**최종 보고서**

****

**T-SA:** **트위터 단어 검색 기반의 트윗 분석**

(19대 대통령 선거)



학 기: 2019-1 학기

과 목: 산학캡스톤디자인

담당교수: 정 현 숙 교수님

팀 명: VI

팀 장: 이석준(20165072)

팀 원: 이윤혁(20165062)

배인규(20165073)

서재익(20144773)

**- 차 례 -**

**제1장. 서론**

**제1절. 개발 동기 및 필요성**

**제2절. 기존 연구**

**제3절. 개발 목표**

**제4절. 시스템 명세**

**1.4.1. T-SA 시스템 동작 과정**

**제2장. 본론**

**제1절. 개발 환경**

**2.1.1. 우분투**

**2.1.2. 파이썬**

**2.1.3. 마리아 디비**

**2.1.4. 이클립스, 공개 자바 개발 키트**

**2.1.5. 하둡 에코 시스템**

**제2절. 트위터 API**

**제3절. T-SA 흐름도**

**제4절. 구현(파이썬)**

**2.4.1. T-SA.py**

**2.4.2. TwitterAPI.py**

**2.4.3. DBModule.py**

**2.4.4. Analysis\_Visual.py**

**제5절. 데이터베이스 및 테이블 정의서**

**제6절. 구현(하둡)**

**2.6.1. 스쿱 (데이터베이스와 하둡 분산 파일 시스템 간의 연동)**

**2.6.2. 맵퍼 및 리듀스(java 파일)**

**2.6.3. 하둡 실행(jar 파일)**

**제3장. 결론**

**제1절. 실행 환경**

**제2절. 트위터 API 파라미트 설정 값**

**제3절. 분석 및 시각화**

**제4절. 최종 결론**

**제5절. 느낀점**

**제4장. 부록**

**부록 1. 마리아 디비 관련 설정**

**부록 2. 테이블 생성 쿼리문**

**부록 3. 참고문헌 및 사이트**

**부록 4. 깃허브**

**제1절. 개발 동기 및 필요성**

**제1장. 서론**

트위터는 트윗에 자신의 의견(생각)을 표출할 수 있는 SNS(Social Network Service)이다. 지금은 페이스북, 인스타그램보다 사용률이 많이 낮아졌지만, 트윗 작성의 편리함과 정보의 빠른 확산성 때문에 사회적 영향은 무시할 수 없다. 계정 생성 시 메일 주소만을 필요로하고, 형식상 휴대폰 인증과정을 거치지만 다른 사용자에게는 비공개되기 때문에 익명성은 다른 SNS에 비해 강한 편에 속한다. 따라서 자신의 의견(생각)을 자신 있게 표현하는 경우가 많아서 사회현상에 대한 다양한 의견(생각)이 표출된 트윗이 많이 생성된다. 트위터의 게시문인 트윗은 한글이든 영문이든, 공백과 기호를 포함해 한 번에 140글자를 작성할 수 있다. 이러한 특징 때문에 다른 사용자의 의견(생각)이 표현된 트윗을 보는 데 필요한 시간이 적게 소요되며, 실시간 이슈가 되는 주제에 대한 내용을 트위터 검색을 통해 다른 SNS(페이스북, 인스타그램)에 비해 빠르고 다양한 결과를 찾아볼 수 있다.

2008년과 2012년 미국 대통령 선거에서 오바마 캠프가 전 세계적인 주목을 받았었다. 오바마 캠프에서는 출마 선언부터 SNS(Social Networking Sites OR Social Network Services)에 공개하였으며, 주요 지지층을 결집하고 정치적 우군으로 동원하는 과정에도 SNS를 활용하였다. 그래서 2008년과 2012년 대통령 선거에서 버락 오바마가 뛰어난 소셜 미디어 운영 방식을 선보이고, 활용하여 긍정적인 결과를 보였기 때문에 버락 오바마는 소셜 미디어 대통령이라는 별명을 가지게 되었다. 또한 버락 오바마 대통령은 선거 캠페인 도중에 '소셜 미디어는 후보들이 저널리즘의 필터 없이 이야기 할 수 있는 플랫폼'이라고 언급도 했다. 그 이후, 2016년 미국 대통령 선거에는 힐러리 클린턴(민주당)과 도널드 트럼프(공화당)가 출마를 하였다. 이때 도날드 트럼프는 1130만 명의 트위터 팔로워와 1056만 명의 페이스북 팔로워를 보유하고 있었고 이는 1000만 명의 트위터 팔로워를 보유하고 있던 전 대통령인 버락 오바마를 훨씬 뛰어넘는 수치였다. 이러한 많은 수의 SNS 팔로워를 통해 도날드 트럼프는 공화당 후보 경쟁 기간 내내 각종 정치 현안과 관련된 메시지를 SNS에 쏟아내며 화제를 몰고 다녔다. 특히 자신의 공약 홍보나 치적을 보여주기보단 상대 후보(힐러리 클린턴)를 노골적으로 비난하고 불법 이민, 인종차별 등의 민감한 문제에 대해 극단적인 발언을 서슴지 않고 내뱉으며 미국 정치권을 충격에 빠트리기도 했었다. 하지만 SNS 속 지지자들은 이러한 도날드 트럼프의 극단적인 발언에 열광했고 그 결과 2016년 대통령 선거에서 당선되었다.

우리나라는 2002년에 대통령 선거에서 인터넷을 적극적으로 활용한 경험을 통해, 2011년 서울 시장 보궐 선거에서 본격적으로 소셜 미디어가 어떻게 선거에 활용될 수 있는가를 보여주었다. 특히 각 후보의 네트워크와 메시지의 전파, 소셜 미디어에서의 투표 참여 독려 등 여러 측면에서 소셜 미디어의 역활에 대한 깊이 있는 분석이 등장하였다.

앞에 나온 미국 대통령 선거(2008년, 2012년, 2016년)와 우리나라의 서울 시장 보궐 선거(2011년)에서 SNS를 이용한 사례를 통해 SNS와 우리나라의 대통령 선거와의 관계성이 있을 수 있다고 예상된다. 그래서 우리는 이번 프로젝트에서 SNS 중 위에 작성한 것처럼 짧은 글로 작성되는 트위터와 우리나라의 대통령 선거 중 19대 대통령 선거를 선택하여 이둘의 관계성을 분석할 예정이다.

**제2절. 기존 연구**

와이즈넛에서 제공하는 트윗 분석 사이트인 트윗 트렌드(http://tweetrend.com)는 약 400만명의 트위터 계정에 대해 과거 2011년 1월 이후의 데이터부터 하루 1천만 트윗을 수집하여 국내 최대의 검색 서비스를 제공하고 있다. 대표적인 서비스로는 리트윗, URL 파싱, 파급력 등 다각적인 분석 결과를 제공한다. 대표적인 예로는 마케팅 등 다양한 분야에서 활용할 수 있도록 원문을 내려받아 볼 수 있는 서비스가 있다. 이를 이용할 수 있는 방법으로는 3가지(비로그인, 무료회원, 유료회원)로 나누어진다. 각각의 방법의 사용 가능한 검색 키워드의 갯수와 검색 기간은 [표 1]과 같다.

**[표 1]** 트윗 트렌드 이용 방법별 검색 권한

|  |  |  |  |
| --- | --- | --- | --- |
|  | 비로그인 | 무료회원 | 유료회원 |
| 검색 키워드 갯수 | 1 | 3 | 3 |
| 검색 기간 | 7일 | 30일 | 무제한 |

콘스탄틴 코브세닌(Konstantin Kovshenin)이 Twitter API를 이용하여 만든 사이트인 팔로우 미(http://foller.me)는 공개된 프로필에 관한 주제, 언급, 해시태그 등 정보를 분석하고 보여준다. 대표적인 서비스로는 요청한 사용자의 프로필과 사용자가 작성한 최신 트윗들을 수집하여, 내용을 분석하고 자주 사용되는 단어(해시태그)들을 출력하여 보여준다.

중앙선거관리위원회의 선거통계 시스템사이트에서는 과거 실시하였던 선거의 주요일정을 시간순으로 확인할 수 있으며, 선거의 (예비)후보자들을 한눈에 확인할 수 있고 (예비)후보자별 상세 정보를 확인할 수 있다. 또한, 선거 당일 투표 마감 후 전체, 지역별 개표율과 후보자별 득표수, 득표율 등을 확인할 수 있다. 선거통계 시스템사이트에서 제공하는 통계정보는 누구나 무료로 이용할 수 있고 이를 다른 정보와 함께 제공하는 경우에는 내려받을 각 자료의 출처를 명확히 명시하여 정보이용자의 오해가 발생하지 않도록 하는 성에서 정보 활용이 가능하다. 그래서 우리가 이번 프로젝트에서 사용할 19대 대통령 선거의 결과를 가져오기 위해 역대선거에서 개표현황을 통해 데이터를 불러올 수 있으며, 이를 PDF나 EXCEL 파일로 내려받을 예정이다.

**제3절. 개발 목표**

기업과 기관에서는 내부의 데이터뿐만 아니라 페이스북, 트위터, 카카오톡 같은 SNS를 통해 데이터를 수집하고 분석하여 그동안 파악하지 못했던 다양한 결과를 얻고자 시도하고 있다. 그러한 이유로는 SNS의 한가지인 트위터를 확인해보면 알 수 있다. 트위터는 전 세계 1억명이 넘는 이용자가 하루 평균 2억개 이상의 트윗이 작성되기 때문이다. 특히 2012년 대통령 선거 당시 174만개의 트윗을 분석하여 민심의 흐름을 추적하였고, 2014년에는 특정 IT 박람회와 관련된 트윗을 분석하여 정보교환의 사회연결망을 분석하고자 하였다. 본 프로젝트에서는 19대 대통령선거와 관련된 키워드(19대 대통령 선거의 후보 상위 5개)와 기간(2017.04.18.~2017.05.09.)을 설정하여 해당 기간 동안 작성된 트윗을 가져와서 포함된 단어와 해시태그의 빈도수를 구하고, 이를 중앙 선거관리위원회의 선거통계 시스템을 통해 얻은 결과와 어떠한 상관관계를 갖는지 분석하고자 한다.

첫 번째로는 대통령 선거기간 동안 특정 키워드가 포함된 트윗을 수집하기 위해서 트위터에서 제공하는 Twitter API를 사용한다. Twitter API를 통해 키워드 질의를 전송하고 질의에 응답하는 결과는 JSON 형식을 제공한다. 이를 이용하기 위해서는 아이디와 비밀번호가 아닌 개발자 등록을 해야 되고 개발자 등록을 한 사람들은 Consumer Key를 발급받아야 한다. 트윗 데이터를 수집할 시에는 무료 버전인 Standard 버전을 이용할 수 있다. 이때 가져올 수 있는 트윗의 개수는 100개로 제한되며 한 달 동안 이용할 수 있는 질의의 횟수는 50으로 제한된다. 더 많은 데이터를 수집하려고 할시에는 결제를 통해 Standard가 아닌 다른 버전으로 업그레이드하면 된다.

