

# 중간 보고서



## T-SA: Twitter keyword Search API based tweet Analysis (트위터 키워드 검색 API기반 트윗 분석)

CHOSUN  
UNIVERSITY  
1946



과 목: 산학캡스톤디자인1

분 반: 01

담당 교수: 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
2.6.3. Hadoop 실행(.jar)	25

## 제3장 결 론

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

## -부 록-

1. 부 록 1 (깃허브)

## -별 첨-

1. Hadoop 설치.odt
2. sqoop1.docx
3. Demonstrate Video (CAP20190429.mp4)

# 제1장 서론


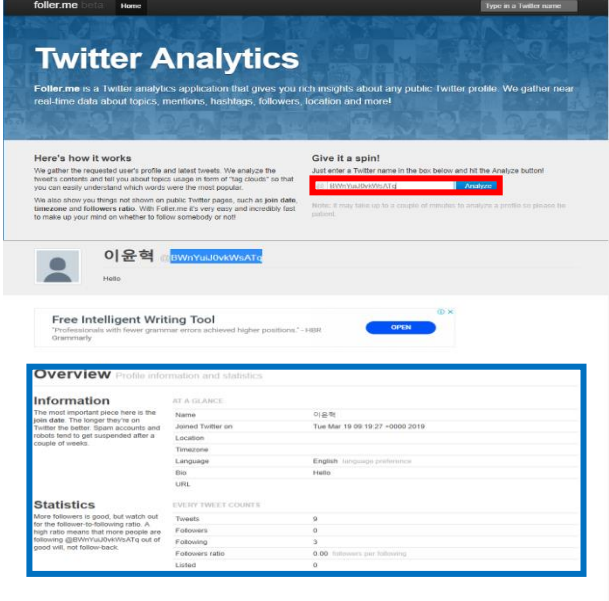
## 제1절 개발 동기 및 필요성

SNS(Social Networking Service)는 사용자 간의 자유로운 의사소통, 정보공유 그리고 인맥 확대 등을 통하여 사회적 관계를 생성하고 강화해주는 온라인 플랫폼을 말한다. 대표적인 SNS로는 페이스북, 인스타그램, 블로그, 트위터 등이 존재한다. 그중에서도 기업, 공공기관, 정부 기관과 같은 단체뿐만 아니라 연예인, 국회의원, 대통령 등 공인이 다른 SNS보다 상대적으로 많이 사용하는 트위터를 선택하게 되었다.

단체가 작성한 트윗, 팔로잉 등을 분석하여 그 단체의 슬로건, 비전, 성향, 인재상 등 다양한 정보를 예상하거나 수집하는데 이용할 수 있을 거라 생각한다.

트위터 대한민국 @TwitterKorea	Tweets 3,044	Following 99	Followers 1.13M	Likes 1,402
대한민국 정부 @hellopolicy	Tweets 20.6K	Following 37.9K	Followers 171K	Likes 371
최재성(더불어민주당 송파 을 국회의원) ● @wtstg21	Tweets 2,840	Following 43.2K	Followers 119K	Likes 247
삼성 라이온즈 @twittions	Tweets 2,697	Following 38	Followers 236K	Likes 16
한국전력공사(KEPCO) @iamkepc	Tweets 5,574	Following 17.2K	Followers 20K	Likes 136
MAMA(영넷아시아뮤직어 워즈) ● @mamanetk	Tweets 8,999	Following 294	Followers 1.61M	Likes 5
더불어민주당 @TheMinjoo_Kr	Tweets 44.8K	Following 85.6K	Followers 253K	Likes 1,409
대한민국 청와대 @TheBlueHouseKR	Tweets 4,952	Following 32.8K	Followers 405K	Likes 9
문재인 @moonriver365	Tweets 3,049	Following 153K	Followers 1.76M	Likes 729

## 제2절 기존 연구

 <p>URL: <a href="http://tweettrend.com/">http://tweettrend.com/</a>          Input: 기간, 단어          Output: 기간별 언급 횟수 그래프, 트윗정보</p>	 <p>URL: <a href="https://foller.me/">https://foller.me/</a>          Input: 사용자 아이디          Output: 사용자의 정보</p>
--	--

