Analysis of Advertisement Data with Traffic Patterns Using The Hadoop Ecosystem

Lawrence Gates
gateslm@rose-hulman.edu

advised by Dr. Sriram Mohan
mohan@rose-hulman.edu

May 15, 2017

Contents

Abstract

The amount of data collected daily is rising. Various themes of data, such as temperature data, agricultural statistics, sales numbers, traffic data, and advertisement data are typically gathered. Using datasets, such as billboard advertisement data and congestion traffic, I have been able to effectively find areas of congestion for a given region or segment along with the best advertisement for the given region or segment. The Hadoop Ecosystem was used in combination with the tools Apache Pig and Apache Hive. The traffic data was aggregated to find the statistics for a given region and segment on a given day. The randomly generated advertisement data was joined with each region and segment for a given day. For each of the days, a common time frame of overlap was found and was used to help determine if any congestion occurred in a given region and segment. To show the effectiveness of the algorithm, different time frames were used to show the impact of a certain advertisement in a given region and segment.

1 Motivation

Many cities and states collect traffic statistics for roads, either to show traffic flow on display boards or to gather statistical information. This thesis serves as an example that any result can be used in other cities in order to attempt to record traffic data to find patterns and optimize advertisement locations.

Since the collection and storage of data has become easier with wireless sensors and the availability of cheaper storage, collecting traffic data within a well-funded location does not pose a problem. The only limitation is the sensors used to collect traffic data. Once a city has the data, providing data to the proposed algorithm should produce a result. With the creation of the algorithm, inputting traffic data should not be a hassle, and will provide useful marketing data for a city.

2 Theory and Concepts

The essential points of this idea is using Big Data and a Big Data management system. The Big Data management system that is used is The Hadoop Ecosystem. Additional details of The Hadoop Ecosystem in respect to this thesis can be found in Section 4.4.

2.1 What is Big Data?

Big Data is "information asset characterized by such a High Volume, Velocity, and Variety to require specific Technology and Analytical Methods for its transformation into Value" [10]. The idea of Big Data is not limited to large companies, for instance, retailers trying to understand consumers shopping patterns, but also scientists performing analysis on data collected from experiments in a more timely manner. Based on the tool and environment used to master Big Data for a project, setting up and deploying the tool is not a simple task. Getting results from Big Data environment has the process of [9]:

1. Acquisition
2. Information extraction and cleaning
3. Data integration
4. Modeling and analysis
5. Interpretation and deployment

Big Data is not guaranteed to produce a result with any given dataset. Data is not guaranteed to produce a pattern. A success from Big Data can come from showing a pattern or trend within a given dataset or by finding a lack of trend in a large amount of data. By using a Big Data environment or tools, analysis of the data can produce a result based on the user designs for the modeling and analysis stage as well as the interpretation and deployment stage, not just running the data through the tool itself.

3 Literature Review

Currently, there has been no academic research into using Big Data and coordinating traffic flow with advertisements. However, there has been plenty of academic research and articles involving Big Data and traffic datasets, along with many articles about the applications of using Big Data and advertisements.

Smith and Demetsky [11] discussed the importance of intelligent transportation systems, at a time before Google Maps and traffic flows were available on the internet. Their paper discusses the need to have forecasting models, using several time prediction techniques, such as machine learning and historical average, to determine traffic forecasting. With the datasets being collected, this project is using historical averaging with the available data. Daas, Puts, Buelens, and van den Hurk [12] highlight that, at the time, Big Data was heavily viewed from an IT-perspective and focus[ed] on soft[ware] and hardware issues. In the Big Data case study done by Chen, Pao, and Lee [4] about the study of traffic loop detection data, a successful plot of peak hours and vehicle flow was shown. The case study shows the high potential of finding a trend in traffic data.

The Big Data and advertisement side of academic articles talked about the collection and use of user data. Couldry and Turow [5] elaborate on personalized advertising constantly mining personalized data. The article also looks more broadly at the consequences of embedding Big Data use in advertising, which is not of concern for this thesis. Bughin, Chui, and Manyika [6] discuss the opportunities companies take with using the data available for a web-based company. The availability of data for marketing is not scarce due to the expanding amount of data, which is to be useful in determining advertisements to display in a certain area.

4 Data and Operationalization

To successfully complete the thesis, a vast amount of traffic data was needed and advertisement data to show a relationship between the two datasets as well as show the different types of analysis compared against each other and the varying results that were available.

4.1 Data

The *Chicago Data Portal* [3] is the preferred mechanism to share all the data that is collected throughout The City of Chicago daily. The *Chicago Data Portal* was established through the Open Data Executive Order

(No. 2012-2) to add a level of transparency to the data being collected by Chicago officials.

The data that is essential for determining traffic flow within Chicago are the Chicago Traffic Tracker - Congestion Estimates by Segment and the Chicago Traffic Tracker - Congestion Estimates by Regions datasets. Please refer to Appendix A.1 and A.2 for information about the downloaded spreadsheets.

4.1.1 Chicago Traffic Tracker - Congestion Estimates by Segments

The segment estimations contains the current estimated vehicle speed for about 1250 segments covering 200 miles of arterial roads [2] in Chicago. The spreadsheet does not update every segment, which is due to an uploading problem on the data portal's end. The segment estimations are roughly collected every 20 minutes by the Data Portal. The important information from each row in the dataset are:

1. Location of the traffic flow
2. Area of the traffic flow
3. Speed of the traffic
4. Time stamp the data was updated

Since the segments are sub-areas of a region, more options for finding optimal advertisement placement will be available, compared to having only 29 regions to find optimal advertisement placement. Refer to the Appendix A.1 for information about the columns of the dataset. Data was collected between September 18th, 2016 to February 23rd, 2017. Details of the automatic download is shown in Algorithm 1 and in the repository in Section 6.5 under the *Scripts/automatedRun.sh* file.

