



Lucidworks

BM25 is so Yesterday

Modern Techniques for Better Search Relevance in Solr

Grant Ingersoll

CTO Lucidworks

Lucene/Solr/Mahout Committer




iPad case





iPad case



"ipad
accessory"~3
OR "ipad
case"~5





1.

```
{  
  "shortDescription":["Designed for Apple® iPad® 2; polyurethane construction; converts to a stand; Cleveland Browns design"],  
  "id":"2789054",  
  "name":"Tribeca - Cleveland Browns Folio Case for Apple® iPad® 2"},
```

15.

```
{  
  "shortDescription":["Compatible with Apple® iPad® 2; ABS material; slim, lightweight design; team design"],  
  "id":"2789946",  
  "name":"Tribeca - Minnesota Vikings Hard Shell Case for Apple® iPad® 2"},
```







So, what do you do?



Index Time:

```
if (doc.name.contains("Vikings")){  
    doc.boost = 100  
}
```

OR

Query Time:

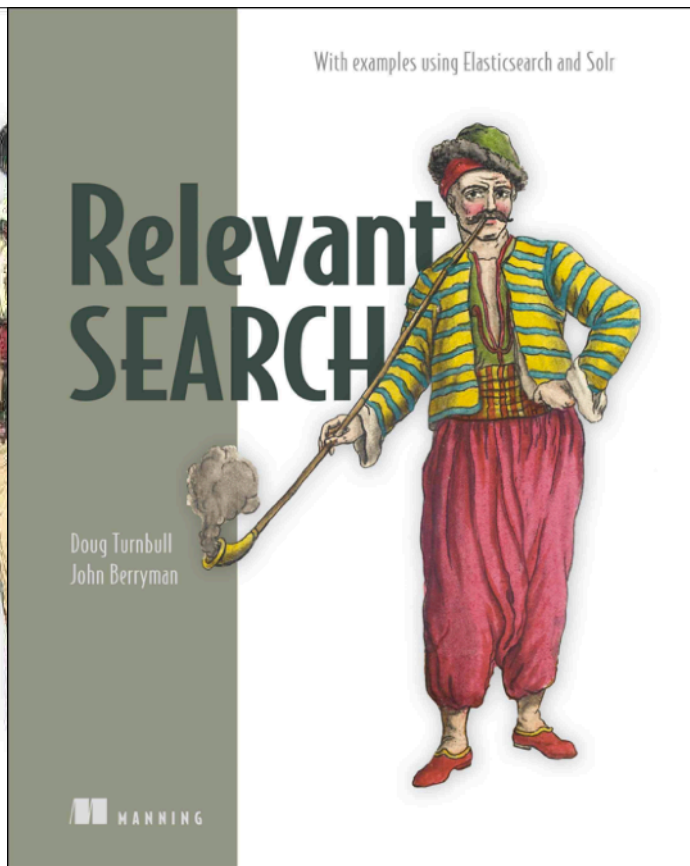
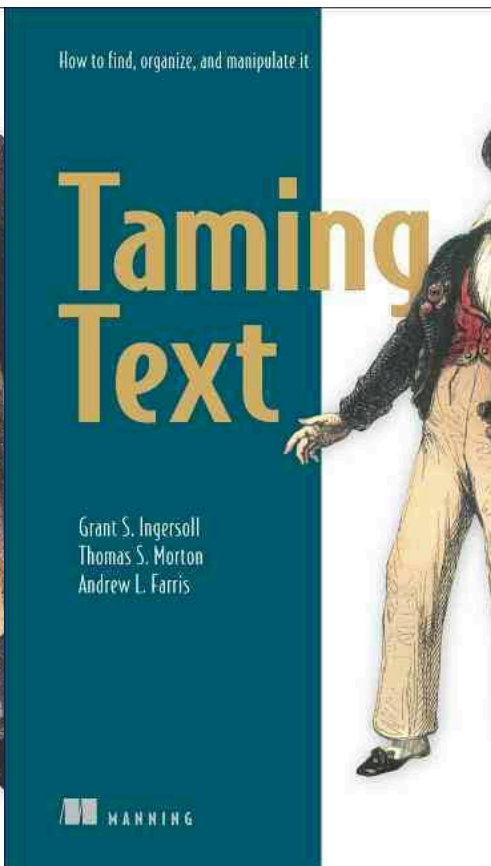
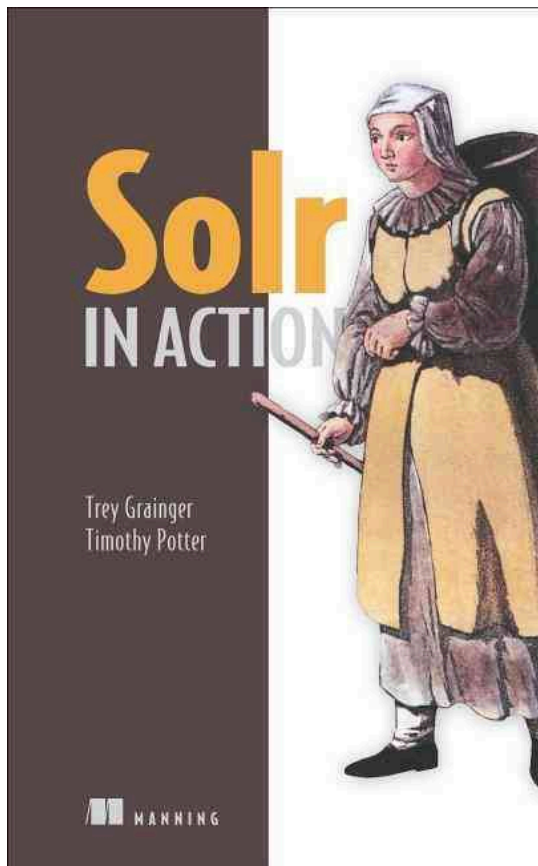
```
q:(MAIN QUERY) OR (name:Vikings)^Y
```



```

"2789054":{
  "match":true,
  "value":13.144117,
  "description":"sum of:",
  "details":{
    "match":true,
    "value":13.144117,
    "description":"weight(_text_:\`ipad case\`-5 in 9267) [SchemaSimilarity], result of:",
    "details":{
      "match":true,
      "value":13.144117,
      "description":"score(doc=9267,freq=2.4 = phraseFreq=2.4\n), product of:",
      "details":{
        "match":true,
        "value":10.070848,
        "description":"idf(), sum of:",
        "details":{
          "match":true,
          "value":6.160664,
          "description":"idf, computed as log(1 + (docCount - docFreq + 0.5) / (docFreq + 0.5)) from:",
          "details":{
            "match":true,
            "value":2691.0,
            "description":"docFreq"},
            {
              "match":true,
              "value":1275077.0,
