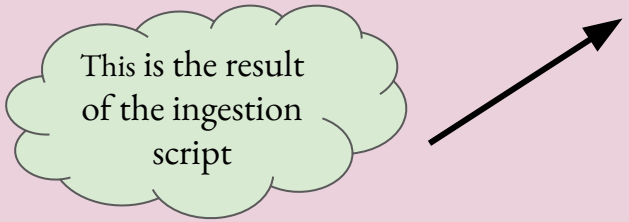
Step 1: Main Method

```python
if __name__ == "__main__":
    all_data = []

    years = range(2020, 2025)

    now = datetime.now()
    current_year = now.year
    current_month = now.month

    for year in years:
        for month in range(1, 13):
            if year > current_year or (year == current_year and month > current_month):
                break

            raw_docs = fetch_month_archive(year, month)
            clean_docs = clean_archive_data(raw_docs)
            all_data.extend(clean_docs)

            print(f"  -> Total collected so far: {len(all_data)}")

            time.sleep(5)

    df = pd.DataFrame(all_data)
    filename = "nyt_2020_2024.csv"
    df.to_csv(filename, index=False)

    print(f"Done! Saved {len(df)} articles to {filename}")
```

To finish the ingestion script, I looped through the years 2020-2025 to fetch and clean the archival data, sleeping 5 seconds between each month. I saved all this data to a csv file.



```
date,headline,abstract,keywords,section
2024-01-01T00:25:32+0000,PGA Tour and Saudi-Backed LIV Extend Deadline to Finalize Deal,"The tentative deal for the men's golf circuits to join forces had a Dec. 31 deadline, but significant questions remained.","['Gol
2024-01-01T00:32:06+0000,Something to Whistle,Harry Zheng makes his New York Times debut.,"['Crossword Puzzles', 'Games']",Crosswords & Games
2024-01-01T00:53:23+0000,"U.S. Helicopters Sink 3 Houthi Boats in Red Sea, Pentagon Says","Iranian-backed Houthi gunmen from Yemen had fired on American helicopters responding to an attack on a commercial ship, America
2024-01-01T02:09:45+0000,"In Times Square, Hundreds of Thousands Ring In 2024",New Year's celebrations took place as protesters against the Israel-Hamas war staged demonstrations in Midtown Manhattan.,"['New Year', 'De
2024-01-01T05:01:23+0000,"Half of Gazans Are at Risk of Starving, U.N. Warns","More than 90 percent of Palestinians in the territory say they have regularly gone without food for a whole day, according to the United Na
2024-01-01T05:01:27+0000,How to Start the New Year? Keep the Sea Goddess Happy.,Followers of Afro-Brazilian religions have been displaced by New Year's revelers. But they still find ways to make their offerings to the
2024-01-01T05:09:24+0000,Day 1: A 5-Minute Trick for More Energy,Today's challenge involves a little daytime rest.,"['Content Type: Service', 'Sleep']",Well
2024-01-01T06:00:14+0000,What's on TV This Week: 'RuPaul's Drag Race' and the Golden Globes,The competition show featuring drag queens comes back for a 16th season. The annual award show airs on CBS.,"['Television', 'M
2024-01-01T08:00:02+0000,Today's Wordle Review,In case you need some puzzle help.,['Wordle (Puzzle)'],Crosswords & Games
2024-01-01T08:00:03+0000,Spelling Bee Forum,Feeling stuck on today's puzzle? We can help.,['Spelling Bee (Game)'],Crosswords & Games
2024-01-01T08:00:03+0000,The Connections Companion,In case you need some puzzle help.,"['Crossword Puzzles', 'Connections (Game)']",Crosswords & Games
2024-01-01T08:00:15+0000,It's 2024. Here Are the New Laws That New Yorkers Should Know About.,"Legislation touching on nearly every aspect of life in the state, including wages, health care and education, is going into
2024-01-01T08:00:17+0000,The Well Team Knows You're Tired. And They're Here to Help.,The Well desk's energy challenge is the product of months of research and test runs with friends and family.,"['Medicine and Health',
2024-01-01T08:00:30+0000,New State Laws on Hot-Button Issues Take Effect Today,"Many state laws take effect on the first day of 2024, including new rules on gun safety, a ban on diversity programs and a curb on telemar
2024-01-01T08:00:34+0000,Can New Haven's Legendary Pizza Joints Play on the National Stage?,The city has long been known as a pizza town. Now other places seem to be trying to cash in on the hype.,"['Pizza', 'Family Bu
2024-01-01T10:00:40+0000,Chill in the Housing Market Seeps Into Other Industries,"The slowdown in the residential real estate market, a crucial cog in the American economy, is threatening sectors like home improvement
2024-01-01T10:00:56+0000,Firecrackers and Ice: 5 Must-See Festivals in Asia This Winter,"Catch a mountain trout on a frozen South Korean river, witness a fiery pelting of devotees in Taiwan or find your cold-weather bl
