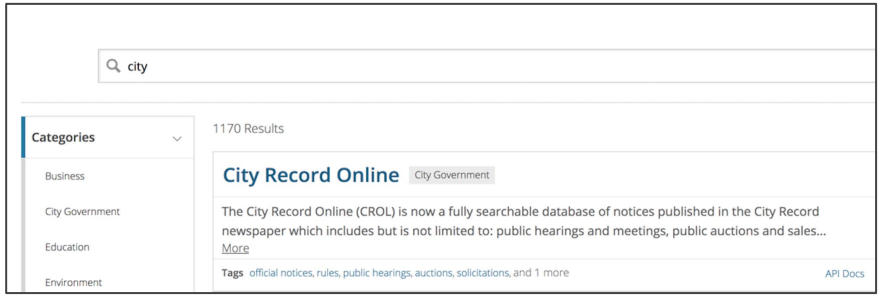
Search Engine for Structured Data

Big Data Term Project
Ksenia-Stella-Zhiwei
Spring 2018

Project Goal: search engine for structured data

NYC OpenData

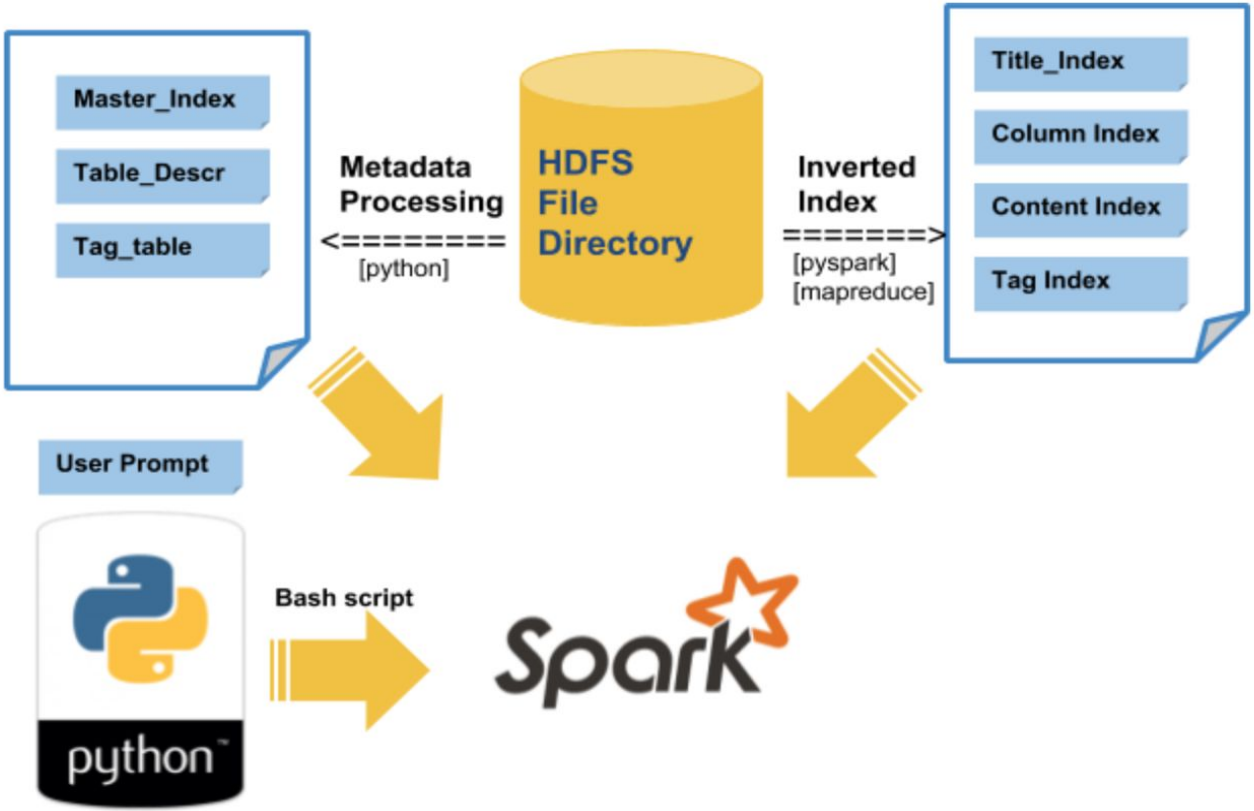


Only Title Search Available!

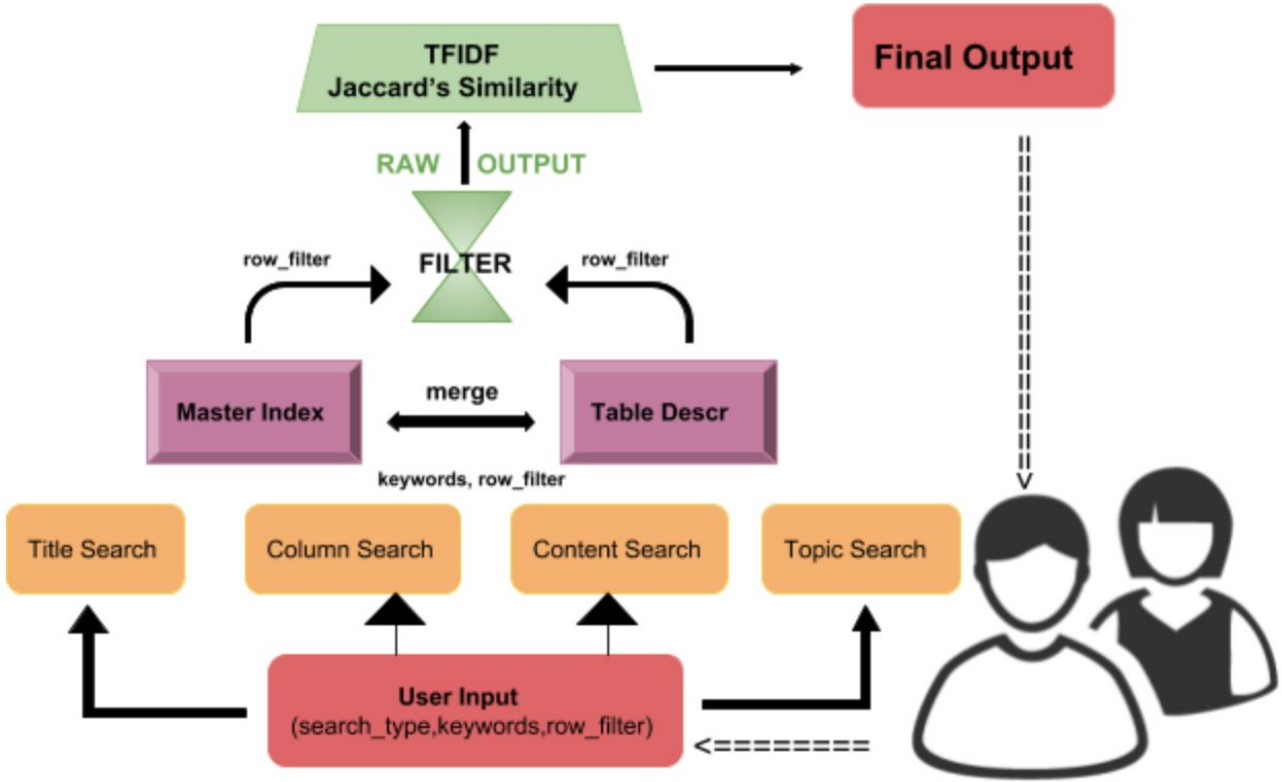
We will offer search by:

- Title
- Column
- Content
- Topic
- Filter by table length

Architecture & Design: Method Design



Architecture & Design: Application Design



Json Processing and Other Tables

Inverted Indices: title, column, content, tags

1. **Collect** the data
2. Parse collection of tables
   a. **Lowercase** all words: “SPARK” -> “spark”
   b. **Tokenize**: “search engine” -> “search”, “engine”
   c. Filter out **stop words**: “the”, “a”, “an”, etc.
   d. Stem each token using **Porter Stemmer Algorithm**: “**fish**er”, “**fish**ing”, “**fish**es”, “**fish**ed” -> “**fish**”

Inverted Index

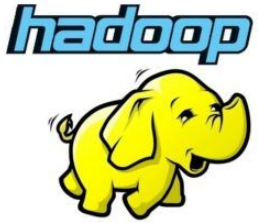
Building the index using **MapReduce**

- **Key**: vocabulary term
- **Value**: postings list (ID's of tables where the term appears sorted by in-table term frequency)

```
manhattan       25      (9,8,7,6,46,39,38,37,35,34,30,3,29,26,23,22,20,19,18,17,16,13,12,10,0)
new     23      (9,7,6,46,39,38,37,35,34,30,29,26,24,23,22,21,2,19,18,16,13,12,0)
island  23      (9,7,6,46,39,38,37,35,34,30,29,26,23,22,2,19,18,17,16,13,12,10,0)
brooklyn        23      (9,7,6,46,39,38,37,35,34,30,3,29,26,23,19,18,17,16,13,12,10,1,0)
```

We compared two methods for constructing indices

- Built **two versions** of inverted indices
  - PySpark
  - Hadoop
- Decided to go with **PySpark**
  - Built-in functionality for querying
  - Machine learning library



Querying

User Queries → Spark.SQL Queries

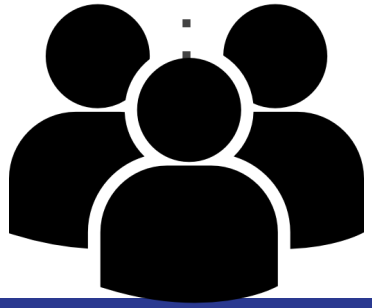
Python

- Search_Type:
  ---1,2
- Keywords:
  ---cab,taxi,city
- Row_Filter:
  ---100000

Spark

title_search([cab,taxi,city],100000)

column_search([cab,taxi,city],100000)



Processing Queries

Spark SQL

[Key, [Doc_ID1, Doc_ID2, Doc_ID3, Doc_ID4...]

Title Index

Column Index

Content Index

Tag Index

Join on Unique Document ID

[Doc_ID,Category,Descr]

Master Index

Table Descriptions

[Doc_ID,Table_Name]

Search Ranking: TF-IDF

- Term Frequency (TF): the number of occurrences of a term in a document (**how frequent**)
- Document Frequency (DF): the number of documents containing a term (**how discriminative**)
- TF-IDF: **how important** a term is in a document

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1} \qquad TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

Search Ranking: Vector Space Model

- Represent each document as a vector, with each entry being the TF.IDF weight of the corresponding term *t* in document *d*
- Represent each query as a vector in the same vector space as the documents
- **Cosine similarity**: the relevance between each document and the query

$$sim(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$

Search Ranking: Implementation

Step 1: Preprocessing

- Calculate **TF.IDF scores** from the inverted indices
- 4 inverted index files -> **4 TF.IDF models**
- Save the models into files