              "description":"docCount"}
          }
        }
      }
    }
  }
}
{
  "match":true,
  "value":3.910184,
  "description":"idf, computed as log(1 + (docCount - docFreq + 0.5) / (docFreq + 0.5)) from:",
  "details":{
    "match":true,
    "value":25548.0,
    "description":"docFreq"},
    {
      "match":true,
      "value":1275077.0,
      "description":"docCount"}
  }
}
{
  "match":true,
  "value":1.3051648,
  "description":"tfNorm, computed as (freq * (k1 + 1)) / (freq + k1 * (1 - b + b * fieldLength / avgFieldLength)) from:",
  "details":{
    "match":true,
    "value":2.4,
    "description":"phraseFreq=2.4"},
    {
      "match":true,
      "value":1.2,
      "description":"parameter k1"},
    {
      "match":true,
      "value":0.75,
      "description":"parameter b"},
    {
      "match":true,
      "value":304.42984,
      "description":"avgFieldLength"},
    {
      "match":true,
      "value":455.1111,
      "description":"fieldLength"}
  }
}

```



TF*IDF

- Term Frequency: “How well a term describes a document?”
 - Measure: how often a term occurs per document
- Inverse Document Frequency: “How important is a term overall?”
 - Measure: how rare the term is across all documents



BM25 (aka Okapi)

Score(q, d) =

$$\sum_{t \text{ in } q} \text{idf}(t) \cdot (\text{tf}(t \text{ in } d) \cdot (k + 1)) / (\text{tf}(t \text{ in } d) + k \cdot (1 - b + b \cdot |d| / \text{avgdl}))$$

Where:

t = term; d = document; q = query; i = index

tf(t in d) = numTermOccurrencesInDocument $\frac{1}{2}$

idf(t) = $1 + \log(\text{numDocs} / (\text{docFreq} + 1))$

|d| = $\sum 1$

t in d

avgdl = $(\sum |d|) / (\sum 1)$

d in i

d in i

k = Free parameter. Usually ~1.2 to 2.0. Increases term frequency saturation point.

b = Free parameter. Usually ~0.75. Increases impact of document normalization.



