INDRI - Overview

Don Metzler
Center for Intelligent Information Retrieval
University of Massachusetts, Amherst
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
  - Query language
- Lemur [http://www-2.cs.cmu.edu/~lemur/]
  - 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

- 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…

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

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

- Scalable
  - Highly efficient code
  - Distributed retrieval
TREC Terabyte Track

- Initial evaluation metric 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
- Data:
  - .GOV2 collection (crawl of entire .gov domain)
  - 25 million documents (html, pdf, ps, Word, text)
  - Not even half of a TB of documents (426 GB)
<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

\[ \theta_{\text{title}} \rightarrow r_1, \ldots, r_N \]

\[ \theta_{\text{body}} \rightarrow r_1, \ldots, r_N \]

\[ \theta_{h1} \rightarrow r_1, \ldots, r_N \]

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

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

\[ \alpha, \beta_{h1} \]

\[ q_1 \rightarrow l \]

\[ q_2 \rightarrow l \]
\[ P( r | \theta ) \]

- 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
  - \( tf.idf \)-based estimates (INQUERY)
  - Mixture models
Model

\[ P = \ldots \]

\[ D = \alpha, \beta_{\text{body}} \]

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

\[ \theta_{\text{title}} \]

\[ \theta_{\text{body}} \]

\[ \theta_{h1} \]

\[ r_1 \ldots r_N \]

\[ q_1 \]

\[ l \]

\[ \alpha, \beta_{h1} \]
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

$$P(r_i | \alpha, \beta, D) = \int P(r_i | \theta)P(\theta | \alpha, \beta, D) = \frac{tf_{w,D} + \mu P(w | C)}{|D| + \mu}$$
Model

\[ \theta_{\text{title}} \rightarrow r_1 \ldots r_N \]

\[ \theta_{\text{body}} \rightarrow r_1 \ldots r_N \]

\[ \theta_{\text{h1}} \rightarrow r_1 \ldots r_N \]

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

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

\[ \alpha, \beta_{\text{h1}} \]

\[ q_1 \rightarrow q_2 \rightarrow l \]
\[ P(q | r) \text{ and } P(l | r) \]

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

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

\[
P_{\text{\#and}}(Q = \text{true}) = \sum_{a,b} P(Q = \text{true} | A = a, B = b)P(A = a)P(B = b)
\]

\[
= P(t \mid f, f) (1-p_A)(1-p_B) + P(t \mid f, t)(1-p_A)p_B + P(t \mid t, f)p_A(1-p_B) + P(t \mid t, t)p_Ap_B
\]

\[
= 0(1-p_A)(1-p_B) + 0(1-p_A)p_B + 0p_A(1-p_B) + 1p_Ap_B
\]

\[
= p_Ap_B
\]
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;&quot;dogs” canine&gt;</td>
<td>All occurrences of <em>dogs</em> (without stemming) or <em>canine</em> (and its stems)</td>
</tr>
<tr>
<td>POS qualified term</td>
<td>&lt;&quot;dogs” canine&gt;.NNS</td>
<td>Same as previous, except matches must also be tagged with the <em>NNS</em> POS tag</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(dog cat) or #5(dog cat)</td>
<td>All occurrences of <em>dog</em> and <em>cat</em> appearing ordered within a window of 5 words</td>
</tr>
<tr>
<td>#uwN(e₁ … eₘ)</td>
<td>#uw5(dog cat)</td>
<td>All occurrences of <em>dog</em> and <em>cat</em> that appear in any order within a window of 5 words</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>
<tr>
<td>#syntax:xx(e₁ … eₘ)</td>
<td>#syntax:np(fresh powder)</td>
<td>System dependent implementation</td>
</tr>
</tbody>
</table>
## Context Restriction

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog.(title)</td>
<td>The term <em>dog</em> evaluated using the <em>title</em> context as the document</td>
</tr>
<tr>
<td>dog.(title, paragraph)</td>
<td>The term <em>dog</em> evaluated using the concatenation of the <em>title</em> and <em>paragraph</em> contexts as the document</td>
</tr>
<tr>
<td>dog.figure(paragraph)</td>
<td>The term <em>dog</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 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>#sum(#sum[section](dog))</code></td>
<td>Returns a single score that is the #sum 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*.

**SCORE** | **DOCID** | **BEGIN** | **END**
--- | --- | --- | ---
0.50 | IR-352 | 51 | 205
0.35 | IR-352 | 405 | 548
0.15 | IR-352 | 0 | 50
... | ... | ... | ...
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
- Cluster-base language models
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]

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

- Example query: Yahoo!
  
  \#combine( \#wsum(0.2 yahoo.(body) \\
  0.5 yahoo.(inlink) \\
  0.3 yahoo.(title) ) )
Question Answering

- More expressive passage- and sentence-level retrieval

Example:

- Where was George Washington born?
  
  \#combine[sentence]( \#1( george washington )
  born \#place( ? ) )

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

- INDRI handles ranking via KL / cross entropy
  - Query models
  - Relevance modeling
- Relevance modeling [Lavrenko and Croft ’01]
  - Do initial query run and retrieve top $N$ documents
  - Form relevance model $P(w | \theta_Q)$
  - Formulate and rerun expanded query as:

$$\#weight \ (P(w_1 \mid \theta_Q) \ w_1 \ ... \ P(w_{|\mathcal{V}|} \mid \theta_Q) \ w_{|\mathcal{V}|})$$

- Ranked list equivalent to scoring by: $KL(\theta_Q \parallel \theta_D)$
Cluster-Based LMs

- Cluster-based language models [Liu and Croft ’04]
- Smoothes each document with a document-specific cluster model
  \[ P(w | \Theta, \Lambda) = \lambda_D P(w | \theta_D) + (1 - \lambda_D) [\lambda_{\text{Cluster}} P(w | \text{Cluster}) + (1 - \lambda_{\text{Cluster}}) P(w | C)] \]
- INDRI allows each document to specify (via metadata) a background model to smooth against
- Many other possible uses of this feature
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