**중간 보고서**

**T-SA: Twitter keyword Search API based tweet Analysis**

**()**



과 목: 산학캡스톤디자인1

분 반: 01

담 당 교 수: 정현숙 교수님

팀 명: 브이아이(VI)

조 원: 이석준(20165072)

이윤혁(20165062)

배인규(20165073)

서재익(20144773)

**- 목 차 -**

**제1장 서 론**

제1절 개발 동기 및 필요성 --------------------------------------------------------- 1

제2절 기존 연구 ------------------------------------------------------------------- 1

제3절 개발 목표 ------------------------------------------------------------------- 1

제4절 시스템 명세 ----------------------------------------------------------------- 2

1.4.1. 시스템 동작 과정 ----------------------------------------------------------- 2

1.4.2. 원본 데이터 SET ----------------------------------------------------------- 2

**제2장 본 론**

제1절 개발 환경 ------------------------------------------------------------------- 3

2.1.1. Ubuntu -------------------------------------------------------------------- 3

2.1.2. Python --------------------------------------------------------------------- 3

2.1.3. MariaDB ------------------------------------------------------------------- 4

2.1.4. Eclipse, OpenJDK ----------------------------------------------------------- 5

2.1.5 Hadoop Echo System -------------------------------------------------------- 6

제2절 TwitterAPI ------------------------------------------------------------------- 7

제3절 T-SA 흐름도 ---------------------------------------------------------------- 8

제4절 구현(Python) ----------------------------------------------------------------- 9

2.4.1. T-SA.py -------------------------------------------------------------------- 9

2.4.2. TwitterAPI.py -------------------------------------------------------------- 10

2.4.3. DBModule.py -------------------------------------------------------------- 11

2.4.4. Visualization.py ------------------------------------------------------------ 12

제5절 DB/Table 정의서 ------------------------------------------------------------ 16

제6절 구현(Hadoop) ---------------------------------------------------------------- 23

2.6.1. Sqoop(DB/HDFS 연동) ------------------------------------------------------ 23

2.6.2. Map/Reduce(.java) ---------------------------------------------------------- 23

//= FULL //

= LIGHT //

2.6.3. Hadoop 실행(.jar) ---------------------------------------------------------- 25

**제3장 결 론**

실행환경 파이썬 – module, pip, dic.uers path , 코모란 (.jar)

데이터 가져오는 것

사각화

기대효과 – 최종 결론

부록

제1절 참고문헌 및 사이트 --------------------------------------------------------- 26

**-부 록-**

각가의 역할 – 각 클래스(세부 메소드)

느낀점

참고문헌

1. 부 록 1 (깃허브)

**-별 첨-**

1. Hadoop 설치.odt

2. sqoop1.docx

3. Demonstrate Video (CAP20190429.mp4)

제1장 서 론

**제1절 개발 동기 및 필요성**

트위터는 140자 이내로 자신의 의견을 적을 수 있는 SNS(Social Network Service) 중 하나이다. 지금은 페이스북, 인스타그램 등에 밀려서 점유율이 많이 낮아졌지만 글쓰기의 편리함과 정보의 빠른 확산성 때문에 사회적 영향은 무시할 수 없다. 계정 생성에 메일 주소만 필요하고, 휴대폰 인증은 형식상 하지만 남에게는 보이지 않기 때문에 익명성 또한 타 SNS에 비해 강하다. 따라서 자신의 의견을 자신 있게 표명하는 경우가 많아 사회현상에 대한 다양한 의견이 많이 만들어진다. 트위터의 게시물인 트윗은 한글을 기준으로 하여 140자 이내의 글로 작성이 되어 있기 때문에 하나의 의견을 보기 위한 시간이 많이 소요되지 않으며, 현재 이슈가 되는 주제에 대해서 하고 있는 이야기를 트위터 검색을 통하여 살펴보면 페이스북 검색 결과에 비하여 빠르고 다양하게 살펴볼 수 있다.

2008년과 2012년 미국 대선에서 오바마 캠프의 성공적 사용으로 전 세계적인 주목을 받았다. 오바마 캠프에서는 처음부터 출마 선언을 SNS(Social Networking Sites or Social Network Services)에 공개했으며 주요 지지층을 결집하고 정치적 우군으로 동원하는 과정에서 SNS를 활용했다. 2008년과 2012년 대선에서 오바마가 매우 뛰어난 소셜 미디어 운영 방식을 보여 선거에서 소셜 미디어의 긍정적 역할이 보였고, 오바마는 소셜 미디어 대통령이라는 말을 들었다. 2012년 오바마 대통령은 선거 캠페인 도중에 ‘소셜 미디어는 후보들이 저널리즘의 필터 없이 얘기할 수 있는 플랫폼’이라고 언급했다. 그 이후, 힐러리 클린턴(민주당)과 도널드 트럼프(공화당) 간의 미국 대선에서의 트럼프는 1130만 명의 트위터 팔로워와 1056만 명의 페이스북 팔러워를 보유하고 있는 트럼프는 공화당 후보 경선 기간 내내 각종 정치 현안과 관련된 메시지를 SNS에 쏟아내며 화제를 끌고 다녔다. 이는 1000만 명의 트위터 팔로워를 보유한 오바마 대통령을 훨씬 뛰어넘은 수치다. 특히 트럼프는 공약 홍보나 치적을 보여주기보단 상대 후보를 노골적으로 비난하고 불법 이민, 인종차별 등 민감한 문제에 대해 극단적인 발언을 서슴지 않고 내뱉으며 미국 정치권을 충격에 빠트리기도 했다. 하지만 SNS 속 지지자들은 이런 극단적인 발언에 열광했고 그는 대통령으로 선출되었다. 우리나라는 2002년 대선에서 인터넷을 적극적으로 활용한 경험으로 시작해, 2011년 서울 시장 보궐 선거에서 본격적으로 소셜 미디어가 어떻게 선거에서 활용될 수 있는 가를 논의하게 만드는 계기가 되었다. 특히 2011년의 시장 선거에서는 각 후보의 네트워크와 메시지의 전파, 소셜 미디어에서의 투표 참여 독려 등 여러 측면에서 소셜 미디어의 역할에 대한 깊이 있는 분석이 등장했다. 이에 따라 이 프로젝트를 통해 19대 대통령 선거 후보들의 트위터 활동 분석을 포함하여, 트위터와 당선의 관계를 분석하고자 한다.



[그림 1] 도널드 트럼프 트위터, <https://twitter.com/realDonaldTrump>

**제2절 기존 연구**

트윗 트렌드 사이트는 와이즈넛에서 제공하는 서비스이다. 트윗 트렌드에서는 약 400만명의 트위터 계정을 수집하여 국내 최대의 검색 서비스를 제공하고 있다. 과거 2011년 1월 이후의 데이터부터 하루 1천만 트윗을 수집하고 있으며, 리트윗, URL 파싱, 파급력 등 다각적인 분석 결과를 서비스로 제공하고 있으며, 마케팅 등 다양한 분야에서 활용할 수 있도록 원문을 내려 받아 볼 수 있는 기능을 제공을 해주고 있다. 트윗 트렌드는 기간을 설정하여 특정 키워드를 통해 시각화를 제공하며, 해당되는 트윗의 정보를 보여준다. 비로그인 사용자일 경우 최대 7일 기간에 해당하는 키워드 하나만 검색이 가능하며, 로그인 사용자의 경우에는 무료 회원과 유료 회원으로 나뉘어 30일, 무제한으로 나뉘어 3개의 키워드에 대한 정보를 얻을 수 있다.

foller.me사이트는 콘스탄틴 코브세닌(Konstantin Kovshenin)이 Twitter API를 이용하여 만든 사이트이다. foller.me는 공개된 프로필에 대해 주제, 언급, 해시태그 등의 정보를 분석하여 보여준다. 사이트는 요청한 사용자의 프로필과 사용자가 작성한 최신 트윗들을 수집하여, 내용을 분석을 통해 자주 사용되는 단어들을 출력하여 보여준다.