두 번째로는 불필요한 정보 수집을 방지하기 위해 19대 대통령 선거와 관련된 키워드를 선거 기간인 2017년 4월 15일부터 2017년 5월 9일까지(22일)의 트윗을 통해 색출할 것이다. 이를 통해 좀 더 다양한 사람들의 트윗을 수집할 수 있으며, 대통령 선거와 트윗의 정확도를 높일 수 있게 된다. 또한 트윗 내용에 대한 중복이 있을 경우 제거를 하고 수집할 것이다. 해당 트윗들은 대한민국의 19대 대통령 선거 기간에 이루어진 트윗이기 때문에 한국어를 사용한다. 한국어로 작성된 트윗은 주어, 서술어, 목적어 등이 빈번히 생략되며, 일부 사용자들은 일부러 띄어쓰기를 안 한 거나 여러 번 하는 경우가 있다. 그렇기 때문에 자연어 처리를 통해 고유명사와 일반 명사로 분류하여 키워드의 빈도수를 구하여, 워드 클라우드 툴을 통해 상위 키워드의 빈도수 등 다양한 시각화를 제공하고자 한다.

마지막으로는 중앙선거관리위원회의 선거통계 시스템에서 제공해주는 19대 대통령 선거 후보자별 득표율과 수집된 트윗들에서 후보자별 언급 횟수를 토대로 비교 분석하고 시각화 툴을 이용하여 더욱 객관적으로 표현하고자 한다.

**제4절. 시스템 명세**

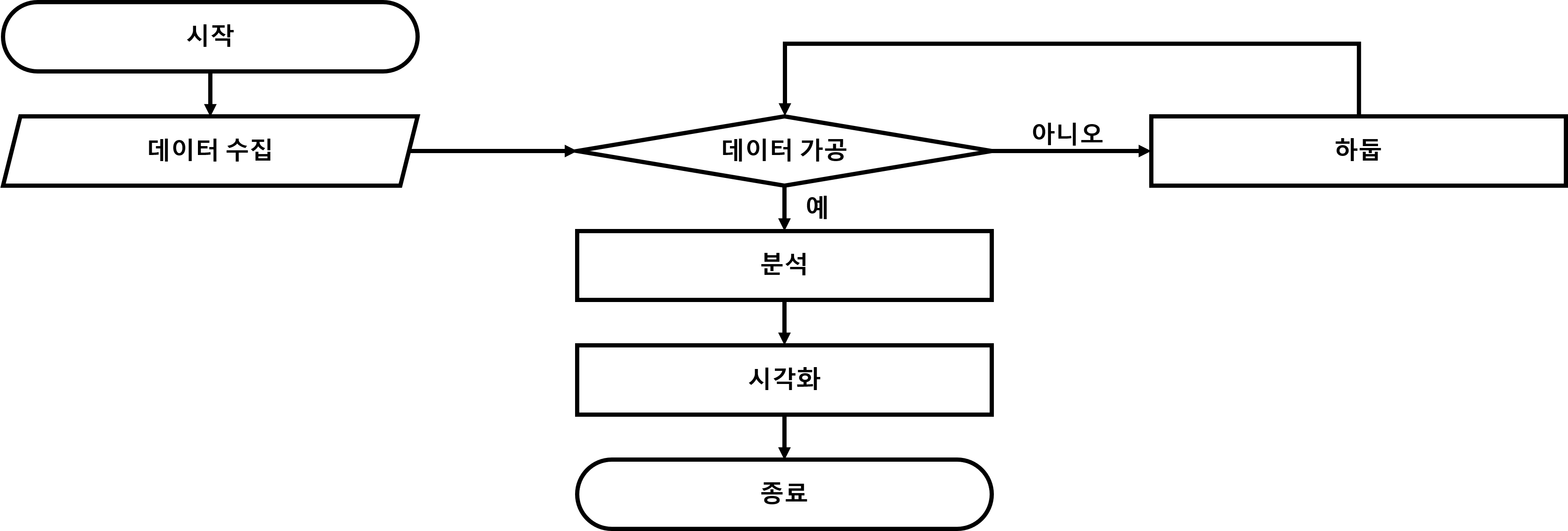
**1.4.1. T-SA 시스템 동작 과정**

T-SA의 시스템 동작 과정은 아래와 같고 이에 대한 순서도는 [그림1]을 참고하면 된다.

1. 데이터(정형 - 중앙선거관리 위원회의 선거통계 시스템 자료, 비정형 - 트위터의 트윗 정보)를 수집한다.

2. 분석 및 시각화를 위해 수집한 데이터를 가공해야할 필요가 있는지 파악한다. 이때 가공해야 된다면 하둡(Hadoop)을 이용하여 데이터를 가공한다.

3. 데이터를 가지고 비교 분석을 하고 이를 시각화하여 표현한다.



**[그림 1]** T-SA 시스템 동작 과정 순서도

**제2장. 본론**

**제1절. 개발 환경**

T-SA의 간략한 개발 환경(버전 정보)은 [표 2]와 같다.

**[표 2]** T-SA의 개발 환경

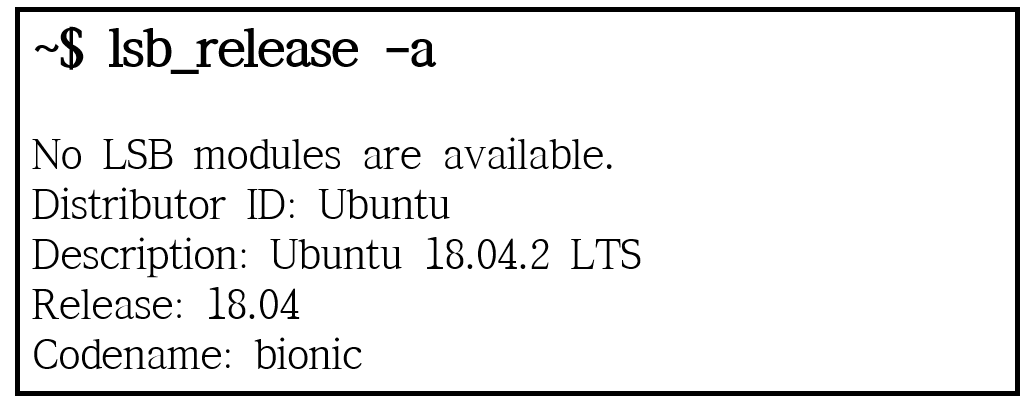
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 우분투 | | | 18.04.2 LTS | |
| 파이썬 | | | 3.6 | |
| 마리아 디비 | | | 10.1.38 | |
| 이클립스 | | | 2019-03(4.11) | |
| 공개 자바 개발 키트 | | | 1.8.0\_191 | |
| 하둡 에코 시스템 | 하둡 | 3.2.0 | |
| 스쿱 | 1.4.7 | |

**2.1.1. 우분투**

우분투(Ubuntu)는 컴퓨터에서 프로그램과 주변기기를 사용할 수 있도록 해주는 운영체제(Operating System: OS) 중 하나이며, 캐노니컬(Canonical)사의 지원을 받아 무료로 배포되며 무료로 사용할 수 있다. 우분투라는 이름은 ‘네가 있기에 내가 있다’는 타인을 향한 인류애를 뜻하는 반투어이다. 반투어는 짐바브웨나 르완다 등 아프리카 중부지역에서 사용하는 말이며, 우분투는 OS로서 이름 속에 담긴 철학을 실천하고 있다. 전 세계의 수많은 사용자가 우분투의 기능을 개선하거나 각국 언어로 번역하는 데 기여하고 있다. 우분투 사용자들은 같은 OS를 사용한다는 공감대 아래 공동체(community)를 이루고 우분투는 강제성이 없는 자발적인 개인들이 모인 이 공동체를 통해 급속도로 성장하고 있다.

데비안 계열 배포판이란 한편 마이크로소프트 윈도우(MS윈도우)에도 XP나 비스타 같은 종류가 있듯이 리눅스에는 400종에 가까운 배포판이 있다. 배포판은 영어로 ‘디스트리뷰션(Distribution)’ 또는 ‘디스트로(Distro)’라 불리며, 우분투도 이런 배포판 중 하나입니다. 각 배포판은 성능이나 장단점과 사용법 차이가 있지만 ‘커널(Kernel)’이라고 부르는 핵심 부분은 모두 리눅스라는 공통점을 갖고 있다. 우분투를 설치하면 업무에 필요한 프로그램들과 컴퓨터 관리프로그램, 간단한 게임들이 모두 컴퓨터에 설치되며, 초보자 눈높이에 맞게 구성되어 있어 사용이 쉬운 운영체제이다. 우분투는 광범위한 분야에 걸쳐 여러 사람의 노력으로 시기별로 개선되는 프로그램을 무료로 제공하고 있다. 캐노니컬사는 유료화 계획은 없는 상태이며 앞으로도 없을 것이라고 한다. 따라서 우분투 사용은 불법적인 프로그램 사용을 피할 수 있다.

우분투의 버전 확인 방법 및 결과는 아래 [그림 2]와 같다.



[그림 2] 우분투의 버전 확인 방법 및 결과

**2.1.2. 파이썬**

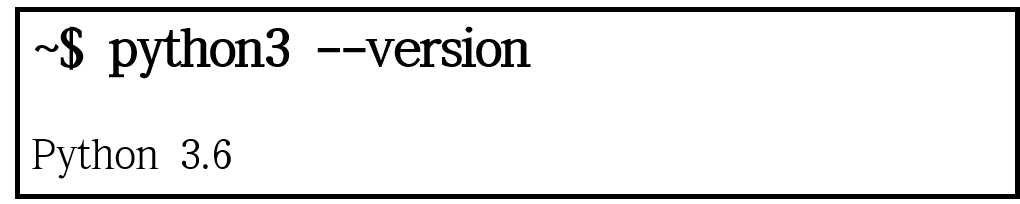
파이썬(Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터실, 객체 지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다.

파이썬3000(혹은 파이썬3k)이라는 코드명을 지닌 파이썬의 3.0버전의 최종판이 긴 테스트를 거쳐 2008년 12월 3일 자로 발표되었다. 2.X버전의 파이썬과 하위호환성이 없다는 것이 가장 큰 특징이다. 파이썬 3의 주요 기능 다수가 이전 버전과 호환되게 2.6과 2.7 버전에도 반영되기도 하였다.

