به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس کلان داده

پروژه پایانی

نام و نام خانوادگی :

فاطمه سلیقه – زهرا نصرالهی

شماره دانشجویی :

۸۱۰۱۹۸۳۰۶-۸۱۰۱۹۸۳۲۶

مرداد ماه ۱۳۹۹

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

بخش اول – دریافت اطلاعات و preprocess

برای دریافت اطلاعات از پیام رسان سروش در ابتدا سعی نمودیم تا یک بات برای دریافت اطلاعات ایجاد کنیم . اما به دلیل مشکلات مربوط به ایجاد بات ها از جمله مشکلات مربوط به عضو کردن بات ها در کانال ها و ... تصمیم برآن شد تا با استفاده از اتصال به API پیام رسان سروش و با استفاده از ID کانال ها اطلاعات را دریافت نماییم .

دریافت اطلاعات را از طریق یک حلقه همیشه true انجام داده و در مرحله بعد وارد کانال preprocess می شویم . در این قسمت هشتگ ها را در متن پیام پیدا نموده و همچنین خود پیام های دریافتی که در قالب json هستند دارای کلمات کلیدی می باشند و فیلد keyword این کلمات را مشخص می نماید . سپس به سراغ جستجوی کلمات معرفی شده در صورت مسئله می پردازیم و همه ی این موارد را به عنوان فیلد metadata به پیام اضافه می کنیم . در متن پیام url ها را جستجو نموده و در فیلدی به نام links می گذاریم سپس در فیلد file.extention را می بینیم ، اگر شامل ادرس فایل jpg بود آن را به فیلد images اضافه می کنیم .

حال نوبت به اتصال به kafka است . برای راه اندازی kafka لازم است تا ابتدا apache-zookeeper را اجرا نموده . بنابراین zookeeper را از این آدرس [۱] دانلود نموده سپس با استفاده از کامند زیر می توان آن را اجرا نمود :

```
\apache-zookeeper-3.6.1-bin\bin> .\zkServer.cmd
```

حال برای اجرای kafka از این آدرس [۲] استفاده نموده و آن را دانلود می کنیم سپس با استفاده از کامند زیر می توان آن را اجرا نمود .

```
\kafka_2.12-2.5.0> .\bin\windows\kafka-server-start.bat .\config\server.properties
```

در این قسمت لازم است تا یک topic به نام final روش زیر تعریف کنیم .

```
bin/windows/kafka-topics.bat      --create      --bootstrap-server
   localhost:9092 --replication-factor 1 --partitions 1 --topic final
```

[۱] https://zookeeper.apache.org/

[۲] https://kafka.apache.org/quickstart

پس از آن که هر دو سرور اجرا شدند می توان از پایتون برای اتصال به کافکا استفاده نمود . بدین منظور می توان از یکی از کتابخانه های pykafka یا kafka-python استفاده نمود .

برای نصب pykafka از کامند زیر استفاده می نماییم :

```
pip install pykafka
```

سپس برای اتصال به کافکا لازم است تا به پورت 9092 از localhost متصل شویم :

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
```

برای نوشتن اطلاعات در کافکا لازم است تا یک producer تعریف کنیم تا پیام های دریافتی را به کافکا بدهد :

```
with topic.get_sync_producer() as producer:
    producer.produce(bytes(str(out['messages'][0]), encoding='utf-8'))
```

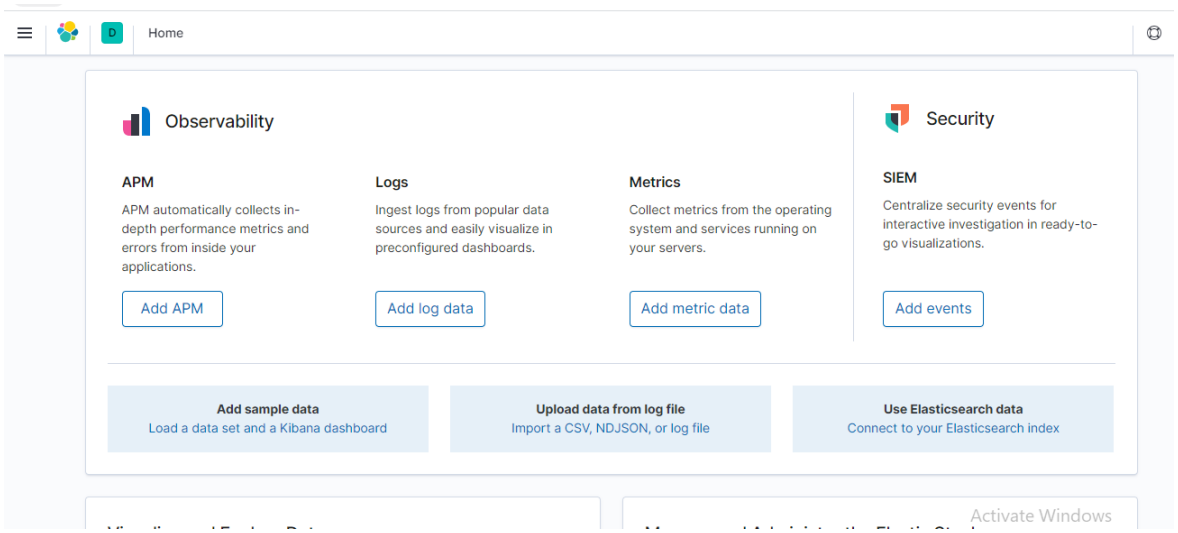
از این پس می توان consumer هایی از جمله elasticsearch و redis و Cassandra و ... را ایجاد نمود تا به صورت real time داده ها را دریافت کنند و برحسب نیاز پردازش نمایند.

گام دوم – persistence

در این قسمت لازم است تا elasticsearch و kibana را اجرا نماییم . نکته ای که وجود دارد این است که برای کار با الاستیک ۸ لازم است تا نسخه ۱۱ به بعد جاوا را داشته باشیم . حال برای اجرای الاستیک سرچ می توان آن را از این آدرس[۱] دانلود نموده و سپس فایل elasticsearch.bat را اجرا نماییم در صورتی که الاستیک سرچ بدون خطا اجرا شود در localhost:9200 قابل مشاهده است :


```
{
  "name" : "DESKTOP-IED3F77",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "k-TFHmpBS3CRZduGFKYA7A",
  "version" : {
    "number" : "7.8.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "757314695644ea9a1dc2fecd26d1a43856725e65",
    "build_date" : "2020-06-14T19:35:50.234439Z",
    "build_snapshot" : false,
    "lucene_version" : "8.5.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


سپس kibana را از آدرس[۲] دانلود نموده و سپس فایل kibana.bat را اجرا می کنیم که در این صورت در آدرس localhost:5601 قابل مشاهده است :



[۱] https://www.elastic.co/downloads/elasticsearch

[۲] https://www.elastic.co/downloads/kibana

در این قسمت می توان از پایتون استفاده نموده و یک consumer برای کافکا تعریف نماییم تا اطلاعات را در الاستیک سرچ به صورت real time ذخیره نماید . برای کار با الاستیک سرچ از طریق پایتون لازم است تا کتابخانه elasticsearch را نصب نماییم بدین منظور از کامند زیر استفاده می نماییم :

pip install elasticsearch

حال یک consumer تعریف نموده تا به صورت real time اطلاعات را از کافکا بخواند در الاستیک سرچ قرار دهد :

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
consumer = topic.get_simple_consumer()
```

چون در هنگام ارسال هر پیام به کافکا به صورت utf-8 کد کرده بودیم حال هر پیامی که دریافت می کنیم را انکود می نماییم :

```
msg = (message.value).decode("utf-8")
```

سپس چون فرمت پیام دریافت شده string است برای تبدیل آن به json از ماژول ast استفاده می نماییم :