## 제3절 개발 목표

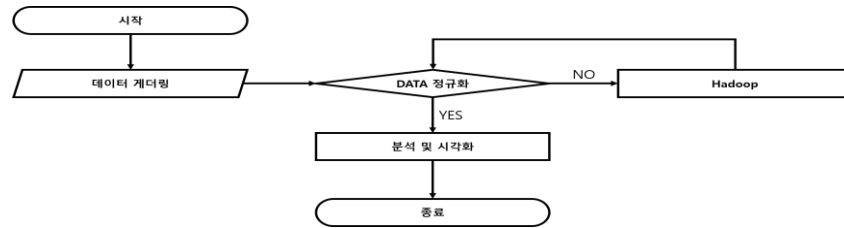
이 프로젝트에서는 개발 동기 및 필요성과 기존 연구를 토대로 2가지의 목표를 정하였다.

첫 번째로는, 키워드와 기간을 설정하여 해당 기간동안 작성된 트윗을 가져와서 포함된 단어와 해시태그의 빈도를 분석하고 시각화할 것이다.

두 번째로는, 사용자의 아이디 혹은 이름을 통해 해당 사용자의 정보를 가져와서 분석하고 시각화할 것이다.

## 제4절 시스템 명세

### 1.4.1. 시스템 동작 과정



### 1.4.2. 원본 데이터 SET

```

Status(api=<tweepy.api.API object at 0x7f6fd80f87b8>, json={'created_at': 'Sun Apr 28 08:12:14 +0000 2019', 'id': 1122413198013911040, 'id_str': '1122413198013911040', 'text': 'aaaaaaaa 이윤혁', 'truncated': False, 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': []}, 'metadata': {'result_type': 'recent', 'iso_language_code': 'ko'}, 'source': '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>', 'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': None, 'in_reply_to_screen_name': None, 'user': {'id': 1107113829123555328, 'id_str': '1107113829123555328', 'name': 'Lee SeokJune', 'screen_name': 'LSeokJune', 'location': '', 'description': '', 'url': None, 'entities': {'description': {'urls': []}}, 'protected': False, 'followers_count': 0, 'friends_count': 0, 'listed_count': 0, 'created_at': 'Sun Mar 17 02:58:00 +0000 2019', 'favourites_count': 0, 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'verified': False, 'statuses_count': 8, 'lang': 'ko', 'contributors_enabled': False, 'is_translator': False, 'is_translation_enabled': False, 'profile_background_color': 'F5F8FA', 'profile_background_image_url': None, 'profile_background_image_url_https': None, 'profile_background_tile': False, 'profile_image_url': 'http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', 'profile_image_url_https': 'https://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', 'profile_link_color': '1DA1F2', 'profile_sidebar_border_color': 'CODEED', 'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333', 'profile_use_background_image': True, 'has_extended_profile': False, 'default_profile': True, 'default_profile_image': True, 'following': False, 'follow_request_sent': False, 'notifications': False, 'translator_type': 'none'}, 'id': 1107113829123555328, 'id_str': '1107113829123555328', 'name': 'Lee SeokJune', 'screen_name': 'LSeokJune', 'location': '', 'description': '', 'url': None, 'entities': {'description': {'urls': []}}, 'protected': False, 'followers_count': 0, 'friends_count': 0, 'listed_count': 0, 'created_at': 'Sun Mar 17 02:58:00 +0000 2019', 'favourites_count': 0, 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'verified': False, 'statuses_count': 8, 'lang': 'ko', 'contributors_enabled': False, 'is_translator': False, 'is_translation_enabled': False, 'profile_background_color': 'F5F8FA', 'profile_background_image_url': None, 'profile_background_image_url_https': None, 'profile_background_tile': False, 'profile_image_url': 'http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', 'profile_image_url_https': 'https://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', 'profile_link_color': '1DA1F2', 'profile_sidebar_border_color': 'CODEED', 'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333', 'profile_use_background_image': True, 'has_extended_profile': False, 'default_profile': True, 'default_profile_image': True, 'following': False, 'follow_request_sent': False, 'notifications': False, 'translator_type': 'none'}, id=1107113829123555328, id_str='1107113829123555328', name='Lee SeokJune', screen_name='LSeokJune', location='', description='', url=None, entities={'description': {'urls': []}}, protected=False, followers_count=0, friends_count=0, listed_count=0, created_at=datetime.datetime(2019, 3, 17, 2, 58), favourites_count=0, utc_offset=None, time_zone=None, geo_enabled=False, verified=False, statuses_count=8, lang='ko', contributors_enabled=False, is_translator=False, is_translation_enabled=False, profile_background_color='F5F8FA', profile_background_image_url=None, profile_background_image_url_https=None, profile_background_tile=False, profile_image_url='http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', profile_image_url_https='https://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png', profile_link_color='1DA1F2', profile_sidebar_border_color='CODEED', profile_sidebar_fill_color='DDEEF6', profile_text_color='333333', profile_use_background_image=True, has_extended_profile=False, default_profile=True, default_profile_image=True, following=False, follow_request_sent=False, notifications=False, translator_type='none'}, geo=None, coordinates=None, place=None, contributors=None, is_quote_status=False, retweet_count=0, favorite_count=0, favorited=False, retweeted=False, lang='ko')
  
