DATA7201 Project Report

Longpeng Xu
MDataSci Candidate
The University of Queensland
May 2023

Abstract

Big data analytics is a broad and complex field that requires careful selection of appropriate data infrastructure to face the challenges effectively, especially data quality issues. The project utilized PySpark to analyze a semi-structured Facebook advertisement dataset stored on HDFS, concerning the US presidential election in 2020. The findings of the project include the style difference between Donald J. Trump and Joe Biden and the difference in their popularity among Facebook users.

Table of Contents

1. Introduction

Big data analytics has been widely applied in private and public sectors, with the general aim of transforming big data into information and insights, involving the appropriate selection of big data infrastructure. However, the emergence of big data is accompanied by challenges dictated by their characteristics: Volume, velocity, variety, veracity, and the most important to the stakeholders, value. With the omnipresence of data collection, the volume of the data is extremely large, and enormous data flows keep being generated in the world every second. Big data can be either structured, semi-structured, or unstructured, and may be fitted into either relational or non-relational models. Furthermore, it seems that big data make data storage become garbage collectors, where people can hardly ensure the data's trustworthiness and extract the value for effective decision-making.

Hence, it is cardinal to introduce distributed system solutions fitting the stakeholders' specific case (including budgets, timeline, expected output, and features of data, etc.) and preserve the value of data. An example is that to prevent securities fraud which requires the real-time monitoring of transactions, Storm is capable of its high processing rates and low latency. For another instance, Spark is a well-regarded candidate if there is a supermarket tycoon who has a large volume, distributed sales data on which the monthly analytics is based, since Spark RDD allows efficient in-memory computation and Spark is capable of machine learning for future sales prediction.

2. Dataset Analytics

2.1 Dataset

The dataset packages the political advertisements with the target audience of Facebook users in the US. It is retrieved from Facebook's Ad Library API from March 2020 to January 2022, and distributed across multiple JSON files which have a schema and the only data type of strings. Hence, the dataset is semi-structured. Moreover, valuable information about the 2020 presidential election can be distilled, for the perfect coverage of the pre-election period by the dataset. The original dataset has 99,630,847 rows in total before preprocessing, and 18 columns of strings or structures of strings (i.e., nested columns of strings that are not in structured data) accommodating date and time, numbers, texts, URLs, etc. The dataset is on HDFS.

Several columns are frequently used later. 'page_name' records the initiator of an advertisement. 'ad_delivery_start_time' indicates the earliest time when an advertisement was posted. 'spend' is structured, containing the lower and the upper limit for budgeting an advertisement. 'demographic_distribution' is also structured by combining age brackets, sex and percentage providing information about the distribution of users, in different age-sex brackets, who were exposed to an advertisement. 'ad_creative_body' captures the caption of an advertisement.

2.2 Data Pre-processing

The Facebook dataset is perfect for exploring interesting findings about the election. Hence, the author pinpoints the main focus on the comparison between the candidates, Donald J. Trump versus

Joe Biden, analyzing their differences in posting advertisements (Task 1), in the demographic distribution of the targeted users (Task 2), in colloquial expressions regarding the election (Task 3), etc.

The preprocessing steps are arranged accordingly to the main focus. First of all, set the PySpark environment and import all the JSON files into a single Spark data frame, dropping the duplicate rows which consist 34% of the original dataset. Then, the preprocessing for Trump and Biden are separated but with the same steps. Filter the advertisement by their respective names in the 'page_name' field. For different tasks, there are different preprocessing steps, the resulting data frames of which are named differently, in order to preserve an in-memory copy of the original dataset during the analytics. The Task 1 dataset converts the 'ad_delivery_start_time' in the original dataset to 'date_type' (i.e., dropping the time). The Task 2 dataset again extracts the year and the month of each row and concatenates them with a hyphen. The preprocessing for Task 3 is more complicated than the former two: Select the columns of interest from the original dataset, split the words in advertisement captions which are transformed to lowercase, and split a row of multiple words into the multiple rows.

2.3 Analytics Steps

As mentioned, PySpark is the platform for the analytics. It is the most suitable big data infrastructure for the author's use case. The high volume of the Facebook dataset raises the difficulty to maintain stable accesses, data veracity, and avoidance of single-node failures where distributed processing is unavoidable. These issues are solvable with Spark's fault tolerance which is ensured by resilient distributed datasets (RDD). The high volume leverages the demand for the filter operation which is supported by Spark. Moreover, the dataset is semi-structured and is stored in HDFS, which is compatible with Spark supporting the full spectrum of data types and various data sources including HDFS. Undoubtedly, Spark is eligible in analyzing the patterns and provides insights for stakeholders in a relatively effortless way since PySpark allows user interaction in the style of Python and SQL, which is user-friendly especially for non-technical executives.

