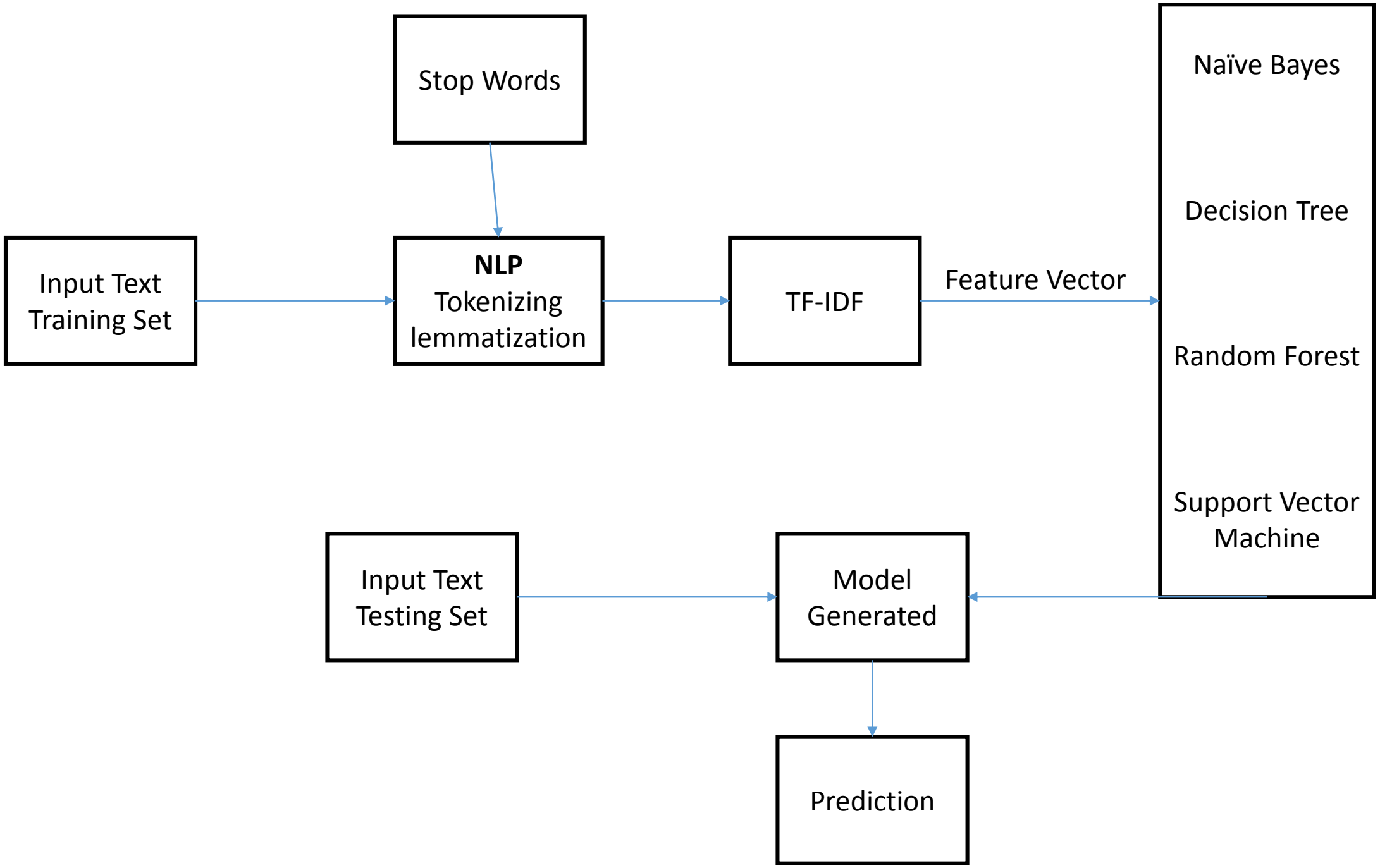
Spark MLLib

2


Stop Words
Input Text
Training Set
NLP
Tokenizing
lemmatization
TF-IDF
Feature Vector
Naïve Bayes
Decision Tree
Random Forest
Support Vector
Machine
Input Text
Testing Set
Model
Generated
Prediction

Natural Language Processing

- This is done using Core NLP from Stanford (http://nlp.stanford.edu/software/corenlp.shtml)