```
msg = ast.literal_eval(msg)
```

سپس به صورت زیر می توان پیام دریافتی را در الاستیک سرچ و در index مورد نظر ذخیره نمود .

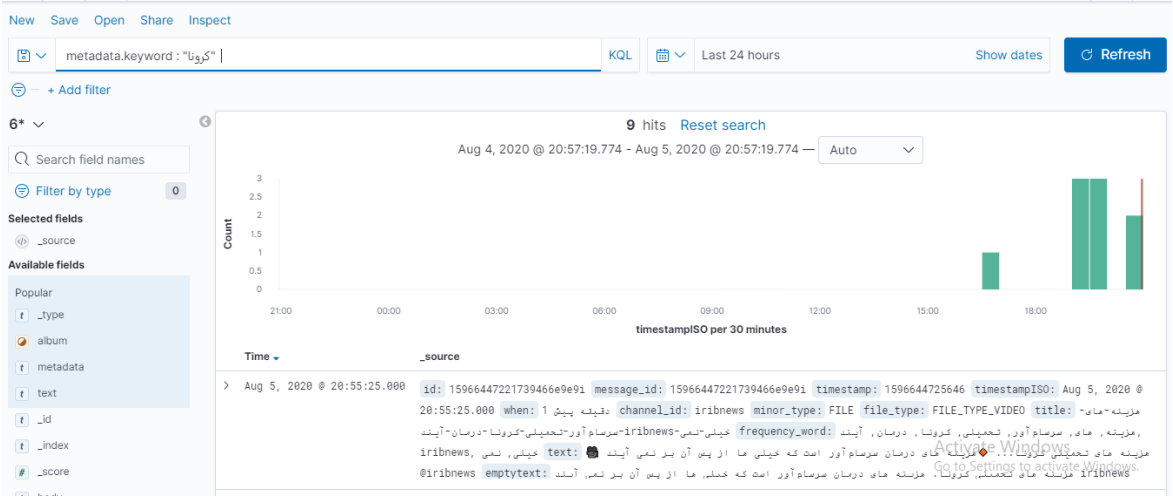
```
es = Elasticsearch()
```

```
es.index(index=index, id=esid, body=msg)
```

حال پس از ذخیره اطلاعات نوبت به استفاده از kibana برای ایجاد داشبورد می رسد .

Kibana از چهار بخش Discover ، Visualize ، Dashboard و Settings تشکیل شده است .

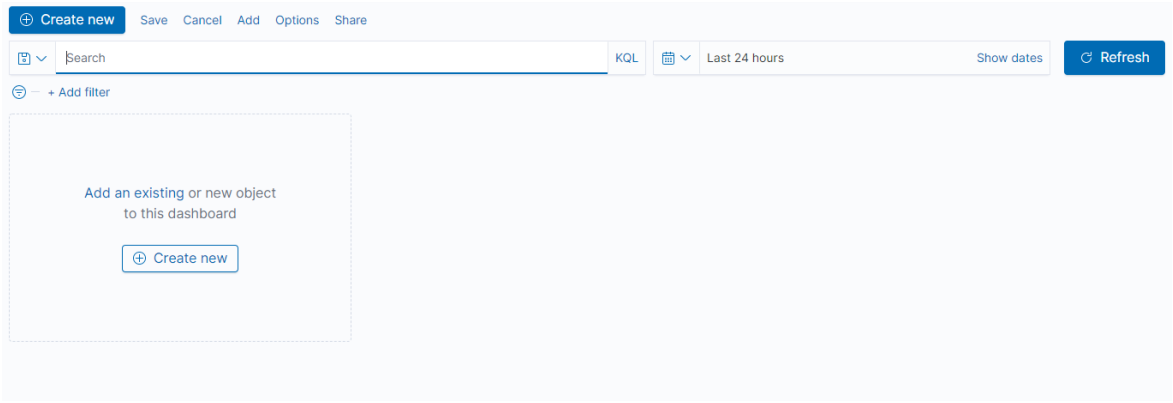
در بخش Discover تمام لاگ های دریافتی توسط الاستیک سرچ نشان داده می شود . ما می توانیم با استفاده از فیلتر و بخش کوئری پیام های مورد نظر را نشان داده و هم چنین می توانیم از طریق فیلتر زمانی آنها را فیلتر کنیم . برای نمونه می توان شکل زیر را مشاهده نمود :



در بخش visualize می توان visualization هایی را ایجاد تغییر و یا مشاهده نمود . چند نوع مختلف visualization از جمله نمودار دایره ای و ستونی نقشه و جدول و ... وجود دارد

در بخش Dashboard می توان چند visualization را به طور هم زمان نمایش داد . که ما در این پروژه از این قسمت استفاده می کنیم و در ادامه با بخش های مختلف آن آشنا می شویم .

برای ایجاد داشبورد در بخش Dashboards روی Create Dashboard کلیک میکنیم . صفحه زیر ایجاد می شود که می توان با استفاده از create new یک visualization جدید ایجاد نمود و یا می توان اگر یک search قبلا در بخش های دیگر انجام گرفته و ذخیره شده است را انتخاب نماییم تا در داشبورد نمایش داده شود .

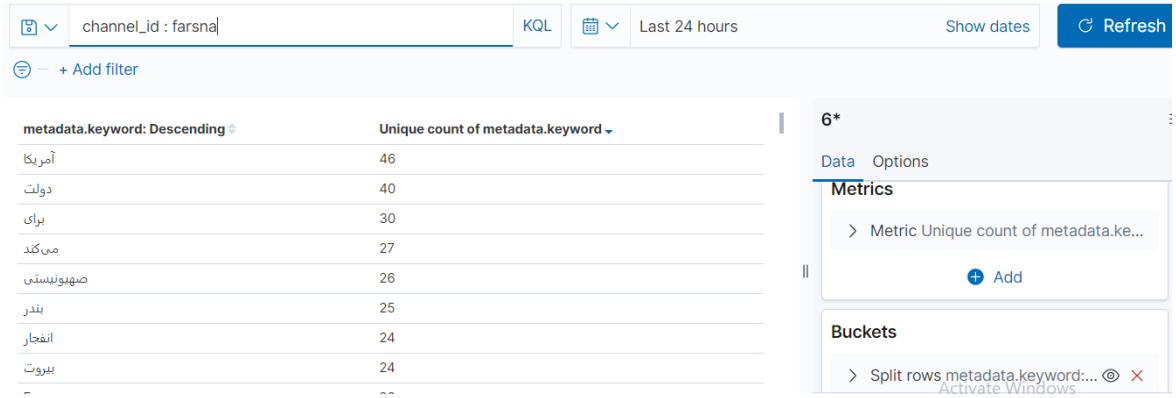


ابتدا می توان با استفاده از کد زیر می توان پیش پردازش های لازم را با استفاده از Persian analyzer انجام داد :