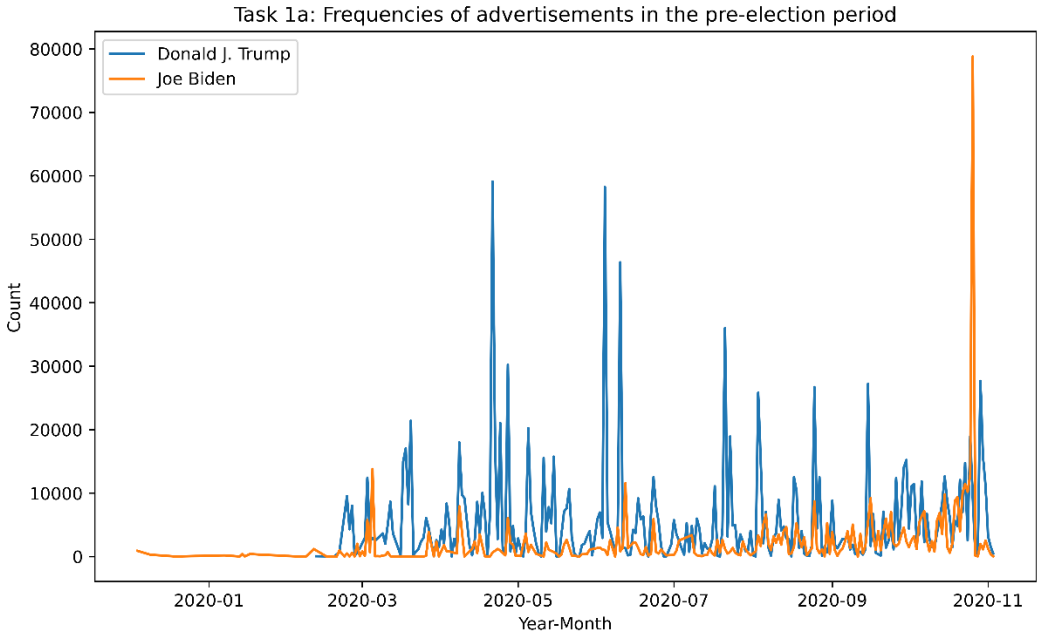
For Task 1, the author explores how the frequencies and the budgets of advertising differ between the candidates. For frequencies, group the preprocessed data by different days, then aggregate the number of rows of each day so that the result counts the number of advertisements each day indicating the activeness of a candidate. To obtain the average (i.e., (upper bound + lower bound) / 2) budget of the advertisements, lower bounds, and upper bounds are extracted and cast to integer, from the structured column 'spend'. Then, group the data frame by 'id' and aggregate by the maximum of the average spending of the same advertisement id. Finally, sort the data frame by the descending order of the average spending.

