**FINAL REPORT**

****

**T-SA:** **Twitter Keyword Search API based Tweet Analysis**

(The 19th President Election)



Semester: 2019-1st Semester

Course: INC-CAPSTONE DESIGN 1

Professor: Jong HyunSook

Team Nam: VI

Leader: Lee SeokJune(20165072)

Member: Lee YunHyuck(20165062)

Bae InGyu(20165073)

Seo JaeIck(20144773)

**- CONTENTS -**

**Chapter 1. Introduction** -------------------------------------------------------------------- 01

**Section 1. Motives and Needs for Development** ---------------------------------------- 02

**Section 2. Existing Studies** --------------------------------------------------------------------- 02

**Section 3 Development Goals** ----------------------------------------------------------------- 03

**Section 4. System Specification** -------------------------------------------------------------- 04

**1.4.1. System Operating Process** ---------------------------------------------------------------- 04

**Chapter 2. The Main Issue** -------------------------------------------------------------- 05

**Section 1 Development Environment** ------------------------------------------------------- 06

**2.1.1. Ubuntu** ------------------------------------------------------------------------------------- 06

**2.1.2. Python** ------------------------------------------------------------------------------------- 06

**2.1.3. MariaDB** ------------------------------------------------------------------------------------ 07

**2.1.4. Eclipse, OpenJDK** ------------------------------------------------------------------------- 07

**2.1.5. Hadoop Echo System** --------------------------------------------------------------------- 08

**Section 2. Twitter API** -------------------------------------------------------------------------- 09

**Section 3. T-SA Flow Diagram** --------------------------------------------------------------- 12

**Section 4. Implementation (Python)** --------------------------------------------------------- 13

**2.4.1. T-SA.py** ------------------------------------------------------------------------------------ 13

**2.4.2. TwitterAPI.py** ------------------------------------------------------------------------------ 21

**2.4.3. DBModule.py** ------------------------------------------------------------------------------- 23

**2.4.4. Analysis\_Visual.py** ------------------------------------------------------------------------- 24

**Section 5. DB and Table Definition** -------------------------------------------------------- 28

**Section 6. Implementation (Hadoop)** -------------------------------------------------------- 31

**2.6.1. Spooq (Interworking between DB and HDFS)** ------------------------------------------ 31

**2.6.2. Mapper and Reducer (java file)** --------------------------------------------------------- 31

**2.6.3. Hadoop Execute (jar file)** ---------------------------------------------------------------- 33

**Chapter 3. Conclusion** ---------------------------------------------------------------------- 35

**Section 1. Execution Environment** ---------------------------------------------------------- 36

**Section 2. Tweeter API Parameter Settings Value** ------------------------------------- 36

**Section 3. Analysis and Visualization** ------------------------------------------------------- 36

**Section 4. Final conclusion** -------------------------------------------------------------------- 38

**Section 5. What we felt** ----------------------------------------------------------------------- 38

**Chapter 4. Appendix** ------------------------------------------------------------------------- 40

**Appendix 1. MariaDB related settings** ------------------------------------------------------ 41

**Appendix 2. Table Generated Query Statement** ---------------------------------------- 42

**Appendix 3. References and Sites** ---------------------------------------------------------- 43

Appendix **4. Github** ----------------------------------------------------------------------------- 43

**Chapter 1. Introduction**

**Section 1. Motives and Needs for Development**

Twitter is a social network service (SNS) that allows people to express their opinions (thoughts) on tweets. Now, the use rate is much lower than Facebook and Instagram, but the social impact cannot be ignored because of the convenience of writing tweets and the rapid spread of information. Anonymity is stronger than other SNS because account creation requires only mail addresses, and is formatted through a mobile phone authentication process, but is private to other users. As a result, many tweets express their opinions (thoughts) about social phenomena as they are often expressed with confidence. Twitter's posting, Twitter's tweet, can write 140 characters at a time, including spaces and symbols, whether in Korean or English. Because of these features, it takes less time to see tweets that express opinions (thoughts) of other users, and Twitter searches for content on topics that become real-time issues provide faster and more diverse results than other SNSs (facesbook, Instagram).

Obama's camp had drawn global attention in the 2008 and 2012 U.S. presidential elections. Obama's campaign announced his candidacy and opened it to social networking sites (SNS) and used it to rally key supporters and mobilize them as political allies. So Barack Obama earned the nickname "social media president" because he showed positive results in the 2008 and 2012 presidential elections by demonstrating and utilizing an excellent way of running social media. President Barack Obama also said during his campaign that "social media is a platform for candidates to talk about without the filters of journalism." Since then, Hillary Clinton (Democratic Party) and Donald Trump (Republican Party) have run in the 2016 U.S. presidential election. At this time, Donald Trump had 11.3 million Twitter followers and 10.56 million Facebook followers, far surpassing former President Barack Obama, who had 10 million Twitter followers. Through this large number of SNS followers, Donald Trump has been the talk of the town throughout the Republican nomination race, pumping messages related to various political issues onto social networks. In particular, he shocked the U.S. political community by openly criticizing his opponent (Hillary Clinton) and making extreme remarks on sensitive issues such as illegal immigration and racism rather than promoting his pledges or showing achievements. However, supporters on SNS were enthusiastic about Donald Trump's extreme remarks, and as a result, he won the 2016 presidential election.

The nation actively used the Internet in the 2002 presidential election, showing how social media could be used in elections in earnest in the 2011 Seoul mayoral by-election. In particular, a deep analysis of the role of social media emerged in many aspects, including the spread of each candidate's network and message, and the encouragement of participation in voting on social media.

The use of SNS in the previous presidential election (2008, 2012 and 2016) in the U.S. and by-elections (2011) in Seoul may have implications for Korea's presidential election. So, in this project, we will analyze the relationship between the two by choosing Twitter, which is written in short letters, and the 19th presidential election during the nation's presidential election, as was written on top of the SNS.

**Section 2. Existing Studies**

Twitter Trend (http://tweetrend.com), a tweet analysis site provided by Wisenut, collects 10 million tweets a day from data since January 2011 for about 4 million Twitter accounts, providing the nation's largest search service. As a representative service, it provides various analysis results such as retweets, URL parsing, and ripple forces. A typical example is a service that allows users to download and view the original text so that it can be used in various fields such as marketing. There are three ways to use it: a Non-login, a free member, and a paid member. The number of search keywords available for each method and the duration of the search are as shown in [Table 1].

**[Table 1]** Search rights by Tweets Trends Usage

|  |  |  |  |
| --- | --- | --- | --- |
|  | Non-login | Free Member | Paid Member |
| Number of Search Keywords | 1 | 3 | 3 |
| Search Period | 7 days | 30 days | Unlimited |

Follow Me, a site created by Konstantin Kovshenin using Twitter API, analyzes and shows information such as topics, references and hashtags regarding public profiles. As a representative service, it collects the profile of the requested user and the latest tweets written by the user, analyzes the content and outputs and displays frequently used words (hashtags).

The Election Statistics System site of the National Election Commission provides a chronological review of past election week schedules, a quick glance of candidates for elections, and detailed information of candidates for each election. In addition, the voting rate for each candidate can be checked after the voting deadline on the day of the election. Statistical information provided by the election statistics system site is available to anyone for free and, if provided with other information, information can be used in the castle to clearly state the source of each data to be downloaded, preventing users from being misunderstood. So, to bring the results of the 19th presidential election that we will use in this project, we can bring in data through counting in previous elections, which will be downloaded as PDF or EXCEL files.

**Section 3. Development Goals**

Companies and institutions are trying to collect and analyze data through SNS such as Facebook, Twitter and Kakao Talk, as well as internal data, to achieve various results that have not been understood before. One such reason is Twitter, which is one of SNS's. Twitter is because more than 100 million users around the world write more than 200 million tweets a day on average. In particular, 1.74 million tweets were analyzed during the 2012 presidential election to track the flow of public sentiment, and in 2014 the social network of information exchange was analyzed by analyzing tweets related to specific IT fairs. In this project, the keywords related to the 19th presidential election (the top five candidates for the 19th presidential election) and the period (2017.04.18.-2017.05.09) are set to be taken to obtain tweets written during that period, to obtain the frequency of the included words and hashtags, and to analyze how they relate to the results obtained through the election statistics system of the Central Election Commission.

The first uses Twitter APIs provided by Twitter to collect tweets containing specific keywords during the presidential election. The results of sending keyword queries through Twitter APIs and responding to queries provide a JSON format. To use it, developers must register, not their ID and password, and those who registered as developers must be issued a Consumer Key. A free version of Standard is available when collecting tweeting data. At this time, the number of tweets available for import is limited to 100, and the number of queries available during the month is limited to 50. If you want to collect more data, you can make payments and upgrade to a version other than Standard.

Second, to prevent unnecessary information collection, keywords related to the 19th presidential election will be retrieved through tweets from the election period of April 15, 2017 to May 9, 2017. This will allow the collection of tweets from more diverse people and increase the accuracy of presidential elections and tweets. It will also be removed and collected if there is a duplicate of the tweet. Those tweets use Korean because they were made during the 19th presidential election in South Korea. Tweets written in Korean are frequently omitted from the subject, narrative and object language, and some users may not be able to post them on purpose or do so many times. Therefore, we want to obtain the frequency of keywords by classifying them as proper and common nouns through natural language processing, and to provide various visualizations, including frequency of upper keywords, through word cloud tools.

Finally, the National Election Commission (NEC) plans to analyze and express the results more objectively using visualization tools based on the voter turnout of the 19th presidential election candidates provided by the election statistics system and the number of comments made by each candidate in the collected tweets.

**Section 4. System Specification**

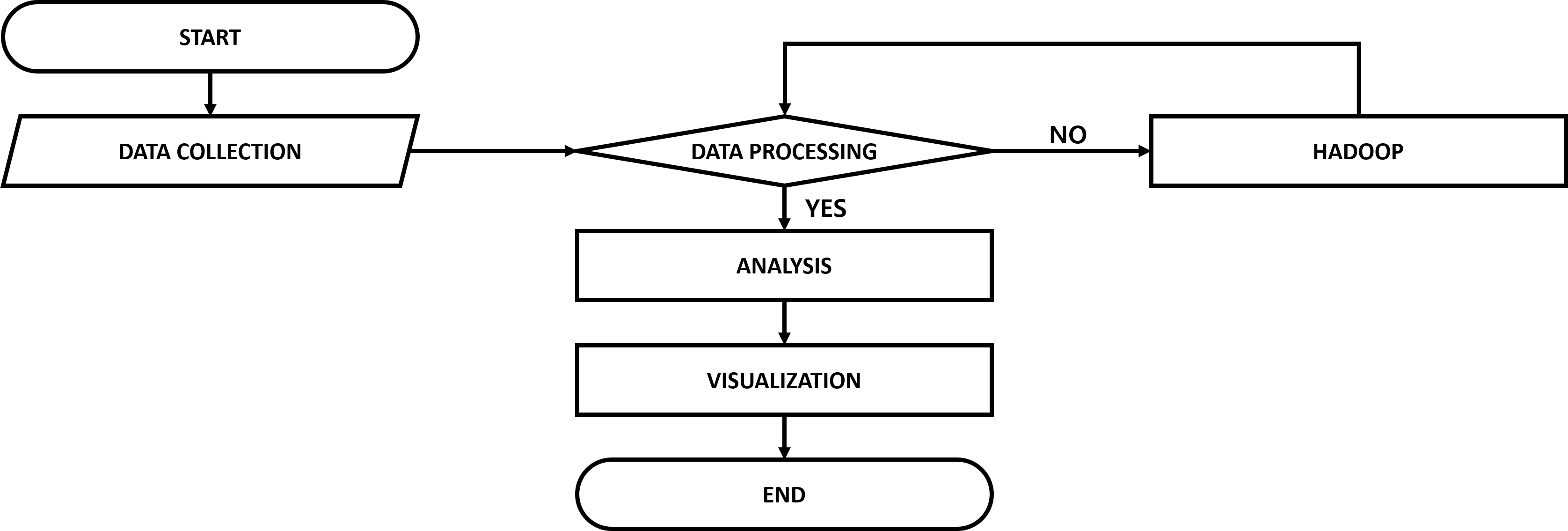
**1.4.1. System Operating Process**

See [Figure 1] for a schematic diagram of the T-SA system operation as follows.

1. Collect data (formality - election statistics system data from the National Election Commission, atypical - tweeting information from Twitter).

2. Determine whether the data collected for analysis and visualization need to be processed. If it is necessary to process, process the data using Hadoop.

3. Compare and analyze data and visualize it.



**[Figure 1]** T-SA System Operational Course Plot

**Chapter 2. The Main Issue**

**Section 1. Development Environment**

T-SA's development environment (version) is as shown in [Table 2].

**[Table 2]** T-SA Development Environment

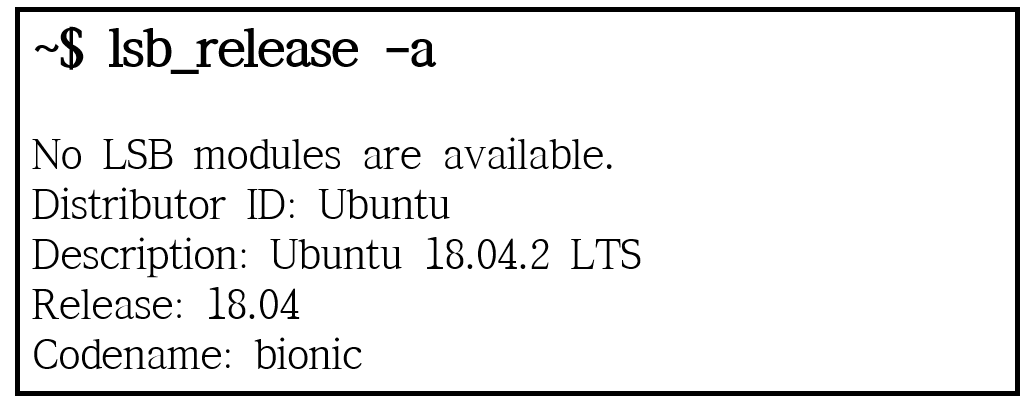
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ubuntu | | | 18.04.2 LTS | |
| Python | | | 3.6 | |
| MariaDB | | | 10.1.38 | |
| Eclipse | | | 2019-03 (4.11) | |
| OpenJDK | | | 1.8.0\_191 | |
| Hadoop Echo System | Hadoop | 3.2.0 | |
| Sqoop | 1.4.7 | |

**2.1.1. Ubuntu**

Ubuntu is one of the operating systems that allow users to use programs and peripheral devices on their computers, which are distributed free of charge and available for free with support from Canonical Inc. The name Ubuntu means 'I am because you are' and 'I am' means a love of humanity towards others. Vantour is used in central Africa, including Zimbabwe and Rwanda, and Ubuntu practices the philosophy contained in its name as an OS. Countless users around the world are helping to improve Ubuntu's capabilities or translate it into languages. Ubuntu users are growing rapidly through the community, which is a community under the consensus of using the same OS and a community of voluntary individuals without coercion.