파이썬 공식 문서에서는 "파이썬 2.X버전은 레거시(오래된 기술)이고, 파이썬 3.X버전이 파이썬의 현재와 미래가 될 것"이라고 요약을 했는데, 처음 배우는 프로그래머들은 파이썬 3으로 시작하는 것을 권장하고 있다.

파이썬 2.X버전과의 차이점으로는 사전형과 문자열형과 같은 내장자료형의 내부적인 변화 및 일부 구형의 구성 요소가 제거되었으며, 표준 라이브러리를 재배치하고, 향상된 유니코드를 지원한다. 그렇기 때문에 한글 변수를 지원한다.

파이썬3은 터미널에 sudo apt-get install python3을 입력하면 간단하게 설치할 수 있고 버전 확인 방법 및 결과는 [그림 3]과 같다.



**[그림 3]** 파이썬3 버전의 버전 확인 방법 및 결과

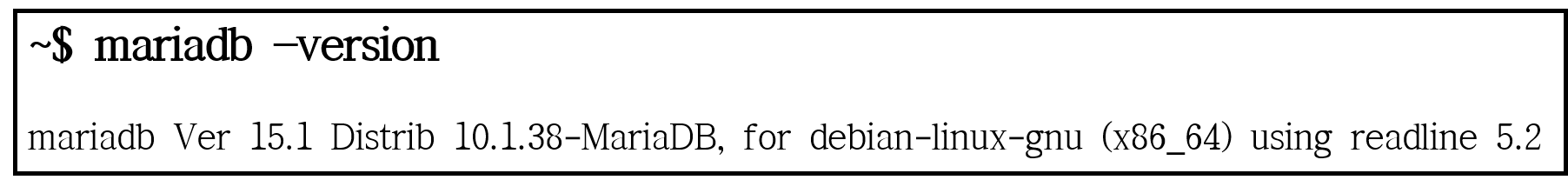
**2.1.3. 마리아 디비**

마리아디비(MariaDB)는 MySQL의 발전된 형태의 대체재로써, https://downloads.mariadb.org/에서 다운로드받을 수 있으며, GPL v2 라이선스로 유지되고 있고, MariaDB 커뮤니티와 MariaDB 재단이 주축이 되어 개발되고 있다.

보안은 현재의 세계에서 매우 중요하며 MariaDB 개발자들의 주의를 기울이고 있다. 이 프로젝트는 MySQL 프로젝트 기반으로 자체적인 보안 패치를 유지하고 있다. 각 MariaDB 릴리즈에 대해 개발자는 MySQL 보안 패치를 merge 할 수 있으며 필요한 경우 개선할 수도 있다. 중요한 보안 이슈가 발견되면, 개발자들은 즉시 이를 해결하는 새로운 MariaDB 릴리즈를 개발, 배포한다. MySQL에서 발견된 많은 보안 이슈는 MariaDB에서도 발견되어 왔으며 MariaDB 팀에 보고되어 왔다. MariaDB 팀은 모든 보안 이슈들이 즉각 보고되고 충분히 자세하게 해결될 수 있도록 http://cve.mitre.org/ 와 긴밀히 협력하고 있다. 자세한 보안 이슈들은 일반적으로 이슈가 해결된 MariaDB 와 MySQL 버전이 배포된 이후에 릴리즈된다.

MariaDB는 현재까지 최신의 MYSQL과 같은 브랜치로부터 릴리즈되며, 대개의 경우 MySQL과 마찬가지로 동작한다. MySQL의 모든 명령어, 인터페이스, 라이브러리와 API가 MariaDB에도 존재한다. 또한 MariaDB로 데이터베이스를 변환할 필요도 없다. 즉, MariaDB는 사실상 MySQL의 완벽한 대체재라고 말할 수 있다.

MariaDB는 터미널에 sudo apt-get install mariadb-server를 입력하면 간단하게 설치할 수 있다. 이는 설치 날짜 기준(2019.03.30)으로 mariaDB의 10.1.38버전과 의존성 패키지가 설치된다. 그 후 sudo mysql\_secure\_installation을 입력한 후 MariaDB의 권한 테이블 설정을 해주어야 하는데 그 과정은 아래 [부록 1]을 참고하면 된다. 그리고 버전 확인 방법 및 결과는 [그림 4-2]와 같다.



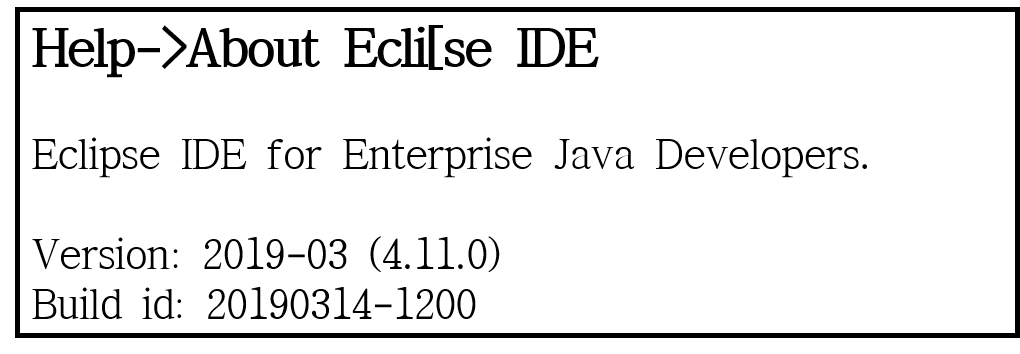
**[그림 4]** 마리아 디비의 버전 확인 방법 및 결과

**2.1.4. 이클립스, 공개 자바 개발 키트**

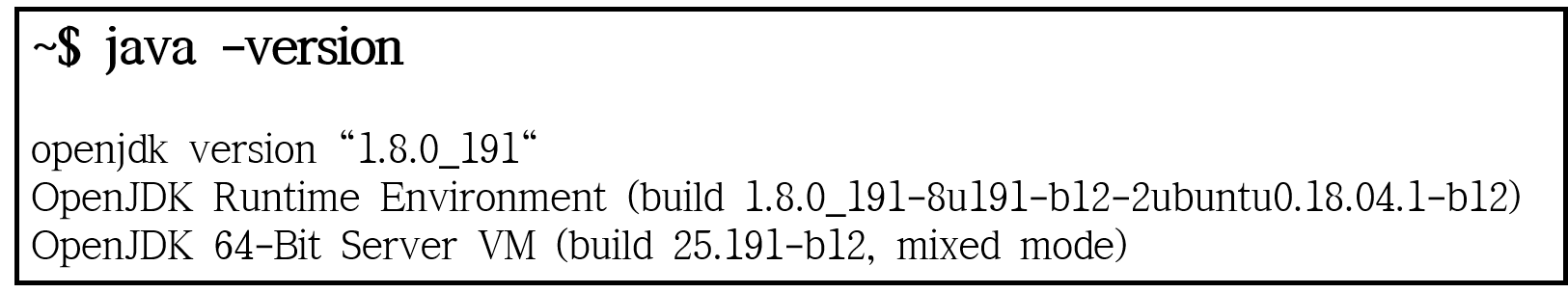
이클립스(Eclipse)는 다양한 플랫폼에서 쓸 수 있으며, 자바를 비롯한 다양한 언어를 지원하는 프로그래밍 통합 개발 환경을 목적으로 시작하였으나, 현재는 OSGi(Open Service Gateway initiative)를 도입하여, 범용 응용 소프트웨어 플랫폼으로 진화하였다. 원래 IBM의 웹스피어 스튜디오 애플리케이션 디밸로퍼(WebSpheare Studio Application Devel-oper)란 이름으로 개발되었던 것인데, 엔진 부분을 오픈소스로 공개한 것을 기반으로 지금의 이클립스로 발전해 왔다.

공개 자바 개발 키트(OpenJDK)는 Java SE(Standard Edition) 기반의 오픈 소스 JDK다. 2006년 Sun Micro System은 Java를 오픈 소스화한다고 발표하였다. 그리고 그해 11월 HotSpot VM과 컴파일러를 GNU General Public License(이하 GPL)로 풀었다. 현재 유수의 IT 기업들이 프로젝트에 참여하고 있는데, IBM은 2010년 10월 기존에 참여하던 Apache Harmony 프로젝트로부터 OpenJDK 프로젝트에 참여하기로 결정하였다. 곧이어 2010년 말에는 Apple이, 2011년 중반에는 SAP가 OpenJDK에 협력하기로 하였다.

이클립스의 경우 www.eclipse.org 에 접속하여 Eclipse IDE 2019-03중 Linux를 설치하는데 이때 자신의 환경을 확인하고 x86, x64 둘 중 하나를 설치하면 되고 버전 확인 방법 및 결과는 [그림 5]와 같다. 공개 자바 개발 키트의 경우 터미널에 sudo apt-get install openjdk-8-jdk를 입력하면 간단하게 설치할 수 있고 버전 확인 방법 및 결과는 [그림 6]과 같다.



