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

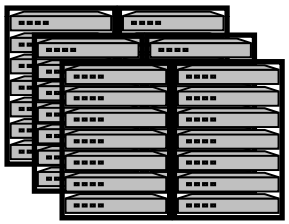
- Search Tier 1
- If necessary, also search Tier 2

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 ($$$)

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

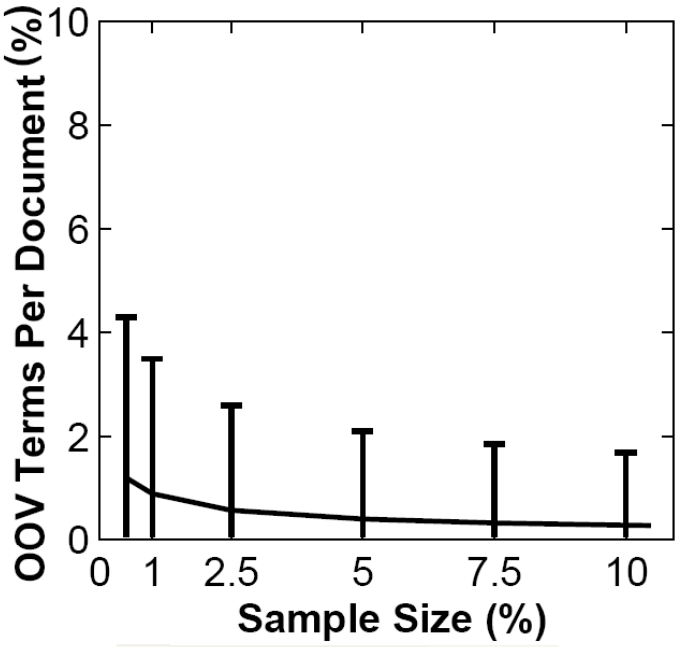
LDA is impractical for collections of any interesting size

