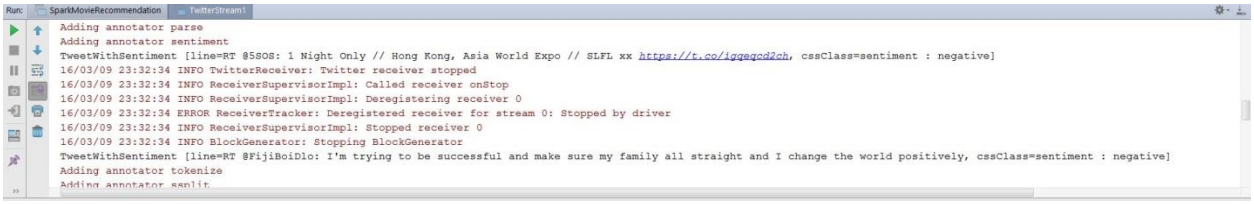
CS5542 Big Data Apps and Analytics
LAB ASSIGNMENT #8
REPORT and SCREEN SHOTS

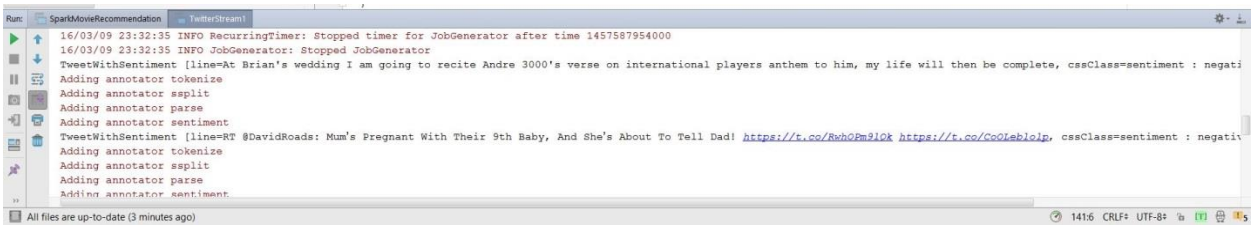
1) Add Sentimental Analysis using Twitter Streaming (related to project)

We begin by streaming data from Twitter. We use the filter feature in spark to only include Tweets in English. We then use the sentiment analyzer to classify sentiment into negative and positive connotations.

We've provided an example here.



We show several more examples.



2) Make Reccomendations (related to project)

For this section, we utilized the SparkMovieReccomendation.scala code to recommend restaurants to the user. Using 25 handpicked restaurants, we picked the top 10 restaurants recommended based on our user's preferences.

We partitioned the data into training (60%), testing (20%), and validation (20%). The training set uses the last digit of the time step to provide a random number between 1 and 9.

```
val numPartitions = 4
val training = ratings.filter(x => x._1 < 6)
  .values
  .union(myRatingsRDD)
  .repartition(numPartitions)
  .cache()
val validation = ratings.filter(x => x._1 >= 6 && x._1 < 8)
  .values
  .repartition(numPartitions)
  .cache()
val test = ratings.filter(x => x._1 >= 8).values.cache()

val numTraining = training.count()
val numValidation = validation.count()
val numTest = test.count()

println("Training: " + numTraining + ", validation: " + numValidation + ", test: " + numTest)
```

We then train the model and optimize to find the best training model:

```
// train models and evaluate them on the validation set

val ranks = List(8, 12)
val lambdas = List(0.1, 10.0)
val numIters = List(10, 20)
var bestModel: Option[MatrixFactorizationModel] = None
var bestValidationRmse = Double.MaxValue
var bestRank = 0
var bestLambda = -1.0
var bestNumIter = -1
for (rank <- ranks; lambda <- lambdas; numIter <- numIters) {
  val model = ALS.train(training, rank, numIter, lambda)
  val validationRmse = computeRmse(model, validation, numValidation)
  println("RMSE (validation) = " + validationRmse + " for the model trained with rank = "
    + rank + ", lambda = " + lambda + ", and numIter = " + numIter + ".")
  if (validationRmse < bestValidationRmse) {
    bestModel = Some(model)
    bestValidationRmse = validationRmse
    bestRank = rank
    bestLambda = lambda
    bestNumIter = numIter
  }
}
```

This screenshot shows the iterations of training:

```
RMSE (validation) = 5.29230621162741 for the model trained with rank = 8, lambda = 0.1, and numIter = 10.
RMSE (validation) = 5.463701808903616 for the model trained with rank = 8, lambda = 0.1, and numIter = 20.
RMSE (validation) = 6.475417707203035 for the model trained with rank = 8, lambda = 10.0, and numIter = 10.
RMSE (validation) = 6.475417707203036 for the model trained with rank = 8, lambda = 10.0, and numIter = 20.
RMSE (validation) = 4.353967647451537 for the model trained with rank = 12, lambda = 0.1, and numIter = 10.
RMSE (validation) = 4.6815966390435815 for the model trained with rank = 12, lambda = 0.1, and numIter = 20.
RMSE (validation) = 6.475417707203036 for the model trained with rank = 12, lambda = 10.0, and numIter = 10.
RMSE (validation) = 6.475417707203036 for the model trained with rank = 12, lambda = 10.0, and numIter = 20.
```

The final reccomendations:

```
The best model was trained with rank = 12 and lambda = 0.1, and numIter = 10, and its RMSE on the test set is 5.420072919734174.
The best model improves the baseline by -392.67%.
Restaurants recommended for you:
 1: Ruby Tuesday
 2: Panera
 3: IHOP
 4: Pizza Hut
 5: Taco Bell
 6: KFC
 7: Red Lobster
 8: Jack Stack
 9: Five Guys
10: Burger King
```

3) Reccomendation sent to smartphone

After creating the recommendations above, our goal is to send these recommendations to the User. Therefore, we have taken this recommendation output, and sent it as a notification to the smartphone. We show the screenshot below: