An Overview of the Indri Search Engine

Don Metzler
Center for Intelligent Information Retrieval
University of Massachusetts, Amherst

Joint work with Trevor Strohman, Howard Turtle, and Bruce Croft
Outline

- Overview
- Retrieval Model
- System Architecture
- Evaluation
- Conclusions
Zoology 101

- Lemurs are primates found only in Madagascar
- 50 species (17 are endangered)
- Ring-tailed lemurs
  - *lemur catta*
Zoology 101

- The *indri* is the largest type of lemur
- When first spotted the natives yelled "*Indri! Indri!*"
- Malagasy for "*Look! Over there!*"
What is INDRI?

- INDRI is a “larger” version of the Lemur Toolkit
- Influences
  - INQUERY [Callan, et. al. ’92]
    - Inference network framework
    - Structured query language
  - Lemur [http://www.lemurproject.org/]
    - Language modeling (LM) toolkit
  - Lucene [http://jakarta.apache.org/lucene/docs/index.html]
    - Popular off the shelf Java-based IR system
    - Based on heuristic retrieval models
- No IR system currently combines all of these features
Design Goals

- Robust retrieval model
  - Inference net + language modeling [Metzler and Croft ’04]

- Powerful query language
  - Extensions to INQUERIY query language driven by requirements of QA, web search, and XML retrieval
  - Designed to be as simple to use as possible, yet robust

- Off the shelf (Windows, *NIX, Mac platforms)
  - Separate download, compatible with Lemur
  - Simple to set up and use
  - Fully functional API w/ language wrappers for Java, etc…

- Scalable
  - Highly efficient code
  - Distributed retrieval
Comparing Collections

<table>
<thead>
<tr>
<th>Collection</th>
<th>CACM</th>
<th>WT10G</th>
<th>GOV2</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documents</td>
<td>3204</td>
<td>1.7 million</td>
<td>25 million</td>
<td>8 billion</td>
</tr>
<tr>
<td>Space</td>
<td>1.4 MB</td>
<td>10GB</td>
<td>426GB</td>
<td>80TB (?)</td>
</tr>
</tbody>
</table>
Outline

- Overview
- Retrieval Model
  - Model
  - Query Language
  - Applications
- System Architecture
- Evaluation
- Conclusions
<html>
<head>
<title>Department Descriptions</title>
</head>
<body>
The following list describes ...
<h1>Agriculture</h1> ...
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
<h1>Zoology</h1>
</body>
</html>
Model

- Based on original inference network retrieval framework [Turtle and Croft ’91]
- Casts retrieval as inference in simple graphical model
- Extensions made to original model
  - Incorporation of probabilities based on language modeling rather than tf.idf
  - Multiple language models allowed in the network (one per indexed context)
Model

\[ \theta_{\text{title}} \]
\[ \theta_{\text{body}} \]
\[ \theta_{\text{h1}} \]
\[ D \]
\[ \alpha, \beta_{\text{body}} \]
\[ \alpha, \beta_{\text{h1}} \]

\[ \alpha, \beta_{\text{title}} \]

\[ r_1 \ ...
\[ r_N \]

\[ q_1 \]
\[ q_2 \]

\[ l \]
\[ P(r | \theta) \]

- Probability of observing a term, phrase, or “concept” given a context language model
  - \( r_i \) nodes are binary
- Assume \( r \sim \text{Bernoulli}(\theta) \)
  - “Model B” – [Metzler, Lavrenko, Croft ’04]
- Nearly any model may be used here
  - \textit{tf.idf}-based estimates (INQUERY)
  - Mixture models
Prior over context language model determined by $\alpha$, $\beta$

Assume $P(\theta | \alpha, \beta) \sim \text{Beta}(\alpha, \beta)$
- Bernoulli’s conjugate prior
- $\alpha_w = \mu P(w | C) + 1$
- $\beta_w = \mu P(\neg w | C) + 1$
- $\mu$ is a free parameter

\[
P(r_i | \alpha, \beta, D) = \int_{\theta} P(r_i | \theta)P(\theta | \alpha, \beta, D) = \frac{tf_{w,D} + \mu P(w | C)}{|D| + \mu}
\]
Belief nodes are created dynamically based on query

Belief node CPTs are derived from standard link matrices
  - Combine evidence from parents in various ways
  - Allows fast inference by making marginalization computationally tractable

Information need node is simply a belief node that combines all network evidence into a single value

