Results from an NSF SGER Grant

Topic Partitioned Search Engine Indexes

Jamie Callan and Anagha Kulkarni
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{callan, anaghak}@cs.cmu.edu
Our Project Has Two Parts

Creation of the ClueWeb09 dataset
• 1 billion high PageRank web documents
  – 25 terabytes
• Distributed to 60+ research groups around the world
… more details this evening at our poster

The index for ClueWeb09 is too big to fit on a single machine
• How do we search it?
Standard Practice:
Tiered Indexes

The index is divided into tiers (e.g., 2)
• The “good” documents go in Tier 1
  – High PageRank, recent clicks, low p(spam), home pages, ...
• Everything else goes in Tier 2

When a new query arrives
• Search Tier 1
• If necessary, also search Tier 2

This helps, but for big collections, tiers are still very big
Standard Practice: Document-Partitioned Indexes

The index is divided into partitions ("shards")

- Each document is assigned to a partition
  - E.g., 25 partitions × 1 TB each
- Each partition is assigned to a machine

When a new query arrives

- Search each partition in parallel
- Merge the results

So, I need a computer cluster ($$$)

© 2009, Jamie Callan
A Related Problem: Federated Search

Some environments have many distinct search engines
• Thomson-Reuters, Government Printing Office, …
  – Dozens or hundreds
• It is impractical to search every engine for every query
  – Efficiency, accuracy, access, cost ($), …

Resource selection algorithms
• Given a set of search engines and a query
  … pick the ones that contain the relevant documents
• E.g., vGLOSS, CORI, ReDDE, RELAX, …
A Related Problem:
Federated Search

Most prior research studied **uncooperative environments**
- Search engines controlled by different organizations
- Legacy search engines

A **partitioned search engine** can be viewed as
a **highly cooperative** form of federated search
- How does that change the problem?
Lessons Learned From Prior Research

It is easier to select the right search engines when partitions are organized by topic than when partitions are organized chronologically

• It is easy to distinguish between sports and politics
• It is hard to distinguish between March and April

This is consistent with the Cluster Hypothesis

“Closely associated documents tend to be relevant to the same requests” – van Rijsbergen
A New Approach: Topic-Partitioned Tiers

The tier index is divided into partitions ("shards")
• Each partition is defined by a ‘topic’
• Each document is assigned to a partition / topic
  – E.g., 25 partitions × 1 TB each
• Each machine gets multiple partitions
  – Disks are inexpensive

When a new query arrives
• Select which partition(s) to search (resource selection)
• Search selected partition(s)
  – In parallel or sequentially
• Merge the results
Defining Topics

What determines a good set of topics?
• Disjoint (more-or-less)
• Easily defined and recognized

Topics are probably corpus-specific

Methods investigated
• Latent Dirichlet Allocation (LDA)
• k-means clustering (recent)
Defining Topics: LDA

LDA is impractical for collections of any interesting size

- \( O(DL^2T) \), where \( D \): #docs, \( L \): avg doc len, \( T \): #topics
  - We use \( T=100 \) for now (100 partitions), but this is arbitrary

LDA can be applied to a sample of \( D \)

- How big a sample is needed?
- Are OOV words a problem?

