Amazon Reviews for Sentiment Analysis

Data Preparation

Data is downloaded from https://www.kaggle.com/bittlingmayer/amazonreviews

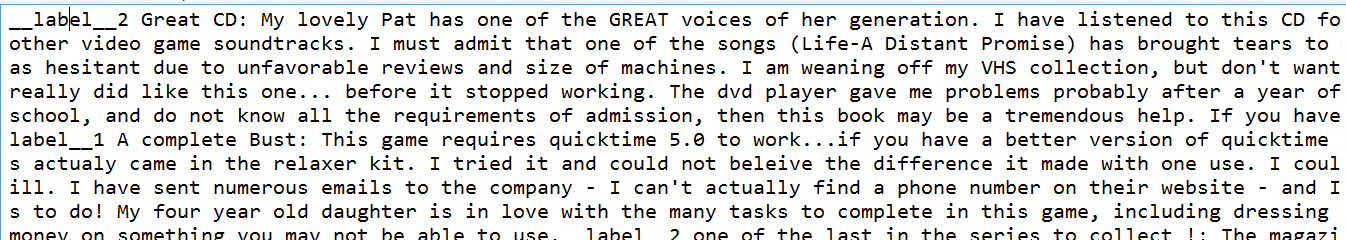
It contains 2 files i.e. test and train in txt format.

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.



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.