
THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

# DATA7201 Project Report

by

**Zichuan Huang**

School of Information Technology and Electrical Engineering,
University of Queensland.

Submitted for the degree of Master of Data Science
in the division of 2020 – 2022

May 26, 2022

# Abstract

This report is for the DATA7201 project, and will be divided into the following three parts. In Chapter 1, I will introduce the general area of big data analytics and the reason that why we need distributed system with an example from Bloomberg. Chapter 2 describe the project itself including the dataset, the tools that I used and the data analytics results. Last in Chapter 3, I will make a conclusion about the data analytic results and do some discussions.

# Table of contents

**Chapter 1 Introduction** ..... 1

**1.1 Definition** ..... 1

**1.2 The need for distributed system** ..... 1

**Chapter 2 Dataset analytics** ..... 3

**2.1 Dataset** ..... 3

**2.2 Pre-processing of the dataset** ..... 3

**2.3 Data analytics and results** ..... 3

**Chapter 3 Discussions and conclusions** ..... 6

**Appendix** ..... 7

# Chapter 1

# Introduction

### 1.1 Definition

With the continuous development of various technologies, the amount of data and information that engineers need to receive is also increasing, which accelerates the birth of big data analysis technology. Big data analysis refers to the use of advanced analysis technology for huge and diverse data sets. These datasets include structured, semi-structured, and unstructured data from various sources, ranging in size from terabytes to zettabytes.

The term big data applies to datasets whose size or type exceeds the ability of traditional relational databases to capture, manage, and process data with low latency. Big data has one or more of the following characteristics: large volume, speed, or variety. Artificial intelligence (AI), mobile, social and Internet of Things (IoT) technologies are increasing data complexity through new data forms and sources. For example, big data comes from sensors, devices, video/audio, networks, log files, transactional applications, the web, and social media—most of which are generated at scale in real-time.

Big data analytics enables analysts, researchers and business users to make smarter and faster decisions using data that was previously inaccessible or unavailable. Businesses can use advanced analytics techniques such as text analytics, machine learning, predictive analytics, data mining, statistics, and natural language processing to gain new insights from previously untapped data sources, either alone or in combination with existing corporate data.

### 1.2 The need for distributed system

The problems to be solved by distributed systems are mainly the lack of system capacity in single-machine systems and the improvement of system availability. As the business becomes more and more complex, the service also becomes more and more complex. There is a good example that can show this situation: Bloomberg is a large multinational news company with about 20,000 employees and can produce about 500~1000 news and 100~200 videos every day, which is a lot for a news reader. Therefore, a recommendation system is very necessary. However, the data in the database is diverse, so it requires a lot of computation. In addition, the time around the world is different, so there will be news happening all the time. It can be seen that the amount of information that Bloomberg needs to process every day is very high. Thus, the CPU and memory of a single machine can no longer meet the performance

requirements of deploying a huge system. Obviously, the problem can be solved temporarily by improving the system configuration, but no matter how the system configuration is improved, a single machine will always reach a performance bottleneck, so multiple machines are needed to cope with these expanded functions. By splitting the system horizontally (adding machines) and vertically (splitting into multiple subsystems), it becomes a distributed architecture.

In the situation of service on a single machine, if the machine fails and the service hangs, the entire system will crash. Therefore, it is necessary to improve the availability of the system. The high availability of the system means that a group of servers runs no different from a single machine. This means solving the single point of failure problem in the system architecture, increasing redundancy by introducing a distributed architecture, thereby improving the availability of the system, and ensuring the normal operation of services even if a single machine is down.

Based on the above two reasons, we need a distributed system to solve the problems of insufficient system capacity and high system availability.

# Chapter 2

# Dataset analytics

### 2.1 Dataset

The data set used for this analysis is the advertising data collected by Facebook during the 23 months of 03/2020 - 01/2022, which includes the US election held in November 2020. Its main content is on Facebook Published political posts. The data will be recorded into a file every 12 hours, recording the currently active advertisements. Therefore, there will be many duplicates in the dataset (i.e., those ads that run for more than 12 hours). Since the limitation of the cluster, I will use part of this dataset (from 06/2020 to 11/2020) for the investigation. The detail of each feature is available in: https://www.facebook.com/ads/library/api/.

