طراحی یک سامانه بلادرنگ برای تحلیل لحظه‌ای داده‌های توئیت فارسی

امیر سرتیپی[۱]، نرگس اسدی[۲]، مهسا عباسپور[۳]

[۱] دانشجوی مقطع ارشد، خوشه کلان داده، دانشگاه اصفهان ، اصفهان،
amirsartipi13@gmail.com

[۲] دانشجوی مقطع ارشد، خوشه کلان داده، دانشگاه اصفهان ، اصفهان،
Asadi.n.eng@gmail.com

[۳] دانشجوی مقطع ارشد، خوشه کلان داده، دانشگاه اصفهان ، اصفهان،
M75.abbaspour@gmail.com

چکیده

در این پروژه یک سیستم بلادرنگ برای تحیل داده‌های توئیتر به زبان فارسی راه‌اندازی می‌شود. این سامانه از تکنلوژی‌هایی همچون اسپارک، کافکا، الستیک و فلسک برای ارائه اطلاعات استفاده می‌کند. همچنین در طول پروژه کتاب‌خانه‌های مختلفی برای پیش‌پردازش داده‌ها، برقراری ارتباط با ای‌پی‌آی استفاده شده. سامانه توئیت‌ها را به صورت بلادرنگ دریافت کرده و اطلاعات جامع و کاملی را در اختیار کاربر سیستم قرار می‌دهد.

کلمات کلیدی

سیستم بلادرنگ، یادگیری ماشین، پردازش متن

۱- مقدمه(سرتیپی)

در ابتدا ۵ فایل پایتونی که هرکدم نماینده یک کانال کافکا می‌باشد ساخته شد. همچنین دو فایل پایتونی مربوط به نمایش داده‌ها از طریق فلسک ساخته شده است.

۱-۱- کانال‌های کافکا(سرتیپی)

از کتاب‌خانه‌ی python-kafka برای برقراری ارتباط کلاینت با سرور کافکا استفاده شده است. در هر مرحله داده‌ها توسط یک producer برای یکی از کانال‌های طراحی شده kafka ارسال می‌شود و در کلاینت‌ها توسط یک consumer این اطلاعات دریافت می‌شوند.
نمونه‌ی دو دستور ارسال داده برای کانال‌های کافکا در زیر آورده شده است.

```
producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda v: json.dumps(v).encode('utf-8'))

consumer = KafkaConsumer('persistence',
                         auto_offset_reset= 'earliest',
                         auto_commit_interval_ms = 1000)

producer.send('channel-history',
value=json.loads(json.loads(msg.value)))
```

همچنین شبه کد زیر برای ساختن topic هایی که برای کانال‌های کافکا نیاز است نوشته شده است که برای نخستین مرتبه که این topic ساخته نشده‌اند نیاز است تا فایل 0-create_topic.py اجرا بشود.

```
from kafka.admin import KafkaAdminClient, NewTopic
admin_client = KafkaAdminClient(
  bootstrap_servers="localhost:9092",
)
topics = ['pre-process', 'persistence', 'channel-history', 'statistics']
topic_list = []
for topic in topics:
  topic_list.append(NewTopic(name=topic, num_partitions=1,
admin_client.create_topics(new_topics=topic_list, validate_only=False)
```

۲- دریافت اطلاعات(سرتیپی)

در این بخش نحوه‌ی دریافت اطلاعات توضیح داده شده است.

۲-۱- دریافت اکانت توسعه‌دهنده(سرتیپی)

در این پروژه برای فراهم کردن داده‌ها از توئیت‌های فارسی شبکه‌ی اجتماعی توئیتر استفاده شده است. نخست یک اکانت توئیتری ساخته شد و پس از آن در بخش دریافت اکانت توسعه دهنده ثبت‌نام شد. پس از توضیح شرح پروژه و هدف از آن توئیتر یک اکانت با پارامترهای کلید ای‌پی‌آی[۱]، راز رمز ای‌پی‌آی[۲]، نشانه‌ی حامل[۳]، نشانه‌ی دسترسی[۴] و نشانه‌ی دسترسی[۵] راز را اهدا کرد. نشانه‌ی یاد شده برای خزش و استخراج توئیت‌های فارسی استفاده شده است تا پس از دریافت آن طبق فایل خزش در پروژه توئیت‌ها را با کانال کافکا ارسال کند.

```
API_Key = 'Eo3ej5ML2UbkkJUEcF78lEyOA'
API_Secret_Key = 'Qt1ED20LEAhrgWk1dBPP7QCy1LO0As24y5Kp2K1gedJPxTqm0II'

Bearer_Token =
'AAAAAAAAAAAAAAAAAAAAAB1USAEAAAAAEfCgv2RY0waIj7mCpnNDmITY0cM%3D99Yx4oDNnX8Ky2oUYeuqvANZR
f5YwlQIO5zH7zWD5dRC5Zsdvs'
Access_Token = '1419011209307639810-n16S77LvjatXpRftPNI2YcBjleiJvV'
Access_Token_Secret = 'tyTVLcojbETSeq0NFfoA8UHP8Qdf50WoDYpnEgXU5m2S2'
```

۲-۲- دریافت توئیت‌ها(سرتیپی)

از کتاب‌خانه‌ی TwitterSearch برای دریاف توئیت‌ها از API های شبکه‌ی اجتماعی توئیتر استفاده شده است که می‌تواند پارامترهایی مانند زبان، تعداد توئیت، کلمه‌های کلیدی و پارامترهای دیگری برای آن تعیین کرد. برای انتخاب کلمه‌های کلیدی استفاده شده در این بخش از google trend استفاده شده است و مجموعه‌ای از کلماتی که بیشتر جستجو شده‌اند به علاوه‌ی کلمات کلیدی در داکیومنت پروژه به عنوان کلماتی استفاده شده‌اند که توئیت‌ها از این کلمات بازیابی شوند.
کلاس twitter_crawller با دریافت تعداد توئیت‌هایی که می‌خواهیم بازیابی کنیم، زبان مورد نظر، کلمات کلیدی

مورد نظر و اطلاعات تعیین هویت ما، شروع به استخراج توئیت‌ها از توئیتر می‌کند.

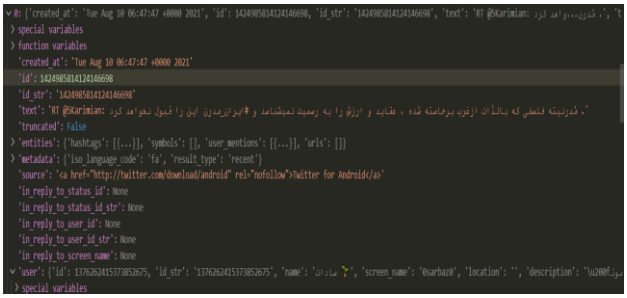
```
def twitter_crawller(count):
    data = []
    try:
        tso = TwitterSearchOrder()
        # Set keywords which we want to crawll
        tso.set_keywords(keywords)
        # Set language
        tso.set_language('fa')
        tso.set_include_entities(True)
        tso.set_count(count)

        # Authorization to twitter API
        ts = TwitterSearch(
            consumer_key = API_Key,
            consumer_secret = API_Secret_Key,
            access_token = Access_Token,
            access_token_secret = Access_Token_Secret
        )

        for tweet in ts.search_tweets_iterable(tso):
            # print( '@%s tweeted: %s' % ( tweet['user']['screen_name'], tweet['text'] )
)
            data.append(tweet)
    except TwitterSearchException as e:
        print(e)

    return data
```

در شکل زیر می‌توان یک از توئیت‌های دریافت شده توسط API را مشاهده کرد که هنوز پیش پردازشی روی آن انجام نشده است.



