SparkStreamNotebook

November 15, 2018

1 Spark Streaming notebook

1.0.1 Soufiane MOUTEI - Ahmed BEN SAAD

```
In [1]: from pyspark import SparkContext
        from pyspark.streaming import StreamingContext
        from pyspark.sql import Row, SQLContext
        from pyspark.sql.functions import desc

        from IPython import display
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd

        import time, json
        from datetime import datetime, timezone, timedelta, date
```

To get the top tweets, we choose the following procedure:

- Take each text and associate it with **seconds**, the difference of seconds between now and the date it was written, thanks to the function *get_text_and_seconds*.
- Get the key-value pair *((tag, rangev), count)*:
  - **rangev** is based on the **seconds**, 30 if it's below 30, 60 if it's below 60 and 180 otherwise.
  - Before that, we split by space and take only the hashtags; then, we map ((tag, range), 1) and finally we apply reduce.
- Sort results by **count**, take the top 10 and then add it to the temporary table *tweets*.
- In a loop of 10 iterations that will be updated each 5 seconds, we create a pandas dataframe based on the temporary table we've created; then, we create a pandas dataframe where we filter by **rangev**, group by **tag** and sum **count**.
- The 3 dataframes will be plotted.
- Each iteration, the output is cleared so that each time we have only the new plots.

```
In [2]: def get_text_and_seconds(line):
            msg = json.loads(line)
            datetime_object = (
                datetime
                .strptime(msg['created_at'], '%a %b %d %H:%M:%S %z %Y')
```

```
            .replace(tzinfo=timezone.utc).astimezone(tz=None)
        )
        seconds = int(datetime.now().strftime("%s")) - int(datetime_object.strftime("%s"))
        return (msg['text'], seconds)

    def rangev(seconds):
        if seconds <= 30:
            return 30
        elif seconds <= 60:
            return 60
        else:
            return 180

    def get_pairs(rdd):
        return (
            # Split by space
            rdd.flatMap(lambda t: [(x, t[1]) for x in t[0].split(" ")])

            # Get only the hashtags
            .filter(lambda t: t[0].startswith("#"))

            # Associate each hashtag, range to 1
            .map(lambda t: ((t[0], rangev(t[1])), 1))

            .updateStateByKey(
                lambda new_values, lastState: sum(new_values) + (lastState or 0)
            )
        )
```

Now we'll create our Spark application.

```
In [3]: sc = SparkContext("local[2]", "TwitterAPP")
        ssc = StreamingContext(sc, 5)
        sqlContext = SQLContext(sc)

        # Setting a checkpoint to allow RDD recovery
        ssc.checkpoint("checkpoint_TwitterApp")

        # Connect to the port that sends tweets
        lines = ssc.socketTextStream('localhost', 7000).window(180)
```

We'll start working now on the data being streamed.

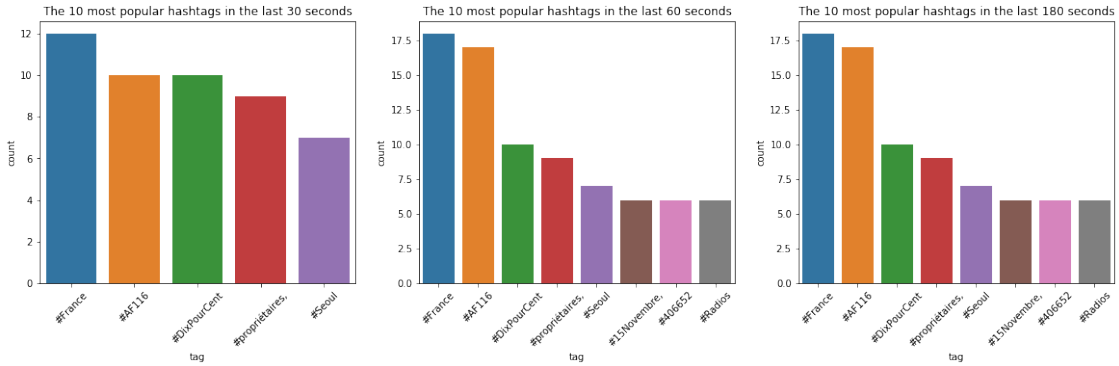
```
In [4]: # Get (text, seconds) pairs
        text_date = lines.map(get_text_and_seconds)

        # Get key-value pairs of tweet counting
        pairs = get_pairs(text_date)
```

```
        # Register the results in sqlContext
        (
            pairs
            .map(lambda w: Row(tag=w[0][0], rangev=w[0][1], count=w[1]))
                .foreachRDD(
                    lambda rdd: (
                        rdd.toDF()
                        .sort(desc("count"))
                        .limit(10)
                        .registerTempTable("tweets")
                    )
                )
        )

        ssc.start()
```

We'll plot the results.

```
In [5]: for i in range(10):
            # Wait for the gathering of result by spark
            time.sleep(5)

            # Clear output and prepare it for the new plots
            display.clear_output(wait=True)

            try:
                df = sqlContext.sql("Select tag, rangev, count from tweets").toPandas()

                plt.figure(figsize=(20, 5))

                for i, threshold in enumerate([30, 60, 180]):
                    df_to_plot = (
                        pd.DataFrame(
                            df[df.rangev <= threshold]
                            .groupby(['tag'])['count']
                            .sum()
                        )
                        .reset_index()
                        .sort_values("count", ascending=False)
                    )

                    if (df_to_plot.empty):
                        print("There is no hashtags in the last %d seconds" %threshold)
                        continue

                    plt.subplot(1, 3, i+1)
                    sns.barplot(
                        x="tag",
```

```
                    y="count",
                    data=df_to_plot
                )
                plt.title("The 10 most popular hashtags in the last %d seconds" % threshold)
                plt.xticks(rotation=45)

            plt.show()

        except:
            print("Empty table")
            continue
```



Finally, we'll close our StreamingContext.

```
In [6]: ssc.stop()
```