중앙선거관리위원회 선거통계시스템사이트는 실시하는 선거의 주요일정을 시간 순으로 확인할 수 있으며, 선거의 (예비)후보자를 한눈에 확인 가능하고 (예비)후보자 별 상세 정보를 확인 할 수 있다. 또 한, 선거당일 투표마감 후 개표율과 후보자별 득표수, 득표율을 확인 할 수 있다. 선거통계시스템에서 제공하는 통계정보는 누구나 무료로 이용 가능하며, 제공받는 정보는 자료의 출처를 반드시 명시하고, 이를 다른 정보와 함께 제공하는 경우에는 각 자료의 출처를 명확히 하여 정보이용자의 오해가 발생하지 않도록 하는 선에서 정보 활용이 가능하다. 중앙선거관리위원회 선거통계시스템사이트에서 19대 대통령 선거의 결과를 가져오기 위해 역대선거에서 개표현황을 통해 데이터를 불러 올 수 있으며, 이는 pdf나 excel로 제공 받을 수 있다.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| [그림 2] tweetrend,  <http://tweetrend.com/> | [그림 3] foller.me,  <https://foller.me/> | [그림 4] 중앙선거관리위원회 선거통계시스템, <http://info.nec.go.kr/> |

**제3절 개발 목표**

기업과 기관에서는 내부의 데이터 뿐만 아니라 페이스북, 트위터, 카카오톡과 같은 SNS를 통해 데이터를 수집하고 분석하여 그동안 파악하지 못했던 다양한 결과를 얻고자 시도하고 있다. 이런 사회 관계망에서 발생하는 대량의 데이터를 분석하여 선거결과를 예측하고자 한다. 트위터는 전세계 1억명이 넘는 이용자가 하루 평균 2억개 이상의 트윗을 발생시킨다. 2012년 대통령 선거 당시 174만개 트윗을 분석하여 민심의 흐름을 추적하였고, 2014년에는 특정 IT박람회와 관련된 트윗을 분석하여 정보교환의 사회연결망을 분석하고자 하였다. 본 프로젝트에서는 키워드와 기간을 설정하여 해당하는 기간 동안 작성된 트윗을 가져와서 포함된 단어와 해시태그의 빈도수를 분석하고, 이를 19대 대선에 적용을 시켜보려고 한다. 대선 후보들 중 상위 5명에 대한 트윗을 가져와 언급된 횟수에 따라 대선에 어느정도 영향을 미치는지 분석하고자 한다. 이 후, 중앙선거관리위원회의 선거통계시스템을 이용하여, 분석을 통해 얻어진 결과와 비교를 통해 결과를 도출하고자 한다.

첫 번째, 대통령 선거에 사용하는 용어가 포함된 트윗을 수집하기 위해서 트위터에서 제공하는 Twitter API를 이용한다. 해당 프로그램은 API를 통해 키워드 쿼리를 전송하고 질의에 응답하는 결과를 JSON형식을 제공하여 준다. Twitter API를 이용하기 위해서는 아이디와 비밀번호가 아닌 개발자 등록을 한 사람들에 한 해서, Consumer key를 발급해 준다. 트윗 데이터를 수집할 시에 무료 버전인 Standard 버전을 이용하는 경우 가져올 수 있는 트윗이 100개로 제한 되며 질의를 50번 요청할 수 있다.

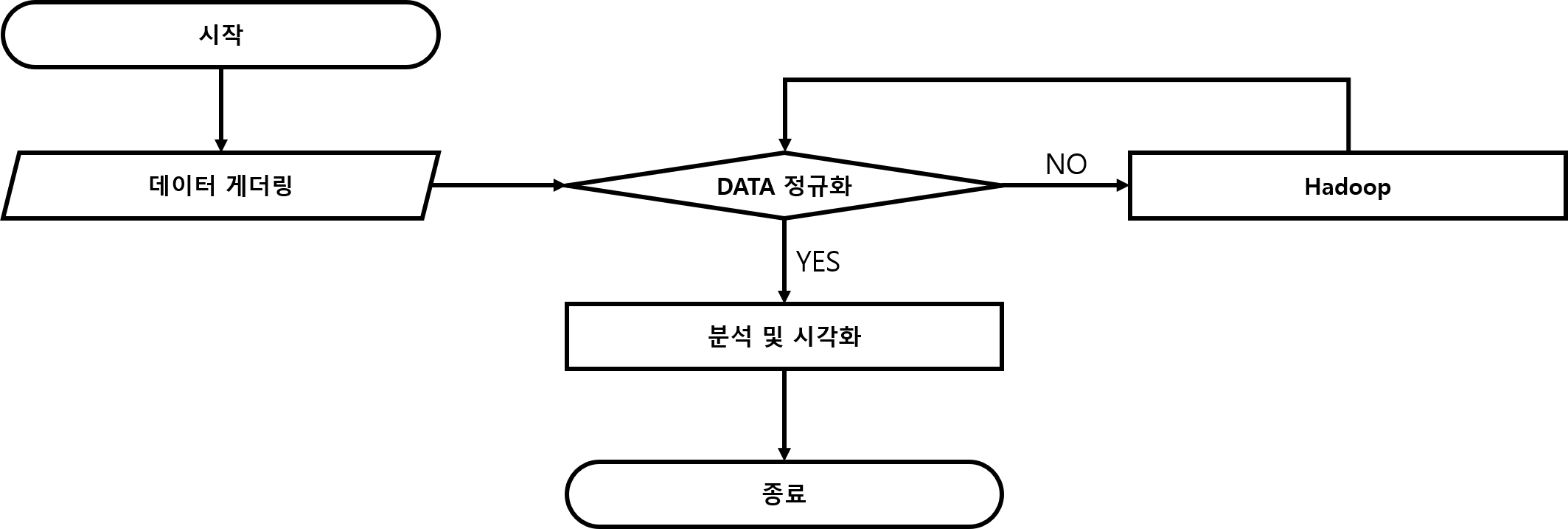
두 번째, 한정된 정보 수집을 방지하기 위해 19대 대통령 선거와 관련된 키워드를 대통령 선거 기간 동안(2017.4.15 ~ 2017.05.09; 24일)의 트윗을 통해 색출을 할 것이다. 이는 좀 더 다양한 사람들이 선거 기간 동안에 대한 트윗들을 수집할 수 있으며, 대선에 있어 정확도를 높이는 것에 있다. 이에 발생하게 되는 중복 된 트윗에 대해서는 중복을 제거를 하고 수집 할 것이다. 해당 트윗들은 대한민국의 19대 대통령 선거기간 동안에 이루어진 트윗이기 때문에 한국어를 사용한다. 한국어로 작성된 트윗은 주어, 서술어, 목적어 등이 빈번히 생략이 되며, 일부 사용자들은 일부러 띄어쓰기를 안 하는 경향이 있다. 그렇기 때문에 자연어 처리를 통해 고유명사와 일반 명사로 분류하여 키워드의 빈도수를 구하여, 워드 클라우드와 같은 시각화 툴들을 통해 상위 10개의 키워드를 색출하고, 색출된 키워드들을 통해 후보와 관련된 키워드가 포함된 트윗을 수집하고자 한다.

세 번째, 중앙선거관리위원회 선거통계시스템에서 제공해주는 19대 대통령 선거 개표현황 데이터를 이용하여, 상위 5명의 후보들에 대해 득표수와 득표율을 얻고자 한다. 본 프로젝트에서는 19대 대통령 선거 후보자별 득표율과 수집된 트윗들의 후보자별 언급 횟수를 토대로 비교 및 분석을 하고자 한다.

마지막으로 유력 대선후보들이 트위터를 이용해 주요한 정치적 메시지를 유권자들에게 전달하기 때문에 후보자들이 작성한 트윗을 분석하여, 가장 많이 사용된 Hashtag, 후보자들의 상세정보를 살펴보고자 한다.