2024-01-01T10:01:01+0000,What Will Life Be Like in 2024?,Here are 12 guesses.,"['Two Thousand Twenty Four', 'Fashion and Apparel', 'Generation Z', 'Social Media', 'Bars and Nightclubs', 'Wearable Computing', 'Nails (Bo
2024-01-01T10:01:01+0000,What Our Critics Are Looking Forward to in 2024,"""Mad Max"" gets a prequel, ""The Wiz"" returns to Broadway and Larry David gets another crack at a series finale.","['Culture (Arts)', 'Two Thousan
2024-01-01T10:01:01+0000,Making a Cooking Resolution? These 9 Recipes Will Get You Started.,"Whether you want to eat less meat, bake more crowd-pleasing desserts or (finally) learn how to cook, here are some first step
2024-01-01T10:01:10+0000,Young Love Meets Russian Repression. They Said 'I Do' in a Moscow Prison,She's 18. He's 23. He sent her a one-sentence letter through his prison's electronic mail system: ""Will you marry me?"" T
2024-01-01T10:01:36+0000,$4.5 Million Homes in California,"A Tudor Revival cottage in Carmel, an equestrian compound in the San Fernando Valley once owned by Patrick Swayze and a waterfront house in Huntington Beach.",
2024-01-01T10:01:38+0000,How Do You Respond to a Young Person Upset by Racist Jokes at School?,"When you talk to students about social media radicalization, racism and bullying, you learn how desperate they are for som
2024-01-01T10:01:58+0000,America Ferrera and the 'Barbie' Monologue We All Talked About,"The key moment in the smash hit went through many revisions: ""We ended in tears. It ended in laughter, it got big, it got small.""
2024-01-01T10:01:58+0000,Emily Blunt Doesn't Care if Her 'Oppenheimer' Character Is Likable,"As the brilliant but flawed Kitty Oppenheimer, the actress plays a woman who had ""extraordinary qualities, as well as ones th
2024-01-01T10:02:02+0000,"Want to Feel, Intellectually, Like Someone Is Rotating Your Tires?",This bracing anthology of Christopher Hitchens's work for The London Review of Books is just the ticket.,"['Hitchens, Christ
2024-01-01T10:02:07+0000,Looking Ahead to 5 Things That Will Shape the 2024 Election,"Trials, a Kennedy and the economy are among the variables to consider.","['Presidential Election of 2024', 'Polls and Public Opinion
2024-01-01T10:02:09+0000,"Trump Team, Burned in 2016, Looks to Close Out Iowa","The former president is leading by impressive margins in the state, but his campaign wants to make sure his supporters turn out.","['Presi
2024-01-01T10:02:12+0000,Trump's Most Ambitious Argument in His Bid for 'Absolute Immunity',"The former president says his acquittal by the Senate in his second impeachment trial, for inciting insurrection, bars any pr
2024-01-01T10:02:37+0000,"Police Officers Are Charged With Crimes, but Are Juries Convicting?","Since the death of George Floyd, a national movement promised sweeping justice reform. So far, police prosecutions have re
2024-01-01T10:45:09+0000,These Classic Characters Are Losing Copyright Protection. They May Never Be the Same.,What could happen to the original version of Mickey Mouse and others after they entered the public domain c
2024-01-01T10:57:10+0000,A New Year's Energy Boost,"This year, we're going to feel less tired.","['New Year', 'Sleep']",Briefing
2024-01-01T11:37:23+0000,"In Quake-Scarred Japan, 2011 Fukushima Disaster Still Looms Large","As another major quake struck Japan, the country was still reckoning with the nuclear crisis triggered by the 8.9-magnitude
2024-01-01T11:44:19+0000,Actress Carrie Bernans Among People Injured in Midtown New Year's Crash,"The motorist struck cars and a food truck in Midtown early Monday morning. At least nine people were injured, including
2024-01-01T12:43:30+0000,China Auto Giant BYD Sells More Electric Vehicles Than Ever,"Sales by BYD, the country's dominant automaker, topped 3 million last year, including 1.6 million fully battery-powered cars. It's a
2024-01-01T13:44:22+0000,Israel Says It Will Pull Several Brigades From Gaza Strip,Analysts say the announcement could signal a shift to the more limited and targeted war against Hamas that the United States has been u
2024-01-01T14:00:03+0000,Modern Swifties Have Transcended the Joke,Examining the linguistic persistence of Swiftian wordplay.,"['Fans (Persons)', 'Social Media', 'Swift, Taylor', 'English Language', 'Pop and Rock Music
2024-01-01T14:00:03+0000,Caregivers Helped Us Be a Family. Everyone Should Have That Option.,This year is a critical moment to make home care accessible to all Americans who need it.,"['Amyotrophic Lateral Sclerosis',
```