There are close to 400 distributions of Devian, as there are types of Microsoft Windows such as XP and Vista. Distribution versions are called 'distribution' or 'distro' in English, and Ubuntu is one of these. Although each distribution has different performance, strengths and weaknesses, all of the key parts that are called "Kernel" have something in common: Linux. With Ubuntu installed, all of the work-related programs, computer management programs and simple games are installed on the computer, and are easy to use operating systems because they are organized at the level of a beginner's eye. Ubuntu offers free programs that improve on a time basis through the efforts of several people across a wide range of areas. Canonical said it has no plans to pay and will not be. Therefore, the use of Ubuntu can avoid using illegal programs that are illegal.

The methods and results of checking the version of Ubuntu are as shown in [Figure 2] below.



**[Figure 2]** How to check the version of Ubuntu and its results

**2.1.2. Python**

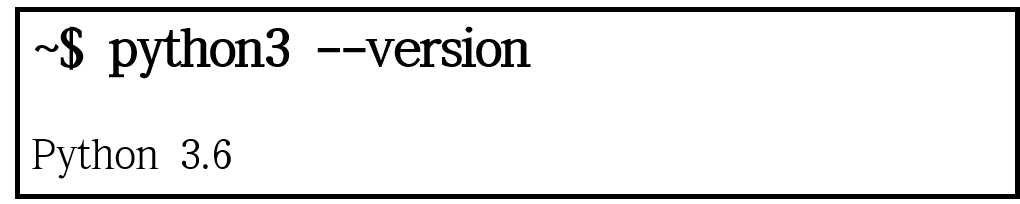
Python is an advanced programming language published in 1991 by programmer Guido van Rossum, a platform-independent, object-oriented, dynamic typing interactive language. Python has an open, community-based development model managed by the nonprofit Python Software Foundation.

The final version of Python's 3.0, code-named Python3000 (or Python3k), was announced on December 3, 2008 after a long test. 2.The biggest feature is that there is no Python and no subcompatibility in the X version. Many of the main features of Python 3 were also reflected in versions 2.6 and 2.7, compatible with previous versions.

The Python official document summarizes, "The Python 2.X version is a legacy (old technology), and the Python 3.X version will be Python's present and future," with first-time programmers recommending starting with Python 3.

Differences from Python 2.X version include internal changes in embedded data types, such as dictionary and string types, and some spherical components have been removed, standard libraries are relocated, and enhanced Unicode is supported. Therefore, it supports Hangul variables.

Python3 can be installed simply by typing sudo apt-get install python3 in the terminal, and the version verification method and results are shown in [Figure 3].



**[Figure 3]** How to check the version of Python3 and its results

**2.1.3. MariaDB**

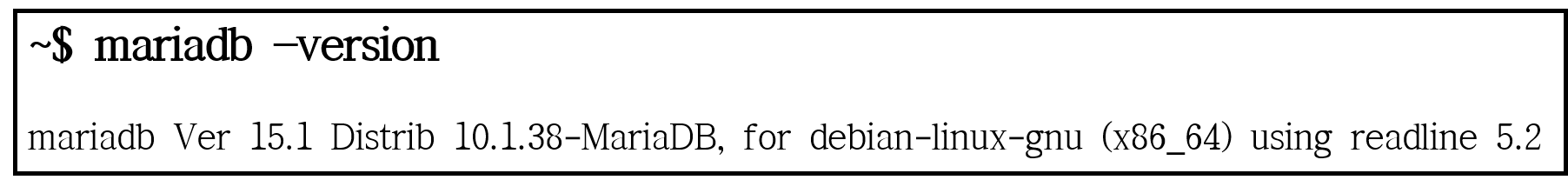
MariaDB is an advanced form of substitute for MySQL, available for download on 다운로드 and maintained as a GPL v2 license, and is being developed as a mainstay of the MariaDB community and the MariaDB Foundation.

Security is very important in today's world and is being watched by MariaDB developers. The project maintains its own security patch based on the MySQL project. For each MariaDB release, the developer can Merge MySQL security patches and improve them if necessary. When important security issues are discovered, developers immediately develop and distribute a new MariaDB release that addresses them. Many security issues found in MySQL have also been found in MariaDB and have been reported to the MariaDB team. The MariaDB team is working closely with http://cve.mitre.org/ to ensure that all security issues are immediately reported and resolved in sufficient detail. Detailed security issues are typically released after the issue is resolved versions of MariaDB and MySQL are released.

MariaDB is being released from the latest MYSQL to date and, in many cases, works just like MySQL. All commands, interfaces, libraries and APIs in MySQL also exist in MariaDB. It also eliminates the need to convert databases into MariaDB. In other words, MariaDB is actually the perfect substitute for MySQL.

MariaDB can be installed simply by typing sudo apt-get install mariadb-server in the terminal. This will install 10.1.38 version of MariaDB and a dependency package on the basis of the installation date (2019.03.30). After that, enter sudo mysql\_secure\_installation and set up MariaDB's authority table, as shown in [Figure 4-1] below. And the version verification methods and results are shown in [Figure 4-2].

**[Figure 4-1]** The process of setting up the Kwon Eibble by MariaDB



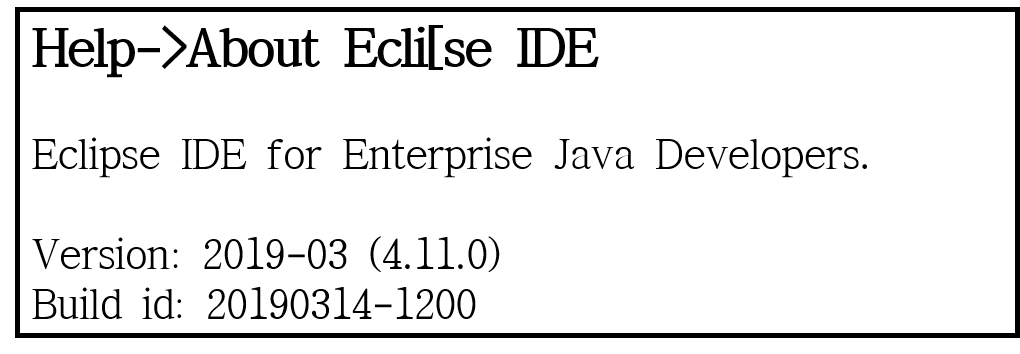
**[Figure 4-2]** How to check MariaDB's version and results

**2.1.4. Eclipse, OpenJDK**

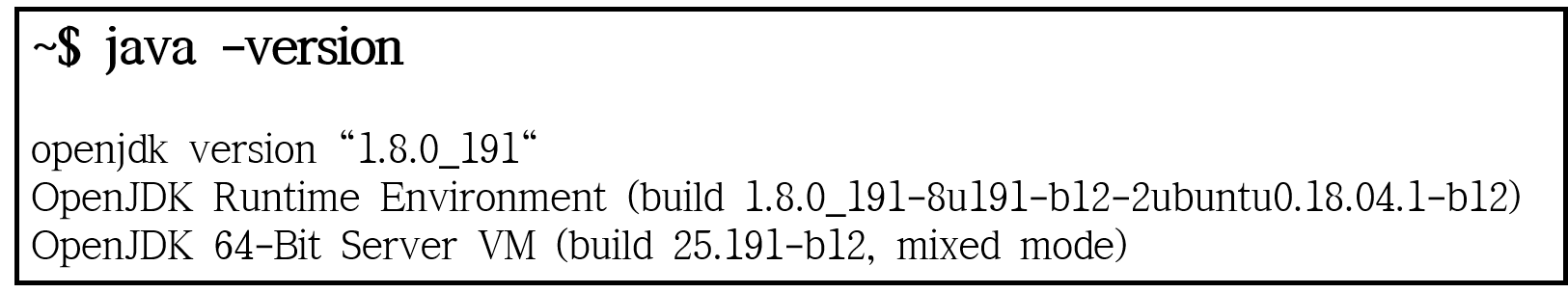
Eclipse (Eclipse) can be written on a variety of platforms and was started for the purpose of an integrated programming development environment that supports Java and other languages, but it has now evolved into a general-purpose application software platform by introducing Open Service Gateway Initiative (OSGi). Originally developed under the name of IBM's WebSpare Studio Application Developer, it has developed into what is now Eclipse based on the open-source release of the engine part.

The open Java development kit is an open source JDK based on Java SE (Standard Edition). In 2006, Sun Micro System announced that it would open source Java. And in November of that year, the company solved the HotSpot VM and compiler with the GNU General Public License (GPL). Currently, leading IT companies are involved in the project, and IBM decided in October 2010 to participate in the OpenJDK project from the Apache Harmony project that it previously participated in. Soon after, Apple agreed to cooperate with OpenJDK by the end of 2010 and SAP by the middle of 2011.

For Eclipse, access www.eclipse.org to install Linux during Eclipse IDE 2019-03, which checks its environment and installs either x86, x64 and the versioning method and results are shown in [Figure 5]. For open Java development kits, typing sudo apt-get install openjdk-8-jdk in the terminal is a simple installation and the version verification methods and results are shown in [Figure 6].



**[Figure 5]** How to check the version of Eclipse and its results



**[Figure 6]** How to check the version of the Public Java Development Kit and its results

**2.1.5. Hadoop Echo System**

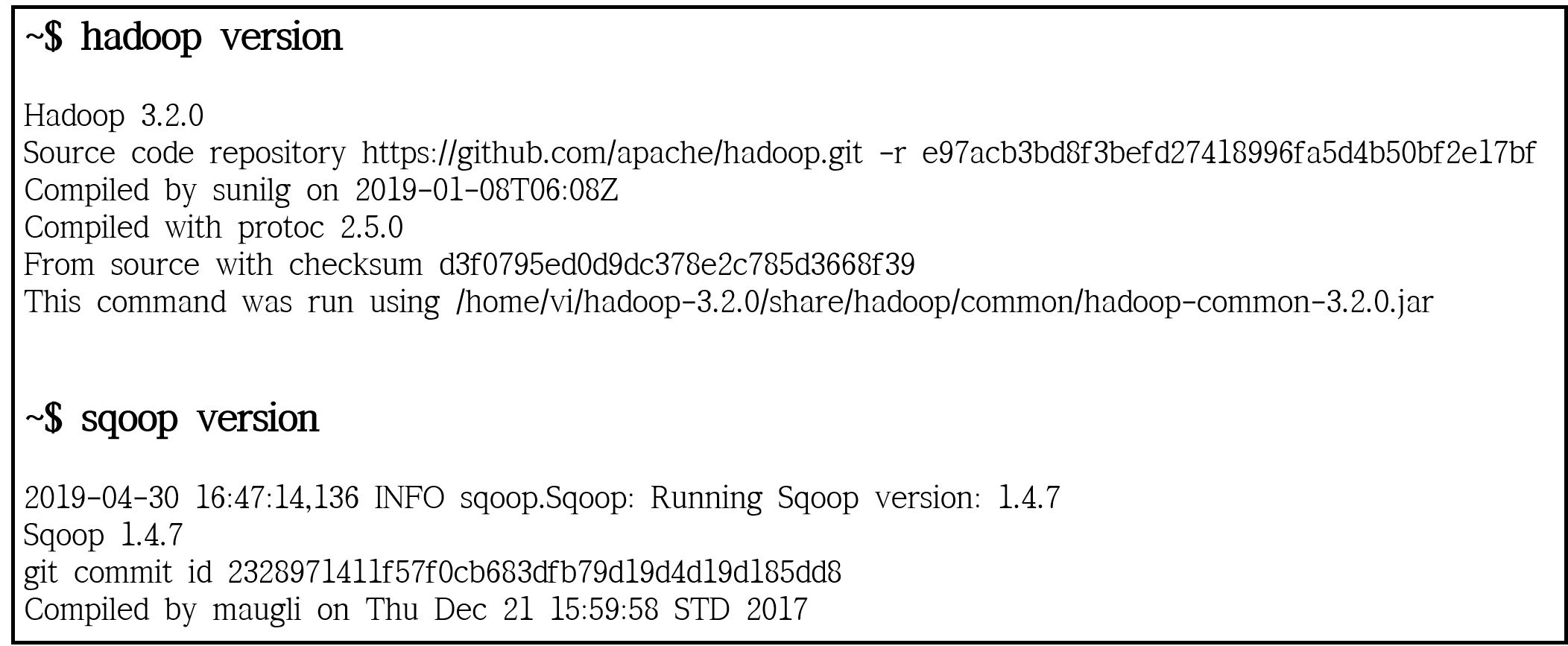
Apache Hadoop (High-Availability Distributed Object-Oriented Platform) is a freeware Java software framework that supports distributed applications that operate on large clusters of computers capable of processing large amounts of data. Originally developed to support distributed processing of the nuts, it is a sub-project of Apache Lushin. It implements Hadoop Distributed File System (HDFS) and MapReduce that can replace Google File System, which is a distributed processing system.

The Hadoop distributed file system (HDFS) is a distributed expansion file system written in Java language for the Hadoop framework. HDFS distributes large files across multiple machines and stores them. Data reliability is achieved by storing data on multiple servers in duplicate.

Map/Reduce is a software framework that Google announced in 2004 for the purpose of handling large data processing in distributed parallel computing. This framework was developed to support parallel processing in a cluster environment consisting of less reliable computers with large amounts of petabytes or more. This framework consists mainly of a Map commonly used in function programming and a function base called Reduce. Map/Reduce is now written to be applicable in Java, C++, and other languages. Typically applied as open source software in Apache Hadoop.

Sqoop is a Command-Line Interface (CLI) application that efficiently transforms large volumes of data between structured relational databases and Apache Hadoop. You can import data from relational databases such as Oracle or MySQL into a Hadoop distributed file system, convert it to Hadoop MapReduce, and export the transformed data back to a relational database. Spoon processes the import and export of data through a mapReduce, enabling parallel processing as well as fault tolerance.

The method and results of checking the version of Hadoop and Spoon are shown in [Figure 7].



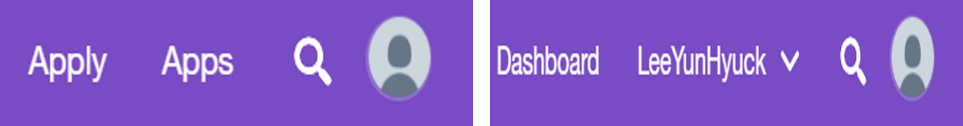
**[Figure 7]** How to check the version of Hadoop and Spoon and Results

**Section 2. Twitter API**

In order to share information as widely as possible, Twitter is helping companies, developers and users access Twitter data programmatically through the Twitter Application Programming Interface (API). To use the Twitter API, it basically proceeds with a Twitter subscription. Developer registration is possible only after sign - up is completed. If developers are not registered, keys are not issued. Once the key is issued, it is freely available according to the area used through the setting of the development environment. The corresponding process is carried out in the following order.

