Cascalog & Clojure

Making the elephant move faster

SOKRATI

Prologue

Expedition [ek-spi-dish-uh n]

“An organized journey or voyage for specific purpose”

Late 15th C. latin ‘expeditus’ which means,

- free the feet from fetters
- *liberate* from difficulties

‘ … infrastructure components such as Hadoop have stabilized ….

… challenges have moved higher up the stack ’

Analytics as Expedition

There is always a specific purpose!

Supervised vs. unsupervised is details

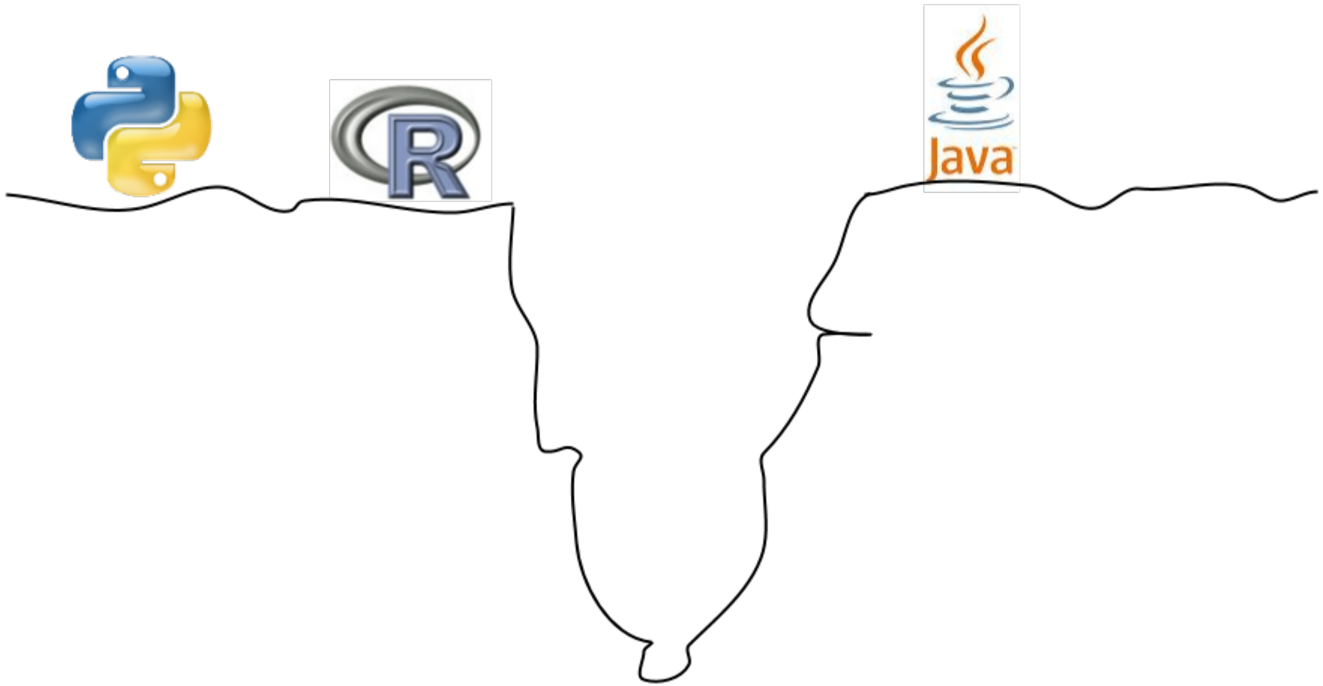
Confusing means and ends?

Cost of expedition outweighs insights

Purpose of computing is insight, not numbers!

Richard Hamming, (1995 - 1998)

Practical ML as Expedition



Real World Example

“ Why are customers churning / not re-filling? ”

# of refills in last 90 days ?	# of refills in last 30 days ?
changed plan in last 30 days ?	changed plan in last 90 days ?
7 day average burn rate ?	last 30 day average burn rate ?
….	….

Represent => Model => Predict/Use

Representation Problem

Feature Engineering, Exploratory Analysis, Data Preparation

Modeling Problem

Fitting, Cross Validations, Tuning

Generalization Problem

Deployment, Standardization

‘ MapReduce provides a programming model that abstracts the problem from disk reads and writes, transforming it into a computation over sets of keys and values. ’

Tom White, Hadoop the Definitive Guide, Page 4

```
package org.myorg;
…..
public class WordCount {

public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
       String line = value.toString();
       StringTokenizer tokenizer = new StringTokenizer(line);
       while (tokenizer.hasMoreTokens()) {
           word.set(tokenizer.nextToken());
           context.write(word, one);
       }
   }
}
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
   public void reduce(Text key, Iterable<IntWritable> values, Context context)
     throws IOException, InterruptedException {
       int sum = 0;
       for (IntWritable val : values) {
           sum += val.get();
       }
       context.write(key, new IntWritable(sum));
   }
}
public static void main(String[] args) throws Exception {
   Configuration conf = new Configuration();

       Job job = new Job(conf, "wordcount");

   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(IntWritable.class);

   job.setMapperClass(Map.class);
   job.setReducerClass(Reduce.class);

   job.setInputFormatClass(TextInputFormat.class);
   job.setOutputFormatClass(TextOutputFormat.class);

   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));

   job.waitForCompletion(true);
}

}
```

Seriously?