Step 2: Pipeline

First, I obtained the CSV data through pandas read function.

```python
@task
def get_data():
    if os.path.exists(CSV_FILE):
        return pd.read_csv(CSV_FILE)
    else:
        raise FileNotFoundError(f"Could not find {CSV_FILE}. Please run ingestion.py first")
```

☆Error checked the data for empty headlines & abstracts, replacing them with empty strings.
☆Combine the headlines & abstracts to help train the model.
☆Randomized 20,000 articles to train the model, requiring at least 150 articles per topic.
☆Sort all the articles into the bucket topics in chunks of 5000.
☆Transform: Attach each headline to their topic.
☆ Convert the topics to a dictionary
☆ Match each topic ID to the dictionary

```python
...sform_and_model(df):
    # error check for empty headlines / abstracts, fill with empty string
    df['headline'] = df['headline'].fillna('')
    df['abstract'] = df['abstract'].fillna('')

    # combine headline and abstract for more context to train model
    docs = (df['headline'] + ". " + df['abstract']).tolist()

    # pick 20,000 random articles to train model
    from sklearn.utils import shuffle
    training_docs = shuffle(docs, random_state=42)[:20000]

    # a topic must have at least 150 articles
    topic_model = BERTopic(min_topic_size=300, verbose=True)
    topic_model.fit(training_docs)

    #transform in chunks of 5000
    chunk_size = 5000
    all_topics = []

    for i in range(0, len(docs), chunk_size):
        chunk = docs[i : i + chunk_size]
        # transform chunk
        chunk_topics, _ = topic_model.transform(chunk)
        all_topics.extend(chunk_topics)

    # attach headline to topic
    df['topic_id'] = all_topics

    # get the table of topics / topic ID and convert it to a dictionary
    topic_info = topic_model.get_topic_info()
    topic_map = topic_info.set_index('Topic')['Name'].to_dict()

    # match each topic id to the dictionary
    df['topic_name'] = df['topic_id'].map(topic_map)

    return df
```