Samples of 25-30k documents works well for 100 topics
## LDA Topics

### 100 topics for a 5 million document subset of gov2

**Top terms from topic models for Gov2-Rel dataset**

<table>
<thead>
<tr>
<th>Topic A</th>
<th>Topic B</th>
<th>Topic C</th>
<th>Topic D</th>
<th>Topic E</th>
<th>Topic F</th>
<th>Topic G</th>
</tr>
</thead>
<tbody>
<tr>
<td>food</td>
<td>program</td>
<td>construct</td>
<td>health</td>
<td>waste</td>
<td>court</td>
<td>habitat</td>
</tr>
<tr>
<td>usda</td>
<td>resource</td>
<td>city</td>
<td>care</td>
<td>hazardous</td>
<td>state</td>
<td>species</td>
</tr>
<tr>
<td>product</td>
<td>national</td>
<td>project</td>
<td>service</td>
<td>facility</td>
<td>order</td>
<td>area</td>
</tr>
<tr>
<td>animal</td>
<td>research</td>
<td>build</td>
<td>medical</td>
<td>environmental</td>
<td>case</td>
<td>wildlife</td>
</tr>
<tr>
<td>farm</td>
<td>develop</td>
<td>new</td>
<td>hospital</td>
<td>material</td>
<td>respondent</td>
<td>critic</td>
</tr>
<tr>
<td>agricultural</td>
<td>manage</td>
<td>area</td>
<td>medicare</td>
<td>recycle</td>
<td>union</td>
<td>plant</td>
</tr>
<tr>
<td>commodity</td>
<td>contractor</td>
<td>contractor</td>
<td>office</td>
<td>epa</td>
<td>decision</td>
<td>bird</td>
</tr>
<tr>
<td>import</td>
<td>develop</td>
<td>municipal</td>
<td>nurse</td>
<td>clean</td>
<td>agree</td>
<td>populate</td>
</tr>
<tr>
<td>trade</td>
<td>project</td>
<td>house</td>
<td>facility</td>
<td>pollution</td>
<td>party</td>
<td>land</td>
</tr>
<tr>
<td>state</td>
<td>public</td>
<td>facility</td>
<td>physician</td>
<td>manage</td>
<td>section</td>
<td>endanger</td>
</tr>
<tr>
<td>milk</td>
<td>university</td>
<td>pay</td>
<td>prevention</td>
<td>prevention</td>
<td>act</td>
<td>designate</td>
</tr>
<tr>
<td>export</td>
<td>meet</td>
<td>plan</td>
<td>disposal</td>
<td>disposal</td>
<td>violate</td>
<td>conservation</td>
</tr>
<tr>
<td>meat</td>
<td>science</td>
<td>land</td>
<td>product</td>
<td>product</td>
<td>file</td>
<td>native</td>
</tr>
<tr>
<td>organic</td>
<td>plan</td>
<td>work</td>
<td>social</td>
<td>train</td>
<td>action</td>
<td>service</td>
</tr>
<tr>
<td>market</td>
<td>fund</td>
<td>future</td>
<td>assist</td>
<td>contain</td>
<td>law</td>
<td>nest</td>
</tr>
</tbody>
</table>
Resource Selection

Given a query, rank partitions by likelihood of satisfying the query if that partition is searched.

There are many good algorithms

• We work with CORI and ReDDE

How many partitions should be selected?

• This is an open research problem
• Usually a static number is chosen (3%, 5%, 10%)
• We study the effect of different choices
Preliminary Results

Searching the best $O(10\%)$ of the partitions gives accuracy comparable to searching everything

• $P@10$, $P@20$, $P@30$, $P@50$

• Search more partitions to get better Recall

• Reasonably robust
Most Queries Produce The Same Ranking

Not surprising, if the right partitions are selected

10-20% do not
• Lower in recent work

Mostly ‘poor’ queries?
Preliminary Results

LDA produces partitions of different sizes

• Does that increase search costs?
  – On average, no
  – It does increase variance
**Very Preliminary Results**

**k-Means > LDA**

- More efficient for forming partitions
- Fewer partitions searched
  - More accurate $\rightarrow$ more efficient
Conclusion

It is not necessary to search the whole corpus (or tier) …when the corpus (or tier) is large …and the goal is Precision-oriented search

• Topic-oriented partitions permit selective search

Key issues

• How to define topics
• How to assign documents to partitions
• How many partitions to search for a given query
What Next?

Our work so far just scratches the surface
• There are many unexplored research problems

We would like to make this default behavior for the Lemur Toolkit’s Indri search engine
• Out of the box support for massive corpora
  – Without having to use an expensive computer cluster
  – Enable scientists to work with more realistic corpora

Perhaps this is useful for industry, too?