Lather, Rinse, Repeat







WWGD?



Measure, Measure, Measure

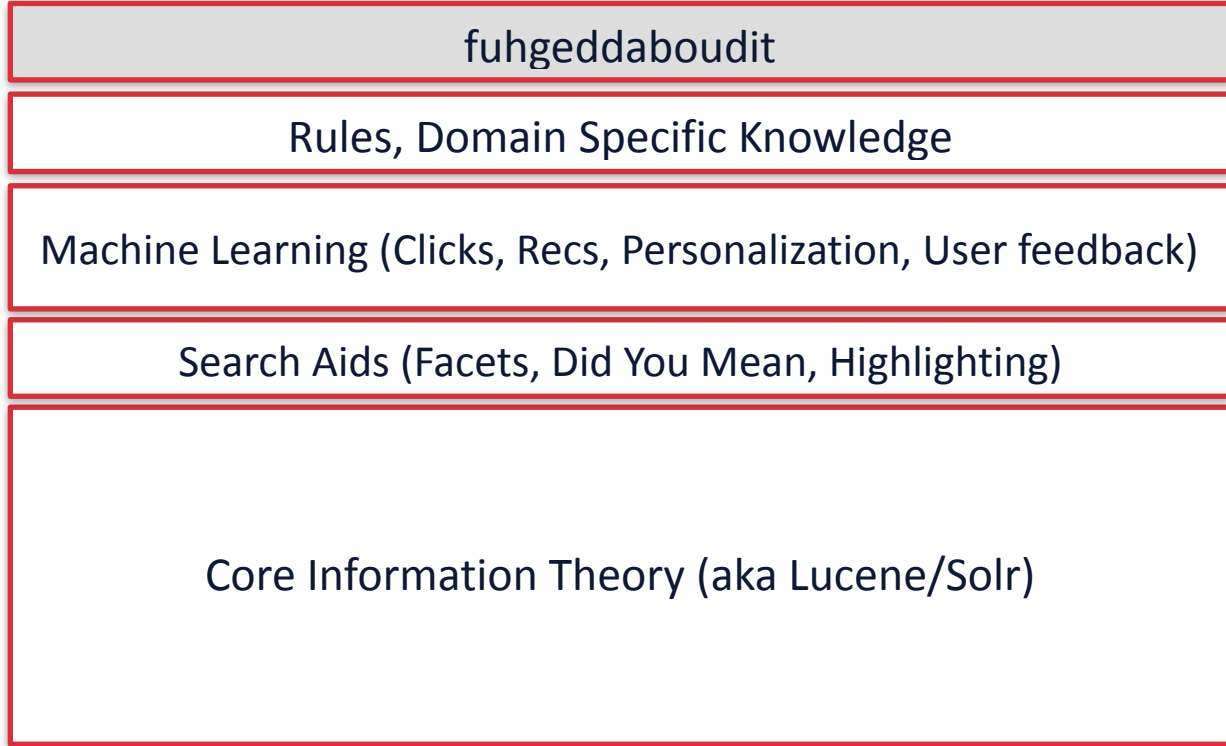
- Capture and log pretty much everything
 - Searches, Time on page/1st click, What was not chosen, etc.
- Precision — Of those shown, what's relevant?
- Recall — Of all that's relevant, what was found?
- NDCG — Account for position



Magic



Guessing



Content



Core Solr capabilities: text matching, faceting, spell checking, highlighting

Business Rules for content: landing pages, boost/block, promotions, etc.

Collaboration



Leverage collective intelligence to predict what users will do based on historical, aggregated data

Recommenders, Popularity, Search Paths

Context



Who are you? Where are you? What have you done previously?