۳-۲- ارسال به داده‌ها به کانال پیش پردازش(سرتیبی)

در برنامه پس از دریافت داده‌ها، با استفاده از کتابخانه‌ی Kafka به کانال طراحی شده با نام pre-process ارسال می‌شود. این عملیات به صورت جریان داده اتفاق می‌افتد و در مرحله‌ی بعدی یعنی پیش پردازش داده‌ها، داده‌ها از طریق کتاب‌خانه کافکا دریافت شده و پیش‌پردازش بر روی آن‌ها صورت می‌گیرد.

۳- پیش‌پردازش داده‌ها(سرتیبی)

در این بخش عملیات‌هایی که برای پیش‌پردازش بر روی توئیت‌ها صورت گرفته است بیان می‌شود.

۳-۱- ایست واژه‌ها و افعال(سرتیبی)

در این بخش از چندین منبع که ایست داده‌های فارسی مانند افعال، حروف اضافه و دیگر موارد از جمله فعل‌ها جمع‌آوری کرده‌اند استفاده شد و همه‌ی آنها در یک فایل txt جمع‌آوری شد. پس از آن این مجموع پیش‌پردازش و به صورت یکتا درون یک آرایه ذخیره شد. تا برای تحلیل‌گر الستیک و همچنین پیش پردازش توئیت‌ها مورد استفاده قرار بگیرد. این فایل در پوشه files با نام stop_wrods قرار دارد.

۳-۲- لغات ایستا(سرتیبی)

لغاتی که درون سند پروژه به عنوان لغات کلیدی در نظر گرفته شده بود درون یک لیست به عنوان لغات ایستا در نظر گرفته شدند تا هنگام پردازش متن در صورت مشاهده به لیست کلمات کلیدی اضافه بشوند. همچنین لیست انواع حالت نوشتاری کویید نیز به این لیست اضافه شده است.

۳-۳- پیش پردازش(سرتیبی)

در این مرحله ابتدا بر روی هر توئیت یک زمان دریافت توئیت توسط سامانه ایجاد می‌شود که برای پیش‌پردازش‌های بعدی مورد استفاده قرار می‌گیرد. پس از دریافت توئیت لازم است زمانی که این توئیت ارسال شده است نیز ذخیره بشود. همانطوری که در شکل زیر قابل مشاهده است پس از دریافت زمان که به صورت یک رشته می‌باشد، ابتدا سال، ماه و روز از آن استخراج شده و همچنین زمان ارسال توئیت نیز استخراج می‌شود. سپس زمان، تاریخ و تعداد ثانیه‌ها از ابتدا تا به حال (time stamp) بازگردانده می‌شود.

همچنین یک شناسه با فرمت uuid4 برای هر توئیت در نظر گرفته می‌شود که توئیت را از بقیه متمایز کند. برای پیش‌پردازش داده‌ها و توکنایز کردن توئیت‌های دریافتی از کتاب‌خانه‌ی هضم استفاده شده است.

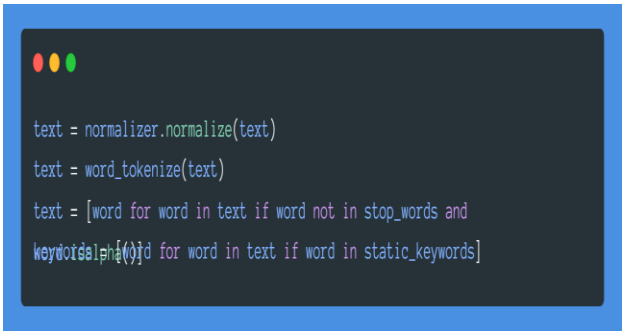
همچنین از regex استفاده شده تا هشتگ‌ها را از متن استخراج کرده و درون یک لیست ذخیره کند.(تکراری‌ها دریافت نمی‌شوند)

همچنین از regex برای استخراج لینک‌ها از متن استفاده شده است که درون یک لیست آن‌ها را ذخیره می‌کنند.

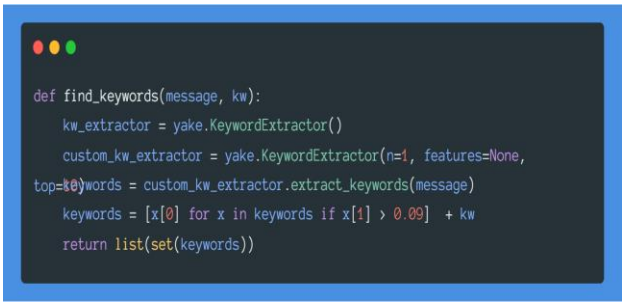
در آخر یک دیکشنری که تبدیل به فایل json می‌شود به صورت زیر می‌باشد.

در ابتدا با نرمال‌سازی متن توسط کتابخانه‌ی هضم و توکنایز کردن آن، اگر کلمات کلیدی که در متن پروژه تعریف شده است در آن وجود داشت از آن استخراج می‌شود. این کلمات به تابع استخراج ویژگی فرستاده می‌شود تا با کمک کتابخانه‌ی yake که برای استخراج کلمه‌ی کلیدی است جمع شود.

در صورتی که کلمات توکنایز شده در لیست ایست‌کلمه‌ها وجود نداشته باشد به توکن‌ها اضافه شده و سپس اگر این توکن‌ها در لیست کلمات استاتیک ما قرار داشت، آن‌ها را به عنوان کلمه‌ی کلیدی در نظر می‌گیریم.



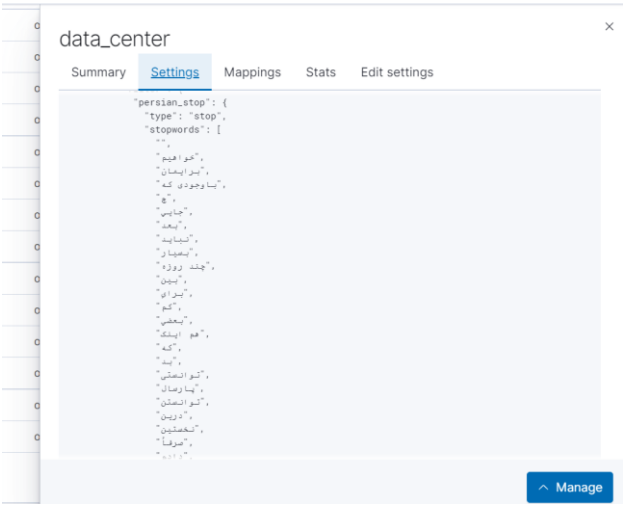
برای استخراج کلمات کلیدی از کتابخانه‌ی yake استفاده شده است. این کتاب‌خانه بر روی مجموعه‌ای از داده‌ها آموزش دیده و برای استخراج ویژگی و کلمه‌ی کلیدی استفاده می‌شود. در شکل زیر این تابع آورده شده است که با دریافت متن، اگر احتمال رخداد بیشتر از مقداری باشد آن‌را به عنوان کلمه‌ی کلیدی در نظر گرفته و با کلمات استخراجی قبلی جمع کرده و لیست یکتایی از کلمات کلیدی را برای ما باز می‌گرداند.