```
PUT /persian_example
{
  "settings": {
    "analysis": {
      "char_filter": {
        "zero_width_spaces": {
          "type":       "mapping",
          "mappings": [ "\\u200C=>\\u0020"]
        }
      },
      "filter": {
        "persian_stop": {
          "type":       "stop",
          "stopwords":  "_persian_"
        }
      },
      "analyzer": {
        "rebuilt_persian": {
          "tokenizer":     "standard",
          "char_filter": [ "zero_width_spaces" ],
          "filter": [
            "lowercase",
            "decimal_digit",
            "arabic_normalization",
            "persian_normalization",
            "persian_stop"
          ]
        }
      }
    }
  }
```

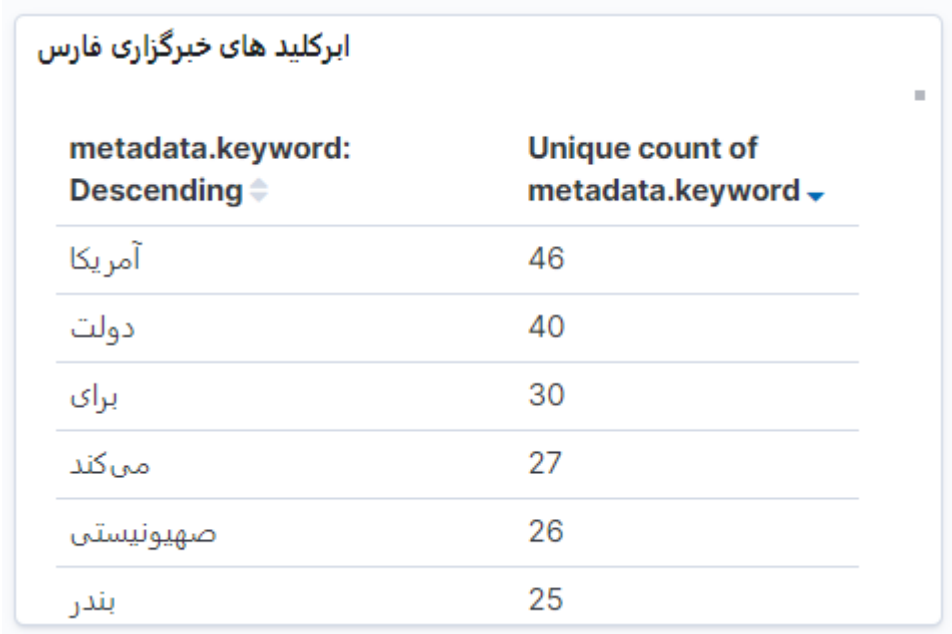
الف) ابر کلمات یک کانال یا خبرگزاری خاص در یک بازه زمانی

برای این بخش در بخش visualize یک table ایجاد می کنیم سپس در بخش سرچ کوئری را به این صورت در نظر می گیریم که پیام هایی که channel__id آنها مثلا farsna است . سپس در سمت راست metric را unique count of metadata.keyword در نظر می گیریم یعنی به صورت uniq تعداد هر ابرکلمه را بشمار سپس در بخش bucket هم می گوییم که فیلد metadata را نمایش بده . سپس در بخش فیلتر زمانی بازه زمانی را مشخص می کنیم مثلا یک روز گذشته .



حال می توان visualization ایجاد شده را ذخیره نمود و در داشبورد آن را استفاده نمود .

نتیجه در داشبورد به صورت زیر خواهد بود :

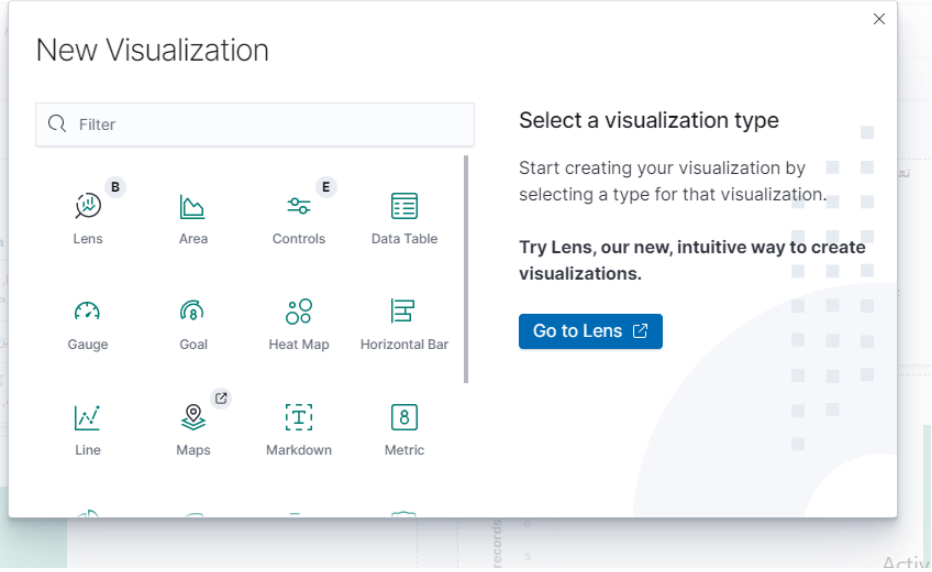


ب) متن ده پست اخیری که دریافت شده است

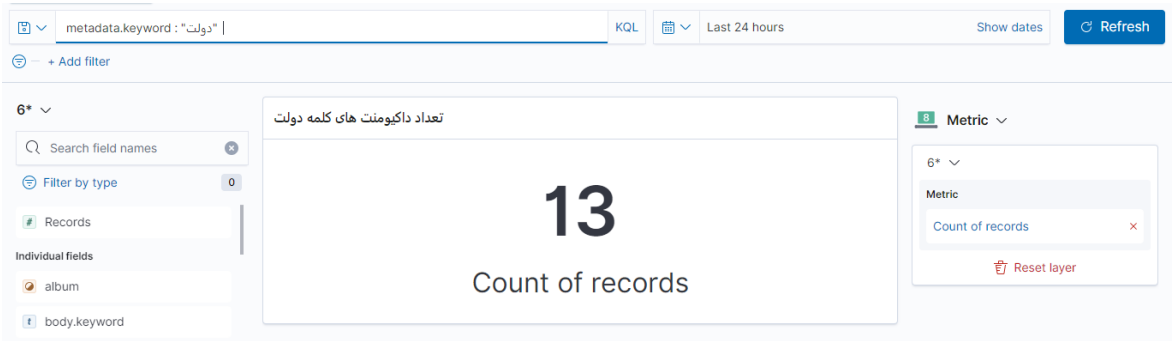
در بخش visualization یک table ایجاد می کنیم . هر پیام دارای یک مقدار timestampISO است که می توان ترتیب دریافت اطلاعات را ازآن طریق متوجه شد . در table ایجاد شده در بخش اول که metric را مشخص می کنیم ، می گوییم که textها را بر اساس timestampISO صورت نزولی مرتب کند سپس ۱۰ ردیف آخر را نشان دهد . در قسمت bucket هم می گوییم که row split انجام داده و timestampISO را به صورت نزولی مرتب و ۱۰ تای اول را نشان دهد . بنابراین می توان ۱۰ پست آخر را ملاحظه نمود .

