Retrieval and Feedback Models for Blog Distillation

CMU at the TREC 2007 Blog Track

Jonathan Elsas, Jaime Arguello, Jamie Callan, Jaime Carbonell
CMU’s Blog Distillation Focus

• Two Research Questions:
  
  What is the appropriate unit of retrieval? Feeds or Entries?
  
  How can we effectively do pseudo-relevance feedback for Blog Distillation?

• Our four submissions investigate these two dimensions.
Outline

• Corpus Preprocessing Tasks
• Two Feedback Models
• Two Retrieval Models
• Results
• Discussion
Corpus Preprocessing

- Used only FEED documents (vs. PERMALINK or HOMEPAGE documents).
- For each FEEDNO, extracted each new entry across all feed fetches.
- Aggregated into 1-document-per-FEEDNO, retaining structural elements from the FEED documents:
  - Title, description, entry, entry title, etc…
- Very helpful: Python Universal Feed Parser
Two Pseudo-Relevance Feedback Models

• Indri’s Built in Pseudo-Relevance Feedback (Lavrenko’s Relevance Model)
  – Using Metzler’s Dependence Model query on the full feed documents
  – Produces weighted unigram PRF query,
    \[ Q_{RM} = \#weight(w_1 \ t_1 \ w_2 \ t_2 \ ... \ w_{50} \ t_{50}) \]

• Wikipedia-based Pseudo-Relevance Feedback
  – Focus on anchor text linking to highly ranked documents wrt baseline BOW query
Wikipedia-based PRF

BOW query → Wikipedia
Wikipedia-based PRF

Relevance Set (top T)

BOW query

Working Set (top N)
Wikipedia-based PRF

Relevance Set (top T)

Working Set (top N)
Wikipedia-based PRF

\[ Q_{\text{wiki}} = \#\text{weight}( w_1 \text{ a-text}_1, w_2 \text{ a-text}_2, \ldots, w_{20} \text{ a-text}_{20} ) \]

where \( w_i \sim \text{sum}( T - \text{rank}(\text{target}(a_i)) ) \)
Wikipedia-based PRF

Query: “Apple iPod”

iPod mini

From Wikipedia, the free encyclopedia

The iPod mini is a mid-range iPod digital audio player designed and marketed by Apple Inc. It was announced on January 6, 2004, and released on February 20 of the same year; a second-generation version was announced on February 23, 2005. The device operates on Macintosh and Windows PCs, and has limited third-party support for Linux and other Unix workalikes. The iPod mini line was officially discontinued on September 7, 2005 and replaced by the iPod nano line.

The iPod mini retained the touch-sensitive scroll wheel of the third generation iPod; however, instead of the four touch buttons located above the wheel, the buttons were made mechanical beneath the wheel itself—hence the name click wheel. To use one of the four buttons, the user must physically push the edge of the wheel inward over one of the four labels. Like its predecessors, the wheel was developed for Apple by Synaptics. The click wheel is now also used in the fourth, fifth and sixth generation iPods and the iPod nano, from first generation through to third; however, in the nano and 5G iPods onwards, the clickwheel is developed by Apple.

Above the wheel is a monochrome LCD that displays a menu or information about the selected track. Newer-generation iPods have since adopted color displays.
Two Pseudo-Relevance Feedback Models

Q 983 “Photography”

<table>
<thead>
<tr>
<th>Wikipedia PRF</th>
<th>Indri’s Relevance Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>photography</td>
<td>photography</td>
</tr>
<tr>
<td>photographer</td>
<td>aerial</td>
</tr>
<tr>
<td>depth of field</td>
<td>digital</td>
</tr>
<tr>
<td>camera</td>
<td>full</td>
</tr>
<tr>
<td>photograph</td>
<td>resource</td>
</tr>
<tr>
<td>pulitzer prize</td>
<td>stock</td>
</tr>
<tr>
<td>digital camera</td>
<td>free</td>
</tr>
<tr>
<td>photographic film</td>
<td>information</td>
</tr>
<tr>
<td>photojournalism</td>
<td>art</td>
</tr>
<tr>
<td>cinematography</td>
<td>wedding</td>
</tr>
<tr>
<td>shutter speed</td>
<td>great</td>
</tr>
</tbody>
</table>
## Two Pseudo-Relevance Feedback Models

**Q 995 “Ruby on Rails”**

<table>
<thead>
<tr>
<th>Wikipedia PRF</th>
<th>Indri’s Relevance Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>pokemon</td>
<td>rail</td>
</tr>
<tr>
<td>ruby</td>
<td>ruby</td>
</tr>
<tr>
<td>pokmon ruby and sapphire</td>
<td>kakutani</td>
</tr>
<tr>
<td>pokmon emerald</td>
<td>que</td>
</tr>
<tr>
<td>ruby programming language</td>
<td>article</td>
</tr>
<tr>
<td>php</td>
<td>weblog</td>
</tr>
<tr>
<td>pokmon firered and leafgreen</td>
<td>activestate</td>
</tr>
<tr>
<td>pokmon adventures</td>
<td>rubyonrail</td>
</tr>
<tr>
<td>pokmon video games</td>
<td>new</td>
</tr>
<tr>
<td>standard gauge</td>
<td>develop</td>
</tr>
<tr>
<td>tram</td>
<td>dontstopmusic</td>
</tr>
<tr>
<td>pokmon</td>
<td></td>
</tr>
<tr>
<td>programming language</td>
<td></td>
</tr>
<tr>
<td>language</td>
<td></td>
</tr>
<tr>
<td>develop</td>
<td></td>
</tr>
<tr>
<td>dontstopmusic</td>
<td></td>
</tr>
</tbody>
</table>
Two Retrieval Models

• Large Document model
  – Entire Feed is the unit of retrieval

• Small Document model
  – Individual entry is the unit of retrieval
  – Ranked Entries are aggregated into a Feed Ranking
Large Document model

Feed Document Collection
Large Document model

Feed Document Collection
Large Document model

Feed Document Collection

Ranked Feeds

Q
Small Document model

Entry Document
Collection
Small Document model

Collection

Entry Document

Q
Small Document model

Entry Document Collection

Q

Ranked Entries
Small Document model

Entry Document Collection

Ranked Entries

Q

Ranked Feeds
Large Document Model

• Language Modeling retrieval model using the feed document structure.

