Searching for Analytics

A Funny Thing Happened on the way to Full Text Search: I Shook my Search Engine and Analytics Fell Out!
Search engines are not just for text anymore.

Apache Solr has become a powerful Business Intelligence and Analytics tool.
OVERVIEW

• Faceting & Statistics
OVERVIEW

• Faceting & Statistics
• Streaming Expressions
OVERVIEW

- Faceting & Statistics
- Streaming Expressions
- Parallel SQL
OVERVIEW

- Faceting & Statistics
- Streaming Expressions
- Parallel SQL
- Graph Traversal
OVERVIEW

- Faceting & Statistics
- Streaming Expressions
- Parallel SQL
- Graph Traversal
- Machine Learning
OVERVIEW

- Faceting & Statistics
- Streaming Expressions
- Parallel SQL
- Graph Traversal
- Machine Learning
- Simple Alerting Engine
OVERVIEW

- Faceting & Statistics
- Streaming Expressions
- Parallel SQL
- Graph Traversal
- Machine Learning
- Simple Alerting Engine
- Demo!
WHAT IS SOLR?

Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene™

USED BY THOUSANDS INCLUDING

• Netflix
• NASA
• AT&T

• Disney
• Cisco
• Moody’s

• Best Buy
• Wal-Mart
• Apple

• Hortonworks
• Cloudera
• DataStax
FULL TEXT INDEXING

• Solr is best known for its Full Text Indexing capability.
• Based on “Inverted Index” technology in Lucene.
FULL DATA INDEXING – BEYOND TEXT

- Full Strings
- Numerics
- Timestamps
- Booleans
- Geospatial Lat/Lon
- Multilingual
Analytics is the discovery, interpretation, and communication of meaningful patterns in data.

- Wikipedia
There are lots of tools available to find patterns in data.

- Grouping
- Averages
- Minimum & Maximum

- Range, Variance, & Standard Deviation
- Skew
Solr provides lots of tools to find patterns in data.

Facets Component

Grouping using Facets (Solr’s version of GROUP BY)
Solr provides lots of tools to find patterns in data.

Stats Component

Min, Max, Count, Missing, Sum, Sum Of Squares, Mean, Std Deviation
Solr provides lots of tools to find patterns in data.

Function Queries

min(), max(), div(), sum()
numdocs(), log(), pow()
if(), and(), or(), not(),
more …
STREAMS

Not all data can be constrained to a finite set. Sometimes data must be processed on a continuously streaming basis.
STREAMING EXPRESSIONS

Simple and powerful Stream Processing language for Solr Cloud.

Composable functions for highly scalable Parallel Computing tasks.

Leverages Solr’s extensive Analytic capabilities.
STREAMING EXPRESSIONS

search
• Performs a standard Solr query
• Allows the output to be a very large continuous stream of documents rather than a single in-memory set.

```
search(
  collName,
  q="*:*",
  fl="a,b,c",
  sort="a asc"
)
```
STREAMING EXPRESSIONS

**topic**

- Subscribe to a query
- Keep track of what documents have already been visited
- Output only new data

```java
topic(
    checkpointCollection,
    q="*:*",
    id="topic1"
)
```
STREAMING EXPRESSIONS

update

- Push streamed objects into a collection
- Output a write transaction

update(
collName,
batchSize=10,
<stream>()
)
commit

- Performs a standard Solr commit
- Most often used with the `update()` expression.
- No output

commit(collName, batchSize=10)
**daemon**

- Wraps another function and executes it on the specified interval.
- Continues to run once it is started.
- Can be checked or stopped using its id.

```java
daemon(
    id="myDaemon",
    runInterval=1000,
    terminate=true,
    topic(...)
)
```
STREAMING EXPRESSIONS

parallel

• Performs a standard Solr query
• Allows the output to be a very large continuous stream of documents rather than a single in-memory set.

parallel(collName, q="*:*", fl="a,b,c", sort="a asc")
reduce

- Groups result data by common fields.
- Outputs ordered groups of documents (aka, “Tuples”)

```javascript
reduce(
  search(
    collName,
    q="*:*",
    fl="a,b,c",
    sort="a asc"),
  by="a",
  group(sort="a asc", n=4)
)
```
parallel

- Takes a stream and parallelizes it, ordering the results.
- Distributed the job across all the nodes in the cluster.

```
parallel(
    targetColl,
    reduce(
        search(
            collName,
            q="*:*",
            fl="a,b,c",
            sort="a asc"),
        by="a",
        group(sort="a asc", n=4)
    ),
    workers=20
)
```
STREAMING EXPRESSIONS

**facet**
- Provides aggregations that are rolled up over buckets.

```plaintext
facet(collection1, 
    q="*:*", 
    buckets="a", 
    bucketSorts="sum(a) desc", 
    sum(a), 
    avg(a), 
    count(*)
)
```
STREAMING EXPRESSIONS

jdbc

```java
jdbc(
    connection="jdbc:hsqldb:...",  
    sql="SELECT a,b,c FROM t",  
    sort="a desc",  
    driver="org.hsqldb.jdbcDriver"
)
```