خروجی در داشبورد به صورت زیر است :

ج) یکی از ویژگی های کیبانا آن است که می توان در بخش create new از داشبورد از ابزار Lens استفاده نمود :



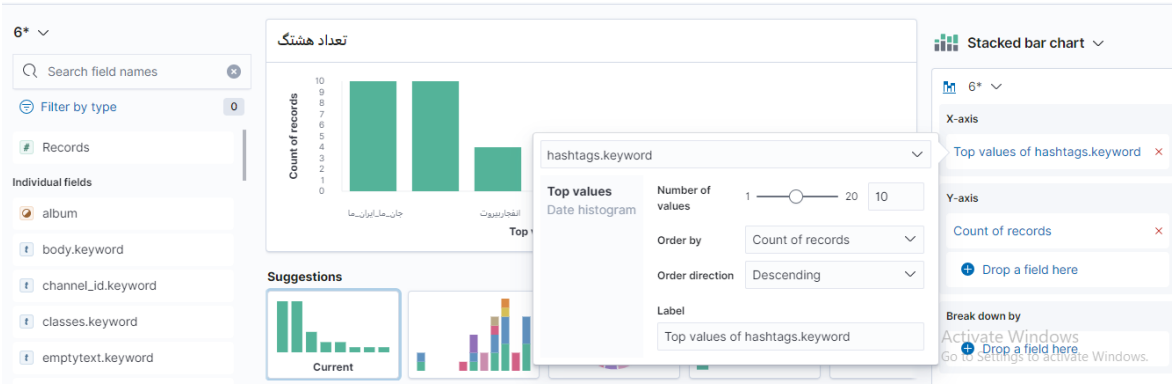
در این بخش از ابزار lens استفاده نموده ایم . لازم است تا metadata.keyword شامل کلمه مورد نظر باشد مثلا دولت یا کرونا یا ... . بازه زمانی با استفاده از فیلترزمان در بالا سمت راست مشخص خواهد شد .



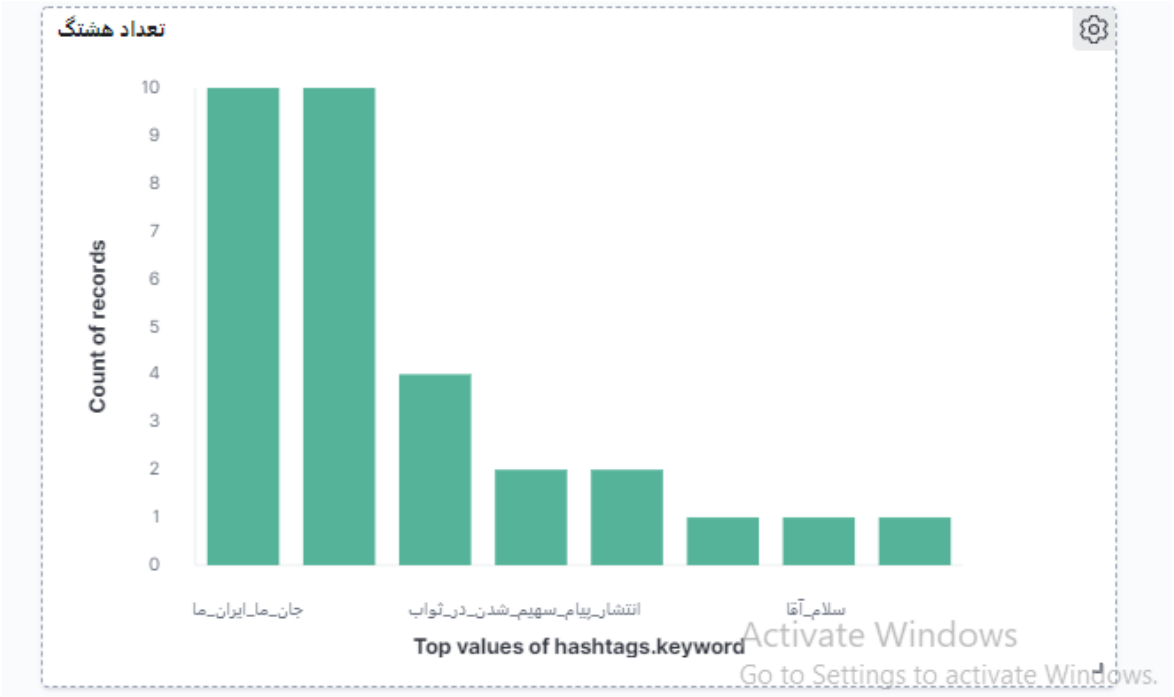
و نتیجه در داشبورد به صورت زیر خواهد بود :

د) 10 هشتگ یا نماد بورسی بیشتر استفاده شده .

این بخش را با استفاده از Lens ایجاد میکنیم . به این صورت که یک نمودار ستونی را انتخاب می نماییم سپس محور x ها را hashtags.keyword در نظر می گیریم و به به صورت نزولی بر اساس تعداد مرتب می کنیم و سپس ۱۰ مقدرا از آن را نمایش می دهیم

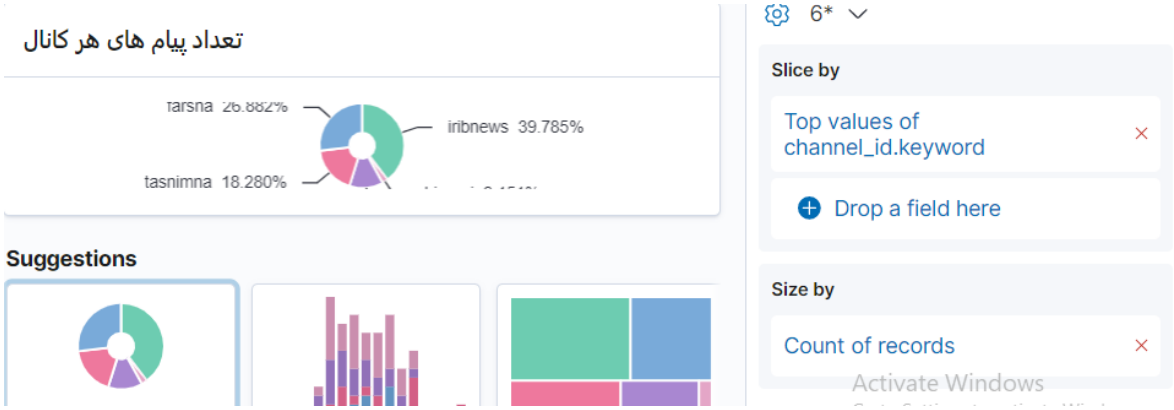


نتیجه در داشبورد به صورت زیر خواهد بود :

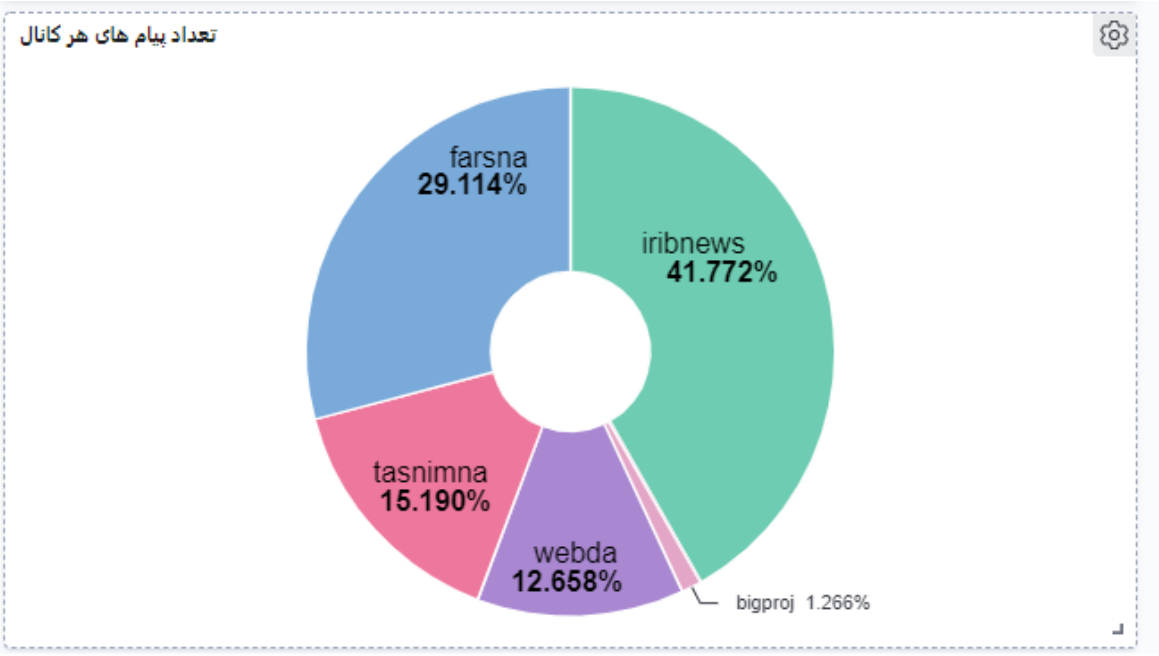


ه) یک نمودار به انتخاب خودتان : درصد پیام های هر کانال

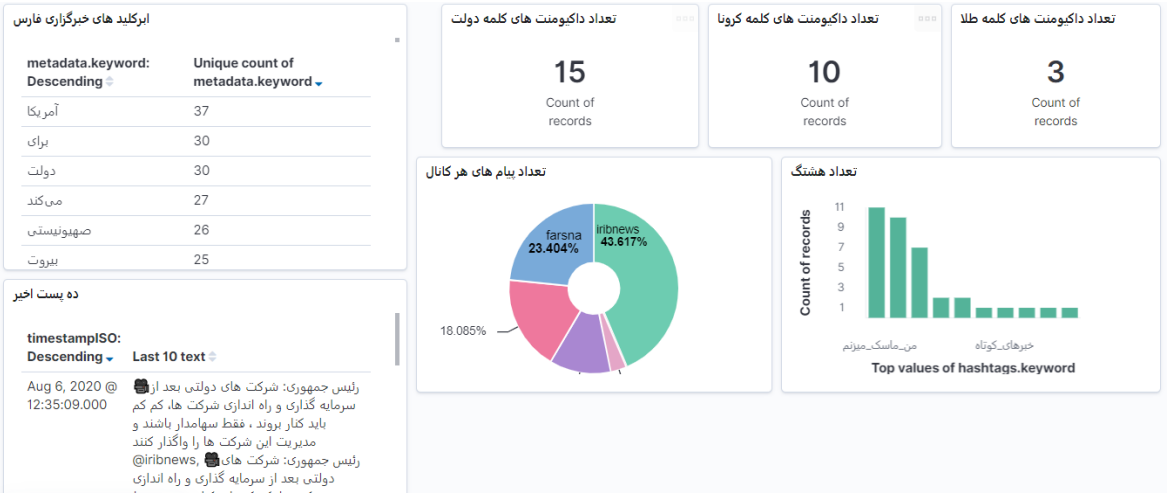
در این قسمت هم از lens استفاده نمودیم. گفته شده خبرگزاری های با بالاترین تعداد را نشان بده . با استفاده از فیلد channel_id.keyword .



خروجی در داشبورد به صورت زیر است .



داشبورد ایجاد شده به صورت زیر خواهد بود :

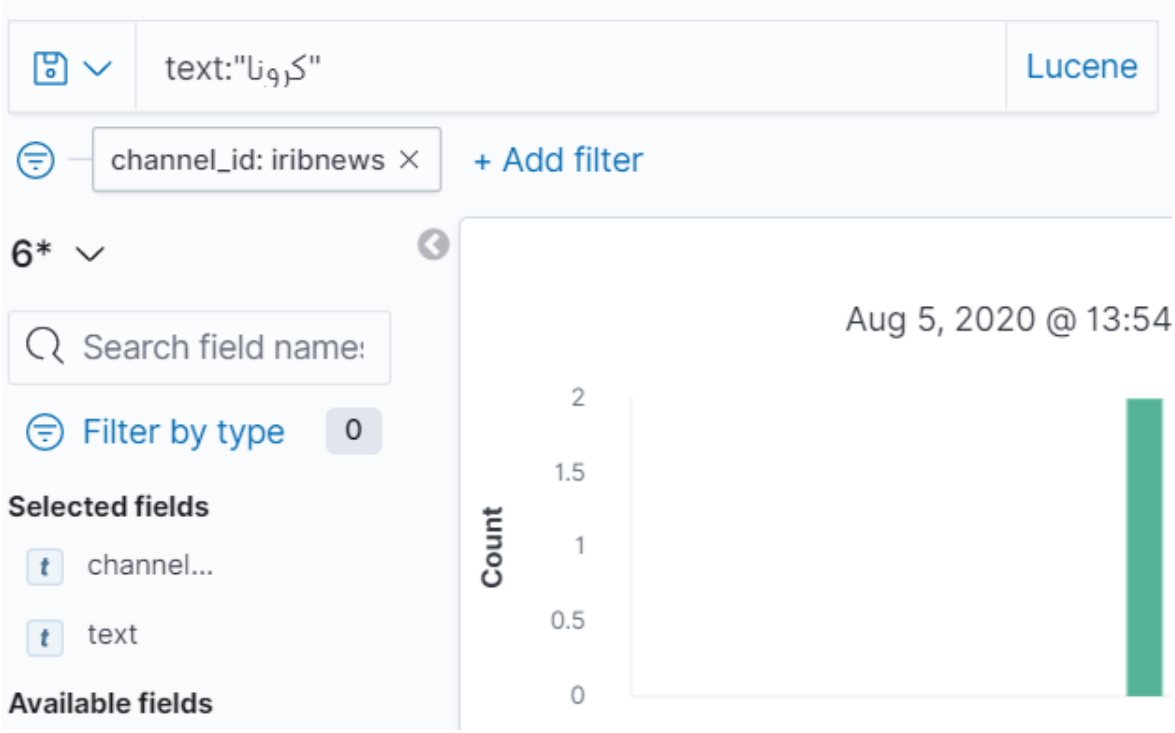


اگر به دنبال پست های یک کلمه خاص از یک خبرگزاری خاص هستیم

می توان از قسمت console و با استفاده از کوئری های DSL کوئری را به صورت زیر ایجاد نمود . must برای عملگر AND استفاده شده و از match برای آنکه در فیلد text کلمه کرونا را جستجو کنیم . سپس نتایج را با استفاده از filter فیلتر می کنیم یعنی می گوییم آن نتایجی که channel_id آن ها iribnews است را فیلتر کن. از source هم استفاده شده است تا فقط فیلدهای text , channel_id , timestampISO نمایش داده شود .