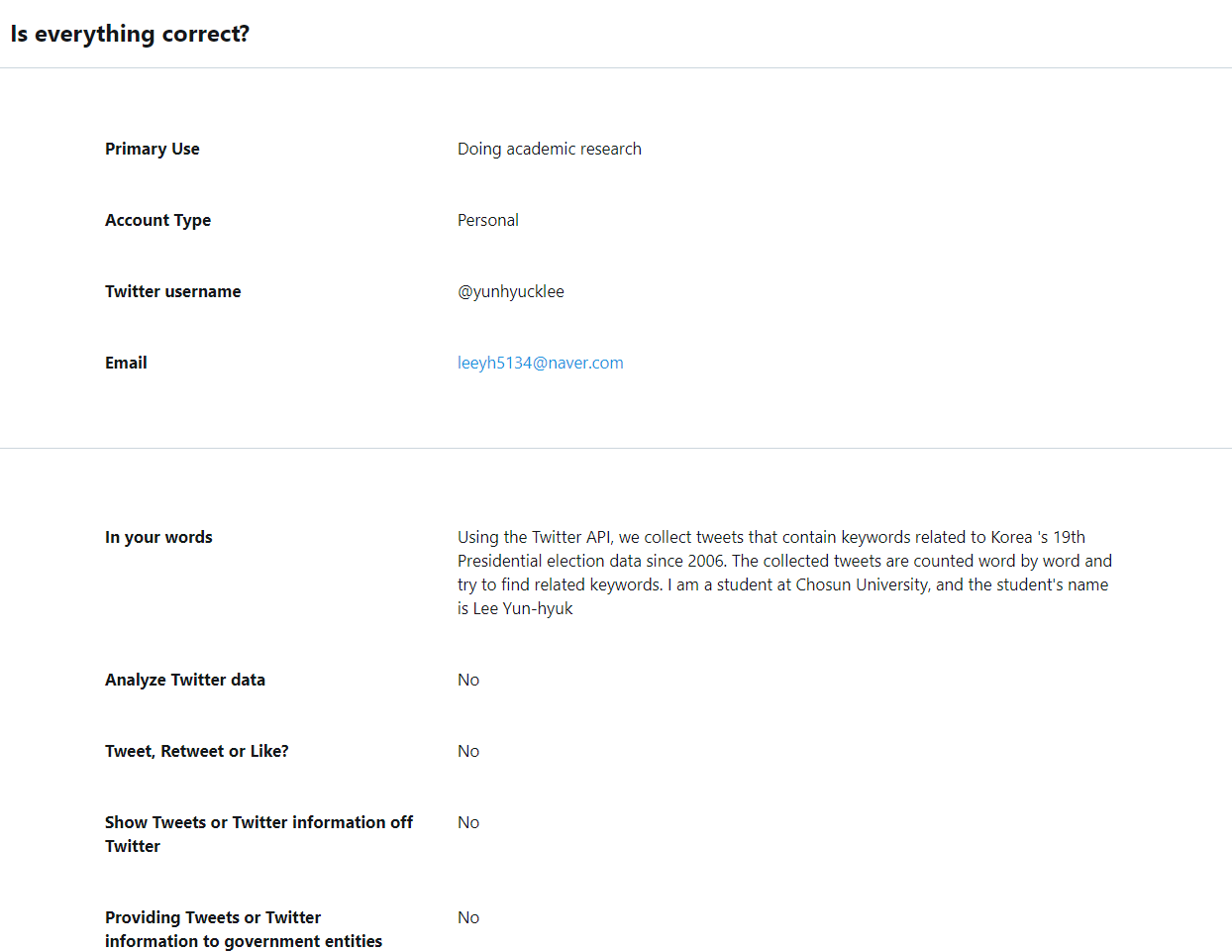
**1. Register developer**

Developer registration can be registered by accessing https://developer.twitter.com/ First press Apply on the top right. The top right of the web page before and after registering as a developer is shown in [Figure 8].



**[Figure 8]** Before (left), after (right)

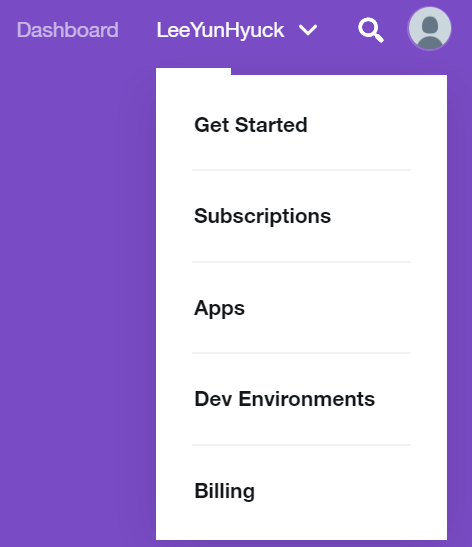
When you click Apply for a developer account, the question of why you use Twitter's development tool is now added. Questions about this were designated and carried out for academic use reasons. Selecting a question activates a page asking if you want to register as a developer based on your currently logged in account. Basically, if you don't have cell phone authentication and e-mail authentication on Twitter, you can add a nickname and add it to the area where you live without any settings and move on to the next step. Then a page appears asking for the purpose of use. All responses had to be filled before the page was changed, but after the change, it was simply written for the purpose of use. The purpose of use is to include more than 200 characters in English and the school name for students. When registering Twitter developers, the project uses 'Twitter APIs' for use purposes to collect tweets that include keywords related to South Korea's 19th presidential election among data since 2006. The tweets collected try to count by word to find the associated keywords. She is a student at Chosun University, and her name is xxx.' When the creation is completed and the next step is taken, a page will appear as shown in [Figure 9] to confirm that the request is correct. Finally, after the e-mail authentication process, it can be used on Twitter through developer approval.



**[Figure 9]** Information materials for registering developers have been completed.

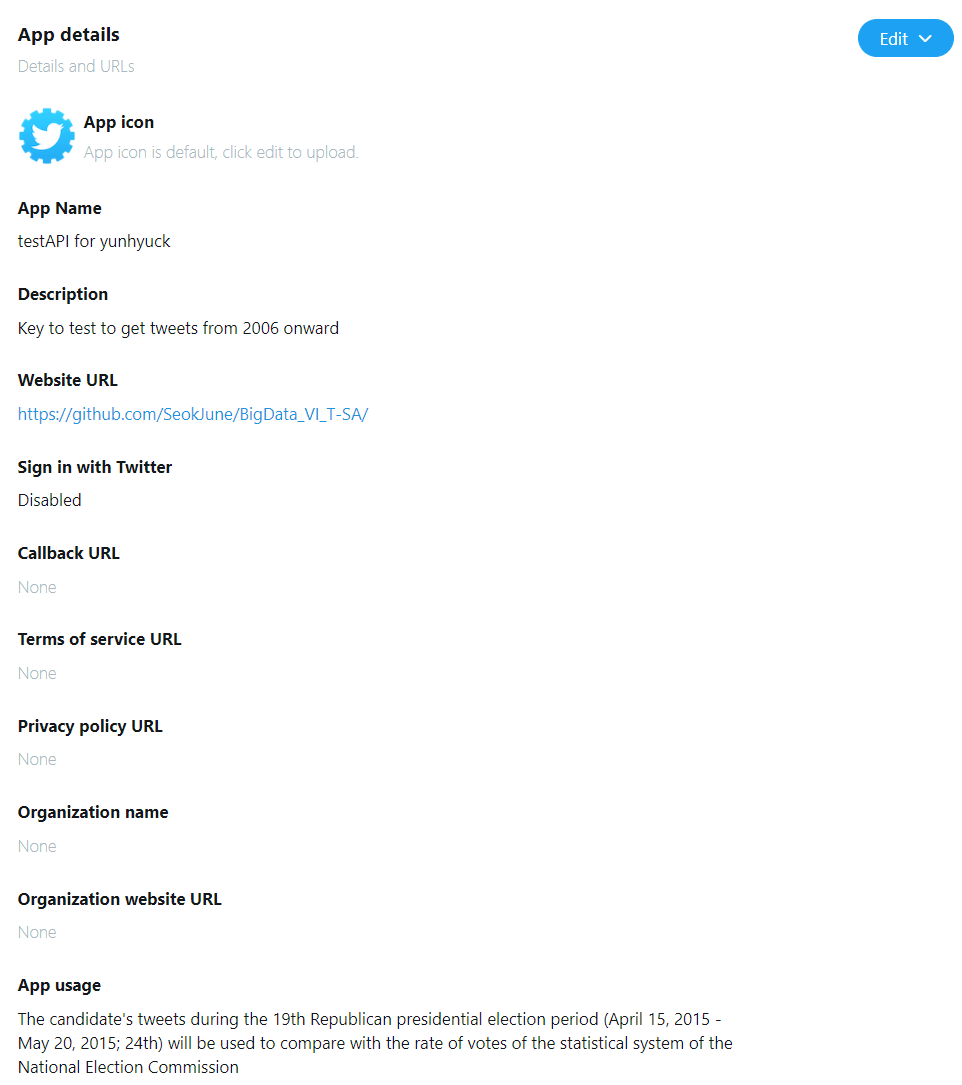
**2. Registering applications and issuing keys**

Applications are registered when developer registration is completed. This course will describe the name of the app to be used and how to use it. To register an application, you create an app by clicking Apps as shown in [Figure 10].



**[Figure 10]** After registering developers, menu available.

The application name used in this project was called Test API Issue. After the test, we wanted to get a new key and solve the limited usage. Then, write an explanation about the app. At this time, the number of letters can be between 10 and 200 characters in English. For this project, a 'Key to be tested to get tweets from 2006 onwards' is written. The Web URL was created using the hub(https://github.com/SeokJune/BigData\_VI\_T-SA/) address where the project was being carried out. Finally, how to use the app should be written with more than 100 characters. In this project, the Tweets of candidates were taken from April 15, 2017, during the 19th presidential election of the Republic of Korea, to May 9, 2017, which is used for a comparative analysis of the percentage of votes cast by the National Election Commission's statistical system. The final draft is shown in [Figure 11]. You can check the keys on the details page of the generated app.



**[Figure 11]** Information of the app to receive generated test APIs

**3. Classification and selection of search APIs**

Before setting up the development environment, we learned about the Search API for Twitter APIs' tweet search. The current version that provides by default is the Standard version. Premium and Enterprise can be divided into three categories, and more can be found in [Table 3].

**[Table 3]** Classification of Twitter's Search APIs

|  |  |  |
| --- | --- | --- |
| Standard | | It offers the latest tweet posted over the past seven days.  It is part of a set of public APIs. |
| premium | 30-days | It provides tweets posted over the past 30 days. |
| Full-archive | It will provide tweets posted from 2006. |
| Enterprise | | It is mainly used by companies to provide two things, such as Premium. |

When the Standard is queried, the length of time it can be imported is limited to seven days. Moon Jae-in, keyword is inside of it gets questions on Twitter that the Twitter messages, tweets this, because only posted in the past seven days.The 19th president to take place in connection with the project confirmed not appropriate to bring about the tweet election time. Premium is divided into versions that provide tweets posted over the past 30 days and those that provide tweets posted since 2006. There are differences in price, but they are basically priced through the number of queries that can be made in a month. [Figure 12] shows the differences in versions from 2006 onwards.



**[Figure 12]** Differences between Sandbox (Standard) and Premium

The configurations of Search APIs that import data from 2006 onwards are divided into Sandbox (Standard) and Premium versions. What the two versions have in common is that they can import data after the Twitter is launched. The biggest difference is that the built-in Sandbox version can only bring in 128 tweets. If the incoming tweet is more than 128 characters...This is indicated by , and 100 queries can be performed in a month. In contrast, the Premium version can basically carry out 500 queries and import 1024 characters of tweets. So we will use Premium for this project.

**4. Setting the Development Environment**

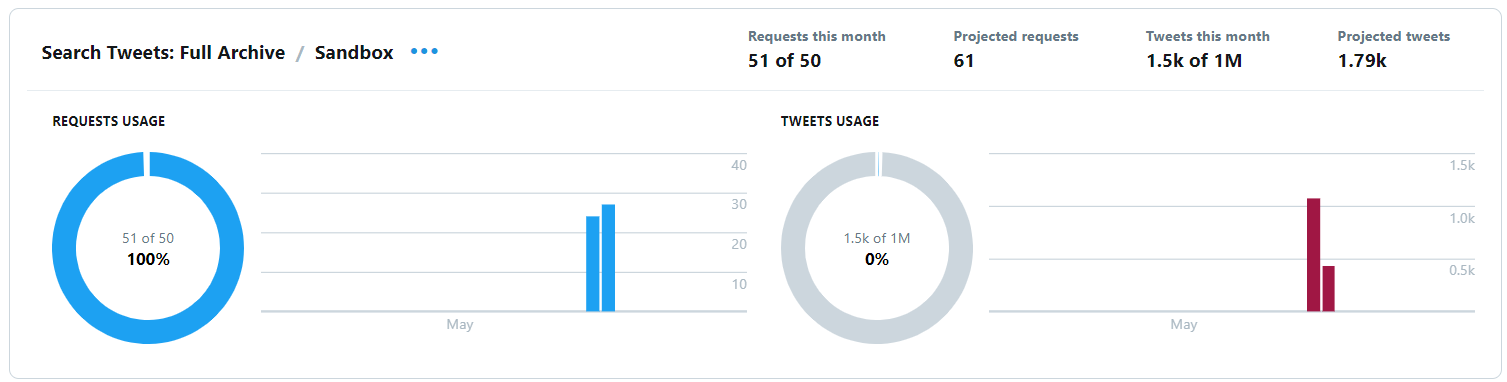
This is the process of setting up the environment in which API is used for issued keys. Only when the development environment is established can the results of a query be made through OAuth certification. In the case of Facebook, the key and token are sent as parameters through URL to perform the function of importing data. However, for Twitter, it is important that you bring your desired data with you in good faith to the basics of OAuth (the means to give access to websites or applications on other websites without providing Internet users' passwords). You can set up your environment through Dev Environments categories, as shown in [Figure 10]. Accessing that category is shown in [Figure 13].



