# Assignment 3: predicting on streamed textual data

?

## Content

* The third assignment consists of the construction of a predictive model using Spark (Structured) Streaming and textual data. You will work with data coming from [Twitch](https://www.twitch.tv/), a popular platform for streaming (mainly games). More precisely, you will work with chat messages as posted on live channels.

The goal of this assignment is threefold:

* + 1 - make sure you collect a historical set of data (i.e. chat messages) coming from at least two channels
  + 2 - construct a predictive model that can predict the channel based on a chat message
  + 3 - show that your model can make predictions in a "deployed" setting on new messages

Setting up Spark

Since the data set we'll work with is still relatively small, you will (luckily) not need a cluster of machines, but can simply run Spark locally on your machine (and save the data locally as well). Here's how you make sure you have a local Spark set up and ready to go:

* + First, download the ZIP file from [this link](http://seppe.net/aa/assignment3/spark.zip) and extract it somewhere, e.g. on your Desktop. This ZIP file contains a portable Spark installation with Java 11 and necessary tooling included. It includes the latest stable Spark version available at this time (3.2.1).
  + Second, make sure you have a Python 3 Anaconda distribution installed on your system. (Python 3 and Jupyter Notebook need to be available on your system.)
  + Windows users: to start Spark, double-click letsgo-win.bat. If all goes well, a Jupyter notebook should open up with a connection to Spark being established and ready to go. If the launcher fails to find Python or Jupyter, it will provide you with an error message. Upon starting, Java might require access to your Windows firewall, which you can safely accept.
  + Mac users: if you're on Mac, open up a Terminal window and navigate to where you've unzipped the file and run letsgo-mac.sh, e.g.:

cd /Users/seppe/Desktop/spark

./letsgo-mac.sh

Again, do make sure you have installed Python 3 Anaconda first. If Mac complains about the file not being executable, you might first have to enter the following command to make it executable:

chmod +x ./letsgo-mac.sh

You might also get a window popup asking if you want to install XCode. You can ignore this, as you don't need it. If all goes well, a Jupyter notebook should open up with a connection to Spark being established.

If you encounter issues, check the FAQ below first -- otherwise, feel free to e-mail me.

Example notebooks

Once you have Jupyter notebook open, feel free to explore the example notebooks under "notebooks".

* + spark\_example.py.ipynb: A simple Spark example (calculating pi). This is a good check to see whether Spark is working correctly

To work with the streaming data source, you first need to start an additional small Python program, included in notebooks/local\_proxy.py. The reason for this is as follows: one of the built-in Spark Streaming connectors allows to connect to a plain text network socket, but the server (running at seppe.net:7778) requires that a connected client first sends a list of channels they want to listen to, which Spark cannot handle. Normally, one would, in cases such as these, [make a custom Spark Streaming receiver](https://spark.apache.org/docs/3.2.1/streaming-custom-receivers.html), which is only really viable when using Java or Scala. (It actually is possible to do this in Python, but would require a lot of Py4J hacks.) Instead, we do something else here, and use a simple program to connect to the remote server, send the channels, and then pass on all received data and expose it on a locally running server. Spark then connects to the local socket.

The program is invoked as follows: python local\_proxy.py channel1 [more channels], i.e. a space-separated list of channels needs to be provided. You should see the following output:

Remote seppe.net:7778 forwarded to 0.0.0.0:8080

Press CTRL+C to stop

You can then try out the Spark notebooks:

* + spark\_streaming\_example.py.ipynb: A simple Spark Streaming example that prints out the data you'll work with
  + spark\_streaming\_example\_saving.py.ipynb: A simple Spark Streaming example that saves the data
  + spark\_streaming\_example\_predicting.py.ipynb: A very naïve prediction approach
  + spark\_structured\_streaming\_example.py.ipynb: An example using Spark Structured Streaming

Objective

Using Spark, your task for this assignment is as follows:

