heavy lifting through distributed computing and machine learning by Adam Hicks

Sorting PDFs with Spark

First things First

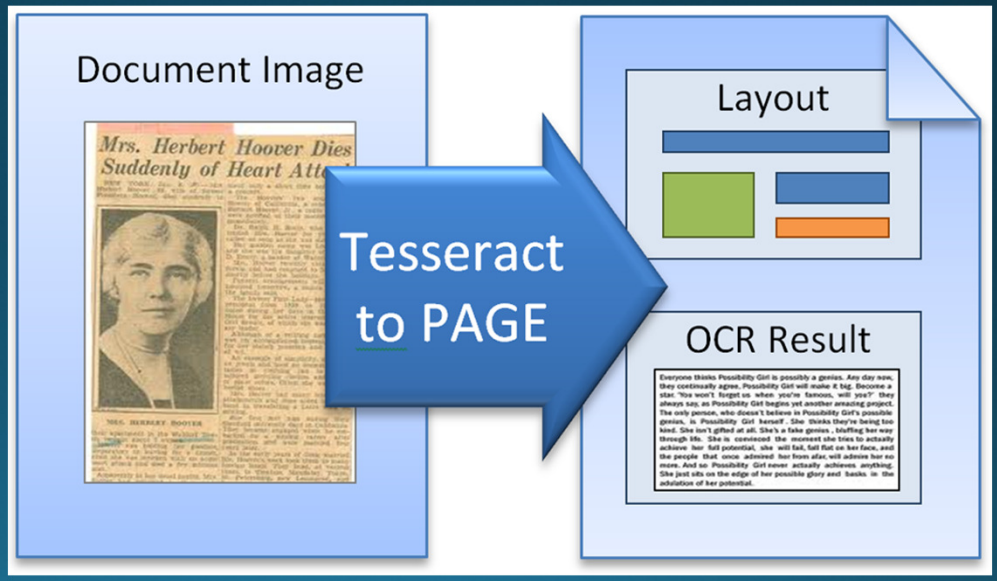
- Who is this guy?
  - An Engineer
  - A Web Dev
  - A DevOps Devout
  - A Pythonist
- Who does he work for?
  - SquareTwo Financial

The Business Case

- Highly regulated industry
  - New consumer protections = new compliance requests
- Immediate need for scrutinized, automated media processing
- Binders of data!

The Solution

- At a very high level, two important stages:
  - Get the text out of PDFs
  - Classify the corpus

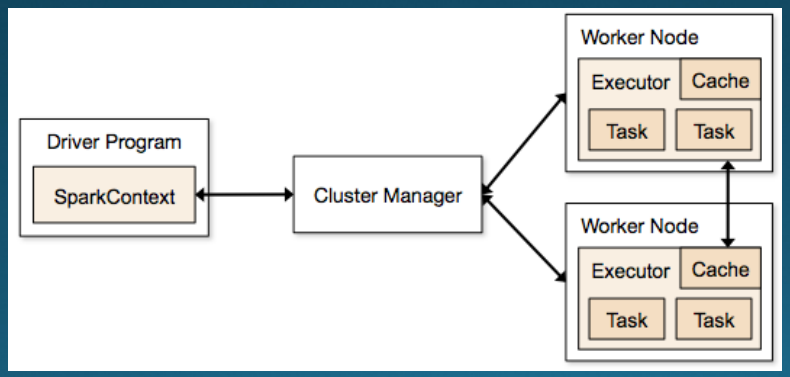


On the Road to Spark

- Important lessons were learned
  - PDFs can be weird
    - Most OCR solutions are all-or-none
    - Valuable information exists in inline images
  - Machine Learning a la NLP the way to go for classification
  - Scaling is HARD

