CISC 5950 - Big Data Programming

Project 2 - Part 1 Report

Team Members:
Revathi Bhuvaneswari
Tianying Luo
Youfei Zhang
Yue Zeng

Spring 2019

P1: Toxic Comment Classification

Bash script:

Import both the train and the test data file from HDFS.

```
#!/bin/bash
source ../env.sh
/usr/local/hadoop/bin/hdfs dfs -rm -r /project2/P1/input/
/usr/local/hadoop/bin/hdfs dfs -mkdir -p /project2/P1/input/
/usr/local/hadoop/bin/hdfs dfs -copyFromLocal ../../data/train.csv /project2/P1/input/
/usr/local/hadoop/bin/hdfs dfs -copyFromLocal ../../data/test.csv /project2/P1/input/
/usr/local/spark/bin/spark-submit --master=spark://$SPARK_MASTER:7077 ./p1.py hdfs://$SPARK_MASTER:9000/project2/P1/input/
```

Python Script:

Stage 1: Initiative spark session and load the data as Spark DataFrame. Each feature column is converted into int data type manually.

```
# set up the session
spark = (SparkSession.builder.appName('Toxic Comment Classification').getOrCreate())

# load the data as pandas dataframe
# then convert to Spark DataFrame
df = pd.read_csv(sys.argv[1])
df.fillna("", inplace=True)
df = spark.createDataFrame(df)

# load the data as dataframe
df = spark.read.format("csv").load(sys.argv[1], header="true", inferSchema="true")

# change class values to int
df1 = df.withColumn('toxic', col('toxic').cast(IntegerType()))
df2 = df1.withColumn('severe_toxic', col('severe_toxic').cast(IntegerType()))
df3 = df2.withColumn('obscene', col('obscene').cast(IntegerType()))
df4 = df3.withColumn('threat', col('threat').cast(IntegerType()))
df5 = df4.withColumn('insult', col('insult').cast(IntegerType()))
df6 = df5.withColumn('identity_hate', col('identity_hate').cast(IntegerType()))
```

Stage 2: Text preprocessing. Regular expression is used to get rid of punctuation, space, newline, and irregular texts in the “comment_text” column. In addition, the columns with NA and Null values are dropped.

```
# clean the text
# get rid of new lines
df = train.select((lower(regexp_replace('comment_text', "[^a-zA-Z\\s]", ""))\
      .alias('text')), 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate')
data = df.select((regexp_replace('text', "[\r\n]+", "")\
      .alias('text')), 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate')
data.na.drop()
data.na.fill(0)
```

Stage 3: Convert the “comment_text” column into a column of sparse vector.

Firstly, each text column is tokenized in to a word lists with **pyspark.ml.feature.Tokenizer:**

```
# Basic sentence tokenizer
tokenizer = Tokenizer(inputCol="text", outputCol="words")
words_token = tokenizer.transform(clean)
```

Next, each word list is is further cleaned with **pyspark.ml.feature.StopWordsRemover:**

```
# Remove stop words
remover = StopWordsRemover(inputCol='words', outputCol='words_clean')
df_words_no_stopw = remover.transform(words_token)
```

Then, the cleaned word lists are converted into a bag of words and TFIDF, in the form of sparse vector. With **pyspark.ml.feature.HashingTF,** the bag of words of each word list is extracted, which simply counts the frequency of each word regards the who corpus, and produce the frequency to represent each work. With **pyspark.ml.feature.IDF,** the TFIDF is computed, which produce an inverse matrix of bag of words, under the assumption that the most frequent words are usually less important in a word list context.

```
# term frequency
hashingTF = HashingTF(inputCol="words_clean", outputCol="rawFeatures")
tf = hashingTF.transform(df_words_no_stopw)

# tfidf
idf = IDF(inputCol = "rawFeatures", outputCol = "features")
idfModel = idf.fit(tf)
tfidf = idfModel.transform(tf).select('features', 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate')
```

Stage 4: Training. After obtained the sparse vector that represent the “comment_text” column, we start fitting the data into a logistic regression model with **pyspark.ml.classification.LogisticRegression.** Here we use an learning rate of 0.1.

```
# logistic regression
REG = 0.1
lr = LogisticRegression(featuresCol="features", labelCol='toxic', regParam=REG)
lrModel = lr.fit(tfidf.limit(5000))
lr_train = lrModel.transform(tfidf)
lr_train.select("toxic", "probability", "prediction").show(20)
```

Stage 4: Testing. With the fitted model on the training dataset, we will be able to train the model on the test dataset. Firstly, the test data is also loaded, parsed and cleaned following following the same procedure in the training step. Then, the cleaned dataset is tokenized, removed the stop words, hashed into bag of words, and transformed into TFIDF sparse vectors, with the fitted model from training step.

```
# ---------------- Testing ----------------
# test set

test_raw = pd.read_csv(sys.argv[2])
test_raw.fillna("", inplace=True)
test_raw = spark.createDataFrame(test_raw)

# clean the text
# get rid of new lines
test = test_raw.select('id', (lower(regexp_replace('comment_text', "[^a-zA-Z\\s]", """)).alias('text')))
test = test.select('id', (regexp_replace('text', "[\r\n]+", "").alias('text')))
test.na.drop()
test.na.fill(0)

test_tokens = tokenizer.transform(test)
test_words_no_stopw = remover.transform(test_tokens)
test_tf = hashingTF.transform(test_words_no_stopw)
test_tfidf = idfModel.transform(test_tf)
```