**[그림 5]** 이클립스의 버전 확인 방법 및 결과



**[그림 6]** 공개 자바 개발 키트의 버전 확인 방법 및 결과

**2.1.5. 하둡 에코 시스템**

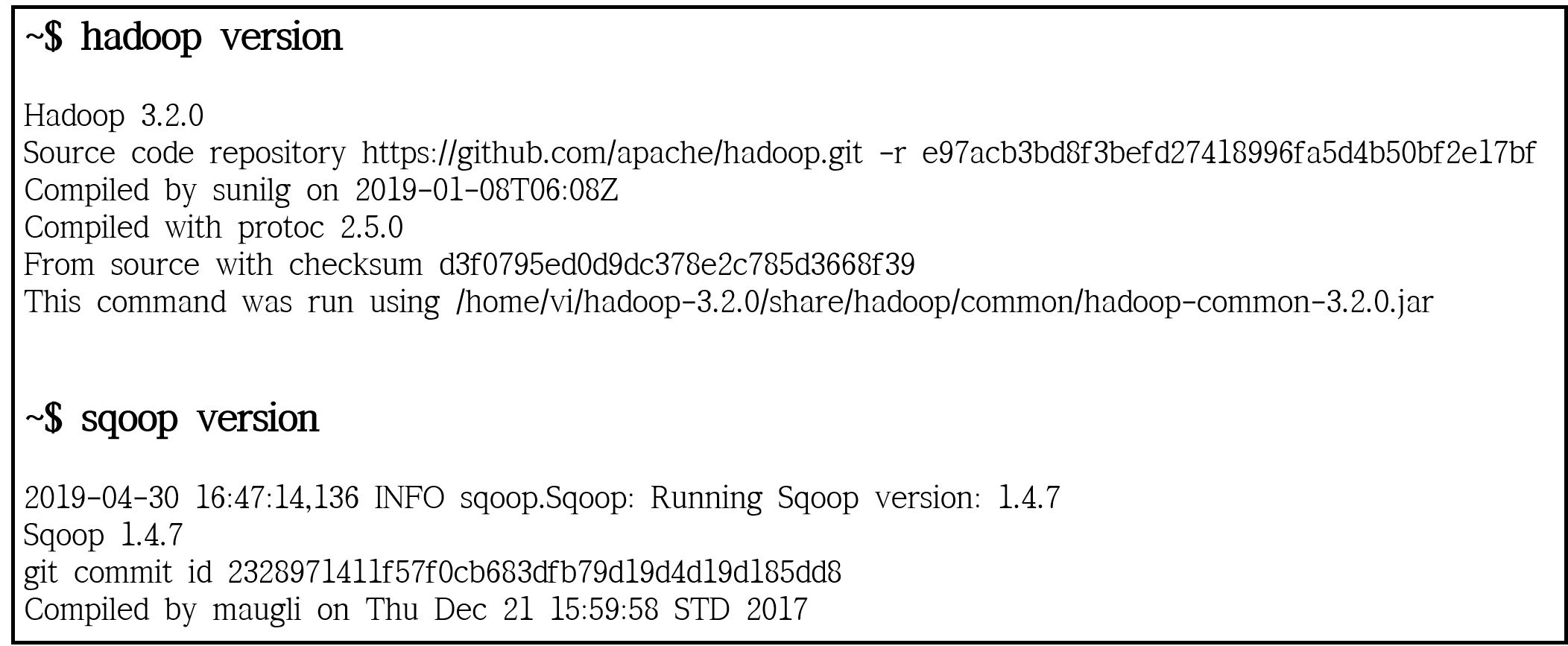
아파치 하둡(Apache, High-Availability Distributed Object-Oriented Platform)은 대량의 자료를 처리할 수 있는 큰 컴퓨터 클러스터에서 동작하는 분산 응용 프로그램을 지원하는 프리웨어 자바 소프트웨어 프레임워크이다. 원래 너치의 분산 처리를 지원하기 위해 개발된 것으로, 아파치 루씬의 하부 프로젝트이다. 분산처리 시스템인 구글 파일 시스템을 대체할 수 있는 하둡 분산 파일 시스템(HDFS: Hadoop Distributed File System)과 맵리듀스를 구현한 것이다.

하둡 분산 파일 시스템(HDFS, Hadoop distributed file system)은 하둡 프레임워크를 위해 자바 언어로 작성된 분산 확장 파일 시스템이다. HDFS는 여러 기계에 대용량 파일들을 나눠서 저장한다. 데이터들을 여러 서버에 중복해서 저장함으로써 데이터 안정성을 얻는다.

맵리듀스(Map/Reduce)는 구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 목적으로 제작하여 2004년 발표한 소프트웨어 프레임워크다. 이 프레임워크는 페타바이트 이상의 대용량 데이터를 신뢰도가 낮은 컴퓨터로 구성된 클러스터 환경에서 병렬 처리를 지원하기 위해서 개발되었다. 이 프레임워크는 함수형 프로그래밍에서 일반적으로 사용되는 Map과 Reduce라는 함수 기반으로 주로 구성된다. 현재 Map/Reduce는 Java와 C++, 그리고 기타 언어에서 적용이 가능하도록 작성되었다. 대표적으로 아파치 하둡에서 오픈 소스 소프트웨어로 적용되었다.

스쿱(Sqoop)은 구조화된 관계형 데이터베이스와 아파치 하둡 간의 대용량 데이터들을 효율적으로 변환하여 주는 CLI(Command-Line Interface) 애플리케이션이다. 오라클 또는 MySQL 같은 관계형 데이터베이스에서 하둡 분산 파일 시스템으로 데이터들을 가져와서 그 데이터들을 하둡 맵리듀스로 변환을 하고, 그 변환된 데이터들을 다시 관계형 데이터베이스로 내보낼 수 있다. 스쿱은 데이터의 가져오기와 내보내기를 맵리듀스를 통해 처리하여 장애 허용 능력뿐만 아니라 병렬 처리가 가능하게 한다.

하둡과 스쿱의 버전 확인 방법 및 결과는 [그림 7]과 같다.



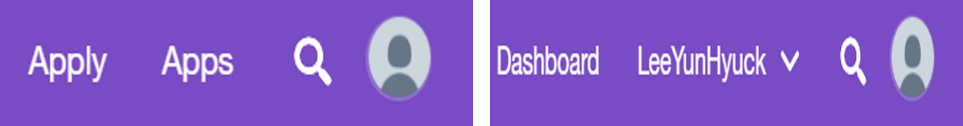
**[그림 7]** 하둡과 스쿱의 버전 확인 방법 및 결과

**제2절. 트위터 API**

트위터에서는 최대한 널리 정보를 공유할 수 있도록 기업과 개발자 및 사용자가 트위터 API(애플리케이션 프로그래밍 인터페이스)를 통해 프로그래밍 방식으로 트위터 데이터에 액세스할 수 있도록 지원하고 있다. 트위터 API를 사용하기 위해서는 기본적으로 트위터에 가입이 된 상태에서 진행한다. 가입이 완료되어야 개발자 등록할 수 있으며, 개발자 등록을 하지 않으면 키를 발급받지 못한다. 키를 발급받으면, 개발 환경 설정을 통해 사용되는 영역에 따라 자유롭게 이용할 수 있다. 그에 해당하는 과정은 아래와 같은 순서로 진행 된다.

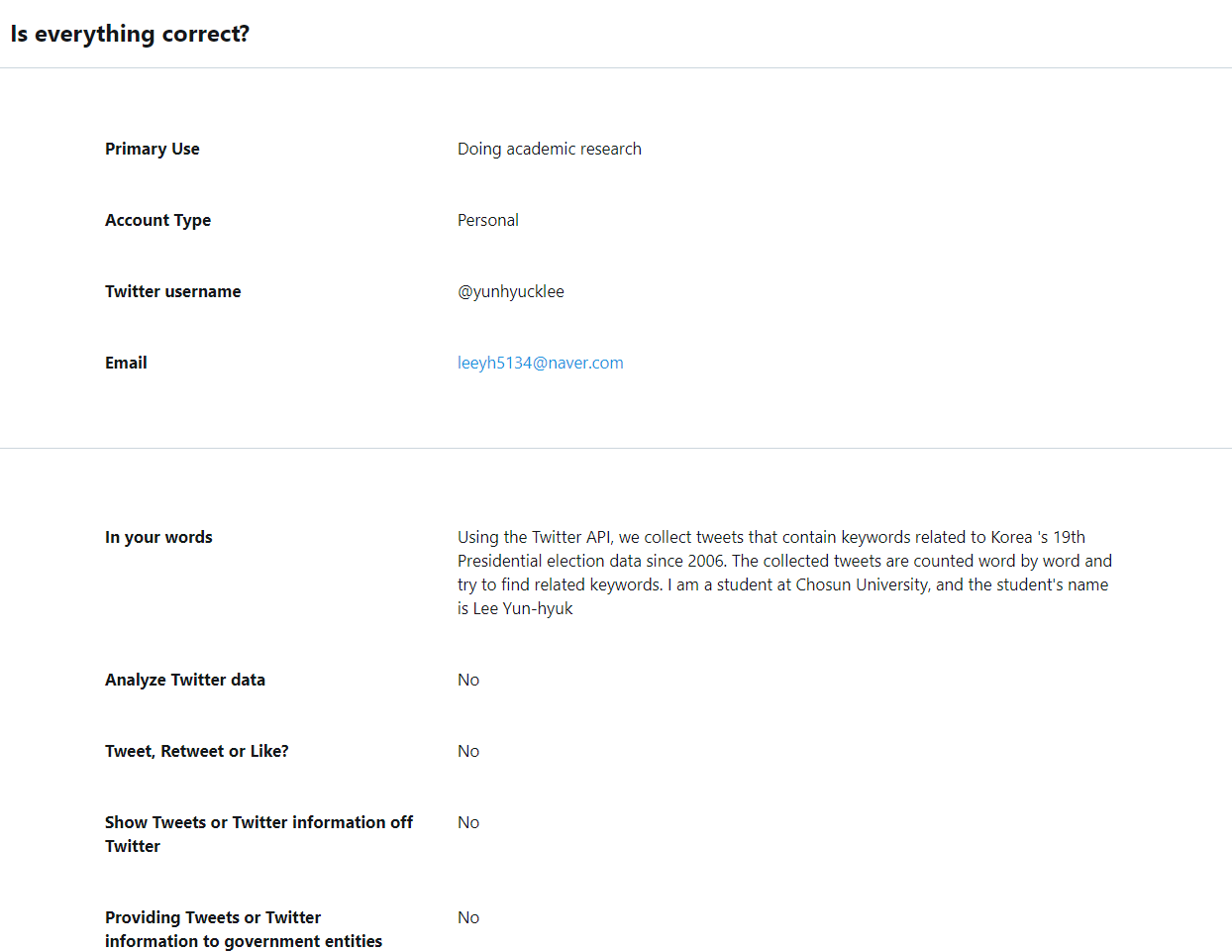
**1. 개발자 등록**

개발자 등록은 https://developer.twitter.com/에 접속하여 등록할 수 있다. 먼저 우측 상단의 Apply를 누른다. 개발자 등록을 하기 전과 후의 웹페이지 우측 상단의 모습은 [그림 8]과 같다.