The algorithm starts by getting the previous region and segment files from the local directories. The current date time is retrieved and the files appropriate file names for region and segment are initialized with the current date time. The new region and segment files are downloaded from The City of Chicago Data Portal. Downloading the *csv*, *json*, and *xml* for each segment, each of these files are stored in the appropriate region and segment directory. The new files are compared to the previous files, checking if the contents are the same. If the contents of each set of files are the same, the new files with the same content are deleted. After the counter of 60 minutes, an email is sent with a status update.

4.1.2 Chicago Traffic Tracker - Congestion Estimates by Regions

The region estimations contains the current estimated congestion for the 29 traffic regions [1] in Chicago. The region spreadsheet stored on the *Chicago Data Portal* has updated all 29 regions. The region estimations are to be collected every 10 minutes by the Data Portal. The important information from each row in the dataset are the boundary of a given region, speed of the traffic, and the time stamp the data was updated. Since the 29 regions each have a given number of segments, a given region will provide an overview of the average speed of all the segments. By having segments in each region, a pattern can be found about the given region. This can be used to find the best advertisement over the region, then finding a better fit of the advertisement in a smaller defined area, such as a segment. Refer to Appendix A.2 for information about the columns of the dataset. Data was collected between September 18th, 2016 to February 23rd, 2017. Details of the automatic download is demonstrated in Algorithm 1 and in the repository 6.5 under the *Scripts/automatedRun.sh* file. The datasets are downoaded along with the segment datasets.

Algorithm 1 Automated Download from The City of Chicago Data Portal

```
1:  procedure MAIN
2:      regPrevF ← ""
3:      segPrevF ← ""
4:      while true do
5:          regPrevF ← GetPrevRegFile
6:          segPrevF ← GetPrevSegFile
7:          dateTimeNow ← GetSysTime
8:          regFileNm ← "regEst −"$dateTimeNow
9:          segFileNm ← "segEst −"$dateTimeNow
10:         DownloadToFiles(regFileNm, segFileName)
11:         if cmp(regPrevF, regFileNm) ≠ SAME then
12:             RemoveNewFile(regFileNm)
13:         if cmp(segPrevF, segFileNm) ≠ SAME then
14:             RemoveNewFile(segFileNm)
15:         sleep(5m)
16:         counter ← counter + 5
17:         if counter < 60 then
18:             DATE = GetSystemTime
19:             counter ← 0
20:             SendStatusUpdateviaEmail
```

4.2 Advertisement Data

Due to the confidentiality and cost of purchasing advertisement data, using accurate data gathered by companies was not available for the scope of the thesis. The type of a billboard advertisement that was successful in a given area and the length of time the advertisement that was left on the billboard was needed in order to find a trend in traffic congestion and advertisement. To overcome this set back, advertisement data was randomly generated for each region and segment to represent an advertisement for the region and segment itself for a given time frame. The data manually generated randomly assigns an advertisement theme from Appendix A.3. A theme for a given region will last

from a single day up to 30 days, which would have a theme randomly assigned for a new given time period.

Details of the random advertisement generation for every segment and region is shown in Algorithm 2 and in the repository 6.5 under the subproject *Projects/GenerateThoroughBillboardData*. The algorithm loops through all the segments and regions, starting with the first date of the traffic data. While looping through the days, a random rating for the theme is assigned from the list of advertisement themes, a random rating of success for the advertisement theme, and a random length of time for the advertisement theme to stay active, from 1 day to 30 days. The random advertisement theme and rating stays the same for the length of the random time length while the day increments. Once the length of the random time is completed, a new random theme, rating, and time length is assigned. This process continued until the last day of data collection.

Algorithm 2 Randomly generate advertisement data for segments and region.

```
1:  procedure MAIN
2:      segmentFile ← GENERATESEGMENTDATA
3:      regionFile ← GENERATEREGIONDATA
4:  procedure GENERATESEGMENTDATA
5:      for s := 1 To 1310 do
6:          curDt ← startDate
7:          while curDt BEFORE endDate do
8:              adThm ← RAND(AD_THM_COUNT)
9:              r ← RAND(MAX_RATING)
10:             ad_len ← RAND(AD_LEN)
11:             for date := 1 TO ad_len do
12:                 curSeg ← SegMdl(s, curDt, adThm, r)
13:                 curDt ← curDt + 1
14:                 segAds[ ] ← curSeg
        return segAds
15: procedure GENERATEREGIONDATA
16:     for rN := 1 To 29 do
17:         rg ← REGION_NAMES[rN]
18:         curDt ← startDate
19:         while curDt BEFORE endDate do
20:             adThm ← RAND(AD_THM_COUNT)
21:             r ← RAND(MAX_RATING)
22:             ad_len ← RAND(AD_LEN)
23:             for date := 1 TO ad_len do
24:                 curReg ← RegMdl(rg, curDt, adThm, r)
25:                 curDt ← curDt + 1
26:                 regAds[ ] ← curReg
        return regAds
```

4.3 Census Data

To overcome the lack of advertisement targeting data, census data could be used as a substitute. Census data gathered by the United States contains many statistics to determine factors of a given area, with such statistics as: age, education, housing, income, poverty, race, and veterans. Users can find data based on a state, county, city, town, or zipcode, available on the United States Census Bureau. The City of Chicago is comprised of 85 zipcodes, but the number of zipcodes all the regions overlap is 74 zipcodes, displayed in the Figure 1.