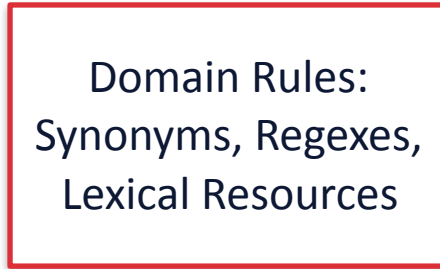
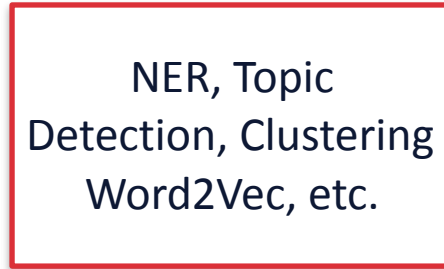
User/Market Segmentation, Roles, Security, Personalization



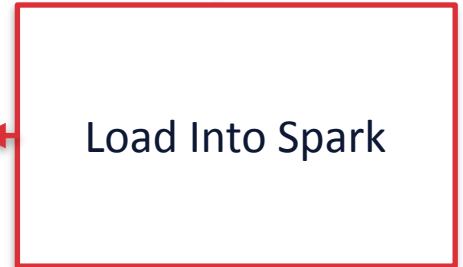
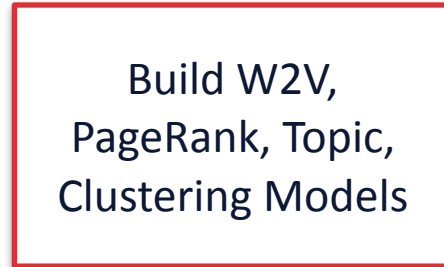
But What About the Real World? Indexing Edition



