SOLR

Final Presentation

Team Members:
Abhinav, Anand, Jeff, Mohit, Shreyas, Siyu, Xinyue, Yu

CS 5604, Fall 2017
Advised by Dr. Fox
Virginia Tech, Blacksburg, VA 24061

Overview

1. Updates
2. In Theory
3. Process, Improvements & Results
4. Quality Control
5. Geolocation
6. Recommendation
7. Future Work

UPDATES

Changes made to HBase Schema from previous presentation

Fields added:

- topic:topic-displaynames
- cluster:cluster-displaynames
- classification:classification-displaynames

Updates - SOLR Fall 2017

	Fall 2016	Fall 2017
schema.xml	No geospatial search function Copyfields #fields = 30	Feature added Copy and geoblacklight fields present #fields = 50+ Using DocValues to speed up faceting Index time field boosting
Morphline ETL	Datetime, multi-valued field parser, multiple field types	Extraction, cleaning, sanity check of data from HBase Timestamp conversion Records manipulation
Indexing	1.2 billion tweets	12,564 web pages and 5.9mil tweets
Incremental Indexing	Tested on VC	Tested on VC
Recommendation & Ranking	Didn't use clustering and classification for	Recommendation handler based on clustering and classification results;

In Theory

SOLR document scoring

1. tf : sqrt(freq)
2. idf : log(numDocs / (docFreq + 1)) + 1
3. fieldWeight
4. fieldNorm : 1 / sqrt(numTerms)

Search for all tweets containing terms “vegas”

"914864995565031425": **2.2832668** = (MATCH) weight(text:vegas in **171455**), result of: 2.2832668 = fieldWeight in 171455, product of: **1.4142135 = tf(freq=2.0)**, with freq of: 2.0; **6.4580536 = idf(docFreq=25492, maxDocs=5981684)**; **0.25** = **fieldNorm(doc=171455)**

Process, Improvements & Results

Software Packages and Versions

Cloudera CDH (Cloudera's Distribution Including Apache Hadoop) version **5.12.0**

Key packages included:

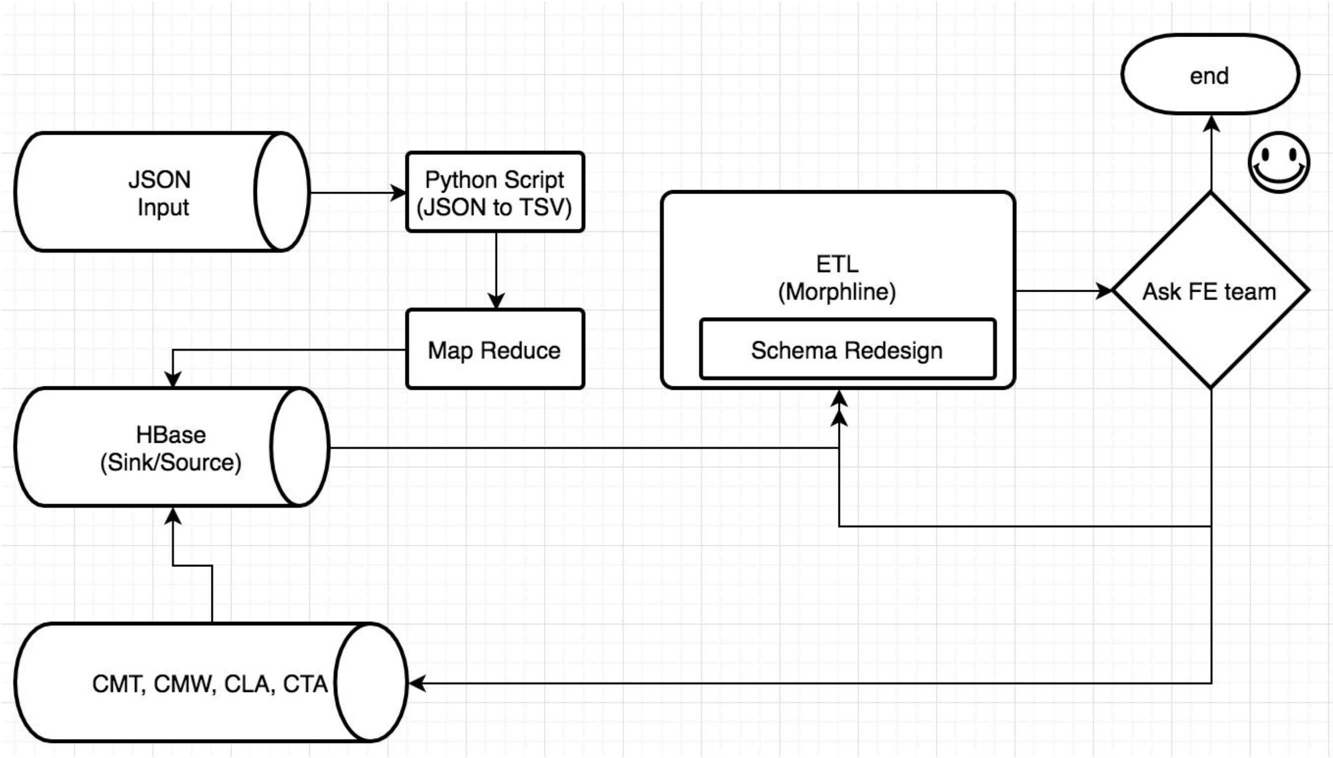
1. hadoop-hdfs (2.6.0)
2. hbase-solr (1.5)
3. hbase (1.2)
4. kite (1.0.0)
5. solr (4.10.3)
6. zookeeper (3.4.5)