**[그림 8]** 개발자 등록 전(좌), 개발자 등록 후(우)

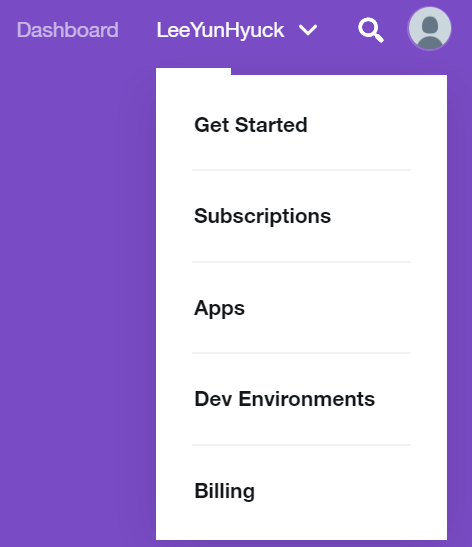
Apply for a developer account를 누르게 되면, 트위터의 개발 도구를 사용하는 이유에 관해 묻는 질문이 현재는 추가가 되었다. 이에 대한 질문은 학술적으로 사용하는 이유로 지정하고 진행하였다. 질문을 선택하게 되면 현재 로그인된 계정을 기반으로 하여, 개발자 등록을 할 건지 묻는 페이지가 활성화된다. 기본적으로 트위터에 핸드폰 인증과 이메일 인증이 없다면, 별다른 설정 없이 사는 지역과 별명을 추가시켜주고 다음 단계로 진행하면 된다. 그런 다음 사용 목적에 묻는 페이지가 나온다. 페이지가 변경되기 전에는 모든 응답을 채워야 했지만, 변경된 후에는 간단하게 사용 목적에 대해 작성만 해주면 된다. 사용 목적은 영어로 200자 이상, 학생인 경우 학교와 이름을 포함하여 작성해 주면 된다. 본 프로젝트에서는 트위터 개발자 등록 시, 사용 목적으로 ‘트위터 API를 이용하여, 2006년 이후의 데이터 중 한국의 19대 대통령 선거와 관련된 키워드들이 포함된 트윗들을 수집한다. 수집한 트윗들은 단어별로 카운트하여 연관된 키워드를 색출하려고 한다. 조선대학교 학생이며, 학생의 이름은 xxx입니다.’라고 작성하였다. 작성이 완료되어 다음 단계로 진행하면, 요청이 맞는지 확인하는 페이지가 [그림 9]과 같이 나오게 된다. 마지막으로 이메일 인증을 진행하게 된 후, 트위터에서 개발자 승인을 통해 사용할 수 있다.



**[그림 9]** 개발자 등록을 위한 정보기재가 완료된 상태

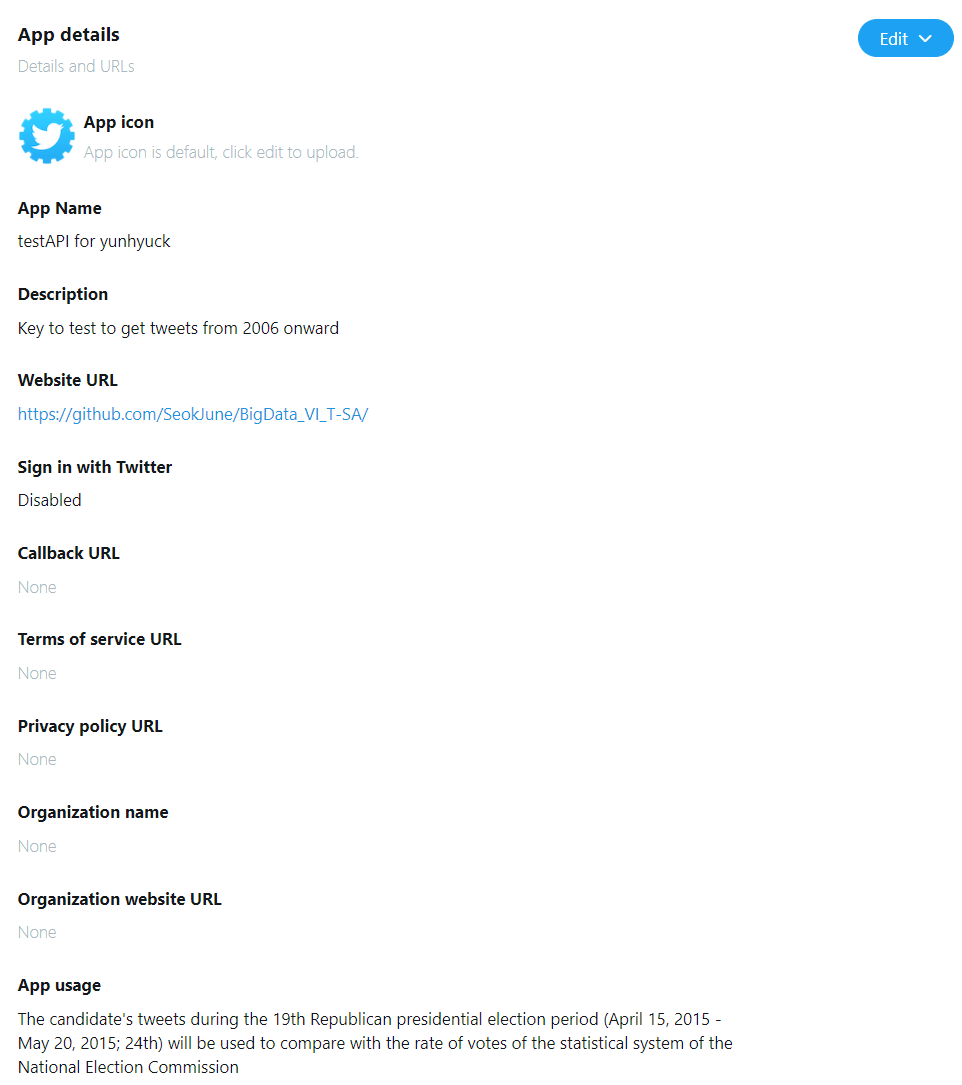
**2. 애플리케이션 등록 및 키 발급**

개발자 등록이 완료된 상태에서 애플리케이션을 등록하여 준다. 이 과정에서는 사용할 앱의 이름과 사용 방법에 대해 기술을 하게 된다. 애플리케이션을 등록을 하기 위해서 [그림 10]과 같이 Apps를 클릭하여 앱을 생성하게 된다.



**[그림 10]** 개발자 등록 후, 사용할 수 있는 메뉴

본 프로젝트에서 사용하는 애플리케이션 이름은 테스트 API 발급이라고 하였다. 테스트를 진행한 후, 새로운 키를 발급받아 제한된 사용량을 해결해 보고자 하였다. 그다음 앱에 대한 설명을 작성해 주면 된다. 이때, 글자 수는 10~200자 사이로 영문으로 작성을 해주면 된다. 본 프로젝트의 경우에는 ‘2006년 이후의 트윗을 가져오기 위해 테스트하는 키’라고 작성하였다. 웹 URL을 작성하는 곳에는 해당 프로젝트가 진행되고 있는 것 허브(https://github.com/SeokJune/BigData\_VI\_T-SA/) 주소를 사용하였다. 마지막으로 앱의 사용 방법은 100자 이상으로 작성을 하면 된다. 본 프로젝트에서는 ‘대한민국의 19대 대통령 선거기간인 2017년 4월 15일부터 2017년 5월 9일까지(22일) 동안의 후보자들 트윗을 가져와 중앙선거관리위원회의 통계 시스템의 득표율과의 비교 분석에 사용된다’라고 작성하였다. 최종 작성된 내용은 [그림 11]와 같다. 생성된 앱의 상세 정보 페이지에서 키를 확인할 수 있다.



**[그림 11]** 생성된 테스트 API 발급을 받기 위한 앱의 정보

**3. 검색 API 분류 및 선택**

개발 환경 설정에 앞서 Twitter API의 트윗 검색을 위한 Search API에 대해 알아보았다. 현재 기본적으로 제공을 해주는 버전은 Standard 버전이다. Premium과 Enterprise 총 3가지로 분류를 할 수 있고 자세한 내용은 [표 3]을 통해 확인할 수 있다.

**[표 3]** 트위터의 Search API의 분류

|  |  |  |
| --- | --- | --- |
| Standard | | 지난 7일간 게시된 최근 트윗을 제공한다.  공개 API 집합의 일부이다. |
| premium | 30-days | 지난 30일간 게시된 트윗을 제공한다. |
| Full-archive | 2006년부터 게시된 트윗을 제공한다. |
| Enterprise | | Premium과 같이 두 가지를 제공하는데 기업에서 주로 사용한다. |

Standard의 질의를 수행했을 때, 가져올 수 있는 기간이 7일로 제한이 되어 있다. 문재인이라는 키워드가 들어가 있는 트윗을 가져온다는 질의를 트위터에 보냈을 때, 지난 7일간에 게시된 트윗 만을 가져오기 때문에, 본 프로젝트에서 진행하는 19대 대통령 선거 기간의 트윗 내용을 가져오기 위해서는 적절하지 못한 것을 확인하였다. Premium의 경우 지난 30일간 게시된 트윗을 제공해 주는 버전과 2006년 이후 게시된 트윗을 제공해 주는 버전으로 나누어져 있다. 가격마다 차이가 있지만, 기본적으로 한 달 동안 몇 번의 질의를 할 수 있는지 횟수를 통해 가격이 책정된다. [그림 12]을 통해 2006년 이후의 데이터를 가져올 수 있는 버전의 차이를 알 수 있다.

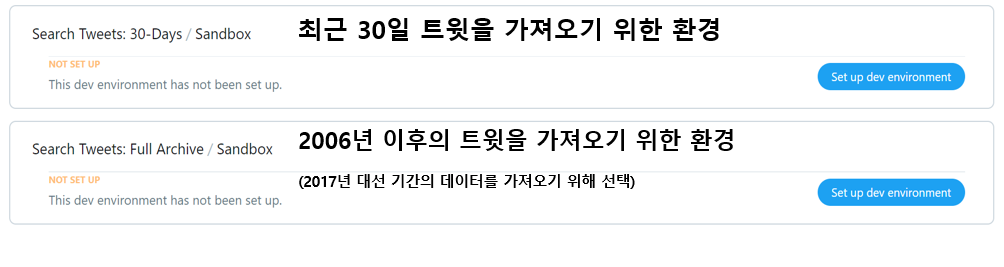


**[그림 12]** Sandbox(Standard)와 Premium의 차이

2006년 이후의 데이터를 가져오는 Search API의 구성에는 Sandbox(Standard) 버전과 Premium 버전으로 나누어져 있다. 두 버전의 공통점은 트위터가 개설되고 난 후의 데이터를 가져올 수 있다는 점이다. 가장 큰 차이점은 기본 제공되는 Sandbox 버전은 128자의 트윗 만을 가져올 수 있다. 가져오는 트윗이 128자보다 많다면 …으로 표시가 되며, 한 달 동안 100번의 질의를 수행할 수 있습니다. 그에 반해 Premium 버전의 경우 기본적으로 500번의 질의를 수행할 수 있으며, 트윗 내용의 1024자를 가져올 수 있다. 그래서 본 프로젝트에서는 Premium을 사용할 것이다.