### 2.2 Pre-processing of the dataset

The whole dataset is too large for the cluster's storage, so I used the dataset from 06/2020 to 11/2020, which contains the US Presidential election in November 2020. The first thing is to load the dataset to the zone and read it by Pyspark. Pyspark is an API(Application Programming Interface) of Python, which can run Spark session on Python. Besides, Spark is a fast and general computing engine designed for large-scale data processing.

Since the dataset has duplicate records, we need to eliminate those redundancy. Here, I used the "page_id" features as the unique identifier, which refers to the id number of the advertisement on Facebook web page. After deleted the duplicate rows, the size of the dataset reduces to 40964 tuples. In this report, we want to observe the ad delivery by some entities. Therefore, we mainly focus on the number of advertisements placed by each entity, the amount of funds used, and the main target of the advertisements. Also, in order to reduce the calculation amount, those records that have no sponsor are eliminated. After we done the pre-processing, the next thing is trying to do some analytics on the dataset.

### 2.3 Data analytics and results

First, I focused on certain accounts and estimated the target demographic segment. The reason to do so is that it is difficult to find a specific advertising model among such amount of funding entities. Thus, it is a good alternative to find several typical and representative funding entities.

In order to find those typical funding entities, I observed the total number of ads posted by all funding entities from March to November, and below is the result of the top 20:

| Funding entity | Number of posts |
|---|---|
| Real Voices Media | 222 |
| Metric Media LLC | 87 |
| AARP | 58 |
| DCCC | 51 |
| AMERICANS FOR PROSPERITY | 32 |
| Local Government Information Services | 32 |
| Realtors Political Advocacy Committee | 30 |
| National Association of REALTORS | 29 |
| Protect Our Winters | 28 |
| IOWA DEMOCRATIC PARTY | 27 |
| TECH FOR CAMPAIGNS | 26 |
| Nature Conservancy | 24 |
| Maine House Majority | 21 |
| BIDEN VICTORY FUND | 21 |
| U.S. Term Limits Inc. | 20 |
| Robert William Caton | 20 |
| DONALD J. TRUMP FOR PRESIDENT, INC. | 20 |
| American Heart Association Inc. | 18 |
| American Heart Association | 18 |

We can find from the chart that the 'Real Voices Media' has the highest number of posts, and there are two funding entities are showing very obvious political leanings, which is 'BIDEN VICTORY FUND'(BVF) and 'DONALD J. TRUMP FOR PRESIDENT, INC.'(DPI). Thus, I observed these three funding entities' and selected the target segment of demographic with the largest percentage of each ad.

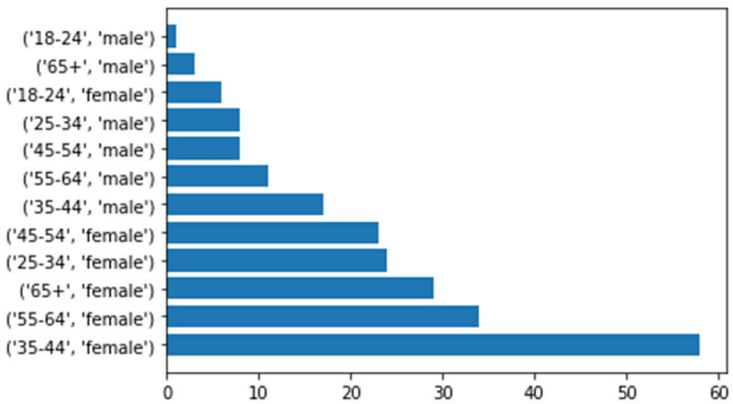



**Figure 1. Main population count for each ad of 'Real Voices Media'**

Fig. 1 shows the result of the counts that how many ads are mainly target at each demographic segment. For example, if one ad is mainly target at female citizens whose age are 35 to 44, then this segment will count once. In this graph we can find that most ads supported by Real Voices Media target at female with age of 35 to 44, and the second largest group is also female but with high age of 55-64.