```
GET 6/_search
{
  "_source":["channel_id","text","timestampISO"],
  "query": {
    "bool": {
      "must": [
        {"match": {
          "text": "کرونا"
          }
        }
      ],
      "filter": [
        {"term": {
          "channel_id": "iribnews"
        }}
      ]
    }
    }
  ,"size": 10000
}
```

هم چنین می توان کوئری خواسته شده را در داشبورد نیز نمایش داد . در بخش Discover به صورت شکل زیر نتایج را به دست می آوریم . از کوئری Lucene استفاده می کنیم و می گوییم آن هایی را بده که در text کلمه کرونا وجود دارد سپس یک فیلتر ایجاد می کنیم که آن هایی را که channel_id آن ها iribnews است را نگهدار.

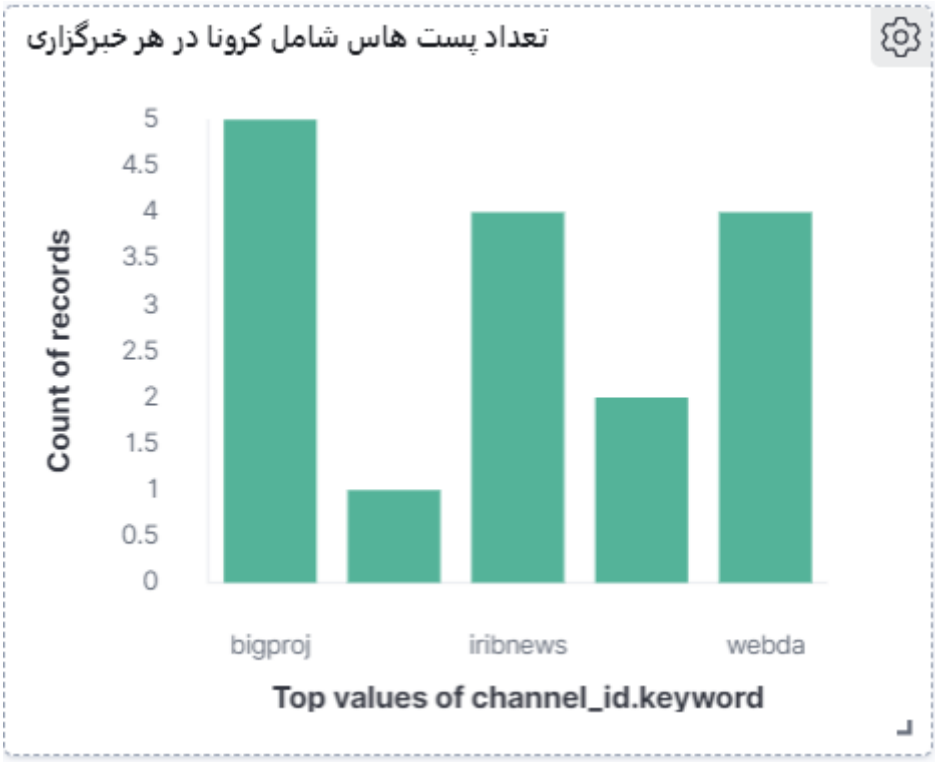


در داشبورد هم به صورت زیر نمایش داده می شود .

تعداد پست های شامل یک کلمه خاص به ازای هر کانال

در این قسمت هم از lens استفاده می کنیم . محور x را channel_id در نظر گرفته و محور عمودی را تعداد record ها و در قسمت سرچ هم از lucene استفاده نموده و می گوییم "کرونا":text . بدین معنی که هر پستی که در متن آن کلمه کرونا آمده باشد .

در داشبورد نمودار به صورت زیر خواهد بود :



گام چهارم – statistics

ردیس یک بانک اطلاعاتی in-memory است که از نوع key-value می باشد . این باک اطلاعاتی رایگان و open source است . ویژگی in-memory بودن باعث افزایس سرعت و بهبود کارایی در پاسخ دهی می باشد . از مزایای دیگر ردیس آن است که از data type های مختلفی پشتیبانی می کند .

Data Types[1] :

۱. Strings :

String ها مجموعه ای از بایت ها هستند که می توانند حداکثر ۵۱۲ مگابایت باشند . برای نمونه در شکل زیر یک کلید به نام name و که مقدار string آن tutorialspoint است و با استفاده از set می توان آن را ایجاد نمود و با استفاده از get و کلید می توان مقدار را دریافت نمود .