**4. 개발 환경 설정**

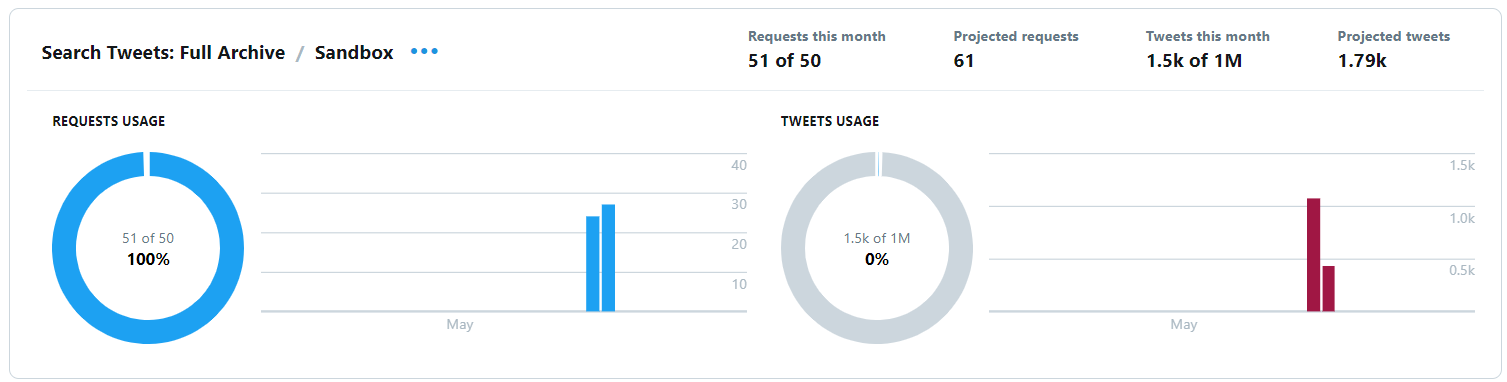
발급받은 키에 대해 API를 사용하는 환경을 설정하는 과정이다. 개발 환경을 설정을 해주어야, OAuth 인증을 통해 질의를 수행한 결과를 가져올 수 있다. 페이스북의 경우 key와 token을 URL을 통해 파라미터로 전송하여 데이터를 가지고 오는 기능을 수행한다. 하지만, 트위터의 경우에는 OAuth(인터넷 사용자들의 비밀번호를 제공하지 않고 다른 웹사이트상의 자신들의 정보에 대한 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 수단)의 기본에 충실하게 액세스 토큰을 가지고 와야 원하는 데이터를 조회할 수 있기 때문이다. [그림 10]과 같이 Dev Environments 카테고리를 통해 환경을 설정할 수 있다. 해당 카테고리를 접속하면 [그림 13]과 같다.



**[그림 13]** Dev Environments 화면

5. 사용량 확인

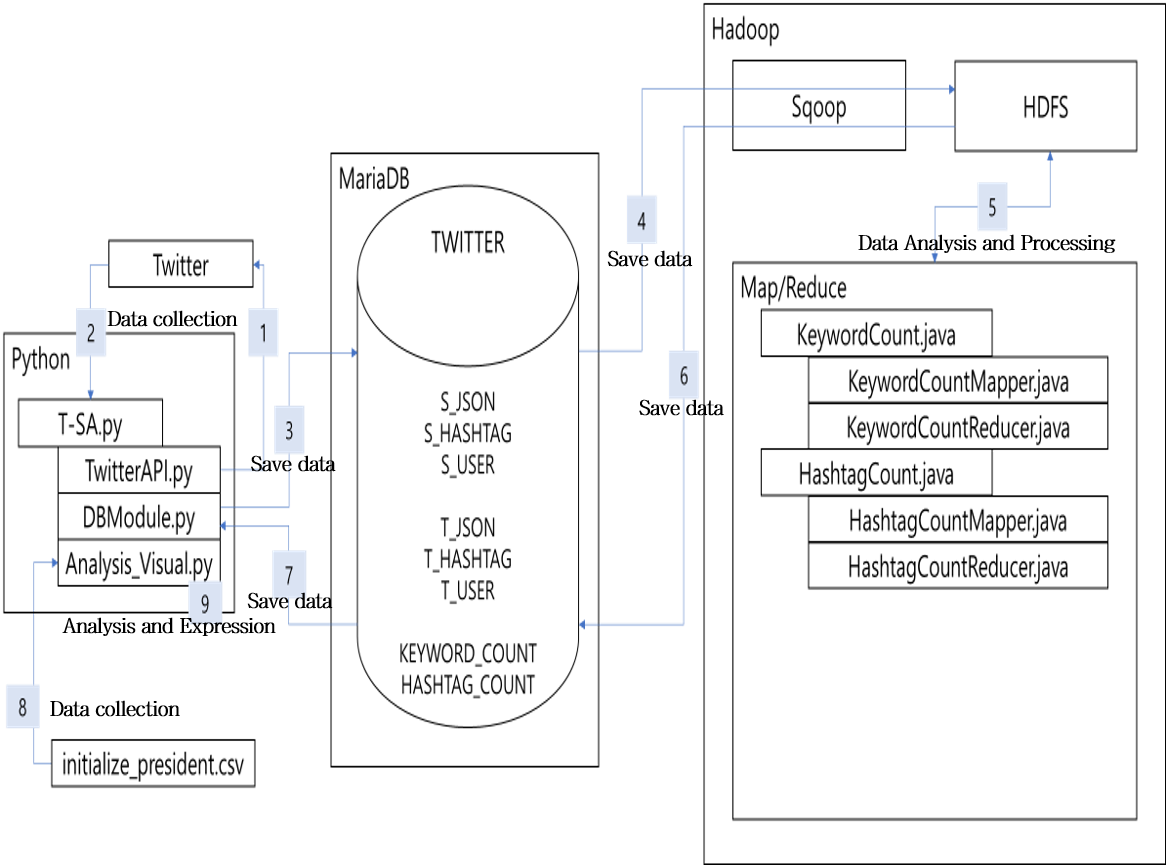
본 프로젝트는 19대 대통령 선거 기간의 데이터를 가져와야 하므로 Full Archive의 개발환경 설정을 하였다. Set up Dev environment를 클릭하여, JSON 형식으로 받을 파일의 label과 사용할 키가 있는 앱의 이름을 선택해 주면 완료된다. 설정이 완료되면 Dashboard를 통해 사용량을 확인할 수 있으며 페이지는 [그림 14]와 같다. SandBox 버전의 경우 한 달에 질의를 수행할 수 있는 횟수가 50번으로 제한이 되어 있기 때문에 주기적으로 확인을 해주어야 한다.



**[그림 14]** Dashboard 화면

**제3절. T-SA 흐름도**

T-SA의 흐름도는 [그림 15] 와 같고 자세한 내용은 아래와 같다.



**[그림 15]** T-SA 흐름도

**1, 2. 데이터(트위터) 수집**

Twitter API를 이용해서 정보(트윗 내용(작성 시간, 트윗, 해시태그 등), 사용자 정보(아이디, 닉네임, 위치정보, 팔로우 수, 팔로잉 수, 언어 등))등을 수집한다.

**3. 데이터 저장(마리아 디비)**

수집된 데이터를 마리아 디비에 저장한다.

**4. 데이터 저장(하둡 분산 파일 시스템)**

스쿱을 이용하여 마리아 디비의 데이터를 하둡 분산 파일 시스템에 저장한다.

**5. 데이터 가공(맵/리듀스)**

하둡 분산 파일 시스템에 저장되어 있는 데이터를 맵/리듀스를 통해 정규화하고 결과를 다시 하둡 분산 파일 시스템에 저장한다. KeywordCount에서는 자연어 처리를 한 후 단어(2~5글자)를 카운트한다. HashtagCount에서는 해시태그(5글자 이하)를 카운트한다.

**6. 데이터 저장(마리아 디비)**

스쿱을 이용하여 하둡 분산 파일 시스템에 저장된 정규화 데이터를 마리아 디비에 저장한다.

**7. 가공된 데이터 수집**

마리아 디비에 저장된 데이터를 파이썬으로 읽어온다.

**8. 데이터(중앙선거관리위원회 자료) 수집**

중앙선거관리위원회에서 제공하는 19대 대통령 선거의 득표수를 수집한다.

**9. 분석 및 표현**

7, 8의 데이터를 분석하고 시각화(워드 클라우드, 선 그래프, 막대 그래프, 원 그래프 등)한다.

**제4절. 구현(파이썬)**

**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/  # -------------------------------------------------------------------------------------------------- |

**제5절. 데이터베이스 및 테이블 정의서**

|  |  |  |
| --- | --- | --- |
| **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** | | 키워드 검색 결과 중 대략적인 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | CREATE\_AT | 작성 일시 | DATETIME | X | PK | 트윗을 작성한 일시 |
| 02 | ID | 작성자 고유번호 | VARCHAR(20) | X | PK |  |
| 03 | TEXT | 트윗 | TEXT | X |  |  |
| 04 | TRUNCATED | 트윗 절단 유무 | CHAR(1) | X |  | 0 - F / 1 - T |
| 05 | HASHTAG | 해시태그 | CHAR(4) | X |  |  |
| 06 | USER | 작성자 | CHAR(4) | X |  |  |
| 07 | RETWEET\_COUNT | 리트윗 수 | INT | O |  |  |
| 08 | FAVORITE\_COUNT | 좋아요 수 | INT | O |  |  |
| 09 | LANG | 언어 | VARCHAR(3) | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.S\_HASHTAG | | | | |
| **Description** | | 트위터 검색 결과 중 햬시태그 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HCODE | 코드 | CHAR(4) | X |  |  |
| 02 | START | 처음 위치 | INT | X |  |  |
| 03 | END | 마지막 위치 | INT | X |  |  |
| 04 | TEXT | 해시태그 | TEXT | X |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.S\_USER | | | | |
| **Description** | | 트위터 검색 결과 중 작성자 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | UCODE | 코드 | CHAR(4) | X |  |  |
| 02 | ID | 고유번호 | VARCHAR(20) | X |  |  |
| 03 | NAME | 이름 | VARCHAR(50) | X |  |  |
| 04 | SCREEN\_NAME | 스크린 이름 | VARCHAR(50) | X |  | @로 시작하는 이름 |
| 05 | LOCATION | 위치 | VARCHAR(10) | O |  |  |
| 06 | DESCRIPTION | 소개글 | TEXT | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_JSON | | | | |
| **Description** | | 타임라인 검색 결과 중 대략적인 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | CREATE\_AT | 작성 일시 | DATETIME | X | PK | 트윗을 작성한 일시 |
| 02 | ID | 작성자 고유번호 | VARCHAR(20) | X | PK |  |
| 03 | TEXT | 트윗 | TEXT | X |  |  |
| 04 | TRUNCATED | 트윗 절단 유무 | CHAR(1) | X |  | 0 - F / 1 - T |
| 05 | HASHTAG | 해시태그 | CHAR(4) | X |  |  |
| 06 | USER | 작성자 | CHAR(4) | X |  |  |
| 07 | RETWEET\_COUNT | 리트윗 수 | INT | O |  |  |
| 08 | FAVORITE\_COUNT | 좋아요 수 | INT | O |  |  |
| 09 | LANG | 언어 | VARCHAR(3) | O |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_HASHTAG | | | | |
| **Description** | | 타임라인 검색 결과 중 해시태그 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HCODE | 코드 | CHAR(4) | X |  |  |
| 02 | START | 처음 위치 | INT | X |  |  |
| 03 | END | 마지막 위치 | INT | X |  |  |
| 04 | TEXT | 해시태그 | TEXT | X |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.T\_USER | | | | |
| **Description** | | 타임라인 검색 결과 중 작성자 정보 저장 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | UCODE | 코드 | CHAR(4) | X |  |  |
| 02 | ID | 고유번호 | VARCHAR(20) | X |  |  |
| 03 | NAME | 이름 | VARCHAR(50) | X |  |  |
| 04 | SCREEN\_NAME | 스크린 이름 | VARCHAR(50) | X |  | @로 시작하는 이름 |
| 05 | LOCATION | 위치 | VARCHAR(10) | O |  |  |
| 06 | DESCRIPTION | 소개글 | TEXT | O |  |  |

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | | TWITTER.HASHTAG\_COUNT | | | | |
| **Description** | | 해시태그 카운트 테이블 | | | | |
| **No** | **Column Name** | | **Type** | **NULL** | **Key** | **Commnet** |
| 01 | HASHTAG | 해시태그 | VARCHAR(10) | X | PK |  |
| 02 | COUNT | 횟수 | INT | X |  |  |

**제6절. 구현(하둡)**

**2.6.1. 스쿱 (데이터베이스와 하둡 분산 파일 시스템 간의 연동)**

**스쿱 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:DB종류://IP주소/DB이름 |
| username | DB 계정 |
| password | DB 암호 |
| table | 데이터를 가져올 테이블 |
| columns | 테이블에서 가져올 컬럼 리스트 |
| target-dir | 저장될 HDFS 디렉토리 경로 |

**스쿱 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:DB종류://IP주소/DB이름 |
| username | DB 계정 |
| password | DB 암호 |
| table | 데이터를 저장할 테이블 |
| columns | 테이블에서 매핑될 컬럼 리스트 |
| export-dir | 데이터를 가져올 HDFS 디렉토리 경로 |
| input-fields-terminated-by | 구분자 |

**2.6.2. 맵퍼 및 리듀스(java 파일)**

**KeywordCount.java / HashtagCount.java(코드 동일)**

|  |
| --- |
| /\*\*  \* @KeywordCount  \* @Title: 드라이버 클래스(맵과 리듀스를 등록하는 일 수행)  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \* @Modifier: Lee\_yun\_Hyuck  \* @Modify\_on: 2019-04-09  \* @text: 주석 추가, 클래스명 변경  \*/  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 {  // 하둡 실행(hdfs-site, core-site)에 필요한 conf 객체 생성  Configuration conf = new Configuration();  // 잡 실행을 위한 잡 객체 생성  Job job = Job.getInstance(conf, "Keywordcount");  // 잡 실행에 필요한 사용자 라이브러리 파일 지정  job.setJarByClass(KeywordCount.class);  // 잡에서 사용할 클래스들 설정  job.setMapperClass(KeywordCountMapper.class);  job.setCombinerClass(KeywordCountReducer.class);  job.setReducerClass(KeywordCountReducer.class);  // 매퍼와 리듀서 클래스의 출력 데이터의 키와 값 타입 설정  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  // 입출력 데이터 경로 설정.  // 첫 번째 인자는 입력 파라미터, 두 번째 인자는 출력 파라미터  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  // 실행에 필요한 경로와 같은 깂들이 정상적으로 들어간다면, 잡 실행  System.exit(job.waitForCompletion(true) ? 0 : 1);  }  } |

**KeywordCountMapper.java / HashtagCountMapper.java(자연어처리 제외, 코드 동일)**

|  |
| --- |
| /\*\*  \* @KeywordCountMapper  \* @Title: 매퍼 클래스(맵에서 선별한 데이터(key, value)를 같은 키를 기준으로 더하는 작업 수행 )  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \* @Modifier: Lee\_yun\_Hyuck  \* @Modify\_on: 2019-04-28  \* @text: 주석 추가, 클래스 명 변경, 자연어처리(한글자 이상 5글자 이하로 제한, 사용자 사전 추가)  \*/  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;  // Mapper 클래스의 generic 타입 <입력키, 입력값, 출력키, 출력값>  // 하둡에서 요구되는 long, int, String에 대응되는 타입으로 변경해서 사용  public class KeywordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  // IntWritable값으로 상수 1을 저장한다.  // 리듀스에서 IntWritable의 값을 가지고 단어 카운트 할 때 사용.  private final static IntWritable one = new IntWritable(1);  // 출력물에서 나오는 단어를 저장하고자 하는 Text 객체  private Text word = new Text();  // 입력되는 키와 값에 대해 리듀스로 넘어갈 키와 값으로 매핑  @Override  protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {  // 공백 단위로 들어온 텍스트를 끊어 온다. 나머지 특수문자를 포함한 나머지 문자에 대해서도 처리  StringTokenizer itr = new StringTokenizer(value.toString(), " \t\n\r,.'\"-=%…()[]{}“▷+ⓒ!?:;#\'");  // 리턴할 다음 토큰이 없을 때(false) 만큼 반복.  while(itr.hasMoreTokens()) {  String token = itr.nextToken();  // ---------------------------------------------자연어 처리 부분---------------------------------------------------  // 코모란 객체 생성 DEFAULT\_MODEL기본 사전 사용 << 사전 정의 가능  Komoran komoran = new Komoran(DEFAULT\_MODEL.FULL);  // 사용자 사전 경로 추가.(사용자 명사 정의 가능)  komoran.setUserDic("/home/vi/eclipse-workspace/KeywordCount/src/dic.user");  // 읽어온 단어 분석  KomoranResult analyzeResultList = komoran.analyze(token);  // tokens 리스트 정의 후, 명사에 대해 분류하여 적재.  List<String> tokens = analyzeResultList.getMorphesByTags("NP","NNP","NNG");  // 요소들을 읽어오기 위한 Iterator 생성 후, tokens 내용 적재.  Ite rator<String> itrs = tokens.iterator();  // ----------------------------------------------------------------------------------------------------------------  // context 객체는 키-값쌍으로 내보낼 때 사용되며, 출력타입으로 인자화 된다.  while(itrs.hasNext()) {  // ktr 변수 생성하여 자연어 처리된 단어 저장.(단어, 공백 제거)  String ktr = itrs.next().trim();  // ktr에 저장된 단어가 한글자 이상이나 5글자 이하일 경우 조건.  if(ktr.getBytes().length > (byte)3 && ktr.getBytes().length < (byte)16) {  //word객체에 ktr 삽입.  word.set(ktr);  //context 객체는 키-값쌍으로 내보낼 때 사용되며, 출력타입으로 인자화 된다.  context.write(word, one);  }  }  }  }  } |

**KeywordCountReducer.java / HashtagCountReducer.java(코드 동일)**

|  |
| --- |
| /\*\*  \* @KeywordCountReducer  \* @Title: 리듀스 클래스(맵에서 선별한 데이터(key, value)를 같은 키를 기준으로 더하는 작업 수행)  \* @author: Lee\_yun\_Hyuck  \* @Create\_at: 2019-04-06  \* @Modifier: Lee\_yun\_Hyuck  \* @Modify\_on: 2019-04-09  \* @text: 주석 추가, 클래스 명 변경  \*/  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Reducer;  // Reducer 클래스 상속, 입력과 출력을 같은 타입으로 출력.  public class KeywordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  private IntWritable result = new IntWritable();  // 출력 파라미터를 가져와서 더해주는 기능을 추가하기 위한 리듀스 메서드 재정의  // Iterable<>로 감싸진 이유는 맵에서 IntWritable에 저장된 값들이 묶여 있기 때문에 values 값들만 추출하기 위해서이다.  @Override  protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {  // 단어의 수만큼 증가하는 값을 저장하는 변수  int sum = 0;  // 각각의 글자 수를 알아내어 단어를 합산한다.  for (IntWritable val : values) {  sum += val.get();  }  // 맵리듀스의 입출력 타입인 IntWritable객체를 생성한 result에 출력값을 설정.  // 이 때, 출력값은 단어의 합산한 값.  result.set(sum);  // context객체의 write메서드를 통해 출력 키로 입력 데이터의 키를 그대로 사용한다.  context.write(key, result);  }  } |

**dic.user - 사용자 정의 사전(자연어 처리) / KOMORAN.jar 필요함**

|  |
| --- |
| #이 파일은 사용자 사전 파일입니다.  #입력 문장 내에 사용자 사전에 포함된 내용이 있는 구간에 대해서는 해당 품사를 출력하게 됩니다.  #형태소의 품사를 적지 않으면 기본적으로 고유명사(NNP)로 인지합니다.  바람과 함께 사라지다 NNG  바람과 함께 NNP  자연어 NNG  아이오아이 NNG  캡틴아메리카  가나다라마 |

**2.6.3. 하둡 실행(jar 파일)**

|  |
| --- |
| yarn jar /home/vi/hadoop/jar/HashtagCount.jar HashtagCount /user/vi/HASHTAG\_INPUT/part-m-00000 HASHTAG\_OUTPUT |
| yarn: Hadoop2.X부터는 yarn에서 클러스터의 관리를 한다.  yarn jar (path/.jar) driverClass (불러올 데이터의 HDFS 경로) (저장될 데이터의 HDFS 경로) |

**제3장. 결론**

**제1절. 실행 환경**

T-SA를 실행을하기 위해서는 [표 4], [표 5], [표 6]과 같이 실행 환경을 구축해야 한다.

