

گام اول: دریافت اطلاعات و Preprocess

برای داده‌های ورودی مشتاق بودم تا بتوانم داده‌های توییت‌ر را دریافت کرده و آن‌ها را پردازش کنم؛ پس از ایجاد کد جهت دریافت داده‌های توییت‌ر و استفاده از API‌های داخل دستورکار، ارور گرفتم و متوجه شدم که نیاز به دریافت API از توییت‌ر است. سپس درخواست API کردم. اما توییت‌ر موافقت نکرد و گفت فعلاً امکان‌پذیر نیست. سپس به سراغ ساخت بات برای فروش رفتم. پس از ایجاد حساب و ساخت بات برای فروش، توی گوگل سرچ کردم ولی موفق نشدم طریقه عضو کردن بات در کانال‌ها را پیدا کنم و انگلیسی هم نمی‌توانستم سرچ کنم.

سپس به سراغ تلگرام رفتم و چند روزی روی این موضوع سرچ می‌کردم. چندجایی در اینترنت خواندم که امکان عضو کردن بات در کانال‌هایی که خوم ادمین آن‌ها هستم وجود دارد و من نمی‌توانم بات را در کانال دیگران عضو کنم. سپس متوجه شدم که می‌توانم پیام‌های حساب کاربری شخصی خودم را به کمک دریافت api و کتابخانه telethon وارد پایتون کنم. به این شکل امکان دریافت پیام کانال‌های تلگرام وجود داشت. اما توی قسمتی از documentation نوشته بود که اگر سرعت دریافت پیام، علی‌الخصوص برای کشورهای حساس مثل ایران و روسیه، بالا باشد، امکان بسته شدن اکانت شما وجود دارد ([لینک](#) مطلب ذکر شده).

به همین دلیل ریسک نکردم و به سراغ توییت‌های سهام‌یاب رفتم

ابتدا کافکا را نصب کرده و به کمک command های زیر server و zookeeper را اجرا می‌کنم.

1st cmd in kafka dir

`.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties`

2nd cmd in kafka dir

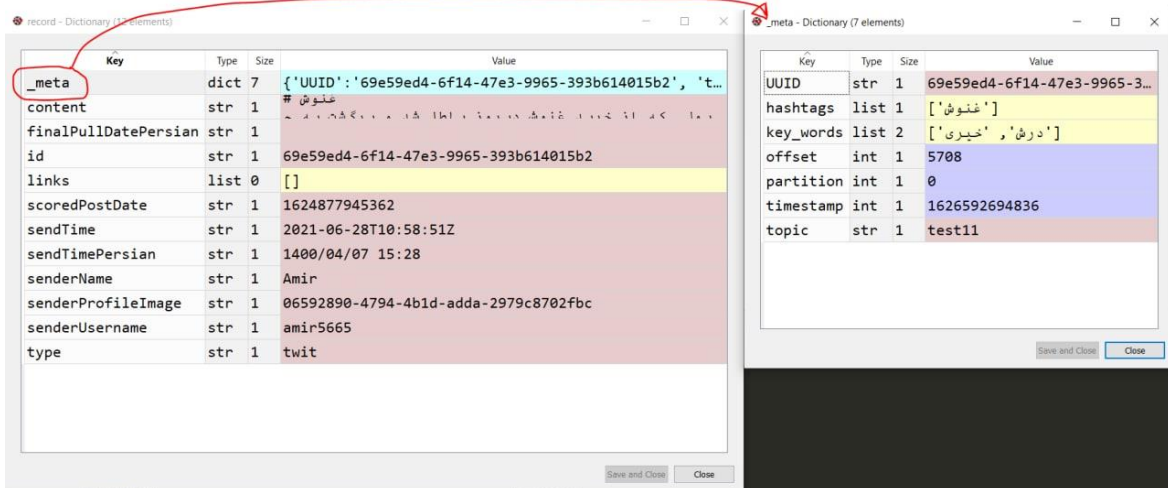
`.\bin\windows\kafka-server-start.bat .\config\server.properties`

حال با فراخوانی مداوم API داده‌ها را از سهام‌یاب دانلود می‌کنیم و سپس آن‌هایی که id تکراری ندارند را به کمک producer وارد topic کافکا به نام raw_data می‌کنیم. در همین حین، با یک consumer داده‌های خام را از تاپیک raw_data دریافت کرده و روی هر توییت پیش پردازش‌های گفته شده را اعمال می‌کنیم.