```

- TwitterAPI 중 Search메서드의 결과값 중 하나의 Row에 해당하는 값

## 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	3.2.0
Sqoop	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(communitiy)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.x는 레거시(낡은 기술)이고, 파이썬 3.x가 파이썬의 현재와 미래가 될 것”이라고 요약했는데, 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
```

## 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 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 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 설정되므로 root계정으로 실행했다면 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 설정하고 싶으면 Y, root 그대로 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 삭제할지 여부(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-server = utf8mb4
# collation-server = utf8mb4_general_ci
collation-server = utf8mb4_unicode_ci
```

utf8mb4\_general\_ci(default)을 사용할 경우 정렬 속도가 utf8mb4\_unicode\_ci에 비해 약간 빠르긴 하지만 거의 차이는 없음, 한글/일본어와 같이 비 라틴계 언어들의 정렬이 조금 어색한 경우가 있기 때문에 utf8mb4\_unicode\_ci 사용

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

- Client에 대한 Character sets 설정 → /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 디벨로퍼(WebSphere 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 Eclipse 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.

스콕(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. 스콕은 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 TwitterAPI

### 플랫폼 종류

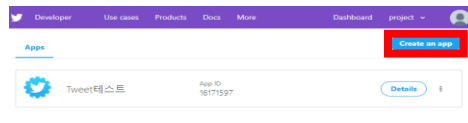
Standard	지난 7일간 게시된 최근 트윗을 제공한다. 공개 API 집합의 일부이다.
Premium	30-days: 지난 30일간 게시된 트윗을 제공한다. Full-archive: 2006년부터 게시된 트윗을 제공한다.
Enterprise	Premium과 같이 두 가지를 제공하는데 기업에서 주로 사용한다.

### Standard와 Premium의 상세 스펙

Category	Standard	Premium	
Product name	<u>Standard</u> Search API	Search Tweets: <u>30-day</u> endpoint	Search Tweets: <u>Full-archive</u> endpoint
Supported history	7 days	30 days	Tweets from as early as 2006
Counts endpoint	Not available	Available	Available
Data fidelity	Incomplete	Full	Full
Tweets per request	-	500	500
Query length	-	1024 characters	1024 characters
Operator availability	-	Premium	Premium
Rate limit per second	-	10 requests/sec	10 requests/sec
Rate limit per minute	-	60 requests/min	60 requests/min
Enrichments	-	URLs, Polls, Profile Geo	URLs, Polls, Profile Geo
Dev environments	-	2	2
Price	Free	\$50	\$99

### 어플리케이션 등록 및 키 발급

- 1) <https://developer.twitter.com/> 접속 » Project » Apps » Create an app



- 2) App name, Application description, Website URL, Tell us how this app will be used 작성

