

Spark Streaming 실습



Spark Streaming으로 Twitter API의 해시태그 분석하기

: Twitter에서 kpop으로 필터링 한 데이터를 실시간으로 가져와 해시태그(#) 분석해보기

[목차]

[실습 결과 미리 보기](#)

[A. 준비단계](#)

[A-1. 도커 이미지 설명](#)

[A-2. 실습 파일 설명](#)

[B. 실습](#)

- [1. 이미지 다운 및 컨테이너 실행](#)
- [2. docker exec](#)
- [3. cd /home/boaz/spark-streaming](#)
- [4. vim read_twitter.py](#)
- [4. python3 read_twitter.py](#)
- [5. python3 spark_twitter.py](#)
- [6. 결과 확인](#)

[참고](#)

실습 결과 미리 보기

```
Batch: 23
```

word	count
#limyoungwoong	6
#CharliePuth	6
#임영웅_음원강자	6
#LeftandRight	6
#임영웅팬덤	6
#임영웅	6
#임영웅_kpop	6
#Jungkook	4
#ITZY	3
#I	3
#StrayKids	3
#TWICE	3
#트와이스	2
#하이라이트	2
#JIMIN	2
#세븐틴	2
#나연	2
#MOMO	2
#SEVENTEEN	2
#BTS	2

only showing top 20 rows

터미널2 실행 결과 (2)

A. 준비단계

docker hub에서 실습 도커 이미지를 다운로드 한다.

A-1. 도커 이미지 설명

| Spark와 Hadoop 설치 및 환경변수 설정 완료

```
root@4bbd477ae7e8:/opt# ls
hadoop spark
```

/opt 아래에 설치된 hadoop과 spark

| Spark streaming을 위한 파이썬 파일들

: read_twitter.py 와 spark_twitter.py

```
root@4bbd477ae7e8:/home/boaz/spark-streaming# ls
read_twitter.py spark_twitter.py
```

/home/boaz/spark-streaming 디렉토리에 위치한 파일 2개

A-2. 실습 파일 설명

구성

1. read_twitter.py

: twitter api에 연결하여 특정 카테고리 정보가 달린 트윗들을 socket으로 보냄 (like 데이터 보내는 서버)

 [read_twitter.py 코드 리뷰](#)

2. spark_twitter.py

: read_twitter와 spark streaming의 socket과 연결됨. socket으로 받은 트윗 데이터들 중 sql 문을 이용해 해시태그만 골라와 테이블에 저장함 (2초 간격)
(like 데이터 받아 분석하는 클라이언트)

 [spark_twitter.py 코드 리뷰](#)

B. 실습

1. 이미지 다운 및 컨테이너 실행

[Spark Streaming 사전준비](#)

2. docker exec

뒤에서 동시에 파이썬 스크립트를 2개를 실행하기 위해서 터미널 및 powershell 2개를 켜서 모두 접속한다.

```
docker exec -it spark-streaming bash
```

! 만약 컨테이너가 종료된 상태라면 아래 명령어로

```
docker start spark-streaming
```

3. cd /home/boaz/spark-streaming

접속한 터미널 둘 다 이동한다.

```
cd /home/boaz/spark-streaming
```

4. vim read_twitter.py

twitter api 사용을 위해 터미널에서 key를 등록해준다.

(실습 종료 후에 key 값을 바꿀 예정이긴 하지만 그래도 key유출에 주의 부탁드립니다!)

```
export BEARER_TOKEN=AAAAAAAAAAAAAAAAAACzRhgEAAAAAZ429VkwYbA8ZDvYo5egbg10eV6g%3DNcz9WTdRD9aNQ9Yo8K0xSDwg11rv
```

4. python3 read_twitter.py

! 터미널1에서 read_twitter.py 부터 실행 후 터미널 2에서 spark_twitter를 실행한다 !

터미널1 에서 read_twitter를 실행한다.

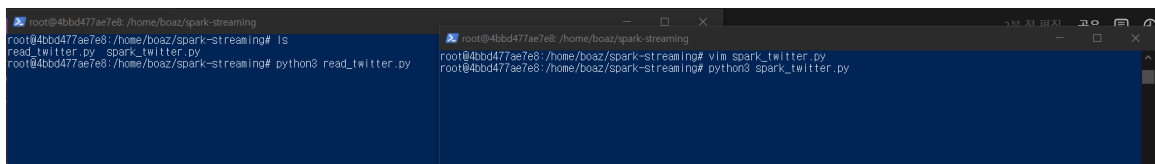
```
python3 read_twitter.py
```

5. python3 spark_twitter.py

터미널 2 에서 spark_twitter를 실행한다.

! 터미널1에서 read_twitter.py 부터 실행 후 터미널 2에서 spark_twitter를 실행한다 !

```
python3 spark_twitter.py
```



6. 결과 확인

- **터미널 1** 에서 정한 규칙으로 twitter 에서 데이터를 불러와 client_socket에 send하는 것을 확인할 수 있음
- **터미널 2** 에서 실시간으로 spark streaming으로 받아온 데이터를 2초 단위로 집계해주는 것을 확인할 수 있다.