در نهایت یک دیکشنری که حاصل پیش‌پردازش متن توئیت است به شکل زیر بازگردانده می‌شود که همین دیکشنری به json تبدیل شده و به کانال‌های دیگر کافکا ارسال و در الستیک نیز ذخیره خواهد شد.

۴-۱- ایست واژه‌ها به صورت سفارشی(سرتیپی)

در تصویر زیر مشاهده می‌شود که لیست ایست واژه‌هایی که در مراحل قبل نیز استفاده شد، به صورت سفارشی به الستیک ارسال شده‌اند.



۴-۲- ساخت ایندکس(سرتیپی)

برای ساخت ایندکس که داده‌ها بر روی آن ذخیره شود از کتاب‌خانه elasticsearch استفاده شده است که یک رابط بین برنامه و سروری که الستیک بر روی آن در حال اجرا است می‌باشد. تابعی که در زیر شبه کد آن آورده شده است در صورتی که index با نام data_center وجود نداشته باشد آن‌را می‌سازد و در غیراینصورت ایندکس را دریافت می‌کند.

```
def create_or_get_es(index):
    es = Elasticsearch([{'host': 'localhost',
                         'requestTimeout': 60000,
                         'port': 9200}],
                         timeout=30)
    try:
      es.indices.create(index=index, body=body)
    except:
      es.indices.get(index=index)
    return es
```

```
return {
        "id":id,
        "text":text,
        "user": tweet['user']['screen_name'],
        "ctext":text,
        "pdate":pdate,
        "ptime":ptime,
        "ptimestamp": ptimestamp,
        "date":date,
        "hashtags":hashtags,
        "urls":urls,
        "keywords":keywords
}
```

۴-۳- ذخیره توئیت‌ها در فایل json(سرتیپی)

برای راحتی و در اختیار داشتن داده‌های کرال شده از API شبکه‌ی توئیتر، همه‌ی داده‌هایی که پیش‌پردازش می‌شوند علاوه بر ذخیره بر روی الستیک‌سرچ در یک فایل json در پوشه فایل نیز ذخیره می‌شوند تا تمامی داده‌های مورد نیاز یک توئیت در اینجا نیز وجود داشته باشد. شبه کد زیر وظیفه‌ی ذخیره یک فایل json در پوشه‌ی files با نام فایل data را بر عهده دارد.

```
def write_json(new_data, filename='./files/data.json'):
    with open(filename,'r') as file:
        file_data = json.load(file)
        file_data["data"].append(new_data)
        file.seek(0)
        json.dump(file_data, file, indent = 4)
```

۴- ماندگاری(سرتیپی)

در این بخش عملیات‌هایی که برای ساخت الستیک صورت می‌گیرد بیان می‌شود.

چون داده‌ها به زبان فارسی می‌باشد از آنالایزر مخصوص برای زبان فارسی استفاده شده که این آنالایز در قسمت بدنه درخواست ساخت ایندکس ارسال می‌شود. در این قسمت کلمات توقفی آنالایز به صورت شخصی سازی از همان کلماتی که در مرحله‌ی قبل بدست آورده شد استفاده می‌شود.

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

پس از دریافت داده‌ها از کانال pre-process کافکا آن‌را برای الستیک ارسال می‌کنیم. از طریق کد زیر داده‌ی json را برای الستیک ارسال می‌کنیم.

```
es.index('data_center', tweet)
```

۴-۳- داده‌های ایندکس شده(اسدی)

داده‌های ارسال شده به الستیک در پورت ۹۲۰۰ قابل مشاهده هستند.

توئیت‌ها در ایندکسی با نام data-center و با ساختار تعیین شده در برنامه، در الستیک ذخیره می‌شوند.

۴-۴- ارسال داده‌ها به کافکا

انجام آنالیز بر روی داده‌های الستیک توسط کیبانا انجام می‌شود که در قسمت‌های بعدی به آن پرداخته می‌شود. اما به این دلیل که ارسال داده‌ها به کانال بعدی کافکا نیازی به این پردازش‌ها ندارد، با استفاده از دستور زیر آن‌ها را به کانال channel-history ارسال می‌کنیم.

```
producer.send('channel-history', value=json.loads(json.loads(msg.value)))
```

۴-۵- آنالیز داده‌ها در کیبانا(اسدی)

پس از نصب کیبانا بر روی سیستم، از طریق پورت ۵۶۰۱ می‌توان به آن دسترسی داشت.

از طریق منوی Discover در کیبانا می‌توان داده‌های ذخیره شده در هر ایندکس الستیک را مشاهده کرد.

از منو Analytics می‌توان یک داشبورد جدید برای ایندکس مدنظر ایجاد کرد. سپس نتایج تحلیل‌ها و نمودارهای مربوط به آن ایندکس را به طور متمرکز در یک داشبورد قرار داد.

۱-۵-۴- ابر کلمات(اسدی)

با استفاده از ابر کلمات می‌توان پرتکرار ترین کلمات در توئیت‌ها را به خوبی نشان داد.

برای این کار از قسمت visualize library، aggregation based و سپس tag cloud انتخاب می‌شود. در تنظیمات سمت راست صفحه، فیلد مدنظر تعیین می‌شود که ما از کلمات کلیدی استفاده کردیم. همچنین در قسمت KQL می‌توان بازه زمانی مدنظر را وارد کرد.

سپس ابر کلمات بدست آمده را می‌توان به داشبورد اضافه کرد.

۲-۵-۴- متن ده توئیت اخیر(اسدی)

برای مشاهده متن ده توئیت اخیر به صورت زیر عمل می‌شود. در منو Discover و در بخش add filter می‌توان از کوئری‌ها برای اعمال فیلتر بر روی داده‌ها استفاده کرد. در کوئری نوشته شده برای پاسخ به این پرسش، مرتب سازی داده‌ها به صورت نزولی و براساس فیلد timestamp انجام می‌شود سپس با مقداردهی عدد ۱۰ به size، ده مقدار اول برگردانده می‌شوند.

در این منو با انتخاب فیلد مدنظر می‌توان تنها بخشی از فیلدهای هر توئیت را برای نمایش در خروجی انتخاب کرد. به طور مثال برای مشاهده متن توئیت‌ها تنها فیلد ctext انتخاب می‌شود.

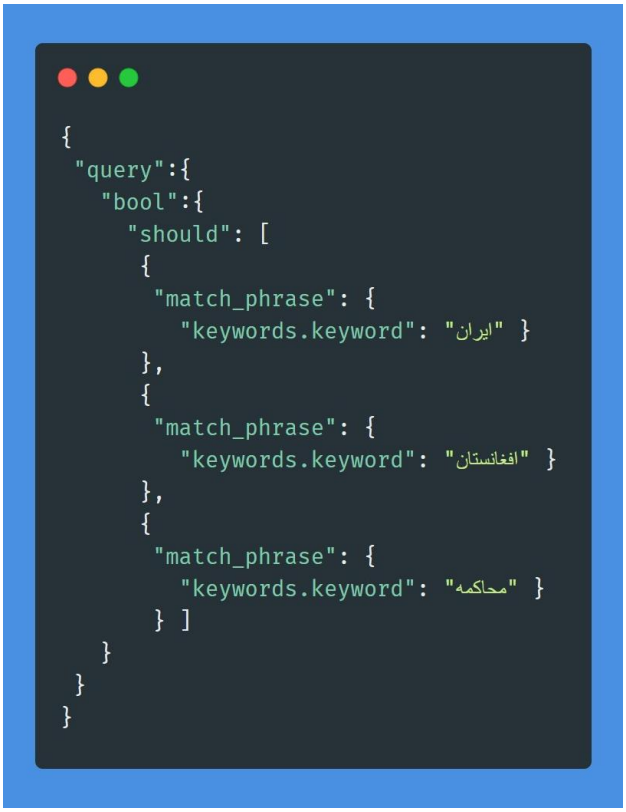
```
{
 "query":{
   "match_all": {}
 },
  "size":10,
  "sort": [
    {
      "ptimestamp": {
        "order": "desc"
      }
    }
  ]
}
```

```
> حاجی ایران بده
> (empty)
> روزها زبان یادشان افتاده وزیر بهداشت بتازند جماعت
> ایران شبیه مافیا اداره حکومت مایک پمپئو
> (empty)
> آه ناله خانوم همسرشو دست قلبمو درد منو زجر میده اینه شاه
> وزهایی صدا سیما اوج وقاحت شوخی میکرد رقص نشون میداد میگفت
> جمهوری اسلامی
> (empty)
> پلاکش ایران اطلاع بده بگیرنش عکس میتونی تاحالا اطلاع
```