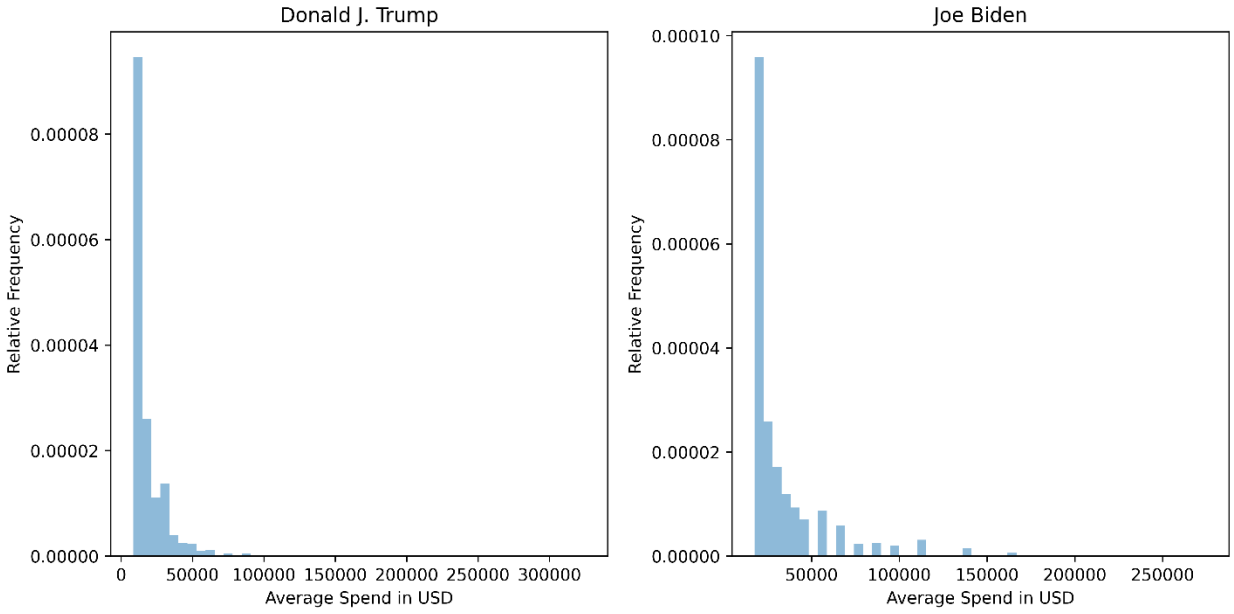
For Task 2, the comparison regarding the age distribution of the targeted individuals is investigated. Given the age distribution fluctuates over time, it is computed for every month. Firstly, explode 'demographic_distribution' in the preprocessed data frame to unfold the nested columns. Then, select the 'year_month', 'age', and 'percentage' cast to floats. Filter out outliers (those aged '13-17'), group by the first two attributes and aggregate by the average of the percentages. Finally, sort the data frame by the two attributes.

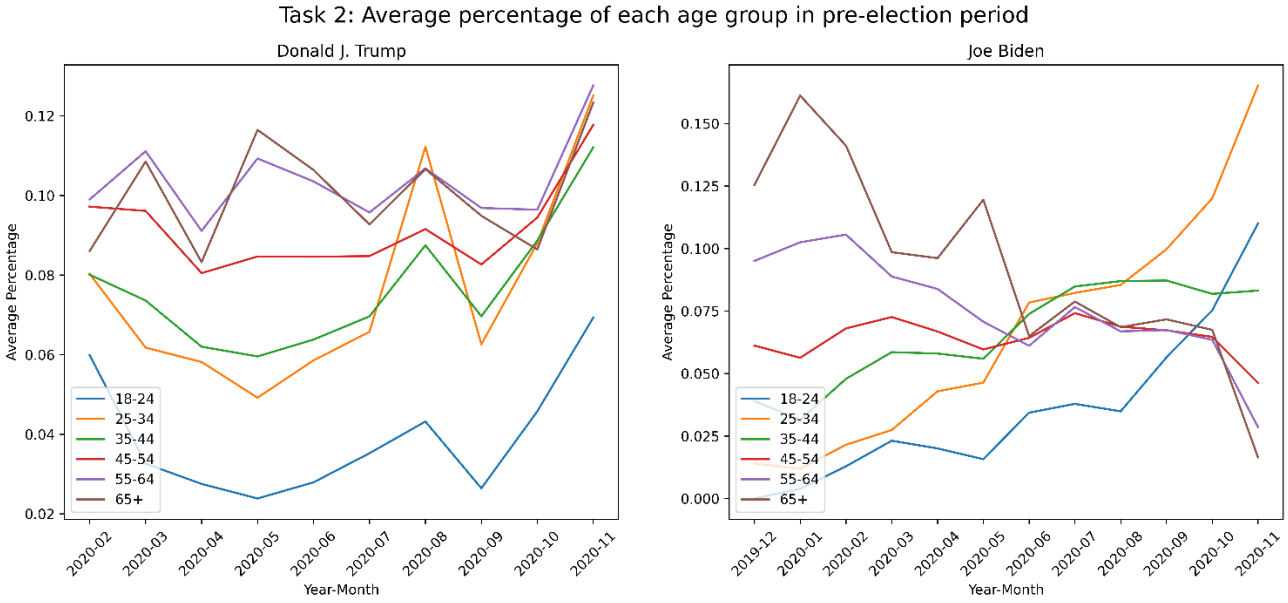
For Task 3, the author analyzed the most frequent words that appeared in the advertisement captions for the Trump-Biden comparison. Given the preprocessed data frame with a column consisting of the split words of a caption, explode the column so that one row contains only one split word. Group the rows by the column of the split word and aggregate by the count of each word. Finally, order the data frame by the descending order of the counts.