**제4절 시스템 명세**

**1.4.1. 시스템 동작 과정**



[그림 5] T-SA 프로젝트의 시스템 명세

본 프로젝트의 전체적인 흐름은 그림 5와 같다. 비정형 데이터의 수집과 분석을 위하여 Twitter API를 이용하여 데이터를 수집하게 된다. 해당 API는 사용자가 설정한 쿼리에 대해 결과를 JSON 형식으로 제공을 해주게 된다. JSON(JavaScript Object Notation)은 키와 값의 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 잇는 텍스트를 사용하는 개방형 표준 포맷이다[2]. JSON 형식의 데이터는 데이터 베이스에 테이블 정의를 통해 우선적으로 저장이 되게 된다. 저장된 데이터에 한해서 분석을 할 수 있도록 정규화 된 데이터인지 아닌지를 판별하게 된다. 정규화가 되어 있지 않다면, 하둡 에코 시스템을 통해 정규화를 시켜 다시 데이터 베이스에 저장을 하게 된다. 정규화된 데이터들은 분석을 통해 최종적으로 결과를 도출하게 된다.

제2장 본 론

**제1절 개발 환경**

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

**1.2.1 Ubuntu**

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 사용자들이 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.

**Version Check(~$ lsb\_release -a)**

|  |
| --- |
| No LSB modules are available.  Distributor ID: Ubuntu  Description: Ubuntu 18.04.2 LTS  Release: 18.04  Codename: bionic |

**1.2.2 Python**

파이썬(Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 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.x대 버전의 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.x 는 레거시(낡은 기술)이고, 파이썬 3.x가 파이썬의 현재와 미래가 될 것"이라고 요약을 했는데, 처음 배우는 프로그래머들은 파이썬 3으로 시작하는 것을 권장하고 있다.

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

**Python 설치**

~$ sudo apt-get install python3

**Version Check(~$ python3 --version)**

|  |
| --- |
| Python 3.6 |

**1.2.3 MariaDB**

MariaDB는 MySQL의 발전된 형태의 대체제로써, https://downloads.mariadb.org/에서 다운로드 받을 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 해결된MariaDB 와 MySQL 버전이 배포된 이후에 릴리즈된다.

MariaDB는 현재까지 최신의 MaySQL과 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 대체제라고 말할 수 있다.

**MariaDB 설치**

~$ sudo apt-get install mariadb-server

- 설치 날짜 기준(2019.03.30)으로 MariaDB의 10.1.38버전와 의존성 패키지 설치

**MariaDB의 권한 테이블 설정**

~$ sudo mysql\_secure\_installation

|  |
| --- |
| 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 MariaDB, 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):**  OK, successfully used password, moving on... |

**Enter current password for root (enter for none)** → MariaDb의 root계정은 쉘인증이 기본적으로 성정되므로 root계정으로 실행됐다면 비밀번호 없이(Enter) 아니면 비밀번호 입력

|  |
| --- |
| 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:  Password updated successfully!  Reloading privilege tables.  ... Success! |

**Set root password? [Y/n]** → 따로 패스워드를 성정하고 싶으면 Y, root그대로 사용할려면 n

- New password: → 패스워드 입력

- Re-enter new password: → 패스워드 재입력

|  |
| --- |
| 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]**  ... Success! |

**Remove anonymous users? [Y/n]** → 익명 사용자를 삭제할지여부(Y-삭제, n-삭제 안함)

|  |
| --- |
| Nomally, root should only be allowed to connect from ‘localhost’. This ensures that someone cannot guess at the root password from the network.  **Disallow root login remotely? [Y/n]**  ... Success! |

**Disallow root login remotely? [Y/n]** → 원격 접속으로 루트 로그인 허용 여부(Y-거부, n-허용)

|  |
| --- |
| 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]**  - Dropping test database...  ... Success!  - Removing privileges on test database...  ... Success! |

**Remove test database and access to it? [Y/n]**

→ 기본적으로 테스트 데이터베이스를 제공해주기 때문에 이와같이 물어봄

→ 테스트 데이터베이스 삭제 여부(Y-삭제, n-삭제 안함)

|  |
| --- |
| **Reload privilege tables now? [Y/n]**  ... Success!  Cleaning up...  All done! If you’ve completed all of the above steps, your MariaDB  installation should now be secure.  Thanks for using MariaDB! |

**Reload privilege tables now? [Y/n]** → 지금까지 작성한 권한 테이블을 적용 여부(Y-적용, n-적용 안함)

**MariaDB의 Character sets 설정(추가 입력)**

~$ sudo gedit /etc/mysql/mariadb.conf.d/50-server.cnf

|  |
| --- |
| # \* Character sets  #  # MySQL/MariaDB default is Latin1, but in Debian we rather default to the full  # utf8 4-byte character set. See also client.cnf  #  **character-set-setver = utf8mb4**  # collation-server = utf8mb4\_general\_ci  **collation-server = utf8mb4\_unicode\_ci** |

utf8m64\_general\_ci(default)을 사용할 경우 정렬 속도가 utf8m64\_unicode\_ci에 비해 약간 빠르 긴 하지만 거의 차이는 없음, 한글/일본어와 같이 비 라틴계 언어들의 정렬이 조금 어색한 경우가 있기 때문에 utf8m64\_unicode\_ci 사용

- MySQL에 대한 Character sets설정 → /etc/mysql/mariadb.conf.d/50-mysql-clients.cnf

- Client에 대한 Character sets설정 → /etc/mysql/mariadb.conf.d/50-client.cnf

**Version Check(~$ mariadb –version)**

|  |
| --- |
| mariadb Ver 15.1 Distrib 10.1.38-MariaDB, for debian-linux-gnu (x86\_64) using readline 5.2 |

**1.2.4 Eclipse, OpenJDK**

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 Developer)란 이름으로 개발되었던 것인데, 엔진 부분을 오픈소스로 공개한 것을 기반으로 지금의 이클립스로 발전해 왔다.

OpenJDK는 Java SE (Standard Edition) 기반의 오픈 소스 JDK다. 2006년 Sun Micro System 은 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.

**OpenJDK 설치**

~$ sudo apt-get install openjdk-8-jdk

**Version Check(Help->About Ecli[se IDE)**

|  |
| --- |
| Eclipse IDE for Enterprise Java Developers.  Version: 2019-03 (4.11.0)  Build id: 20190314-1200 |

**Version Check(~$ java -version)**

|  |
| --- |
| openjdk version "1.8.0\_191"  OpenJDK Runtime Environment (build 1.8.0\_191-8u191-b12-2ubuntu0.18.04.1-b12)  OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode) |

**1.2.5 Hadoop Echo System**

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.

**Version Check(~$ hadoop version)**

|  |
| --- |
| Hadoop 3.2.0  Source code repository https://github.com/apache/hadoop.git -r e97acb3bd8f3befd27418996fa5d4b50bf2e17bf  Compiled by sunilg on 2019-01-08T06:08Z  Compiled with protoc 2.5.0  From source with checksum d3f0795ed0d9dc378e2c785d3668f39  This command was run using /home/vi/hadoop-3.2.0/share/hadoop/common/hadoop-common-3.2.0.jar |

**Version Check(~$ sqoop version)**

|  |
| --- |
| 2019-04-30 16:47:14,136 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7  Sqoop 1.4.7  git commit id 2328971411f57f0cb683dfb79d19d4d19d185dd8  Compiled by maugli on Thu Dec 21 15:59:58 STD 2017 |

**제2절 Twitter 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 진행을 한다. 가입이 완료가 되어야 개발자 등록할 수 있으며, 개발자 등록을 하지 않으면 키를 발급받지 못한다. 키를 발급 받으면, 개발 환경 설정을 통해 사용되는 영역에 따라 자유롭게 이용할 수 있다.

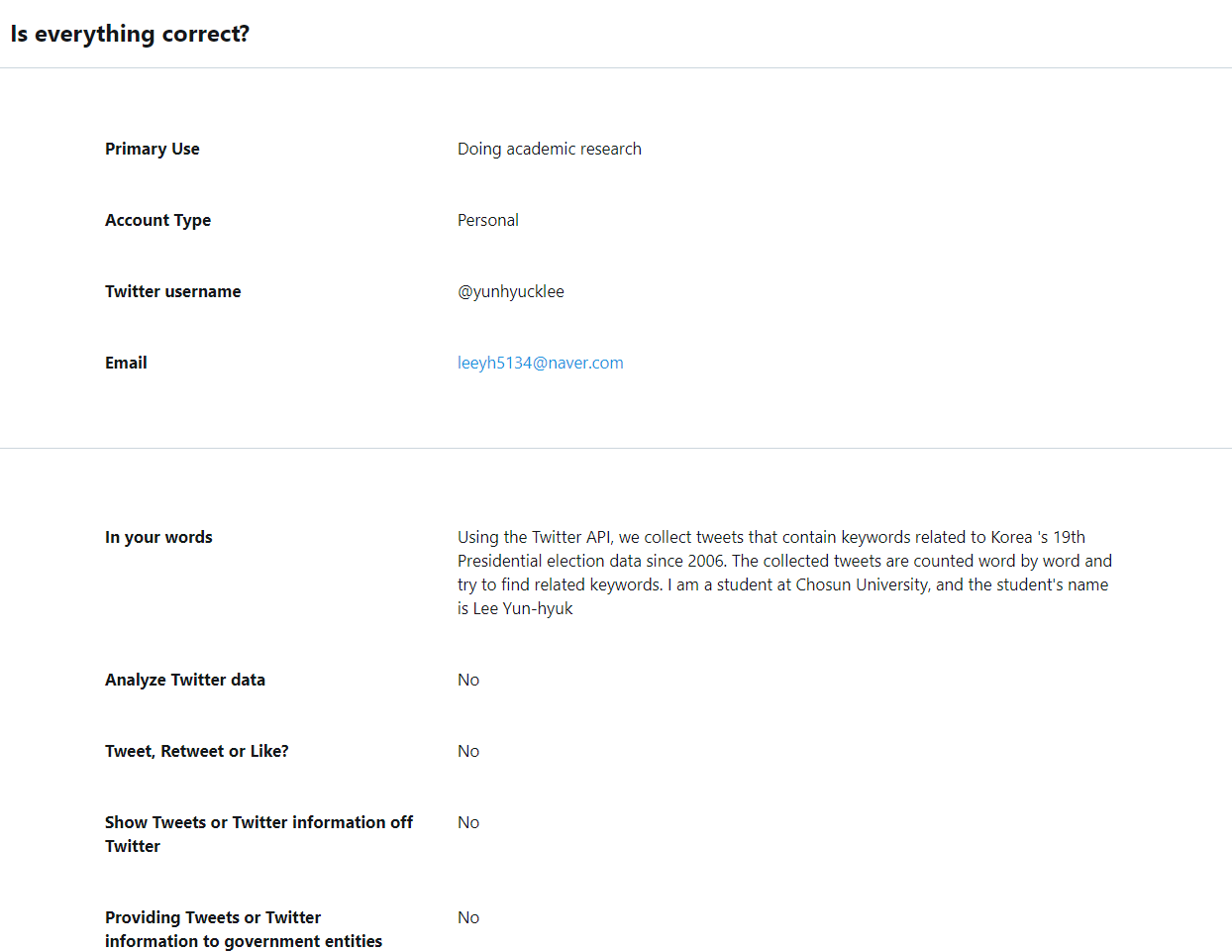
**개발자 등록**

개발자 등록을 하기위해 <https://developer.twitter.com/> 에 접속하여 등록 할 수 있다. 개발자 등록을 하기 전에는 웹페이지 우측 상단의 모습은 그림6과 같다. 다음 개발자 등록을 하기위해서는 Apply를 통해 다음 단계로 진행한다.

[그림 6] 좌 개발자 등록 전, 우 개발자 등록 후

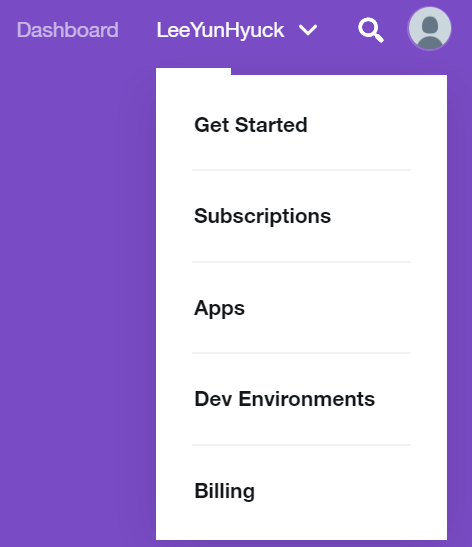
Apply for a developer account를 누르게 되면, 트위터의 개발 도구를 사용하는 이유에 대해 묻는 질문이 현재는 추가가 되었다. 이에 대한 질문은 학술적으로 사용하는 이유로 지정하고 진행하였다. 질문을 선택하게 되면 현재 로그인 된 계정을 기반으로 하여, 개발자 등록을 할 건지 묻는 페이지가 활성화 된다. 기본적으로 트위터에 핸드폰 인증과 이메일 인증이 없다면, 별다른 설정 없이 살고 있는 지역과 별명을 추가 시켜주고 다음 단계로 진행하면 된다. 그런 다음 사용 목적에 묻는 페이지가 나온다. 페이지가 변경 되기 전에는 모든 응답을 채워야 했지만, 변경이 된 후에는 간단하게 사용목적에 대해 작성만 해주면 된다. 사용목적은 영어로 200자 이상, 학생인 경우 학교와 이름을 포함하여 작성해 주면 된다. 본 프로젝트에서는 트위터 개발자 등록 시, 사용목적으로 ‘트위터API를 이용하여, 2006년 이후의 데이터 중 한국의 19대 대통령 선거와 관련된 키워드들이 포함된 트윗들을 수집한다. 수집한 트윗들은 단어 별로 카운트 하여 연관된 키워드를 색출하려고 한다. 조선대학교 학생이며, 학생의 이름은 xxx입니다.’라고 작성 하였다. 작성이 완료 되어 다음 단계로 진행하면, 요청이 맞는지 확인하는 페이지가 그림7과 같이 나오게 된다. 마지막으로 이메일 인증을 진행하게 된 후, 트위터에서 개발자 승인을 통해 사용 가능하다.