**App name**

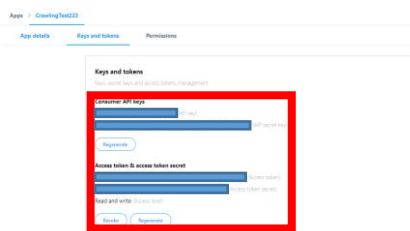
**Application description**

**Website URL**

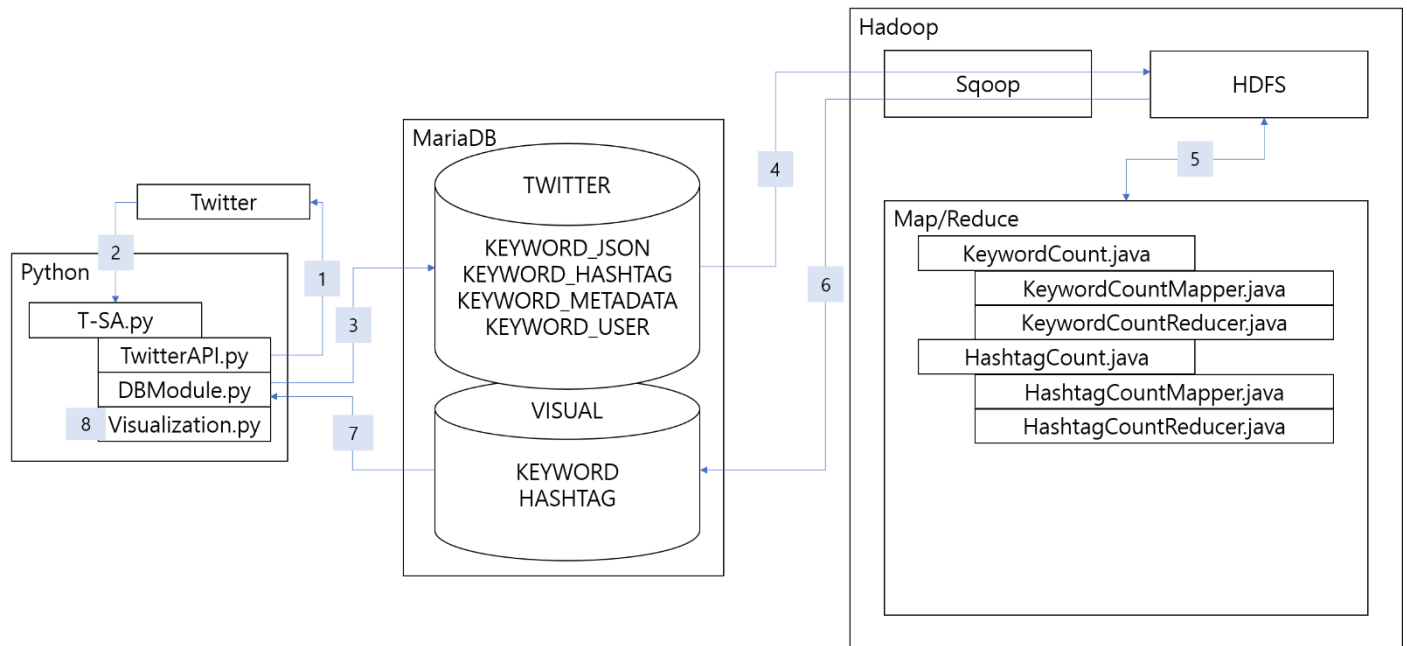
**Tell us how this app will be used (required)**

Cancel Create app

- 3) 앱이 등록되었다면, Keys and tokens를 눌러서 API 키와 Access token을 발급받는다.  
- 다시 받고 싶다면 'Regenerate'를 눌러서 다시 새롭게 받을 수 있다.



### 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 = ('MGRK5lsX8xwxhz0FYv5Llm5ps',          # consumer_key
               'JRh3fHqPqEq6VWcyoKax6MG4nE21z0zatiDjEGnvmHm99cyrLA', # consumer_secret
               '1103843008670121984-qw1ooMrZLzK10AcQkuixvq0dizVfR',   # access_token
               'dqplsyeDz5n7kkYB8kW6klkDW7IPkoFnL3r4vpCR0brdJ')       # access_token_secret
...
# 이윤혁
statTwitter = ('INZwPI2dQ5l89K1nOGW6Sod6u',          # consumer_key
               'D6eGld20D99yrL89SMYPHJsjiHqmNKG5LznkNKOQQPoloQxWA',   # consumer_secret
               '1107934597189263361-E83WGFw4XnDGpPkmYewJA7alecHru6',   # access_token
               'CvbR5ga3liWxQVrWcnzdnP7NGBbmAFGW RntjuZbXnpAet')       # 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()

