Galago Tutorial

Text Technology for Data Science
INFR11145

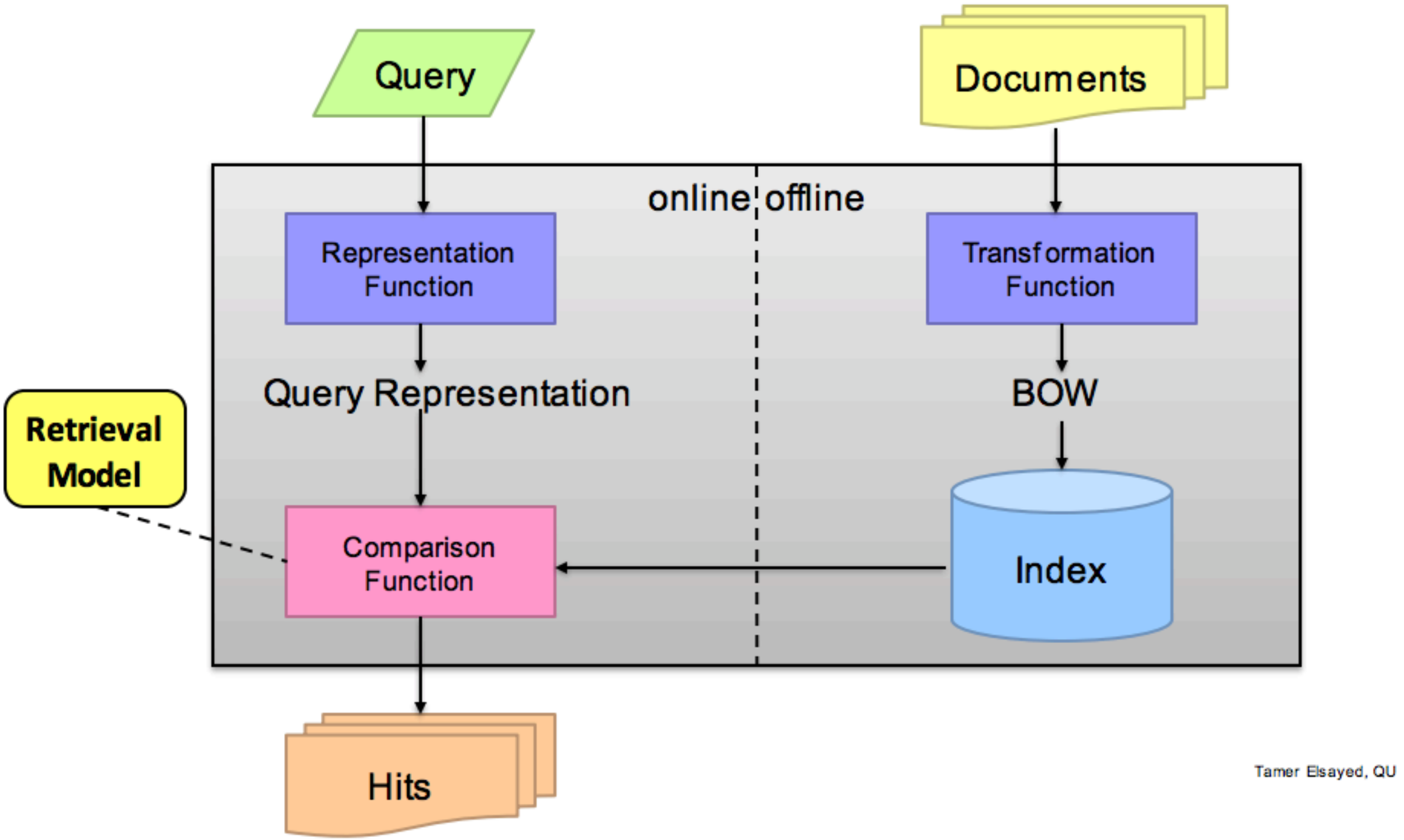
Clara Vania

Galago

- A search engine toolkit written in Java, specifically designed for research.
- It consists of various search engine components which are pluggable for indexing and retrieval.
- Requirements: Java, Apache Maven installed
- Most of this tutorial are taken from:

  https://sourceforge.net/p/lemur/wiki/Galago/

IR Black Box (from Lecture 2)



Tamer Elsayed, QU

Configuration

- Galago uses *configuration* to perform indexing and retrieval.
- The configuration defines the *parameters* or *options* that we want to apply to our index or search process.
- There are two ways to define configuration:
  - JSON file, see (http://www.json.org/)
  - command line

JSON

- JSON (JavaScript Object Notation) is data format which is easy for machines to parse and generate.
- It is based on two structures:
  - A collection of (name, value) pairs, similar to *hash/ dictionary* in Python
  - An ordered list, similar to *array*.