۳-۵-۴- تعداد توئیت به ازای کلمات کلیدی خاص(اسدی)

برای مشاهده تعداد توئیت‌های ارسال شده به ازای چند کلمه کلیدی خاص و در یک بازه زمانی مشخص به صورت زیر عمل می‌شود.

پس از انتخاب NewVisualization، این بار گزینه‌ی Lens انتخاب می‌شود. همچون قبل، بازه زمانی مورد نظر تعیین و با انتخاب گزینه add filter کوئری مناسب وارد می‌شود.
در این کوئری تعداد توئیت‌هایی که دارای کلمات «ایران» یا «افغانستان» و یا «محاکمه» باشند مورد پرسش قرار گرفته است.

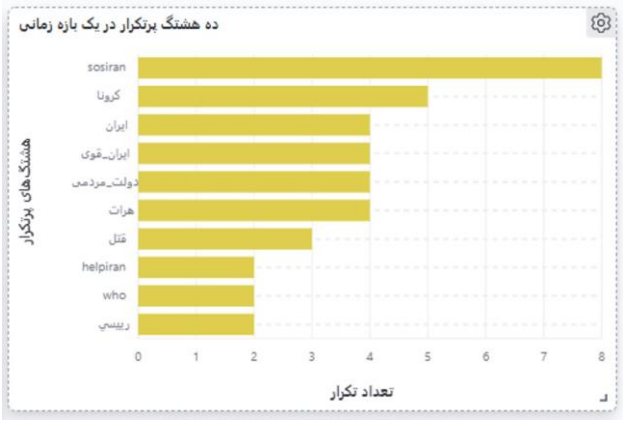


```
{
 "query":{
   "bool":{
     "should": [
      {
       "match_phrase": {
         "keywords.keyword": "ایران" }
      },
      {
       "match_phrase": {
         "keywords.keyword": "افغانستان" }
      },
      {
       "match_phrase": {
         "keywords.keyword": "محاکمه" }
       } ]
   }
 }
}
```

با انتخاب metric به عنوان نوع نمودار، تعداد این توئیت‌ها نشان داده می‌شود.

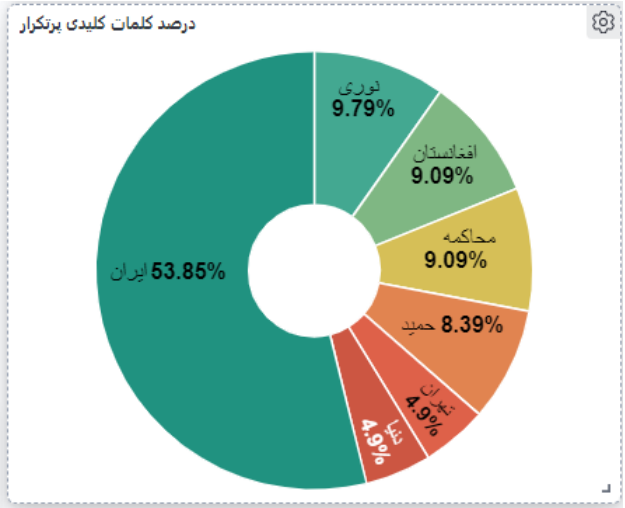
۴-۵-۴- ده هشتگ پرتکرار(اسدی)

برای مشاهده ده هشتگ پرتکرار در یک بازه زمانی مشخص به صورت زیر عمل می‌شود.
مراحل انتخاب Visualization و بازه زمانی مانند قبل انجام می‌شود اما این بار فیلد hashtag.keyword مورد بررسی قرار می‌گیرد. با توجه به خواسته‌ی مسئله از نمودار ستونی برای نمایش تعداد تکرار استفاده می‌شود. تنظیمات مربوط به تعداد ستون‌ها، تابع مدنظر (برای این پرسش از count استفاده شده است)، عناوین، رنگ و غیره در تنظیمات این صفحه قابل تعیین است.

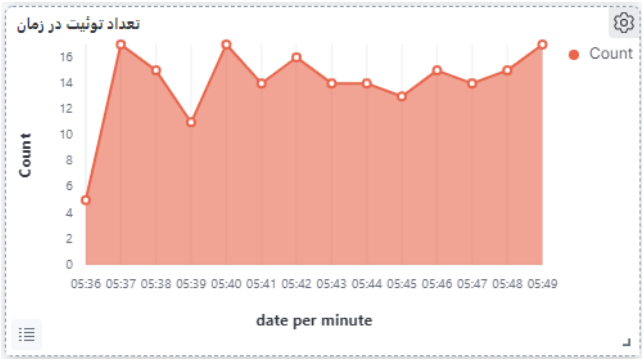


۵-۵-۴- نمودار انتخابی(اسدی)

در پاسخ به پرسش آخر یعنی رسم نمودار انتخابی، دو نمودار توسط ما انتخاب شد. نمودار اول درصد تکرار پراستفاده‌ترین کلمات کلیدی در کل توئیت‌های دریافت شده را نشان می‌دهد. برای این کار از تابع percentage و فیلد keywords.keyword استفاده می‌شود.



نمودار دوم تعداد توئیت دریافت شده بر حسب زمان را نشان می‌دهد. در این نمودار بردار x نشان دهنده زمان است و فاصله زمانی را می‌توان برحسب ثانیه، دقیقه، ساعت و یا غیره انتخاب کرد. در بردار y نیز از تابع count استفاده شده است تا تعداد توئیت‌های دریافتی را نشان دهد.



با افزودن نمودارهای بدست آمده در هر مرحله، به داشبور اولیه، در نهایت نمایش کلی داشبورد به صورت زیر خواهد بود.

۶-۵-۴- پرس‌وجو(اسدی)

در منوی Dev tools در کیابانا امکان اعمال کوئری به داده‌های ایندکس شده در الستیک وجود دارد.

در پرسش اول تمام توئیت‌های یک هشتگ مشخص، خواسته شده است که برای پاسخ به آن از کوئری زیر که یک Boolean query است، می‌توان استفاده کرد. با استفاده از must می‌توان دو شرط را همزمان اعمال کرد.

```
GET /data_center/_search
{
  "query":{
    "bool": {
      "must": [ {
          "term":{
            "hashtags.keyword": {
              "value": "کرونا" } }
        },
        {
          "range": {
            "pdate": {
              "gte": 12,
               "lte": 15 }
          }
        }]}
 }
```

در این کوئری هشتگ «کرونا» و بازه زمانی دوازده تا پانزده انتخاب شده است.