* + Construct a data set using the provided stream
    - Important: get started with this as soon as possible. We will discuss Spark and text mining in more detail later on, but you can already start gathering your data
    - Make sure to set up Spark using the instructions posted above
  + Construct a predictive model to predict the channel based on the chat message
    - You can use your own favorite channels, but probably it is a good idea to use relatively frequent streamers with an active chat
    - The stream is text-based with each line containing one message (one instance) formatted as a JSON dictionary
    - You can use extra data and libraries if you want, but this is not required
    - You are strongly encouraged to build your model using spark.ml (MLlib), but you can use scikit-learn as a fallback if things don't work out
  + Use your trained model to show you can make predictions as the stream comes in
    - I.e. show that you can connect to the data source, preprocess/featurize incoming messages, have your model predict the label, and show it, similar to spark\_streaming\_example\_predicting.py.ipynb (but hopefully using a smarter, real predictive model)
    - This means that you'll need to look for a way to save and load your trained model
    - This sounds like a difficult task, but you will probably find lots of easy ways to featurize. I.e. users tend to use certain emojis in particular channels. You could also keep a list of which users were seen in which channel and use that to enhance your model, etc.
    - Additional fun things you could do: apply a sentiment prediction model such as [VADER](https://github.com/cjhutto/vaderSentiment), create a small dashboard showing chat engagement (chats per second, hype moments, most used emojis etc.) in real time. Not necessary for the assignment, but feel free to play around

The third part of your lab report should contain:

* + Overview of the steps above, the source code of your programs, as well as the output after running them
  + Feel free to include screen shots or info on encountered challenges and how you dealt with them
  + Even if your solution is not fully working or not working correctly, you can still receive marks for this assignment based on what you tried and how you'd need to improve your end result (i.e. when you can prove a conceptual understanding of the problem and solution)

Further remarks

* + Get started with setting up Spark and fetching data as quickly as possible. If you encounter troubles getting Spark to run, do let me know
  + Make sure to have enough data to train your model
  + You can use as many channels as you like, but two will most likely lead to the easiest setup (as it boils down to binary classification)
  + The data stream is line delimited with every line contain a review in JSON format, but can be easily converted to a DataFrame (and RDD). The example notebooks give some ideas on how to do so
  + You can use both Spark Streaming or Spark Structured Streaming. The former is probably easier to work with
  + The focus of this assignment is on getting the full pipeline as outlined above constructed, and not on getting spectacularly high accuracies, though do think about particular featurization approaches that work in a streaming context (i.e. something you can calculate or derive from the last n messages)
  + Preferably, your predictive model needs to be build using MLlib (so read documentation and tutorials). In case you encounter trouble, you can use scikit-learn as well to still perform the "deployment" stage. As stated above, other libraries can be used as well, as long as you can show that your model can provide predictions in real-time
  + Let me know in case the streaming server would crash (don't hesitate to do so)