[그림 7] 개발자 등록을 위한 정보기입이 완료된 상태

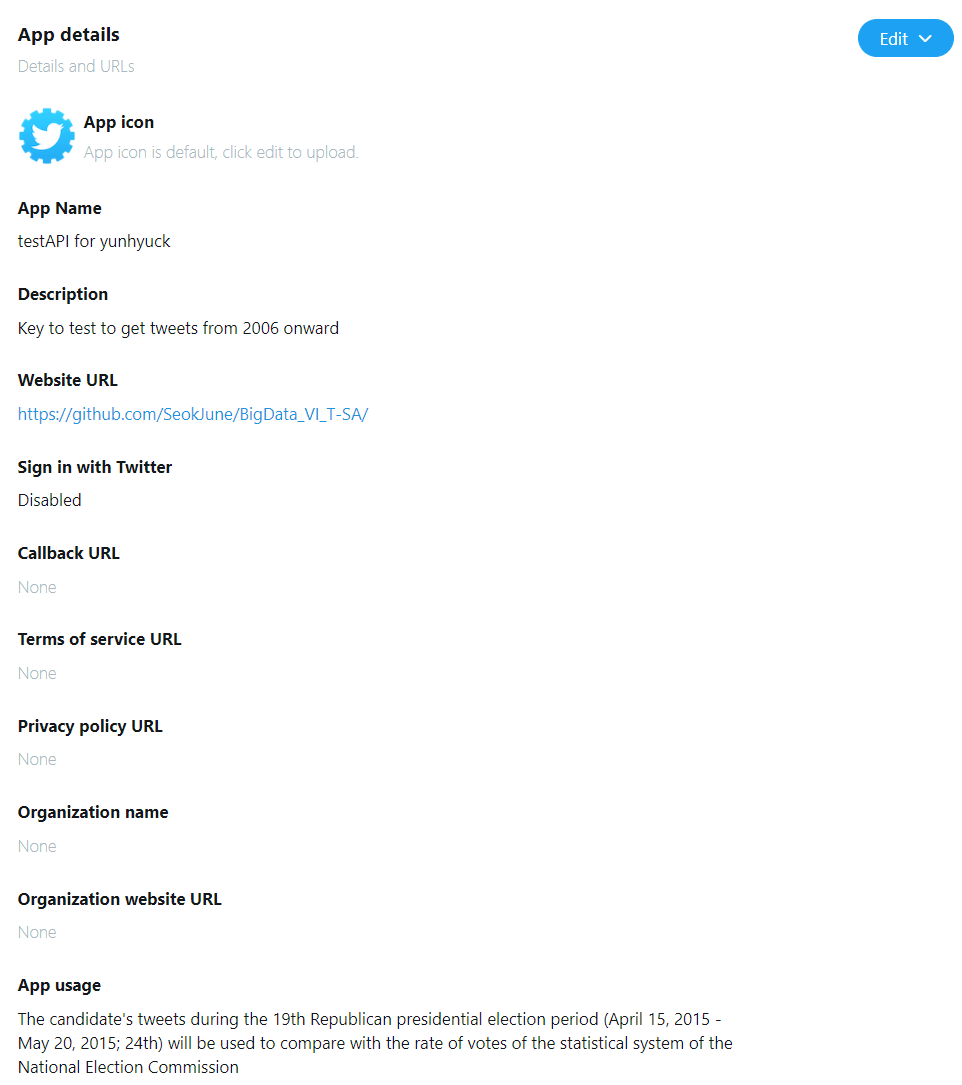
**어플리케이션 등록 및 키 발급**

개발자 등록이 완료가 된 상태에서 어플리케이션을 등록하여 준다. 이 과정에서는 사용할 앱의 이름과 사용 방법에 대해 기술을 하게 된다. 어플리케이션을 등록을 하기 위해서 그림 8과 같이 Apps를 클릭하여 앱을 생성하게 된다.



[그림 8] 개발자 등록 후, 사용할 수 있는 카테고리

본 프로젝트에서 사용하는 어플리케이션 이름은 테스트 API발급이라고 하였다. 테스트를 진행한 후, 새로운 키를 발급받아 제한된 사용량을 해결해 보고자 하였다. 그 다음 앱에 대한 설명을 작성을 해 주면 된다. 이 때, 글자수는 10~200자 사이로 영문으로 작성을 해주면 된다. 본 프로젝트의 경우에는 ‘2006년 이후의 트윗을 가져오기 위해 테스트 하는 키’라고 작성 하였다. 웹 URL을 작성 하는 곳에는 해당 프로젝트가 진행되고 있는 깃 허브(https://github.com/SeokJune/BigData\_VI\_T-SA/) 주소를 사용하였다. 마지막으로 앱의 사용 방법은 100자 이상으로 작성을 하면 된다. 본 프로젝트에서는 ‘한국의 19대 대통령 선거기간(2017.4.15 ~ 2017.05.09 ; 24일) 동안의 후보자들의 트윗을 가져와 중앙선거관리위원회의 통계시스템의 득표율과의 비교 분석에 사용된다’라고 작성 하였다. 최종 작성된 내용은 그림 9와 같다. 생성된 앱의 상세 정보 페이지에서 키를 확인 할 수 있다.



[그림 9] 생성된 테스트 API발급을 받기 위한 앱의 정보

**개발 환경 설정**

개발 환경 설정에 앞서 Twitter API의 트윗 검색을 위한 Search API에 대해 알아 보았다. 현재 기본적으로 제공을 해주는 버전은 Standard버전이다. Premium과 Enterprise 총 3가지로 분류를 할 수 있는데, 이는 표 1에 정리되어 있다.

|  |  |
| --- | --- |
| 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. |

(표 1) 트위터의 Search API의 버전

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 가져 오기 때문에, 본 프로젝트에서 진행하는 19대 대통령 선거 기간의 트윗의 내용을 가져오기 위해서는 적절하지 못한 것을 확인 하였다. Premium의 경우 지난 30일간 게시된 트윗을 제공해 주는 버전과 2006년 이후 게시된 트윗을 제공해 주는 버전으로 나뉘어져 있다. 가격 마다 차이가 있지만, 기본적으로 한달 동안 몇 번의 질의를 할 수 있는지 횟수를 통해 가격이 책정이 된다. 그림 10을 통해 2006년 이후의 데이터를 가져 올 수 있는 버전의 차이를 알 수 있다.

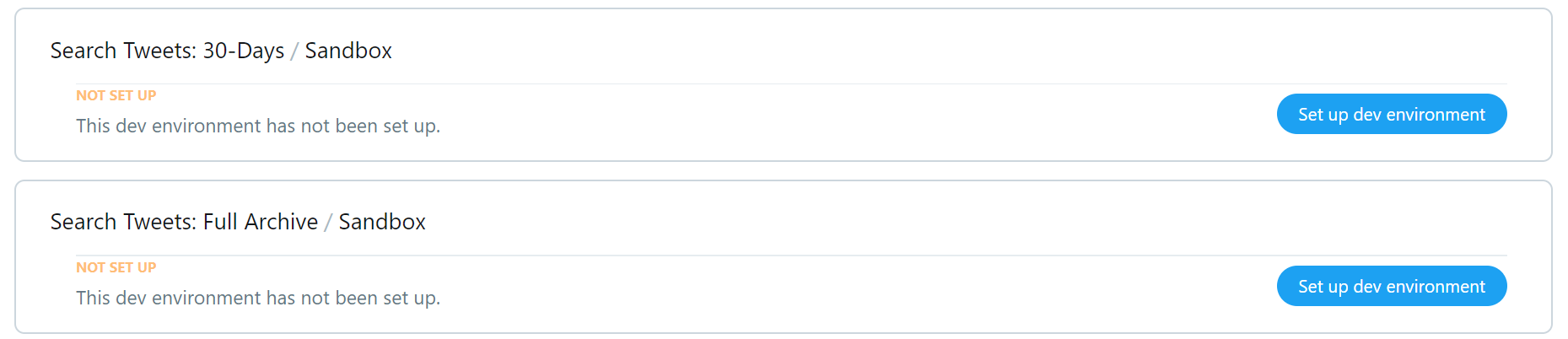


[그림 10] Sandbox와 Premium의 차이

2006년 이후의 데이터를 가져오는 Search API의 구성에는 Sandbox 버전과 Premium 버전으로 나뉘어져 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 128자 보다 많다면 …으로 표시가 되며, 한달 동안 100번의 질의를 수행할 수 있습니다. 그에 반해 Premium 버전의 경우 기본적으로 500번의 질의를 수행할 수 있으며, 트윗 내용의 1024자를 가져 올 수 있다.

하지만, 한국에서는 트위터 정책상 결제를 막아 두고 있다. 이는 결제 카드를 등록하는 과정에서 크롬에서 제공해 주는 개발자 모드를 통해 확인 하였다. 그에 대한 이유를 확인 하려고 하였지만, 트위터는 정책상 고객센터가 챗 봇을 통해 운영이 되고 있으며, 회사 전화 번호를 공개를 하지 않고 있기 때문에 알 수 가 없다. 그렇기 때문에 본 프로젝트는 Standard버전을 사용하며, Sandbox 버전의 Search API의 Full-archive모드를 사용하고자 한다.

다음 과정은 발급 받은 키에 대해 API를 사용하는 환경을 설정하는 과정이다. 개발 환경을 설정을 해주어야, OAuth인증을 통해 질의를 수행한 결과를 가져 올 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 웹사이트 상의 자신들의 정보에 대해 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 수단)의 기본에 충실하게 액세스 토큰을 가지고 와야 원하는 데이터를 조회할 수 있기 때문이다. 그림 8에서와 같이 Dev Environments 카테고리를 통해 환경을 설정 할 수 있다. 해당 카테고리를 접속하면 그림 11과 같다.