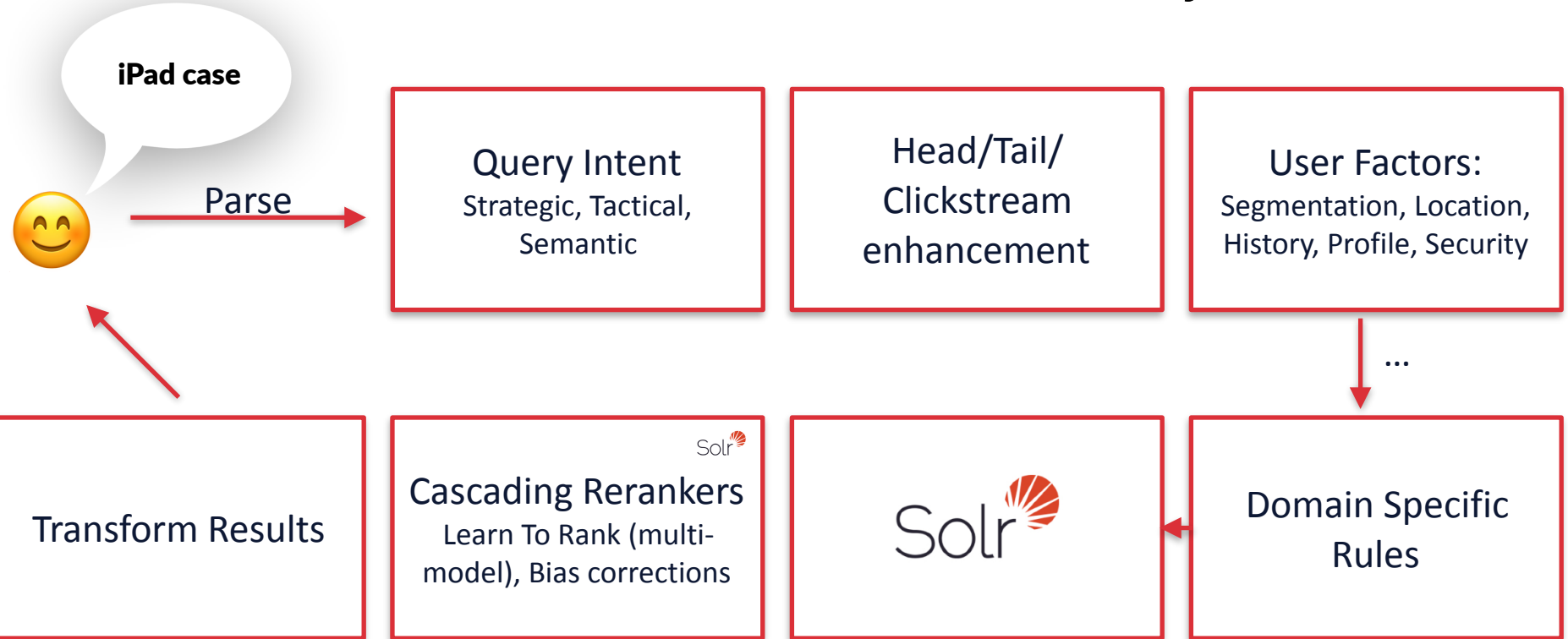
Extraction →



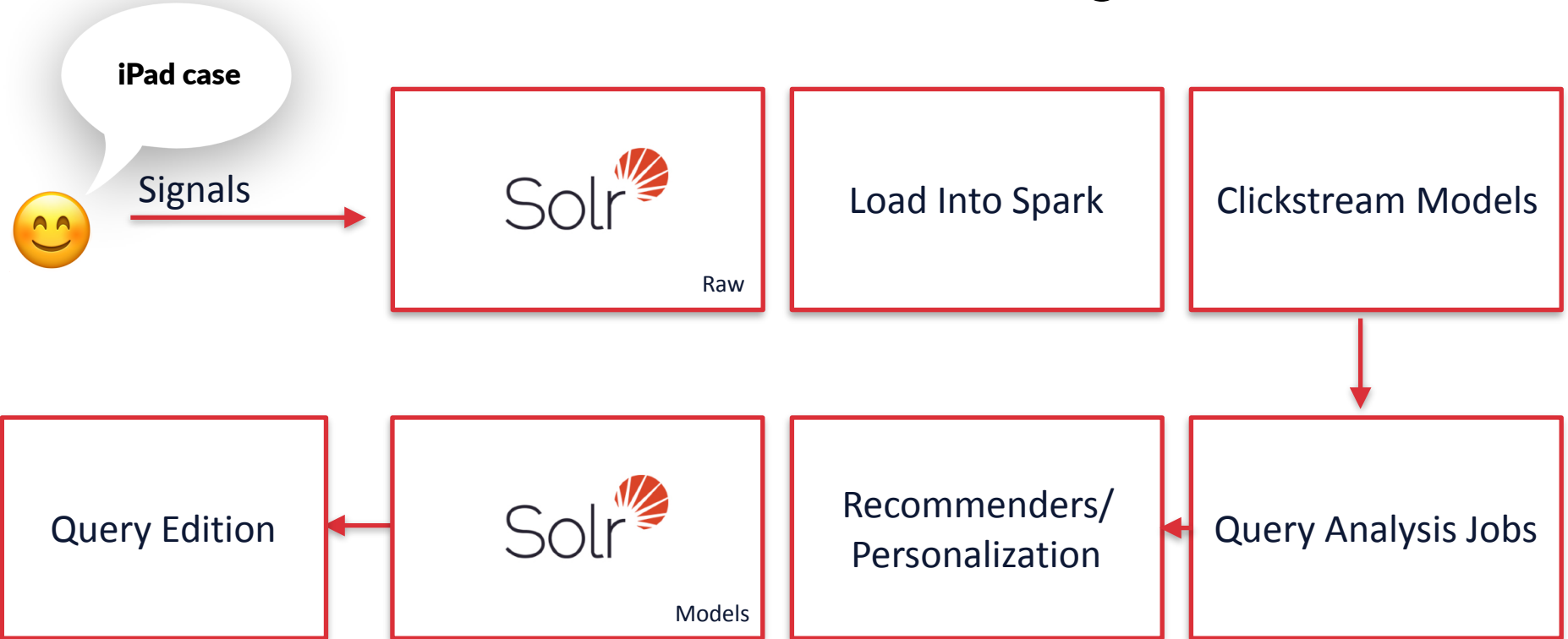
↓ Offline



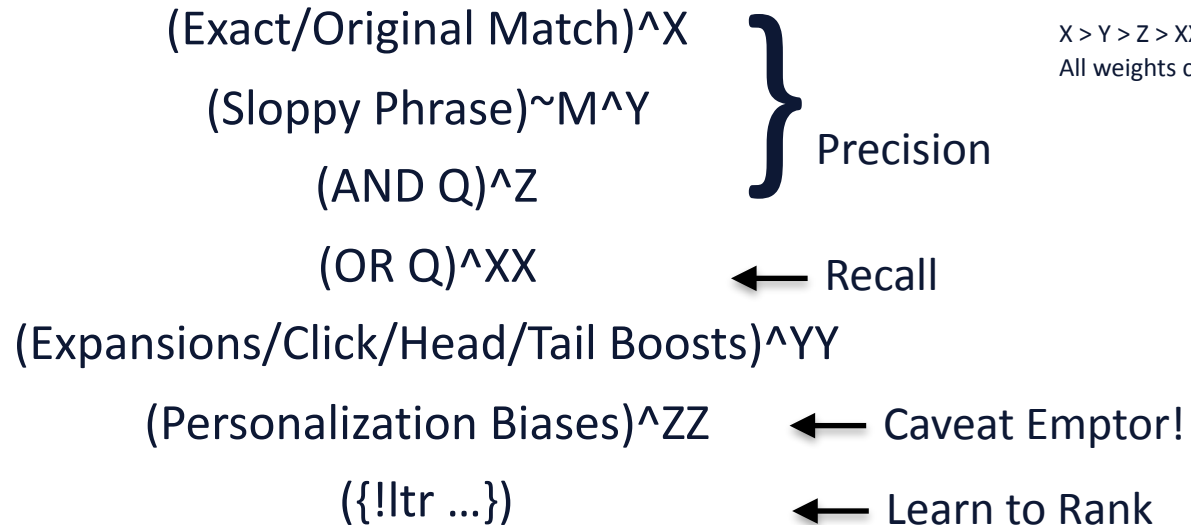
But What About the Real World? Query Edition



But What About the Real World? Signals Edition



The Perfect(?!?) Query* YMMV!