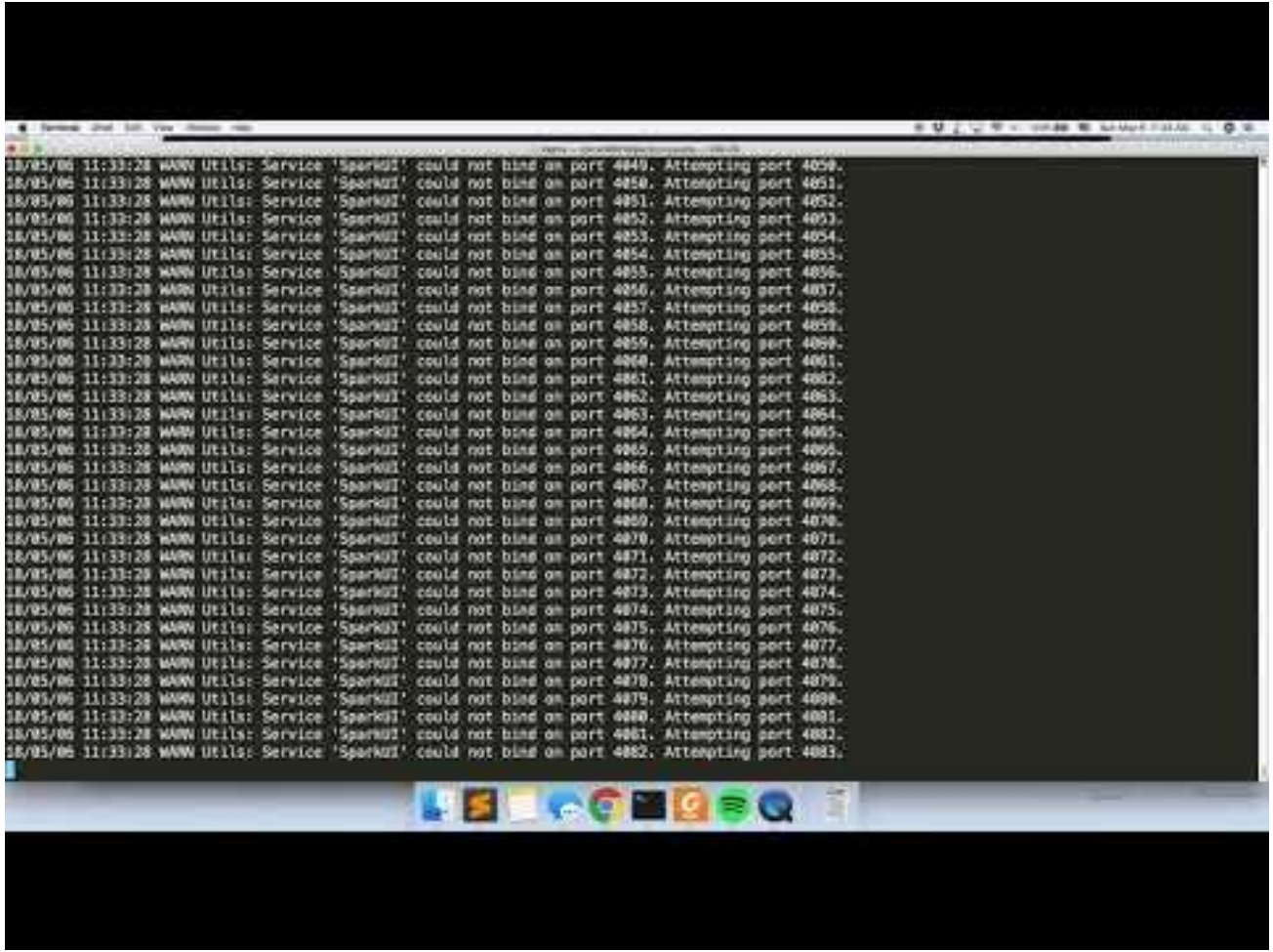
Step 2: Ranking raw output

- Get the query & Parse
- Run TF.IDF models
- Calculate the **cosine similarities** and sort by scores
- **Rank Output**

Demo

```
Here is your content search result
+--------------------+--------------------+--------------------+
|          Table_Name|            Category|         Description|
+--------------------+--------------------+--------------------+
|2015 2016 Student...|           Education|Student Disciplin...|
|Capital Project S...|Housing & Develop...|List of capital p...|
|Mayor's Office to...|     Social Services|The dataset conta...|
|Capital Commitmen...|     City Government|This dataset cont...|
|LinkNYC New Site ...|     Social Services|A map of proposed...|
|Benefits and Prog...|     Social Services|"This dataset pro...|
|2017 Diversity Re...|           Education|             Missing|
|Parking Violation...|     City Government|Parking Violation...|
|Full-Time And Ful...|     City Government|This dataset cont...|
|Verified Location...|     City Government|An agency-verifie...|
|2005 Street Tree ...|         Environment|Citywide street t...|
|2015-16 Student D...|           Education|             Missing|
|FHV Base Aggregat...|      Transportation|Monthly report in...|
|2015-16 Demograph...|           Education|             Missing|
|Deaths by Year an...|       Public Safety|This is a breakdo...|
|2015 - 2016 Audit...|           Education|Official audited ...|
|FY17 MMR Agency P...|     City Government|NYC agency perfor...|
|Doing Business Se...|     City Government|The Doing Busines...|
|2016-17 Physical ...|           Education|             Missing|
|NYPD Complaint Ma...|       Public Safety|This dataset incl...|
+--------------------+--------------------+--------------------+

Here is your topic search result
+--------------------+--------------------+--------------------+
|          Table_Name|            Category|         Description|
+--------------------+--------------------+--------------------+
|Most Popular Baby...|              Health|The most popular ...|
|NYCHA Application...|Housing & Develop...|Priority codes as...|
+--------------------+--------------------+--------------------+
```



Results

- Custom **Search Capabilities**
  - Title, Column, Content, Tags
  - Row Filter
- Improved **Ranking**
  - TF-IDF using MLlib, Jaccard similarity
- **Scalable** Implementation
  - Inverted index implemented in PySpark
  - Can build additional applications on top

References

[1] Cafarella, M. J. et al. WebTables:Exploring the Power of Tables on the Web.

[2] Dasu, T. et al. Mining Database Structure; Or, How to Build a Data Quality Browser.

[3] Agrawal, S. et al. DBXplorer: A System for Keyword-Based Search over Relational Database.

[4] Luo, Y. (n.d.). Spark:A Keyword Search Engine on Relational Databases.

[5] Rejaraman, A., & Ullman, J. (n.d.). *Mining of Massive Datasets*.

[6] Cloud9. (n.d.). Retrieved April 13, 2018, from https://lintool.github.io/Cloud9/docs/exercises/indexing.html