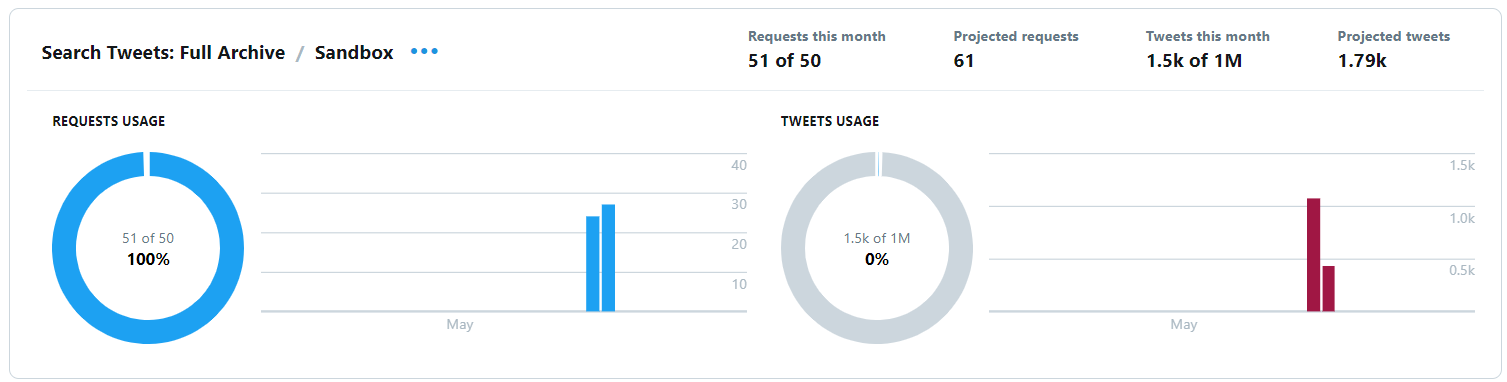
**최근 30일 트윗을 가져오기 위한 환경**

**2006년 이후의 트윗을 가져오기 위한 환경**

**(2017년 대선 기간의 데이터를 가져오기 위해 선택)**

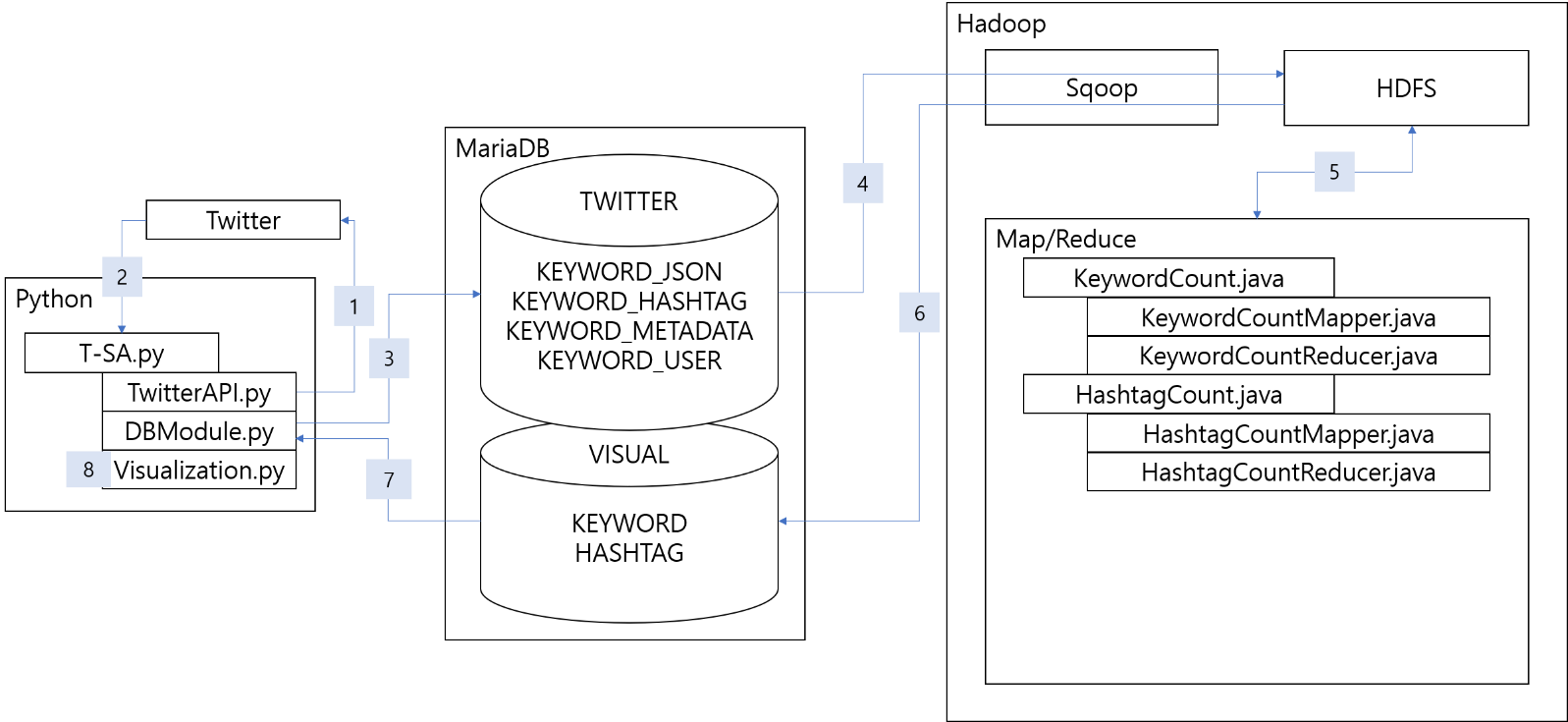
[그림 11] Dev Environments 화면

본 프로젝트는 19대 대통령 선거 기간의 데이터를 가져와야 하기 때문에 Full Archive의 개발환경 설정을 하였다. Set up dev environment를 클릭하여, json형식으로 제공받을 파일의 label과 사용할 키가 있는 앱의 이름을 선택해 주면 완료 된다. 설정이 완료 되면 DashBoard를 통해 사용량을 확인 할 수 있다. SandBox 버전의 경우 한달에 질의를 수행할 수 있는 횟수가 50번으로 제한이 되어 있기 때문에 주기적으로 확인을 해주어야 한다.



[그림 12] DashBoard에서 키의 사용량 확인

**제3절 T-SA 흐름도**



[1번, 2번]

TwitterAPI를 이용해서 정보(트윗 내용(작성시간, 트윗, 해시태그 등), 사용자 정보(아이디, 닉네임, 위치정보, 팔로우 수, 팔로잉 수, 언어 등)) 크롤링을 한다.

[3번]

크롤링된 데이터를 MariaDB에 저장한다.

[4번]

Sqoop을 이용하여 MariaDB에 저장된 데이터를 HDFS에 저장한다.

[5번]

HDFS에 업로드된 데이터를 Map/Reduce과정을 통해 정규화하고 결과를 HDFS에 저장한다.

- KeywordCount: 자연어 처리된 트윗에 포함된 단어(한글자 초과 다섯글자 이하)를 카운트한다.

- HashtagCount: 해시태그(다섯글자 이하)를 카운트한다.

[6번]

Sqoop을 이용하여 HDFS에 저장된 정규화된 데이터를 MariaDB에 저장한다.

[7번]

MariaDB에 저장된 데이터를 Python으로 불러온다.

[8번]

불러온 데이터를 시각화한다.