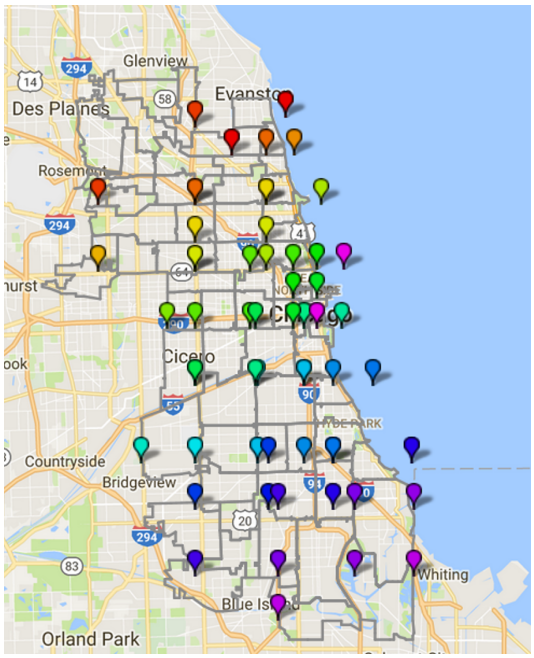

Figure 1: Map of Chicago overlayed with the zipcode (gray) and the North-East corner and South-West corner matching in color for all 29 region

To use zip codes for the census data is a large task in itself. A region or segment was not bounded by a given zip code, which allowed overlap for zip codes within a region. To determine the amount a given zip code overlays inside a region would require a significant amount of mathematics and accurate geographical data to ensure that the area of a zip code was factored in correctly to a region. The amount of work was beyond the scope of the thesis, described in Future Work

In the repository 6.5, a subproject called *Projects/DataCollection* used a library called *jsoup*. *jsoup* is used to parse real-world HTML. The project takes the URLs of census data for given zipcodes and extracts data from HTML tables to CSV readable files. The data downloaded is essential for incorporating census data to zipcodes without finding or paying for easily accessible datasets.

The *jsoup* tool is designed for Java, allowing for easy extraction of data. To extra census data from a zipcodes, the general layout of a given site needs to be understood. Following the HTML on the site, a general flow is located and *jsoup* can walk the HTML tree and pull out the HTML element nodes. Once the specific nodes are collected, the data can be stored separately from the HTML and then placed all together in a spreadsheet for easy reading and analysis.

4.4 Hadoop Ecosystem

The Hadoop Ecosystem was created by Doug Cutting, who created Apache Lucene. Hadoop takes advantage of using a distributed file system, the Hadoop Distributed File System (HDFS), to handle storage needs of large files that need to be browsed. To handle large jobs, a MapReduce implementation was created to accomplish the job in The Hadoop Ecosystem [13].

4.4.1 Hadoop Distributed File System

Datasets have the potential to be larger than the storage of a single computer, resulting in breaking the data apart to store on multiple computers. The multiple computers are not required to have expensive hardware, but rather can be commodity computers. The computers, each a node in the cluster, are designed to store and pass data between each other and perform jobs on the node itself. An additional feature of HDFS is handling very large files, that can be split up across multiple nodes in a server, allowing for easy access to specific parts of file for processing. The streaming of data across the nodes is essential to follow the most efficient data processing pattern, "write-one, read-many-times pattern" [14], to allow a dataset to be copied from the originating node and have various operations performed on the whole or portions of the dataset over time [15].

4.4.2 MapReduce

MapReduce jobs are specifically designed for large dataset problems by splitting up the data to be run in parallel. In The Hadoop Ecosystem, the job submitted on data is separated into two different tasks: map tasks and reduce tasks. The map task takes the data and applies a filter and will sort the data into a desired output based on parameters and store the mapped result back to disk, to avoid replicating an intermediate value. The mapped results will then be processed through a reducer, where an operation is performed, such as aggregation, among the filtered data. In figure 2, multiple maps of data being split are being pushed into the reduce function to get an output based on some sort of user-defined function.

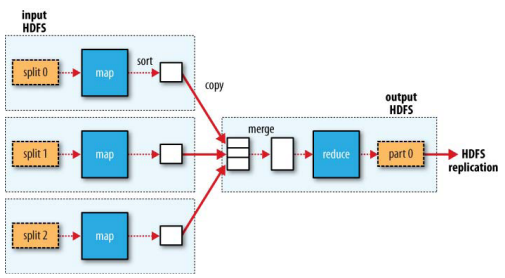

Figure 2: MapReduce data flow with a single reduce task [7]

4.5 The Hadoop Ecosystem Tools

The Hadoop Ecosystem has a library of predefined projects that can be incorporated into the ecosystem. These projects often provide a high level language to process data in a manner similar to native MapReduce operations, but allowing a user to forgo the requirement of making such a MapReduce program. Tools can be language-neutral data serialization systems, storage format systems, extracting data from structured data stores, and MapReduce abstraction. All these available tools can be used along side a user-defined MapReduce function.

4.5.1 Apache Pig

Apache Pig allows users to take MapReduce jobs to a higher level of abstractions when processing datasets. Compared to a MapReduce job, the time required to construct a Pig script to process data is extremely shorter and much more straightforward. Many operations that are supported by Pig are not ideal in MapReduce jobs, such as handling a join of data. Data can be passed into Pig to allow for cleaning, removing columns from spreadsheet datasets, or removing rows that have data that does not match a type or is a certain value [16].

4.5.2 Apache Hive

Apache Hive is a data warehouse framework that utilizes the way data is stored in HDFS and accesses such data in SQL type format. Hive handles processing a query on a large number of datasets stored in HDFS, allowing for the creation of tables, joining results together easily, and aggregating results for additional processing [17].

5 Architecture

Due to the large amount data stored within the dataset, an architecture flow was needed to do a complete analysis of the data and combined with advertisement data. The download data flow pulls in *Congestion Estimates by Segment* and *Congestion Estimates by Region* from the Chicago Data Portal. Three files are downloaded, as back-up copies in alternative forms, as visualized in Figure 3.

Using an available API, data is downloaded directly to a node used primarily for initial storage and alerting status of files download. The details of the download are mention in Algorithm 1. Since the data is stored remotely from the Hadoop cluster, the specific file format, *csv*, that is used is transferred to the cluster, leaving data that is not used off the cluster, *xml* and *json*.

