1 RDDs

The Hadoop job to calculate χ^2 values was implemented in Spark using Scala. The implementation can be seen in `chisquare.ipynb`. When comparing the implementations, one notices that the Spark implementation is way shorter. It is also faster than the Hadoop one.

2 Datasets/DataFrames

The DataFrame implementation is again shorter and faster than the RDD implementation. It is included in `pipeline.ipynb`. The main difference of the resulting terms is that the RDD implementation selects the most important terms per category, while the DataFrame implementation selects the most important terms overall.

3 Text Classification

Since there were resource issues on the cluster, some shortcuts had to be mase. Instead of k-fold cross validation, `TrainValidationSplit` (i.e. 1-fold cross validation) is used. The pipeline was run only on the devset. Also, a decision tree is used instead of an SVM. Overall, performance could probably substantially be improved by limiting the terms of CountVectorizer or reducing the dimensionality of TFIDF. Both running the job with the whole dataset (`reviewscombined.json`) and with LinearSVC failed with Figure 1 after 35h of runtime.

The validation results can be seen in Table 1. One sees that minInstancesPerNode displays only minor differences, but lower values are better, since this results in more splits. Somehow like neural networks, the deeper the tree is, the better. Also, the Gini impurity works better than entropy since its function is smoother. Very interestingly, there is virtually no difference in numTopFeatures. It seems that using 400 terms instead of 4000 does not severely limit performance. This suggests that there are only a few terms which thouroghly discriminate the dataset.

Surprisingly, the F1 score on the test dataset with the best performing model was 0.68.

numTopFeatures	maxDepth	impurity	minInstancesPerNode	F1
400	2	entropy	1	0.270
400	2	entropy	100	0.270
400	2	gini	1	0.274
400	2	gini	100	0.274
400	4	entropy	1	0.314
400	4	entropy	100	0.314
400	4	gini	1	0.312
400	4	gini	100	0.312
400	9	entropy	1	0.371
400	9	entropy	100	0.370
400	9	gini	1	0.390
400	9	gini	100	0.383
400	15	entropy	1	0.413
400	15	entropy	100	0.404
400	15	gini	1	0.420
400	15	gini	100	0.415
4000	2	entropy	1	0.270
4000	2	entropy	100	0.270
4000	2	gini	1	0.274
4000	2	gini	100	0.274
4000	4	entropy	1	0.314
4000	4	entropy	100	0.313
4000	4	gini	1	0.313
4000	4	gini	100	0.312
4000	9	entropy	1	0.372
4000	9	entropy	100	0.368
4000	9	gini	1	0.389
4000	9	gini	100	0.383
4000	15	entropy	1	0.413
4000	15	entropy	100	0.404
4000	15	gini	1	**0.423**
4000	15	gini	100	0.415

Table 1: Validation metrics for all hyperparameter combinations

```
scheduler.DAGScheduler: ShuffleMapStage 686 (countByValue at MulticlassMetrics.scala:
ed in 171.929 s due to Job aborted due to stage failure: Task 1 in stage 686.0 failed
Lost task 1.3 in stage 686.0 (TID 1328, c114.local, executor 8): ExecutorLostFailure
e of the running tasks) Reason: Container killed by YARN for exceeding memory limits.
ory used. Consider boosting spark.yarn.executor.memoryOverhead or disabling yarn.node
use of YARN-4714.
```

Figure 1: Resource error message