یکی از توئیت‌های بازیابی شده با استفاده از کوئری در این شکل نشان داده شده است.

در پرسش دوم تعداد توئیت‌های دریافتی به ازای یک هشتگ مشخص، خواسته شده است. در کوئری قبلی اگر به جای search_ از count_ استفاده شود. تعداد توئیت‌ها برگردانده می‌شود.

```
GET /data_center/_count
{
  "query":{
    "bool": {
      "must": [ {
          "term":{
            "hashtags.keyword": {
              "value": "کرونا" } }
        },
        {
          "range": {
            "pdate": {
              "gte": 12,
               "lte": 15 }
          }
        }]}
 }
```

خروجی به صورت زیر است.

۵- تاریخچه کانال / هشتگ(اسدی)

این بخش به ذخیره داده‌ها و دریافت پاسخ پرسش‌های مربوط به داده‌ها، با استفاده از کاساندرا می‌پردازد.

۵-۱- راه‌اندازی اولیه(اسدی)

پس از نصب کاساندرا در سیستم و یا افزودن image آن به داکر با استفاده از کتابخانه Cassandra در پایتون می‌توان به آن دسترسی داشت (نصب این کتابخانه به Cassandra-driver نیاز دارد). اولین قدم برای استفاده از کاساندرا تعریف KeySpace است که مفهومی مانند Database در پایگاه‌داده‌های رابطه‌ای دارد.

```
cluster= Cluster(['localhost'],port=9042)
    session= cluster.connect("twitt")
    session.execute("CREATE KEYSPACE IF NOT EXISTS twitter
                    WITH REPLICATION ={ 'class' : 'SimpleStrategy',
                    'replication_factor' : 1 }")
```

برای اجرا بر روی سیستم لوکال بهتر است تعداد تکرار داده، عدد یک انتخاب شود. برای مقداردهی به class می‌توان از SimpleStrategy و یا NetworkTopologyStrategy استفاده کرد که محل ذخیره داده‌ها را تعیین می‌کند. در استفاده لوکال SimpleStrategy کفایت می‌کند.

کاساندرا دارای مفاهیم خاصی مانند primary key است و ویژگی‌های خاصی مانند محدودیت اعمال شرط where تنها بر روی primary key و عدم امکان مرتب سازی را دارد. به همین دلایل در ساخت جداول و تعیین partition key و

clustering key باید به کوئری‌ها توجه کرد و متناسب با آن‌ها جداول را طراحی کرد.
همچنین در پروژه، بیان شده است که هدف تمامی کوئری‌ها دریافت شناسه توئیت‌ها است زیرا با ارسال این شناسه‌ها به الستیک سرچ می‌توان به توئیت‌ها دسترسی داشت.
در این پروژه از سه جدول استفاده شده است. جدول اول با نام twitts برای ذخیره تمام توئیت‌ها است و کلید پارتیشن آن، (روز، ساعت) انتخاب شده است.

```
session.execute("CREATE TABLE IF NOT EXISTS twitter.twitts
                (day int, hour int,minute int, twitt_id text,
                PRIMARY KEY ((day, hour), minute, twitt_id))")
```

جدول دوم با نام persons برای ذخیره توئیت‌های یک شخص است. در این جدول اسم کاربر، کلید پارتیشن است. یعنی برای هر کاربر یک سطر وجود دارد.

```
session.execute("CREATE TABLE IF NOT EXISTS twitter.persons
                (user_name text, day int, hour int, twitt_id text,
                PRIMARY KEY(user_name, day, hour, text_id))")
```

جدول سوم با نام hashtags برای ذخیره توئیت‌های هر هشتگ هستند. کلید پارتیشن در این جدول هشتگ است.

```
session.execute("CREATE TABLE IF NOT EXISTS twitter.hashtags
                (hashtag text, tdate date , twitt_id text,
                PRIMARY KEY (hashtag, tdate , twitt_id))")
```

۲-۵- ذخیره توئیت‌ها در جدول‌های کاساندرا(اسدی)

سپس باید هر توئیتی که از طریق کانال کافکا دریافت می‌شود به این جدول‌ها اضافه شود.

با توجه به آنکه از روز و ساعت در جدول‌ها استفاده شده است؛ پس از دریافت هر توئیت این مقادیر از رشته‌های مربوط به تاریخ و ساعت همان توئیت استخراج می‌شوند.

```
for msg in consumer:
    data = json.loads(msg.value)
    time1= data["ptime"]
    hour,minute,second = [int(x) for x in time1.split(':')]
    date1 = data["pdate"]
    year,month,day = [int(x) for x in time1.split('-')]
```

هر توئیت دریافت شده، با استفاده از روز و ساعت و دقیقه و شناسه در جدول اول ذخیره می‌شود.

```
add_to_twitts="INSERT INTO twitts (day,hour,minute,twitt_id)
               VALUES (%s,%s,%s)"
session.execute(add_to_twitts, (day,hour,minute,data["id"]))
```

و با استفاده از نام کاربر، روز، ساعت و شناسه توئیت در جدول دوم ذخیره می‌شود.

```
add_to_persons="INSERT INTO persons (user_name,day,hour,twitt_id)
                VALUES (%s, %s, %s, %s)"
session.execute(add_to_persons, (data["user"],day,hour,data["id"]))
```

همچنین با استفاده از هشتگ، تاریخ و شناسه توئیت، در جدول سوم ذخیره می‌شود.

```
add_to_hashtags="INSERT INTO hashtags (hashtag,tdate,twitt_id)\
                 VALUES (%s,%s,%s)"

if data["hashtags"]≠[]:
   for hashtag in data["hashtags"]:
       session.execute(add_to_hashtags,
       (hashtag, data["date"], data["id"]))
```

۳-۵- پرس‌وجو(اسدی)

این بخش به پاسخ پرسش‌های مطرح شده می‌پردازد.

۱-۳-۵- توئیت‌های یک ساعت اخیر(اسدی)

برای بازیابی توئیت‌های یک ساعت اخیر از جدول twitts می‌توان استفاده کرد. با وارد کردن روز و ساعت، شناسه توئیت‌های یک ساعت اخیر نشان داده می‌شوند (البته می‌توانیم دریافت زمان فعلی را نیز به خود برنامه بسپاریم و در کوئری از آن‌ها استفاده کنیم).

```
session.execute("SELECT twitt_id FROM twitts where
                day=15 AND hour⩾20")
```

۲-۳-۵- توئیت‌های یک شخص در ۲۴ساعت اخیر(اسدی)

به این دلیل که ما به دریافت توئیت‌های فارسی پرداخته‌ایم و کانال نداریم، پرسش دوم که مربوط به «بازیابی پست‌های یک کانال در ۲۴ساعت اخیر» بود را به «بازیابی توئیت‌های یک شخص در ۲۴ساعت اخیر» تغییر داده‌ایم. با کوئری زیر با استفاده از جدول persons با تعیین نام کاربر و روز و ساعت، پست‌های ۲۴ساعت اخیر شخص بازیابی می‌شوند.

```
session.execute("SELECT twitt_id FROM persons where
                user_name='Rahmadzade' AND
                (day=15 OR (day=14 AND hour>21)")
```

۳-۳-۵- پست‌های یک هشتگ(اسدی)

برای بازیابی توئیت‌های مرتبط با یک هشتگ در بازه زمانی مشخص از جدول سوم استفاده می‌شود. در این کوئری با وارد کردن هشتگ و بازه زمانی مدنظر پست‌ها بازیابی می‌شوند.

```
session.execute("SELECT twitt_id FROM hashtags where
                hashtag='help' AND
                date>'2021-8-13' AND date<'2021-8-15'")
```

۴-۳-۵- توئیت‌های یک شخص در یک هفته(اسدی)

پرسش آخر را نیز با توجه به استفاده از توئیتر، به صورت «بازیابی پست‌های یک شخص در یک هفته گذشته» تغییر داده‌ایم.

