Toolkits in IR
-- Indri/Lemur and trec_eval

Jing He

hejing@iro.umontreal.ca

What will you know after the lecture?

- How to build index/retrieve documents in Indri/Lemur?
- How to evaluate the retrieved results in trec_eval ?
- How to develop new retrieval model in Lemur by extending its base classes?

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur

Indri/Lemur: Indexing

- Supported Document Format
  - TREC Text
  - TREC Web
  - Plain Text
  - HTML
  - XML
  - PDF
  - MS Word, PowerPoint (only on Windows and each is installed)

Indri/Lemur: Indexing

- Supported Document Format
  - TREC Text
  - TREC Web
  - Plain Text
  - HTML
  - XML
  - PDF
  - MS Word, PowerPoint (only on Windows and each is installed)

Do you still remember TREC?

Text REtrieval Conference

TREC Text Format

```
<DOC>
      <DOCNO>docid1</DOCNO>
      <TEXT>
        I am the first document.
      </TEXT>
</DOC>
<DOC>
      <DOCNO>docid2</DOCNO>
      <TEXT>
        Unfortunately, I am the second and the last.
      </TEXT>
</DOC>
```

Indri/Lemur: Indexing

- Command for indexing
  - IndriBuildIndex <parameter_files>
- What should be contained in a parameter file?
  - Location/format of the document collection?
  - Location of the built index?
  - Preprocessing: stopwords, stem, etc.?
  - Fields?

Indexing Parameter File

- General Format in XML style (for both indexing and retrieval)

```
<parameters>
    <option></option>
    <option></option>
      …
    <option></option>
</parameters>
```

Indexing Parameter File

- Location/Format of the document collection

```
<corpus>
   <path>/path/to/source/files</path>
   <class>trectext</class>
</corpus>
```

- Location of the index

```
<index>/path/to/the/index</index>
```

- Memory Used

```
<memory>256M</memory>
```

Indexing Parameter File

- Location/Format of the document collection

```
<corpus>
   <path>/path/to/source/files</path>
   <class>trectext</class>
</corpus>
```

- Location of the index

```
<index>/path/to/the/index</index>
```

- Memory Used

```
<memory>256M</memory>
```

Q: Why does indexing process use so much memory?

Indexing Parameter File

- Stop words

```
<stopper>
        <word>a</word>
        <word>the</word>
                …
        <word>is</word>
</stopper>
```

- Stemmer

```
<stemmer>
        <name>krovetz</name>
</stemmer>
```

Indexing Parameter File

- Stop words

```
<stopper>
        <word>a</word>
        <word>the</word>
                …
        <word>is</word>
</stopper>
```

- Stemmer

```
<stemmer>
        <name>krovetz</name>
</stemmer>
```

Q: Why do we need a stemmer in indexing?

Indri/Lemur: Indexing

- Run a Demo

Indri/Lemur: Indexing

- Check an Index with “dumpindex”
  - dumpindex <index> v //show the vocabulary
  - dumpindex <index> s //statistics of the index
  - dumpindex <index> t <term> //get the inverted list of a term
  - dumpindex <index> dt <doc-id>//get the doc text of a doc
  - ……

Indri/Lemur: Indexing

- Run a Demo

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur

Indri/Lemur: Retrieval

- IndriRunQuery <parameter-file>
- What does the program need to know?

Indri/Lemur: Retrieval

- IndriRunQuery <parameter-file>
- What does the program need to know?
  - Location of the index

```
<index>/path/to/the/index</index>
```

Indri/Lemur: Retrieval

- IndriRunQuery <parameter-file>
- What does the program need to know?
  - Location of the index
  - Queries

```
<query>
   <number>1</number>
   <text>this is the first query</text>
 </query>
<query>
   <number>2</number>
   <text>another query to run</text>
 </query>
```

Indri/Lemur: Retrieval

- IndriRunQuery <parameter-file>
- What does the program need to know?
  - Location of the index
  - Queries
  - Length of the returned list

    `<count>`**`50`**`</count>`

Indri/Lemur: Retrieval

- IndriRunQuery <parameter-file>
- What does the program need to know?
  - Location of the index
  - Queries
  - Length of the returned list
  - Output format

```
<runID>runName</runID>
<trecFormat>true</trecFormat>
```

Lemur/Indri: Retrieval

- Demo

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur

Evaluating with trec_eval

- trec_eval <judgment-file> <result-file>

Review: What are the components for a test collection?