Step 2: Pipeline continued

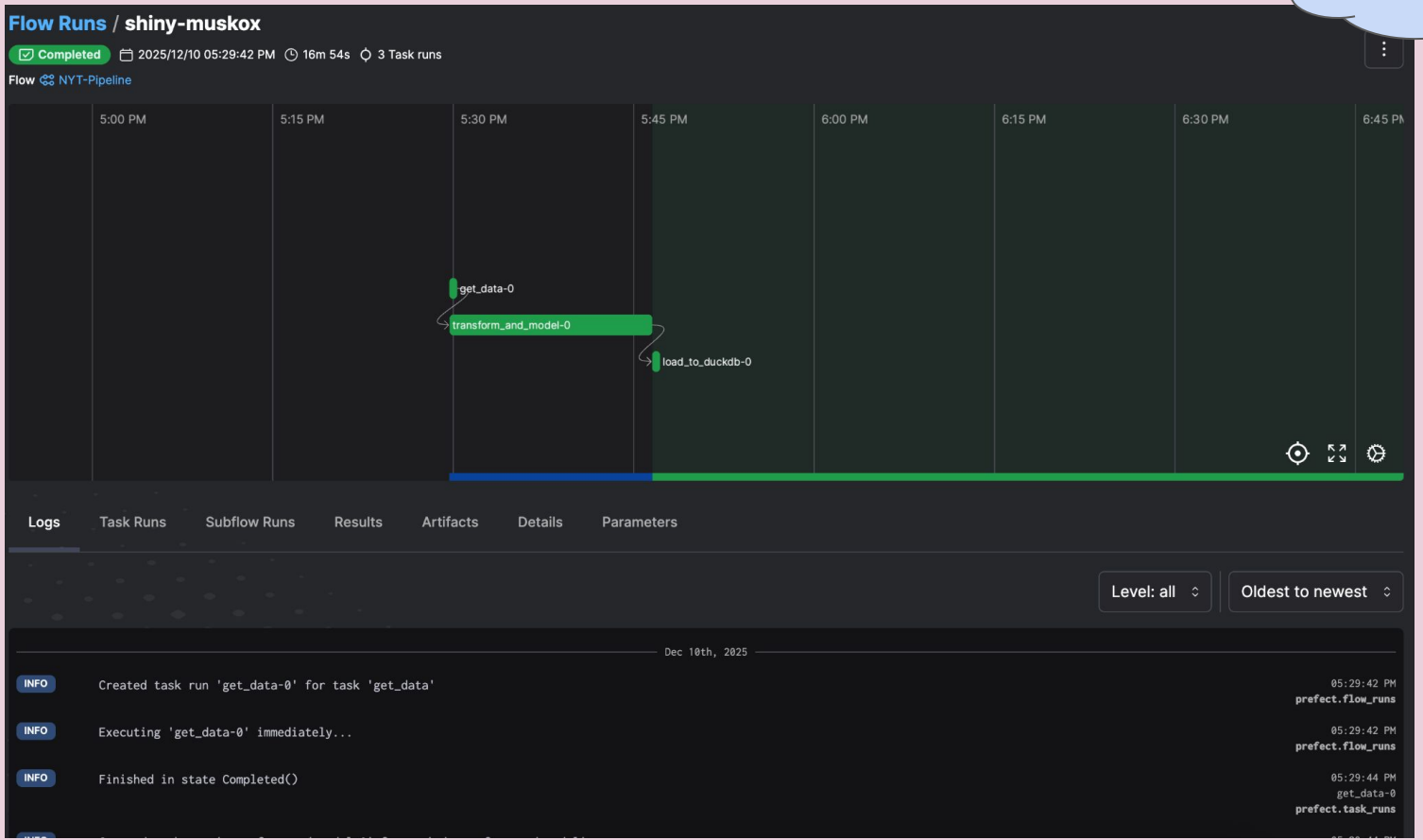
Connect to Duck DB and create a SQL table. This step transforms the 400,000+ processed records into a query-optimized SQL table

```python
@task
def load_to_duckdb(df):
    conn = duckdb.connect(DB_FILE)
    conn.execute("CREATE OR REPLACE TABLE processed_articles AS SELECT * FROM df")
    conn.close()
    print("Loaded to DuckDB")
```

This function defines the pipeline's DAG. It is responsible for dependency management, ensuring that tasks execute in the strict order of Extraction →Transformation → Loading

```python
@flow(name="NYT-Pipeline")
def main_flow():
    raw_df = get_data()

    final_df = transform_and_model(raw_df)

    load_to_duckdb(final_df)

if __name__ == "__main__":
    main_flow()
```

A successful Prefect flow run!



```
D SELECT topic_name, count(*) FROM processed_articles WHERE topic_id != -1 GROUP BY 1 ORD
ER BY 2 DESC LIMIT 10;
```

topic_name varchar	count_star() int64
0_trump_president_donald_the	34054
3_restaurant_food_chef_and	13904
2_novel_books_book_and	12293
1_police_shooting_man_was	12110
5_coronavirus_vaccine_covid_virus	11441
6_china_korea_north_chinese	10383
4_art_museum_artist_of	10252
7_met_couple_bride_groom	8976
8_fashion_designer_week_paris	7744
9_ukraine_russia_russian_putin	7623
10 rows	2 columns