Documents are ranked according to:

\[ P(I | \alpha, \beta, D) \]
Example: #AND

| P(Q=true|a,b) | A  | B  |
|-------------|----|----|
| 0           | false | false |
| 0           | false | true |
| 0           | true  | false |
| 1           | true  | true |

\[
P_{\#\text{and}}(Q = true) = \sum_{a,b} P(Q = true| A = a, B = b)P(A = a)P(B = b) \\
= P(t|f, f)(1-p_A)(1-p_B) + P(t|f, t)(1-p_A)p_B + P(t|t, f)p_A(1-p_B) + P(t|t, t)p_Ap_B \\
= 0(1-p_A)(1-p_B) + 0(1-p_A)p_B + 0p_A(1-p_B) + 1p_Ap_B \\
= p_Ap_B
\]
Query Language

- Extension of INQUERY query language
- Structured query language
  - Term weighting
  - Ordered / unordered windows
  - Synonyms
- Additional features
  - Language modeling motivated constructs
  - Added flexibility to deal with fields via contexts
  - Generalization of passage retrieval (extent retrieval)
- Robust query language that handles many current language modeling tasks
## Terms

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stemmed term</td>
<td>dog</td>
<td>All occurrences of <em>dog</em> (and its stems)</td>
</tr>
<tr>
<td>Surface term</td>
<td>“dogs”</td>
<td>Exact occurrences of <em>dogs</em> (without stemming)</td>
</tr>
<tr>
<td>Term group (synonym group)</td>
<td>&lt;”dogs” canine&gt;</td>
<td>All occurrences of <em>dogs</em> (without stemming) or <em>canine</em> (and its stems)</td>
</tr>
<tr>
<td>Extent match</td>
<td>#any:person</td>
<td>Any occurrence of an extent of type <em>person</em></td>
</tr>
</tbody>
</table>
# Date / Numeric Fields

<table>
<thead>
<tr>
<th>Example</th>
<th>Example</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>#less</td>
<td>#less(URLDEPTH 3)</td>
<td>Any URLDEPTH numeric field extent with value less than 3</td>
</tr>
<tr>
<td>#greater</td>
<td>#greater(READINGLEVEL 3)</td>
<td>Any READINGLEVEL numeric field extent with value greater than 3</td>
</tr>
<tr>
<td>#between</td>
<td>#between(SENTIMENT 0 2)</td>
<td>Any SENTIMENT numeric field extent with value between 0 and 2</td>
</tr>
<tr>
<td>#equals</td>
<td>#equals(VERSION 5)</td>
<td>Any VERSION numeric field extent with value equal to 5</td>
</tr>
<tr>
<td>#date:before</td>
<td>#date:before(1 Jan 1900)</td>
<td>Any DATE field before 1900</td>
</tr>
<tr>
<td>#date:after</td>
<td>#date:after(June 1 2004)</td>
<td>Any DATE field after June 1, 2004</td>
</tr>
<tr>
<td>#date:between</td>
<td>#date:between(1 Jun 2000 1 Sep 2001)</td>
<td>Any DATE field in summer 2000.</td>
</tr>
</tbody>
</table>
## Proximity

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>#odN(e₁ ... eₘ) or #N(e₁ ... eₘ)</td>
<td>#od5(saddam hussein) #5(saddam hussein)</td>
<td>All occurrences of <em>saddam</em> and <em>hussein</em> appearing ordered within 5 words of each other</td>
</tr>
<tr>
<td>#uwN(e₁ ... eₘ)</td>
<td>#uw5(information retrieval)</td>
<td>All occurrences of <em>information</em> and <em>retrieval</em> that appear in any order within a window of 5 words</td>
</tr>
<tr>
<td>#uw(e₁ ... eₘ)</td>
<td>#uw(john kerry)</td>
<td>All occurrences of <em>john</em> and <em>kerry</em> that appear in any order within any sized window</td>
</tr>
<tr>
<td>#phrase(e₁ ... eₘ)</td>
<td>#phrase(#1(willy wonka) #uw3(chocolate factory))</td>
<td>System dependent implementation (defaults to #odm)</td>
</tr>
</tbody>
</table>
## Context Restriction

<table>
<thead>
<tr>
<th>Example</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>yahoo.title</td>
<td>All occurrences of <code>yahoo</code> appearing in the <em>title</em> context</td>
</tr>
<tr>
<td>yahoo.title,paragraph</td>
<td>All occurrences of <code>yahoo</code> appearing in both a <em>title</em> and <em>paragraph</em> contexts (may not be possible)</td>
</tr>
<tr>
<td><code>&lt;yahoo.title yahoo.paragraph&gt;</code></td>
<td>All occurrences of <code>yahoo</code> appearing in either a <em>title</em> context or a <em>paragraph</em> context</td>
</tr>
<tr>
<td>#5(apple ipod).title</td>
<td>All matching windows contained within a <em>title</em> context</td>
</tr>
</tbody>
</table>
## Context Evaluation