**[Figure 13]** Dev Environments screen

**5. Check usage**

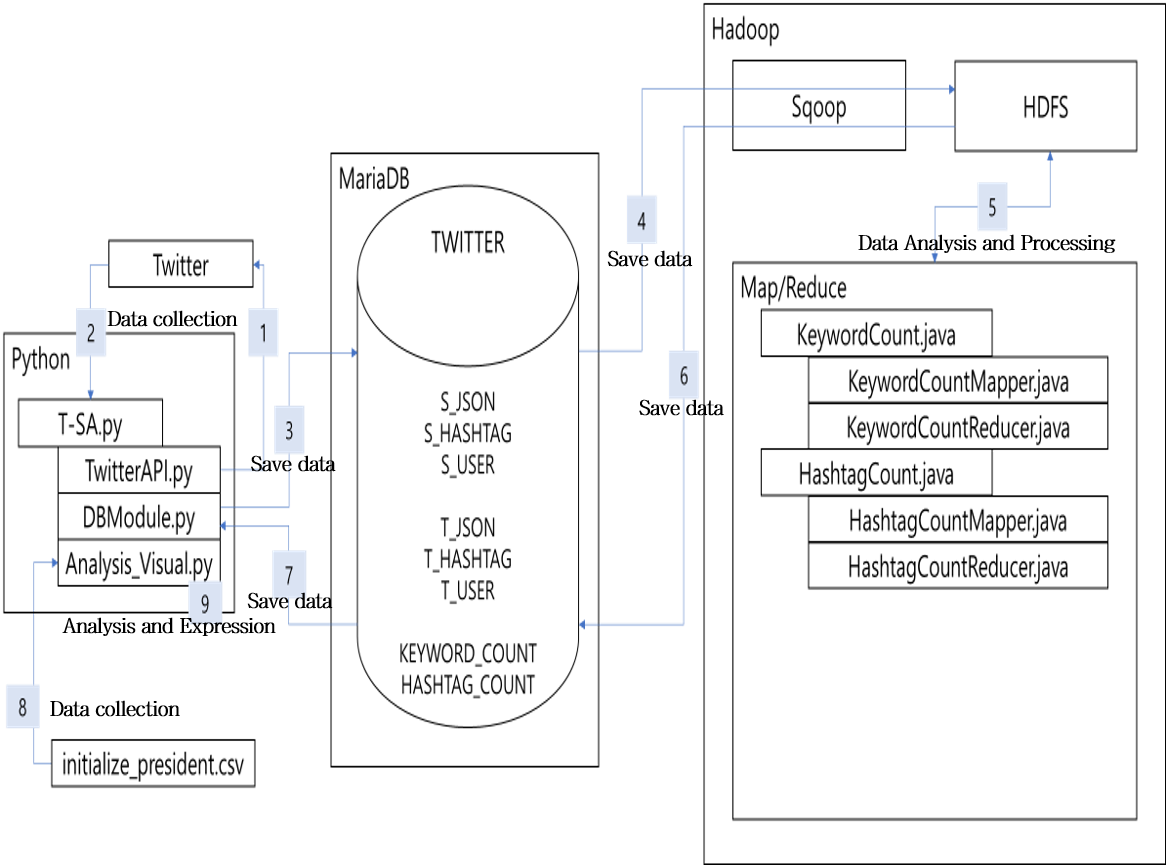
This project has set up the development environment of the Full Archive as data for the 19th presidential election have to be imported. Click Set up Dev Environment to select the label of the file to receive in JSON format and the name of the app that has the key to use. Once setup is complete, the usage can be checked through the Dashboard as shown in [Figure 14]. SandBox version should be checked periodically because the number of queries a month is limited to 50.



**[그림 14]** DashBoard 화면

**Section 3. T-SA Flow Diagram**

The flowchart for T-SA is shown in [Figure 15] and the details are as follows.



**[Figure 15]** T-SA Flow Diagram

**1, 2. Collect data (Twitter)**

Using Twitter API, collect information (Tweets (time to write, tweet, hashtag, etc.)) and user information (IDs, nicknames, location information, follow count, number of followers, language, etc.).

**3. Saving Data (MariaDB)**

Store the collected data in MariaDB.

**4. Save Data (HDFS: Hadoop Distributed File System)**

Store MariaDB data in the Hadoop distributed file system using a swipe.

**5. Processing data (Map/Reduce)**

Normalize the data stored in the Hadoop distributed file system through a map/reduce and save the results back to the Hadoop distributed file system. KeywordCount counts words (2 to 5 characters) after natural language processing. HashtagCount counts hashtags (less than 5 characters).

**6. Saving Data (MariaDB)**

Using a swipe, regularization data stored in the Hadoop distributed file system is stored in MariaDB.

**7. Collecting processed data**

Data stored in MariaDB is read by Python.

**8. Collect data (Central Election Commission data)**

It collects the number of votes from the 19th presidential election provided by the National Election Commission.

**9. Analysis and Expression**

Analyze and visualize data in 7 and 8 (word cloud, line graph, bar graph, circle graph, etc.).

**Section 4. Implementation (Python)**

**2.4.1. T-SA.py**

|  |
| --- |
| # T-SA.py  # Title: Main for T-SA  # Author: Lee SeokJune  # --------------------------------------------------------------------------------------------------  # import module  # --------------------------------------------------------------------------------------------------  # Class that implement functions related to Twitter API  import TwitterAPI  # Class that implement functions related to DataBase  ## pip3 install pymysql  import DBModule  # Class that implement functions related to Visualization  ## pip3 install pandas  ## pip3 install wordcloud  ## pip3 install matplotlib  ## pip3 install pyecharts  ## sudo apt-get install fonts-nanum\*  import Analysis\_Visual  # OS related Module  import os  # Time related Module  from datetime import datetime, timedelta  # --------------------------------------------------------------------------------------------------  # Set Parameter(Twitter)  # --------------------------------------------------------------------------------------------------  # Setting Parameters realted to KEY: consumer\_key, consumer\_secret, label  paramAPI = (('lNZwPI2dQ5l89K1nOGW6Sod6u', 'D6eGld20D99yrL89SMYPhJsjiHqmNKGL5LznkNKOQQPoIoQxWA', 'TSA0'),  ('MGRK5IsX8xwxhz0FYv5Llm5ps', 'JRh3fHqPqEq6VWcyoKax6MG4nE21z0zatiDjEGnvmHm99cyrLA', 'TSA1'),  ('NJhOCzxZPiPFckKJGUdSXRwz2', 'Cki8kGwebzFdEPHXOmZbX1lIdsMAgkx7Wauqey2Ll2Gwou5eCA', 'TSA')) # don't touch  # Setting Parameters related to SearchAPI: query, date(from, to), maxResults  ## Query length: Sandbox - 128 characters, Premium - 1024 characters  query = '"문재인" OR "홍준표" OR "안철수" OR "유승민" OR "심상정"'  ## date(from, to): Sandbox - Full history, Premium - Full history  date = ('201704180000', '201705100000') # 2017.04.17 자정 ~ 2017.05.09  ## Tweets per request: Sandbox - 10~100, Premium - 10~500  maxResults = 100  # Setting Parameters related to Timeline: code(user\_id, screen\_name), count  ## code(user\_id - , screen\_name - Name starting with '@')  code = ('Lee SeokJune', 'LSeokjune')  ## Specifies the number of Tweets to try and retrieve: ~200  count = 100  ## false - Timeline will strip any native retweets  include\_rts = False  # --------------------------------------------------------------------------------------------------  # Set Parameter(DB)  # --------------------------------------------------------------------------------------------------  # Setting Parameters realted to DB Connect: HostIP, UserID, Password, DB Name, encoding Character Type  paramDB = ('localhost', 'T-SA', '1234', 'TWITTER', 'utf8')  # Table list  tableName = ('S\_JSON', 'S\_HASHTAG', 'S\_USER', 'T\_JSON', 'T\_HASHTAG', 'T\_USER', 'KEYWORD\_COUNT', 'HASHTAG\_COUNT')  # --------------------------------------------------------------------------------------------------  # Result Parameter(Twitter, DB)  # --------------------------------------------------------------------------------------------------  sJson = []  sHashtag = []  sUser = []  tJson = []  tHashtag = []  tUser = []  keywordCount = []  hashtagCount = []  # --------------------------------------------------------------------------------------------------  # Set Parameter(Hadoop)  # --------------------------------------------------------------------------------------------------  paramSqoop = ['TWITTER', # DB Name  'T-SA', # User Name  '1234', # Password  ['S\_JSON', 'KEYWORD\_COUNT', 'S\_JSON', 'HASHTAG\_COUNT'], # Table List  ['TEXT', 'KEYWORD,COUNT', 'TEXT', 'HASHTAG,COUNT'], # Columns  [['target-dir', 'KEYWORD\_INPUT'], ['export-dir', 'KEYWORD\_OUTPUT'], ['target-dir', 'HASHTAG\_INTPUT'], ['export-dir', 'HASHTAG\_OUTPUT']], # [target-dir/export-dir, Path]  [['m', '1'], ['input-fields-terminated-by','"\t"']]]  # --------------------------------------------------------------------------------------------------  # Creating 'TwitterAPI', 'dbModule', 'Analysis\_Visual' object  # --------------------------------------------------------------------------------------------------  # 'TwitterAPI' object  twitter = TwitterAPI.TwitterAPI('https://api.twitter.com/')  # 'DBModule' object  db = DBModule.DBModule(paramDB[0], paramDB[1], paramDB[2], paramDB[3], paramDB[4])  # 'Bisualization' object  visual = Analysis\_Visual.Analysis\_Visual(date, query)  # --------------------------------------------------------------------------------------------------  # Operating Part  # --------------------------------------------------------------------------------------------------  # Output Title list  Title = (('Twitter',  'Search', 'Timeline'), # LSJ, LSJ  ('DataBase',  'Select(JSON - S, T)', 'Select(COUNT - Keyword, Hashtag)',  'Insert(Search)', 'Insert(Timeline)', # BIG, BIG  'Delete(Search)', 'Delete(Timeline)', 'Delete(KeywordCount)', 'Delete(HashtagCount)'),  ('Hadoop',  'Keyword', 'Hashtag', 'Start', 'Stop', # LYH, LYH, LYH, LYH  'INPUT','OUTPUT'),  ('Visualization(Base)',  'Line Graph', 'Word Cloud', 'Bar Graph', 'Stacked Bar Graph', 'Pie Graph'),  ('Visualization(Analysis)',  'Bar Graph', 'Stacked Bar Graph', 'Pie Graph'),  ('BACK-UP(TWITTER)',), # LSJ  ('ROLL-BACK(TWITTER)',), #LSJ  ('EXIT',))  # --------------------------------------------------------------------------------------------------  while True:  # ----------------------------------------------------------------------------------------------  # Output Title  # ----------------------------------------------------------------------------------------------  for x in range(0, len(Title)):  for y in range(0, len(Title[x])):  print('%s.%s' % (x + 1, Title[x][y])) if y == 0 else print('\t%s%s.%s' % (x + 1, y, Title[x][y]))  print('Choice Number(XX) >>>')  # ----------------------------------------------------------------------------------------------  # Input Number  # ----------------------------------------------------------------------------------------------  cNum = input()  # ----------------------------------------------------------------------------------------------  # 1 ============================================================================================  # ----------------------------------------------------------------------------------------------  # Twitter - Search  # ----------------------------------------------------------------------------------------------  if cNum == '11':  # ------------------------------------------------------------------------------------------  # Issuing Access Token  accessToken = twitter.encodeKey(paramAPI[0][0], paramAPI[0][1])  # ------------------------------------------------------------------------------------------  # Bearer Authentication(=Token Authentication): HTTP authentication scheme that involves security tokens  bearerKey = twitter.getAuthResponse(accessToken)  # ------------------------------------------------------------------------------------------  # Single Operation  # ------------------------------------------------------------------------------------------  '''  # Get Tweets using TwitterAPI  tweets = twitter.searchTweet(bearerKey,  paramAPI[0][2],  query,  date[0],  date[1],  maxResults)  # Prepeocessing Tweets  sJson, sHashtag, sUser = twitter.preprocess(tweets)  '''  # ------------------------------------------------------------------------------------------  # Multi Operation  # ------------------------------------------------------------------------------------------  '''  # Date Converting  fromDate = datetime.strptime(date[0] + '00', '%Y%m%d%H%M%S')  toDate = datetime.strptime(date[1] + '00', '%Y%m%d%H%M%S')  # toDate - fromDate: int(str(toDate - fromDate)[:2])  # 1day = 4 Search  # 00:00:00 ~ 06:00:00  # 06:00:00 ~ 12:00:00  # 12:00:00 ~ 18:00:00  # 18:00:00 ~ 00:00:00  # Get Tweets using TwitterAPI  for CntHour in range(0, int(str(toDate - fromDate)[:2]) \* 2):  print('========== %s ~ %s ==========' % (str((fromDate + timedelta(hours = 12 \* CntHour)).strftime('%Y%m%d%H%M')),  str((fromDate + timedelta(hours = 12 \* (CntHour + 1))).strftime('%Y%m%d%H%M'))))  tweets = twitter.searchTweet(bearerKey,  paramAPI[0][2],  query,  str((fromDate + timedelta(hours = 12 \* CntHour)).strftime('%Y%m%d%H%M')),  str((fromDate + timedelta(hours = 12 \* (CntHour + 1))).strftime('%Y%m%d%H%M')),  maxResults)  # Prepeocessing Tweets  sJson, sHashtag, sUser = twitter.preprocess(tweets, sJson, sHashtag, sUser)  '''  # ------------------------------------------------------------------------------------------  print("Success: Twitter - Search")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Twitter - Timeline  # ----------------------------------------------------------------------------------------------  elif cNum == '12':  # ------------------------------------------------------------------------------------------  # Issuing Access Token  accessToken = twitter.encodeKey(paramAPI[0][0], paramAPI[0][1])  # ------------------------------------------------------------------------------------------  # Bearer Authentication(=Token Authentication): HTTP authentication scheme that involves security tokens  bearerKey = twitter.getAuthResponse(accessToken)  # ------------------------------------------------------------------------------------------  # Get Timelines using get\_Timeline  timelines = twitter.searchTimeline(bearerKey, code[1], count, include\_rts)  # ------------------------------------------------------------------------------------------  # Prepeocessing Timelines  tJson, tHashtag, tUser = twitter.preprocess(timelines)  # ------------------------------------------------------------------------------------------  print("Success: Twitter - Timeline")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # 2 ============================================================================================  # ----------------------------------------------------------------------------------------------  # DataBase - Select(JSON - S, T)  # ----------------------------------------------------------------------------------------------  elif cNum == '21':  # ------------------------------------------------------------------------------------------  sJsonData = db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"group by date\_format(create\_at, '%Y-%m-%d')")  # ------------------------------------------------------------------------------------------  # '"문재인" OR "홍준표" OR "안철수" OR "유승민" OR "심상정"'  sJsonData = [];  sJsonData.append(db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"WHERE TEXT like '%문재인%' group by date\_format(create\_at, '%Y-%m-%d')"))  sJsonData.append(db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"WHERE TEXT like '%홍준표%' group by date\_format(create\_at, '%Y-%m-%d')"))  sJsonData.append(db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"WHERE TEXT like '%안철수%' group by date\_format(create\_at, '%Y-%m-%d')"))  sJsonData.append(db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"WHERE TEXT like '%유승민%' group by date\_format(create\_at, '%Y-%m-%d')"))  sJsonData.append(db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'S\_JSON',"WHERE TEXT like '%심상정%' group by date\_format(create\_at, '%Y-%m-%d')"))  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Select(S\_JSON)")  # ------------------------------------------------------------------------------------------  tJsonData = db.dbSelect("date\_format(create\_at,'%Y-%m-%d') create\_at, count(\*)",'T\_JSON',"group by date\_format(create\_at, '%Y-%m-%d')")  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Select(T\_JSON)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Select(COUNT - Keyword, Hashtag)  # ----------------------------------------------------------------------------------------------  elif cNum == '22':  # ------------------------------------------------------------------------------------------  kCountData = db.dbSelect('\*','KEYWORD\_COUNT','')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Select(KEYWORD\_COUNT)")  # ------------------------------------------------------------------------------------------  hCountData = db.dbSelect('\*','HASHTAG\_COUNT','')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Select(HASHTAG\_COUNT)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Insert(Search)  # ----------------------------------------------------------------------------------------------  elif cNum == '23':  # ------------------------------------------------------------------------------------------  for val in sJson:  db.dbInsert('S\_JSON',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(S\_JSON)")  # ------------------------------------------------------------------------------------------  for val in sHashtag:  db.dbInsert('S\_HASHTAG',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(S\_HASHTAG)")  # ------------------------------------------------------------------------------------------  for val in sUser:  db.dbInsert('S\_USER',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(S\_USER)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Insert(Timeline)  # ----------------------------------------------------------------------------------------------  elif cNum == '24':  # ------------------------------------------------------------------------------------------  for val in tJson:  db.dbInsert('T\_JSON',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(T\_JSON)")  # ------------------------------------------------------------------------------------------  for val in tHashtag:  db.dbInsert('T\_HASHTAG',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(T\_HASHTAG)")  # ------------------------------------------------------------------------------------------  for val in tUser:  db.dbInsert('T\_USER',val)  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Insert(T\_USER)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Delete(Search)  # ----------------------------------------------------------------------------------------------  elif cNum == '25':  # ------------------------------------------------------------------------------------------  db.dbDelete('S\_JSON')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(S\_JSON)")  # ------------------------------------------------------------------------------------------  db.dbDelete('S\_HASHTAG')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(S\_HASHTAG)")  # ------------------------------------------------------------------------------------------  db.dbDelete('S\_USER')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(S\_USER)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Delete(Timeline)  # ----------------------------------------------------------------------------------------------  elif cNum == '26':  # ------------------------------------------------------------------------------------------  db.dbDelete('T\_JSON')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(T\_JSON)")  # ------------------------------------------------------------------------------------------  db.dbDelete('T\_HASHTAG')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(T\_HASHTAG)")  # ------------------------------------------------------------------------------------------  db.dbDelete('T\_USER')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(T\_USER)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Delete(KeywordCount)  # ----------------------------------------------------------------------------------------------  elif cNum == '27':  # ------------------------------------------------------------------------------------------  db.dbDelete('KEYWORD\_COUNT')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(KEYWORD\_COUNT)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Delete(HashtagCount)  # ----------------------------------------------------------------------------------------------  elif cNum == '28':  # ------------------------------------------------------------------------------------------  db.dbDelete('HASHTAG\_COUNT')  # ------------------------------------------------------------------------------------------  print("Success: DataBase - Delete(HASHTAG\_COUNT)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # 3 ============================================================================================  # ----------------------------------------------------------------------------------------------  # Hadoop - Keyword  # ----------------------------------------------------------------------------------------------  elif cNum == '31':  # ------------------------------------------------------------------------------------------  os.system('sqoop import --connect jdbc:mysql://localhost/%s --username %s --password %s --table %s --columns %s --%s hdfs://localhost:9000/user/vi/%s -%s %s'  % (paramSqoop[0], paramSqoop[1], paramSqoop[2],  paramSqoop[3][0], paramSqoop[4][0],  paramSqoop[5][0][0], paramSqoop[5][0][1], paramSqoop[6][0][0], paramSqoop[6][0][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Keyword(import)")  # ------------------------------------------------------------------------------------------  #os.system('yarn jar ../KeywordCount\_full.jar KeywordCount /user/vi/%s/part-m-00000 %s'  # % (paramSqoop[5][0][1], paramSqoop[5][1][1]))  os.system('yarn jar ../KeywordCount\_light.jar KeywordCount /user/vi/%s/part-m-00000 %s'  % (paramSqoop[5][0][1], paramSqoop[5][1][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Keyword(jar)")  # ------------------------------------------------------------------------------------------  os.system('sqoop export --connect jdbc:mysql://localhost/%s --username %s --password %s --table %s --columns %s --%s hdfs://localhost:9000/user/vi/%s/part-r-00000 --%s %s'  % (paramSqoop[0], paramSqoop[1], paramSqoop[2],  paramSqoop[3][1], paramSqoop[4][1],  paramSqoop[5][1][0], paramSqoop[5][1][1], paramSqoop[6][1][0], paramSqoop[6][1][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Keyword(export)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Hadoop - Hashtag  # ----------------------------------------------------------------------------------------------  elif cNum == '32':  # ------------------------------------------------------------------------------------------  os.system('sqoop import --connect jdbc:mysql://localhost/%s --username %s --password %s --table %s --columns %s --%s hdfs://localhost:9000/user/vi/%s -%s %s'  % (paramSqoop[0], paramSqoop[1], paramSqoop[2],  paramSqoop[3][2], paramSqoop[4][2],  paramSqoop[5][2][0], paramSqoop[5][2][1], paramSqoop[6][0][0], paramSqoop[6][0][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Hashtag(import)")  # ------------------------------------------------------------------------------------------  os.system('yarn jar ../HashtagCount.jar HashtagCount /user/vi/%s/part-m-00000 %s'  % (paramSqoop[5][2][1], paramSqoop[5][3][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Hashtag(jar)")  # ------------------------------------------------------------------------------------------  os.system('sqoop export --connect jdbc:mysql://localhost/%s --username %s --password %s --table %s --columns %s --%s hdfs://localhost:9000/user/vi/%s/part-r-00000 --%s %s'  % (paramSqoop[0], paramSqoop[1], paramSqoop[2],  paramSqoop[3][3], paramSqoop[4][3],  paramSqoop[5][3][0], paramSqoop[5][3][1], paramSqoop[6][1][0], paramSqoop[6][1][1]))  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - Hashtag(export)")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Hadoop - Start  # ----------------------------------------------------------------------------------------------  elif cNum == '33':  # ------------------------------------------------------------------------------------------  os.system('start-all.sh')  os.system('hadoop dfsadmin -safemode leave')  os.system('jps')  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Hadoop - Stop  # ----------------------------------------------------------------------------------------------  elif cNum == '34':  # ------------------------------------------------------------------------------------------  os.system('stop-all.sh')  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Hadoop - INPUT  # ----------------------------------------------------------------------------------------------  elif cNum == '35':  # ------------------------------------------------------------------------------------------  os.system('hdfs dfs -rmr /user/vi/\*\_INPUT')  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - \*\_INPUT")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Hadoop - OUTPUT  # ----------------------------------------------------------------------------------------------  elif cNum == '36':  # ------------------------------------------------------------------------------------------  os.system('hdfs dfs -rmr /user/vi/\*\_OUTPUT')  # ------------------------------------------------------------------------------------------  print("Success: Hadoop - \*\_OUTPUT")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # 4 ============================================================================================  # ----------------------------------------------------------------------------------------------  # Visualization(Base) - Line Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '41':  # ------------------------------------------------------------------------------------------  visual.line(sJsonData)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Base) - Line Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Base) - Word Cloud  # ----------------------------------------------------------------------------------------------  elif cNum == '42':  # ------------------------------------------------------------------------------------------  visual.wordCloud(kCountData)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Base) - Word Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Base) - Bar Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '43':  # ------------------------------------------------------------------------------------------  visual.bar(kCountData, 1)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Base) - Bar Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Base) - Stacked Bar Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '44':  # ------------------------------------------------------------------------------------------    # ------------------------------------------------------------------------------------------  print("Success: Visualization(Base) - Stacked Bar Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Base) - Pie Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '45':  # ------------------------------------------------------------------------------------------    # ------------------------------------------------------------------------------------------  print("Success: Visualization(Base) - Pie Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # 5 ============================================================================================  # ----------------------------------------------------------------------------------------------  # Visualization(Analysis) - Bar Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '51':  # ------------------------------------------------------------------------------------------  visual.bar(kCountData)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Analysis) - Bar Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Analysis) - Stacked Bar Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '52':  # ------------------------------------------------------------------------------------------  visual.stackedBar(kCountData)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Analysis) - Stacked Bar Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Visualization(Analysis) - Pie Graph  # ----------------------------------------------------------------------------------------------  elif cNum == '53':  # ------------------------------------------------------------------------------------------  visual.pie(kCountData)  # ------------------------------------------------------------------------------------------  print("Success: Visualization(Analysis) - Pie Graph")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # 6, 7, 8 ======================================================================================  # ----------------------------------------------------------------------------------------------  # DataBase - Back-Up(ALL)  # ----------------------------------------------------------------------------------------------  elif cNum == '6':  # ------------------------------------------------------------------------------------------  os.system('mysqldump -uT-SA -p1234 TWITTER > /home/vi/T-SA.sql')  # ------------------------------------------------------------------------------------------  print("Success: Bacl-UP")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # DataBase - Roll-Back(ALL)  # ----------------------------------------------------------------------------------------------  elif cNum == '7':  # ------------------------------------------------------------------------------------------  os.system('mysql -uT-SA -p1234 TWITTER < /home/vi/T-SA.sql')  # ------------------------------------------------------------------------------------------  print("Success: Roll-Back")  # ------------------------------------------------------------------------------------------  # ----------------------------------------------------------------------------------------------  # Input Number Check(5) - EXIT  # ----------------------------------------------------------------------------------------------  elif cNum == '8':  print('EXIT!!')  break  # ----------------------------------------------------------------------------------------------  # Input Number is not NumList: 11, 12  # : 21, 22, 23, 24, 25, 26, 27, 28  # : 31, 32, 33, 34, 35, 36  # : 41 ,42, 43, 44 ,45  # : 51, 52, 53  # : 6 , 7 , 8  # ----------------------------------------------------------------------------------------------  else:  # ------------------------------------------------------------------------------------------  print('Re-enter')  # ------------------------------------------------------------------------------------------  continue  # ------------------------------------------------------------------------------------------  # -------------------------------------------------------------------------------------------------- |