• Features used for Large Document model:
  - $P(\text{Feed} \mid Q_{\text{feed-title}})$
  - $P(\text{Feed} \mid Q_{\text{entry-text}})$
  - $P(\text{Feed} \mid Q_{\text{RM}})$
  - $P(\text{Feed} \mid Q_{\text{wiki}})$
Large Document Model

• Language Modeling retrieval model using the feed document structure.

Large Document Indri Query:

\[ \#weight( \lambda_{\text{title}} D_{\text{title}} \lambda_{\text{entry}} D_{\text{entry}} \lambda_{\text{RM}} Q_{\text{RM}} \lambda_{\text{wiki}} Q_{\text{wiki}} ) \]

- \( P(\text{Feed} \mid Q_{\text{RM}}) \)
- \( P(\text{Feed} \mid Q_{\text{wiki}}) \)
Small Document Model

- Feed ranking in blog distillation is analogous to resource ranking in federated search
  Feed $\sim$ Resource
  Entry $\sim$ Document

- Created sampled collection
  - Sampled (with replacement) 100 entries from each feed
  - Control for dramatically different feed lengths
Small Document Model

- **Feed ranking in blog distillation** is analogous to **resource ranking in federated search**

- **Created sampled collection**
  - Sampled (with replacement) 100 entries from each feed
  - Control for dramatically different feed lengths
Small Document Model

Adapted Relevant Document Distribution Estimation (ReDDE) resource ranking.

ReDDE: well-known state-of-the-art federated search algorithm

\[ \hat{R}el_q(j) = \sum_{d_i \in C_j} P(rel|d_i)P(d_i|C_j)NC_j \]
Small Document Model

Adapted Relevant Document Distribution Estimation (ReDDE) resource ranking.

Assuming uniform prior, equal feed length:

$$\hat{Rel}_q(j) = \sum_{d_i \in C_j} P(\text{rel}|d_i)P(d_i|C_j)NC_j$$
Small Document Model

Adapted Relevant Document Distribution Estimation (ReDDE) resource ranking.

Assuming uniform prior, equal feed length:

\[ \hat{\text{Rel}}_q(j) = \sum_{d_i \in C_j} P(\text{rel}|d_i) \]
Small Document Model

• Features used in the small document model:
  – \( P(\text{Feed} \mid Q_{\text{entry-text}}) \)
  – \( P(\text{Feed} \mid Q_{\text{RM}}) \)
  – \( P(\text{Feed} \mid Q_{\text{wiki}}) \)
Small Document Model

• Features used in the small document model:

Small Document Indri Query:

\[
\#wsum(1.0 \#combine[entry](
\#weight(\lambda_{entry}DM_{entry}
\lambda_{RM}Q_{RM}
\lambda_{wiki}Q_{wiki}) ))
\]
Parameter Setting

- Selecting feature weights (\(\lambda\)'s) required training data
- Relevance judgments produced for a small subset of the queries (6+2)
  - BOW title query, 50 docs judged/query
- Simple grid search to choose parameters that maximized MAP
Results

<table>
<thead>
<tr>
<th>Run</th>
<th>MAP</th>
<th>R-prec</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMUfeed</td>
<td>0.3385</td>
<td>0.4087</td>
<td>0.4733</td>
</tr>
<tr>
<td>CMUfeedW</td>
<td>0.3695</td>
<td>0.4245</td>
<td>0.5356</td>
</tr>
<tr>
<td>CMUentry</td>
<td>0.2453</td>
<td>0.3277</td>
<td>0.4089</td>
</tr>
<tr>
<td>CMUentryW</td>
<td>0.2552</td>
<td>0.3384</td>
<td>0.4267</td>
</tr>
</tbody>
</table>

Results

Queries (ordered by CMUfeedW performance)
Feature Elimination

MAP

-All
-DM
-W
-W-PRF
-W-PRF-DM

CMUfeedW

CMUfeed

+Training

-Training
Conclusions

• What worked
  – Preprocessing the corpus & using only feed XML
  – Simple retrieval model with appropriate features
  – Wikipedia expansion
  – (small amount of) training data

• What didn’t  (… or what we tried without success)
  – Small Document Model (but we think it can)
  – Spam/Splog detection, anchor text, URL segmentation
Thank You
## Feature Elimination

<table>
<thead>
<tr>
<th>Configuration</th>
<th>+Training</th>
<th>-Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>+All (CMUfeedW)</td>
<td>0.3695</td>
<td>0.3536</td>
</tr>
<tr>
<td>-Dependence Model</td>
<td>0.3676</td>
<td>0.3457</td>
</tr>
<tr>
<td>-Wiki (CMUfeed)</td>
<td>0.3385</td>
<td>0.3152</td>
</tr>
<tr>
<td>-Wiki, -Indri’s PRF</td>
<td></td>
<td>0.2989</td>
</tr>
<tr>
<td>-Wiki, -Indri’s PRF, Dep Model</td>
<td></td>
<td>0.2907</td>
</tr>
</tbody>
</table>