The spreadsheet files for regions and segments are transferred to the cluster and stored in HDFS. Once

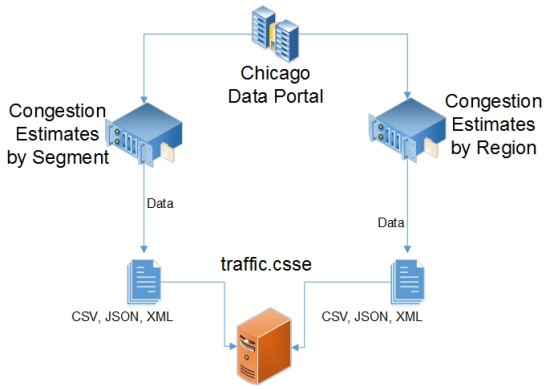

Figure 3: Dataflow of downloading data from the Chicago Data Portal.

on the cluster, the traffic data for regions and segments are cleaned and processed separately. The region traffic data has the the region name, speed and date time for each row of every region spreadsheet stored in the folder matching the day the data was collected.

The data stored in the cluster can be retrieved, which is then passed to The Hadoop Ecosystem with various aggregation and analysis done on the given data, visualized in Figure 4. The region and segment traffic data is cleaned and processed in a similar format, storing the region name and segment id, speed and date time for each row of every region and segment spreadsheet stored in the folder matching the day the data was collected. For each day of every region and segment, the traffic data is calculated to form sums of specific congestion classifications. Using the descriptions from the Chicago Data Portal, the speeds are used to classify the type of congestion.

The advertisement data is stored in HDFS, which is processed by a Pig script. The Pig script separates the advertisement data by segment and region, then storing the advertisement ratings for a given segment and region into separate folders. The advertisement data requires less cleaning and aggregation initially, as compared to the region and segment data. The advertisement data is randomly generated, thereby avoiding noise, removing the need for such cleaning.

The advertisement data is joined with the region and segment traffic data, as shown in the middle of Figure 4. After combining the data two different datasets, forming region and segment traffic advertisement. Each one of the theme sets are processed through the MapReduce jobs, separated into four types of date grouping. The data is either all grouped together, by week of a year, month of a year, or by year. Each one of the groupings is then exported to a Hive table allowing for the aggregating of the highest for each region and segment individually, based on the grouping of the date, and stored into a final result.

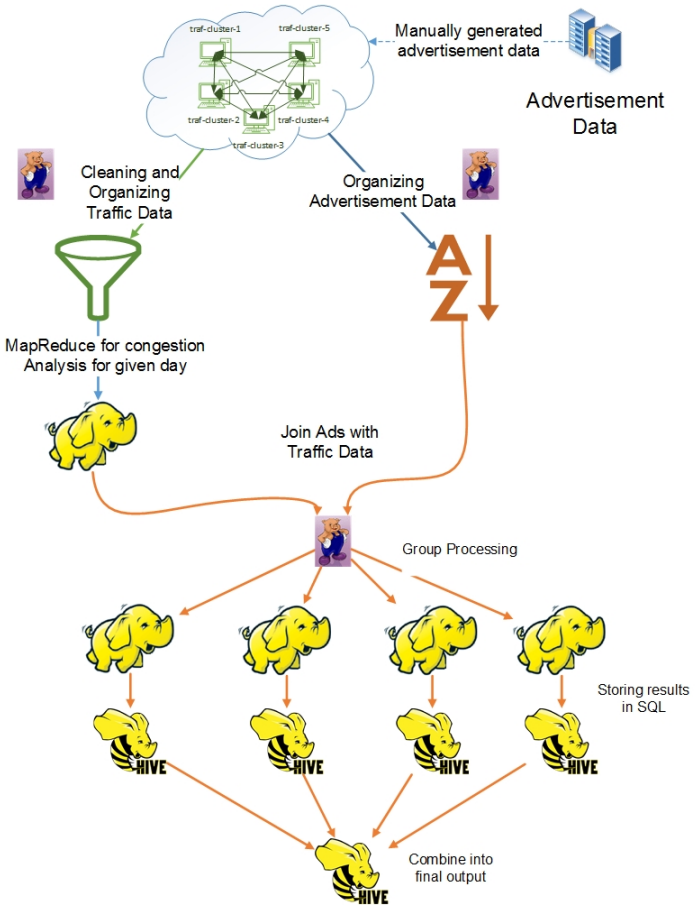

Figure 4: Dataflow for processing the traffic and advertisement data

6 Algorithms

Pig and Hive help aggregate the data, however a more unique user-defined function is needed to continue processing. To handle the classification of congestion for each date time, the speed is collected and the data is aggregated to find the best advertisement for that region or segment.

6.1 Advertisement Storage

The data is easily transferred from the initial spreadsheet file to a formatted file on the cluster. This operation was done with a Pig script for both the segment and region advertisement data. An example dataset is shown in Figure 6.1. The segment advertisement data looks very similar, except the first column is *Segment ID*, rather than *Region Name*.

6.2 Traffic Analysis

Once the traffic data is downloaded, it needs to be cleaned and sorted. Then traffic data and the advertisement data need to be integrated together. To success-

Region Name	Date	Theme	Rating
Near South-Douglas	2017-13-2	Religious	5
Near South-Douglas	2017-14-2	Religious	5
Near South-Douglas	2017-15-2	Religious	5
Near South-Douglas	2017-16-2	Religious	5
Near South-Douglas	2017-17-2	Religious	5
Near South-Douglas	2017-18-2	Religious	5
Near South-Douglas	2017-19-2	Religious	5
Near South-Douglas	2017-20-2	Restaurant	7
Near South-Douglas	2017-21-2	Restaurant	7
Near South-Douglas	2017-22-2	Restaurant	7
Near South-Douglas	2017-23-2	Restaurant	7
South West Side	2016-18-9	Theater	1
South West Side	2016-19-9	Theater	1
South West Side	2016-20-9	Theater	1
South West Side	2016-21-9	Theater	1
South West Side	2016-22-9	Theater	1
South West Side	2016-23-9	Theater	1
South West Side	2016-24-9	Theater	1

Figure 5: Subset of data from region advertisement data that was randomly generated by Algorithm 2. The algorithm that controls the generation for random advertisement data is shown in Section 4.2

fully process the traffic data and combine with the advertisement data, two different algorithms are needed.