SparkW2V.scala × | SparkSVM.scala × | NLPUtils.scala × | SparkNaiveBayes.scala × | ModelEvaluation.scala ×

```scala
def tokenizeAndStem(text: String, stopWords: Set[String] ): Seq[String] = {
  val props = new Properties()
  props.put("annotators", "tokenize, ssplit, pos, lemma")

  val pipeline = new StanfordCoreNLP(props)
  val doc = new Annotation(text)

  pipeline.annotate(doc)

  val lemmas = new ArrayBuffer[String]()
  val sentences = doc.get(classOf[SentencesAnnotation])
  for (sentence <- sentences;
       token <- sentence.get(classOf[TokensAnnotation])) {
    val lemma = token.get(classOf[LemmaAnnotation])
    if (lemma.length > 2 && !stopWords.contains(lemma)
      && isOnlyLetters(lemma)) {
      lemmas += lemma.toLowerCase
    }
  }
  println(lemmas);
  lemmas
```

Stop Word Removal

Lemmatization

Natural Language Processing : Input

If I install X11R5 with backward compatibility for motif, will motif |>1.2 clients work on the X11R5 servers?, It works for me., I've run Motif 1.1.3,1.1.4,1.1.5,1.2,1.2.1, and 1.2.2 on

Natural Language Processing : Tokenization

If
I
install
X11R5
with
backward
compatibility
for
motif
,
will

motif
|
>
1.2
clients
work
on
the
X11R5
servers
?

It
works
for
me
.
I
've
run
Motif
1.1.3,1.1.4,1.1.5,1.2,1.2.1

Natural Language Processing : Lemmatization and Stop Words

ArrayBuffer(daniel, mccoy, subject, motifbc, organization, line, article, ray, stell, write, install, backward, compatibility, motif, will, motif, client, work, server, work, run, motif, server, motifbc, define, daniel, mccoy, space, nasa, mail, code, tel, space, center, fax, houston, texas, future)

Multiclass Classification

Naïve Bayes

SparkNaiveBayes.scala

```scala
// tokenize, stem,
val training = sc.wholeTextFiles("data/training/*")
  .map(rawText => createLabeledDocument(rawText, labelToNumeric, stopWords))
val test = sc.wholeTextFiles("data/test/*")
  .map(rawText => createLabeledDocument(rawText,labelToNumeric, stopWords))

//create features
val X_train = tfidfTransformer(training)
val X_test = tfidfTransformer(test)

//Train / Predict
val model = NaiveBayes.train(X_train,lambda = 1.0)
val predictionAndLabel = X_test.map(x => (model.predict(x.features), x.label) )
predictionAndLabel.foreach(x=> print(x))
val accuracy = 1.0 *  predictionAndLabel.filter(x => x._1 == x._2).count() / X_test.count()

println("*************Accuracy Report:**********************")
println(accuracy)
```

Maven Projects
SBT
Ant Build

Decision Tree

SparkNaiveBayes.scala × SparkDecisionTree.scala ×

```scala
  .map(rawText => createLabeledDocument(rawText,labelToNumeric, stopWords))

//create features
val X_train = tfidfTransformer(training)
val X_test = tfidfTransformer(test)
val numClasses = 2
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val maxDepth = 5
val maxBins = 32
val model = DecisionTree.trainClassifier(X_train, numClasses, categoricalFeaturesInfo,
  impurity, maxDepth, maxBins)
val predictionAndLabel = X_test.map(x => (model.predict(x.features), x.label))
val accuracy = 1.0 *  predictionAndLabel.filter(x => x._1 == x._2).count() / X_test.count()

println("*************Accuracy Report:**********************")
println(accuracy)
evaluateModel(predictionAndLabel,"Decision Tree Results")
```

Maven Projects SBT Ant Build

Random Forest

SparkNaiveBayes.scala × SparkDecisionTree.scala × SparkRandomForest.scala ×

```scala
val training = sc.wholeTextFiles("data/training/*")
  .map(rawText => createLabeledDocument(rawText, labelToNumeric, stopWords))
val test = sc.wholeTextFiles("data/test/*")
  .map(rawText => createLabeledDocument(rawText,labelToNumeric, stopWords))

//create features
val X_train = tfidfTransformer(training)
val X_test = tfidfTransformer(test)
val numClasses = 2
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val maxDepth = 5
val maxBins = 32
val numTrees=3
val featureSubsetStartegy="auto"
val model = RandomForest.trainClassifier(X_train, numClasses, categoricalFeaturesInfo,numTrees,featureSubsetStartegy,impurity, maxDepth,
val predictionAndLabel = X_test.map(x => (model.predict(x.features), x.label))
val accuracy = 1.0 *  predictionAndLabel.filter(x => x._1 == x._2).count() / X_test.count()
```