**[표 4]** T-SA 실행에 필요한 파이썬 모듈과 유틸리티 및 설치 명령어

|  |  |
| --- | --- |
| 모듈 및 유틸리티 | 설치 명령어 |
| 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\* |

**[표 5]** T-SA 실행에 필요한 하둡 관련 유틸리티 및 설치 명령어

|  |  |
| --- | --- |
| 유틸리티 | 설치 명령어 |
| ssh | sudo apt-get install ssh |
| pdsh | sudo apt-get install pdsh |

**[표 6]** T-SA 실행에 필요한 기타 유틸리티 및 설치 명령어(URL)

|  |  |
| --- | --- |
| 유틸리티 | 설치 명령어(URL) |
| OpenJDK8 | sudo apt-get install openjdk-8-jdk |
| KOMORAN | https://github.com/shin285/KOMORAN |

**제2절. 트위터 API 파라미트 설정 값**

T-SA에서는 트위터 관련 정보를 가져오는 기능을 2가지(Search, Timeline) 제공하고 각각의 기능에 필요한 파라미터는 다르다. Search에는 query, fromDate, toDate, maxResults를 사용하고 Timeline에서는 screen\_name, count, include\_rts를 사용한다. 그중 19대 대통령 선거에 관한 자료를 가져오기 위해 Search를 사용하고 각각의 파라미터(query, maxResults)에 순서대로 "문재인" OR "홍준표" OR "안철수" OR "유승민" OR "심상정", 100을 주었다. 또한 기간의 시작 시간과 종료 시간을 '201704180000', '201705100000'로 주었다.그 이유는 19대 대통령 선거 기간이 2017년 4월 17일 자정부터 2017년 5월 9시 투표마감 시간까지이기 때문이다. 그러나 Primium을 사용하였지만 500개의 트윗을 100번만 검색할 수 있는 제한과 Search의 검색하는 기준이 최신일시(toDate)이기 때문에 Sandbox와 Primium을 혼용하여 22일 동안 하루를 4개로 나누어서 검색 하였다. 그결과 선거 기간(22) 동안 일별 트윗(1200=100+500+100+500)을 검색하여 약 26,000개의 트윗을 수집하였다.

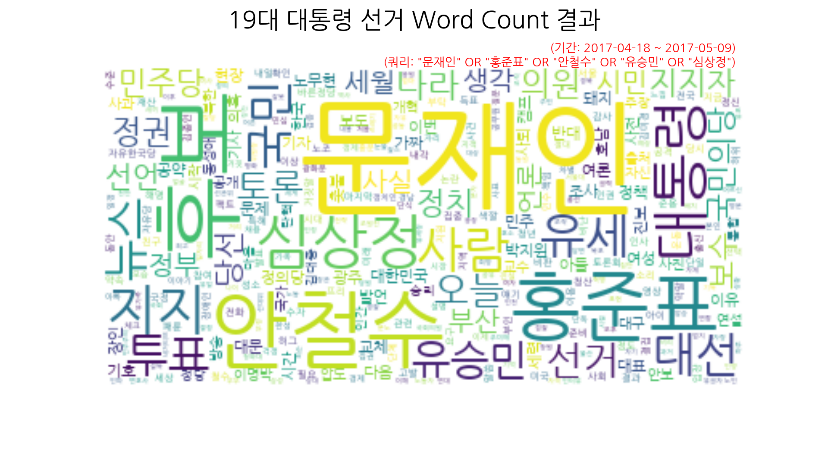
**제3절. 분석 및 시각화**

T-SA에서는 크게 3가지를 분석하였다.

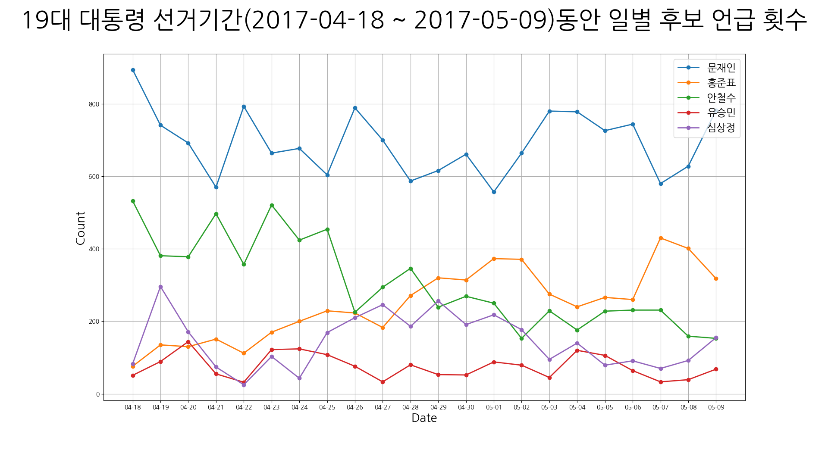
첫 번째로는 19대 대통령 선거 기간 동안 사용한 키워드의 카운트를 진행하여 Word Cloud를 통해 어떠한 키워드가 많이 사용되었는지 분석하였고 Word Cloud를 통해 시각화([그림 16])하였다.

두 번째로는 19대 대통령 선거 기간 동안 선거 후보자 중 득표가 많은 상위 5명의 이름이 매일 몇 번 언급되었는지 분석하였고 선 그래프([그림 17])로 시각화하였다.

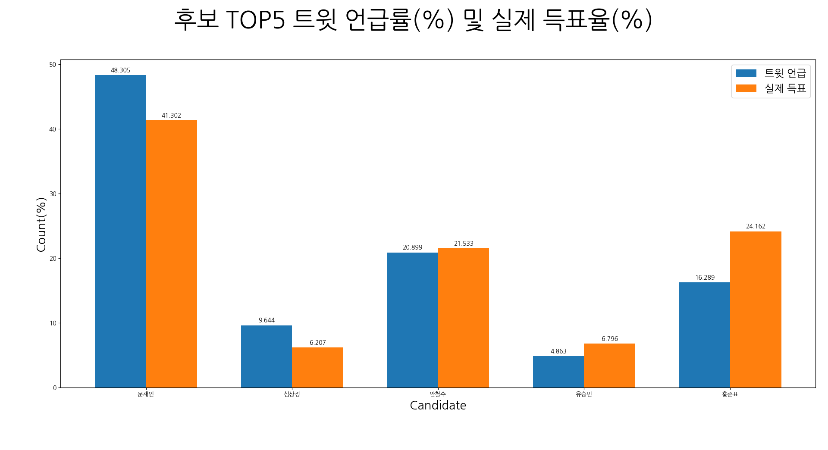
마지막으로는 19대 대통령 선거 기간 동안 사용한 키워드의 카운트와 중앙선거관리위원회에서 제공하는 득표수를 비교 분석하여 막대 그래프([그림 18])와 원 그래프([그림 19]) 등으로 시각화하였다.



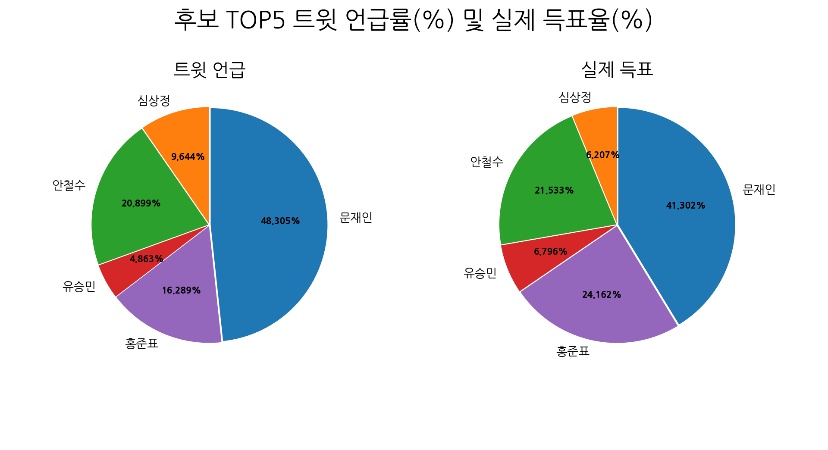
**[그림 16]** Word Cloud을 통한 키워드 사용횟수 분석



**[그림 17]** 선 그래프를 통한 일별 언급 분석



**[그림 17]** 막대 그래프를 통한 트윗 언급과 실제 득표율 비교 분석



**[그림 18]** 원 그래프를 통한 트윗 언급과 실제 득표율 비교 분석

**제4절. 최종 결론**

[그림 17] 또는 [그림 18]을 통해서 T-SA에서 분석한 트윗에 후보들의 언급률과 실제 득표율이 비슷한 것을 확인할 수 있었다. 추후 이번에 분석한 19대 대통령 선거뿐만 아니라 이전 대통령 선거와 국회의원 선거 등을 통해 더 많은 표본에 대한 분석을 진행하여 적중률을 확인하고 이를 토대로 대한민국 제20대 대통령 선거 때 누가 당선될 것인지 예측할 수 있을 것이라고 생각된다.

**제5절. 느낀점**

**제4장. 부록**

**부록 1. 마리아 디비 관련 설정**

**권한 테이블 설정**

마리아 디비의 권한 테이블 설정 방법은 아래와 같다.

1. 터미널에 sudo mysql\_secure\_installation을 실행시킨다.

2. root 계정의 비밀번호를 입력한다.

|  |
| --- |
| 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):** |
| root 계정의 비밀번호를 입력한다. 단, 마리아 디비의 root 계정은 셸 인증이 기본적으로 설정되므로 root 계정으로 실행하였다면 비밀번호를 입력하지 않고 넘어가도 된다. |

3. 마리아 디비에 접속할 때 사용할 root 계정의 비밀번호를 설정한다.

|  |
| --- |
| 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: |
| 2에서 사용한 비밀번호를 사용할 경우 n, 다른 비밀번호를 사용할 경우 Y를 입력한다.  Y를 입력한 경우 패스워드 입력과 패스워드 재입력 과정을 거친다. |

4. 익명 사용자의 삭제 여부를 설정한다.

|  |
| --- |
| 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]** |
| 익명 사용자를 삭제할 경우 Y, 삭제하지 않을 경우 n을 입력한다. |

5. root 계정을 이용한 원격 접속 허가 여부를 설정한다.

|  |
| --- |
| 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]** |
| root 계정의 원격접속을 허가할 경우 Y, 허가하지 않을 경우 n을 입력한다. |

6. 테스트 데이터베이스를 삭제 할 것인지 설정한다.

|  |
| --- |
| 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]** |
| 마리아 디비의 경우 기본적으로 테스트 데이터베이스를 제공해주기 때문에 이러한 설정을 하는 과정을 거치게 된다.  테스트 데이터베이스를 삭제할 경우 Y, 삭제하지 않을 경우 n을 입력한다. |

7. 2에서 6까지 설정한 것들을 적용한다.

|  |
| --- |
| **Reload privilege tables now? [Y/n]** |
| 지금까지 작성한 권한 테이블의 설정을 적용할 경우 Y, 적용하지 않을 경우 n을 입력한다. |

**부록 2. 테이블 생성 쿼리문**

**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));

**부록 3. 참고문헌 및 사이트**

https://www.ubuntu.com/

https://developer.twitter.com

http://www.tweepy.org

tweetrend, 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/

중앙선거관리위원회, http://www.nec.go.kr/portal/main.do

정재화, 시작하세요! 하둡 프로그래밍 빅데이터 분석을 위한 하둡 기초부터 YARN까지[개정2판], 2016.05.13, 위키북스

**부록 4. 깃허브**

Github URL: **https://github.com/SeokJune/BigData\_VI\_T-SA/**