The results of the mapping produced in DuckDB!

```
D SELECT * FROM processed_articles WHERE topic_name = '0_trump_president_donald_the' LIMI
T 5;
```

date varchar	headline varchar	…	topic_id int64	topic_name varchar
2015-01-01T01:36:1…	Much of David Duke…	…	0	0_trump_president_…
2015-01-01T22:15:4…	Teach Congress a L…	…	0	0_trump_president_…
2015-01-01T22:43:2…	Jeb Bush Resigns F…	…	0	0_trump_president_…
2015-01-02T12:35:1…	Jeb Bush, and Two …	…	0	0_trump_president_…
2015-01-02T16:38:1…	‘To Make Men Free,…	…	0	0_trump_president_…
5 rows				9 columns (4 shown)

```
D UPDATE processed_articles SET topic_name = 'Climate/Energy' WHERE topic_id = 19;
D UPDATE processed_articles SET topic_name = 'General Housing' WHERE topic_id = 20;
D UPDATE processed_articles SET topic_name = 'Israel/Palestine' WHERE topic_id = 21;
D UPDATE processed_articles SET topic_name = 'Macroeconomics' WHERE topic_id = 22;
D UPDATE processed_articles SET topic_name = 'Pop Music/Albums' WHERE topic_id = 23;
D UPDATE processed_articles SET topic_name = 'Puzzles/Games' WHERE topic_id = 24;
D UPDATE processed_articles SET topic_name = 'Dance/Ballet' WHERE topic_id = 25;
D UPDATE processed_articles SET topic_name = 'Classical Music' WHERE topic_id = 26;
D UPDATE processed_articles SET topic_name = 'Olympics' WHERE topic_id = 27;
D UPDATE processed_articles SET topic_name = 'Legislation/Tax' WHERE topic_id = 28;
D UPDATE processed_articles SET topic_name = 'NBA/Basketball' WHERE topic_id = 29;
D UPDATE processed_articles SET topic_name = 'Social Issues' WHERE topic_id = 30;
D UPDATE processed_articles SET topic_name = 'NFL/Football' WHERE topic_id = 31;
D UPDATE processed_articles SET topic_name = 'Brexit/UK Politics' WHERE topic_id = 32;
D UPDATE processed_articles SET topic_name = 'Autos/EVs' WHERE topic_id = 33;
D UPDATE processed_articles SET topic_name = 'Tennis' WHERE topic_id = 34;
D UPDATE processed_articles SET topic_name = 'NYT Internal Notes' WHERE topic_id = 35;
D UPDATE processed_articles SET topic_name = 'Soccer/World Cup' WHERE topic_id = 36;
D UPDATE processed_articles SET topic_name = 'NYC Local Politics' WHERE topic_id = 37;
D UPDATE processed_articles SET topic_name = 'Corrections' WHERE topic_id = 38;
D UPDATE processed_articles SET topic_name = 'Health/Disease' WHERE topic_id = 39;
D UPDATE processed_articles SET topic_name = 'Awards Shows' WHERE topic_id = 40;
D UPDATE processed_articles SET topic_name = 'Spelling Bee' WHERE topic_id = 41;
D UPDATE processed_articles SET topic_name = 'Social Media' WHERE topic_id = 42;
D UPDATE processed_articles SET topic_name = 'Metropolitan Diary' WHERE topic_id = 43;
D UPDATE processed_articles SET topic_name = 'Vatican/Pope' WHERE topic_id = 44;
D UPDATE processed_articles SET topic_name = 'College Football' WHERE topic_id = 45;
D UPDATE processed_articles SET topic_name = 'NYC Transit' WHERE topic_id = 46;
D UPDATE processed_articles SET topic_name = 'Real Estate Listings' WHERE topic_id = 47;
D UPDATE processed_articles SET topic_name = 'Golf' WHERE topic_id = 48;
D UPDATE processed_articles SET topic_name = 'France/Macron' WHERE topic_id = 49;
D UPDATE processed_articles SET topic_name = 'Parenting' WHERE topic_id = 50;
D UPDATE processed_articles SET topic_name = 'US-Mexico Border' WHERE topic_id = 51;
D UPDATE processed_articles SET topic_name = 'European Migration' WHERE topic_id = 52;
```

Using DuckDB to rename the topics to more concise wording.