- Allows Solr to reach out to any JDBC compliant data store and pull data.
- Can send any SQL that the target DB can handle.
- Pairs nicely with an "update()" to store the results in Solr.
features (collection1, q="*:*", featureset="features1", field="body", outcome="out_i", numTerms=250)

features
• Extracts key terms from a text field in a training set.
• Uses Information Gain algorithm.
• Designed to work with the “train” function.
train(collection1, 
features(...), 
q="*:*", 
name="model1", 
field="body", 
outcome="out_i", 
maxIterations=100 )

train

• Trains a Logistic Regression text classifier
• Uses a parallel iterative, batch Gradient Descent approach.
• Uses the features function to provide terms
model(modelCollection, id="myModel1", cacheMillis=200000)

- Retrieves and caches logistic regression text classification models
classify

- Classifies tuples as to the degree to which they do or do not match the model, measured on 0..1
- Uses a cached model
- Draws data from a standard search.

classify(
model(
modelCollection,
    id="model1",
    cacheMillis=5000),
search(contentCollection,
    q="id:(a b c)",
    fl="text_t, id",
    sort="id asc"),
field="text_t")
**STREAMING EXPRESSIONS - GRAPH**

**gatherNodes**
- Walks the data in a collection, using the “from” field as the edge to be traversed, and the “to” field as the field to be returned.
- Most basic graph traversal function.
- Can be nested.

```python
gatherNodes(
    emails,  
    walk="johndoe@apache.org->from",  
    gather="to")
```
gatherNodes

gatherNodes(
  emails,
  walk="johndoe@apache.org->from",
  gather="to")

- Tracks traversal.
- Cycle detection (within a collection)
- Can traverse across collections.
- Supports “fq” param to filter traversal.
- Combines well with other expressions.
STREAMING EXPRESSIONS - SQL

**innerJoin**

- Wraps two streams, Left and Right.
- Emits tuples where values match.
- Can cross Collections.
- Supports 1:1, M:1, 1:M, M:M join scenarios.

```sql
innerJoin(  
    search(people, q=*,
        fl="personId,name",
        sort="personId asc"),
    search(pets, q=type:cat,
        fl="ownerId,petName",
        sort="personId asc"),
    on="personId=ownerId"
)
```
leftOuterJoin(  
    search(people, q=:\*:*, fl=\"personId,name\", sort=\"personId asc\"),  
    search(pets, q=\text{type:cat}, fl=\text{\"ownerId,petName\"}, sort=\"personId asc\"),  
    on=\"personId=ownerId\"  
)
**intersect**

- Wraps two streams, A and B.
- Emits tuples from A (and only A) that DO exist in B.
- Can cross Collections.

```sql
intersect(
    search(people, q=*, fl="personId,name", sort="personId asc"),
    search(pets, q=type:cat, fl="ownerId,petName", sort="personId asc"),
    on="personId=ownerId"
)
```
PARALLEL SQL INTERFACE

REST Interface
• /sql Request Handler.
• Pass in SQL in the URL or JSON params.

Solr JDBC Driver
• Ships with SolrJ.
• Called from Java.

```
SELECT field1, field2
FROM Collection
WHERE field1='value'
```

```
SELECT field1, count(*)
FROM Collection
GROUP BY field1
ORDER BY count(*)
```
daemon(id="alertsDaemon",
    runInterval="1000",
    terminate="true",
    update(alertsCollection,
        batchSize=100,
        topic(alertsCheckpointCollection,
            logDataCollection,
            q="Exception",
            fl="id, timestamp, level, text",
            id="topicId",
            initialCheckpoint=0)
    )
)
The expression above uses a daemon expression to run a query once per second against the logDataCollection to see if there are any log lines that contain the text, “Exception.”

Only documents that have arrived since the last execution of the query are considered.

If found, the id, timestamp, level, and text are written to the “alertsCollection”.

Any external client cloud read records from the alertsCollection and send them to an email system or SMS system or Chat system to notify administrators.
DEMONSTRATION
PATRICK HOEFFEL

Senior Software Engineer
Search Consultant
Data Junkie

- BS, Computer Science, Catholic University, Wash., DC, 1989
- Worked for multiple startup companies
- Worked as a commercial contractor in Baghdad and Kabul
- Testified as a Data Expert at the WikiLeaks trial of (then) Bradley Manning
- Wrote Solr implementation for iTrace, a European Union-funded project, which provides policy makers with dynamic, quantified data on transfers of diverted conventional weapons, ammunition, and related materiel. http://www.conflictarm.com/itrace/
- Helped write and internal Log Analysis product based on Solr.

patrick.hoeffel@polarisalpha.com