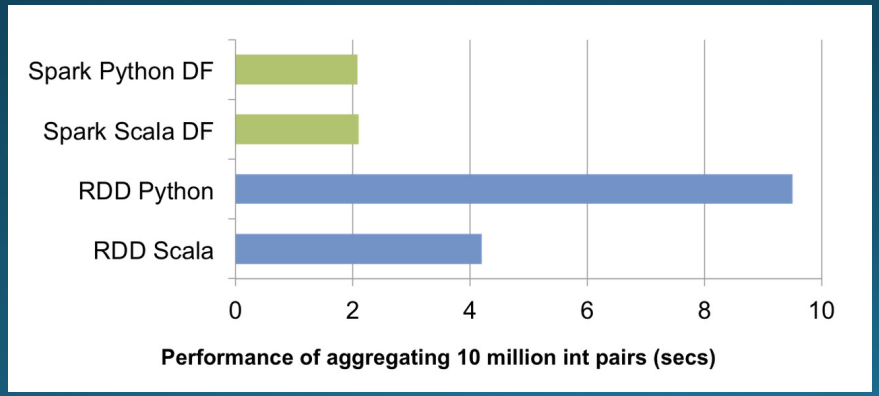
Salvation Found

- Apache Tika is Java, so plays nicely with Scala
- Spark MlLib and ML make it easy to build
- Distributed computing with YARN is EASY



Gettin' 'er dun

- Spark DataFrames win the day
  - UDFs parallelize work for you
  - Fast
  - Pipelines are smooth and easy to build



Tika as a UDF

```
// Function literal using Apache Tika and Tesseract to extract text from a byte array (doc)
val extractInfo: (Array[Byte] => String) = (fp: Array[Byte]) => {

  try {
    val parser:Parser = new AutoDetectParser()
    val handler:BodyContentHandler = new BodyContentHandler(Integer.MAX_VALUE)
    val config:TesseractOCRConfig  = new TesseractOCRConfig()
    val pdfConfig:PDFParserConfig = new PDFParserConfig()
    pdfConfig.setExtractInlineImages(true)

    val inputstream:InputStream = new ByteArrayInputStream(fp)

    val metadata:Metadata = new  Metadata()
    val parseContext:ParseContext = new ParseContext()
    parseContext.set(classOf[TesseractOCRConfig], config)
    parseContext.set(classOf[PDFParserConfig], pdfConfig)
    parseContext.set(classOf[Parser], parser)
    parser.parse(inputstream, handler, metadata, parseContext)
    handler.toString
    } catch {
      case e: TikaException => "TIKAEXCEPTION"
      case e: SAXException => "SAXEXCEPTION"
    }
}


val extract_udf = udf(extractInfo)


// Create a dataframe that replaces the base64 encoded string document with a byte array (binary file)
val df2 = dfFill.withColumn("unbased_media", unbase64($"media_file")).drop("media_file")

// Another dataframe that replaces the byte array doc with the extracted text of the doc
val dfRenamed = df2.withColumn("media_corpus", extract_udf(col("unbased_media"))).drop("unbased_media")
```

Building a pipeline

```
// Send Label and Sentence to a DataFrame
val sentenceData = labelAndDepunct.toDF("label", "sentence")

// Tokenize the sentence data
val tokenizer = new RegexTokenizer().setInputCol("sentence").setOutputCol("words")

val hashingTF = new HashingTF()
  .setInputCol("words").setOutputCol("rawFeatures").setNumFeatures(50000)

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

// Train a NaiveBayes model.
val nb = new NaiveBayes().setLabelCol("label").setFeaturesCol("features")

val pipeline = new Pipeline().setStages(Array(tokenizer, hashingTF, idf, nb))

val model = pipeline.fit(sentenceData)
```

Using the Model

```
// Load the model. Tokenizer -> Tf-Idf -> Naive Bayes
val model = PipelineModel.load("hdfs://HOSTNAME:PORTNO/path/to/model-v1")

val with_predictions = model.transform(withoutPunct)
```

Just That Easy

- Questions?

Adam Hicks
thomas.adam.hicks@gmail.com
https://github.com/tadamhicks