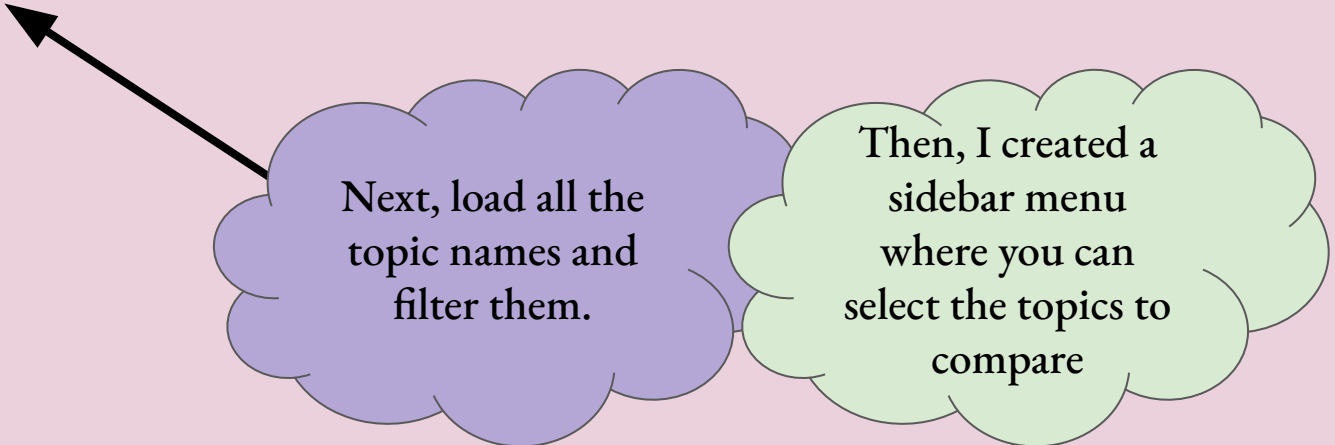
Step 3: Building the App using StreamLit

Streamlit is an open-source Python framework that allows data scientists and machine learning engineers to create interactive web applications and dashboards using only Python

```python
@st.cache_data # data only loaded once
def load_data():

    try:
        conn = duckdb.connect(DB_FILE, read_only=True)
        # count the number of articles per topic, grouped by month
        query = f"""
        SELECT
            topic_name,
            date_trunc('month', CAST(date AS TIMESTAMP)) as month,
            count(*) as count
        FROM processed_articles
        WHERE topic_id != -1
        GROUP BY 1, 2
        ORDER BY 2
        """
        df = conn.execute(query).df()
        conn.close()
        return df
    except Exception as e:
        st.error(f"Error loading DuckDB data. Have you run pipeline.py yet? Error: {e}")
        return pd.DataFrame()
```

First step: load all of the dad from the DuckDB file

```python
if not df.empty:
    st.sidebar.header("Filter Trends")

    df['topic_name'] = df['topic_name'].fillna('Unclassified').astype(str)
    df['display_topic'] = df['topic_name']
    all_display_topics = sorted(df['display_topic'].unique())

    initial_filter = [t for t in all_display_topics if t not in ['Noise Random Stuff', 'Abstract', 'Main']] # filter

    # allow user to select multiple topics
    selected_topics = st.sidebar.multiselect(
        "Select Topics to Compare",
        all_display_topics,
        default=initial_filter[:0]
```



```python
if selected_topics:
    filtered_df = df[df['display_topic'].isin(selected_topics)] # filter based on user se

    # Create interactive line chart with Plotly
    fig = px.line(
        filtered_df,
        x='month',
        y='count',
        color='display_topic',
        title="Topic Frequency Over Time (2015-2024)",
        labels={'count': 'Number of Articles', 'month': 'Date', 'display_topic': 'Topic'},
        template="plotly_white",
        height=600
    )

    # Customize chart appearance
    fig.update_xaxes(dtick="M12", tickformat="%Y") # Show labels annually
    fig.update_traces(mode="lines", hovertemplate="%{y} articles<br>%{x|%b %Y}")
    fig.update_layout(hovermode="x unified")

    st.plotly_chart(fig, use_container_width=True)
```