6.2.1 Cleansing Data

The data obtained from The City of Chicago Data Portal is not set up to be directly integrated into the analysis algorithms, requiring the datasets to be parsed and stored in a format more suitable to be processed through the algorithms. The total number of CSV files downloaded were:

- Region Files: 14800
- Segment Files: 14829

Figure 6 shows the data flow for taking the downloaded segment data and removes any rows in the data that do not match the criteria, for more detail, refer to Appendix A.1. Such data could be the column names from the spreadsheet and data that does not match the correct data type format. Following that, using the start date when the downloading began and the date the downloading was ceased, any date that did not lie within that range was removed. Some rows in various datasets had dates that ranged from weeks or months older than the rest of the rows in the datasets. Such data would be outliers and would not contribute to the calculations for traffic, such as dates before September 18th, 2016. Every row from all spreadsheets are then stored grouped by the date collected and stored in the appropriate folder on the cluster.

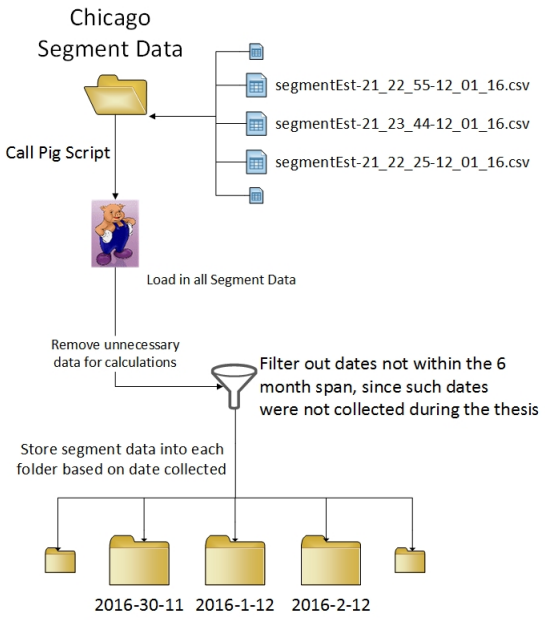

Figure 6: Dataflow for cleaning and sorting the downloaded segment datasets

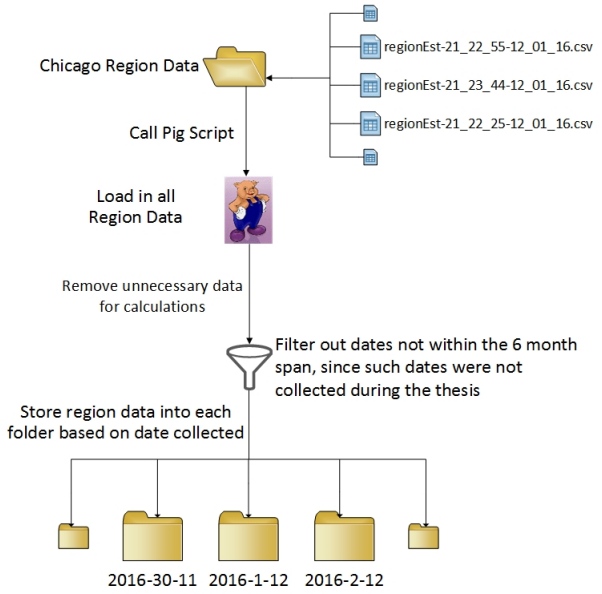

Figure 7: Dataflow for cleaning and sorting the downloaded region datasets

6.2.2 Day Analysis

Region data will take all the regions with the matching name and day to find the number of data points collected for a given day, number of times the speed is classified as high flow congestion, medium high flow congestion, medium flow congestion, light flow congestion, free flow congestion, the start time for the longest period of time of consecutive high flow and medium high flow congestion, and the end time for the high

flow and medium high flow congestion, and the average speed during the time period. Algorithm 3 demonstrates the work of the reducer. All the counters are carried forward to be used as statistics and weights for determining if certain days are valuable to the analysis.

Algorithm 3 Reducer from the MapReduce of analyizing each data for every region and outputting the data.

```
1: procedure REDUCE(key, values)
2:     //Initialize values to zero
3:     dc ← 0                                  ▷ dc = Data Count
4:     hfc ← 0                                 ▷ hfc = High Flow Count
5:     mhfc ← 0                                ▷ mhfc = Medium High Flow Count
6:     mfc ← 0                                 ▷ mfc = Medium Flow Count
7:     lfc ← 0                                 ▷ lfc = Light Flow Count
8:     ffc ← 0                                 ▷ ffc = Free Flow Count
9:     sth = []                                ▷ List of speeds at time
10:    for value : values do
11:        dc++
12:        speed = value.speed
13:        time = value.time
14:        //Increment flow count based on speed
15:        if 15 > speed >= 10 then
16:            //mhfc value
17:            sth[] ← value.speed, value.time
18:        if 10 > speed >= 0 then
19:            //hfc value
20:            sth[] ← value.speed, value.time
21:    sth.sort()                              ▷ Sort on time
22:    ct = []                                 ▷ Periods of congestions.
23:    ch ← NULL                               ▷ New congestion holder
24:    counter ← 0                             ▷ Used for average speed
25:    for i : sth do
26:        //Find consecutive times of congestion
27:        if ch == NULL then
28:            startTime ← NOW
29:            endTime ← NULL
30:            avgSpd ← i.speed
31:            ch ← startTime, endTime, avgSpd, (holders)
32:            //Holders ← dc, hfc, mhfc, mfc, lfc, ffc
33:        if ch NOT IsConsecutive(i) then
34:            ch.endTime ← NOW
35:            ch.speed ← AVG(ch.speed, counter)
36:            ct ← ch
37:            ch ← NULL                       ▷ New congestion holder
38:            counter ← 0                     ▷ Used for average speed
39:        if ch IsConsecutive(i) then
40:            ch ← (holder)++
41:            counter++
42:            ch.speed ← ch.speed + i.speed
43:    maxCongHolder ← MAX(ch)
44:    STORE(maxCongHolder)
```

Segment data will take all of the segments with the matching identifier and day to find the number of data points collected for a given day, the number of times the speed is classified as high flow congestions, medium flow congestion, free flow congestion, the start time for the longest period of time of consecutive high flow, and the end time for the longest period of time of consecutive high flow, and the number of negative speeds. The negative speeds were unique to segment data, showing that there has been no Chicago Traffic Authority buses through this segment, denoting that the segment has not been tracked. Algorithm 4 shows the work of the reducer. Algorithm 4 is similar to 3, with the exception of different counters used. All the counters are carried forward to be used as statistics and weights for determining if certain days are valuable to the analysis.