<table>
<thead>
<tr>
<th>Example</th>
<th>Evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>google.(title)</td>
<td>The term <em>google</em> evaluated using the <em>title</em> context as the document</td>
</tr>
<tr>
<td>google.(title, paragraph)</td>
<td>The term <em>google</em> evaluated using the concatenation of the <em>title</em> and <em>paragraph</em> contexts as the document</td>
</tr>
<tr>
<td>google.figure(paragraph)</td>
<td>The term <em>google</em> restricted to <em>figure</em> tags within the <em>paragraph</em> context.</td>
</tr>
</tbody>
</table>
## Belief Operators

<table>
<thead>
<tr>
<th>INQUERY</th>
<th>INDRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>#sum / #and</td>
<td>#combine</td>
</tr>
<tr>
<td>#wsum*</td>
<td>#weight</td>
</tr>
<tr>
<td>#or</td>
<td>#or</td>
</tr>
<tr>
<td>#not</td>
<td>#not</td>
</tr>
<tr>
<td>#max</td>
<td>#max</td>
</tr>
</tbody>
</table>

* #wsum is still available in INDRI, but should be used with discretion*
## Extent / Passage Retrieval

<table>
<thead>
<tr>
<th>Example</th>
<th>Evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#combine[section](dog canine)</code></td>
<td>Evaluates <code>#combine(dog canine)</code> for each extent associated with the <em>section</em> context</td>
</tr>
<tr>
<td><code>#combine[title, section](dog canine)</code></td>
<td>Same as previous, except is evaluated for each extent associated with either the <em>title</em> context or the <em>section</em> context</td>
</tr>
<tr>
<td><code>#combine[passage100:50](white house)</code></td>
<td>Evaluates <code>#combine(dog canine)</code> 100 word passages, treating every 50 words as the beginning of a new passage</td>
</tr>
<tr>
<td><code>#sum(#sum[section](dog))</code></td>
<td>Returns a single score that is the <code>#sum</code> of the scores returned from <code>#sum(dog)</code> evaluated for each <em>section</em> extent</td>
</tr>
<tr>
<td><code>#max(#sum[section](dog))</code></td>
<td>Same as previous, except returns the maximum score</td>
</tr>
</tbody>
</table>
Extent Retrieval Example

Query: #combine[section](dirichlet smoothing)

1. Treat each section extent as a “document”
2. Score each “document” according to #combine(…)
3. Return a ranked list of extents.

<table>
<thead>
<tr>
<th>SCORE</th>
<th>DOCID</th>
<th>BEGIN</th>
<th>END</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>IR-352</td>
<td>51</td>
<td>205</td>
</tr>
<tr>
<td>0.35</td>
<td>IR-352</td>
<td>405</td>
<td>548</td>
</tr>
<tr>
<td>0.15</td>
<td>IR-352</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
### Other Operators

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter require</td>
<td><code>#filreq( #less(READINGLEVEL 10) ben franklin) )</code></td>
<td>Requires that documents have a reading level less than 10. Documents then ranked by query <em>ben franklin</em></td>
</tr>
<tr>
<td>Filter reject</td>
<td><code>#filrej( #greater(URLDEPTH 1) microsoft) )</code></td>
<td>Rejects (does not score) documents with a URL depth greater than 1. Documents then ranked by query <em>microsoft</em></td>
</tr>
<tr>
<td>Prior</td>
<td><code>#prior(DATE )</code></td>
<td>Applies the document prior specified for the DATE field</td>
</tr>
</tbody>
</table>
Example Tasks

- Ad hoc retrieval
  - Flat documents
  - SGML/XML documents
- Web search
  - Homepage finding
  - Known-item finding
- Question answering
- KL divergence based ranking
  - Query models
  - Relevance modeling
Ad Hoc Retrieval

- Flat documents
  - Query likelihood retrieval:
    \[ q_1 \ldots q_N \equiv \text{combine}( q_1 \ldots q_N ) \]

- SGML/XML documents
  - Can either retrieve documents or extents
  - Context restrictions and context evaluations allow exploitation of document structure
Web Search

- Homepage / known-item finding
- Use mixture model of several document representations [Ogilvie and Callan ’03]

\[ P(w \mid \Theta, \Lambda) = \lambda_{\text{body}} P(w \mid \theta_{\text{body}}) + \lambda_{\text{inlink}} P(w \mid \theta_{\text{inlink}}) + \lambda_{\text{title}} P(w \mid \theta_{\text{title}}) \]

- Example query: Yahoo!
  