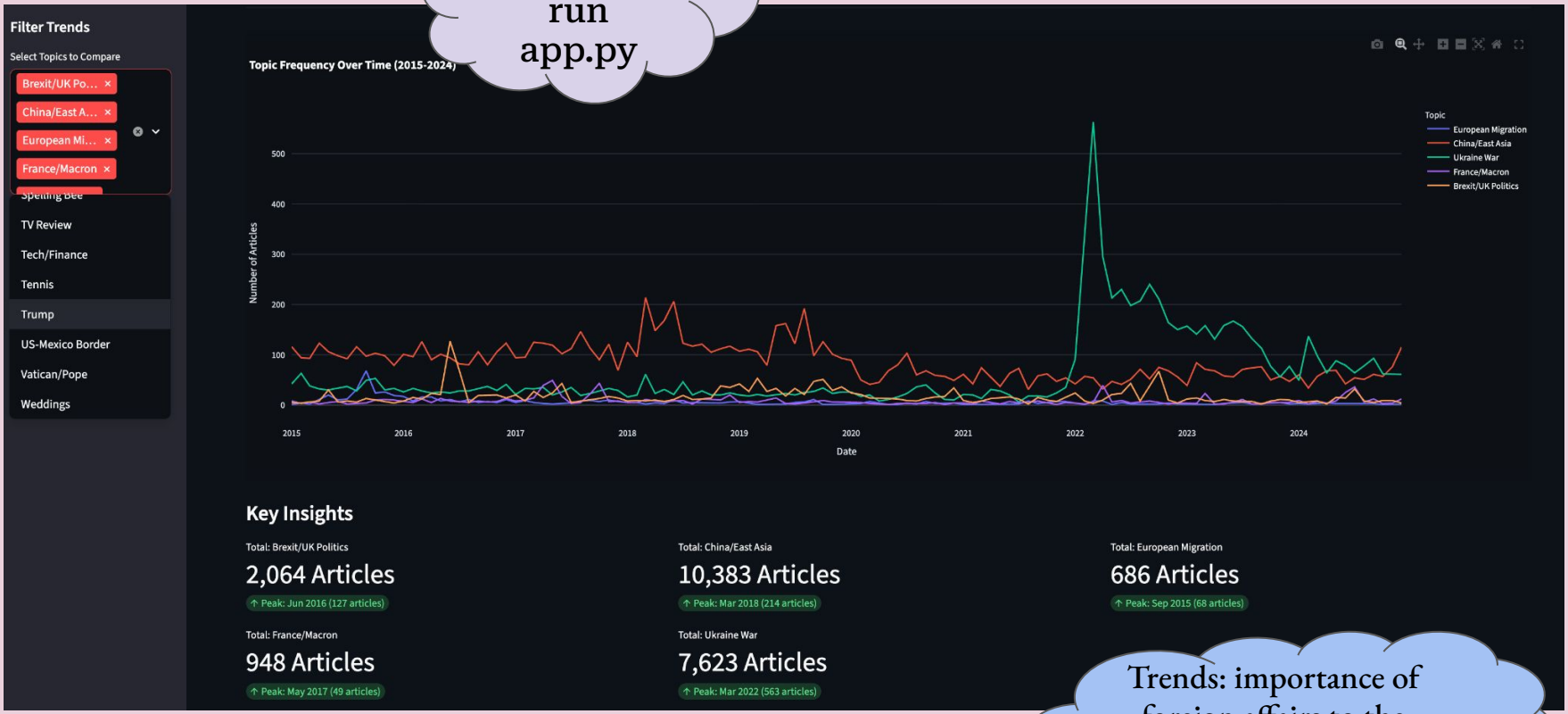
For the chart, for each selected topic, Plotly draws a line with the topic frequency by counting the number of articles under that topic for each month

```python
# INSIGHTS
st.subheader("Key Insights")
cols = st.columns(3)

for idx, topic in enumerate(selected_topics):
    topic_data = filtered_df[filtered_df['display_topic'] == topic]

    # Find the peak month and total count for the topic
    total_articles = topic_data['count'].sum()
    peak_row = topic_data.sort_values('count', ascending=False).iloc[0]
    peak_month_str = peak_row['month'].strftime('%b %Y')
    peak_count = peak_row['count']

    with cols[idx % 3]:
        st.metric(
            label=f"Total: {topic}",
            value=f"{total_articles:,} Articles",
            delta=f"Peak: {peak_month_str} ({peak_count:,} articles)"
        )
```

