

Amazon Reviews for Sentiment Analysis

Data Preparation

Data is downloaded from

<https://www.kaggle.com/bittlingmayer/amazonreviews>

It contains 2 files i.e. test and train in txt format.

Content :

Every Review is in format of :

“_label_x summaryText : Text”

Where x can be 1 or 2, 1 corresponds to negative and 2 corresponds to positive review.

```
__label__2 Great CD: My lovely Pat has one of the GREAT voices of her generation. I have listened to this CD fo  
other video game soundtracks. I must admit that one of the songs (Life-A Distant Promise) has brought tears to  
as hesitant due to unfavorable reviews and size of machines. I am weaning off my VHS collection, but don't want  
really did like this one... before it stopped working. The dvd player gave me problems probably after a year of  
school, and do not know all the requirements of admission, then this book may be a tremendous help. If you have  
label__1 A complete Bust: This game requires quicktime 5.0 to work...if you have a better version of quicktime  
s actually came in the relaxer kit. I tried it and could not beleive the difference it made with one use. I coul  
ill. I have sent numerous emails to the company - I can't actually find a phone number on their website - and I  
s to do! My four year old daughter is in love with the many tasks to complete in this game, including dressing  
money on something you may not be able to use. label__2 one of the last in the series to collect !: The magazi
```

So firstly we have create schema using case class.

```
case class AmzView(label : Int, view : String)
```

Then, we save paths of train and test data in String format.

```
val train_path = "/home/harsh/Desktop/amazon  
views/1305_800230_compressed_train.ft.txt.bz2/train.ft.txt"
```

```
val test_path = "/home/harsh/Desktop/amazon  
views/1305_800230_compressed_test.ft.txt.bz2/test.ft.txt"
```

Then, we create a method that load the data along with pre-processing (remove links, emojis, special characters, unnecessary spaces, converting text into vectors)

```
def pre_process(path:String) = {  
  
  val data = sc.textFile(path).map(attributes => AmzView(attributes(9),  
    attributes.substring(11,  
    attributes.length()).replace("\\", "").toLowerCase()  
  
      .replaceAll("\\n", "")  
  
      .replaceAll("rt\\s+", "")  
  
      .replaceAll("\\s+@\\w+", "")  
  
      .replaceAll("@\\w+", "")  
  
      .replaceAll("\\s+#\\w+", "")  
  
      .replaceAll("#\\w+", "")  
  
      .replaceAll("(?:https?|http?)://[\\w/%.-]+", "")  
  
      .replaceAll("(?:https?|http?)://[\\w/%.-]+\\s+", "")  
  
      .replaceAll("(?:https?|http?):/[\\w/%.-]+\\s+", "")  
  
      .replaceAll("(?:https?|http?):/[\\w/%.-]+", "")  
  
      .replaceAll("[^\\u0000-\\uFFFF]", "")  
  
      .replaceAll("(\\u00a9\\u00ae|[\\u2000-\\u3300]|\\ud83c[\\ud000-  
\\udfff]|\\ud83d[\\ud000-\\udfff]|\\ud83e[\\ud000-\\udfff])", "")  
  
      .trim()  
  
    ).toDF()  
  
  def lo(d:Int) :Int = {if(d==49){2} else{1}}
```

```
val lco = udf(lo _)
```

```
val p = data.select( (lco($"label")).alias("label"), $"view")
```

```
val tokenizer = new
```

```
Tokenizer().setInputCol("view").setOutputCol("words")
```

```
val wordsData = tokenizer.transform(p)
```

```
val hashingTF = new HashingTF()
```

```
.setInputCol("words").setOutputCol("rawFeatures").setNumFeatures(  
10000)
```

```
val featurizedData = hashingTF.transform(wordsData)
```

```
val idf = new
```

```
IDF().setInputCol("rawFeatures").setOutputCol("features")
```

```
val idfModel = idf.fit(featurizedData)
```

```
val rescaledData = idfModel.transform(featurizedData)
```

```
rescaledData
```

```
}
```

Loading data using data path and above method :

```
val training = pre_process(train_path)
```

```
val test = pre_process(test_path)
```

Model Selection

We tried Logistic Regression for classification.

```
val lr = new LogisticRegression()
```

```
.setMaxIter(10)
```

```
.setRegParam(0.01)
```

```
.setLabelCol("label")
```

```
.setElasticNetParam(0.5)
```

```
val model = lr.fit(training)
```

We evaluated model using MulticlassClassification Evaluator

```
val evaluator = new
```

```
MulticlassClassificationEvaluator()
```

```
.setLabelCol("label")
```

```
.setPredictionCol("prediction")
```

```
.setMetricName("accuracy")
```

```
val predict_train = model.transform(training)
```

```
val training_accuracy =
```

```
evaluator.evaluate(predict_train)
```

```
val predict_test = model.transform(test)
```

```
val test_accuracy =  
evaluator.evaluate(predict_test)
```

We got an accuracy of 84% using above model.

Model Tuning

We got a slight increase in accuracy (87%) when we used nlp stemming.

```
val document = new DocumentAssembler()  
  
    .setInputCol("view")  
  
    .setOutputCol("document")
```

```
val d1 = document.transform(training)
```

```
val token = new Tokenizer()  
  
    .setInputCols("document")  
  
    .setOutputCol("token")
```

```
val t1 = token.fit(d1).transform(d1)
```

```
val normalizer = new Normalizer()  
  
    .setInputCols("token")
```

```
.setOutputCol("normal")
```

```
val n1 = normalizer.fit(t1).transform(t1)
```

```
val stemmer = new Stemmer()
```

```
.setInputCols("normal")
```

```
.setOutputCol("stem")
```

```
val s1 = stemmer.transform(n1)
```

```
val finisher = new Finisher()
```

```
.setInputCols("stem")
```

```
.setOutputCols("final")
```

```
val f1 = finisher.transform(s1)
```

```
val hashingTF = new HashingTF()
```

```
.setInputCol("filtered").setOutputCol("rawFeatures").setNumFeatures(10000)
```

```
val featurizedData = hashingTF.transform(f1)
```

```
val idf = new  
IDF().setInputCol("rawFeatures").setOutputCol("features")
```

```
val idfModel = idf.fit(featurizedData)
```

```
val rescaledData = idfModel.transform(featurizedData)
```

```
val d2 = document.transform(testing)
```

```
val t2 = token.fit(d2).transform(d2)
```

```
val n2 = normalizer.fit(t2).transform(t2)
```

```
val s2 = stemmer.transform(n2)
```

```
val f2 = finisher.transform(s2)
```

```
val featurizedData2 = hashingTF.transform(f2)
```

```
val rescaledData2 = idfModel.transform(featurizedData2)
```

Conclusion

So, using nlp stemming and logistic Regression, we got accuracy at maximum.