Example Parameters (1)

JSON format

```
{
  "key1" : "string",
  "key2" : true,
  "key3" : 5,
  "key4" : 3.14159
}
```

Command-line format

```
--key1=string --key2=true --key3=5 --key4=3.14159
```

Example Parameters (2)

JSON format

```
{
   "arrkey" : [ val1, val2, val3 ],
   "mapkey" : { 'a' : 5,
                'b' : false
              }
}
```

Command-line format

```
--arrkey+val1 --arrkey+val2 --arrkey+val3 --mapkey/a=5 --mapkey/b=false
```

Indexing

Parameters	Note
`indexPath`	The directory path of the index
`inputPath`	The path to a file or directory
`fileType`	The file format `{trectext, txt, pdf, html}`
`stemmedPostings` `nonStemmedPostings`	`{true/false}` Selects whether to build stemmed/nonstemmed inverted index (default=`true`)
`stemmer+porter|krovetz`	`{porter/krovetz}` Selects which stemmers to use (default=`porter`)
`tokenizer/fields+` `{field-name}`	Selects field parts to index (omitted by default)

Example: Indexing

indexParam.json

```
{
  "fileType"      : "trectext",
  "inputPath"     : "collections/trec.sample",
  "indexPath"     : "sample_index",
  "stemmedPostings": true,
  "nonStemmedPostings": true,
  "stemmer": ["krovetz"],
  “tokenizer":["docno", "headline", "text"],
  "corpus": true
}
```

Don’t use known extension file for **trectext** file type!

and then run: `galago build indexParam.json`

Example: Indexing

The equivalent command-line *configuration*:

```
galago build --fileType=trectext
 --inputPath=collections/trec.sample
 --indexPath=sample_index
--stemmedPostings=true
--nonStemmedPostings=true
--stemmer+krovetz
--tokenizer/fields+docno
--tokenizer/fields+headline
--tokenizer/fields+text
```

View the `buildManifest.json` file inside the index directory to see the parameter definitions for the index that was build.

Example: Indexing

Check if your index is built successfully:

```
Stage parsePostings completed with 0 errors.
Stage writeFields completed with 0 errors.
Stage writeCorpusKeys completed with 0 errors.
Stage writeLengths completed with 0 errors.
Stage writeNamesRev completed with 0 errors.
Stage writeNames completed with 0 errors.
Stage writePostings completed with 0 errors.
Stage writePostings-krovetz completed with 0 errors.
Done Indexing.
  - 0.00 Hours
  - 0.21 Minutes
  - 12.83 Seconds
Documents Indexed: 1000.
```

View the `buildManifest.json` file inside the index directory to see the parameter definitions for the index that was build.

Retrieval

- Similar with indexing, retrieval in Galago is done using a *configuration*, either with JSON file or command-line.

- The configuration contains the queries and the preprocessing steps that we want to apply to the queries.

- We can search for various types of queries which follow **Galago Query Language** (inspired by Indri Query Language).

Galago Query Language

- Can be treated as a query tree.
- Each node represents an *Operator* which has a specific *function*.
- The function has two types of parameters:
  - Static: do not depend on the document currently being scored
  - Values: depend on the document.

Galago Query Language

Operators	Note	Examples
`term`	single/multi terms query.	`income tax`
`#od:N`	terms must appear ordered, with at most N-1 terms between each.	`#od:5(income tax)`
`#uw:N`	all terms must appear within window of length N in any order.	`#uw:10(new york)`

`#combine`	combine all results from the sub-operators, optionally with weights.	`#combine( #od:1(new york) #od:3(us president))`
		
		`#combine:0=0.8:1=0.2(term1 term2))`
		
		`#combine:0=0.8:1=0.2(#opt1 #opt2))`

Retrieval Models

- Galago uses various retrieval models that we can use during searching, for example boolean model, BM25, or relevance feedback model.

- The default model is for query likelihood with Dirichlet term smoothing.

- Please refer to: https://sourceforge.net/p/lemur/wiki/Galago%20Operators for the available models*.

- During retrieval, we can define the model using the **Galago Query Language** *operators*.