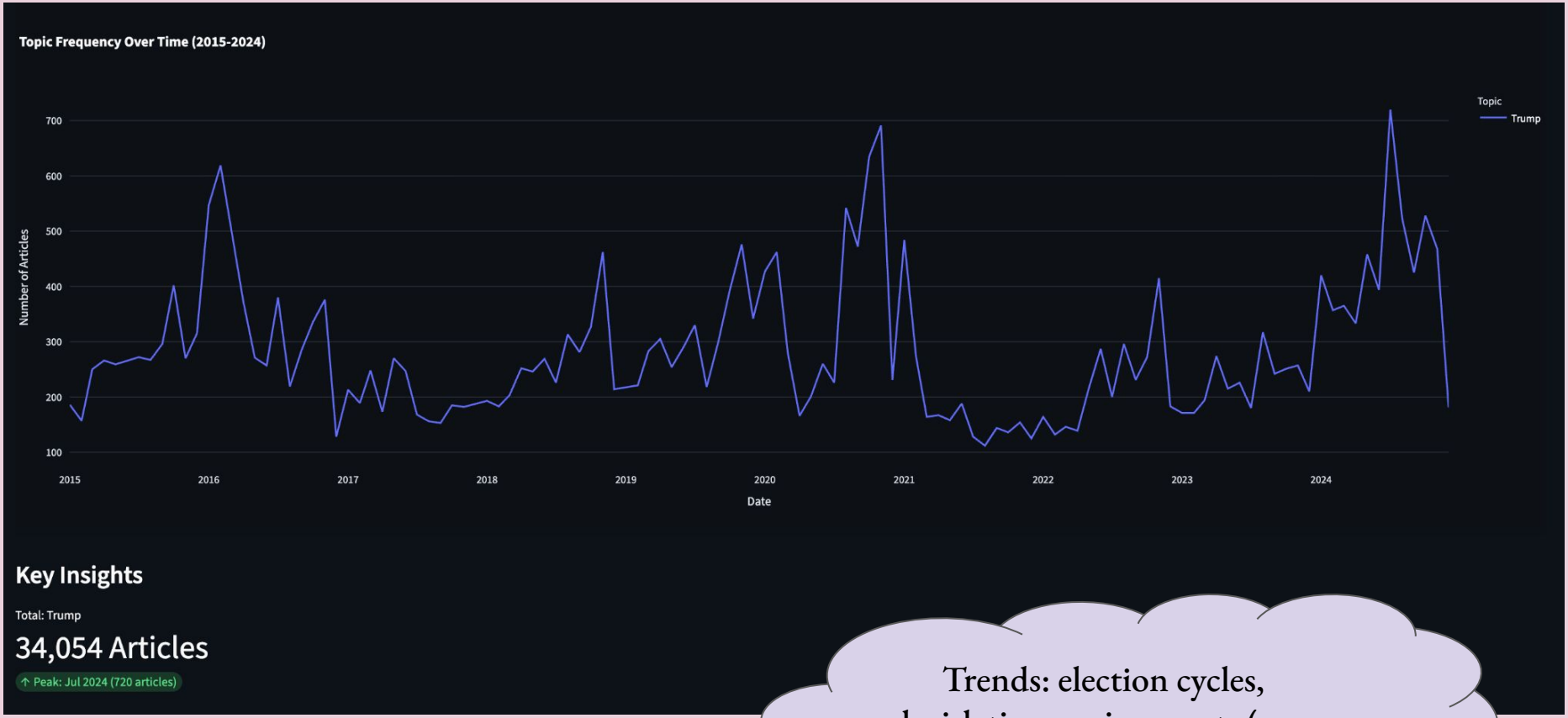
Then, the bottom shows the key insights for the selected topics. It finds the peak month for the topic compared to its total count

Running the App:

Command: streamlit run app.py



Trends: importance of foreign affairs to the NYT and American citizens



Trends: election cycles, legislation, major events (e.g. attempted assasination)