در این کوئری از جدول persons استفاده می‌شود و نام کاربر و بازه زمانی روزهای مدنظر در آن تعیین می‌شوند. برای بدست آوردن تعداد این توئیت‌ها از count(*) و همچنین برای بدست آوردن تعداد توئیت‌های روزانه از GROUP BY day می‌توان استفاده کرد.

```
session.execute("SELECT twitt_id, count(*) FROM person where
                user_name='Rahmadzade' day IN (8,15)")
```

۶- ساخت یک مدل پیشبینی کننده با اسپارک(عباسپور)

این بخش خواندن داده‌ها از کاساندرا و ساخت مدل پیش بینی کننده با استفاده از اسپارک می‌پردازد.

۱-۶- راه اندازی اولیه(عباسپور)

پس از نصب کاساندرا در سیستم و یا افزودن image آن به داکر با استفاده از کتابخانه Cassandra در پایتون می‌توان به آن دسترسی داشت. همتنطور که در بخش قبل نیز گفته شد اولین قدم برای استفاده از کاساندرا تعریف KeySpace است.

```
from cassandra.cluster import Cluster

cluster = cluster = Cluster(['cassandra'], port=9042)
session = cluster.connect()
session.set_keyspace('twitt')
```

در ادامه می بایست از کتابخانه PySpark استفاده شود تا بتوانیم مدل خواسته شده را ایجاد کنیم.

```
from pyspark import SparkContext, SparkConf, SQLContext
import pyspark.sql.functions as F

sc = SparkContext.getOrCreate()
sqlContext = SQLContext(sc)
```

۶-۲- خواندن اطلاعات از کاساندرا(عباسپور)

جدول Cassandra استخراج شده برای هر دو گام، Hashtag است که در زیر بخشی از آن مشاهده میشود. این جدول در ابتدا تبدیل به یک دیتافریم اسپارک میشود.

```
rows = session.execute('select * from hashtags')

rdd = sc.parallelize(list(rows))
df = sqlContext.createDataFrame(rdd)
df.registerTempTable('hashtags')
df.toPandas()
```

۶-۳- پیشبینی زمان ارسال پست بعدی(عباسپور)

در این قسمت از پروژه به جای کانال، از هشتگ استفاده میشود و به جای روز هفته و ساعت، از ساعت و دقیقه و ثانیه استفاده میشود. این تطبیق به دلیل کمبود داده و نمونه های گرفته شده از توییتر انجام میشود.

ابتدا معروفترین هشتگ‌های ذخیره شده در اسپارک را Queryکردیم و داده های پرتکرارترین هشتگ را طبق برای انجام پیشبینی زمان بعدی پست نگه داشتیم.

```
sqlContext.sql("""select count(hashtag) as recurrence, hashtag
                from hashtags
                group by hashtag
                order by recurrence desc""").toPandas()
```

در ادامه برای پرتکرار ترین هشتگ، تمامی توییت های مربوط به همراه زمان آن ها را کوئری کردیم.

```
df2 = sqlContext.sql("""select ROW_NUMBER() over
(ORDER BY id) AS id, hour, minute, second
                from hashtags
                where hashtag='ایران'""")

df2.registerTempTable('hashtag_time')

df2.toPandas()
```

برای فرموله کردن مساله، برای هر پست، زمان روز به ثانیه را در یک ستون جدید به نام timestamp و اختلاف زمانی به ثانیه تا پست بعدی را نیز در ستون جدید next ذخیره کردیم.

```
df3 = sqlContext.sql("""select t1.id, t1.hour, t1.minute,
t1.second (t1.hour*3600+t1.minute*60+t1.second) as timestamp,
(t2.hour-t1.hour)*3600+(t2.minute-t1.minute)*60+(t2.second-
t1.second) as next
                from hashtag_time as t1 join hashtag_time as
t2
                where t2.id = t1.id+1""")

df3.registerTempTable('hashtag_timestamp')
df3.toPandas()
```

۶-۴- مدل پیش بینی کننده(عباسپور)

حال میبایست داده های خوانده شده را برای مدل آماده کنیم. در واقع دیتای timeStamp را تحت عنوان features به مدل می دهیم تا دیتای next را برای ما پیش بینی کند.

```
from pyspark.ml.feature import VectorAssembler

vectorAssembler = VectorAssembler(inputCols = ['timestamp'],
outputCol = 'features')
model_df = vectorAssembler.transform(df3)
model_df = model_df.select(['features', 'next'])
model_df.toPandas()
```

۶-۴-۱- آموزش مدل(عباسپور)

به صورت تصادفی ۸۰ درصد داده های در دسترس را برای آموزش مدل و ۲۰ درصد آنها را برای ارزیابی مدل کنار میگذاریم.

```
splits = model_df.randomSplit([0.8, 0.2])
train_df = splits[0]
test_df = splits[1]
```

برای پیشبینی از مدل GradientBoostedTree(GBT) regression modelبا تنظیمات پیش فرض استفاده کردیم.
و با استفاده از داده هعای آموزش مدل را آموزش دادیم و بر روی داده های تست آن را اجرا گرفتیم.

```
from pyspark.ml.regression import GBTRegressor
from pyspark.sql.functions import round

gbt = GBTRegressor(featuresCol = 'features', labelCol='next',
maxIter=10)
gbt_model  = gbt.fit(train_df)

gbt_predictions = gbt_model.transform(test_df)

gbt_predictions = gbt_predictions.select('prediction',
round(F.col('prediction'), 0).alias('pred_round'), 'next',
'features')
gbt_predictions.toPandas()
```

۲-۴-۶- ارزیابی مدل(عباسپور)

مدل آموزش داده شده را برای داده های تست اجرا گرفتیم و نتیجه را ارزیابی کردیم.

```
from pyspark.ml.evaluation import RegressionEvaluator

gbt_evaluator = RegressionEvaluator(
    labelCol="next", predictionCol="prediction",
metricName="rmse")
rmse = gbt_evaluator.evaluate(gbt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" %
rmse)
```

۷- آمار(سرتیپی)

در این بخش اطاعات دریافت شده از مراحل قبل را درون ردیس ذخیره می‌کنیم تا برای واکشی اطلاعات آماری از آن استفاده بشود.