Then, I made a comparison between two funding entities with obvious political leanings. The graphs are shown below:

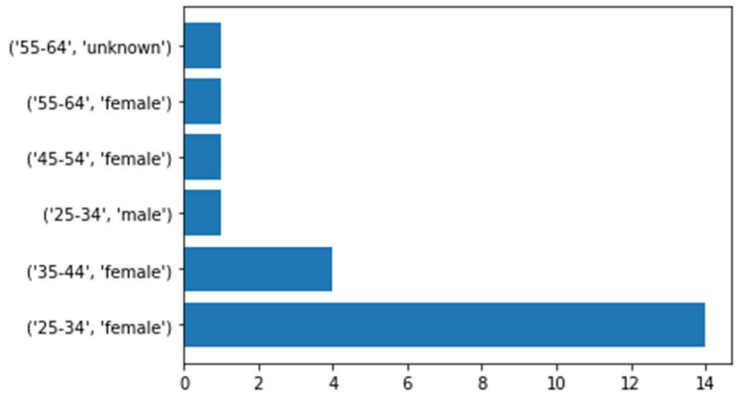



**Figure 2. Main population count for each ad of 'BIDEN VICTORY FUND'**

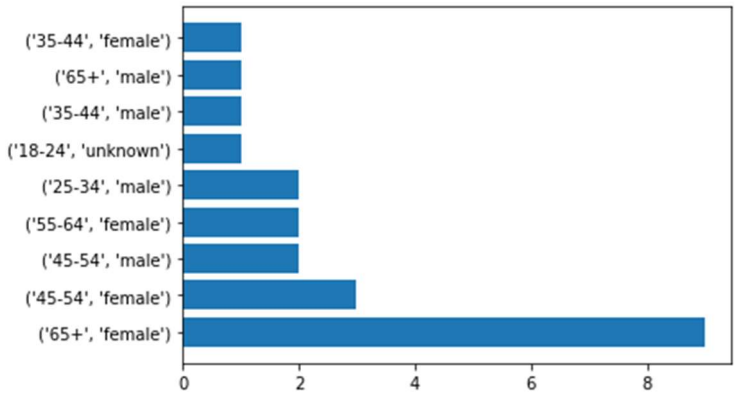



**Figure 3. Main population count for each ad of 'DONALD J. TRUMP FOR PRESIDENT, INC.'**

We can observe from Fig. 2 and 3 that the target population composition of these two funding entities vary widely. BVF focused primarily on outreach to the 25- to 44-year-old female, while the DPI put more ads on senior female group, with 65 or older. On the other hand, BVF seems hardly intended to do much to advertise to male, comparing to DPI, which almost covering all ages in both genders.

# Chapter 3

# Discussion and conclusions

From the results in Chapter 2, we can draw the following conclusions: female group is the main target for advertising during this election for most of the ad funding entities. We know from the distribution of BVF's advertising target population that its main target is young and middle-aged women, which shows that women are playing an increasingly important role in political activities. Moreover, it can also be seen from the results of the general election that most of Biden's votes came from women. At the same time, society is also calling on female to participate in the general election more actively.

# Appendix



In [6]:
```
sc
```

Out[6]: **SparkContext**

Spark UI

| | |
|---|---|
| **Version** | v3.1.2 |
| **Master** | yarn |
| **AppName** | pyspark-shell |

In [1]:
```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

In [43]:
```
json_list = []
for month in range(6, 12):
        json_list.append("/data/ProjectDatasetFacebook/FBads-US-2020" + str(month)
print(json_list)
```

['/data/ProjectDatasetFacebook/FBads-US-202006*', '/data/ProjectDatasetFacebook/FBads-US-202007*', '/data/ProjectDatasetFacebook/FBads-US-202008*', '/data/ProjectDatasetFacebook/FBads-US-202009*', '/data/ProjectDatasetFacebook/FBads-US-202010*', '/data/ProjectDatasetFacebook/FBads-US-202011*']

[Stage 75:==========> (3154 + 149) / 17066]

In [44]:
```
df = sqlContext.read.json(json_list)
```

