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،
amirsartipi13@gmail.com

[۳] دانشجوی مقطع ارشد، خوشه کلان داده، دانشگاه اصفهان ، اصفهان،
amirsartipi13@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 = 'Eo3ej5ML2UbkkJUEcF781EyOA'
API_Secret_Key = 'QtED20LEAhrgWk1dBPP7QCy1LO0As24y5Kp2K1gedJPxTqm0II'

Bearer_Token =
'AAAAAAAAAAAAAAAAAAAAAB1USAEAAAAAEfCgv2RY0waIj7mCpnNOmIT70cM%3D99Yx4oCNnX8Ky2oUTeuqvANZR
f5YwlQIO5zH7zWD5dRC5Zsdvs'
Access_Token = '1419011209307639810-ni6S77Lvjat1QpRftPNI2YcBjleiJvV'
Access_Token_Secret = 'tyTVLcojbETSeq0NFfoA8UHP8Qdf50WoDYpnEgIU5m2S2'
```

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.

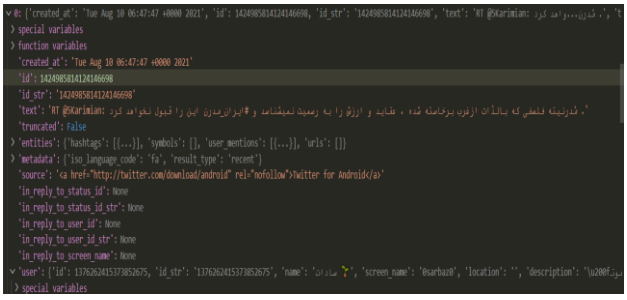
```
def twitter_crawller(count):
    data = []
    try:
        tso = TwitterSearchOrder()
        # Set keywords which we want to crawl
        tso.set_keywords(keywords)
        # Set language
        tso.set_language('fa')
        tso.set_include_entities(True)
        tso.set_count(count)

        # 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(سرتیتی)

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(سرتیتی)

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

۳-۱- ایست واژه‌ها و افعال(سرتیتی)

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(سرتیتی)

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(سرتیتی)

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.

```
def split_date_time(date):
    date = date.split(' ')
    year = date[-1]
    day = date[2]
    month = datetime.datetime.strptime(date[1], "%b").month
    time = date[3]
    date = str(year) + '-' + str(month) + '-' + str(day)
    date_time = date + ' ' + time
    timestamp = datetime.datetime.timestamp(datetime.datetime.strptime(date_time, "%Y-
%m-%d %H:%M:%S"))
    return date, time, timestamp
```

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)

```
hashtags = list(set(re.findall(r"#(\w+)", text)))
```

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

```
urls = list(set(re.findall("(?P<url>https?://[^\s]+)",
text)))
```

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.

```
text = normalizer.normalize(text)
text = word_tokenize(text)
text = [word for word in text if word not in stop_words and
keywords = [word for word in text if word in static_keywords]
```

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.

```
def find_keywords(message, kw):
    kw_extractor = yake.KeywordExtractor()
    custom_kw_extractor = yake.KeywordExtractor(n=1, features=None,
top=10)
    keywords = custom_kw_extractor.extract_keywords(message)
    keywords = [x[0] for x in keywords if x[1] > 0.09]  + kw
    return list(set(keywords))
```

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.

```
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(سرتیتی)

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.

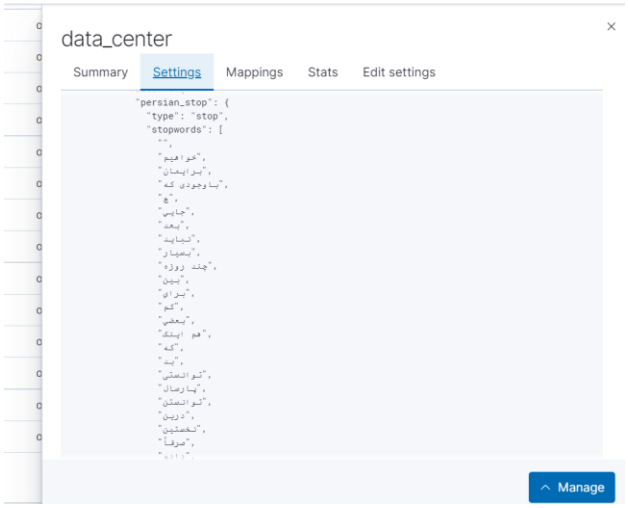
```
def write_json(new_data, filename='./files/data.json'):
    with open(filename,'r') as file:
        file_data = json.load(file)
        file_data["data"].append(new_data)
        file.seek(0)
        json.dump(file_data, file, indent = 4)
```

۴- ماندگاری(سرتیتی)

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

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

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(سرتیتی)

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
```