**2.4.2. TwitterAPI.py**

|  |
| --- |
| # TwitterAPI.py  # Title: Collect Twitter Data, Data preprocessing  # Author: Lee SeokJune  # --------------------------------------------------------------------------------------------------  # import module  # --------------------------------------------------------------------------------------------------  ## Base64 is a way in which 8-bit binary data is encoded into a format that can be represented in 7 bit.  import base64 # Using by 'encodeKey'  ## Module sending HTTP request  import requests # Using by 'getAuthResponse'  # --------------------------------------------------------------------------------------------------  # Class Name: TwitterAPI  # Method list: Generator  # : encodeKey, getAuthResponse  # : searchTweet, searchTimeline  # : preprocess  # --------------------------------------------------------------------------------------------------  class TwitterAPI:  # ----------------------------------------------------------------------------------------------  # Generator  # Set: baseUrl, mapMonth  # ----------------------------------------------------------------------------------------------  def \_\_init\_\_(self, baseUrl):  self.baseUrl = baseUrl  # Drictionary for converting Alphabetic into Number  self.mapMonth = {'Jan':'01', 'Feb':'02', 'Mar':'03', 'Apr':'04', 'May':'05','Jun':'06',  'Jul':'07', 'Aug':'08', 'Sep':'09', 'Oct':'10', 'Nov':'11','Des':'12'}  # ----------------------------------------------------------------------------------------------  # Encode: clientKey, clientSecret  # ----------------------------------------------------------------------------------------------  def encodeKey(self, clientKey, clientSecret):  # Change 'clientKey' and 'clientSecret' to formatting '{}:{}'  # Character encoding type is ASCII(American Standard Code for Information Interchange)  keySecret = '{}:{}'.format(clientKey, clientSecret).encode('ascii')  # Encoding 'keySecret' through Base64.b64encode()  b64EncodedKey = base64.b64encode(keySecret)  # Decoding 'b64EncodedKey' through Str.decode()  b64EncodedKey = b64EncodedKey.decode('ascii')  # 'b64EncodedKey' return  return b64EncodedKey  # ----------------------------------------------------------------------------------------------  # Get Auth Response  # ----------------------------------------------------------------------------------------------  def getAuthResponse(self, b64\_encoded\_key):  # Set URL  authUrl = '{}oauth2/token'.format(self.baseUrl)  # Set Header: Authorization, Content-Type  authHeaders = { 'Authorization': 'Basic {}'.format(b64\_encoded\_key),  'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8' }  # Set Data: grant\_type  authData = { 'grant\_type': 'client\_credentials' }  # POST Transmission Method(URL, header, data)  authResponse = requests.post(authUrl, headers = authHeaders, data = authData)  # 'authResponse' return  return authResponse  # ----------------------------------------------------------------------------------------------  # Get Search Tweets(Return Type: list)  # ----------------------------------------------------------------------------------------------  def searchTweet(self, authResponse, devEnvironmentLabel, query, fromDate, toDate, maxResults):  # Set URL  searchUrl = '{}1.1/tweets/search/fullarchive/{}.json'.format(self.baseUrl, devEnvironmentLabel)  # Keys in data response are token\_type (bearer) and access\_token (your access token)  accessToken = authResponse.json()['access\_token']  # Set Header: Authorization  searchHeaders = { 'Authorization': 'Bearer {}'.format(accessToken) }  # Set Param: query, fromData, toData, maxResults  searchParams = { 'query': query, # Query of the words you want to find  'fromDate': fromDate, # First Date to Search (YYYYmmddHHMM)  'toDate': toDate, # Last Date to Search (YYYYmmddHHMM)  'maxResults' : maxResults } # Tweets Per Response (10~500, Sandbox(~100), Premium(~500))  # GET Transmission Method(URL, header, param)  searchResponse = requests.get(searchUrl, headers = searchHeaders, params = searchParams)  # result list  tweets = []  # Add a line(tweet) to 'tweets'(result list)  for tweet in searchResponse.json()['results']:  tweets.append(tweet)  # tweets(result list) return  return tweets  # ----------------------------------------------------------------------------------------------  # Get Search Timeline(Return Type: 'User' object)  # ----------------------------------------------------------------------------------------------  def searchTimeline(self, authResponse, screenName, count, include\_rts):  # Set URL  searchUrl = '{}1.1/statuses/user\_timeline.json'.format(self.baseUrl)  # Keys in data response are token\_type (bearer) and access\_token (your access token)  accessToken = authResponse.json()['access\_token']  # Set Header: Authorization  searchHeaders = { 'Authorization': 'Bearer {}'.format(accessToken) }  # Set Param: screenName, count  searchParams = { 'screen\_name': screenName,  'count': count,  'include\_rts': include\_rts}  # GET Transmission Method(URL, header, param)  searchResponse = requests.get(searchUrl, headers = searchHeaders, params = searchParams)  # return object  timeline = searchResponse.json()  # 'User' object return  return timeline  # ----------------------------------------------------------------------------------------------  # Preproceeing  # ----------------------------------------------------------------------------------------------  def preprocess(self, tweets, json = [], hashtag = [], user = [], keyNum = 1):  #Load Tweets One Line  for t in tweets:  # Table(TWEET\_JSON)  json.append(['-'.join([t['created\_at'][26:30], # Creation Date and Time (YYYY-mm-dd HH:MM:SS)  self.mapMonth[t['created\_at'][4:7]],  t['created\_at'][8:19]]),  t['id\_str'], # str(id)  t['text'], # tweet  t['truncated'], #  str(keyNum).zfill(4), # TWEET\_HASHTAG(F.K)  str(keyNum).zfill(4), # TWEET\_USER(F.K)  t['retweet\_count'], # Retweet Count  t['favorite\_count'], # Favorite Count  t['lang']]) # Written Language  # Table(TWEET\_HASHTAG)  for h in t['entities']['hashtags']:  hashtag.append([str(keyNum).zfill(4), # P.K  h['indices'][0], # Start Position of Hashtag  h['indices'][1], # End Position of Hashtag  h['text']]) # Hashtag  # Table(TWEET\_USER)  user.append([str(keyNum).zfill(4), # P.K  t['user']['id\_str'], # str(id)  t['user']['name'], # User Name  t['user']['screen\_name'], # User Screen Name(@)  t['user']['location'], # User Location  t['user']['description']]) # User Description  # Increase in Key Value  keyNum += 1  # (json, hashtag, user) return  return json, hashtag, user  # -------------------------------------------------------------------------------------------------- |