Algorithm 4 Reducer from the MapReduce of analyizing each data for every segment and outputting the data.

```
1: procedure REDUCE(key, values)
2:     //Initialize values to zero
3:     dc ← 0                                  ▷ dc = Data Count
4:     hfc ← 0                                 ▷ hfc = High Flow Count
5:     mfc ← 0                                 ▷ mfc = Medium Flow Count
6:     ffc ← 0                                 ▷ ffc = Free Flow Count
7:     ns ← 0                                  ▷ ns = Negative Speeds
8:     sth = []                                ▷ List of speeds at time
9:     for value : values do
10:        dc++
11:        speed = value.speed
12:        time = value.time
13:        //Increment flow count based on speed
14:        if 15 > speed >= 0 then
15:            //hfc value
16:            sth[] ← value.speed, value.time
17:    sth.sort()                              ▷ Sort on time
18:    ct = []                                 ▷ Periods of congestions.
19:    ch ← NULL                               ▷ New congestion holder
20:    counter ← 0                             ▷ Used for average speed
21:    for i : sth do
22:        //Find consecutive times of congestion
23:        if ch == NULL then
24:            startTime ← NOW
25:            endTime ← NULL
26:            avgSpd ← i.speed
27:            ch ← startTime, endTime, avgSpd, (holders)
28:            //Holders ← dc, hfc, mfc, ffc, ns
29:        if ch NOT IsConsecutive(i) then
30:            ch.endTime ← NOW
31:            ch.speed ← AVG(ch.speed, counter)
32:            ct ← ch
33:            ch ← NULL                       ▷ New congestion holder
34:            counter ← 0                     ▷ Used for average speed
35:        if ch IsConsecutive(i) then
36:            ch ← (holder)++
37:            counter++
38:            ch.speed ← ch.speed + i.speed
39:    maxCongHolder ← MAX(ch)
40:    STORE(maxCongHolder)
```

6.3 Grouping Analysis

After the MapReduce jobs complete, data for every day between the start date and the end date for segments and regions exist and a theme and rating for an advertisement exists for a segment and region. Using Pig,

the traffic statistics and advertisement for a day and a given region and segment are joined together.

After joining the advertisement data and the day analysis data, finding the best theme for a region and segment is be done using different time frames. By splitting up the data that fall on the same week of a year, month of a year, year, or keeping all the data together, there is a possibility of seeing a different outcome in determining a best theme during those time periods for each region and segment.

Algorithm 5 Reducer from the MapReduce of determining the best theme for a region and segment based on the time grouping. Different time groupings can be processed through this reducer. The difference between region and segment calculations are minute details in the code.

```
1: procedure REDUCE(key, values)
2:     hcl ← []                          ▷ hcl = High Congestion List
3:     th ←<>                            ▷ th = Theme Holder
4:     //Maps ← theme− > totals, maxRating
5:     td ← 0                            ▷ td = Total Days
6:     for v : values do
7:         //v ← sT, eT, avgSp, t, r,
8:         //sT(Start Time), eT(End Time)
9:         //avgSp(AverageSpeed), t(theme), r(Rating)
10:        td + +
11:        if VALID(v.sT)ANDVALID(v.eT) then
12:            hc ← DUMPATTR(v)
13:            hcl ← hc
14:            th ← v.t                  ▷ Handles theme string
15:            th ← MAX(th.maxRating, v.rating)
16:    if hcl.size/td < .5 OR hcl.empty == TRUE then
17:        RETURN(Ignore)
18:    hcl ← OVERLAP(hcl)                ▷ Find overlap of times
19:    if GETMAXSPEED(hcl) < 25 then
20:        SORT(hcl)
21:        max ← MAX(hcl)
22:        thm ← max.theme
23:        RETURN(max, td, th.getRating(thm))
24:    RETURN(Ignore)
```

The reducer loops through the given values and gets times of high congestion that did not start and end at the exact same time. Using multiple maps, saving the maximum rating for a theme and the number of times the theme appears to easily update the values for a given theme. After looping through and finding all the high congestion times, various statistics that are calculated dynamically or provided from the day analysis factor in early if the time period and region are worthy of determining a theme. If low enough, the algorithm ignores the region for the time period. Otherwise, the algorithm attempts to combine all the congestion hour times to remove overlap for the given time period of a week, month, year, or of all time. Taking the largest congestion time, the segment or region, theme, and rating are returned for the given time period.

6.4 Querying the Results

The traffic and advertisement sets for regions and segments are joined together then broken up by week, month, year, or all time to show the best advertisement for a given region or segment. Comparing the time break ups is not initially obvious, so the data needs to be aggregated to make comparisons easy between the different time periods.

The results need to be presented so that a region and segment for a week, month, year, or all time have a similar layout. Using The Hadoop Ecosystem and Hive, the resulting data from Algorithm 5 can be queried and aggregated. All data produced by the algorihm Section 6.3 are imported into a table structure in Hive. For the time periods of week, month, and year, the maximum rating for the longest time period an advertisement is shown over all time period break ups.

6.5 Code Repository

The code can be found at the following GitHub repository: gateslm/CSSE-Senior-Thesis [8]

7 Results and Discussion

To see a subset of the results described before or a link to the full results, please refer to Appendix A.5 and A.4.