For complete list visit:
https://archive.cloudera.com/cdh5/parcels/5.12.0/manifest.json

In Context

- Information
  - Tweets and Webpages
- Storage
  - Extract
    - From Hbase (using morphline)
  - Transform
    - Into Solr document (using morphline)
    - Cleaning (using morphline)
    - Deriving new field (using morphline and Hadoop)
  - Load
    - Indexing
- Retrieval
  - Queries from GeoBlackight and Visualization team
  - Recommendation

The Process


end
JSON Input
Python Script (JSON to TSV)
ETL (Morphline)
Ask FE team
Map Reduce
Schema Redesign
HBase (Sink/Source)
CMT, CMW, CLA, CTA

Solr Schema Challenges & Solutions

Challenges:

- All the data needs to be indexed in a single core
- Support 2 FE clients
- Multiple (2) input formats (webpage, tweet) and single output format

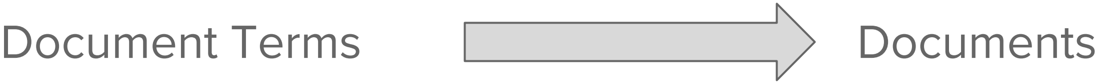
Solution:

- Separate morphline file for each input format
- Separate indexing operation for each input format
- Input format identified by ‘dc_format_s’ field

Better performance using DocValues

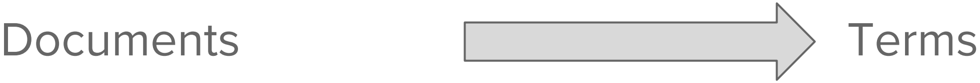
DocValues are a way of recording field values internally that is more efficient for some purposes, such as sorting and faceting, than traditional indexing

Traditional Solr Indexing



Document Terms → Documents

Indexing with DocValues



Documents → Terms

Examples

```
<field name="cluster" type="string" indexed="true"  multiValued="false" docValues="true"/>

<field name="category" type="string" indexed="true" stored="true" multiValued="true" docValues="true"/>
```

Index time document boosting

1. Using copyField

   Solr provides a functionality to copy values from one field to another during indexing. We can use this feature to copy the field which needs to be boosted in our default search field i.e. ‘text’.

```
<field name="keywords_boost" type="text_general" indexed="true"  multiValued="true"/>
<copyField source="*_boost" dest="text"/>
```

   This form of boosting works because after copying, the value of the boosted field is present twice as many times as other values.

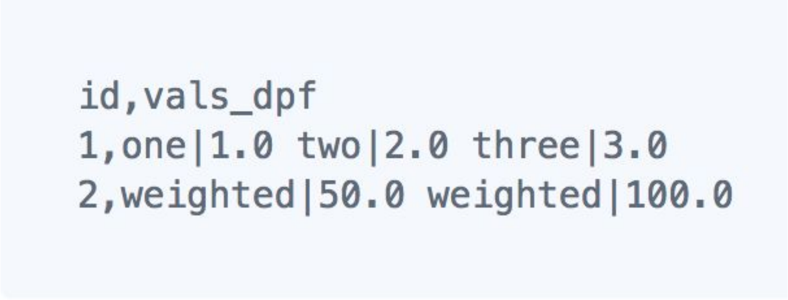
Index time document boosting

2. Using payloads (available from Solr 6.x)

   Our use cases:

   a. Boosting NER terms
   b. Multivalue boosting of topics, clusters and keyword terms

Example: Below we have 2 documents where content of the field ‘vals_dpf’ is boosted variably based on the words. In document id ‘1’, word ‘one’ has a boosting of 1.0, word ‘two’ has a boosting of 2.0, and so on.



```
id,vals_dpf
1,one|1.0 two|2.0 three|3.0
2,weighted|50.0 weighted|100.0
```

Key Stats & Figures

Collection Type	Document Count	Time	Memory (Heap Usage)
Webpage	12,564	Map: 88 sec Reduce: 531 sec	789 MB
Tweets	5,969,120	Map: 780 sec Reduce: 5300 sec	3 GB

Quality Control

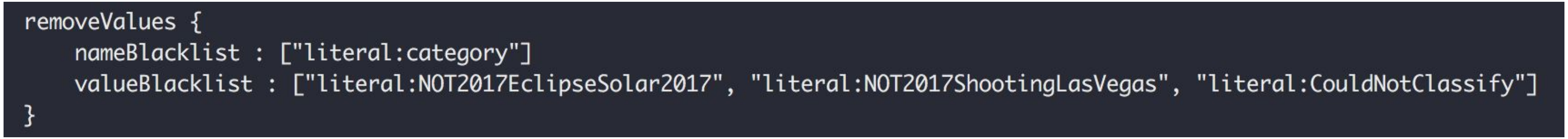
Index Quality Control

Validation and conversion of indexed data through morphline process in lily indexer.

- Fix records that are not formatted well in HBase
- Timestamp conversion: According to Solr datetime format
- Geo-data conversion: from string to double
- URL status code check: depends on given status code, decide whether to drop records
- Sanitize unknown Solr fields: remove fields that could not be recognized by Solr to avoid Solr errors
- Drop invalid and unnecessary records
- Log record at debug level to SLF4J: Log record content to log file or screen for debug and test

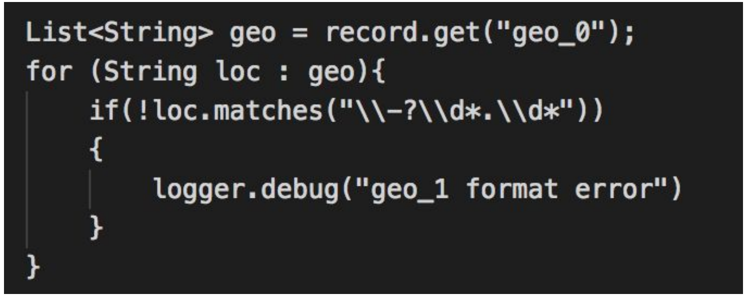
Examples

- Remove unrelated categories



```
removeValues {
    nameBlacklist : ["literal:category"]
    valueBlacklist : ["literal:NOT2017EclipseSolar2017", "literal:NOT2017ShootingLasVegas", "literal:CouldNotClassify"]
}
```