Given that we have 6 binary class labels ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate'], that each indicate the characteristics of toxicity of the comment, we run logistic regression iteratively, for each of the label.

```
# a udf to extract the probability of class 1: x[1]
extract_prob = F.udf(lambda x: float(x[1]), T.FloatType())
out_cols = [i for i in train.columns if i not in ["id", "comment_text"]]
test_res = test.select('id')

# fit the model to each label of the test data
for col in out_cols:
    print(col)
    lr_test = LogisticRegression(featuresCol="features", labelCol=col, regParam=REG)
    print("...fitting")
    model = lr_test.fit(tfidf)
    print("...predicting")
    res = model.transform(test_tfidf)
    print("...appending result")
    test_res = test_res.join(res.select('id', 'probability'), on = "id")
    print("...extracting probability")
    test_res = test_res.withColumn(col, extract_prob('probability')).drop("probability")
    test_res.show(20)
```

Output:

```
2019-05-05 16:17:08 WARN  TaskSetManager:66 - Stage 912 contains a task of very l
+--------------------+-----+------------+-------+------+------+-------------+
|                text|toxic|severe_toxic|obscene|threat|insult|identity_hate|
+--------------------+-----+------------+-------+------+------+-------------+
|explanationwhy th...|    0|           0|      0|     0|     0|            0|
|daww he matches t...|    0|           0|      0|     0|     0|            0|
|hey man im really...|    0|           0|      0|     0|     0|            0|
|morei cant make a...|    0|           0|      0|     0|     0|            0|
|you sir are my he...|    0|           0|      0|     0|     0|            0|
|congratulations f...|    0|           0|      0|     0|     0|            0|
|cocksucker before...|    1|           1|      1|     0|     1|            0|
|your vandalism to...|    0|           0|      0|     0|     0|            0|
|sorry if the word...|    0|           0|      0|     0|     0|            0|
|alignment on this...|    0|           0|      0|     0|     0|            0|
|fair use rational...|    0|           0|      0|     0|     0|            0|
|bbq be a man and ...|    0|           0|      0|     0|     0|            0|
|hey what is it  t...|    1|           0|      0|     0|     0|            0|
|before you start ...|    0|           0|      0|     0|     0|            0|
|oh and the girl a...|    0|           0|      0|     0|     0|            0|
|juelz santanas ag...|    0|           0|      0|     0|     0|            0|
|bye dont look com...|    1|           0|      0|     0|     0|            0|
|redirect talkvoyd...|    0|           0|      0|     0|     0|            0|
|the mitsurugi poi...|    0|           0|      0|     0|     0|            0|
|dont mean to both...|    0|           0|      0|     0|     0|            0|
+--------------------+-----+------------+-------+------+------+-------------+
only showing top 20 rows
```

Figure 1: Initial DataFrame (first 20 rows)

```
2019-05-05 16:51:45 INFO  BlockManagerInfo:54 - Added broadcast_71_piece0 in memory on 10.150.
0.6:40613 (size: 23.7 KB, free: 413.3 MB)
2019-05-05 16:51:45 INFO  TaskSetManager:54 - Finished task 0.0 in stage 63.0 (TID 43) in 884
ms on 10.150.0.6 (executor 1) (1/1)
2019-05-05 16:51:45 INFO  TaskSchedulerImpl:54 - Removed TaskSet 63.0, whose tasks have all co
mpleted, from pool
2019-05-05 16:51:45 INFO  DAGScheduler:54 - ResultStage 63 (showString at NativeMethodAccessor
Impl.java:0) finished in 0.917 s
2019-05-05 16:51:45 INFO  DAGScheduler:54 - Job 33 finished: showString at NativeMethodAccesso
rImpl.java:0, took 0.922620 s
+-----+--------------------+----------+
|toxic|         probability|prediction|
+-----+--------------------+----------+
|    0|[0.97289468024575...|       0.0|
|    0|[0.98601176759147...|       0.0|
|    0|[0.95194005879466...|       0.0|
|    0|[0.93307277335159...|       0.0|
|    0|[0.95341944050432...|       0.0|
|    1|[0.58328183305881...|       0.0|
|    0|[0.94984793360803...|       0.0|
|    0|[0.98252801938725...|       0.0|
|    0|[0.96750352823240...|       0.0|
|    0|[0.97841304286166...|       0.0|
|    1|[0.27481006931446...|       1.0|
|    0|[0.97435929408729...|       0.0|
|    0|[0.97845090218522...|       0.0|
|    0|[0.97054064957178...|       0.0|
|    0|[0.98679870155216...|       0.0|
|    0|[0.96560681669964...|       0.0|
|    0|[0.98730901842992...|       0.0|
|    0|[0.97824916740781...|       0.0|
|    0|[0.96240440390016...|       0.0|
|    0|[0.94497492652189...|       0.0|
+-----+--------------------+----------+
only showing top 20 rows
```

Figure 2: Logistic Regression Fitting on Training Data