  \#combine( \#wsum(0.2 yahoo.(body)
                      0.5 yahoo.(inlink)
                      0.3 yahoo.(title)) ) )
More expressive passage- and sentence-level retrieval

Example:

- *Where was George Washington born?*

  `#combine[sentence]( #1( george washington )
  born #any:LOCATION )`

- Returns a ranked list of sentences containing the phrase George Washington, the term born, and a snippet of text tagged as a LOCATION named entity
**KL / Cross Entropy Ranking**

- INDRI handles ranking via KL / cross entropy
  - Query models [Zhai and Lafferty ’01]
  - Relevance modeling [Lavrenko and Croft ’01]

- **Example:**
  - Form user/relevance/query model $P(w \mid \theta_Q)$
  - Formulate query as:

  $$\text{#weight} \ (P(w_1 \mid \theta_Q) \ w_1 \ldots P(w_{|V|} \mid \theta_Q) \ w_{|V|})$$

- Ranked list equivalent to scoring by: $KL(\theta_Q \parallel \theta_D)$
- In practice, probably want to truncate
Outline

- Overview
- Retrieval Model
- System Architecture
  - Indexing
  - Query processing
- Evaluation
- Conclusions
System Overview

- Indexing
  - Inverted lists for terms and fields
  - Repository consists of inverted lists, parsed documents, and document vectors

- Query processing
  - Local or distributed
  - Computing local / global statistics

- Features
Repository Tasks

- Maintains:
  - inverted lists
  - document vectors
  - field extent lists
  - statistics for each field

- Store compressed versions of documents

- Save stopping and stemming information
Inverted Lists

- One list per term
- One list entry for each term occurrence in the corpus
- Entry: (termID, documentID, position)
- Delta-encoding, byte-level compression
  - Significant space savings
  - Allows index size to be smaller than collection
  - Space savings translates into higher speed
Inverted List Construction

- All lists stored in one file
  - 50% of terms occur only once
  - Single term entry = approximately 30 bytes
  - Minimum file size: 4K
  - Directory lookup overhead

- Lists written in segments
  - Collect as much information in memory as possible
  - Write segment when memory is full
  - Merge segments at end
Field Extent Lists

- Like inverted lists, but with extent information
- List entry
  - `documentID`
  - `begin` (first word position)
  - `end` (last word position)
  - `number` (numeric value of field)
Term Statistics

- Statistics for collection language models
  - total term count
  - counts for each term
  - document length

- Field statistics
  - total term count in a field
  - counts for each term in the field
  - document field length

- Example:
  - “dog” appears:
    - 45 times in the corpus
    - 15 times in a title field
    - Corpus contains 56,450 words
    - Title field contains 12,321 words
Query Architecture

- `runquery`
  - `QueryEnvironment`
    - `NetworkServerProxy`
    - `LocalServer`
  - `NetworkServerStub`
  - `LocalServer`

- `indrid`
  - `NetworkServerStub`
  - `LocalServer`
Query Processing

- Parse query
- Perform query tree transformations
- Collect query statistics from servers
- Run the query on servers
- Retrieve document information from servers
#combine( white house #1(white house) )
Query Optimization
Evaluation

Diagram:
- white
- house
- #1
- scorer
- scorer
- scorer
- #combine
- heap

Diagram represents a hierarchy of nodes with connections indicating relationships or processes.
Off the Shelf

- Indexing and retrieval GUIs
- API / Wrappers
  - Java
  - PHP
- Formats supported
  - TREC (text, web)
  - PDF
  - Word, PowerPoint (Windows only)
  - Text
  - HTML
Programming Interface (API)

- Indexing methods
  - open / create
  - addFile / addString / addParsedDocument
  - setStemmer / setStopwords

- Querying methods
  - addServer / addIndex
  - removeServer / removeIndex
  - setMemory / setScoringRules / setStopwords
  - runQuery / runAnnotatedQuery
  - documents / documentVectors / documentMetadata
  - termCount / termFieldCount / fieldList / documentCount
Outline

- Overview
- Retrieval Model
- System Architecture
- Evaluation
  - TREC Terabyte Track
  - Efficiency
  - Effectiveness
- Conclusions
TREC Terabyte Track