```
redis 127.0.0.1:6379> SET name "tutorialspoint"
OK
redis 127.0.0.1:6379> GET name
"tutorialspoint"
```

دیگر توابعی که برای strings می توان استفاده نمود به صورت زیر هستند :

```
APPEND, BITCOUNT, BITFIELD, BITOP, BITPOS, DECR, DECRBY, GET, GETBIT, GETRANGE,
GETSET, INCR, INCRBY, INCRBYFLOAT, MGET, MSET, MSETNX, PSETEX, SET, SETBIT, SETEX,
SETNX, SETRANGE, STRLEN
```

۲. Sets

این نوع در واقع یک کالکشن از string ها است . با توجه به مثال زیر یک کالکشن با نام کلید myset داریم و عضو ها ی ۱ و ۲ و ۳ را با استفاده از SADD به آن اضافه می کنیم . سپس با استفاده از smember می توان مقادیر آن را مشاهده نمود .

[1] https://redis.io/topics/data-types-intro

```
> sadd myset 1 2 3
(integer) 3
> smembers myset
1. 3
2. 1
3. 2
```

از دیگر توابع مربوط به set ها :

```
SADD, SCARD, SDIFF, SDIFFSTORE, SINTER, SINTERSTORE, SISMEMBER, SMEMBERS, SMOVE,
SPOP, SRANDMEMBER, SREM, SSCAN, SUNION, SUNIONSTORE
```

۳. Lists :

لیست ها در واقع مجموعه ای از string ها هستند که به ترتیب اضافه شدن در لیست مرتب شده اند و می توان به اول و آخر لیست یک عضو جدید اضافه نمود . در زیر مثالی را می بینیم . می توان از RPUSH برای اضافه نمودن عضو جدید به انتهای لیست و LPUSH برای اضافه نمودن عضو جدید به ابتدای لیست استفاده نمود . از LRANGE هم برای دریافت اعضای متعلق به یک بازه استفاده می شود .

```
> rpush mylist A
(integer) 1
> rpush mylist B
(integer) 2
> lpush mylist first
(integer) 3
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
```

از دیگر توابع مربوط به لیست ها می توان به موارد زیر اشاره نمود :

```
BLPOP, BRPOP, BRPOPLPUSH, LINDEX, LINSERT, LLEN, LPOP, LPUSH, LPUSHX, LRANGE,
LREM, LSET, LTRIM, RPOP, RPOPLPUSH, RPUSH, RPUSHX
```

Sortes Sets.۴

شبیه sets است با این تفاوت که یک مقدار score دارد و به صورت مرتب شده براساس score قرار می گیرند و نمی توان عضو تکراری به آن اضافه نمود . برای نمونه مثال زیر آورده شده است . با استفاده از ZADD می توان عضو هایی به set اضافه نمود و با استفاده از ZRANGE می توان یک range از اعضا رانمایش داد .

```
redis>    ZADD myzset 1 "one"
(integer) 1
redis>    ZADD myzset 1 "uno"
(integer) 1
redis>    ZADD myzset 2 "two" 3 "three"
(integer) 2
redis>    ZRANGE myzset 0 -1 WITHSCORES
1) "one"
2) "1"
3) "uno"
4) "1"
5) "two"
6) "2"
7) "three"
8) "3"
```

Hashes.۵:

برای عمل mapping بین مقدر field و value است . برای نمونه در زیر می بینیم که یک کلید داریم به نام user:1000 که می توان نام خانوادگی و تاریخ تولد را برای آن set نمود . و با استفاده از hset , hget می توان اطلاعات را ذخیره و دریافت نمود . اگر چند فیلد باشد از hmset استفاده می نماییم .

```
> hmset user:1000 username antirez birthyear 1977 verified 1
OK
> hget user:1000 username
"antirez"
> hget user:1000 birthyear
"1977"
```

دیگر توابع مورد استفاده برای hash ها :

```
HDEL, HEXISTS, HGET, HGETALL, HINCRBY, HINCRBYFLOAT, HKEYS, HLEN, HMGET, HMSET,
HSCAN, HSET, HSETNX, HSTRLEN, HVALS
```

حال با توجه به Data Type ها مختلف و توابع هر کدام می خواهیم اطلاعات مورد نیازمان را از پست های پیام رسان سروش به دست آوریم .

برای اجرای ردیس تنها لازم است تا از سایت https://redis.io/ ردیس را دانلود نموده وسپس فایل redis-server.exe را اجرا نماییم . برای کار با ردیس هم می توان از redis-cli.exe استفاده نمود و هم از زبان ها مختلفی که از ردیس پشتیبانی می کنند . ما در این پروژه از زبان پایتون استفاده می کنیم . بنابراین لازم است تا کتابخانه redis را نصب کنیم . ردیس روی پورت ۶۳۷۹ localhost اجرا می شود . بنابراین به صورت زیر می توان از طریق پایتون به ردیس متصل شد :

```
r = redis.Redis(host='localhost', port=6379, db=0)
```

سپس به راختی می توان از تمام توابع معرفی شده در بالا استفاده نمود .

برای دریافت اطلاعات مورد نیازهر مسئله باید مشخص کنیم که از چه data type استفاده می کنیم و کلید ها را چگونه تعریف کنیم و سایر مواردی که مورد نیاز است که در هر بخش توضیح داده خواهد شد .

برای دریافت اطلاعات از کافکا و ذخیره در ردیس لازم است تا در مرحله اول یک consumer ایجاد کنیم و سپس به صورت real time هر پست جدید را خوانده و اطلاعات مورد نیاز را استخراج نماییم و در ردیس قرار دهیم

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
consumer = topic.get_simple_consumer()
```

هر پستی که از طریق consumer دریافت می شود را دیکود نموده و با استفاده از ast به json تبدیل می کنیم .

```
msg = (message.value).decode("utf-8")
msg = ast.literal_eval(msg)
```

سپس به سراغ استخراج اطلاعات و قرار دادن آن ها در ردیس می رویم .