```
2022-06-21 16:50:16,871 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 131 for reason Container marked as failed: container_1652753568617_1390_01_000177 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:16,871 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 143 for reason Container marked as failed: container_1652753568617_1390_01_000189 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:16,871 ERROR cluster.YarnScheduler: Lost executor 131 on ip-100-64-74-57.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000177 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:16,875 ERROR cluster.YarnScheduler: Lost executor 143 on ip-100-64-74-57.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000189 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:16,880 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 225 for reason Container marked as failed: container_1652753568617_1390_01_000271 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:16,881 ERROR cluster.YarnScheduler: Lost executor 225 on ip-100-64-74-57.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000271 on host: ip-100-64-74-57.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,944 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 137 for reason Container marked as failed: container_1652753568617_1390_01_000183 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,944 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 201 for reason Container marked as failed: container_1652753568617_1390_01_000247 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,944 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Requesting driver to remove executor 176 for reason Container marked as failed: container_1652753568617_1390_01_000222 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,945 ERROR cluster.YarnScheduler: Lost executor 137 on ip-100-64-74-23.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000183 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,948 ERROR cluster.YarnScheduler: Lost executor 201 on ip-100-64-74-23.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000247 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
2022-06-21 16:50:50,950 ERROR cluster.YarnScheduler: Lost executor 176 on ip-100-64-74-23.ap-southeast-2.compute.internal: Container marked as failed: container_1652753568617_1390_01_000222 on host: ip-100-64-74-23.ap-southeast-2.compute.internal. Exit status: -100. Diagnostics: Container released on a *lost* node.
```

In [51]:

```
from pyspark.sql import Row
df = df.dropDuplicates(subset = ['page_id'])
df.count()
```

Out[51]: 40364

In [74]:

```
df.printSchema()
```

```
root
 |-- _corrupt_record: string (nullable = true)
 |-- ad_creation_time: string (nullable = true)
 |-- ad_creative_body: string (nullable = true)
 |-- ad_creative_link_caption: string (nullable = true)
 |-- ad_creative_link_description: string (nullable = true)
 |-- ad_creative_link_title: string (nullable = true)
 |-- ad_delivery_start_time: string (nullable = true)
 |-- ad_delivery_stop_time: string (nullable = true)
 |-- ad_snapshot_url: string (nullable = true)
 |-- currency: string (nullable = true)
 |-- demographic_distribution: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- age: string (nullable = true)
 |    |    |-- gender: string (nullable = true)
 |    |    |-- percentage: string (nullable = true)
 |-- funding_entity: string (nullable = true)
 |-- id: string (nullable = true)
 |-- impressions: struct (nullable = true)
 |    |-- lower_bound: string (nullable = true)
 |    |-- upper_bound: string (nullable = true)
 |-- page_id: string (nullable = true)
 |-- page_name: string (nullable = true)
 |-- region_distribution: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- percentage: string (nullable = true)
 |    |    |-- region: string (nullable = true)
 |-- spend: struct (nullable = true)
 |    |-- lower_bound: string (nullable = true)
 |    |-- upper_bound: string (nullable = true)
```

In [57]:

```
entity_count = df.groupby('funding_entity').count()
entity_count.orderBy(entity_count['count'].desc()).show()
```