- Use regex to validate geo coordinates format



```
List<String> geo = record.get("geo_0");
for (String loc : geo){
    if(!loc.matches("\\-?\\d*.\\d*"))
    {
        logger.debug("geo_1 format error")
    }
}
```

Example for fixing data format:

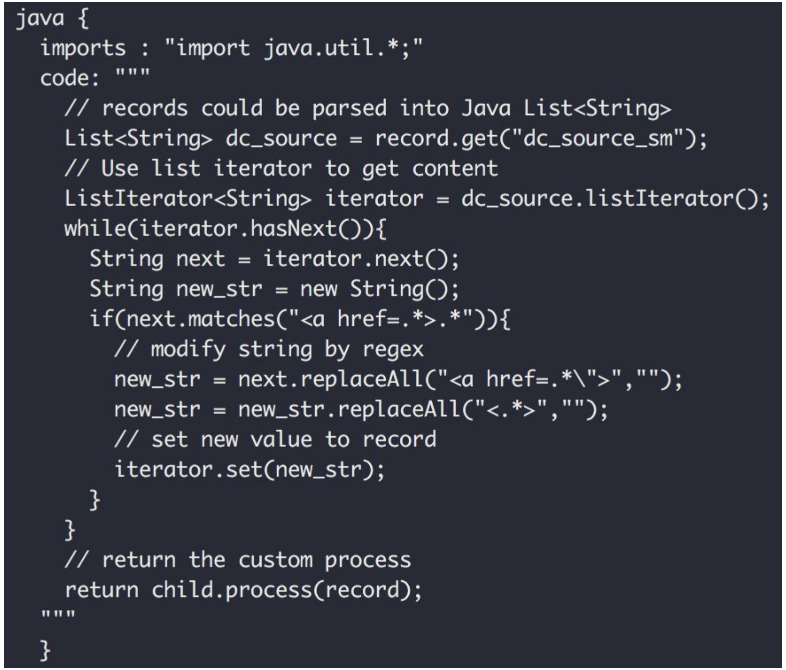
Input from HBase:

“<a href=\"https://about.twitter.com/products/tweetdeck\" rel=\"nofollow\">TweetDeck</a>”

Output for Solr records:

TweetDeck

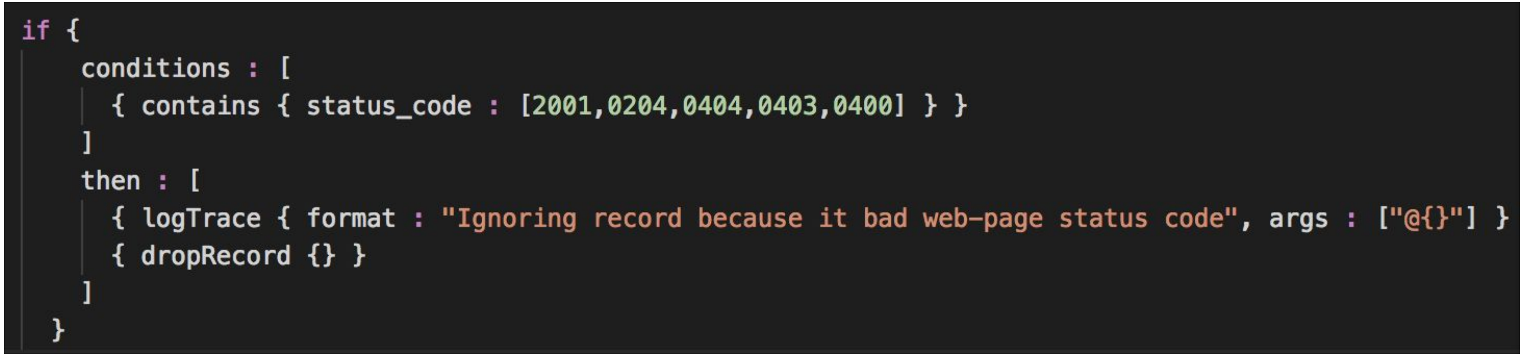
This practice could be extended to many kinds of data format issues in HBase.



```
java {
  imports : "import java.util.*;"
  code: """
    // records could be parsed into Java List<String>
    List<String> dc_source = record.get("dc_source_sm");
    // Use list iterator to get content
    ListIterator<String> iterator = dc_source.listIterator();
    while(iterator.hasNext()){
      String next = iterator.next();
      String new_str = new String();
      if(next.matches("<a href=.*>.*")){
        // modify string by regex
        new_str = next.replaceAll("<a href=.*\">","");
        new_str = new_str.replaceAll("<.*>","");
        // set new value to record
        iterator.set(new_str);
      }
    }
    // return the custom process
    return child.process(record);
  """
  }
```