2.4 Results

The results are presented in the following four figures.






Task 2: Average percentage of each age group in pre-election period
Donald J. Trump
Joe Biden
Average Percentage
Year-Month
18-24
25-34
35-44
45-54
55-64
65+
0.12
0.10
0.08
0.06
0.04
0.02
0.150
0.125
0.100
0.075
0.050
0.025
0.000
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11

Task 3 The 50 most frequent words

Rank	Donald J. Trump		Joe Biden	
1	the	2847071	to	1184710
2	to	2344902	the	817767
3	and	1580116	and	691909
4	trump	1386357	we	486627
5	your	1073207	in	440896
6	president	1034819	you	431158
7	for	948743	a	385360
8	is	902544	of	363067
9	a	885715	our	315509
10	in	857331	is	282709
11	of	667295	your	281707
12	our	660656	trump	280583
13	you	594649	this	252193
14	we	566334	for	219087
15	official	520386	donald	216754
16	has	418096	joe	209035
17	on	406001	on	205462
18	this	349113	that	203706
19	will	346203	us	199665
20	joe	341808	can	194279
21	support	340353	need	192268
22	that	307455	but	169980
23	have	298714	will	163376
24	about	297189	biden	153704
25	with	296111	now	136026
26	now	268649	help	134581
27	his	260446	have	134208
28	biden	255941	i	133614
29	made	247515	make	128197
30	2020	239570	more	126083
31	are	237484	if	123142
32	by	235460	are	118302
33	statement	235011	know	113669
34	proudly	233485	vote	112838
35	be	232153	so	108827
36	america	231250	we’re	108409
37	everything	229114	defeat	103609
38	usa!	217258	up	102775
39	take	212571	take	100670
40	want	211979	as	97574
41	high	206131	with	95604
42	sellout	204040	before	95310
43	always,	203789	from	93994
44	2020.\n\nas	203789	just	93010
45	american	202881	–	92246
46	risk\n\nmake	199911	be	88328
47	make	197469	it	87653
48	show	192050	right	84935
49	he	191699	going	82232
50	their	180883	democrats	81654

3. Discussion and Conclusions

From Task 1a figure, Trump tended to invest in advertisements in short-term valiant attempts. Trump's blue line has much more frequent peaks than Biden's orange line. Thus, Trump exploited (and possibly, abused) the platform much more frequently than Biden. Nevertheless, Biden is more prepared for the election and started sparse advertisement campaigns in November 2019, three months earlier than the opposite. Biden kept a stable count of advertisements from March to October 2020, while in the last month before the election, the frequency of advertisements launched by Biden surged to at most 80,000 a day, which can be regarded as a powerful kickoff for his win.

However, Task 1b figure illustrates that Trump invested less than Biden in terms of the total amount, evidenced by the percentage (i.e., relative frequencies) difference of investing 40000 ~ 250000 USD for a single advertisement id. This is even more interesting, considering that the percentages for investing in cheaper advertisements are approximately the same between Trump and Biden. Combining with the findings from Task 1a figure, it can be inferred that investing more in a single advertisement is better than investing a lot in overflowing advertisements, which may cause more dislike among Facebook users.

From Task 2 figure, compared to Trump, Biden won higher popularity among Facebook users aged from 18 to 34, from September 2020 to the election. In contrast, the percentage of 35-to-66 age brackets who were exposed to Trump's advertisements is higher than the opposite most of the time. This means that Biden is more supported by the young while Trump is more recognized by the elderly. Only Trump encountered a case where the exposure rates among all age groups slumped (in September 2020), which may also reflect the style difference between the two.

From Task 3 table, while Trump focused more on himself, Biden is more aware of the opposite. There are 2847071 'the', 1386357 'trump' and 1034819 'president' in Trump's advertisements, while in Biden's, there are 817767 'the', 153704 'biden' and 'president' is even not a top 50 keyword from Biden. 'trump' ranks 12th in Biden's keywords, while 'joe' ranks only 20th in Trump's keywords.

Essentially, the key takeaway messages for the stakeholders are

- Trump invested less in the total amount but posted much more advertisements within a shorter period.
- Biden was more prepared, investing more in the total amount distributed across fewer advertisements. The number of advertisements was stable except one month before the election.
- Trump was preferred by those aged over 35, while Biden won more popularity among those under 35.
- Trump focused more on himself in his advertisement campaigns, while Biden was more cautious about the opposite over himself.

