Identifying Trending Twitter Hashtags Using Apache Spark

Runyu Hao(rh2962), Xinluan Tian(xt2233), Lixin Zhang(lz2684), Zhili Huang(zh2397)

Abstract

In this project, we use Apache Spark to get tweets from Twitter API and do data visualization on the top 10 trending hashtags. What's more, we use NLP related techniques to realize Named Entity Recognition on the tweets with top trending hashtags. All the results are shown on a platform that is constructed by using Python, JavaScript, and Flask framework.

● Problem Definition

With the rise of social media, the web has become a vibrant and lively Social Media realm in which billions of individuals all around the globe interact, share, post, and conduct numerous daily activities. Collecting information on social media gives us a chance to get a huge amount of raw, on time, and comprehensive information. We can get real-time and comprehensive information on social media. However, the overwhelming data is hard to process.

Collecting information on social media gives us a chance to get a huge amount of raw, on time, and comprehensive information. The combination of social media and big data has created a new area of research, namely social media mining, which is similar to data mining but limited to the worlds of Twitter, Facebook, Instagram, etc. Social media mining is "the process of representing, analyzing, and extracting operational patterns from social media data."[1] Therefore, we use Apache Spark and Python to track the most frequently mentioned hashtags.

● Challenges faced

1. Connection in NER progress

   Firstly, we did some NLP related work to preprocess the obtained tweets. Here we used regular expressions to finish data cleaning. The specific procedures of data cleanings are tokenization, lowercases, removing URLs, removing punctuations, lemmatization, and normalization.

   Here we use displaCy.js to realize NER visualization on the web page. In order to do NER tasks at the backend server, we attempted many methods to send data back to the server. Finally, we established a HTTP "POST" request through an AJAX request, it can send our data(tweets) back to the server side. In the success callback function of AJAX, we push that processed data into an array, then re-render it on the page with some well-designed styling.

```
src_Sentence.forEach((s)=>{
    $.ajax({
        type:"POST",
        url: '/ent',
        data: JSON.stringify({'text':s, 'model':'en_core_web_sm'}),
        contentType: "application/json",
        dataType: 'json',
        success: function(data) {
            var input = {text:s, data:data};
            console.log(input);
            retrieved_data.push(input);
        }
    });
});
```

AJAX code

2. NER tasks

At the server side, what we utilized for NER tasks is a Python package named “Spacy”. It can help us recognize and then extract predefined entities, such as Person, Organization, etc. It will send back some jsonfied data back to the frontend. With the above frontend JavaScript code, we can then see the NER results.

```
@app.route("/ent", methods=['POST'])
def entitry():
    content = request.json
    text = content['text']
    model = content['model']
    nlp = MODELS[model]
    doc = nlp(text)
    res = [
        {"start": ent.start_char, "end": ent.end_char, "type": ent.label_}
        for ent in doc.ents
    ]
    return jsonify(res)
```

- **Technical Details**

1. Get access to Twitter API

Here we use Twitter API to get keys and access tokens, which is used for getting the online stream in the following steps. And then we can get a connection to Twitter.

2. Get tweets

We get all tweets in JSON format and then send them into Spark streaming by TCP connection.

```
lines=http.iter_lines(chunk_size=2048)
for line in lines:
    full_tweet = json.loads(line)
    tweet_text = full_tweet['text']
    tcp_connection.send((tweet_text + '\n').encode("utf-8"))
```

To capture hashtags, we first split tweets by blank space, and then we filter and get the phrases with the head of ‘#’.

```
words = dataStream.flatMap(lambda line: line.split(" "))
words.pprint()
# filter hashtag mark
hashtags = words.filter(lambda w: '#' in w).map(lambda x: (x, 1))
```

3. Checkpoint mechanism

In order to get updated values, we need to accumulate the count of each hashtag together. Since the online Twitter streaming is too large to fit in memory simultaneously, we need to set a checkpoint to allow RDD recovery for relatively reliable data persistence. Firstly, we make a checkpoint to do the transformation on streaming in every two seconds.

By using Spark SQL Context, we convert RDD into temp table and use select command to get top 10 trending hashtags.

```
def get_sql_context_instance(spark_context):
    if ('sqlContextSingletonInstance' not in globals()):
        globals()['sqlContextSingletonInstance'] = SQLContext(spark_context)
    return globals()['sqlContextSingletonInstance']

hashtag_counts_df = sql_context.sql(
    "select hashtag, hashtag_count from hashtags order by hashtag_count desc limit 10")
hashtag_counts_df.show()
send_df_to_dashboard(hashtag_counts_df)
```

4. Application platform

We use Python, JavaScript, and Flask framework to demonstrate the data visualization result. In order to get updated trending hashtags, we use AJAX to refresh the chart information without reloading the whole webpage. Here we show the mentioned counts of the top 10 trending hashtags which are represented in descending order in the chart.