- Initial evaluation platform for INDRI
- Task: *ad hoc* retrieval on a web corpus
- Goals:
  - Examine how a larger corpus impacts current retrieval models
  - Develop new evaluation methodologies to deal with hugely insufficient judgments
Terabyte Track Summary

- **GOV2 test collection**
  - Collection size: 25,205,179 documents (426 GB)
  - Index size: 253 GB (includes compressed collection)
  - Index time: 6 hours (parallel across 6 machines) \( \sim \) 12GB/hr/machine
  - Vocabulary size: 49,657,854
  - Total terms: 22,811,162,783

- **Parsing**
  - No index-time stopping
  - Porter stemmer
  - Normalization (U.S. \( \Rightarrow \) US, etc…)

- **Topics**
  - 50 .gov-related standard TREC ad hoc topics
UMass Runs

- indri04QL
  - query likelihood
- indri04QLRM
  - query likelihood + pseudo relevance feedback
- indri04AW
  - phrases
- indri04AWRM
  - phrases + pseudo relevance feedback
- indri04FAW
  - phrases + fields
**indri04QL / indri04QLRM**

- **Query likelihood**
  - Standard query likelihood run
  - Smoothing parameter trained on TREC 9 and 10 main web track data
  - Example:
    ```
    #combine( pearl farming )
    ```

- **Pseudo-relevance feedback**
  - Estimate relevance model from top $n$ documents in initial retrieval
  - Augment original query with these term
  - Formulation:
    ```
    #weight( 0.5 #combine( Q_{ORIGINAL} )
    0.5 #combine( Q_{RM} ) )
    ```
Goal:
- Given only a title query, automatically construct an Indri query
- How can we make use of the query language?

Include phrases in query
- Ordered window (#N)
- Unordered window (#uwN)
Example Query

- prostate cancer treatment =>

#weight(
  1.5 prostate
  1.5 cancer
  1.5 treatment
  0.1 #1(prostate cancer )
  0.1 #1(cancer treatment )
  0.1 #1(prostate cancer treatment )
  0.3 #uw8(prostate cancer )
  0.3 #uw8(prostate treatment )
  0.3 #uw8(cancer treatment )
  0.3 #uw12(prostate cancer treatment )
)
Combines evidence from different fields

- Fields indexed: anchor, title, body, and header (h1, h2, h3, h4)

- Formulation:
  \[
  \texttt{#weight}(0.15 \texttt{ Q\_ANCHOR} \\
  0.25 \texttt{ Q\_TITLE} \\
  0.10 \texttt{ Q\_HEADING} \\
  0.50 \texttt{ Q\_BODY})
  \]

- Needs to be explore in more detail
### MAP

<table>
<thead>
<tr>
<th>fields -&gt;</th>
<th>T</th>
<th>TD</th>
<th>TDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>QL</td>
<td>0.2565</td>
<td>0.2730</td>
<td>0.3088</td>
</tr>
<tr>
<td>QLRM</td>
<td>0.2529</td>
<td>0.2675</td>
<td>0.2928</td>
</tr>
<tr>
<td>AW</td>
<td>0.2839</td>
<td>0.2988</td>
<td>0.3293</td>
</tr>
<tr>
<td>AWRM</td>
<td>0.2874</td>
<td>0.2974</td>
<td>0.3237</td>
</tr>
</tbody>
</table>

**Indri Terabyte Track Results**

T = title  
D = description  
N = narrative

### P10

<table>
<thead>
<tr>
<th>fields -&gt;</th>
<th>T</th>
<th>TD</th>
<th>TDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>QL</td>
<td>0.4980</td>
<td>0.5510</td>
<td>0.5918</td>
</tr>
<tr>
<td>QLRM</td>
<td>0.4878</td>
<td>0.5673</td>
<td>0.5796</td>
</tr>
<tr>
<td>AW</td>
<td>0.5857</td>
<td>0.6184</td>
<td>0.6306</td>
</tr>
<tr>
<td>AWRM</td>
<td>0.5653</td>
<td>0.6102</td>
<td>0.6367</td>
</tr>
</tbody>
</table>

*italicized* values denote statistical significance over QL
Conclusions

- INDRI extends INQUERY and Lemur
  - Off the shelf
  - Scalable
- Geared towards tagged (structured) documents
- Employs robust inference net approach to retrieval
- Extended query language can tackle many current retrieval tasks
- Competitive in both terms of effectiveness and efficiency
Questions?

Contact Info
Email: metzler@cs.umass.edu
Web: http://ciir.cs.umass.edu/~metzler