**제4절 구현(Python)**

**2.4.1. T-SA.py**

|  |
| --- |
| '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  T-SA.py  Title: T-SA의 기능 실행  Author: Lee SeokJune  Create on: 2019.04.08  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  import TwitterAPI  import dbModule  import Visualization  # TwitterAPI, dbModule 관련 변수 설정 ---------------------------------------------------------------  # 서재익  statTwitter = ('MGRK5IsX8xwxhz0FYv5Llm5ps', # consumer\_key  'JRh3fHqPqEq6VWcyoKax6MG4nE21z0zatiDjEGnvmHm99cyrLA', # consumer\_secret  '1103843008670121984-qw1ooMrZLzK10AcQkuixvvq0dizVfR', # access\_token  'dqplsyeDz5n7kkYB8kW6kIkDW7lPkoFnL3r4vpCR0brdJ') # access\_token\_secret  '''  # 이윤혁  statTwitter = ('lNZwPI2dQ5l89K1nOGW6Sod6u', # consumer\_key  'D6eGld20D99yrL89SMYPhJsjiHqmNKGL5LznkNKOQQPoIoQxWA', # consumer\_secret  '1107934597189263361-E83WGFw4XnDGpPkmYewJA7aIecHru6', # access\_token  'CvbR5ga31iWxQVrWcnzdnp7NGBbmAFGWRntjuZbXnpAet') # access\_token\_secret  '''  statKeyword = ('이윤혁', # keyword  '2019-04-25', # sinceDate  '2019-04-29', # untilDate  'extended', # mode  10) # count  statUserInfo = ('BWnYuiJ0vkWsATq') #screen\_name / @으로 시작하는 이름  twitterDB = ('localhost', # hostIP  'T-SA', # userID  '1234', # password  'TWITTER', # DB 종류  'utf8') # charset  visualDB = ('localhost', # hostIP  'T-SA', # userID  '1234', # password  'VISUAL', # DB 종류  'utf8') # charset  tableName = ('KEYWORD\_JSON',  'KEYWORD\_HASHTAG',  'KEYWORD\_METADATA',  'KEYWORD\_USER',  'USER\_JSON')  # TwitterAPI, dbModule 객체 생성 -------------------------------------------------------------------  twitter = TwitterAPI.TwitterAPI()  db = dbModule.dbModule(twitterDB[0],twitterDB[1],twitterDB[2],twitterDB[3],twitterDB[4])  db1 = dbModule.dbModule(visualDB[0],visualDB[1],visualDB[2],visualDB[3],visualDB[4])  visual = Visualization.Visualization()  # TwitterAPI의 OAuth 실행 -------------------------------------------------------------------------  api = twitter.OAuth(statTwitter[0], statTwitter[1], statTwitter[2], statTwitter[3])  # 작동부 -------------------------------------------------------------------------------------------  while True:  '''  기능 선택 입력 받기  1. KeyWord Search  2. User Search  3. Visualization  4. Exit  '''  print('1. Keyword Search')  print('2. User Search')  print('3. Visualization')  print('4. Exit')  print('Choice Number: ')  cNum = input()  # 입력 오류 체크 --------------------------------------------------------------------------------  if cNum not in ['1', '2', '3', '4']:  print('Re-enter')  continue  # 종료 실행(4) ---------------------------------------------------------------------------------  if cNum == '4':  print('Exit!!')  break  # 작업할 테이블명 입력 후 데이터가 존재할 시 삭제 ---------------------------------------------------  '''  Modifier: Bae InGyu  Modify on: 2019-04-09  작업할 테이블명 입력 후 데이터가 존재할 시 삭제하는 기능 추가  '''  if db.getRowByCheck(tableName[1]) == True:  print("========테이블에 존재4하는 데이터 삭제시작========")  db.deleteDB(tableName[1])  else:  print("========테이블에 존재하는 데이터 없음========")  '''  기능 실행(1, 2, 3)  '''  # Keyword Search 실행(1) -----------------------------------------------------------------------  if cNum == '1':  print('Keyword Search 시작')  # TwitterAPI.py 작업 -----------------------------------------------------------------------  tweets = twitter.search\_Keyword(api, statKeyword[0], statKeyword[1], statKeyword[2], statKeyword[3], statKeyword[4])  keyword\_Json, keyword\_Hashtags, keyword\_Metadata, keyword\_User = twitter.result\_Keyword(tweets)  # dbModule.py 작업 -------------------------------------------------------------------------  for val in keyword\_Hashtags:  db.insertDB('KEYWORD\_HASHTAG', val)  # -----------------------------------------------------------------------------------------  print('Keyword Search 완료')  # UserInfo Search 실행(2) --------------------------------------------------------------------------  elif cNum == '2':  print('UserInfo Search 시작')  # TwitterAPI.py 작업 -----------------------------------------------------------------------  userInfo = twitter.search\_User(api, statUserInfo[0])  user\_Json = twitter.result\_User(userInfo)  # dbModule.py 작업 -------------------------------------------------------------------------  # -----------------------------------------------------------------------------------------  print('User Search 완료')  # Visualization 실행(3) ------------------------------------------------------------------------  elif cNum == '3':  print('Visualization 시작')  # Visualization.py 작업 --------------------------------------------------------------------  hashtagCount = dict(db1.selectDB('HASHTAG'))  visual.visualize(hashtagCount)  # -----------------------------------------------------------------------------------------  print('Visualization 완료') |

**2.4.2. TwitterAPI.py**

|  |
| --- |
| '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  TwitterAPI.py  Title: TwitterAPI의 어플리케이션 인증, 데이터 수집(search, getuser), db저장 할 형식으로 변환  Author: Lee SeokJune  Create on: 2019.04.17 09:42  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  import tweepy  class TwitterAPI:  # Twitter Application 인증 -> api -------------------------------------------------------------  def OAuth(self, consumer\_key, consumer\_secret, access\_token, access\_token\_secret):  auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)  auth.set\_access\_token(access\_token, access\_token\_secret)  api = tweepy.API(auth)  return api  # Keyword Search in Tweet --------------------------------------------------------------------  def search\_Keyword(self, api, keyword, sinceD, untilD, mode, count):  tweets = []  try:  for tweet in tweepy.Cursor(api.search, q = keyword, since = sinceD, until = untilD, tweet\_mode = mode,count = count).items():  tweets.append(tweet)  return tweets  except tweepy.error.TweepError:  print("Tweet Per Minute")  # User Search --------------------------------------------------------------------------------  def search\_User(self, api, screen\_name):  userInfo = api.get\_user(screen\_name)  return userInfo  # result Keyword -----------------------------------------------------------------------------  def result\_Keyword(self, tweets):  keywordJson = [] # Table(KEYWORD\_JSON)  keywordHashtags = [] # Table(KEYWORD\_HASHTAGS)  keywordMetadata = [] # Table(KEYWORD\_METADATA)  keywordUser = [] # Table(KEYWORD\_J\_USER)  keyNum = 0  # tweet 하나씩 가져오기 --------------------------------------------------------------------  for t in tweets:  # \_json 선택 -------------------------------------------------------------------------  json = t.\_json  # Table(KEYWORD\_JSON) ----------------------------------------------------------------  keywordJson.append([json['created\_at'],  json['id\_str'],  json['text'],  json['truncated'],  str(keyNum + 1).zfill(4),  str(keyNum + 1).zfill(4),  str(keyNum + 1).zfill(4),  json['retweet\_count'],  json['favorite\_count'],  json['lang']])  # Table(KEYWORD\_HASHTAGS) ------------------------------------------------------------  hashtags = json['entities']['hashtags']  for h in hashtags:  keywordHashtags.append([str(keyNum + 1).zfill(4),  h['text'],  h['indices']])  # Table(KEYWORD\_METADATA) ------------------------------------------------------------  metadata = json['metadata']  keywordMetadata.append([str(keyNum + 1).zfill(4),  metadata['iso\_language\_code'],  metadata['result\_type']])  # Table(KEYWORD\_USER) ----------------------------------------------------------------  user = json['user']  keywordUser.append([str(keyNum + 1).zfill(4),  user['id'],  user['id\_str'],  user['name'],  user['screen\_name'],  user['location'],  user['description'],  user['url'],  user['protected'],  user['followers\_count'],  user['friends\_count'],  user['listed\_count'],  user['created\_at'],  user['favourites\_count'],  user['utc\_offset'],  user['time\_zone'],  user['geo\_enabled'],  user['verified'],  user['statuses\_count'],  user['lang'],  user['contributors\_enabled'],  user['is\_translator'],  user['is\_translation\_enabled'],  user['following'],  user['follow\_request\_sent'],  user['notifications'],  user['translator\_type']])  # key 증가 ----------------------------------------------------------------------------  keyNum += 1  return keywordJson, keywordHashtags, keywordMetadata, keywordUser |

