

Spark and stream processing: Twitter Sentiment Analysis

Lab exercise

Vincent Leroy, Natasha Tagasovska

2017

1 Getting started

1.1 Running spark programs

This lab exercise will be developed in [Scala](#), a functional programming language that is interoperable with Java. Scala may be a programming language you are not familiar with, but we will only be writing simple code for this exercise. In addition, we will use [Scala-IDE](#) to benefit from auto-completion and syntax coloring.

Download the zip file containing the initial version of the project from moodle. The dependencies and configuration of the project are listed in `twitter_sentiment.sbt` (no need to modify them). You can use the `sbt` ([Scala Build Tool](#)) to generate files for Scala-IDE to import. To do this, open a terminal, and at the root of the project (you should see `twitter_sentiment.sbt` when you type `ls`), run `sbt eclipse`. Now, in Scala-IDE, use the import wizard and the *Import existing projects into workspace* option to open your project in the IDE. You will not need to use command line again for this exercise.

The project contains 3 Scala source files:

- `BatchAnalysis.scala` will be used in the first part of the exercise to analyze a static dataset.
- `StreamAnalysis.scala` will be used in the second part of the exercise to process streaming data.
- `TweetUtilities.scala` contains functions that will help you extract information from tweets. You can look at the content of the file to see the functions provided and how they are implemented, but you do not need to modify it.

Both analysis files can be executed by right-clicking on the in the right part of the IDE and selecting *Run as Scala application*. Each of them executes Spark on your machine, using 6 threads (`local[6]` in Spark configuration). For a real case of large-scale data analysis you would deploy your execution on a YARN cluster to get more processing power, but this is not needed for this exercise as your computer should be powerful enough.

1.2 Dataset

Twitter can be used to monitor people's opinion on different topics. Brands like to use it to know what their reputation is and react quickly in case of negative trends. This type of analysis is generally called *Sentiment Analysis*. For this exercise, we will use two datasets:

- [SentiWordNet](#) is a dictionary that lists, for many english words, the sentiment associated. It can be a positive value (good opinion), or a negative value (bad opinion). For our exercise, the dictionary used is the `SentiWordNet_3.0.0_20130122.txt` file present at the root of the project. You can have a look to see what the content is, but you will not need to manipulate this data yourself. Instead, `TweetUtilities` will load it automatically for you, and you can use `TweetUtilities.getSentiment` on a tweet to obtain a sentiment value.
- `1Mtweets_en.txt` contains recent tweets crawled through the Twitter API. There is one tweet per line, and 1M tweets in total. Since our sentiment dictionary is in english, we focused on tweets written in english. Tweets contain words (`TweetUtilities.getWords`), hashtags (`TweetUtilities.getHashTags`) and mentions (`TweetUtilities.getMentions`). If you are interested in opinion on topics and brands, you are probably want to monitor hashtags ([#apple](#), [#rolex](#)), while if you are interested in important personalities, you are probably want to monitor mentions ([@realdonaldtrump](#), [@justinbieber](#)).

2 Batch processing

This exercise is done by modifying `BatchAnalysis.scala`. We want to process a static dataset of tweets `1Mtweets_en.txt`. We start by loading the dataset in a Spark RDD in order to process it: `val tweets: RDD[String] = sc.textFile("1Mtweets_en.txt")`. You can now process this dataset using all of the [RDD transformations](#) available. Keep in mind that Spark is lazy, and does not actually process data unless you execute an [Action](#). Finally, if you execute multiple actions on a RDD, it is a good idea to [cache](#) it in memory to avoid generating its content multiple time.

1. How many tweets mention Donald Trump? `String.contains` should be sufficient, no need to use `TweetUtilities` here.
2. Build a RDD that contains (tweet, sentiment) pairs indicating the sentiment of each tweet. To check that it is working properly, display the content of 5 elements (`take(5).foreach(println)`).
3. Using the previous RDD, create a RDD that represents the sentiment associated to each hashtag (or mention). Once again, display a few elements to check things work properly.
4. Now we wish to see the k most positive / negative hashtags. `takeOrdered(k)` gives you the k smallest elements according to the key, while `top(k)` gives you the k highest. When working on a pair, `swap` can be used to exchange the position of the key and the value.

3 Stream processing

We now wish to process tweets as they are published to know about the latest trends, so we switch to stream processing. We will now edit `StreamAnalysis.scala`. At the beginning of this file, you can see code that connects to a Kafka stream in order to obtain tweets. We prepared 2 stream you can connect to, and you can chose which one you want to use by modifying the `topics` value. The first one (by default) is called `live_tweets` and contains the latest tweets obtained through the Twitter API. We also have a backup stream, called `batch_tweets`, in case Twitter revokes our authorization, that loops over the batch dataset and emits it as a stream of new tweets.

Keep in mind that Spark processes streams by doing mini-batches on RDDs, and the interval between these executions is currently set to `Seconds(30)`. You can modify this value if you want to see the impact. Most of the code you wrote for batch processing can be directly applied for stream processing. When you want to display results using a transformation, you should encapsulate this code in `foreachRDD`. More documentation is available at [Spark Streaming Guide](#). Our goal is now to perform the same analysis as in the batch exercise but on streams instead.