Parallel and Distributed IR

Lecture 16
Parallel Information Retrieval

- Scale retrieval to immense collections using a parallel computer
- Distribute terms across nodes
  - Spread terms across nodes so that each node does equal work
- Compute document scores in parallel
  - Each node computes partial document scores
  - Scores are summed and normalized at end
Parallel indexing – basic

- Basic inversion algorithms are parallelizable
- Partitioning
  - divide documents among nodes for processing
  - problem: still need to distribute terms
- Sort-based
  - Process documents at central (master) node
  - Rather than sorting, send tuples to nodes
  - Problem: either need to know ahead of time where terms should go, or redistribute terms at end
Parallel indexing – 2

- (Gravano and Garcia-Molina paper)
(Centralized) Cosine algorithm

1. $A = \emptyset$ (set of accumulators for documents)
2. For each query term $t$
   - Get term, $f_{t}$, and address of $I_{t}$ from lexicon
   - Set $idf_{t} = \log(1 + N/f_{t})$
   - Read inverted list $I_{t}$
   - For each $<d, f_{d,t}>$ in $I_{t}$
     - If $A_{d} \notin A$, initialize $A_{d}$ to 0 and add it to $A$
     - $A_{d} = A_{d} + (1 + \log(f_{d,t})) \times idf_{t}$ (assumes query term weight is 1/0)
3. For each $A_{d}$ in $A$, $A_{d} = A_{d}/W_{d}$
4. Fetch and return top $r$ documents to user
Parallelizing the Cosine Algorithm

- At the master node,
  - Get \( f_t \) and node address for each query term
  - Send \( <t, f_t> \) to compute node for term \( t \)

- At compute node,
  - Accumulate (partial) document scores for each query term \( t \) housed at this node

- At master node,
  - Merge document scores (gather operation)
  - Apply doc-length normalization and return top \( n \)
Further scaling

- Parallel algorithm is slow
  - 1 disk access < 1 network msg. + 1 disk access
- To make this faster:
  - Compute nodes should hold subindex in memory
  - Terms should be replicated on several nodes
    - Query terms can be routed randomly to any node housing that term
  - Cache query results at the master for common queries
Parallel File System approach

- A parallel file system distributes files transparently across a network
- RAMA: RAID over a network
  - Data striped across disks on network nodes
  - Network has to be fast
  - SAN architectures such as fibre-channel fabrics
- RAMA-IR
  - Index is one large file striped across the network
  - Processing can be centralized or with multiple processes sharing the index
Distributed Information Retrieval

- Retrieval across distinct collections
  - Separated by topic, origin, publisher, date, ...
  - Local or spread over the Internet
  - AKA metasearch

- Three problems
  - Collection representation
  - Collection selection
  - Results merging
Primary DIR References

- University of Massachusetts
  - Based on INQUERY work (Turtle and Croft)
  - Jamie Callan (1995-2000)
- University of Virginia
  - French and Viles
- Stanford
  - Gravano and Garcia-Molina
DIR Testbeds

- Early testbeds were small by today's standards
- TREC-based testbeds
  - Divide TREC collections by source and date
  - Usually TREC CD's 1-3, or VLC (20GB)
  - Some recent work using TREC Web collections
- Characteristics
  - Collections more homogeneous than the testbed
  - Collections diverse enough to make selection interesting
  - Main testbeds: 100, 236, and 921 collections
Representing Collections

- Manual representations
  - Source metadata, hand-written descriptions, cataloguing information
- Unigram language models
  - Frequency of each term in the collection
- Relevance models
  - Learned from relevance feedback
Unigram Language Models

- A vector representation of a collection
- Usually document frequency (df) values
- Centroids – average weight vector
- More sophisticated language models
  - Smoothing
  - Bigrams
Acquiring the representation

- Cooperatively
  - Systems send a representation upon request
  - STARTS protocol (Gravano et al. 1996)
- Query-based sampling
  - Initial query: one random word
  - Initial model built from top 2-8 documents
  - Next round: select a random word from the current model
    - Works better than frequency-guided heuristics
Collection Selection

- Given a query, rank the collections
- Optimal ranking is by the number of relevant documents in each collection
- Goal: send query to as few collections as possible
- Common measure

\[ R(n) = \frac{\sum_{i=1}^{n} rg_i}{\sum_{i=1}^{n} rd_i} \]
CORI ranking

\[
T = \frac{df}{df + 50 + 150 \cdot cw/\text{avg.cw}}
\]

\[
I = \log \left( \frac{C + 0.5}{\text{cf}} \right) / \log (C + 1.0)
\]

\[
p(r_k | R_i) = b + (1 - b) \cdot T \cdot I
\]

- df = number of docs containing term
- cw = number of terms in collection; avg.cw is average cw
- C = number of collections
- cf = number of collections containing term
- b is a minimum belief component, usually 0.4
Combining CORI weights

- **INQUERY operators** \( ( p_j = p(r_j | R_i) ) \)

\[
\begin{align*}
\text{bel}_{\text{sum}}(Q) &= \frac{(p_1 + p_2 + \ldots + p_n)}{n} \\
\text{bel}_{\text{wsum}}(Q) &= \frac{(w_1 p_1 + w_2 p_2 + \ldots + w_n p_n) w_q}{(w_1 + w_2 + \ldots + w_n)} \\
\text{bel}_{\text{not}}(Q) &= 1 - p_1 \\
\text{bel}_{\text{or}}(Q) &= 1 - (1 - p_1) \cdot \ldots \cdot (1 - p_n) \\
\text{bel}_{\text{and}}(Q) &= p_1 \cdot p_2 \cdot \ldots \cdot p_n
\end{align*}
\]
Merging results

- Document scores are not comparable between collections
  - local document frequencies
  - completely different retrieval model?
- Collections may have documents in common
- We may not have control over, or even understanding of the collections' search mechanism
CORI Merging

- CORI approach: score normalization
  - scale document scores by collection scores
  - scale range of possible collection scores to [0,1]
- $R_{\text{max}} = \text{CORI score with (}T = 1\text{)}$
- $R_{\text{min}} = \text{CORI score with (}T = 0\text{)}$

\[
R_i' = \frac{R_{\text{max}} - R_i}{R_{\text{max}} - R_{\text{min}}} \\
D' = \frac{D + 0.4 \cdot D \cdot R_i'}{1.4}
\]
CORI Merging (2)

- Problem: assumes document score distributions are “reasonable”
  - If collections are divided by topic, then IDF values can be highly skewed between collections
  - Solution: rescale document scores also

\[
R_i' = \frac{(R_{\text{max}} - R_i)}{(R_{\text{max}} - R_{\text{min}})}
\]

\[
D' = \frac{(D_{\text{max}_i} - D)}{(D_{\text{max}_i} - D_{\text{min}_i})}
\]

\[
D'' = \frac{D' + 0.4 \cdot D' \cdot R_i'}{1.4}
\]