- $O(DL^2T)$, 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

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

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

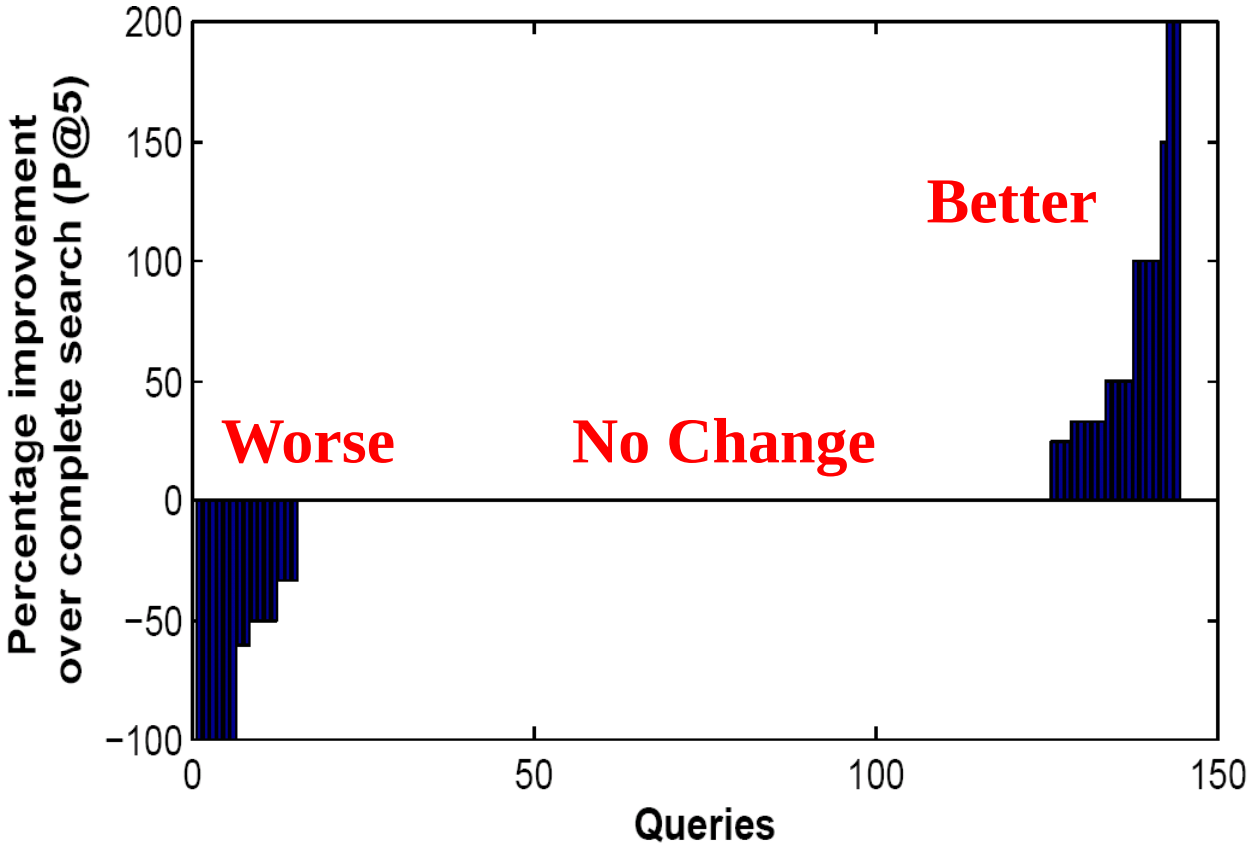
Most Queries Produce The Same Ranking

Not surprising, if the right partitions are selected

10-20% do not

- Lower in recent work

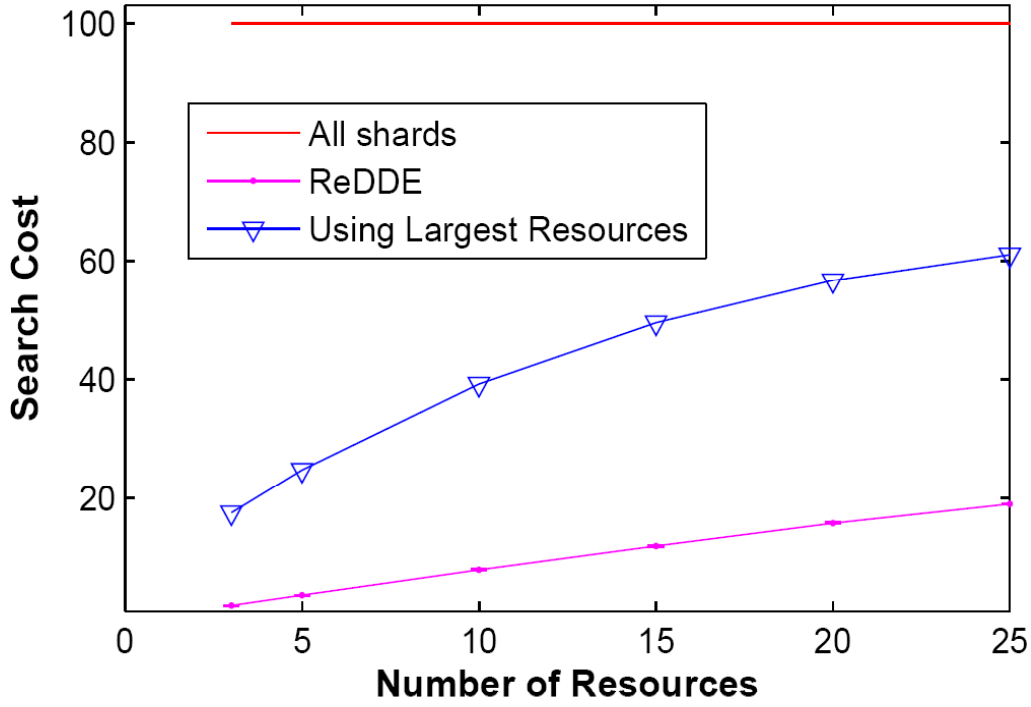
Mostly 'poor' queries?



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 → 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?