**2.4.3. DBModule.py**

|  |
| --- |
| # DBModule.py  # Title: MariaDB connection and close, SQL command DML(Data Manipulation Language) processing  # Author: Bae InGyu  # --------------------------------------------------------------------------------------------------  # Modules for Python to MariaDB Interworking  import pymysql  # --------------------------------------------------------------------------------------------------  # Class Name: DBModule  # Method list: dbConnect, dbClose  # : dbSelect, dbInsert, dbDelete  # -------------------------------------------------------------------------------------------------  class DBModule:  # ----------------------------------------------------------------------------------------------  # Generator  # Set: host, user, pswd, dbNm, char  # ----------------------------------------------------------------------------------------------  def \_\_init\_\_(self, host, user, pswd, dbNm, char):  self.host = host  self.user = user  self.pswd = pswd  self.dbNm = dbNm  self.char = char  # ----------------------------------------------------------------------------------------------  # Connect DataBase(MariaDB)  # ----------------------------------------------------------------------------------------------  def dbConnect(self):  # Set up DB Connection Environment  conn = pymysql.connect(host = self.host, # IP  user = self.user, # User ID  password = self.pswd, # User Password  db = self.dbNm, # DB Name  charset = self.char) # Encoding Character  # Create Cursor  curs = conn.cursor()  # 'conn', 'curs' return  return conn, curs  # ----------------------------------------------------------------------------------------------  # Close: Cursor, DataBase Connection  # ----------------------------------------------------------------------------------------------  def dbClose(self, curs, conn):  curs.close()  conn.close()  # ----------------------------------------------------------------------------------------------  # Load Data From Database  # ----------------------------------------------------------------------------------------------  def dbSelect(self, cols, tabs, cond):  try:  # DB Connect  conn, curs = self.dbConnect()  # Write Query: Select Data  sql = 'SELECT %s FROM %s %s;' % (cols, tabs, cond)  # Execute Query  curs.execute(sql)  # Fetch  selectData = curs.fetchall()  # 'selectData' return  return selectData  except:  print('Error: Select Failed')  finally:  # DB Close  self.dbClose(curs, conn)  # ----------------------------------------------------------------------------------------------  # Save Data to DataBase  # ----------------------------------------------------------------------------------------------  def dbInsert(self, tableNm, value):  try:  # DB Connect  conn, curs = self.dbConnect()  # Write Query: Select Data / Execute Query  sql = 'INSERT INTO ' + tableNm  if tableNm in ['S\_JSON', 'T\_JSON']:  # Check List: CREATED\_AT, ID  if self.dbSelect('EXISTS(SELECT \*', tableNm, "WHERE CREATE\_AT = '%s' AND ID = '%s')" % (value[0], value[1]))[0][0] == 0:  sql += ' VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s);'  curs.execute(sql, (value[0], value[1], value[2], value[3], value[4], value[5], value[6], value[7], value[8]))  elif tableNm in ['S\_HASHTAG', 'T\_JHASHTAG']:  sql += ' VALUES(%s, %s, %s, %s);'  curs.execute(sql, (value[0], value[1], value[2], value[3]))  elif tableNm in ['S\_USER', 'T\_USER']:  sql += ' VALUES(%s, %s, %s, %s, %s, %s);'  curs.execute(sql, (value[0], value[1], value[2], value[3], value[4], value[5]))  # Apply Query  conn.commit()  except:  print('Error: Insert Failed')  finally:  # DB Close  self.dbClose(curs, conn)  # ----------------------------------------------------------------------------------------------  # Elimination Data to DataBase  # ----------------------------------------------------------------------------------------------  def dbDelete(self, tableNm):  try:  # DB Connect  conn, curs = self.dbConnect()  # Write Query: Select Data  sql = 'DELETE FROM %s;' % tableNm  # Execute Query  curs.execute(sql)  # Apply Query  conn.commit()  except:  print('Error: Delete Failed')  finally:  # DB Close  self.dbClose(curs, conn)  # ------------------------------------------------------------------------------------------------- |

**2.4.4. Analysis\_Visual.py**

|  |
| --- |
| # Analysis\_Visual.py  # Title: Analysis and Visualization  # Author: Lee SeokJune  # --------------------------------------------------------------------------------------------------  # Import Module and Install  # --------------------------------------------------------------------------------------------------  # Library for handing DataFrame  ## pip3 install pandas  import pandas as pd  ## pip3 install numpy  import numpy as np  # Word Cloud related Module  ## pip3 install wordcloud  from wordcloud import WordCloud  # Graph drawing and Word Count Output Module  ## pip3 install matplotlib  import matplotlib.pyplot as plt  from matplotlib import font\_manager, rc  ## sudo apt-get install fonts-nanum\*  # --------------------------------------------------------------------------------------------------  # Class Name: Analysis\_Visual  # Method list: Generator  # : preprocessData, setPlt, setLabel, autoLabel, setTextList  # : wordCloud  # : line, bar, stackedBar, pie  # --------------------------------------------------------------------------------------------------  class Analysis\_Visual:  # ----------------------------------------------------------------------------------------------  # Generator  # Read CSV File: initialize\_president.csv  # ----------------------------------------------------------------------------------------------  def \_\_init\_\_(self, date, query):  # Set Param  self.date = date  self.query = query  # Read Data(the 19th presiential election)  self.realVoteData = pd.read\_csv('../initialize\_president.csv', encoding = 'euc-kr', header = 1)  # Set Text  self.title = '19대 대통령 선거'  # Path(Hangle Font)  ## sudo apt-get install fonts-nanum\*  self.hPath = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'  # Set Hangle Font, size / Graph size  font\_name = font\_manager.FontProperties(fname = self.hPath).get\_name()  rc('font', family = font\_name)  rc('figure', figsize = (10, 5))  # ----------------------------------------------------------------------------------------------  # Preprocessing  # ----------------------------------------------------------------------------------------------  def preprocessData(self, kCountData, stat = 0):  # Preprocessing  data = pd.DataFrame(list(kCountData), columns = ['KEYWORD', 'COUNT']).sort\_values(['COUNT'], ascending = [False])  if stat == 1:  data = data.head(10)  else:  data = data[data['KEYWORD'].isin(['문재인', '홍준표', '안철수', '유승민', '심상정'])]  data = pd.merge(data.sort\_values(['KEYWORD']),  np.transpose(self.realVoteData)[0]['문재인':'심상정'].sort\_index(),  left\_on = ['KEYWORD'], right\_index = True)  data = data.rename(columns = {0 : 'COUNT\_'})  data['COUNT\_'] = data['COUNT\_'].replace(',', '', regex = True).astype('int')  data['PP(3)'] = round(data['COUNT'] / data['COUNT'].sum() \* 100, 3)  data['PP(3)\_'] = round(data['COUNT\_'] / data['COUNT\_'].sum() \* 100, 3)  return data  # ----------------------------------------------------------------------------------------------  # Set matplotlib Param  # ----------------------------------------------------------------------------------------------  def setPlt(self, title, titleSize = 40, textNum = 0):  # Set Title  plt.suptitle(title, fontsize = titleSize)  # Set Text(Date, Query)  fromDate = '-'.join([self.date[0][0:4], self.date[0][4:6], self.date[0][6:8]])  toDate = '-'.join([self.date[1][0:4], self.date[1][4:6], str(int(self.date[1][6:8]) - 1).zfill(2)])  if textNum == 1:  plt.text(10 \* 2.54 \* 15.5, -10, '(기간: ' + fromDate + ' ~ ' + toDate + ')', ha = 'right', wrap=True, fontsize = 20, color = 'red')  plt.text(10 \* 2.54 \* 15.5, -1, '(쿼리: ' + self.query + ')', ha = 'right', wrap=True, fontsize = 20, color = 'red')  # ----------------------------------------------------------------------------------------------  # Set X, Y label  # ----------------------------------------------------------------------------------------------  def setLabel(self, xl, yl):  # Set X Label  plt.xlabel(xl, fontsize = 20)  # Set X Label  plt.ylabel(yl, fontsize = 20)  # ----------------------------------------------------------------------------------------------  # Set X Value  # ----------------------------------------------------------------------------------------------  def autoLabel(self, ax, rects, xpos = 'center'):  ha = {'center': 'center', 'right': 'left', 'left': 'right'}  offset = {'center': 0, 'right': 1, 'left': -1}  for rect in rects:  height = rect.get\_height()  ax.annotate('{}'.format(height),  xy = (rect.get\_x() + rect.get\_width() / 2, height),  xytext = (offset[xpos] \* 3, 3),  textcoords = "offset points",  ha = ha[xpos], va = 'bottom')  # ----------------------------------------------------------------------------------------------  # Set Text list(fontsize, weight)  # ----------------------------------------------------------------------------------------------  def setTextList(self, tList, fs = 10, w = 'bold'):  for i in range(0, len(tList)):  tList[i].set\_fontsize(fs)  if w != '':  tList[i].set\_weight(w)  # ----------------------------------------------------------------------------------------------  # Word Cloud with kCountData(TWITTER - KEYWORD\_COUNT)  # ----------------------------------------------------------------------------------------------  def wordCloud(self, kCountData):  # Set plt  self.setPlt('19대 대통령 선거 Word Count 결과', 40, 1)  # Set Word Cloud  wc = WordCloud(font\_path = self.hPath, # Font Path  background\_color = 'white', # Backgroud Color  max\_words = 1000, # Maximum Number of Words  contour\_width = 5)  # Add Data(tweetData(Word, Frequency)) in Word Cloud  wc = wc.generate\_from\_frequencies(dict(kCountData))  # Set Display an Image  plt.imshow(wc, # Array-like or PIL image  interpolation = 'bilinear') # 'none', 'nearest', 'bilinear', 'bicubic', etc.  # Set x,y axis (off - invisible)  plt.axis('off')  # Graph Output  plt.get\_current\_fig\_manager().full\_screen\_toggle()  plt.show()  # ----------------------------------------------------------------------------------------------  # Line Graph with sJsonData(TWITTER - S\_JSON)  # ----------------------------------------------------------------------------------------------  def line(self, sJsonData):  # Preprocessing  data = pd.merge(pd.merge(pd.merge(pd.merge(pd.DataFrame(list(sJsonData[0]), columns = ['DATE', '문재인']),  pd.DataFrame(list(sJsonData[1]), columns = ['DATE', '홍준표']), on = 'DATE'),  pd.DataFrame(list(sJsonData[2]), columns = ['DATE', '안철수']), on = 'DATE'),  pd.DataFrame(list(sJsonData[3]), columns = ['DATE', '유승민']), on = 'DATE'),  pd.DataFrame(list(sJsonData[4]), columns = ['DATE', '심상정']), on = 'DATE')  # Set plt  self.setPlt('19대 대통령 선거기간(2017-04-18 ~ 2017-05-09)동안 일별 후보 언급 횟수')  # Set Line  plt.plot(data['DATE'], data['문재인'], lw = 2, marker = 'o')  plt.plot(data['DATE'], data['홍준표'], lw = 2, marker = 'o')  plt.plot(data['DATE'], data['안철수'], lw = 2, marker = 'o')  plt.plot(data['DATE'], data['유승민'], lw = 2, marker = 'o')  plt.plot(data['DATE'], data['심상정'], lw = 2, marker = 'o')  plt.legend(('문재인', '홍준표', '안철수', '유승민', '심상정'), loc = 'upper right', fontsize = 'xx-large')  plt.grid()  # Set Label  self.setLabel('Date', 'Count')  #Graph Output  plt.get\_current\_fig\_manager().full\_screen\_toggle()  plt.show()  # ----------------------------------------------------------------------------------------------  # Bar Graph with kCountData(TWITTER - KEYWORD\_COUNT)  # ----------------------------------------------------------------------------------------------  def bar(self, kCountData, stat = 0):  # Preprocessing  data = self.preprocessData(kCountData, stat)  # Set Bar  if stat == 1:  # Set plt  self.setPlt('19대 대통령 선거기간(2017-04-18 ~ 2017-05-09)동안 언급된 키워드(TOP 10)')  plt.bar(data['KEYWORD'], data['COUNT'])  # Set Label  self.setLabel('Keyword', 'Count')  else:  # the x locations for the groups  ind = np.arange(len(data['KEYWORD']))  # the width of the bars  width = 0.35  # Set ax  fig, ax = plt.subplots()  p1 = ax.bar(ind - width / 2, round(data['COUNT'] / data['COUNT'].sum() \* 100, 3), width, label = '트윗 언급')  p2 = ax.bar(ind + width / 2, round(data['COUNT\_'] / data['COUNT\_'].sum() \* 100, 3), width, label = '실제 득표')  # title  ax.set\_title('\n\n\n\n', fontsize = 5)  # label  ax.set\_xlabel('Candidate', fontsize = 20)  ax.set\_ylabel('Count(%)', fontsize = 20)  # x axis  ax.set\_xticks(ind)  ax.set\_xticklabels(data['KEYWORD'])  ax.legend(fontsize = 'xx-large')  self.autoLabel(ax, p1, 'center')  self.autoLabel(ax, p2, 'center')  # add layout  fig.tight\_layout()  # title  self.setPlt('후보 TOP5 트윗 언급률(%) 및 실제 득표율(%)')  #Graph Output  plt.get\_current\_fig\_manager().full\_screen\_toggle()  plt.show()  # ----------------------------------------------------------------------------------------------  # Stacked Bar Graph with kCountData(TWITTER - KEYWORD\_COUNT)  # ----------------------------------------------------------------------------------------------  def stackedBar(self, kCountData):  # Preprocessing  data = self.preprocessData(kCountData)  dataList = data[['PP(3)', 'PP(3)\_']].values.tolist()  # the x locations for the groups  ind = np.arange(2)  # the width of the bars: can also be len(x) sequence  width = 0.35  # bar  fig, ax = plt.subplots()  p1 = ax.bar(ind, dataList[0], width)  p2 = ax.bar(ind, dataList[1], width, bottom = dataList[0])  p3 = ax.bar(ind, dataList[2], width, bottom = [a + b for a, b in zip(dataList[0], dataList[1])])  p4 = ax.bar(ind, dataList[3], width, bottom = [a + b + c for a, b, c in zip(dataList[0], dataList[1], dataList[2])])  p5 = ax.bar(ind, dataList[4], width, bottom = [a + b + c + d for a, b, c, d in zip(dataList[0], dataList[1], dataList[2], dataList[3])])  # title  ax.set\_title('\n\n\n\n', fontsize = 5)  # label  ax.set\_ylabel('Count(%)', fontsize = 20)  ax.set\_xticks(ind)  ax.set\_xticklabels(('트윗 언급', '실제 득표'))  ax.set\_yticks(np.arange(0, 110, 5))  ax.legend((p5[0], p4[0], p3[0], p2[0], p1[0]), data['KEYWORD'].sort\_values(ascending = [False]), fontsize = 'xx-large')  # add layout  fig.tight\_layout()  # Set title  self.setPlt('후보 TOP5 트윗 언급률(%) 및 실제 득표율(%)')  #Graph Output  plt.get\_current\_fig\_manager().full\_screen\_toggle()  plt.show()  # ----------------------------------------------------------------------------------------------  # Pie Graph with kCountData(TWITTER - KEYWORD\_COUNT)  # ----------------------------------------------------------------------------------------------  def pie(self, kCountData):  # Preprocessing  data = self.preprocessData(kCountData)  # pie  fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize = (9, 9))  ax1.set\_title('트윗 언급', fontsize = 30)  patches, texts, autotexts = ax1.pie(data['COUNT'], explode = (0.01, 0.01, 0.01, 0.01, 0.01), labels = data['KEYWORD'], autopct = '%1.3f%%', startangle = -360 \* data['PP(3)'].head(1).values[0] / 100 + 90)  self.setTextList(texts, 20, '')  self.setTextList(autotexts, 15)  ax2.set\_title('실제 득표', fontsize = 30)  patches, texts, autotexts = ax2.pie(data['COUNT\_'], explode = (0.01, 0.01, 0.01, 0.01, 0.01), labels = data['KEYWORD'], autopct = '%1.3f%%', startangle = -360 \* data['PP(3)\_'].head(1).values[0] / 100 + 90)  self.setTextList(texts, 20, '')  self.setTextList(autotexts, 15)  # add layout  fig.tight\_layout()  # Set title  self.setPlt('후보 TOP5 트윗 언급률(%) 및 실제 득표율(%)')  #Graph Output  plt.get\_current\_fig\_manager().full\_screen\_toggle()  plt.show()  # --------------------------------------------------------------------------------------------------  # test  # https://zzsza.github.io/development/2018/08/24/data-visualization-in-python/  # -------------------------------------------------------------------------------------------------- |