Evaluating with trec_eval

- trec_eval <judgment-file> <result-file>
- Format of the result file

```
<queryID> Q0 <DocID> <rank> <score> <runID>

150 Q0 AP890101-0001 1 -4.83646 OMGIR
150 Q0 AP890101-0015 2 -7.06236 OMGIR
151 Q0 AP890101-0004 2 -3.11372 OMGIR
151 Q0 AP890101-0008 2 -9.26431 OMGIR
```

Evaluating with trec_eval

- trec_eval <judgment-file> <result-file>
- Format of the result file
- Format of the judgment file

```
<queryID> 0 <DocID> <judgment>
51   0    AP900607-0001  0
53   0    AP900607-0235  1
```

Evaluating with trec_eval

- demo

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur

Indri Query Language

- Structured Query
- Explicitly Presenting Rich Query Information
  - Term

term	dog	occurrences of dog (Indri will stem and stop)
“term”	“dog”	occurrences of dog (Indri will not stem or stop)

Indri Query Language

- Structured Query
- Explicitly Presenting Query Information
  - Term importance
  - Term's relation
    - Proximity

ordered window	`#od`*n*`(blue car)`	`blue` *n* words or less before `car`
unordered window	`#ud`*n*`(blue car)`	`blue` within *n* words of `car`

Indri Query Language

- Structured Query
- Explicitly Presenting Query Information
  - Term importance
  - Term's relation
    - Proximity
    - Synonyms

synonym list	`#syn(car automobile)`	occurrences of `car` or `automobile`
weighted synonym	`#wsyn(1.0 car 0.5 automobile)`	like synonym, but only counts occurrences of `automobile` as 0.5 of an occurrence

Indri Query Language

- Structured Query
- Explicitly Presenting Query Information
  - Term, Term's relation
  - Field Information

restriction	`dog.title`	counts only occurrences of `dog` in `title` field
	`dog.title,header`	counts occurrences of `dog` in `title` or `header`
evaluation	`dog.(title)`	builds belief *b*`(dog)` using `title` language model
	`dog.(title,header)`	*b*`(dog)` estimated using language model from concatenation of all `title` and `header` fields

Indri Query Language

- Structured Query
- Explicitly Presenting Query Information
  - Term, Term's relation, Field Information
  - Aggregate weights

combine	`#combine(dog train)`	0.5 log(*b*(`dog`)) + 0.5 log(*b*(`train`))
weight, wand	`#weight(1.0 dog 0.5 train)`	0.67 log(*b*(`dog`)) + 0.33 log(*b*(`train`))
wsum	`#wsum(1.0 dog 0.5 dog.(title))`	log(0.67 *b*(`dog`) + 0.33 *b*(`dog.(title)`))
not	`#not(dog)`	log(1 - *b*(`dog`))
max	`#max(dog train)`	returns maximum of *b*(`dog`) and *b*(`train`)
or	`#or(dog cat)`	log(1 - (1 - *b*(`dog`)) * (1 - *b*(`cat`)))

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extending Lemur

Extending Lemur

- APIs
  - Index
  - Run queries in Lemur
  - Retrieval framework
  - Extending
- An example of extending