The outcome of region and segment results is as expected, however the result needs to be treated with caution as advertisement data is randomly generated. The results for any given region and segment does show that it is obvious that a billboard in a certain area would be highly ineffective, due to traffic flow in the area. This shows as a proof of concept that advertisement and traffic data can be aggregated together to provide evidence of advertisement do depend on traffic flow.

Looking at how the data is grouped by week, month, year, or all together also shows the impact of a given length of an advertisement. Looking at the complete dataset, the algorithm found that some shorter time spans for a region and segment can be effective placements for billboards and the appropriate theme to use for that region/segment in a given time frame.

7.1 Region Result and Discussion

The region results proved quite interesting. A large amount of advertisements and ratings stayed the same across different time periods (month, year, all time), but the week advertisement theme and rating would change. This can be associated with the algorithms found ties for advertisement lengths and ratings, and alphabetical order would be the determining factor for the resulting theme.

Few regions had no themes and ratings that were all "ignored" for week, month, year, and all together. From the spreadsheet in the Appendix A.2, a single row had month, year, and all together "ignored". A week with an "ignored" value is not an anomaly, since the length of the congestion is long enough for a week, but not long enough to be a congestion period for a larger time period, such as month or year. A single week of the year has enough congestion to be considered for placing an advertisement.

7.2 Segment Result and Discussion

The segment results had little variation. A large amount of advertisements and ratings stayed the same compared between each month, year, and all time, but the week advertisement theme and rating would change. This can be connected to the the fact that the algorithm would have ties of advertisement lengths and ratings, and alphabetical order would be the determining factor for the resulting theme.

Many segments had no themes and ratings that were all "ignored" for week, month, year, and all together. By reading the description column on The City of Chicago Data Portal, many segments were no longer being recorded and had been left in the data sets. From the spreadsheet in Appendix A.1, a vast amount of rows had month, year, and all together "ignored." A week with an "ignored" value is not an anomaly, since the length of the congestion is long enough for a week, but not long enough to be a congestion period for a larger time period, such as month or year. A single week of the year has enough congestion to be considered for placing an advertisement.

8 Future Work

8.1 Data

To substantially improve the accuracy of results would require different types of data. In this paper, all advertisement data is manually generated to be joined with region and segment traffic data. Advertisement data aimed at a specific city is not often available to the public due to privacy concerns and business practices. To get the success of billboards in a city, one would need to acquire the themes for the billboards, the time a given theme is presented on a billboard, and the location of all the billboards in a city.

As described in Section 4.3, census data can provide a variety of useful statistics that can factor in the theme of an advertisement much better than comparing the results of themes on a billboard. In a future extension of the thesis, census data could be incorporated into the regions and segments of Chicago or another city. By finding the percentage a zip code overlaps with a region or a segment, the influence of a certain type of advertisement could potentially be more influential than statistics based on studies of advertisement successes. Finding accurate listings of the zip codes in an easy to interpret file and access to current census data is needed.

The amount of traffic data collected from The City of Chicago Data Portal was for the span of approximately 6 months, from September 18th, 2016 to February 23, 2017. To compare the success of aggregating the data based on week, month, year, or overall collection, a longer range of data would be needed to show relations. If data had been collected for over a year, then comparisons to similar times of a year could have been more obvious and highlighted patterns in the data.

8.2 Machine Learning

An initial idea was to incorporate machine learning into the thesis could not be used. The Hadoop Ecosystem contained a tool that had a machine learning library, *Spark*, could be used to apply filters over the data to aggregate the data a completely different way. Incorporating a type of machine learning into the data aggregation process could be ideal to show another potential overlap for Big Data to find patterns in the data. Due to time constraints and learning curves needed for software, the application of machine learning was not incorporated into this thesis.

9 Bibliography

References

[1] Chicago traffic tracker - congestion estimates by region. https://data.cityofchicago.org/transportation/chicago-traffic-tracker-congestion-estimates-by-re/t2qc-9pjd/about.

[2] Chicago traffic tracker - congestion estimates by regions. https://data.cityofchicago.org/transportation/chicago-traffic-tracker-congestion-estimates-by-re/t2qc-9pjd/about.

[3] City of chicago - data portal. https://data.cityofchicago.org/.

[4] X. Y. Chen, H. K. Pao, and Y. J. Lee. Efficient traffic speed forecasting based on massive heterogenous historical data. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 10–17, Oct 2014.

[5] Nick Couldry and Joseph Turow. Big data, big questions— advertising, big data and the clearance of the public realm: Marketers' new approaches to the content subsidy. *International Journal of Communication*, 8(0), 2014.

[6] William Dutton, Kris Pister, Hal Varian, Rob Bernard, Rob Salkowitz, Jacques Bughin, Michael Chui, and James Manyika. Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. *McKinsey Quarterly*, (4):26–43, 2010.

[7] Xu Fei. Oreilly hadoop the definitive guide (06-2009). https://autofei.wordpress.com/2010/06/27/oreilly-hadoop-the-definitive-guide-06-2009/, Apr 2011.

[8] Lawrence Gates. Csse-senior-thesis. https://github.com/gateslm/CSSE-Senior-Thesis, 2017.

[9] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[10] Andrea De Mauro, Marco Greco, and Michele Grimaldi. A formal definition of big data based on its essential features. *Library Review*, 65(3):122–135, 2016.

[11] Brian L Smith and Michael J Demetsky. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering*, 123(4):261–266, 1997.

[12] Peter Struijs, Barteld Braaksma, and Piet JH Daas. Official statistics and big data. *Big Data & Society*, 1(1):2053951714538417, 2014.

[13] Tom E. White. Hadoop: The definitive guide (4th edition). chapter 1. O'Reilly Media, 2015.

[14] Tom E. White. Hadoop: The definitive guide (4th edition). chapter 2, page 44. O'Reilly Media, 2015.