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)
```

۵- تاریخچه کانال / هشتگ

۶- ساخت یک مدل پیشبینی کننده با اسپارک

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.

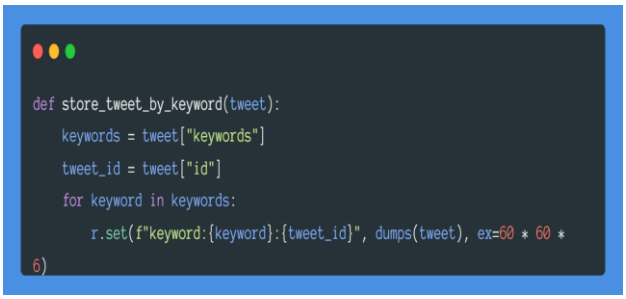
۷-۱- ذخیره اطلاعات در REDIS (سرتیپی)

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.

۷-۱-۱- 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.

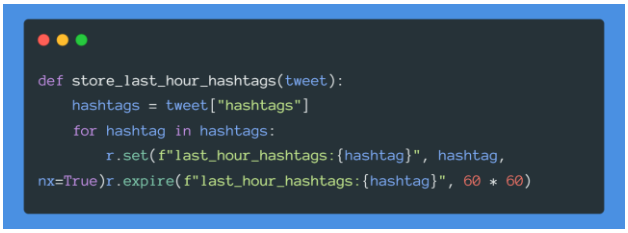


۷-۱-۲- ذخیره‌ی توئیت‌ها بر اساس زمان(سرتیپی)

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

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

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.



۷-۱-۴- ذخیره‌ی ۱۰۰۰ هشتگ اخیر(سرتیپی)

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(سرتیپی)

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(سرتیپی)

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(سرتیپی)

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_hashtags:*")]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)
```

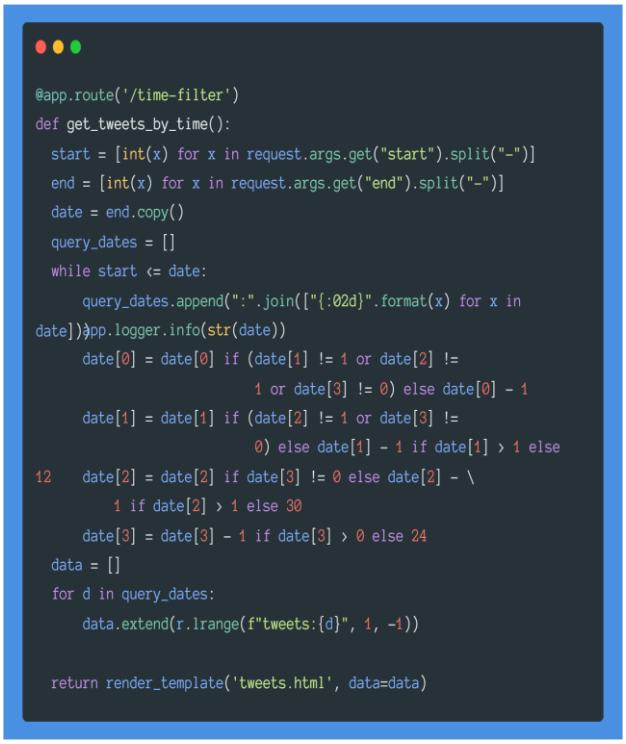
از طریق آدرس زیر، می‌توان تعداد هشتگ‌های ۶ ساعت اخیر را مشاهده کرد. که تکه کد نوشته شده برای آن در زیر مشاهده می‌شود.

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)
```

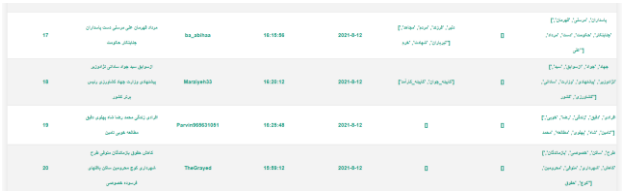
۲-۷-۲- فرانت اند(سرتیپی)

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

۳-۷-۲- دریافت ۱۰۰ توئیت اخیر(سرتیپی)

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



۴-۷-۲- هشتگ‌های یک ساعت اخیر(سرتیپی)

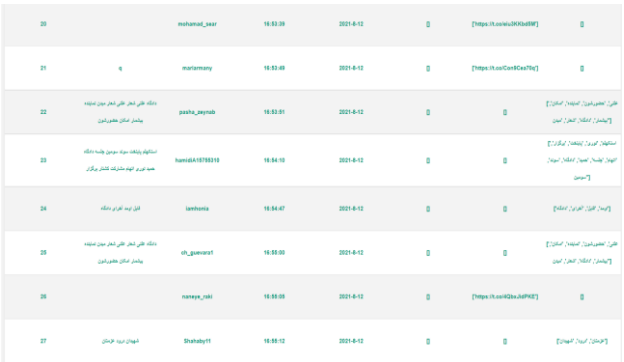
در شکل زیر آخرین هشتگ‌های ۱ ساعت اخیر قابل مشاهده است.

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

index	kewords	count
1	علوم	1
2	سلطنت	1
3	ساداتی	1
4	پهلوی	4
5	حقوق	1
6	شدنداز شنیدن	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

۷-۲-۵- 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.



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

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

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

[1] API Key
[2] API Secret Key
[3] Bearer Token
[4] Access Token
[5] Access Token Secret