Lemur: Index

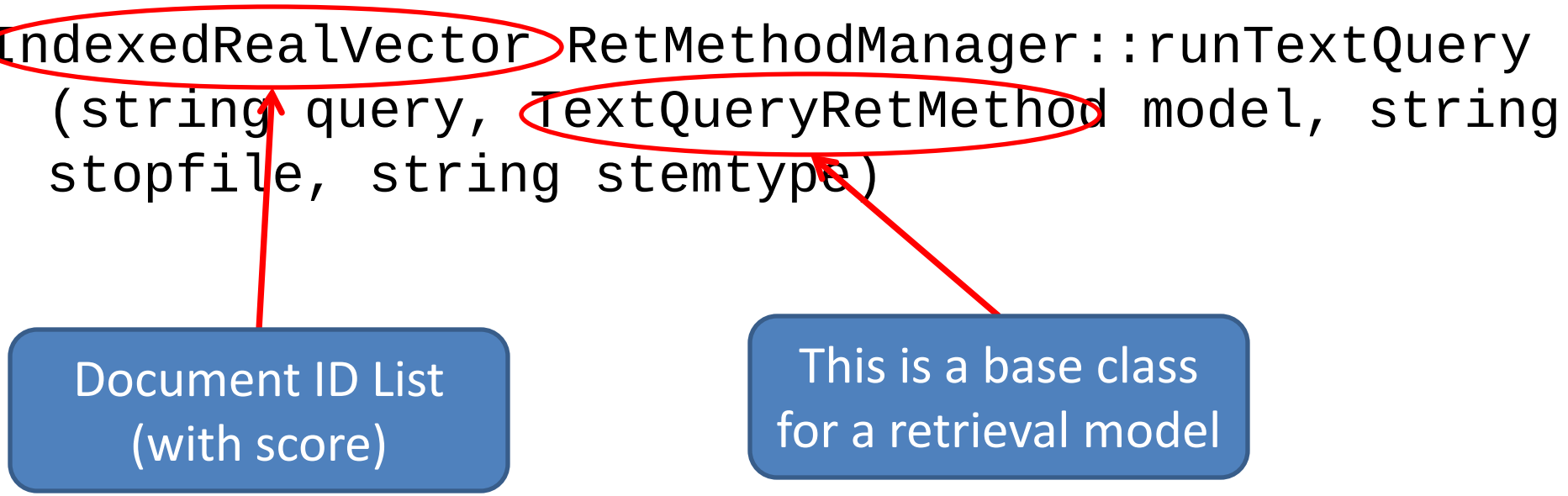
- Open an Index

```
Lemur::Index
  IndexManager::openIndex(string indexFile)
```

- What can you get from an Index object?
  - http://www.lemurproject.org/doxygen/lemur/html/classLemur_1_1Index.html

Lemur: Run a Query

```
IndexedRealVector RetMethodManager::runTextQuery
  (string query, TextQueryRetMethod model, string
  stopfile, string stemtype)
```



Lemur: TextQueryRetMethod

- Constructor
  - `TextQueryRetMethod (const Index &ind, ScoreAccumulator &accumulator)`
- Virtual Functions
  - `TextQueryRep * computeTextQueryRep (const TermQuery &qry)//query->query model (VSM, Language Model, etc.)`
  - `DocumentRep * computeDocRep (DOCID_T docID)//doc->doc model (VSM, LM, etc.)`
  - `ScoreFunction * scoreFunc// comparing doc model and query model`

Lemur: TextQueryRetMethod

- Constructor
  - `TextQueryRetMethod (const Index &ind, ScoreAccumulator &accumulator)`
- Virtual Functions
  - `TextQueryRep * computeTextQueryRep (const TermQuery &qry)//query->query model (VSM, Language Model, etc.)`
  - `DocumentRep * computeDocRep (DOCID_T docID)//doc->doc model (VSM, LM, etc.)`
  - `ScoreFunction * scoreFunc// comparing doc model and query model`

Abstract Factory Pattern in Software Engineering

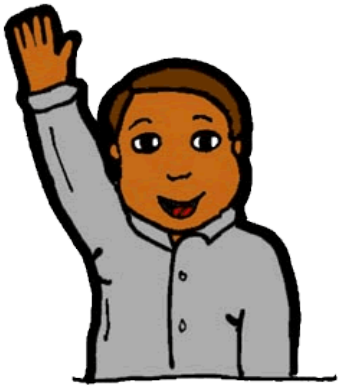
Example: CosSimRetMethod

```
TextQueryRep * computeTextQueryRep (const
   TermQuery &qry)//create a
  CosSimQueryRep object
DocumentRep * computeDocRep (DOCID_T docI
  D)// create a CosSimDocRep object
ScoreFunction * scoreFunc// return a
  CosSimScoreFunc object
```

If you want to extend the vector space model that uses a different query representation, you can create a subclass of CosSimRetMethod can overload the computeTextQueryRep method

Outline

- Indexing
- Retrieval
- Evaluation
- Indri Query Language
- Extend Lemur




Thank You!