نکته ای که وجود دارد این است که با توجه به خواسته مسئله که بیان شده اطلاعات هر کلید تا یک هفته در ردیس باقی بماند و پس از آن به طور خودکار حذف شود ، لازم است تا از تابع expire از ردیس استفاده نماییم . به اینصورت عمل می کند که key و یک مقداری به ثانیه را دریافت نموده و پس از گذشتن آن میزان ثانیه کلید مورد نظر از ردیس حذف می شود . بنابراین ما هر بار که کلید جدیدی را ایجاد می کنیم از تابع expire برای حذف آن بعد از یک هفته استفاده می کنیم .

الف) تعداد پست های ارسال شده توسط یک کانال در ۶ ساعت گذشته .

برای اینکه بتوانیم به خواسته سوال برسیم لازم است تا کلیدی تعریف کنیم که شامل نام کانال و تاریخ دریافت آن پیام از کانال باشد . برای تاریخ هم تنها به روز و ماه و ساعت دریافت پیام نیاز داریم .

هر بار که پیامی دریافت میشود می توان نام کانال را از فیلد channel_id مشاهده نمود و به عنوان channel_key در نظر می گیریم . سپس روز و ماه و ساعت دریافت این پیام را از ()datetime.now بدست می آوریم و یک hash ایجاد می کنیم که کلید آن نام کانال به همراه تاریخ دریافت پیام توسط ردیس است و فیلد آن تاریخ دریافت پیام و مقدار آن برابر تعداد باری که در آن تاریخ آن کانال پیام داده است .

```
r.hincrby(channel_key+nowtime,nowtime,1)
r.expire(channel_key+nowtime,14515200)
```

تابع hincrby به این صورت عمل می کند که کلید و فیلد مورد نظر را به همراه یک عدد می گیرد و سپس آن عدد را به مقدار قبلی آن کلید و فیلد اضافه می کند . اگر آن کلید و فیلد در ردیس وجود نداشته

باشد آن ها را ایجاد می کند و مقدار قبلی را صفر در نظر می گیرد . به عنوان مثال اگر از خبرگزاری farsna در روز 15 مرداد ساعت 17 یک پیام دریافت کنیم کلید به صورت farsna051517 و فیلد به صورت 051517 خواهد بود . حال اگر برای اولین بار این پیام از farsna در این زمان دریافت شده باشد این کلید و فیلد را به دیتابیس اضافه نموده و مقدار قبلی را 0 درنظر می گیرد و یکی به آن اضافه می شود . اما اگر از قبل باشد مقدار قبلی را با یک جمع می کند . در خط بعدی از expire استفاده نموده تا پس از گذشت یک هفته (۱۲۵۱۵۲۰۰ ثانیه) آن کلید به صورت خودکار حذف شود .

برای نمایش پاسخ سوال خواسته شده به این صورت عمل می شود که برای نشان دادن تعداد پست های ارسال شده از یک کانال در ۶ ساعت قبل ، ماه و روز و ساعت زمان حال تا ۶ ساعت قبل را محاسبه می نماییم و به همراه نام خبرگزاری به ردیس می دهیم تا تعداد پیام در هر ساعت را به ما بدهد و این مقادیر را با هم جمع می کنیم تا تعداد پیام در ۶ ساعت قبل را بدست آوریم .

```
if(r.hexists(channel_key+last_time,last_time)):
    count += int(r.hget(channel_key+last_time,last_time))
```

همان طور که می بینیم ابتدا با استفاده از hexists چک می کنیم که آن خبرگزاری در آن تاریخ پیامی داشته یا نه . سپس اگر موجود بود مقدارش را با مقادیر سایر ساعات جمع می کنیم .

Count تعداد پیام های یک خبرگزاری در ۶ ساعت قبل را نشان می دهد .

ب)تعداد کل پیام های دریافت شده در یک بازه زمانی مثلا روز گذشته

در این سوال تعداد پیام های دریافت شده در ۲۴ ساعت گذشته را نشان می دهیم . بدین منظور از hash ها استفاده می کنیم .کلید را همان طور که در شکل زیر مشاهده می کنید به صورت count به همراه روز و ماه و ساعت دریافت پیام در نظر می گیریم و فیلد مورد نظر را همان روز و ماه و ساعت . اگر آن کلید در ردیس موجود باشد با دیدن یک پیام در آن ساعت فقط یک مقدار به مقدار قبلی اضافه میشود در غیر این صورت آن کلید و فیلد به ردیس اضافه شده و مقدار یک می گیرد . با استفاده از expire نیز کلید مربوطه پس از یک هفته به طور خودکار حذف میشود .

```
r.hincrby('count'+nowtime,nowtime,1)
r.expire(channel_key+nowtime,14515200)
```

برای نشان دادن تعداد پیام ها در ۲۴ ساعت گذشته روز و ماه و ساعت را تا ۲۴ ساعت قبل به دست می آوریم و با کلمه count ترکیب نموده و کلید و فیلد مورد نظر را ابتدا با hexists چک می کنیم که موجود باشد سپس با استفاده از hget به ردیس می دهیم و تعدا پست در هر ساعت را به دست آورده و در نهایت این مقادیر را جمع می کنیم .

```
if(r.hexists('count'+last_time,last_time)):
    count += int(r.hget('count'+last_time,last_time))
```

پاسخ نهایی در count خواهد بود .

ج) تعداد هشتگ های دریافت شده در یک ساعت گذشته (به صورت منحصر به فرد)

برای به دست آوردن پاسخ این سوال پس از دریافت هر پیام ، فیلد hashtags را گرفته و هشتگ های موجود در ان را به ردیس اضافه می کنیم . برای این قسمت از hashها استفاده می کنیم . با توجه به شکل زیر کلید را برابر hashtags به همراه ماه و روز و ساعت قرار داده و فیلد ها را هم به صورت هشتگ موجود در پیام به همراه روز و ماه و ساعت و دقیقه قرار می دهیم . دلیل اینکه فیلد را به دقیقه وابسته می کنیم آن است که می خواهیم تعداد هر هشتگ در ۶۰ دقیقه گذشته را بدست آوریم .

```
r.hincrby('hashtags'+nowtime,hashtag+nowtimemin,1)
r.expire('hashtags'+nowtime,14515200)
```

زمانی که یک پست دریافت میشود به ازای هر هشتگ موجود در متن یک فیلد به کلید hashtags + nowtime اضافه می شود که این فیلد وابسته به روز و ماه وساعت و دقیقه است .

برای پیدا کردن تعداد تکرار هر هشتگ در ۶۰ دقیقه گذشته ابتدا باید ببینیم که در ۶۰ دقیقه قبل چه هشتگ هایی دریافت شده است بنابراین می توان به صورت زیر فیلدهای مربوط به کلید ساعت حالا و یک ساعت قبل را به دست آورد .

```
hashtags = [i.decode("utf-8")[:-8] for i in r.hgetall('hashtags'+nowtime)] +
        [i.decode("utf-8")[:-8] for i in r.hgetall('hashtags'+last_time)]
```

بنابراین تا اینجا به دست آوردیم که در یک الی دو ساعت قبل چه هشتگ هایی داشتیم و پس از آن برای هر کدام از این هشتگ ها می خواهیم ببینیم که آیا در ۶۰ دقیقه قبل تکرار شده یا خیر ؟ و اگر تکرار شده است چه تعداد بار تکرار شده است . بنابراین به صورت زیر عمل می کنیم

```
if(r.hexists('hashtags'+last_time,hashtag+last_time_min) ):
    c += int(r.hget('hashtags'+last_time,hashtag+last_time_min))
if(r.hexists('hashtags'+ nowtime,hashtag+last_time_min) ):
    c += int(r.hget('hashtags'+nowtime,hashtag+last_time_min))
```

برای بررسی حضور یک هشتگ در ۶۰ دقیقه قبل لازم است تا فیلد را شامل هشتگ مورد نظر به همراه روز و ماه و ساعت و دقیقه زمان حال و ۶۰ دقیقه قبل درنظر بگیریم و مقدار آن را که برابر تعداد تکرار آن هشتگ در آن دقیقه است را به دست آورده و همه را با هم جمع کنیم .