Appendix

```
# Overall preprocessing

from pyspark.sql import SparkSession
import pyspark
spark = SparkSession \
    .builder \
    .appName('project') \
    .getOrCreate()
# This returns a SparkSession object if already exists, and creates a new one if not exist.
spark

fb = spark.read.option("header",True).json('/data/ProjectDatasetFacebook/*').dropDuplicates()



# Trump preprocessing

trump = fb.filter(fb['page_name'] == 'Donald J. Trump')
trump.show()

from pyspark.sql.functions import to_date
trump = trump.withColumn("date_type", to_date("ad_delivery_start_time"))

trump.printSchema()

# Trump task 1

from pyspark.sql.functions import count, desc
trump_counts = trump.groupBy('date_type').agg(count('*').alias('count'))
trump_counts = trump_counts.orderBy('date_type')
trump_counts.show(1000)

from pyspark.sql.functions import col, max, expr, desc
trump = trump.withColumn("average_spend", (expr("CAST(spend.lower_bound AS INT)") +
expr("CAST(spend.upper_bound AS INT)")) / 2)
trump_max_spend = trump.groupBy('id').agg(max('average_spend').alias('max_average_spend'))
trump_max_spend = trump_max_spend.orderBy(desc('max_average_spend')).limit(1000)
trump_max_spend.show(1000)

# Trump task 2

from pyspark.sql.functions import explode, avg, expr, concat_ws, year, month

trump_age = trump.withColumn('year', year('date_type')).withColumn('month', month('date_type'))
trump_age = trump_age.withColumn('year_month', concat_ws('-', 'year', 'month'))
trump_age = trump_age.select('year_month',
explode('demographic_distribution').alias('demographics'))
trump_age = trump_age.select('year_month', 'demographics.age', expr("CAST(demographics.percentage
AS FLOAT)").alias('percentage'))
trump_age = trump_age.filter(trump_age.age != '13-17')
trump_avg_pc = trump_age.groupBy('year_month',
'age').agg(avg('percentage').alias('average_percentage'))
trump_avg_pc = trump_avg_pc.orderBy('year_month','age').limit(1000)
trump_avg_pc.show(1000)

# Trump task 3

trump_words = trump.select('date_type', 'ad_creative_body')

from pyspark.sql.functions import split, lower
trump_words = trump_words.withColumn("ad_words", split(lower(trump_words["ad_creative_body"]), "
"))
```

```
trump_words = trump_words.selectExpr("*", "explode(ad_words) as ad_words_exploded")

from pyspark.sql.functions import count, desc
trump_words = trump_words.groupBy('ad_words_exploded').agg(count('*').alias('count_'))

trump_words = trump_words.orderBy(desc('count_')).limit(50)

trump_words.show(50)



# Biden preprocessing

biden = fb.filter(fb['page_name'] == 'Joe Biden')
biden.show()

from pyspark.sql.functions import to_date
biden = biden.withColumn("date_type", to_date("ad_delivery_start_time"))

biden.printSchema()

# Biden task 1

from pyspark.sql.functions import count, desc
biden_counts = biden.groupBy('date_type').agg(count('*').alias('count'))
biden_counts = biden_counts.orderBy('date_type')
biden_counts.show(1000)

from pyspark.sql.functions import col, max, expr, desc
biden = biden.withColumn("average_spend", (expr("CAST(spend.lower_bound AS INT)") +
expr("CAST(spend.upper_bound AS INT)")) / 2)
biden_max_spend = biden.groupBy('id').agg(max('average_spend').alias('max_average_spend'))
biden_max_spend = biden_max_spend.orderBy(desc('max_average_spend')).limit(1000)
biden_max_spend.show(1000)

# Biden task 2

from pyspark.sql.functions import explode, avg, expr, concat_ws, year, month

biden_age = biden.withColumn('year', year('date_type')).withColumn('month', month('date_type'))
biden_age = biden_age.withColumn('year_month', concat_ws('-', 'year', 'month'))
biden_age = biden_age.select('year_month',
explode('demographic_distribution').alias('demographics'))
biden_age = biden_age.select('year_month', 'demographics.age', expr("CAST(demographics.percentage
AS FLOAT)").alias('percentage'))