**Section 5. DB and Table Definition**

|  |  |  |
| --- | --- | --- |
| **TABLE LIST** | | |
| **DB** | **TABLE** | **COMMENT** |
| TWITTER | S\_JSON | SearchAPI의 JSON |
| S\_HASHTAG | SearchAPI의 HASHTAG |
| S\_USER | SearchAPI의 USER |
|  |  |
| T\_JSON | Timeline의 JSON |
| T\_HASHTAG | Timeline의 HASHTAG |
| T\_USER | Timeline의 USER |
|  |  |
| KEYWORD\_COUNT |  |
| HASHTAG\_COUNT |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.S\_JSON | | | | |
| **Description** | | A table for storing approximate information among keyword search results. | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | CREATE\_AT | | DATETIME | X | PK | Tweets Created |
| 02 | ID | | VARCHAR(20) | X | PK |  |
| 03 | TEXT | | TEXT | X |  |  |
| 04 | TRUNCATED | | CHAR(1) | X |  | 0 - F / 1 - T |
| 05 | HASHTAG | | CHAR(4) | X |  |  |
| 06 | USER | | CHAR(4) | X |  |  |
| 07 | RETWEET\_COUNT | | INT | O |  |  |
| 08 | FAVORITE\_COUNT | | INT | O |  |  |
| 09 | LANG | | VARCHAR(3) | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.S\_HASHTAG | | | | |
| **Description** | | Save hashtag information table of Twitter search results | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HCODE | | CHAR(4) | X |  |  |
| 02 | START | | INT | X |  |  |
| 03 | END | | INT | X |  |  |
| 04 | TEXT | | TEXT | X |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.S\_USER | | | | |
| **Description** | | Twitter Search Results Save Author Information Table | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | UCODE | | CHAR(4) | X |  |  |
| 02 | ID | | VARCHAR(20) | X |  |  |
| 03 | NAME | | VARCHAR(50) | X |  |  |
| 04 | SCREEN\_NAME | | VARCHAR(50) | X |  | Name starting with @ |
| 05 | LOCATION | | VARCHAR(10) | O |  |  |
| 06 | DESCRIPTION | | TEXT | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_JSON | | | | |
| **Description** | | A table for storing approximate information among timeline search results. | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | CREATE\_AT | | DATETIME | X | PK | Tweets Created |
| 02 | ID | | VARCHAR(20) | X | PK |  |
| 03 | TEXT | | TEXT | X |  |  |
| 04 | TRUNCATED | | CHAR(1) | X |  | 0 - F / 1 - T |
| 05 | HASHTAG | | CHAR(4) | X |  |  |
| 06 | USER | | CHAR(4) | X |  |  |
| 07 | RETWEET\_COUNT | | INT | O |  |  |
| 08 | FAVORITE\_COUNT | | INT | O |  |  |
| 09 | LANG | | VARCHAR(3) | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_HASHTAG | | | | |
| **Description** | | Save hashtag information table during timeline search results | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HCODE | | CHAR(4) | X |  |  |
| 02 | START | | INT | X |  |  |
| 03 | END | | INT | X |  |  |
| 04 | TEXT | | TEXT | X |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_USER | | | | |
| **Description** | | Save Author Information Table During Timeline Search Results | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | UCODE | | CHAR(4) | X |  |  |
| 02 | ID | | VARCHAR(20) | X |  |  |
| 03 | NAME | | VARCHAR(50) | X |  |  |
| 04 | SCREEN\_NAME | | VARCHAR(50) | X |  | Name starting with @ |
| 05 | LOCATION | | VARCHAR(10) | O |  |  |
| 06 | DESCRIPTION | | TEXT | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.KETWORD\_COUNT | | | | |
| **Description** | | Keyword Count Table | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | KEYWORD | | VARCHAR(10) | X | PK |  |
| 02 | COUNT | | INT | X |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.HASHTAG\_COUNT | | | | |
| **Description** | | Hashtag count table | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HASHTAG | | VARCHAR(10) | X | PK |  |
| 02 | COUNT | | INT | X |  |  |

**Section 6. Implementation (Hadoop)**

**2.6.1. Spooq (Interworking between DB and HDFS)**

**Sqoop Import**

|  |  |
| --- | --- |
| sqoop import --connect jdbc:mysql://localhost/TWITTER --username T-SA --password 1234 --table KEYWORD\_HASHTAG --columns TEXT --target-dir hdfs://localhost:9000/user/vi/HASHTAG\_INPUT -m 1 | |
| connect | jdbc:DB type://IP address/DB name |
| username | DB Connection Account |
| password | DB Password |
| table | Table to import data |
| columns | List of columns to import from the table |
| target-dir | HDFS directory path to be stored |

**Sqoop Export**

|  |  |
| --- | --- |
| sqoop export --connect jdbc:mysql://localhost/VISUAL --username T-SA --password 1234 --table HASHTAG --export-dir hdfs://localhost:9000/user/vi/HASHTAG\_OUTPUT/part-r-00000 --columns HASHTAG,COUNT --input-fields-terminated-by "\t" | |
| connect | jdbc:DB type://IP address/DB name |
| username | DB Connection Account |
| password | DB Password |
| table | Table to store data |
| columns | List of columns to be mapped in the table |
| export-dir | HDFS directory path from which to import data |
| input-fields-terminated-by | Separator |

**2.6.2. Mapper and Reducer (java file)**

**KeywordCount.java / HashtagCount.java(equal code)**

|  |
| --- |
| /\*\*  \* @KeywordCount  \* @Title: Driver class (doing the registration of maps and revisions)  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \*/  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class KeywordCount {  public static void main(String[] args) throws Exception {  // create the conf object required for running Hadoop (hdfs-site, core-site)  Configuration conf = new Configuration();  // create Job Object for Job execution  Job job = Job.getInstance(conf, "Keywordcount");  // specify the user library files required to run the job  job.setJarByClass(KeywordCount.class);  // set the classes to be used in the Job  job.setMapperClass(KeywordCountMapper.class);  job.setCombinerClass(KeywordCountReducer.class);  job.setReducerClass(KeywordCountReducer.class);  // set the key and value type of output data for the mapper and reducer classes  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  // set I/O data path.  // The first factor is the input parameter, the second is the output parameter  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  // If the 깂, such as the path required to execute, run the Job.  System.exit(job.waitForCompletion(true) ? 0 : 1);  }  } |

**KeywordCountMapper.java / HashtagCountMapper.java(Excluding natural language processing, equal code)**

|  |
| --- |
| /\*\*  \* @KeywordCountMapper  \* @Title: Mapper classes (to add selected data (key, value) to the same key)  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \*/  import java.io.IOException;  import java.util.Iterator;  import java.util.List;  import java.util.StringTokenizer;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  import kr.co.shineware.nlp.komoran.constant.DEFAULT\_MODEL;  import kr.co.shineware.nlp.komoran.core.Komoran;  import kr.co.shineware.nlp.komoran.model.KomoranResult;  // the generic type of Mapper class  // change to the type corresponding to long, int, and string required by Hadoop  public class KeywordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  // Store constant 1 as IntWritable value.  // From when word count with a value of the intwritable ridyuseu  private final static IntWritable one = new IntWritable(1);  // Text object that you want to save words  private Text word = new Text();  // Mapping keys to redirect to a revision for the entered key and value  @Override  protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  // Cut the text entered by the space unit. Processes the rest of the text, including the rest of the special characters  StringTokenizer itr = new StringTokenizer(value.toString(), " \t\n\r,.'\"-=%…()[]{}“▷+ⓒ!?:;#\'");  // Repeat as long as there is no next token to return (false).  while(itr.hasMoreTokens()) {  String token = itr.nextToken();  // ---------------------------------------------자연어 처리 부분---------------------------------------------------  // Create a Comoran Object Use DEFAULT\_MODEL Default Dictionary << Can be predefined  Komoran komoran = new Komoran(DEFAULT\_MODEL.FULL);  // Add user dictionary paths (user nouns can be defined)  komoran.setUserDic("/home/vi/eclipse-workspace/KeywordCount/src/dic.user");  // Analyze Received Words  KomoranResult analyzeResultList = komoran.analyze(token);  // After defining the tokens list, sort and load for nouns.  List<String> tokens = analyzeResultList.getMorphesByTags("NP","NNP","NNG");  // After creating an Iterator for reading elements, load the tokens content.  Ite rator<String> itrs = tokens.iterator();  // ----------------------------------------------------------------------------------------------------------------  // The context object is used to export as a key-value pair and is characterised by the output type.  while(itrs.hasNext()) {  // Create a ktr variable to save natural language processed words (words, clear spaces)  String ktr = itrs.next().trim();  // Conditions if the word stored in the ktr is greater than or equal to 5 characters.  if(ktr.getBytes().length > (byte)3 && ktr.getBytes().length < (byte)16) {  // word객체에 ktr 삽입.  word.set(ktr);  // The context object is used to export as a key-value pair and is characterised by the output type.  context.write(word, one);  }  }  }  }  } |

**KeywordCountReducer.java / HashtagCountReducer.java(equal code)**

|  |
| --- |
| /\*\*  \* @KeywordCountReducer  \* @Title: Reuse class (to add selected data (key, value) from the map by the same key)  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \*/  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  // Reducer class inheritance, output of input and output in the same type.  public class KeywordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  private IntWritable result = new IntWritable();  // Override the override method for adding additional functionality by importing and adding output parameters  // The reason why it is enclosed with Iterable<> is to extract values only from the values stored in IntWritable because the values stored in the map are tied up.  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  // a variable that stores values that increase by the number of words.  int sum = 0;  // Find out the number of each letter and add up the words.  for (IntWritable val : values) {  sum += val.get();  }  // Set the output value to the result that generated IntWritable object, the input and output type of MapReduce.  // In this case, the output value is the sum of the words.  result.set(sum);  // Use the key of the input data as the output key through the write method of the object.  context.write(key, result);  }  } |

**dic.user - user defined dictionary (natural language processing) / Requires KOMORAN.jar**

|  |
| --- |
| #This is a user dictionary file.  #The user will print the corresponding verbatim for the section containing contents in the user's dictionary within the input statement  #If you don't write down the form of a phrase, it is recognized as an NNP by default.  바람과 함께 사라지다 NNG  바람과 함께 NNP  자연어 NNG  아이오아이 NNG  캡틴아메리카  가나다라마 |

**2.6.3. Hadoop Execute (jar file)**

|  |
| --- |
| yarn jar /home/vi/hadoop/jar/HashtagCount.jar HashtagCount /user/vi/HASHTAG\_INPUT/part-m-00000 HASHTAG\_OUTPUT |
| yarn: From Hadoop2.X, yarn manages the cluster.  yarn jar (path/)jar) driverClass (HDFS path of data to be stored) |

**Chapter 3. Conclusion**

**Section 1. Execution Environment**

To implement T-SA, the implementation environment must be established as shown in [Table 4], [Table 5] and [Table 6].