‘When someone says "I want a programming language in which I need only say what I wish done," give him a lollipop.’

‘A programming language is low level when its programs require attention to the irrelevant.’

Alan J Perlis, first winner of Turing award

Clojure 101 & Cascalog

```
(defn hello
  [input-string]
    (println
      (str "hello, " input-string) ) )
```

What’s the Point ?

count empty not-empty into conj

distinct? empty? every? not-every? some not-any?

sequential? associative? sorted? counted? reversible?

first nth peek .indexOf .lastIndexOf

cons conj rest pop

join select project union difference intersection

assoc assoc-in dissoc merge merge-with select-keys update-in

hash-map array-map zipmap sorted-map sorted-map-by bean
frequencies group-by

‘It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures.’

Alan J Perlis

ID	Timestamp	Action
6010282163410	1395241769	“Did A”
6010282163410	1395922229	“Did B”
6010245163456	1395413781	“Did X”
6010282163410	1395997829	“Did C”
6010245163456	1402588821	“Did Y”
6010245163456	1399885829	“Did D”
….	….	….

Set of tuples!

Total Actions?

```
(defbufferfn count-actions
   [tuples]
     [ ( count tuples ) ] )

(defn main
  [ ]
  (let [src-data (hfs-delimited "/path/to/input") :classes [String Long String]
         out-data (hfs-delimited "/path/to/output" ) ]
    (?<- out-data
      [?userId ?total-actions]
        (src-data :> ?userId ?time ?action)
        (count-actions :< ?action :> ?total-actions) ) ) )
```

Most Recent Action?

```
(defbufferfn recent-action
   [tuples]
     [ ( first tuples ) ] )

(defn main
  [ ]
  (let [src-data (hfs-delimited "/path/to/input") :classes [String Long String]
         out-data (hfs-delimited "/path/to/output" ) ]
    (?<- out-data
      [?userId ?recent-act]
        (src-data :> ?userId ?time ?action)
        (:sort ?time)
        (:reverse true)
        (recent-action :< ?action :> ?recent-act ) ) ) )
```

n Most Recent Actions?

```
(defbufferfn recent-actions
   [tuples]
     (take 5 tuples ) )

(defn main
  [ ]
  (let [src-data (hfs-delimited "/path/to/input") :classes [String Long String]
        out-data (hfs-delimited "/path/to/output" ) ]
    (?<- out-data
      [?userId ?recent-acts]
        (src-data :> ?userId ?time ?action)
        (:sort ?time)
        (:reverse true)
        (recent-actions :< ?action :> ?recent-acts ) ) ) )
```

Mean Time Between Actions?

```
(defn avg
  [coll]
    (/ (reduce + 0 coll)
       (count coll) )

(defbufferfn mtba
   [tuples]
    (let [pairs (partition 2 1 tuples) ]
      (avg
         (map
           #(- (second %1) (first %1) )
           pairs) ) ) )
```

What is gained ?

- Familiarity
- Convolution of logic, execution plan…
- Focus on how to compute

- Simplicity
- Logic in small, testable pure functions
- Focus on what to compute

@ SOKRATI

- ML pipelines
  - ‘Templatized’ data pipelines
  - Keep adding new feature through simple functions
- Quick hypothesis testing
  - Do customers with behaviour X do Y

Vs. DSL ‘X’ ?

- How much Impedance
  - From ‘solution in the head’ to ‘solution running on cluster’?
- Moving farther from word count
  - Does the complexity increase exponentially?
- Is it a real programming language?
  - Does it allow arbitrary user defined functions?

Vs. SQL on Hadoop?

- Need data processing pipelines, not ad-hoc queries
- Similarities ?
  - Map-reduce Vs. Relational algebra
  - Unstructured data Vs. Structured Data
  - Data locality Vs. Normalization
  - Extrinsic keys Vs. Intrinsic keys
- Love of familiarity not data model!

SOKRATI Experience

- Complex ML pipeline
  - 3-4 chained jobs
  - 2-3 thousand LOC
- Simple, intuitive ML pipeline
  - 10-12 functions
  - ~ 500 LOC

Creating 100 features for ML with unit tested functions takes only couple of man days!

Liberate from difficulties

You can reach a point with Lisp where, …
... you are able to write only code that matters. ”

Rich Hickey, Creator of Clojure

Questions ?

SOKRATI