biden_age = biden_age.filter(biden_age.age != '13-17')
biden_avg_pc = biden_age.groupBy('year_month',
'age').agg(avg('percentage').alias('average_percentage'))
biden_avg_pc = biden_avg_pc.orderBy('year_month','age').limit(1000)
biden_avg_pc.show(1000)

# Biden task 3

biden_words = biden.select('date_type', 'ad_creative_body')

from pyspark.sql.functions import split, lower
biden_words = biden_words.withColumn("ad_words", split(lower(biden_words["ad_creative_body"]), "
"))

biden_words = biden_words.selectExpr("*", "explode(ad_words) as ad_words_exploded")

from pyspark.sql.functions import count, desc
biden_words = biden_words.groupBy('ad_words_exploded').agg(count('*').alias('count_'))

biden_words = biden_words.orderBy(desc('count_')).limit(50)
```

```
biden_words.show(50)



# Plots
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.ticker import FuncFormatter

# Task 1a Plot

trump_counts_pandas = trump_counts.toPandas()
biden_counts_pandas = biden_counts.toPandas()
df1 = trump_counts_pandas
df2 = biden_counts_pandas

# Convert to datetime type
df1['date_type'] = pd.to_datetime(df1['date_type'])
df2['date_type'] = pd.to_datetime(df2['date_type'])

# Plot two series
plt.figure(figsize=(10,6), dpi=600, facecolor='white')

plt.plot(df1['date_type'], df1['count'], label='Series 1')
plt.plot(df2['date_type'], df2['count'], label='Series 2')

plt.xlabel('Year-Month')
plt.ylabel('Count')
plt.title('Task 1a: Frequencies of advertisements in the pre-election period')
plt.legend(['Donald J. Trump','Joe Biden'])

plt.show()

# Task 1b Plot

trump_max_spend_pandas = trump_max_spend.toPandas()
biden_max_spend_pandas = biden_max_spend.toPandas()
df3 = trump_max_spend_pandas
df4 = biden_max_spend_pandas

# Plot the data
fig = plt.figure(figsize=(12, 6), dpi=600, facecolor='white')

# Formatter function
def to_fixed(number, digits=5):
    return f"{number:.{digits}f}"
formatter = FuncFormatter(lambda y, _: to_fixed(y))

ax1 = fig.add_subplot(1, 2, 1)
ax1.hist(df3['max_average_spend'], density=True, bins=50, alpha=0.5)
ax1.set_title('Donald J. Trump')
ax1.set_xlabel('Average Spend in USD')
ax1.set_ylabel('Relative Frequency')
ax1.yaxis.set_major_formatter(formatter)

ax2 = fig.add_subplot(1, 2, 2)
ax2.hist(df4['max_average_spend'], density=True, bins=50, alpha=0.5)
ax2.set_title('Joe Biden')
ax2.set_xlabel('Average Spend in USD')
ax2.set_ylabel('Relative Frequency')
ax2.yaxis.set_major_formatter(formatter)

plt.suptitle('Task 1b: The distribution of 1000 greatest average spends')
plt.subplots_adjust(wspace=0.25, top=0.85)
plt.show()
```

```
# Task 2 Plot

trump_avg_pc_pandas = trump_avg_pc.toPandas()
biden_avg_pc_pandas = biden_avg_pc.toPandas()
df1 = trump_avg_pc_pandas
df2 = biden_avg_pc_pandas

# Pivot the data to have one column per age group
df1_pivot = df1.pivot(index='year_month', columns='age', values='average_percentage').fillna(0)
df2_pivot = df2.pivot(index='year_month', columns='age', values='average_percentage').fillna(0)

# Create a figure with 1*2 subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6), dpi=600, facecolor='white')

# Plot df1
for column in df1_pivot.columns:
    axes[0].plot(df1_pivot.index, df1_pivot[column], label=column)
axes[0].set_xlabel('Year-Month')
axes[0].set_ylabel('Average Percentage')
axes[0].set_title('Donald J. Trump')
axes[0].legend(loc="lower left")

# Plot df2
for column in df2_pivot.columns:
    axes[1].plot(df2_pivot.index, df2_pivot[column], label=column)
axes[1].set_xlabel('Year-Month')
axes[1].set_ylabel('Average Percentage')
axes[1].set_title('Joe Biden')
axes[1].legend(loc="lower left")

# Rotate x-axis labels by 45 degrees for both subplots
axes[0].tick_params(axis='x', rotation=45)
axes[1].tick_params(axis='x', rotation=45)

# Supreme title
plt.suptitle('Task 2: Average percentage of each age group in pre-election period')
plt.show()
```