[15] Tom E. White. Hadoop: The definitive guide (4th edition). chapter 2. O'Reilly Media, 2015.

[16] Tom E. White. Hadoop: The definitive guide (4th edition). chapter 16. O'Reilly Media, 2015.

[17] Tom E. White. Hadoop: The definitive guide (4th edition). chapter 17. O'Reilly Media, 2015.

A Data Description

A.1 Chicago Traffic Tracker - Congestion Estimates by Segments

Column Names in Spreadsheet

- **SEGMENT_ID**: Unique arbitrary number to represent each segment
- **STREET**: Street name of the traffic segment
- **DIRECTION**: Traffic flow direction for the segment
- **FROM_STREET**: Start street for the segment in the direction of traffic flow
- **TO_STREET**: End street for the segment in the direction of traffic flow
- **LENGTH**: Length of segment in miles
- **STREET_HEADING**: Direction of the STREET from the origin point
- **START_LATITUDE**, **START_LONGITUDE**, **END_LATITUDE**, **END_LONGITUDE**: These four points represent the start and end points of the segment in the direction of traffic flow
- **CURRENT_SPEED**: Real-time estimated speed in miles per hour
- **LAST_UPDATED**: Date and time of update to spreadsheet

A.2 Chicago Traffic Tracker - Congestion Estimates by Region

Column Names in Spreadsheet

- **REGION**: Name of the region made up of the names of the community areas within the region
- **REGION_ID**: Unique arbitrary number to represent each region
- **WEST**: Lowest longitude values on the regions boundary
- **EAST**: Highest longitude values on the regions boundary
- **NORTH**: Highest latitude value on the regions boundary
- **SOUTH**: Lowest latitude value on the regions boundary
- **DESCRIPTION**: Describes the streets that demark the regions boundary
- **CURRENT_SPEED**: Real-time estimated congested level
- **LAST_UPDATE**: Time stamp for the latest congestion estimation run.

A.3 Advertisement Data Themes

Themes available to be selected when manually generating advertisements:

Automotive	Clothing	Educational
Cellular Service	Technology	Medical
Job	Recruitment	Housing
Religious	Travel	Political
Theater	Restaurant	Grocery
Insurance	Mobile Devices	Toiletries
Lawyer	Movie	Television

A.4 Region Analysis Data

Figure A.4 shows a subset of the data collected for regions. The subset of data is for running the algorithms over all the data without separation such as by weeks, months, or years. To see the full set of data for regions, refer to Github repository with all the data separated by running the algorithm over weeks, months, or years.

NOTE: The values that have Ignore and negative numbers are place holders to denote where it was not relevant enough to place an advertisement. The different negative numbers are in place for debugging in case all output is a negative number for a given region.

	All Data Result		
Region ID	Theme	Theme Count	Rating
Ashburn	Insurance	48	9
Auburn Gresham-Chatham	Ignore	-1	-1
Austin	Lawyer	51	5
Beverly-Mt Greenwood-Morgan Park	Cellular Service	47	10
Bridgeport-McKinley-Lower West	Clothing	25	10
Chicago Loop	Travel	54	9
Downtown Lakefront	Political	16	9
Dunning-Portage-Belmont Cragn	Medical	47	9
Edge Water-Uptown	Technology	24	10
Far North West	Mobile Devices	53	10
Fuller-Grand Blvd-Washington Park	Travel	50	8
Hermosa-Logan Square	Mobile Devices	28	9
Humboldt-Garfield Prk E/W	Ignore	-1	-1
Hyde Park-Kenwood-Woodlawn	Mobile Devices	31	9
Irving Park-Avondale-North Ctr	Housing	45	10
Lawndale N/S	Religious	54	6
Lawndale N/S	Religious	54	6
Lincoln Park-Lake View	Mobile Devices	38	10
Midway-Garfield Rdg-Clearing	Political	24	9
Near North	Restaurant	25	4
Near South-Douglas	Job	26	10
Near South-Douglas	Job	26	10
New City-Englewood-W Englewood	Ignore	-1	-1
North Park-Albany-Linconl Sq	Movie	30	10
North Park-Albany-Linconl Sq	Movie	30	10
Riverdale-Hegewisch	Job	49	7
Riverdale-Hegewisch	Job	49	7
Riverdale-Hegewisch	Job	49	7
Rogers Park - West Ridge	Restaurant	57	9
South Deering-East Side	Automotive	37	8
South Shore-S Chicago-Avlon	Political	32	10
South West Side	Ignore	-1	-1
Washington Hts-Roseland-Pullman	Medical	20	9
West Town-Near West	Ignore	-1	-1

Figure 8: Subsection of Region data. The results were calculated using the described algorithms 6

A.5 Segment Analysis Data

Figure A.5 shows a subset of the data collected for segments. The subset of data is for running the algorithms over all the data without separation such as by weeks, months, or years. To see the full set of data for segments, refer to Github with all the data separated by running the algorithm over weeks, months, or years.

NOTE: The values that have Ignore and negative numbers are place holders to denote where it was not relevant enough to place an advertisement. The different negative numbers are in place for debugging in case all output is a negative number for a given segment.

	All Data Result		
Segment ID	Theme	Theme Count	Rating
1	Political	40	5
2	Clothing	25	9
2	Clothing	25	9
3	Technology	12	10
4	Movie	19	9
4	Movie	19	9
5	Ignore	-1	-1
6	Housing	21	8
7	Clothing	3	10
8	Mobile Devices	33	8
9	Job	16	10
10	Mobile Devices	46	8
11	Toiletries	52	7
12	Mobile Devices	14	9
13	Television	5	7
14	Housing	37	4
15	Ignore	-1	-1
16	Ignore	-1	-1
17	Ignore	-1	-1
19	Ignore	-1	-1
20	Clothing	3	9
21	Ignore	-1	-1
22	Ignore	-1	-1
23	Ignore	-1	-1
24	Toiletries	1	9
25	Movie	1	10

Figure 9: Subsection of Segment data. The results were calculated using the described algorithms 6