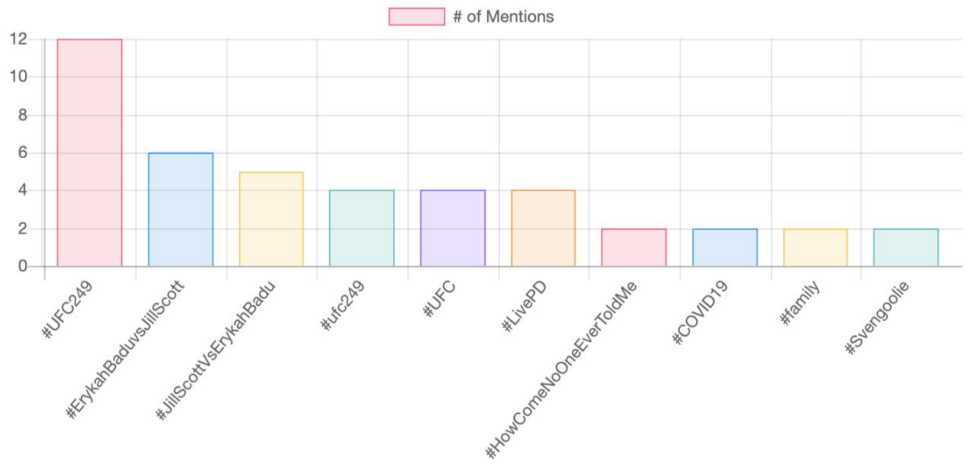

Fig1. Top Trending Twitter Hashtags Chart

As shown in Fig2, those tweets with top trending hashtags are also illustrated on the same webpage, which is also updated every two seconds. Named Entity Recognition is also represented with relation types.

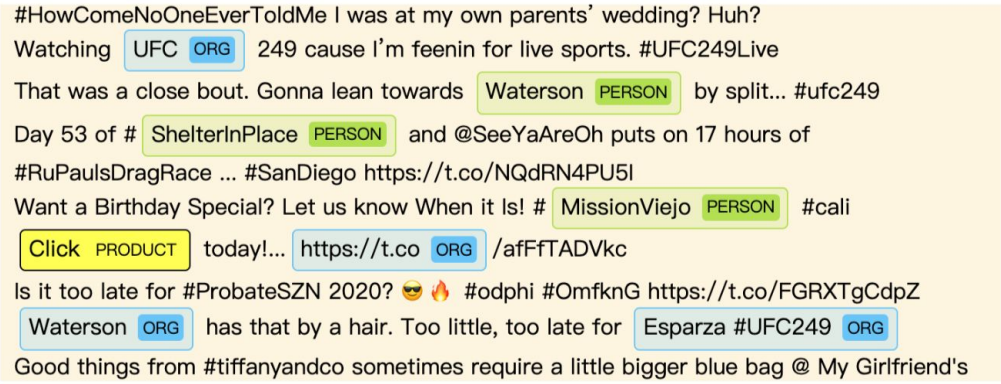

Fig2. Tweets with hot hashtags

- **Streaming Processing**

Here we build up our Spark streaming application that does real-time processing for the incoming tweets, extract the hashtags from them, and calculate how many hashtags have been mentioned.

To begin with, we set each batch interval as two seconds and set the log level as ERROR to avoid showing log information.

```
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
# interval is 2 seconds
ssc = StreamingContext(sc, 2)
```

Secondly, from the scheduled port, we read data which are tweets in Dstream. The input tweets are all split into words RDD and filtered by hashtag mark '#', which is mapped to pairings of (hashtag, 1).

To allow RDD transformation periodically, we use updateStateByKey to update the count for each hashtag. Not only can the checkpoint mechanism ensure the stateful transformations periodically, but it also can allow the persistence of the data.

Due to the huge amount of Twitter users, there are many tweets with different contents but are with the same hashtag. Thus we use reduceByKey to get how many times of each hashtag has been mentioned in the streaming. For the whole processing batches, we accumulate all counts of each hashtag by calling the checkpoint mechanism.

We sum the value for each hashtag across all the batches and do processing on RDD in two seconds' batch. Then RDD are stored in a temp table using Spark SQL Context and then perform a select statement that is used to retrieve the top ten hashtags with their counts.

Finally, we send dataframe to the dashboard application through the REST API. Hashtags are extracted from dataframe. Here we use two arrays to store dataframe, one is for the count of each hashtag and the other is for the title of each hashtag.

- **Evaluation**

In this project, we can easily see the dynamic trending hashtags on our webpage, which is refreshed every two seconds. It shows a concise and clear result of the real-time trending. But in a few experiments, the chart may show a bar named with only '#' which is caused by some users only use hashtag marks '#' to post tweets without any content. Thus, this is another aspect we need to consider.

The NER accuracy is more than 60%. When it encounters a hashtag, it will recognize this hashtag word as a Person. We even saw once it marked the word "Trump" as an object. However, it can precisely judge the organization. The overall performance is not bad.

- **Discussion and Future work**

In this project, we use Named Entity Recognition to identify named entities in tweets such as names of people, places, and organizations in the corpus, and identify entities with specific meanings in the text. What's more, we utilize python and Apache Spark to read online streaming through Twitter API, and then extract top trending hashtags to show users' common interests towards news topics.

To polish our current project, in the future, we can add other NLP techniques such as sentiment analysis to classify tweets into different polarity categories and demonstrate users' attitudes towards different hashtags.

In this day and age, with the characteristics of portability, mobile devices are getting more and more popular. It is important to realize the optimization of the cluster on mobile devices that have limited hardware sources. Therefore, developing mobile apps to identify trending hashtags seems more practical for users, which will be deployed widely in the future.

- **Reference**

[1] R Zafarani, MA Abbasi, H Liu. Social media mining: an introduction
[2] https://spacy.io/usage/processing-pipelines
[3] https://explosion.ai/blog/displacy-js-nlp-visualizer
[4] https://explosion.ai/demos/displacy
[5] https://spark.apache.org/docs/latest/