۱-۷- ذخیره اطلاعات در REDIS (سرتیپی)

از این پایگاه داده به عنوان ذخیره‌سازی در RAM استفاده می‌شود و در صورتی که نیاز به عملیات خیلی خاص نداشته باشد می‌توان از پایگاه استفاده کرد. برای ذخیره‌ها سازی داده چون ماهیت داده‌ها به صورت توئیت است و کانال نداریم کلمات کلیدی را به عنوان کانال در نظر می‌گیریم و انتشار توئیتی در آن کلمه‌ی کلیدی گزارش می‌شود. برای بحث زمانبندی برای توئیت‌ها که چه زمانی حذف شوند و دیگر در RAM نگه داشته نشوند را می‌توان با استفاده ویژگی expire در REDIS مدیریت کرد و با تعیین زمان برای آن توئیت‌ها بعد از مدت مشخص حذف خواهند شد.
در REDIS از ویژگی ltrim برای مشخص کردن نگاه داشتن تعدادی خاص (در یک رنج دلخواه) استفاده می‌شود. برای مثال ۱۰۰۰ توئیت اخیر.
برای ذخیره‌ی کلی داده‌ها توابعی نوشته شده است که پس از دریافت هریک از توئیت‌ها در کانال آمار کافکا، توابعی برای ذخیره‌سازی موارد مورد نیز صدا زده می‌شود. در نهایت نیز داده‌ها به کانال exit رفته و از مدار کافکا خارج می‌شود. شبه کد این تکه کد در زیر آمده است.

```
producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda v: json.dumps(v).encode('utf-
8'))umer = KafkaConsumer('statistics')

redis = get_redis_connection()
while consumer:
  consumer = KafkaConsumer('statistics')
  for msg in consumer:
      tweet = json.loads(msg.value)
      store_tweet_by_keyword(tweet)
      store_tweet_by_time(tweet)
      store_last_hour_hashtags(tweet)
      store_last_hashtags(tweet)
      store_last_tweets(tweet)
      print("stored tweet with id " + str(tweet["id"]))
      producer.send('exit', value=data)
```

۱-۱-۷- ذخیره به اعضای هر کلمه‌ی کلیدی(سرتیپی)

متدی که در شبه کد زیر وجود دارد با دریافت هر توئیت و کلمه‌ی کلیدی آنها، در ابتدا کلمه‌ی کلیدی و سپس شناسه‌ی فرد را در ادامه‌ی آن قرار می‌دهد و در REDIS ذخیره می‌کند. در این صورت ما تمامی توئیت‌ها و همچنین کلمات کلیدی که در آن توئیتی ارسال شده است را خواهیم

داشت. همچنین زمان expire شدن را نیز برای توئیت‌های ذخیره شده در نظر می‌گیریم. این زمان برابر با ۶ ساعت خواهد بود.

```
def store_tweet_by_keyword(tweet):
    keywords = tweet["keywords"]
    tweet_id = tweet["id"]
    for keyword in keywords:
        r.set(f"keyword:{keyword}:{tweet_id}", dumps(tweet), ex=60 * 60 * 6)
```

۷-۱-۲- ذخیره‌ی توئیت‌ها بر اساس زمان(سرتیبی)

تابع زیر برای ذخیره‌سازی توئیت‌ها بر اساس زمان ایجاد شدن آن استفاده می‌شود.

```
def store_tweet_by_time(tweet):
    time_key = datetime.strftime(datetime.strptime(
        tweet["created_at"], '%a %b %d %H:%M:%S +0000 %Y'),
'%Y:%m]%ds%(f)tweets:{time_key}", dumps(tweet))
```

۷-۱-۳- ذخیره‌ی هشتگ‌های یک ساعت اخیر(سرتیبی)

از تابع زیر برای ذخیره‌ی هشتگ‌های یک ساعت اخیر استفاده شده است. هر توئیتی که دریافت می‌شود برای تک تک هشتگ‌هایی که در این توئیت قرار دارد، سطری ساخته شده و زمان انقضای آن نیز یک ساعت تعیین می‌شود.

```
def store_last_hour_hashtags(tweet):
    hashtags = tweet["hashtags"]
    for hashtag in hashtags:
        r.set(f"last_hour_hashtags:{hashtag}", hashtag,
nx=True)r.expire(f"last_hour_hashtags:{hashtag}", 60 * 60)
```

۷-۱-۴- ذخیره‌ی ۱۰۰۰ هشتگ اخیر(سرتیبی)

از تابع زیر برای ذخیره‌ی ۱۰۰۰ هشتگ اخیر استفاده می‌شود و به این صورت است که یک بازه‌ی ۱۰۰۰تایی برای آن‌ها در حافظه در نظر گرفته می‌شود.

```
def store_last_hashtags(tweet):
    hashtags = tweet["hashtags"]
    for hashtag in hashtags:
        r.lpush("last_hashtags", hashtag)
        r.ltrim("last_hashtags", 0, 999)
```

۷-۱-۵- ذخیره‌ی ۱۰۰ توئیت اخیر(سرتیبی)

از تکه کد زیر نیز برای ذخیره ۱۰۰ توئیت اخیر در REDIS استفاده شده است تا در مراحل دیگر برای نمایش به کاربر استفاده بشود.

```
def store_last_tweets(tweet):
    r.lpush("last_tweets", dumps(tweet))
    r.ltrim("last_tweets", 0, 99)
```

۷-۲- نمایش داده‌های ذخیره شده در REDIS(سرتیبی)

در این بخش با استفاده از flask و صفحات HTML داده‌هایی که در REDIS ذخیره شده‌اند را به کاربر نمایش می‌دهیم.

۷-۲-۱- بک‌اند(سرتیبی)

بخش بک‌اند پروژه که داده‌ها را از REDIS خوانده و به کاربر نمایش می‌دهد در فایل app.py قرار دارد. با استفاده از اجرای این فایل با دستور flask run می‌توان داده‌ها در URL هایی که برای توابع و کوئری‌ها تعریف شده‌اند اجرا

کرد و به سمت فرانت‌اند ارسال کرد. خروجی‌ها در آدرس زیر در دسترس می‌باشند.

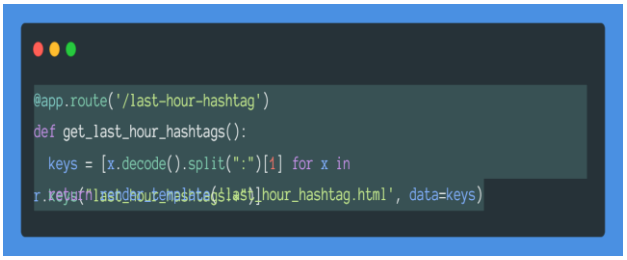
از طریق آدرس زیر، ۱۰۰ توئیت اخیر ذخیره شده در REDIS را می‌توان بازیابی کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

http://localhost:5000/last-tweets-100

```
@app.route('/last-tweets-100')
def get_last_tweets():
    data = [loads(x.decode()) for x in r.lrange("last_tweets", 0,
-1)]return render_template('last_100_tweet.html', data=data)
```

از طریق آدرس زیر، می‌توان هشتگ‌های یک ساعت اخیر که در ردیس ذخیره شده‌اند را مشاهده کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

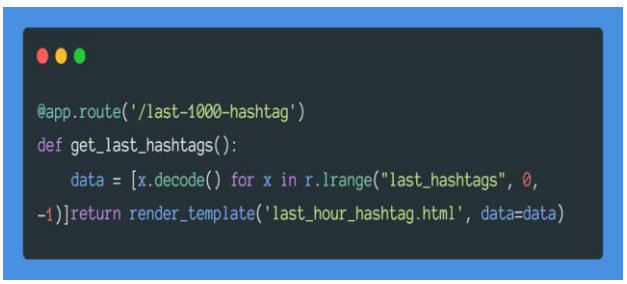
http://localhost:5000/last-hour-hashtag



```
@app.route('/last-hour-hashtag')
def get_last_hour_hashtags():
  keys = [x.decode().split(":")[1] for x in
r.keys("last_hour_hashtag:*")]return render_template('last_hour_hashtag.html', data=keys)
```

