Team hadoop05
Nikolaos Tsikoudis
Dimokritos Stamatakis
Tyler Lichten
William Edgecomb

CS129a Programming Assignment 3

1. What we did:

The framework of our code consists of a package for creating the vectors from the article lemma index and professions.txt resource file by running two MapReduce jobs (CreateVectorMapred.java and ExportLemmasMapred.java), and a package for running our code with Mahout using Naive Bayes Classifier (TrainTestNBayes.java). We did not explore additional Mahout machine learning algorithms beyond Naive Bayes Classifier.

1.A. Package code.vectorcreate

1.A.1. ExportLemmasMapred

This class runs the 1st MapReduce job (referred to as MR1), using both map and reduce, which performs some preprocessing on the ARTICLE_LEMMA_INDEX that is provided on the cs129a account. The output of MR1 gives us our vocabulary, a list of all the (tokenized and lemmatized) unique words that appear in the provided wikipedia documents, as well as the document frequencies (df) for each lemma. The input to MR1 is the training data, a selection of of about 80% of the ARTICLE_LEMMA_INDEX file. The preprocessing excludes words only appearing once as we considered these lemmas just to be noise and we also throw away the article title.

1.A.2. CreateVectorMapred

This class runs the 2nd MapReduce job (referred to as MR2), which only uses a mapper and not a reducer, in order to build vectors for both the test and training sets. These vectors are used to train and test the Naive Bayes model. The MapReduce job creates a Sequence File that contains feature vectors that are associated with professions from the professions.txt resource file. The Sequence File is copied to a local disk in order to be accessed by our code that runs the classifier. A lemma information map data structure is built to contain the vector index and its document frequency (df). We parsed through the professions.txt resource file in order to build another data structure that holds the article title with their associated list of professions. The map function in MR2 creates and outputs the profession vector pair. The feature values which are set to vector are calculated using TF-IDF (tfIDF). The context that is written as part of the map function depends on whether or not the set is the test or training set. An important note is that our test vector excludes any lemmas that are out of our vocabulary built in MR1. Additionally, for the test set, we exclude from the output any articles that do not have a profession because we will not be able to use these to evaluate the output of our classifier. When running MR2, the commands parameters are different for test and training sets (see part 2 of report).

1.B. Package code.runmahout

1.B.1. TrainTestNBayes

This class utilizes Mahout math and classifier packages for testing and training the Naive Bayes Model. To build our auxiliary data structures, we read over the training Sequence File (path containing training vectors) to build a list of unique professions and also read over the test Sequence File (path containing test vectors) to collect the article titles and expected professions used for evaluation. We checked our predictions using the three most likely professions, which are the first three elements in the sorted list we maintain containing the profession probabilities returned by the classifier. The three most likely professions were extracted and checked against the actual professions using the Comparable custom class PredictionIndexPair. A counter keeps track of the number of correct predictions that were made, and we divide this value by the total number of predictions in order to calculate our accuracy percentage. At first, our prediction percentages were very low in Java (around 1%), but we made some modifications to our code that helped fix this our results, such as using TF-IDF in our calculations and excluding for the vocabulary the words that appear in only one document. An observation that we had while testing was that building a test set of articles beginning with letters all earlier in the alphabet (A, B, etc.) might not be ideal because it could introduce bias to the results, as individual professions of may be disproportionately associated with different first letters of their article titles. In addition to the Java code, we also tried using Mahout commands in the command line.

2. How to run our code:

We have split the provided lemma index file into two files. One file contains the documents used for training our model (stored in the pa3LemmaIndexTrain folder) and the other file contains the documents used to test our classifier (stored in the pa3LemmaIndexTest folder).