*) Some definitions are also available in: https://www.lemurproject.org/lemur/IndriQueryLanguage.php

Example Models

- **#bm25(): The BM25 Okapi Model**

  `#combine(#bm25(international) #bm25(organized) #bm25(crime))`

- **#rm(): Relevance Feedback Model**

  Params: `“fbDocs”, “fbTerm”, “fbOrigWeight”, etc.`

- **#prms(): Probabilistic Model for Semi Structured Data**

  Useful for indexing semi-structured data with “tags”, such as Twitter, news articles, etc. Params:
  - fields: the fields from which terms should be evaluated.
  - weights: the weights for the specified fields.

Retrieval Configuration

Similar with indexing, we can also use JSON file or command line to perform searching.

```
{
 "casefold" : true,
 "fields" : ["headline", "text"],
 "requested" : 5,
 "queries"  : [
   {
     "number" : "q1",
     "text"   : "politicians"
   },
   {
     "number" : "q2",
     "text"   : "#combine( #od:5 (income taxes))"
   },
   {
     "number" : "q3",
     "text"   : "#combine(income taxes)",
     "windowLimit" : 10
   },
   {
     "number" : "q4",
     "text"   : "#combine(#bm25(income) #bm25(taxes))"
   }]
}
```

field(s) to search

how many documents to retrieve

window size used for searching (proximity search)

See the documentation for each operators for more params!

Batch-Search

Query file: queryParam.json

```
galago batch-search --verbose=true
    --index=sample_index
    --defaultTextPart=postings.krovetz
    --requested=10 queryParam.json
```

Searching through command-line:

```
galago batch-search  --requested=10 \
          --index=/myindexes/ap89.idx \
          --query="#combine(#od:1(six.head survivors.head))"
```

Batch-Search

Example results:

```
Oct 23, 2017 4:40:00 PM org.lemurproject.galago.core.tools.apps.BatchSearch run
INFO: RUNNING: q6 : #combine(#bm25(income) #bm25(taxes))
Oct 23, 2017 4:40:00 PM org.lemurproject.galago.core.tools.apps.BatchSearch run
INFO: Transformed Query:
#combine:w=1.0(
    #bm25:collectionLength=417913:documentCount=1000:maximumCount=8:nodeDocumentCount=93:nodeFrequency=156:w=0.5(
        #lengths:document:part=lengths()
        #counts:income:part=postings.krovetz()
      )
    #bm25:collectionLength=417913:documentCount=1000:maximumCount=23:nodeDocumentCount=191:nodeFrequency=511:w=0.5(
        #lengths:document:part=lengths()
        #counts:taxes:part=postings.krovetz()
      )
  )

q6 Q0 92 1 3.74657242 galago
q6 Q0 65 2 3.70148919 galago
q6 Q0 3533 3 3.49014705 galago
q6 Q0 3817 4 3.43985858 galago
q6 Q0 3706 5 3.26636618 galago
q6 Q0 3708 6 3.19154172 galago
q6 Q0 3734 7 3.10640340 galago
q6 Q0 3519 8 3.08521029 galago
q6 Q0 3710 9 3.05305408 galago
q6 Q0 163 10 3.01678030 galago
```

Relevance Feedback

A relevance feedback model, where the original query term are augmented by the *specified number of feedback expansion terms* at specified *weight*.

#RM()

```
{
  "verbose"    : true,
  "casefold"   : true,
  "requested" : 5,
  "index"      : "/myindexes/ap89_fields.idx",
  "relevanceModel" : "org.lemurproject.galago.core.retrieval.prf.RelevanceModel1",
  "fbDocs" : 10,
  "fbTerm" : 5,
  "fbOrigWeight" : 0.75,
  "passageQuery" : true,  [passage query requires size and shift parameters]
  "passageSize" : 10,
  "passageShift" : 20,
  "extentQuery" : true,
  "rmstopwords" : "rmstop",
  "rmwhitelist" : "/myqueries/whitelist.txt",   [be careful with this one!]
  "rmStemmer" : "org.lemurproject.galago.core.parse.stem.KrovetzStemmer",
  "queries" : [
    {
      "number" : "rm",
      "text" : "#rm(six survivors)"
    }
  ]
}
```

References

- https://sourceforge.net/p/lemur/wiki/Galago/
- https://www.lemurproject.org/lemur/IndriQueryLanguage.php
- https://sourceforge.net/p/lemur/wiki/Galago%20Operators/
- https://sourceforge.net/p/lemur/wiki/Galago%20Functions/
- https://github.com/jiepujiang/cs646_tutorials