```
[Stage 119:==========================================================>(199 + 1) / 200]
+--------------------+-----+
|      funding_entity|count|
+--------------------+-----+
|                null|  642|
|   Real Voices Media|  222|
|    Metric Media LLC|   87|
|                AARP|   57|
|                DCCC|   51|
|AMERICANS FOR PRO...|   32|
|Local Government ...|   32|
|Realtors® Politic...|   30|
|National Associat...|   29|
|  TECH FOR CAMPAIGNS|   26|
| Protect Our Winters|   25|
|  Nature Conservancy|   24|
|IOWA DEMOCRATIC P...|   24|
|  BIDEN VICTORY FUND|   23|
|Robert William Caton|   20|
|Maine House Majority|   20|
|DONALD J. TRUMP F...|   20|
|      Resource Media|   20|
|Protect Our Winte...|   20|
|U.S. Term Limits ...|   20|
+--------------------+-----+
only showing top 20 rows
```

In [59]:
```
entity_count = entity_count.orderBy(entity_count['count'].desc())
entity_count.take(20)
```

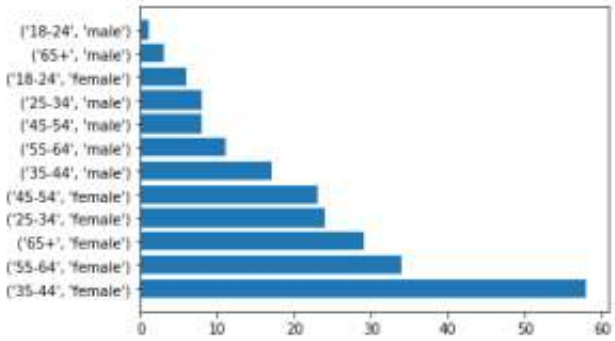
Out[59]:
```
[Row(funding_entity=None, count=644),
 Row(funding_entity='Real Voices Media', count=222),
 Row(funding_entity='Metric Media LLC', count=87),
 Row(funding_entity='AARP', count=58),
 Row(funding_entity='DCCC', count=51),
 Row(funding_entity='AMERICANS FOR PROSPERITY', count=32),
 Row(funding_entity='Local Government Information Services', count=32),
 Row(funding_entity='Realtors® Political Advocacy Committee', count=30),
 Row(funding_entity='National Association of REALTORS', count=29),
 Row(funding_entity='Protect Our Winters', count=28),
 Row(funding_entity='IOWA DEMOCRATIC PARTY', count=27),
 Row(funding_entity='TECH FOR CAMPAIGNS', count=26),
 Row(funding_entity='Nature Conservancy', count=24),
 Row(funding_entity='Maine House Majority', count=21),
 Row(funding_entity='BIDEN VICTORY FUND', count=21),
 Row(funding_entity='U.S. Term Limits Inc.', count=20),
 Row(funding_entity='Robert William Caton', count=20),
 Row(funding_entity='DONALD J. TRUMP FOR PRESIDENT, INC.', count=20),
 Row(funding_entity='American Heart Association, Inc.', count=18),
 Row(funding_entity='American Heart Association', count=18)]
```

In [60]:
```
# rvm = Real Voices Media
rvm_ad = df.filter(df['funding_entity'] == 'Real Voices Media')
```

In [61]:
```
from pyspark.sql.functions import col
rvm_dis = rvm_ad.select(col("demographic_distribution.percentage"), col("demographi
```

In [62]:
```
from operator import add
rvm_dis_rdd = rvm_dis.rdd.map(lambda x: (x.age[x.percentage.index(max(x.percentag
rvm_dis_reduce = rvm_dis_rdd.reduceByKey(add)
rvm_dis_reduce = rvm_dis_reduce.sortBy(lambda x: x[1], False)
rvm_dis_reduce = rvm_dis_reduce.collect()
```

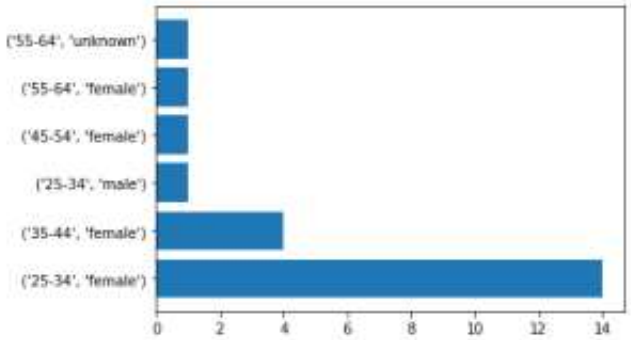
In [63]:
```
import matplotlib.pyplot as plt
import numpy as np
title = []
value = []
for item in rvm_dis_reduce:
    title.append(str(item[0]))
    value.append(item[1])
plt.barh(np.arange(len(title)), value)
plt.yticks([i for i in range(len(title))], tuple(title))
plt.show()
```




In [64]:
```
# bvf= BIDEN VICTORY FUND
from pyspark.sql.functions import col
bvf_ad = df.filter(df['funding_entity'] == 'BIDEN VICTORY FUND')
bvf_dis = bvf_ad.select(col("demographic_distribution.percentage"), col("demographi
```

In [66]:
```
from operator import add
bvf_dis_rdd = bvf_dis.rdd.map(lambda x: ((x.age[x.percentage.index(max(x.percentag
bvf_dis_reduce = bvf_dis_rdd.reduceByKey(add)
bvf_dis_reduce = bvf_dis_reduce.sortBy(lambda x: x[1], False)
bvf_dis_reduce = bvf_dis_reduce.collect()
# print(bvf_dis_reduce)
```

In [67]:
```
import matplotlib.pyplot as plt
import numpy as np
title = []
value = []
for item in bvf_dis_reduce:
    title.append(str(item[0]))
    value.append(item[1])
plt.barh(np.arange(len(title)), value)
plt.yticks([i for i in range(len(title))], tuple(title))
plt.show()
```

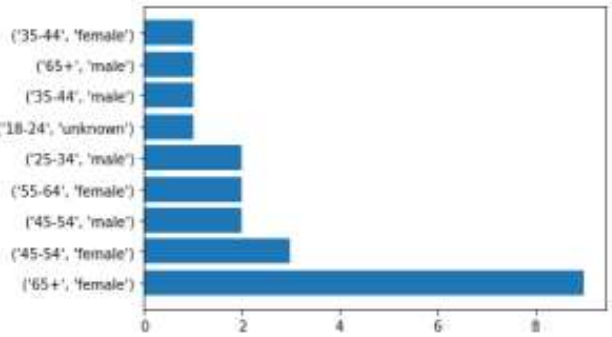



In [68]:
```
# dpi = DONALD J. TRUMP FOR PRESIDENT, INC.
from pyspark.sql.functions import col
```