با دریافت هر توییت، ابتدا یک timestamps و UUID برای آن‌ها تولید کردم. به کمک یک تابع هشتگ‌های موجود در content هر توییت را یافته و آن را به اطلاعات همان توییت اضافه می‌کنیم. همین‌طور لینک‌ها هم استخراج شد.

جهت محاسبه کلمات کلیدی، از معیار tf/idf استفاده شده است. ابتدا لیستی از ایست‌واژه‌هایافتم و با حذف کردن آن‌ها از متن توییت، tf/idf را محاسبه کردم. در tf/idf فراوانی هر کلمه را در هر توییت می‌یابیم و همین‌طور فراوانی همان کلمه را در همه‌ی توییت‌ها بررسی می‌کنیم. کلمه‌ای از tf/idf بالا برخوردار است (کلمه کلیدی است) که در یک توییت وجود داشته باشد اما در بقیه توییت‌ها کمتر تکرار شده باشد. سپس برای هر توییت علاوه بر کلمات خاصی که صورت سوال ذکر کرده است، دو کلمه‌ای که tf/idf بزرگتری دارند را به عنوان کلمات کلیدی در نظر گرفتیم.

هر توییت ما یک دیکشنری است. در عکس زیر key, value یک توییت را نمایش داده‌ام و در سمت راست تصویر اطلاعاتی که به عنوان متادیتا ذخیره خواهد شد، قرار دارد



در عکس بالا می‌بینیم که `partition` برابر صفر و `offset` برابر ۵۷۰۸ است. یعنی ایت تویت در تاپیک `test11` و در پارتیشن اول آن و همین‌طور در خانه‌ی ۵۷۰۸ پارتیشن اول قرار دارد.

در نهایت به کمک یک `producer` جدید داده‌های پیش‌پردازش شده را در تاپیک `proccesed_data` قرار می‌دهیم.

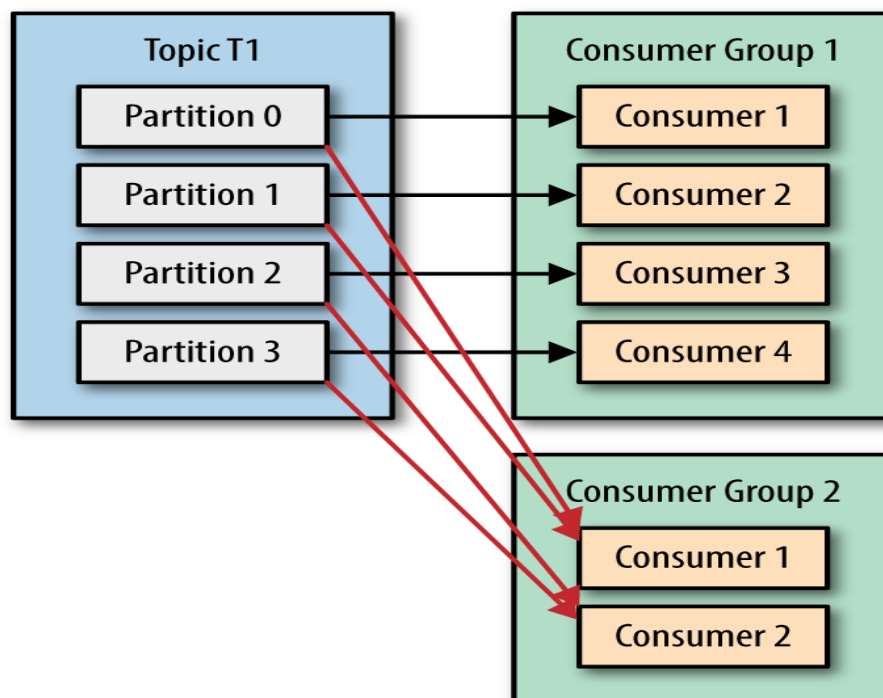
معماری کافکا:

کافکا شامل جمعی از `record`، `topic`، `consumer`، `producer`، `broker`، `log` و `cluster` است و هر `record` دارای `key` (اختیاری)، `value` و `timestamp` است.

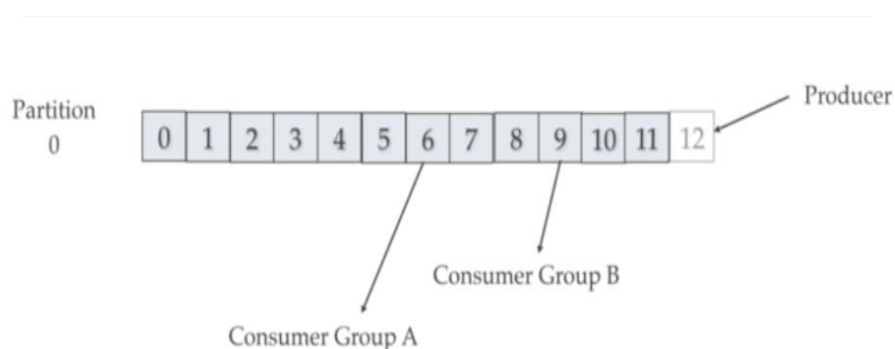
`record` هایی که وارد کافکا می‌شوند `immutable` هستند؛ یعنی `producer` به تنهایی فقط می‌تواند پیام‌ها را به تاپیک `topic` اضافه کند و امکان تغییر در `record` های موجود را ندارد.

هر تاپیک یک `log` دارد که محل ذخیره‌ی `topic` روی دیسک است. هر `topic` متشکل از تعدادی `partition` است. هر پارتیشن متشکل از تعدادی `offset` است (عکس زیر درک بهتری ایجاد می‌کند). هر `record` وارد `topic` مورد نظر و سپس وارد `partition` مورد نظر می‌شود و در آنجا به ترتیب درون `offset` ها ذخیره می‌شود. برای آنکه چندین `consumer` بتوانند همزمان `record` دریافت کنند، نیاز به وجود `partition` داریم. در عکس زیر

نحوه‌ی اتصال consumer ها به partition را می‌بینیم. با ایجاد consumer با group id متفاوت، می‌توان همزمان از یک تاپیک پیام دریافت کرد.



هر پارتیشن بدین شکل است:



Consumer groups remember offset where they left off.
Consumers groups each have their own offset.

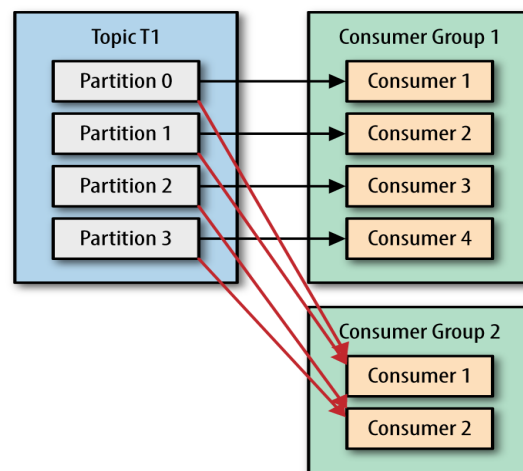
Producer writing to offset 12 of Partition 0 while...
Consumer Group A is reading from offset 6.
Consumer Group B is reading from offset 9.

Producer برای تولید جریان داده و ارسال آن به topic استفاده می‌سود. Consumer پیام‌ها را از partition کافکا می‌خواند (هر topic شامل تعدادی partition است).

Cluster کافکا متشکل از تعدادی broker است و هر broker شامل تعدادی topic است.