Filters+Options: security, rules, hard preferences, categories



* Note: there are a lot of variations on this. edismax handles most

Experimentation, Not Editorialization

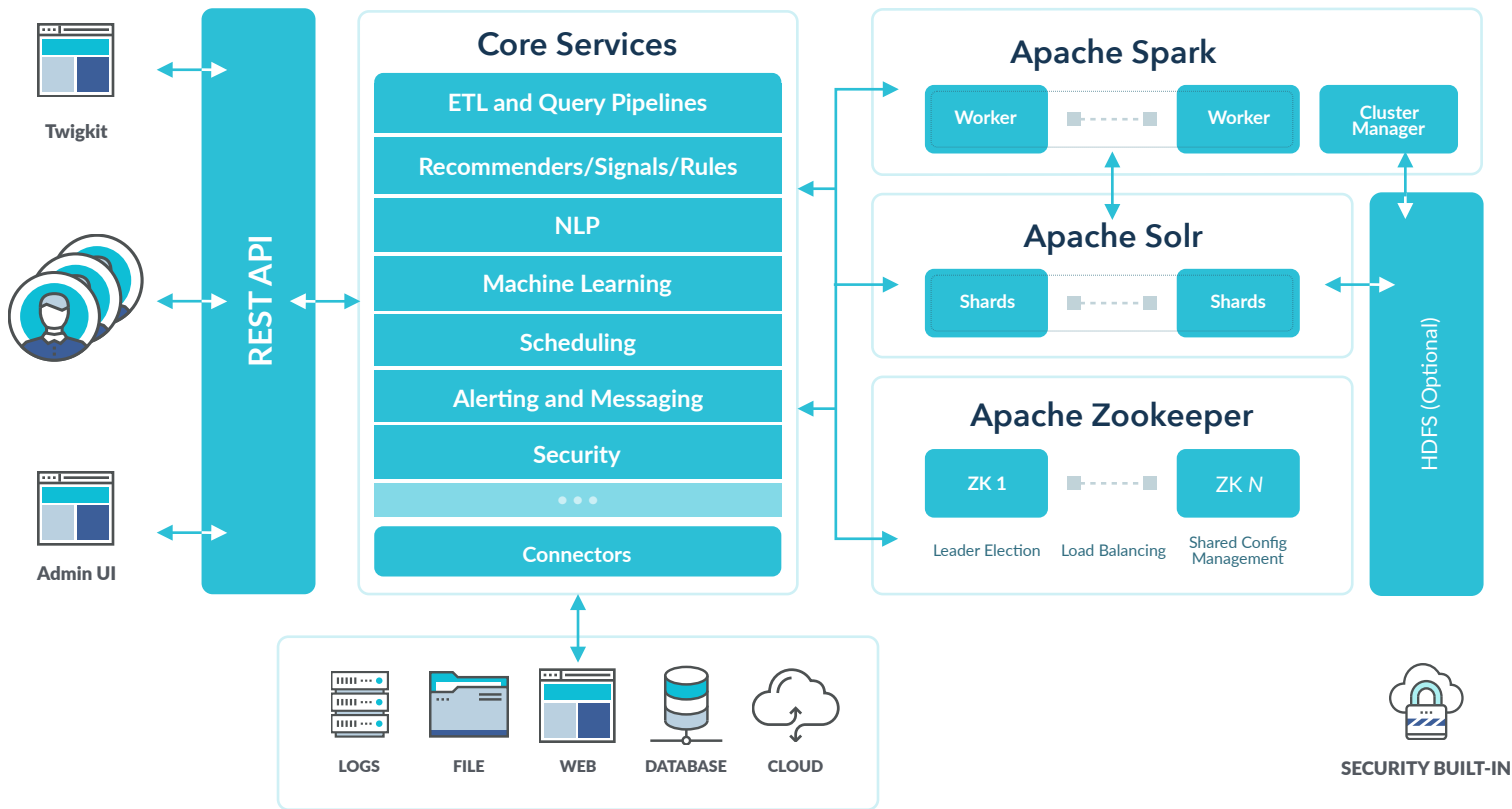
- Don't take my word for it, experiment!
- Good primer:
 - <http://www.slideshare.net/InfoQ/online-controlled-experiments-introduction-insights-scaling-and-humbling-statistics>
- Rules are fine, as long as they are contained, have a lifespan and are measured for effectiveness



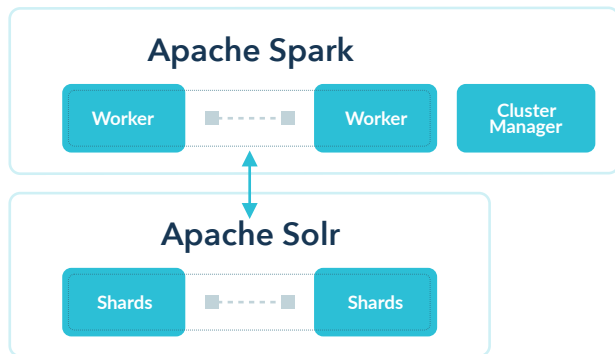
Show Us Already, Will You!



Fusion Architecture



Key Features



- Solr:
 - Extensive Text Ranking Features
 - Similarity Models
 - Function Queries
 - Boost/Block
 - Pluggable Reranker
 - Learn to Rank contrib
 - Multi-tenant
- Spark
 - SparkML (Random Forests, Regression, etc.)
 - Large scale, distributed compute



Demo Details

- Best Buy Kaggle Competition Data Set
 - Product Catalog: ~1.3M
 - Signals: 1 month of query, document logs
- Fusion 3.1 Preview + Recommenders (sampled dataset) + Rules (open source add-on module) + Solr LTR contrib
- Twigkit UI (<http://twigkit.com>)



Resources

- <http://lucidworks.com>
- <http://lucene.apache.org/solr>
- <http://spark.apache.org/>
- <https://github.com/lucidworks/spark-solr>
- <https://cwiki.apache.org/confluence/display/solr/Learning+To+Rank>
- Bloomberg talk on LTR <https://www.youtube.com/watch?v=M7BKwJoh96s>