```
dpi_ad = df.filter(df['funding_entity'] == 'DONALD J. TRUMP FOR PRESIDENT, INC.')
dpi_dis = dpi_ad.select(col("demographic_distribution.percentage"), col("demographi
```

In [69]:
```
from operator import add
dpi_dis_rdd = dpi_dis.rdd.map(lambda x: ((x.age[x.percentage.index(max(x.percenta
dpi_dis_reduce = dpi_dis_rdd.reduceByKey(add)
dpi_dis_reduce = dpi_dis_reduce.sortBy(lambda x: x[1], False)
dpi_dis_reduce = dpi_dis_reduce.collect()
# print(dpi_dis_reduce)
```

In [71]:
```
import matplotlib.pyplot as plt
import numpy as np
title = []
value = []
for item in dpi_dis_reduce:
    title.append(str(item[0]))
    value.append(item[1])
plt.barh(np.arange(len(title)), value)
plt.yticks([i for i in range(len(title))], tuple(title))
plt.show()
```



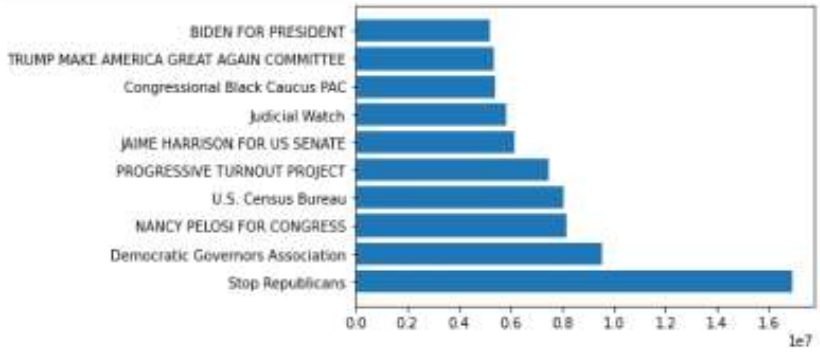

In [15]:
```
from pyspark.sql.functions import isnull
demo_dis = df.select("funding_entity", "spend")
demo_dis.filter(isnull("funding_entity")).count()
```

Out[15]: 128

In [27]:
```
from operator import add
demo_dis = demo_dis.na.drop()
demo_dis_rdd = demo_dis.rdd.map(lambda x: (x.funding_entity, (int(x.spend[1]) + in
demo_dis_reduce = demo_dis_rdd.reduceByKey(add)
demo_dis_reduce = demo_dis_reduce.sortBy(lambda x: x[1], False)
demo_dis_reduce.collect()
demo_dis_reduce = demo_dis_reduce.take(10)
```

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np
```

```
entity = []
spend = []
for item in demo_dis_reduce:
    entity.append(item[0])
    spend.append(item[1])
plt.barh(np.arange(len(entity)), spend)
plt.yticks([] for i in range(len(entity))], tuple(entity))
plt.show()
```




In [26]:

```
print(demo_dis_reduce)
```

[('AMERICANS FOR PROSPERITY', 910428.5), ('Stanley R Jr Odell', 6020.5), ('Jose Luis Lopez', 26788.0), ('Christopher Anthony Vigliotti', 247.5), ('Jason H Beauregard', 497.0), ('Cardinale For Assembly', 594.0), ('Marcel Alonzo Santix', 396.0), ('Natrina Gaar Altieri', 396.0), ('William Douglass Campaign Fund, Republician for District 1 County Commissioner', 1188.0), ('JoAnna for Wisconsin', 495.0)]