از طریق آدرس زیر، می‌توان ۱۰۰۰ هشتگ اخیر که در ردیس ذخیره شده‌اند را مشاهده کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

http://localhost:5000/last-1000-hashtag



```
@app.route('/last-1000-hashtag')
def get_last_hashtags():
    data = [x.decode() for x in r.lrange("last_hashtags", 0,
-1)]return render_template('last_hour_hashtag.html', data=data)
```

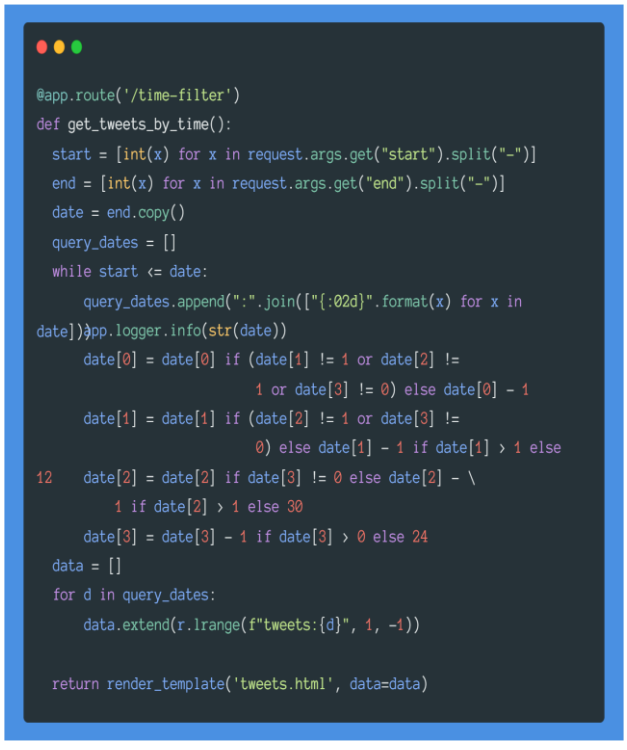
از طریق آدرس زیر، می‌توان تعداد هشتگ‌های ۶ ساعت اخیر را مشاهده کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

http://localhost:5000/keywords

```
@app.route('/keywords')
def last_6_hour_keywords():
  keywords = [x.decode().split(":")[1] for x in r.keys("keyword:*")]
  keywords_counts = {}
  for k in keywords:
      keywords_counts[k] = keywords.count(k)
  return render_template('last_6_hour_keywords.html',
data=keywords_counts)
```

از طریق آدرس زیر، با ارسال یک بازه‌ی زمانی توئیت‌های که در آن بازه ارسال شده‌اند را دریافت کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

http://127.0.0.1:5000/time-filter/?start=2021-8-11-00&end=2021-8-13-00



```
@app.route('/time-filter')
def get_tweets_by_time():
  start = [int(x) for x in request.args.get("start").split("-")]
  end = [int(x) for x in request.args.get("end").split("-")]
  date = end.copy()
  query_dates = []
  while start <= date:
      query_dates.append(":".join(["{:02d}".format(x) for x in
date]))app.logger.info(str(date))
      date[0] = date[0] if (date[1] != 1 or date[2] !=
                            1 or date[3] != 0) else date[0] - 1
      date[1] = date[1] if (date[2] != 1 or date[3] !=
                            0) else date[1] - 1 if date[1] > 1 else
12    date[2] = date[2] if date[3] != 0 else date[2] - \
          1 if date[2] > 1 else 30
      date[3] = date[3] - 1 if date[3] > 0 else 24
  data = []
  for d in query_dates:
      data.extend(r.lrange(f"tweets:{d}", 1, -1))

  return render_template('tweets.html', data=data)
```

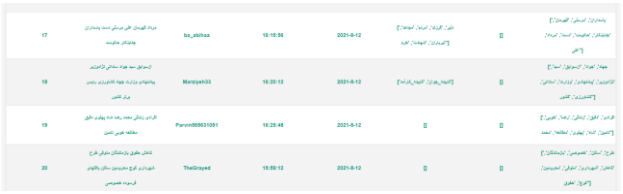
۲-۲-۷- فرانت اند(سرتیپی)

برای نمایش کوئری‌هایی که بر بروی پایگاه داده‌ی REDIS زده شده است، در فایل templates تعدادی فایل HTML وجود دارد که اطلاعات را از API ها دریافت کرده و نمایش می‌دهد. در زیر صفحات نمایش اطلاعات قابل مشاهده است. که با رفرش کردن آن اطلاعات بیشتری را می‌توان به کاربر نمایش داد. در تصویر زیر لیست API های درخواست شده در سند پروژه قابل مشاهده است.

index	API Name	Address
1	time-filter	link to page
2	last_6_hour_keywords	link to page
3	get_last_hashtags	link to page
4	last-hour-hashtag	link to page
5	get_last_tweets	link to page

۳-۲-۷- دریافت ۱۰۰ توئیت اخیر(سرتیپی)

نمونه‌ای از ۱۰۰توئیت اخیر که در صفحه‌ی وب نمایش داده شده است.



۴-۲-۷- هشتگ‌های یک ساعت اخیر(سرتیپی)

در شکل زیر آخرین هشتگ‌های ۱ ساعت اخیر قابل مشاهده است.

index	hashtag
1	فرزند
2	Iran
3	مجاهد
4	زندانیان_قتل_عام
5	جنبش_دادخواهی
6	مردم
7	حمید_نوری
8	کابینه_کارآمد
9	خرم
10	شهادت
11	تیرباران
12	قتل_عام۶۷
13	کابینه_جوان

۵-۲-۷- توئیت‌های یک‌بازه‌ی زمانی(سرتیپی)

همانطور که در شکل زیر مشاهده می‌شود اطلاعات مربوط توئیت‌ها در یک بازه‌ی دلخواه که پیشتر URL آن معرفی شد، به شکل زیر می‌باشد.

۶-۲-۷- کلمات کلیدی و تعداد تکرار آن‌ها(سرتیپی)

در شکل زیر کلمات کلیدی اخیر و تعداد تکرار آن‌ها قابل مشاهده است.

index	kewords	count
1	علوم	1
2	سلطنت	1
3	ساداتی	1
4	پهلوی	4
5	حقوق	1
6	شدندازشنیدن	1
7	حکومت	1
8	بری	1
9	سید	1
10	کشتار	1
11	پاسداران	5
12	مطالعه	1
13	جواد	1
14	قدرت	1
15	امکان	3

۷-۲-۷- دریافت هزار هشتگ اخیر(سرتیپی)

همانطور که در شکل زیر مشاهده می‌شود این صفحه لیست هزار هشتگ اخیر را به ما باز می‌گرداند.

index	hashtag
1	رضا_پهلوی
2	پیمان_نوین
3	رضا_پهلوی
4	جاویدشاه
5	رضا_پهلوی
6	پیمان_نوین
7	رضا_پهلوی
8	Iran
9	قتل_عام۶۷
10	حمید_نوری
11	جنبش_دادخواهی
12	حمید_نوری
13	زندانیان_قتل_عام

مراجع

پانویس ها

[۱] API Key
[۲] API Secret Key
[۳] Bearer Token
[۴] Access Token
[۵] Access Token Secret