    # 모든 테이블에 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/vil/.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.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	RE-TWEET_COUNT	리트윗 수	INT	X		
09	FAVORITE_COUNT	좋아요 수	INT	X		
10	LANG	언어	VARCHAR(3)	X		





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						



[illegible]

## 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 값들이 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.komorant.constant.DEFAULT_MODEL;
import kr.co.shineware.nlp.komorant.core.Komorant;
import kr.co.shineware.nlp.komorant.model.KomorantResult;
// 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\\r\\.\\\"'-=%...0[]\">+@!?:#\"");
        // 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
            Komorant komorant = new Komorant(DEFAULT_MODEL.FULL);
            // 사용자 사전 경로 추가.(사용자 명사 정의 가능)
            komorant.setUserDic("/home/vi/eclipse-workspace/KeywordCount/src/dic.user");
            // 읽어들인 단어 분석
            KomorantResult analyzeResultList = komorant.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.
            Iterator<String> itr2 = tokens.iterator();
            // -----
            // context 객체는 키-값쌍으로 내보낼 때 사용되며, 출력타입으로 인자화 된다.
            while(itr2.hasNext()) {
                // ktr 변수 생성하여 자연어 처리된 단어 저장.(단어, 공백 제거)
                String ktr = itr2.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 제정의
    // 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 입력력 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 (볼러올 데이터의 HDFS 경로) (저장될 데이터의 HDFS 경로)

# 제3장 결 론

# 제1절 참고문헌 및 사이트

Ubuntu Site: <https://www.ubuntu.com/>

Twitter Developer Site: <https://developer.twitter.com>

Tweepy Site: <http://www.tweepy.org>

Twitter Analysis Site (1): <http://tweetrend.com/>

Twitter Analysis Site (2): <https://foller.me/>

Python Site: <http://www.python.org/>

MariaDB Site: <https://mariadb.com/kb/ko/mariadb>

Eclipse Site: <http://www.eclipse.org/>

OpenJDK Site: <https://openjdk.java.net/>

Hadoop Site: <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/)

BigData\_VI\_T-SA /

You've activated the file finder. Start typing to filter the file list. Use **↑** and **↓** to navigate, **enter** to view files, **esc** to exit.

00.Doc\01.제출중 문서\4조\_브이아이.docx  
00.Doc\01.제출중 문서\4조\_브이아이.pptx  
00.Doc\01.제출중 문서\info  
00.Doc\01.제출중 문서\w04\4조\_브이아이.pdf  
00.Doc\01.제출중 문서\w04\4조\_브이아이.pptx  
00.Doc\01.제출중 문서\w05\4조\_브이아이.pdf  
00.Doc\01.제출중 문서\w05\4조\_브이아이.pptx  
00.Doc\01.제출중 문서\w06\4조\_브이아이.mp4  
00.Doc\01.제출중 문서\w06\4조\_브이아이.pdf  
00.Doc\01.제출중 문서\w06\4조\_브이아이.pptx  
00.Doc\01.제출중 문서\w08\4조\_보고서.docx  
00.Doc\02.대모\CAP20190429.mp4  
00.Doc\02.대모\info  
00.Doc\03.참고 문서\API Object 정리.docx  
00.Doc\03.참고 문서\API 정리.docx  
00.Doc\03.참고 문서\DB\08\_Table 정의서.docx  
00.Doc\03.참고 문서\VISUAL\KEYWORD\HASHTAG\DDL.odt  
00.Doc\03.참고 문서\DB\개정 생성 및 권한부여.odt  
00.Doc\03.참고 문서\FC.pptx  
00.Doc\03.참고 문서\python으로 MariaDB연결 DML작업 함수설정.docx  
00.Doc\03.참고 문서\result.odt  
00.Doc\03.참고 문서\원료\03.MariaDB 설치 및 환경 설정.docx  
00.Doc\03.참고 문서\원료\Hadoop 설치.odt  
00.Doc\03.참고 문서\원료\sqoop1.docx  
00.Doc\03.참고 문서\트위터 API 발급 방법.pptx  
00.Doc\03.참고 문서\트위터에 대해(수정2).docx  
00.Doc\03.참고 문서\학습 설치 과정.odt  
00.Doc\info  
01.Environment\00.필수확인 - 각각의 폴더에 환경변수 파일 등 수정한 파일 올리고 info에 검토와 그 파일의 간략한 정보 입력\info  
01.Environment\01.Python(3.6.8)\info  
01.Environment\02.MariaDB(XXX)\info  
01.Environment\03.OpenDK(1.8.0\_XXX)\info  
01.Environment\04.Eclipse(2019-03(4.10))\info  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\bashrc  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\01.Hadoop(3.2.0)\core-site.xml  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\01.Hadoop(3.2.0)\hadoop-env.sh  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\01.Hadoop(3.2.0)\hdfs-site.xml  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\01.Hadoop(3.2.0)\mapred-site.xml  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\01.Hadoop(3.2.0)\yam-site.xml  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\02.Sqoop(1.4.7)\commons-lang-2.6.jar  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\02.Sqoop(1.4.7)\mysql-connector-java-5.1.38-bin.jar  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\02.Sqoop(1.4.7)\sqoop-env.sh  
01.Environment\05.Hadoop(3.2.0).Sqoop(1.4.7)\info  
02.Code\01.Python\T-SA.py  
02.Code\01.Python\Visualization.py  
02.Code\01.Python\dbModule.py  
02.Code\01.Python\twitterAPI.py  
02.Code\02.Hadoop\01.KeywordCount\KOMORAN.jar  
02.Code\02.Hadoop\01.KeywordCount\KeywordCount.jar  
02.Code\02.Hadoop\01.KeywordCount\KeywordCount.java