Example for dropping specific records

Drop webpage records that contain bad status code: 0404, 0400 etc.

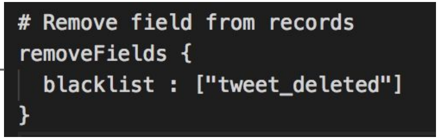


```
if {
  conditions : [
   { contains { status_code : [2001,0204,0404,0403,0400] } }
  ]
  then : [
   { logTrace { format : "Ignoring record because it bad web-page status code", args : ["@{}"] }
   { dropRecord {} }
  ]
 }
```

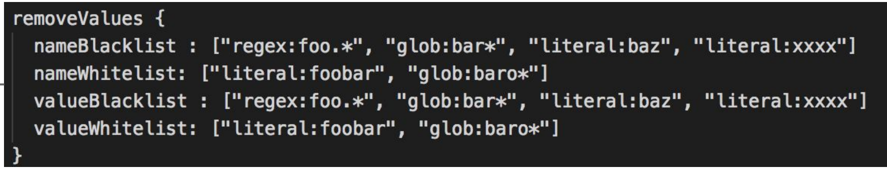
This is a good practice showing that any data related teams could use customized indicators in HBase and then Solr team could recognize it to ensure the indexed data quality.

Other things could be done

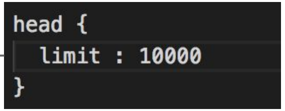
- Remove fields
- Add/Remove values to/from fields
- Index certain amount of records
- Find and replace fields



```
# Remove field from records
removeFields {
  blacklist : ["tweet_deleted"]
}
```



```
removeValues {
  nameBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxxx"]
  nameWhitelist: ["literal:foobar", "glob:baro*"]
  valueBlacklist : ["regex:foo.*", "glob:bar*", "literal:baz", "literal:xxxx"]
  valueWhitelist: ["literal:foobar", "glob:baro*"]
}
```



```
head {
  limit : 10000
}
```

More complex:

- Use Grok to extract information through regex (E.g., log data)
- Produce GeoIP from IP Addresses
- Etc.

Geolocation

Document Geolocation

Determine bounding box for place names from SNER-location

- Primarily for FE/visualization, additionally for indexing/searching

Data from Mapzen, OpenStreetMap, OpenAddresses.io

- Countries, States, Counties, Cities, Parks, Bodies of Water, etc…
- >4,500,000 place names

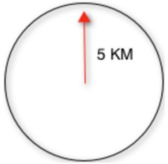
Using map-reduce and suffix arrays, each NER location name was matched to a location name from the bounding box data. The corresponding bounding box was then added to each document.