گام دوم : persistence

در این قسمت داده‌های پیش پردازش شده‌ی مرحله قبل را به کمک **consumer** با **group id** معین دریافت می‌کنیم (چون در گام‌های بعدی از تاپیک **processed_data** داده دریافت خواهیم کرد و در واقع داریم به صورت همزمان از **partition** های واحد توسط چند **consumer** پیام دریافت می‌کنیم ، تعیین **group id** حیاتی است. این موضوع به عکس زیر اشاره دارد:



پس از دریافت داده از از کافکا، داده‌ها را وارد **index** الاستیک سرچ می‌کنیم. سپس به کمک کیبانا، داشبوردهای خواسته شده را نمایش می‌دهیم.

ابر کلمات یک کانال یا خبرگزاری خاص در یک بازه زمانی

از آنجایی که ما از داده‌های سهام‌یاب استفاده می‌کنیم،

من ابر کلمات هشتگ‌ها در یک بازه‌ی خاص را رسم کرده‌ام



متن ده پست اخیری که دریافت شده است

The screenshot shows the Elasticsearch Kibana interface. The main panel displays a list of text snippets with timestamps. The right sidebar shows the 'test_12*' visualization configuration with 'Metrics' and 'Buckets' sections.

test_12*

Metrics

- Metric Last 10 content
- + Add

Buckets

- Split group sendTime: Descending
- Discard
- Update

پارامترهای تعیین شده جهت نمایش داشبورد در عکس‌های زیر آورده شده است (یعنی metrics و buckets).

Metrics

Aggregation: Top Hit

Field: content

Aggregate with: Concatenate

Size: 10

Sort on: sendTime

Order: Descending

Custom label:

> Advanced

Buckets

Split group

Aggregation: Terms

Field: sendTime

Order by: Custom metric

Aggregation: Unique Count

Field: meta.timestamp

> Advanced

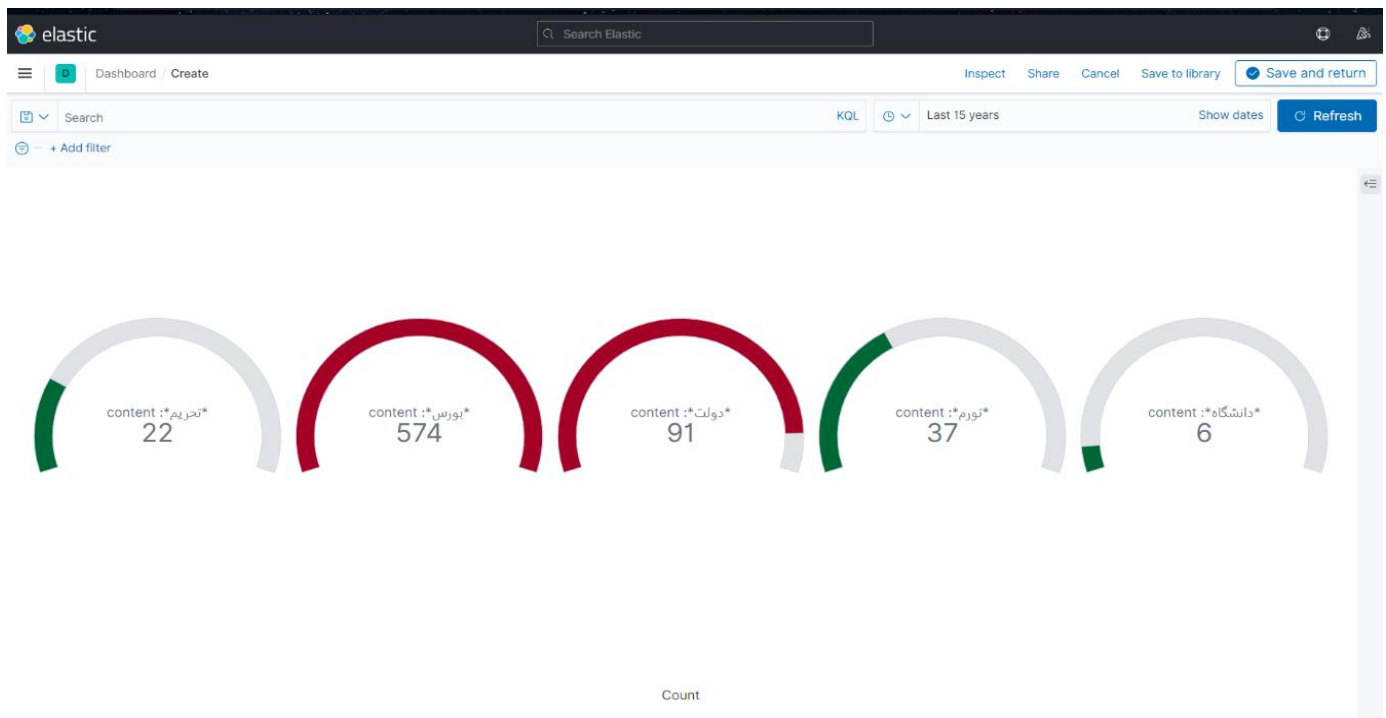
Order: Descending

Size: 10

Discard

Update

تعداد پست‌های ارسال شده به ازای چندتا از کلمات کلیدی خاص
که در مرحله قبل مشخص شده است در یک بازه زمانی خاص