مقادیر به دست آمده برای هر هشتگ را در یک set ذخیره می کنیم .

د) آخرین هشتگ های دریافت شده .

در این قسمت از لیست ها استفاده می کنیم به این صورت که هر پستی که به دست می آوریم ، هشتگ های موجود در آن را به ابتدا یک لیست با نام hashtagList با استفاده از تابع LPUSH اضافه می کنیم و پس از آن با استفاده از تابع LTRIM فقط عضوهای 0 تا 999 را نگه می داریم و بقیه حذف می شوند .

```
r.lpush('hashtagList',hashtag)
r.ltrim('hashtagList',0,999)
```

برای به دست آوردن ۱۰۰۰ هشتگ آخر دریافت شده نیز می توان از تابع LRANGE استفاده نمود :

```
r.lrange('hashtagList',0,999)
```

ه) اخرین پس های دریافت شده

در این قسمت هم از لیست ها استفاده می کنیم به این صورت که هر بار پیامی دریافت شد متن موجوددر فیلد text را در ابتدای لیست postList قرار می دهیم و سپس با استفاده از LTRIM تنها ۱۰۰ داده ابتدای لیست را نگه می داریم و بقیه را حذف می کنیم .

```
r.lpush('postList',msg['text'])
r.ltrim('postList',0,99)
```

برای نشان دادن ۱۰۰ پیام آخر نیز از تابع LRANGE استفاده می کنیم :

```
r.lrange('postList',0,99)
```

برای هر کدام از نمونه های بالا یک تابع ایجاد شده است . همه این توابع را در یک تابع به نام result فراخوانی می کنیم و پاسخ را به صورت string بازمی گردانیم .

سپس با استفاده از یک app فلسک آن ها را نمایش می دهیم که به صورت زیر نوشته شده است :

```
app = Flask(__name__)

@app.route('/')
def resflask():
    return result()

if __name__ == '__main__':
    app.run()
```

نتیجه به صورت زیر است :

که با هر بار رفرش نمودن صفحه توابع فراخوانی شده و داده ها به صورت real time نمایش داده می شود .

number of post from channel tasnimna in 6 hour ago is 7

---

total number of posts in 24 hours ago is 41

---

count of each hashtags in 60 min ago : {'انتشار_پیام_سهیم_شدن_در_ثواب': 58, 'ایران': 29, 'من_ماسک_میزنم': 87, 'کرونا': 58, 'سلام_آقا': 29, 'انفجاربیروت': 58, 'جان_ما_ایران_ما': 87}

---

last 1000 hashtags :

انفجاربیروت|جان_ما_ایران_ما|من_ماسک_میزنم|سلام_آقا|کرونا|جان_ما_ایران_ما|من_ماسک_میزنم|انتشار_پیام_سهیم_شدن_در_ثواب|انتشار_پیام_سهیم_شدن_در_ثواب|
جان_ما_ایران_ما|من_ماسک_میزنم|انفجاربیروت|ایران|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|ایران|خبرهای_کوتاه|
خبرهای_کوتاه|خبرهای_کوتاه|خبرهای_کوتاه|جان_ما_ایران_ما|من_ماسک_میزنم|کرونا|کرونا|کرونا|کرونا|
کرونا|هر_ایرانی_یک_مهرورز_سلامت،|جان_ما_ایران_ما|من_ماسک_میزنم|هر_ایرانی_یک_مهرورز_سلامت،|جان_ما_ایران_ما|من_ماسک_میزنم|اپلفوگرافیک|جان_ما_ایران_ما|من_ماسک_میزنم|
کووید19|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|
من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|جان_ما_ایران_ما|من_ماسک_میزنم|

---

last 100 post is :

0) @iribnews .مقدسی، رئیس انجمن صنفی گاوداران: گوشت قرمز دام سنگین با بهترین کیفیت در هر کیلو نباید بیش از 85 هزار تومان به فروش برسد
1) @iribnews اذان مغرب به افق تهران
2) @iribnews .مقدسی، رئیس انجمن صنفی گاوداران: گوشت قرمز دام سنگین با بهترین کیفیت در هر کیلو نباید بیش از 85 هزار تومان به فروش برسد
3) ز شهرها در دستور کار قرار گیرد رئیس سازمان پدافند غیرعامل کشور: ◆حادثه بیروت برای همه دنیا از منظر پدافند غیرعامل، میتواند عبرت باشد، چه در حوزه جانمایی و چه نگهداری از مواد تیمیایی و چه نحوه .ر برابر بلایایی که می تواند در یک روز تمام یک جامعه را بهم بریزد. ◆به نظرم ضرورت دارد تا موضوع پدافند تیمیایی و خروج تاسیسات پرخطر از شهرها بیشتر از گذشته در دستور کار قرار گیرد. #انفجاربیروت
4) @iribnews وضعیت ترافیکی بزرگراه های تهران
5) ـه سازمان اورژانس کشور: ♦اورژانس یکی از مهمترین و مطمئن ترین تکیه گاه‌های ما بوده و خواهد بود ♦با توسعه اتوبوس آمبولانس‌ها و تقویت داوگان زمینی اورژانس توانستیم به ۳.۵میلیون زائر اربعین خدمات w مدافعان سلامت #من_ماسک_میزنم #جان_ما_ایران_ما پویش همدلی با . https://b2n.ir/845413 مورد نیاز را ارائه کنیم ♦ تعداد موتور آمبولانس‌ها در شهرهای شلوغ و پر ترافیک نیز ۲ برابر خواهد شد
6) @iribnews حاجتش بی برو برگرد روا خواهد شد هر کسی دست به دامان رضا خواهد شد دست خالی نرود هیچکسی از حرمتش سجده کردند ملائک همه پای علمتش السلام علیک یا امام الرئوف #سلام_آقا
7) @TasnimNs آخرین اطلاعات درباره #کرونا و ویژگی‌های آن از زبان فرمانده عملیات مدیریت بیماری کووید۱۹ در تهران
8) @TasnimNa علت اختلاف قیمت گوشت قرمز در مناطق مختلف تهران
9) @TasnimNs آخرین اطلاعات درباره #کرونا و ویژگی‌های آن از زبان فرمانده عملیات مدیریت بیماری کووید۱۹ در تهران
10) @iribnews تصویری از آتش سوزی انبار مواد غذایی در نجف اشرف
11) @iribnews آخرین اطلاعات درباره ویروس کرونا و ویژگی های آن از زبان فرمانده عملیات مدیریت بیماری کووید19 در کلان شهر تهران
12) چه پروتکل های نگهداری‌هشدار سازمان پدافند غیرعامل: خروج تاسیسات پرخطر از شهرها در دستور کار قرار گیرد رئیس سازمان پدافند غیرعامل: ◆حادثه بیروت برای همه دنیا از منظر پدافند غیرعامل، نیز مو .از مواد تیمیایی. ◆ضرورت دارد تا موضوع خروج تاسیسات پرخطر از شهرها بیشتر از گذشته در دستور کار قرار گیرد. دولت باید برای تحقق آن کمر همت ببندند fna.ir/eywj3d @Farsna

---