[**터미널 1** : python3 spark_twitter.py]

```

root@4bbd477ae7e8:/home/boaz/spark-streaming# ^C
root@4bbd477ae7e8:/home/boaz/spark-streaming# python3 read_twitter.py
Listening on port: 5050

Received request from: ('127.0.0.1', 36126)
|----- get_rules -----|
{"data": [{"id": "1577124436029104129", "value": "kpop has:images", "tag": "kpop"}], "meta": {"sent": "2022-10-04T03:25:21.544Z", "result_count": 1}}
|----- delete_all_rules -----|
{"meta": {"sent": "2022-10-04T03:25:22.333Z", "summary": {"deleted": 1, "not_deleted": 0}}}
|----- set_rules -----|
{"data": [{"value": "kpop has:images", "tag": "kpop", "id": "1577137779867537408"}], "meta": {"sent": "2022-10-04T03:25:23.530Z", "summary": {"created": 1, "not_created": 0, "valid": 1, "invalid": 0}}}
200
|----- get_stream -----|

Original : {
  "data": {
    "edit_history_tweet_ids": [
      "1577137837744394240"
    ],
    "id": "1577137837744394240",
    "text": "[ https://t.co/j1Y8Dtb1cy #EJ #EJ #TEAM @KPOP_JUICE_JP#u3088#u308a https://t.co/Ng5JrwEbgu"
  },
  "matching_rules": [
    {
      "id": "1577137779867537408",
      "tag": "kpop"
    }
  ]
}

Send : b'[ https://t.co/j1Y8Dtb1cy #EJ #EJ #TEAM @KPOP_JUICE_JP#xe3#x82#x88#xe3#x82#x8a https://t.co/Ng5JrwEbgu'

Original : {
  "data": {
    "edit_history_tweet_ids": [
      "1577137865058045954"
    ],
    "id": "1577137865058045954",
    "text": "@kpoppingcom @hiiamcow @kpopmart Thank you for this giveaway!!#ud83e#udef6 https://t.co/CBnHCFQJNg"
  },
  "matching_rules": [

```

터미널1 실행 결과

ctrl+c로 강제 중지하지 않는 이상 계속해서 twitter 데이터를 불러와 client_socket에 보내는 데이터가 콘솔에 보여진다.

[**터미널 2** : python3 spark_twitter.py]

```

root@4bbd477ae7e8:/home/boaz/spark-streaming# python3 spark_twitter.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/10/04 03:25:09 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/10/04 03:25:15 WARN TextSocketSourceProvider: The socket source should not be used for production applications! It does not support recovery.
22/10/04 03:25:19 WARN ResolveWriteToStream: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-e69c572c-fe91-41d9-8f07-1d75d4e5ed2b. If it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
22/10/04 03:25:19 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not supported in streaming DataFrames/Datasets and will be disabled.

-----
Batch: 0
-----
+----+-----+
|word|count|
+----+-----+
+----+-----+

22/10/04 03:25:41 WARN ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 2000 milliseconds, but spent 21565 milliseconds

-----
Batch: 1
-----
+----+-----+
| word|count|
+----+-----+
| #EJ|    2|
| #TEAM|  1|
+----+-----+

22/10/04 03:26:04 WARN ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 2000 milliseconds, but spent 10385 milliseconds

-----
Batch: 2
-----
+----+-----+
| word|count|
+----+-----+
| #EJ|    2|
| #TEAM|  1|
+----+-----+

22/10/04 03:26:16 WARN ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 2000 milliseconds, but spent 11683 milliseconds

-----
Batch: 3
-----
+----+-----+
| word|count|
+----+-----+
| #CharliePuth|    2|
+----+-----+

```

터미널2 실행 결과 (1)

Batch: 23

word	count
#limyoungwoong	6
#CharliePuth	6
#임영웅_음원강자	6
#LeftandRight	6
#임영웅팬덤	6
#임영웅	6
#임영웅_kpop	6
#Jungkook	4
#ITZY	3
#1	3
#StrayKids	3
#TWICE	3
#트와이스	2
#TWIC...	2
#하이라이트	2
#JIMIN	2
#세븐틴	2
#나연	2
#MOMO	2
#SEVENTEEN	2
#BTS	2

only showing top 20 rows

터미널2 실행 결과 (2)

소켓 통신이 계속 되는 동안 스트리밍으로 받아온 데이터를 2초 간격으로 집계하여 해시태그별로 count를 하는 것을 콘솔로 확인할 수 있다.



실습을 마친 후에는 꼭 소켓 통신을 종료해주세요!!

Twitter API사용 시, tweet을 pull 해올 수 있는 한계가 있어서 과금 방지를 위해 꼭 종료 부탁드립니다.



참고

(Twitter API 연결 파트) Twitter API v2 공식 github

Twitter-API-v2-sample-code/filtered_stream.py at main · twitterdev/Twitter-API-v2-sample-code

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

https://github.com/twitterdev/Twitter-API-v2-sample-code/blob/main/Filtered-Stream/filtered_stream.py

twitterdev/Twitter-API-v2-sample-code


Sample code for the Twitter API v2 endpoint

20 Contributors 22 Issues 2k Stars 652 Forks

(Spark 스트리밍 파트) structured-spark streaming

Easy to Play with Twitter Data Using Spark Structured Streaming

Data is all around and twitter is one of the golden source of data for any kind of sentiment analysis. There are lot of ways we can read twitter live data and process them. In this article I will demonstrate how easily we can create a connection with

 <https://ch-nabarun.medium.com/easy-to-play-with-twitter-data-using-spark-structured-streaming-76fe86f1f81c>




(소켓 연결 파트) dstream-spark-streaming

python-spark-streaming/TweetRead.ipynb at master · jleetutorial/python-spark-streaming

Contribute to jleetutorial/python-spark-streaming development by creating an account on GitHub.

jleetutorial/**python-spark-streaming**

 https://github.com/jleetutorial/python-spark-streaming/blob/master/1_start/TweetRead.ipynb

 1 Contributor  2 Issues  127 Stars  158 Forks