- Populated solr_geom field for 324714 tweets and 4442 webpages

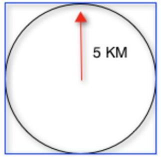
Considered using Google maps API but would have been slow due to rate-limiting

Geospatial Search

- Indexed geolocation as **SpatialRecursivePrefixTreeFieldTypelocation_rpt** instead of string
- This gives us ability to search in following ways
  - Geofilt
    - Input **[centre lat,lon and radius of circle]**



  - Bbox
    - Input **[centre lat,lon and radius of circle]**



  - Range query on a rectangular box
    - Input **[lower left lat,lon TO top right lat,lon]**

Recommendation

Recommendation System

1.MoreLikeThis Handler(MLT)

Based on **text similarity**, we made MoreLikeThis Handler recommend more similar documents to users when they are viewing one document. ----------(Specific recommendation)

2.Cluster_Classification Recommend Handler

Based on **Clustering and Classification results**, we made a Handler which can recommend documents belong to same cluster and class. ---------- (More general recommendation)

In order to avoid large cluster or large class problems, we first will recommend documents belonging to the same cluster and class. This can make the recommendation more accurate.

Recommendation System

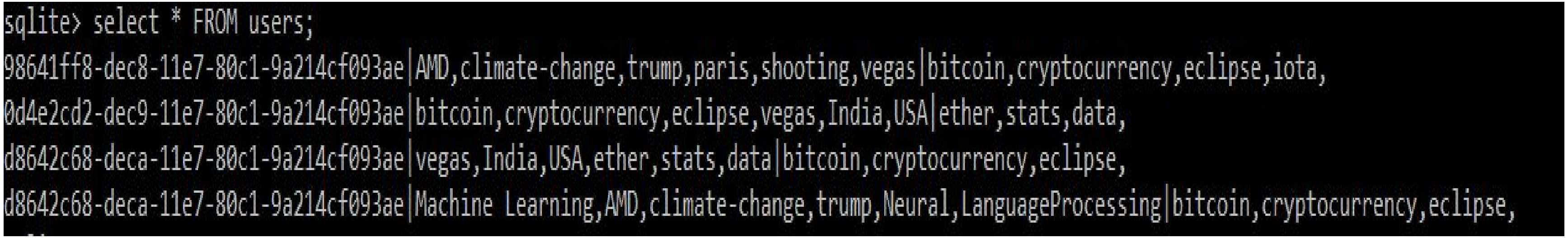
3. Personal Recommendation System

When users share common search interests they will be recommended documents they haven’t searched but were searched by user with common search interest.

E.g. A : Hurricane,Houston,safety,shooting - Recommendation - vegas,hospitals

B : Houston,safety,vegas,hospitals, - Recommendation - Hurricane,shooting

Image below shows a SQLite table (columns separated by |) , The first column is user id, second is keywords and third column are recommended items for the user.



```
sqlite> select * FROM users;
98641ff8-dec8-11e7-80c1-9a214cf093ae|AMD,climate-change,trump,paris,shooting,vegas|bitcoin,cryptocurrency,eclipse,iota,
0d4e2cd2-dec9-11e7-80c1-9a214cf093ae|bitcoin,cryptocurrency,eclipse,vegas,India,USA|ether,stats,data,
d8642c68-deca-11e7-80c1-9a214cf093ae|vegas,India,USA,ether,stats,data|bitcoin,cryptocurrency,eclipse,
d8642c68-deca-11e7-80c1-9a214cf093ae|Machine Learning,AMD,climate-change,trump,Neural,LanguageProcessing|bitcoin,cryptocurrency,eclipse,
```

Future Work

1. Collect more user data, implement more complex recommendation handler.
2. Configure Live Indexing mode on Hadoop cluster.
3. Load testing

ACKNOWLEDGMENTS

This material is based upon work in the Global Event and Trend Archive Research (GETAR) project, supported by the National Science Foundation under Grant No. IIS-1619028