SeokJune / BigData\_VI\_T-SA

Watch 0 Unstar 3 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

Branch: master

Commits on May 1, 2019

Delete ~\$sqoop1.docx  
yunhyuck committed 31 minutes ago  
Create ~\$sqoop1.docx  
yunhyuck committed 31 minutes ago  
Update 4조 보고서.docx  
SeokJune committed 32 minutes ago

Commits on Apr 30, 2019

Update KeywordCount.java  
yunhyuck committed 15 hours ago  
Update HashtagCount.java  
yunhyuck committed 15 hours ago  
Update KeywordCount.java  
yunhyuck committed 15 hours ago  
Update 4조 보고서.docx  
SeokJune committed 20 hours ago  
Create FC.pptx  
SeokJune committed 20 hours ago  
Delete ~\$ 설치 과정.odt  
SeokJune committed 20 hours ago  
Update 4조 보고서.docx  
SeokJune committed 21 hours ago  
Doc 수정  
SeokJune committed 21 hours ago  
Update 4조 보고서.docx  
SeokJune committed 22 hours ago  
보고서 위치 수정  
SeokJune committed a day ago

Commits on Apr 29, 2019

Create 보고서.docx  
SeokJune committed 2 days ago  
Create CAP20190429.mp4  
yunhyuck committed 2 days ago  
Add files via upload  
yunhyuck committed 2 days ago  
Create Visualization.py  
yunhyuck committed 2 days ago  
Create TwitterAPI.py  
yunhyuck committed 2 days ago  
Create T-SA.py  
yunhyuck committed 2 days ago  
Create dbModule.py  
yunhyuck committed 2 days ago

Commits on Apr 28, 2019

Update T-SA.py  
BaelnGyu committed 3 days ago  
Update dbModule.py  
BaelnGyu committed 3 days ago  
Add files via upload  
nero8879 committed 3 days ago  
Update KeywordCountMapper.java  
yunhyuck committed 3 days ago  
Update KeywordCount.jar  
yunhyuck committed 3 days ago  
Add files via upload  
yunhyuck committed 3 days ago  
Delete KeywordCountMapper.java  
yunhyuck committed 3 days ago  
Update KeywordCountMapper.java  
yunhyuck committed 3 days ago  
Update KeywordCountMapper.java  
yunhyuck committed 3 days ago  
Create KeywordCountMapper.java  
yunhyuck committed 3 days ago  
Create dic.user  
yunhyuck committed 3 days ago  
Update HashtagCountReducer.java  
yunhyuck committed 3 days ago  
Update HashtagCountMapper.java  
yunhyuck committed 3 days ago  
Update HashtagCountMapper.java  
yunhyuck committed 3 days ago  
Create HashtagCountReducer.java  
yunhyuck committed 3 days ago

Newer Older