**[Table 4]** Python modules and utility and installation commands required to run T-SA.

|  |  |
| --- | --- |
| Modules and Utilities | Installation Command |
| PIP3 | sudo apt-get install python3-pip |
| Pandas | pip3 install pandas |
| Word Cloud | pip3 install wordcloud |
| Matplotlib | pip3 install matplotlib |
| Pymysql | pip3 install pymyql |
| 한글 폰트(나눔 글꼴) | sudo apt-get install fonts-nanum\* |

**[Table 5]** Hadoop related utilities and installation commands required to run T-SA

|  |  |
| --- | --- |
| Utilities | Installation Command |
| ssh | sudo apt-get install ssh |
| pdsh | sudo apt-get install pdsh |

**[Table 6]** Other utilities and installation commands (URLs) required to run T-SA

|  |  |
| --- | --- |
| Utilities | Installation Command (URL) |
| OpenJDK8 | sudo apt-get install openjdk-8-jdk |
| KOMORAN | https://github.com/shin285/KOMORAN |

**Section 2. Tweeter API Parameter Settings Value**

T-SA offers two features (Search, Timeline) that bring Twitter-related information, and the parameters required for each function are different. Use query, fromDate, toDate, maxResults in Search and screen\_name, count, include\_rts in Timeline. The 19th president of the election, use the data on each of the order on the (query, maxresults) parameters and use search."Moon Jae-in" or "Hong Joon-pyo" or "Ahn Cheol-soo" or "Yoo Seung-min" or "Sim Sang-jung", a 100. It also gave the start and end times of the period as "201704180000" and "201705100000."The reason is that the period of the 19th presidential election is from midnight on April 17, 2017 to 9 p.m. voting closing in May 2017. However, because of the limit of 500 tweets being searched 100 times even though Primium was used, and because Search's search criteria were up-to-date, Sandbox and Primium were mixed together to search four times a day for 22 days. As a result, daily tweets (1200=100+500+100+500) were retrieved during the election (22) to collect about 26,000 tweets.

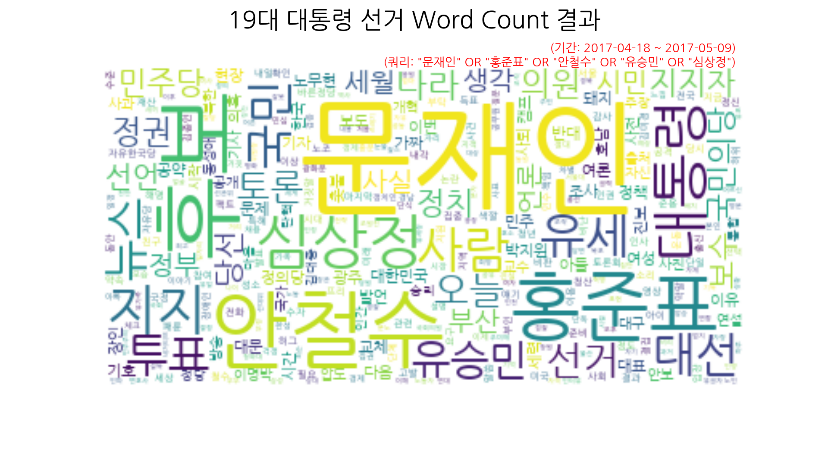
**Section 3. Analysis and Visualization**

T-SA analyzed three main things.

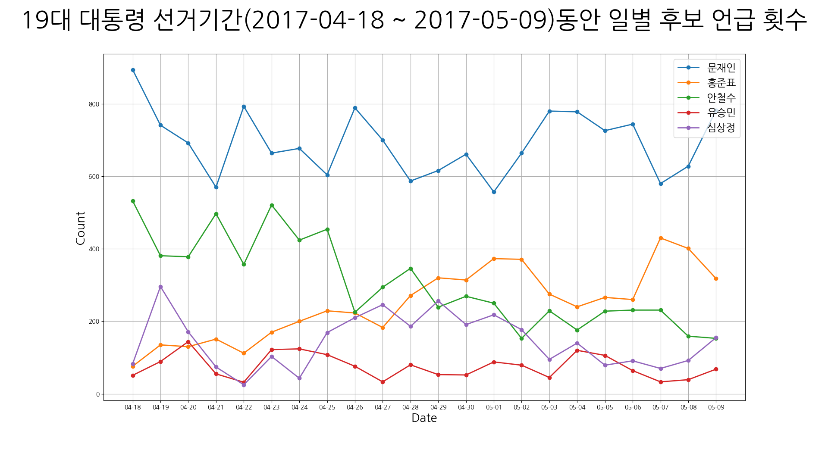
The first was to count the keywords used during the 19th presidential election to analyze which keywords were used a lot through Word Cloud and visualize them ([Figure 16]) through Word Cloud.

Second, during the 19th presidential election, the names of the top five candidates with the most votes were analyzed and visualized on a line graph ([Figure 17]) each day.

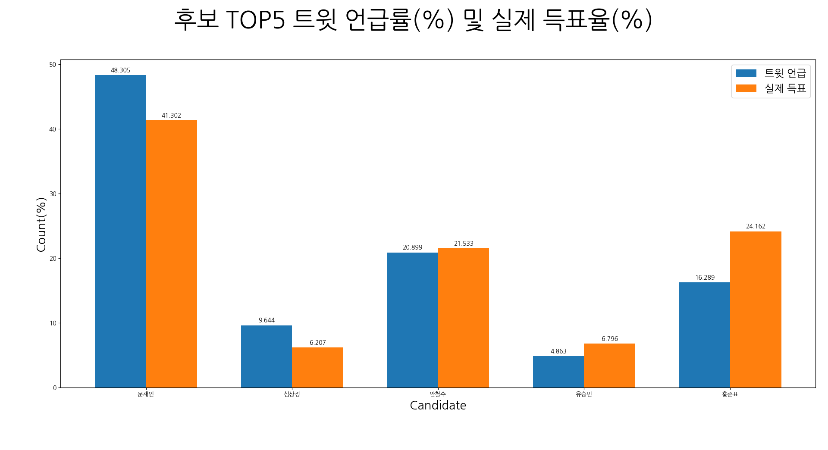
Finally, a bar graph ([Figure 18]) and a circle graph ([Figure 19]) were visualized by comparing the count of keywords used during the 19th presidential election with the number of votes provided by the National Election Commission.



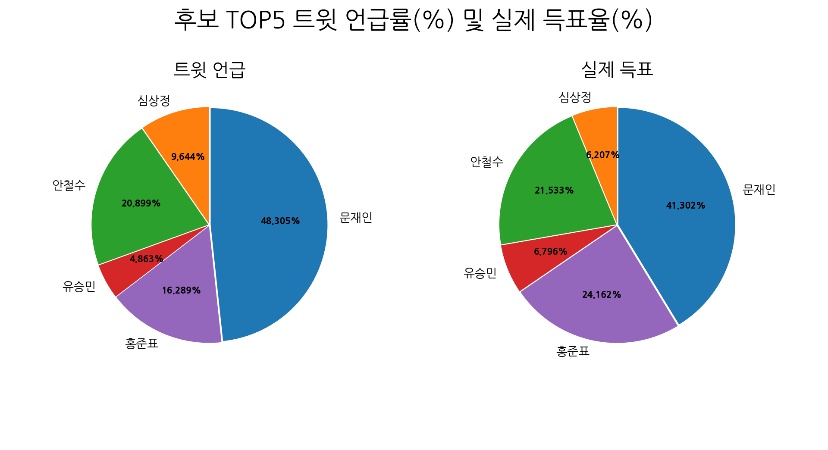
**[Figure 16]** Analyze the number of keywords used through Word Cloud



**[Figure 17]** Daily Reference Analysis through Line Graphs



**[Figure 17]** Tweets referenced by bar graph and actual vote comparison analysis



**[Figure 18]** Tweets referenced by One graph and actual vote comparison analysis

**Section 4. Final conclusion.**

[Figure 17] or [Figure 18] showed that the tweets analyzed by T-SA were similar to the actual turnout of the candidates. Based on this analysis, more samples will be analyzed through the previous presidential and parliamentary elections as well as the 19th presidential elections, so that we can check the hit rate and predict who will be elected in the 20th presidential election of the Republic of this year.

**Section 5. What we felt**

**Lee SeokJune**

When I first started the project, I thought that because of my experience in learning at the conference or in the company as a team leader, I felt that my thoughts were different as I progressed. In particular, I only worked as a team member before, but I felt a lot of pressure because I was not able to communicate well and lead the team well. So, in the second semester, if I am the team leader, I will communicate well with the team members, and if I am not the team leader, I will think about the team leader and work hard.

**Lee YunHyuck**

As I was working on the project, I was able to understand how to use APIs provided by Twitter. Based on this, it occurred to me that not only Twitter but also sites such as Google, Naver and Facebook could carry out projects with added new features. And in this project, if the version of Hadoop's implementation was not compatible, the interaction would not work, which led us to rethink how to efficiently handle distributed data. Finally, when final results were reached, we analyzed the tweets during the election period under the theme of the 19th presidential election and compared the actual turnout with the mentioned number of votes, we felt that we wanted to see similar results and that they would be useful for future elections based on the results applied to local elections in foreign countries, not the 19th presidential election, after this semester.

**Bae InGyu**

I was in charge of the connection between Python and Db while working on the project.

In my role, I learned the knowledge that I did not know by searching for the database and programming knowledge and asking the team leader. I also didn't know how to use APIs, but I found out through this project. Finally, a comparative analysis of the 19th presidential election. It was interesting to see that the actual rate of votes cast and the rate of twittering was quite similar.

**Seo JaeIck**

As I was working on the team project this time, I felt that there was a huge difference between working on the project alone. Feedback from the team members of the team that they did not know or would have passed on if they had done so alone could reveal the deficiencies and correct them in detail. I could also feel that my mistake came very far to the other team members. By making mistakes, I felt that it was not only damaging to myself, but also to all members of the team, and I tried to reduce my mistakes. From using TwitterAPI, visualization using data was carried out, and data containing various information was selected and processed. The process showed that the data was easier to use by removing unnecessary data and using only the data to be used. That is why we can see the importance of the pre-processing process. Lastly, I would like to use a library other than Matplotlib that I use now to proceed with visualization.

**Chapter 4. Appendix**

**Appendix 1. MariaDB related settings**

**Setting the Permission Table**

The method for setting the authority table in Maria DB is as follows.

1. Run sudo mysql\_secure\_installation in the terminal.

2. Enter the root account password.

|  |
| --- |
| NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB SERVER IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!  In order to log into MariaDB to secure it, we’ll need the current password for the root user. If you’ve just installed Mari-aDB, and you haven’t set the root password yet, the password will be blank, so you should just press enter here.  **Enter current password for root (enter for none):** |
| Enter the password for the root account. However, MariaDB's root account has shell authentication set by default, so if you run it with the root account, you can move on without entering a password. |

3. Set the password for the root account to use when accessing MariaDB.

|  |
| --- |
| Setting the root password ensures that nobody can log into the MariaDB  root user without the proper authorisation.  **Set root password? [Y/n]**  New password:  Re-enter new password: |
| Enter n for the password used in 2 or Y for the other password.  If Y is entered, the password is entered and the password re-entered process is followed. |

4. Set whether or not to delete anonymous users.

|  |
| --- |
| By default, a MariaDB installation has an anonymous user, allowing anyone to log into MariaDB without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.  **Remove anonymous users? [Y/n]** |
| Enter Y if you want to delete anonymous users or n if you want to delete an anonymous user. |

5. Set whether or not to grant remote access using the root account.

|  |
| --- |
| Nomally, root should only be allowed to connect from ‘localhost’. This ensures that someone cannot guess at the root pass-word from the network.  **Disallow root login remotely? [Y/n]** |
| Enter Y for remote access to the root account or n for non-approval. |

6. Set whether you want to delete the test database.

|  |
| --- |
| By default, MariaDB comes with a database named ‘test’ that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.  **Remove test database and access to it? [Y/n]** |
| Maria DB basically provides a test database, so you go through the process of making these settings.  Enter Y if you want to delete the test database or n if you do not want to delete it. |

7. Apply those set from 2 to 6.

|  |
| --- |
| **Reload privilege tables now? [Y/n]** |
| Enter Y if you want to apply the settings of the permission table you have created so far, or n if not. |

**Appendix 2. Table Generated Query Statement**

**TWITTER.S\_HASHTAG**

CREATE TABLE S\_HASHTAG( HCODE CHAR(4) NOT NULL, START INT NOT NULL, END INT NOT NULL, TEXT TEXT NOT NULL);

**TWITTER.S\_USER**

CREATE TABLE S\_USER( UCODE CHAR(4) NOT NULL, ID VARCHAR(20) NOT NULL, NAME VARCHAR(50) NOT NULL, SCREEN\_NAME VARCHAR(50) NOT NULL, LOCATION VARCHAR(20), DESCRIPTION TEXT);

**TWITTER.S\_JSON**

CREATE TABLE S\_JSON( CREATE\_AT DATETIME NOT NULL, ID VARCHAR(20) NOT NULL, TEXT TEXT NOT NULL, TRUNCATED CHAR(1) NOT NULL, HASHTAG CHAR(4) NOT NULL, USER CHAR(4) NOT NULL, RETWEET\_COUNT INT, FAVORITE\_COUNT INT, LANG VARCHAR(10), PRIMARY KEY(CREATE\_AT, ID));

**TWITTER.T\_HASHTAG**

CREATE TABLE T\_HASHTAG( HCODE CHAR(4) NOT NULL, START INT NOT NULL, END INT NOT NULL, TEXT TEXT NOT NULL);

**TWITTER.T\_USER**

CREATE TABLE T\_USER( UCODE CHAR(4) NOT NULL, ID VARCHAR(20) NOT NULL, NAME VARCHAR(50) NOT NULL, SCREEN\_NAME VARCHAR(50) NOT NULL, LOCATION VARCHAR(20), DESCRIPTION TEXT);

**TWITTER.T\_JSON**

CREATE TABLE T\_JSON( CREATE\_AT DATETIME NOT NULL, ID VARCHAR(20) NOT NULL, TEXT TEXT NOT NULL, TRUNCATED CHAR(1) NOT NULL, HASHTAG CHAR(4) NOT NULL, USER CHAR(4) NOT NULL, RETWEET\_COUNT INT, FAVORITE\_COUNT INT, LANG VARCHAR(10), PRIMARY KEY(CREATE\_AT, ID));

**TWITTER.KEYWORD\_COUNT**

CREATE TABLE KEYWORD\_COUNT( KEYWORD VARCHAR(100) NOT NULL, COUNT INT NOT NULL, PRIMARY KEY(KEYWORD));

**TWITTER.HASHTAG\_COUNT**

CREATE TABLE HASHTAG\_COUNT( HASHTAG VARCHAR(100) NOT NULL, COUNT INT NOT NULL, PRIMARY KEY(HASHTAG));

**Appendix 3. References and Sites**

https://www.ubuntu.com/

https://developer.twitter.com

http://www.tweepy.org

twetrend, http://tweetrend.com/

foller.me beta, https://foller.me/

http://www.python.org/

https://mariadb.com/kb/ko/mariadb

http://www.eclipse.org/

https://openjdk.java.net/

http://hadoop.apache.org/

National Election Commission, http://www.nec.go.kr/portal/main.do

Jung Jae Hwa, start! Hadoop Programming Hadoop Foundation for Big Data Analysis to YARN, 2016.05.13

Appendix **4. Github**

Github URL: [**https://github.com/SeokJune/BigData\_VI\_T-SA/**](https://github.com/SeokJune/BigData_VI_T-SA/)

