EECS E6893 HW3

Jing Qian (jq2282)

a. Screenshot of code to do all the tasks.

1. Top hashtags.

```
def aggregate_tags_count(new_values, total_sum):
    return sum(new_values) + (total_sum or 0)

def hashtagCount(words):
    """
    Calculate the accumulated hashtags count sum from the beginning of the stream
    and sort it by descending order of the count.
    Ignore case sensitivity when counting the hashtags:
        "#Ab" and "#ab" is considered to be a same hashtag
    You have to:
    1. Filter out the word that is hashtags.
       Hashtag usually start with "#" and followed by a series of alphanumeric
    2. map (hashtag) to (hashtag, 1)
    3. sum the count of current DStream state and previous state
    4. transform unordered DStream to a ordered Dstream
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (hashtag, count)
    """

    # TODO: insert your code here
    import re
    tagCounts = words.filter(lambda word: word.lower().startswith("#") and re.match('^[a-zA-Z0-9]+$',word[1:])).\
                map(lambda word:(word.lower(),1))
    tagCounts = tagCounts.updateStateByKey(aggregate_tags_count)
    return tagCounts.transform(lambda rdd: rdd.sortBy(lambda x: x[1], ascending=False))
```

2. Word counts.

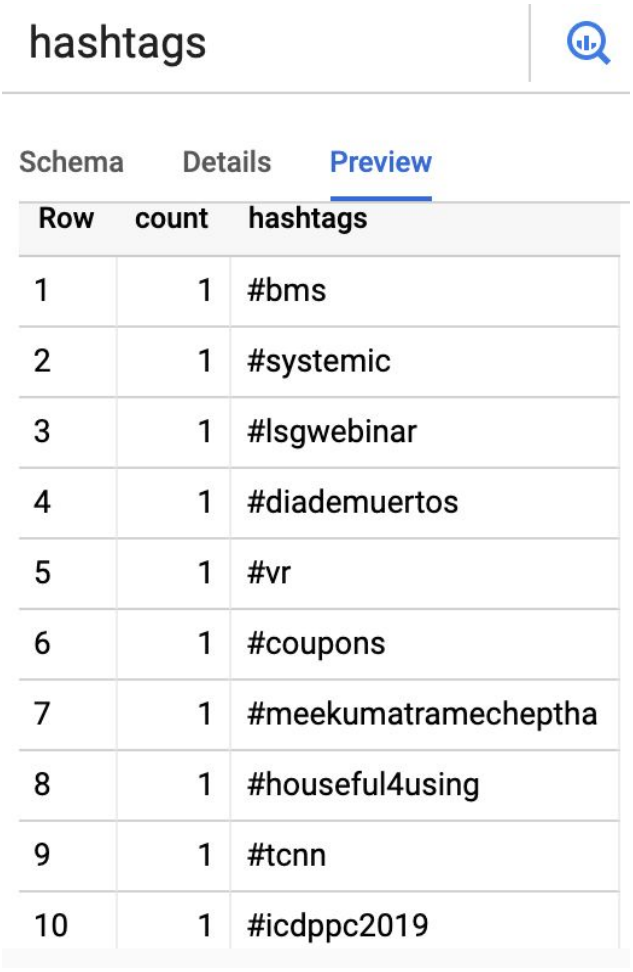
```
def wordCount(words):
    """
    Calculte the count of 5 sepcial words in 60 seconds for every 60 seconds (window no overlap)
    Your should:
    1. filter the words, case insensitive.
    2. count the word during a special window size
    3. add a time related mark to the output of each window, ex: a datetime type
    Hints:
        You can take a look at reduceByKeyAndWindow transformation
        Dstream is a series of rdd, each RDD in a DStream contains data from a certain interval
        You may want to take a look of transform transformation of DStream when trying to add a
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (word, count, time)
    """

    # TODO: insert your code here
    selected = words.filter(lambda x: any(word == x.lower() for word in WORD)).\
               map(lambda word:(word.lower(),1))
    counts = selected.reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 60, 60)
    return counts.transform(lambda time, rdd: rdd.map(lambda x: (x[0],x[1],time)))
```

3. Save to BigQuery.

```
words = dataStream.flatMap(lambda line: line.split(" "))

# calculate the accumulated hashtags count sum from the beginning of the stream
topTags = hashtagCount(words)
topTags.pprint()

# Calculte the word count during each time period 60s
wordCount = wordCount(words)
wordCount.pprint()

def saveTopTags(rdd):
    saveToStorage(rdd, output_directory_hashtags, columns_name_hashtags, "overwrite")
topTags.foreachRDD(saveTopTags)

def saveWordCounts(rdd):
    saveToStorage(rdd, output_directory_wordcount, columns_name_wordcount, "append")
wordCount.foreachRDD(saveWordCounts)

# start streaming process, wait for 600s and then stop.
ssc.start()
time.sleep(STREAMTIME)
ssc.stop(stopSparkContext=False, stopGraceFully=True)

# put the temp result in google storage to google BigQuery
saveToBigQuery(sc, output_dataset, output_table_hashtags, output_directory_hashtags)
saveToBigQuery(sc, output_dataset, output_table_wordcount, output_directory_wordcount)
```

b. Screenshot of the preview of data stored in BigQuery.

1. Table of “hashtags”:



hashtags

Schema Details Preview

Row	count	hashtags
1	1	#bms
2	1	#systemic
3	1	#lsgwebinar
4	1	#diademuertos
5	1	#vr
6	1	#coupons
7	1	#meekumatramecheptha
8	1	#houseful4using
9	1	#tcnn
10	1	#icdppc2019

The hashtags are ordered descendingly in the pprint():

```
----------------------------------------
Time: 2019-10-31 13:35:50
----------------------------------------
('#bigil', 531)
('#atlee', 121)
('#ai', 117)
('#srk', 108)
('#sanki', 98)
('#shahrukhkhan', 91)
('#viswasam', 91)
('#hvinoth', 53)
('#halloween', 35)
('#artificialintelligence', 29)
...
```

2. Table of “wordcount”

wordcount

Schema Details Preview

Row	time	count	word
1	2019-10-31 13:29:50 UTC	15	ai
2	2019-10-31 13:26:50 UTC	19	ai
3	2019-10-31 13:28:50 UTC	10	ai
4	2019-10-31 13:25:50 UTC	9	ai
5	2019-10-31 13:27:50 UTC	13	ai
6	2019-10-31 13:30:50 UTC	37	ai
7	2019-10-31 13:30:50 UTC	5	data
8	2019-10-31 13:29:50 UTC	1	data
9	2019-10-31 13:26:50 UTC	4	data
10	2019-10-31 13:28:50 UTC	2	data

The output in the terminal is:

```
-------------------------------------------
Time: 2019-10-31 13:30:50
-------------------------------------------
('movie', 207, datetime.datetime(2019, 10, 31, 13, 30, 50))
('good', 9, datetime.datetime(2019, 10, 31, 13, 30, 50))
('ai', 37, datetime.datetime(2019, 10, 31, 13, 30, 50))
('spark', 8, datetime.datetime(2019, 10, 31, 13, 30, 50))
('data', 5, datetime.datetime(2019, 10, 31, 13, 30, 50))
```