2.A. Running MR1:

```
time yarn jar cs129a-pa3-0.0.1-SNAPSHOT.jar code.vectorcreate.ExportLemmasMapred
-Dmapred.job.queue.name=hadoop05 pa3LemmaIndexTrain pa3MR1outputTrain
```

2.B. Running MR2:

2.B.1. Command to create training vectors:

```
hdfs dfs -rm -r pa3TrainVectors; time yarn jar cs129a-pa3-0.0.1-SNAPSHOT.jar
code.vectorcreate.CreateVectorMapred -Dmapred.job.queue.name=hadoop05 -Dtype=train
-DtrainingLemmasPath=/user/hadoop05/pa3MR1outputTrain/part-r-00000
-DnumTrainingDocs=533670 pa3LemmaIndexTrain pa3TrainVectors
```

2.B.2. Command to create test vectors:

```
hdfs dfs -rm -r pa3TestVectors ; time yarn jar cs129a-pa3-0.0.1-SNAPSHOT.jar
code.vectorcreate.CreateVectorMapred -Dmapred.job.queue.name=hadoop05 -Dtype=test
-DtrainingLemmasPath=/user/hadoop05/pa3MR1outputTrain/part-r-00000
-DnumTrainingDocs=533670 pa3LemmaIndexTest pa3TestVectors
```

2.B.3. Parameters for Running MR2:

Type of set is training: -Dtype=train
Type of set is test: -Dtype=test
Path of the training set's
lemmas:-DtrainingLemmasPath=/user/hadoop05/pa3MR1outputTrainNoDF1/part-r-00000
Number of documents in training set: -DnumTrainingDocs=533670

2.C. Training the model and running the classifier:

```
java -cp cs129a-pa3-0.0.1-SNAPSHOT.jar code.runmahout.TrainTestNBayes 1
```

This part also creates a file (prediction-results.txt) with our 3 predictions for each article, as:
article_name : profession1, profession2, profession3

2.D. Important Note About Additional JARs:

In order to avoid [Exception in thread "main" java.lang.IncompatibleClassChangeError: Found interface org.apache.hadoop.mapreduce.JobContext, but class was expected] we needed to install additional cloudera jars. We installed *mahout-core-0.9-cdh5.2.0.jar* and *mahout-core-0.9-cdh5.2.0-job.jar* in our local maven repository and then added them to the pom as a dependency.

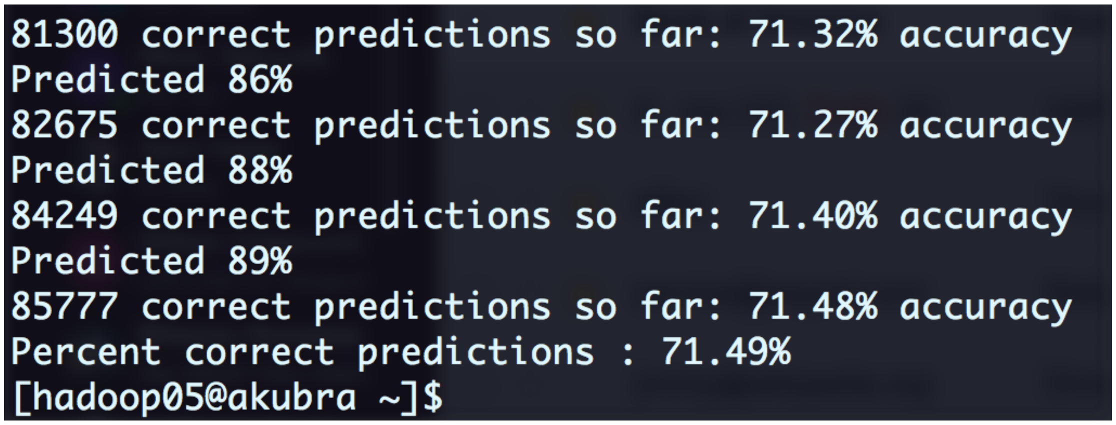
3. Evaluation results:

Accuracy of each split:

90/10 split = 72.5% accurate
80/20 split = 71.49% accurate

Mahout Command Line:

When we ran with the mahout command line we got 60% accuracy, since it only takes the highest prediction, rather than the top 3 (Also for each article with n professions, there are n identical vectors, of which at most 1 will be classified correctly).



```
81300 correct predictions so far: 71.32% accuracy
Predicted 86%
82675 correct predictions so far: 71.27% accuracy
Predicted 88%
84249 correct predictions so far: 71.40% accuracy
Predicted 89%
85777 correct predictions so far: 71.48% accuracy
Percent correct predictions : 71.49%
[hadoop05@akubra ~]$
```

Screenshot of the 80/20 evaluation