از bucket زیر استفاده شده است

Data Options

Buckets

Split group

Aggregation

Filters

Filter 1

content:*تحریم*

Filter 2

content:*بورس*

Filter 3

content:*دولت*

Filter 4

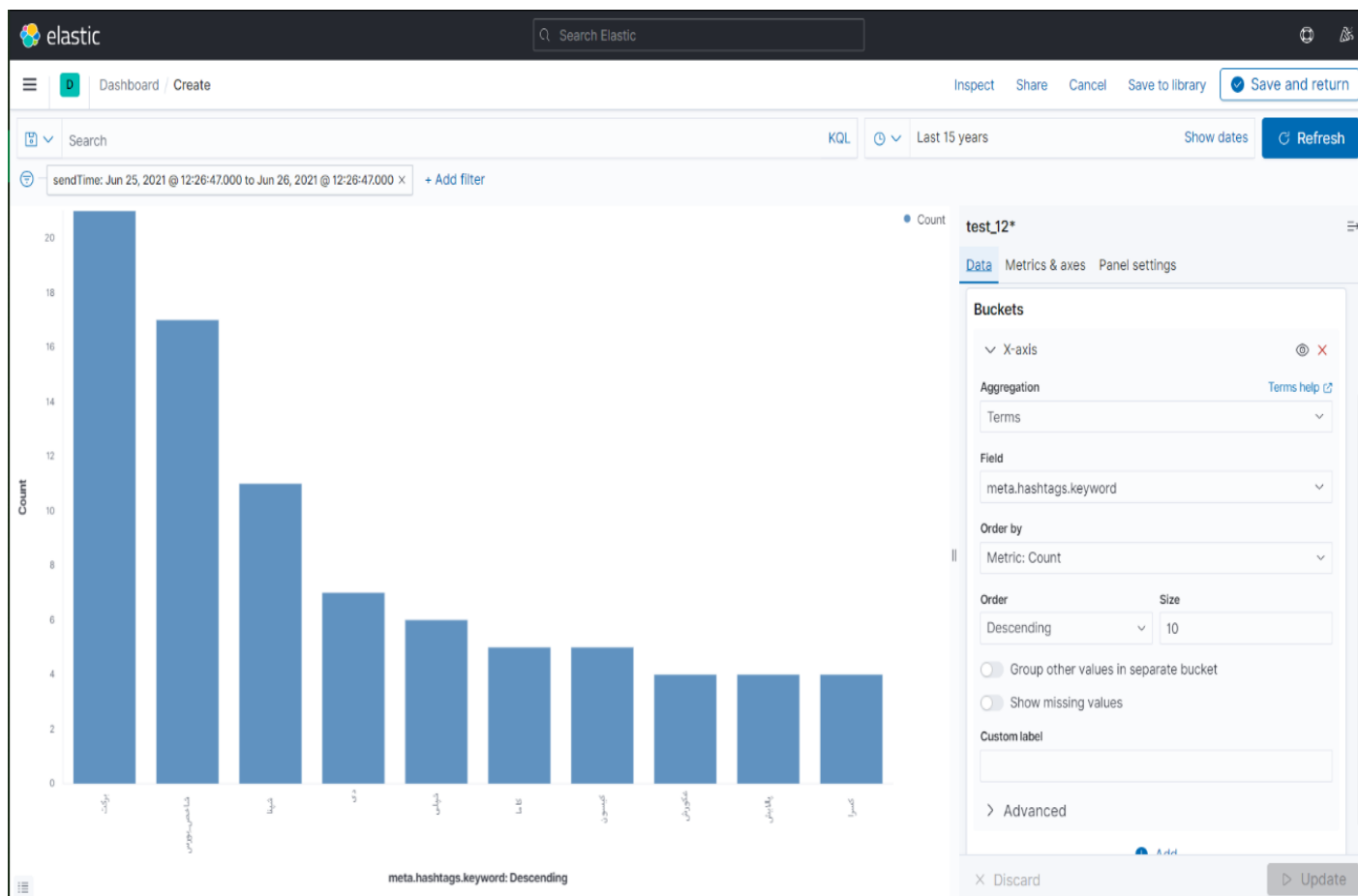
content:*تورم*

Filter 5

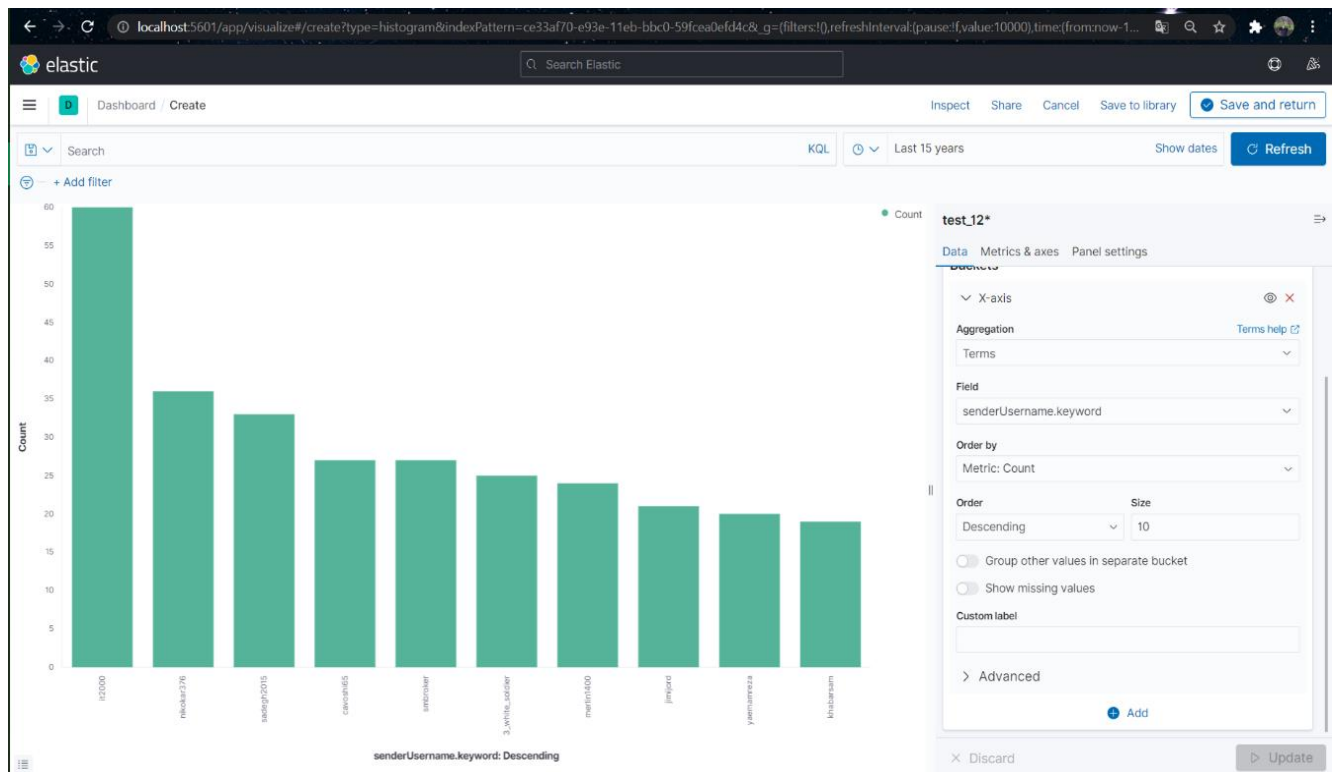
content:*دانشگاه*

Discard Update

ده هشتگ بیشتر استفاده شده در پستهای تمام کانالها در یک بازه زمانی با تعداد تکرار هر هشتگ (یک نمودار ستونی) مثلا هشتگهای بیشتر استفاده شده در یک روز اخیر



یک نمودار به انتخاب خودتان
در این قسمت تعداد توییت‌های یک کاربر را در یک نمودار
ستونی نمایش داده‌ام



اگر به دنبال تمام پست‌های حاوی یک کلمه خاص از یک خبرگزاری یا کانال خاص در یک بازه مشخص هستیم، چه دستوری باید بنویسیم (و یا یک هشتگ خاص یا یک کاربر خاص در توئیته‌ها)

The screenshot shows a REST client interface with the following details:

- Request:**

```
GET /test12/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "meta.hashtags": "توییت"
          }
        },
        {
          "range": {
            "sendTime": {
              "gte": "2020-01-01T00:00:00",
              "lt": "now"
            }
          }
        }
      ]
    }
  }
}
```
- Response:**

```
{
  "took": 11,
  "timed_out": false,
  "shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 515,
      "relation": "eq"
    },
    "max_score": 3.1629546,
    "hits": [
      {
        "_index": "test12",
        "_type": "tweeter",
        "_id": "Uq1Jw3oBHFU3q77c_EC",
        "_score": 3.1629546,
        "_source": {
          "id": "1feb3bd5-8d75-46ab-a14c-dd16bfda3660",
          "sendTime": "2021-06-26T07:49:16Z",
          "sendTimePersian": "1400/04/05 12:19",
          "senderName": "Anonymous ",
          "senderUsername": "amirhoseinkh7",
          "senderProfileImage": "default",
          "content": "توییت‌ها"
        }
      }
    ]
  }
}
```

تعداد توئیتهای ارسالی به ازای یک کلمه خاص را به ازای یک هشتگ خاص در توئیته‌ها در یک بازه زمانی

The screenshot shows a REST client interface with the following details:

- Request:**

```
GET /test12/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "meta.hashtags": "شخص بورس"
          }
        }
      ]
    }
  }
}
```
- Response:**

```
{
  "took": 4,
  "timed_out": false,
  "shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 8,
      "relation": "eq"
    },
    "max_score": 7.2275667,
    "hits": [
      {
        "_index": "test12",
        "_type": "tweeter",
        "_id": "WalJw3oBHFU3q77d_A",
        "_score": 7.2275667,
        "_source": {
          "id": "47a92453-e871-4485-9e37-b1ff2820c5c6",
          "sendTime": "2021-06-26T12:35:59Z",
          "sendTimePersian": "1400/04/05 17:05",
          "parentSendTime": "2021-04-19T18:51:42Z",
          "parentSendTimePersian": "1400/01/30 23:21",
          "parentId": "271539403",
          "parentSenderName": "سعید آقایی",
          "parentSenderUsername": "sfaghahi",
          "parentSenderProfileImage": "1ec38f47-70ac-4d3a-ae77"
        }
      }
    ]
  }
}
```