**2.4.3. DBModule.py**

|  |
| --- |
| '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  DBModule.py  Title: MariaDB연결,종료 및 DML 작업  Author: Bae InGyu  Create\_at: 2019.04.02.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  import pymysql  class DBModule:  # 클래스의 생성자----------------------------------------------------------------------------  def \_\_init\_\_(self, host, user, pswd, db, charset):  self.host = host  self.user = user  self.pswd = pswd  self.db = db  self.charset = charset  # MariaDB연결함수----------------------------------------------------------------------------  def dbConnect(self):  conn = pymysql.connect(host = self.host, user = self.user, password = self.pswd, db = self.db, charset = self.charset)  curs = conn.cursor()  return conn, curs  # MariaDB연결종료함수-------------------------------------------------------------------------  def dbClose(self):  conn, curs = self.dbConnect()  return conn.close(), curs.close()  # 모든테이블에 ROW 여부파악 함수------------------------------------------------------------  '''  Modifier: Bae InGyu  Modify on: 2019-04-17  기존의 하나의 테이블만 ROW 여부파악에서 모든 테이블의 ROW 여부파악으로 수정  '''  def getRowByCheck(self,tableName):  try:  # MariaDB연결 및 Cursor생성  conn, curs = self.dbConnect()  # 테이블 조회  result = []  i = 0  for table in tableName:  sql = "select \* from " + table.strip() + ";"  curs.execute(sql)  # 테이블 데이터출력  result.append(curs.fetchall())  print(result[i])  i += 1  # 모든 테이블 ROW 여부파악 하나의 테이블이라도 ROW가 존재하면 True로 반환  if bool(result[0] or result[1] or result[2] or result[3] or result[4]) == True:  return True  except:  print("파악실패")  finally:  # Select후 Cursor종료 및 MariaDB연결종료  self.dbClose()  # 테이블의 데이터 모두조회함수----------------------------------------------------------------  def selectDB(self,tableName):  try:  # MariaDB연결 및 Cursor생성  conn, curs = self.dbConnect()  # 테이블 조회  sql = "select \* from " + tableName.strip() + ";"  curs.execute(sql)  # 테이블 데이터출력  rows = curs.fetchall()  result = rows  return result  except:  print("조회실패")  finally:  # Select후 Cursor종료 및 MariaDB연결종료  self.dbClose()  # 모든테이블의 데이터 모두삭제함수----------------------------------------------------------------  '''  Modifier: Bae InGyu  Modify on: 2019-04-17  기존의 하나의 테이블의 ROW만 삭제에서 모든 테이블의 ROW 삭제로 변경  '''  def deleteDB (self,tableName):  try:  # MariaDB연결 및 Cursor생성  conn, curs = self.dbConnect()  # Data삭제  for table in tableName:  sql = "delete from " + table.strip() + ";"  curs.execute(sql)  conn.commit()  print("삭제완료")  except:  print("삭제실패")  finally:  # Cursor종료 및 MariaDB연결종료  self.dbClose()  # 테이블의 데이터 삽입함수--------------------------------------------------------------------  def insertDB (self, tableName, values):  try:  # MariaDB연결 및 Cursor생성  conn, curs = self.dbConnect()  # Data삽입  sql = "insert into " + tableName.strip() + " values("  sql += "'" + values[0] + "',"  sql += str(values[2][0]) + ","  sql += str(values[2][1]) + ","  sql += "'" + values[1] + "');"  curs.execute(sql)  conn.commit()  print("삽입완료")  except:  print("삽입실패")  finally:  #Cursor종료 및 MariaDB연결종료  self.dbClose() |

**2.4.4. Visualization.py**

|  |
| --- |
| from wordcloud import WordCloud  import matplotlib.pyplot as plt  import matplotlib.font\_manager as fm  class Visualization:  def visualize(self, b):  # 한글폰트 적용  path = '/home/vi/.local/lib/python3.6/site-packages/matplotlib/mpl-data/fonts/ttf/NanumBarunGothicUltraLight.ttf'  fontprop = fm.FontProperties(fname=path, size=18)  # 워드 클라우드 설정  wc=WordCloud(font\_path=path,background\_color='white',max\_words=2000)  wc=wc.generate\_from\_frequencies(b)  # 시각화 이미지 설정  plt.figure(figsize=(12,12))  plt.imshow(wc, interpolation='bilinear')  plt.axis('off')  plt.show() |

**제5절 DB/Table 정의서**

|  |  |  |
| --- | --- | --- |
| DB | Table | Comment |
| TWITTER | KEYWORD\_JSON |  |
| KEYWORD\_HASHTAG |  |
| KEYWORD\_METADATA |  |
| KEYWORD\_USER |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| VISUAL | KEYWORD | 키워드 카운트 |
| HASHTAG | 해쉬태그 카운트 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | TWITTER.KEYWORD\_JSON | | | | | |
| Description | | 키워드 검색 결과 중 대략적인 정보 저장 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | CREATED\_AT | | 작성일자 및 시간 | DATETIME | X | PK |  |
| 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 | METADATA | | 메타데이터 | CHAR(4) | X |  |  |
| 07 | USER | | 작성자 | CHAR(4) | X |  |  |
| 08 | RETWEET\_COUNT | | 리트윗 수 | INT | X |  |  |
| 09 | FAVORITE\_COUNT | | 좋아요 수 | INT | X |  |  |
| 10 | LANG | | 언어 | VARCHAR(3) | X |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | TWITTER.KEYWORD\_HASHTAG | | | | | |
| Description | | 키워드 검색 결과 중 해시태그 정보 저장 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | HCODE | | 코드 | CHAR(4) | X | PK |  |
| 02 | START | | 시작 위치 | INT | X | PK |  |
| 03 | END | | 종료 위치 | INT | X | PK |  |
| 04 | TEXT | | 해시태그 | TEXT | X |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | TWITTER.KEYWORD\_METADATA | | | | | |
| Description | | 키워드 검색 결과 중 메타데이터 정보 저장 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | MCODE | | 코드 | CHAR(4) | X | PK |  |
| 02 | ISO\_LANGUAGE\_CODE | | 작성 장치의 언어 | VARCHAR(3) | X |  | ISO 639 언어 분류 |
| 03 | RESULT\_TYPE | | 반환 타입 | ENUM | X |  | mixed, recent, popular |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | TWITTER.KEYWORD\_USER | | | | | |
| Description | | 키워드 검색 결과 중 작성자 정보 저장 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | UCODE | | 코드 | CHAR(4) | X | PK |  |
| 02 | ID | | 작성자 고유번호 | VARCHAR(20) | X |  |  |
| 03 | NAME | | 작성자 이름 | VARCHAR(50) | X |  |  |
| 04 | SCREEN\_NAME | | 작성자 이름(@) | VARCHAR(50) | X |  |  |
| 05 |  | |  |  |  |  |  |
| 06 |  | |  |  |  |  |  |
| 07 |  | |  |  |  |  |  |
| 08 |  | |  |  |  |  |  |
| 09 |  | |  |  |  |  |  |
| 10 |  | |  |  |  |  |  |
| 11 |  | |  |  |  |  |  |
| 12 |  | |  |  |  |  |  |
| 13 |  | |  |  |  |  |  |
| 14 |  | |  |  |  |  |  |
| 15 |  | |  |  |  |  |  |
| 16 |  | |  |  |  |  |  |
| 17 |  | |  |  |  |  |  |
| 18 |  | |  |  |  |  |  |
| 19 |  | |  |  |  |  |  |
| 20 |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | VISUAL.KEYWORD | | | | | |
| Description | | 키워드 카운트 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | KEYWORD | | 단어 | VARCHAR(10) | X | PK |  |
| 02 | COUNT | | 횟수 | INT | X |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | | VISUAL.HASHTAG | | | | | |
| Description | | 해시태그 카운트 테이블 | | | | | |
| No | Column Name | | | Type | NULL | Key | Comment |
| 01 | HASHTAG | | 해시태그 | VARCHAR(10) | X | PK |  |
| 02 | COUNT | | 횟수 | INT | X |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |

**제6절(Hadoop)**

**2.6.1. Sqoop(DB/HDFS 연동)**

**MariaDB » HDFS**

|  |
| --- |
| 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 |

**HDFS » MariaDB**

|  |
| --- |
| 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  export-dir: 데이터를 가져올 HDFS 디렉토리 경로  columns: 테이블에서 매핑될 컬럼 리스트  input-fields-terminated-by: 구분자 |

**2.6.2. Map/Reduce(.java)**

**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. Hadoop 실행(.jar)**

|  |
| --- |
| 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 결 론

**제1절 참고문헌 및 사이트**

[1] Sean Bradley, “트위터, 월 활동 사용자 통계 발표 중단”, ITWORLD, 2019년 2월 11일자.

[2] 위키피디아, JSON <https://ko.wikipedia.org/wiki/JSON>

<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/>

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

-부 록 1-

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