FAQ

* + letsgo-win.bat doesn't start the notebook -- it says I have spaces in my path
    - Try again by moving the installation to a directory without spaces.
  + letsgo-win.bat doesn't start the notebook -- it says it can't find Python
    - Make sure to start the script from an Anaconda terminal. I.e. the name of your Anaconda environment should show up in the prompt between parenthesis (like (base) C:\>). You can start an Anaconda terminal using "Anaconda Navigator", press the small arrow next to your environment in the "Environments" tab, and picking "Open Terminal".
  + letsgo-win.bat doesn't start the notebook -- the command line window just disappears
    - Most likely, the script has trouble to find your Anaconda installation. Try to start the script from an Anaconda terminal (see above) and check the output.
  + sc is not defined when I run the example notebook.
    - You probably opened up Jupyter yourself rather than through the letsgo file. In this case, variables such as sc will not be initialized.
  + Spark starts, but the example notebooks fail and I see the following error in the terminal: "Python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from Settings > Manage App Execution Aliases."
    - This is Windows being silly. Go to Settings, search for "Manage App Execution Aliases", and turn off "python" and "python3".
  + Spark starts, but the example notebooks fail and I see the following error in the terminal: "spawn error" on Mac
    - Edit the letsgo-mac.sh script and remove the line starting with "export JAVA\_HOME". You will have to install Java 11 yourself.
  + Spark starts, but the example notebooks fail and I see the following error in the terminal: "java.io.IOException: Cannot run program "python": CreateProcess error=2, The system cannot find the file specified"
    - Try editing the script and change the line containing: set PYSPARK\_PYTHON=python for Windows or export PYSPARK\_PYTHON=python for Mac so it says python3.
  + I can't save the stream... everything else seems fine
    - Make sure you're calling the "saveAsTextfiles" function with "file:///" prepended to the path: lines.saveAsTextFiles("file:///C:/..."). Also make sure that the folder where you want to save the files exist. Note that the "saveAsTextfiles" method expects a directory name as the argument. It will automatically create a folder for each mini-batch of data.
  + Can I prevent the saveAsTextFiles function from creating so many directories and files?
    - You can first repartition the RDD to one partition before saving it: lines.repartition(1).saveAsTextFiles("file:///C:/..."). To prevent multiple directories, change the trigger time to e.g. ssc = StreamingContext(sc, 60), though this will still create multiple directories. Setting the trigger interval higher is not really recommended, as you wouldn't want to lose data in case something goes wrong.
  + So if I still end up with multiple directories, how do I read them in?
    - It's pretty easy to loop over subdirectories in Python. This can be easily found on Google. Another tip: alternatively, the sc.textFile command is pretty smart and can parse through multiple files in one go.
  + Is it normal all my folders only contain \_SUCCESS files but no actual data files?
    - That depends. A \_SUCCESS file indicates that the mini-batch was saved correctly. part-\* files contain the actual data. And files ending with .crc contain a checksum which you can ignore. It's normal if not all of your folders contain part-\* data, as it might be that no data was received in that time frame. However, if none of your folders are having data and you've been trying to run the code for some time, something else has gone wrong. Try the spark\_streaming\_example.py.ipynb notebook to verify whether you're at least receiving data at all.
  + The local\_proxy.py script shows an error.
    - Try changing the local port LOCAL\_PORT = 8080 to another value (you will also have to change it in the notebooks). Also try setting LOCAL\_HOST = "127.0.0.1". If this doesn't work, let me know.
  + Is there a way how I can monitor Spark?
    - Yes, go to http://127.0.0.1:4040/ in your browser while Spark is running and you'll get access to a monitoring dashboard. Under the "Environment" tab, you should be able to find a "spark.speculation" entry for instance w.r.t. the question above. Under "Jobs", "Stage", and "Streaming", you can get more info on how things are going.
  + I'm trying to convert my saved files to a DataFrame, but Spark complains for some files?
    - Data is always messy, especially the ones provided by this instructor. Make sure you can handle badly formatted lines and discard them.
  + My stream crashes after a while with an "RDD is empty" error...
    - Make sure you're checking for empty RDDs, e.g. if rdd.isEmpty(): return.
  + I've managed to create a model. When I try to apply it on the stream, Spark crashes with a Hive / Derby error, e.g. when I try to .load() my model(s) or once the first RDD arrives
    - Check the example notebooks for ideas on how to load in your model in "globals()" once.
  + When I call ssc\_t.stop(), Spark never seems to stop the stream
    - You can try changing stopGraceFully=True to False. Even then, Spark might not want to stop its stream processing pipeline in case you're doing a lot with the incoming data, preventing Spark from cleaning up. Try decreasing the trigger time, or simply restart the Jupyter kernel to start over.
  + Spark complains that only one StreamingContext can be active at a time (and other general "it doesn't work anymore" questions)
    - In case of trouble, a good idea is always to (save and) close all running notebooks and start again fresh. Spark doesn't like having multiple StreamingContext notebooks running, so it is best to only have one notebook running at a time. (Note: closing a tab with a notebook does not mean that the kernel is stopped, check the "Running" tab on the Jupyter main page.)
  + Can I use R?
    - There are two main Spark R packages being maintained right now: SparkR (the official one) and sparklyr (from the folks at RStudio and fits better with the tidyverse). Both are fine to use, but you'll have to do some setting up in order so R can find your Spark installation. The example below assumes you have the "SparkR" library installed. However, I'd strongly recommend using Python.

# Set up environment variables: do this before loading the the library

# Otherwise, R will attempt to download spark on its own

# Make sure to adjust the paths below to match your system

if (nchar(Sys.getenv("SPARK\_HOME")) == 0) {

Sys.setenv(SPARK\_HOME = "C:\\Users\\Seppe\\Desktop\\spark\\spark-3.2.1-bin-hadoop2.7\\")

# HADOOP\_HOME is only required on Windows, comment out the line below on Max/Linux

Sys.setenv(HADOOP\_HOME = "C:\\Users\\Seppe\\Desktop\\spark\\winutils\\")

}

library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK\_HOME"), "R", "lib")))

# Set up the Spark Context:

sparkR.session(master = "local[\*]")

# Read a structured streaming data frame:

data = read.stream("socket", host = "localhost", port = 8080)

# Provide a sink, using "console" will not show anything on your main R console

# So use memory with a queryName instead:

query = write.stream(data, "memory", queryName="data")

# Your operations:

head(sql("SELECT \* FROM data"))

# Once you are finished, stop the query:

stopQuery(query)