Maven Projects SBT Ant Build

Binary Classifier

Input for SVM

https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_libsvm_data.txt

```
0 128:51 129:159 130:253 131:159 132:50 155:48 156:238 157:252 158:252 159
1 159:124 160:253 161:255 162:63 186:96 187:244 188:251 189:253 190:62 214
1 125:145 126:255 127:211 128:31 152:32 153:237 154:253 155:252 156:71 180
1 153:5 154:63 155:197 181:20 182:254 183:230 184:24 209:20 210:254 211:254
1 152:1 153:168 154:242 155:28 180:10 181:228 182:254 183:100 209:190 210:
0 130:64 131:253 132:255 133:63 157:96 158:205 159:251 160:253 161:205 162
1 159:121 160:254 161:136 186:13 187:230 188:253 189:248 190:99 213:4 214:
1 100:166 101:222 102:55 128:197 129:254 130:218 131:5 155:29 156:249 157:
0 155:53 156:255 157:253 158:253 159:253 160:124 183:180 184:253 185:251 18
0 128:73 129:253 130:227 131:73 132:21 156:73 157:251 158:251 159:251 160:
1 155:178 156:255 157:105 182:6 183:188 184:253 185:216 186:14 210:14 211:
0 154:46 155:105 156:254 157:254 158:254 159:254 160:255 161:239 162:41 186
0 152:56 153:105 154:220 155:254 156:63 178:18 179:166 180:233 181:253 182
1 130:7 131:176 132:254 133:224 158:51 159:253 160:253 161:223 185:4 186:17
0 155:21 156:176 157:253 158:253 159:124 182:105 183:176 184:251 185:251 18
1 151:68 152:45 153:131 154:131 155:131 156:101 157:68 158:92 159:44 187:19
0 125:29 126:170 127:255 128:255 129:141 151:29 152:198 153:255 154:255 15
```

Support Vector Machine

SparkNaiveBayes.scala × | SparkDecisionTree.scala × | SparkRandomForest.scala × | SparkSVM.scala ×

```scala
// Load training data in LIBSVM format.
val data = MLUtils.loadLibSVMFile(sc, "sample_libsvm_data.txt")

// Split data into training (60%) and test (40%).
val splits = data.randomSplit(Array(0.6, 0.4), seed = 11L)
val training = splits(0).cache()
val test = splits(1)

// Run training algorithm to build the model
val numIterations = 100
val model = SVMWithSGD.train(training, numIterations)

// Clear the default threshold.
model.clearThreshold()

// Compute raw scores on the test set.
val scoreAndLabels = test.map { point =>
  val score = model.predict(point.features)
  (score, point.label)
```

Maven Projects | SBT | Ant Build

Confusion Matrix

SparkNaiveBayes

```
15/07/14 14:43:50 INFO TaskSetManager: Finished task 1.0 in stage 10.0 (TID 21) in 11 ms on localhost (2/2)
15/07/14 14:43:50 INFO DAGScheduler: ResultStage 10 (collectAsMap at MulticlassMetrics.scala:60) finished in 0
15/07/14 14:43:50 INFO TaskSchedulerImpl: Removed TaskSet 10.0, whose tasks have all completed, from pool
15/07/14 14:43:50 INFO DAGScheduler: Job 7 finished: collectAsMap at MulticlassMetrics.scala:60, took 2.123345
Naive Bayes Results
 |=================== Confusion matrix ==========================
316.0  0.0    0.0    2.0    0.0    1.0
15.0   308.0  6.0    12.0   10.0   38.0
21.0   64.0   160.0  56.0   14.0   79.0
2.0    12.0   13.0   315.0  34.0   16.0
3.0    20.0   6.0    31.0   315.0  10.0
1.0    40.0   5.0    10.0   3.0    336.0
```

References

- https://github.com/brokendata/Spark20NewsGroup
- https://github.com/logicalguess/tf-idf-spark-and-python
- https://spark.apache.org/docs/latest/mllib-guide.html