HathiTrust Large Scale Search: Scalability meets Usability

www.hathitrust.org

Tom Burton-West
Information Retrieval Programmer
Digital Library Production Service
University of Michigan Library
www.hathitrust.org/blogs/large-scale-search

Code4lib
February 7, 2012
HathiTrust

• HathiTrust is a shared digital repository
• 50+ member libraries
• Large Scale Search is one of many services built on top of the repository
• Currently about 10 million books
• 450 Terabytes
  – Preservation page images; jpeg 2000, tiff (438TB)
  – OCR and Metadata about (12TB)
**Large Scale Search Challenges**

- **Goal:** Design a system for full-text search that will scale to 10 million - 20 million volumes (at a reasonable cost.)

- **Challenges:**
  - Must scale to 20 million full-text volumes
  - Very long documents compared to most large-scale search applications
  - Multilingual collection (400+ languages)
  - OCR quality varies
Long Documents

- Average HathiTrust document is 700KB containing over 100,000 words.
  - Estimated size of 10 million Document collection is 7 TB.
- Average HathiTrust document is about 38 times larger than the average document size of 18KB used in Large Research test collections.
- 2 quadrillion words in corpus.

<table>
<thead>
<tr>
<th>Collection</th>
<th>Size</th>
<th>Documents</th>
<th>Average Doc size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HathiTrust</td>
<td>7 TB</td>
<td>10 million</td>
<td>700 KB</td>
</tr>
<tr>
<td>TREC GOV2</td>
<td>0.456 TB</td>
<td>25 million</td>
<td>18 KB</td>
</tr>
<tr>
<td>SPIRIT</td>
<td>1 TB</td>
<td>94 million</td>
<td>10 KB</td>
</tr>
<tr>
<td>NW1000G-04</td>
<td>1.3 TB*</td>
<td>100 million</td>
<td>16 KB</td>
</tr>
</tbody>
</table>
Multilingual

• 400+ languages, 50 languages with over 1,000 volumes.
• About 500,000 (5%) of volumes in multiple languages
• Currently all languages in one index
• Lowest common denominator tokenizing
• Some languages are challenging for Information Retrieval
  – CJK
  – Arabic
  – Finnish, Turkish, German
The top 10 languages make up ~86% of all content

* As of January 4, 2012
The next 40 languages make up ~13% of total (percentages shown are percentage of the 13%)

* As of January 4, 2012
“Dirty” OCR

The OCR varies in quality

Dirty OCR results in large terms index, and high memory consumption

Dirty OCR affects term statistics and makes some volumes difficult to find

Example: Hebrew characters not recognized
Plafonds - Decorationen.

Gewinde zur Verzierung der Decken von Zimmern und Sälen.

Druck von Johann M. Schreiber, Nürnberg.

F. Zungen, privilegiert und eingetragen im Kabinett der Tafel- und Graphik-Studien.

Plachtaufen aller Art,


Für die Theilnehmer an der Kunstwerkstatt.

Das Buch ist durch die königlich privilegierte Druckerei von Johann M. Schreiber, Nürnberg, gedruckt.

Plachtaufen aller Art,


Für die Theilnehmer an der Kunstwerkstatt.
Scalability meets Usability: Challenges

- Must have reasonable response time as collection scales to 20 million volumes
- Provide good results on the first result page (relevance ranking)
- Provide good results for non-English searching
- Help users manage large result sets or drill down within them
- All the usability issues for search interfaces apply regardless of scale
Response time and scalability

• Ideally response time should be under 1 second*
• Response time of over 10 seconds distracts users from their tasks*
• Early scalability tests with 1 million volumes showed the slowest 1% of queries took from 10 seconds to two minutes

*See: http://www.useit.com/papers/responsetime.html
Response Time Varies with Query

Response time 1 Million Volumes
2 Shards 2 Machines 16GB Mem each
(log scale)

Average: 673
Median: 91
90th: 328
99th: 7,504
Slowest 5% of queries
Performance problem: phrase queries too slow

• Slowest query: “The lives and literature of the beat generation” took 2 minutes
• Cause of slow queries is high disk I/O
• Phrase queries require reading the “positions index” from disk
• Common words such as “the” occur so frequently that reading their positions list requires many GB of disk I/O
Stop Words

• The word “the” occurs 40 billion times in the corpus or about 4 billion times per million documents.
• Removing “stop” words (“the”, “of” etc.) not desirable
• Couldn’t search for many phrases
  – “to be or not to be”
  – “the who”
  – “man in the moon” vs. “man on the moon”
Stop Words

• Stop words in one language are content words in another language
• German stopwords “war” and “die” are content words in English
• English stopwords “is” and “by” are content words (“ice” and “village”) in Swedish
CommonGrams

• Instead of stop words, create bi-grams for common words

• Example: Slowest query: “The lives and literature of the beat generation”
  • “the-lives” “lives-and”
  • “and-literature” “literature-of”
  • “of-the” “the-beat” “generation”
### Standard Index vs. Common Grams

#### (500,000 document index)

**• Standard Index**

<table>
<thead>
<tr>
<th>WORD</th>
<th>TOTAL OCCURRENCES IN CORPUS (MILLIONS)</th>
<th>NUMBER OF DOCS (THOUSANDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2,013</td>
<td>386</td>
</tr>
<tr>
<td>of</td>
<td>1,299</td>
<td>440</td>
</tr>
<tr>
<td>and</td>
<td>855</td>
<td>376</td>
</tr>
<tr>
<td>literature</td>
<td>4</td>
<td>210</td>
</tr>
<tr>
<td>lives</td>
<td>2</td>
<td>194</td>
</tr>
<tr>
<td>generation</td>
<td>2</td>
<td>199</td>
</tr>
<tr>
<td>beat</td>
<td>0.6</td>
<td>130</td>
</tr>
<tr>
<td>TOTAL</td>
<td>4,176</td>
<td></td>
</tr>
</tbody>
</table>

**• Common Grams**

<table>
<thead>
<tr>
<th>TERM</th>
<th>TOTAL OCCURRENCES IN CORPUS (MILLIONS)</th>
<th>NUMBER OF DOCS (THOUSANDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>of-the</td>
<td>446</td>
<td>396</td>
</tr>
<tr>
<td>generation</td>
<td>2.42</td>
<td>262</td>
</tr>
<tr>
<td>the-lives</td>
<td>0.36</td>
<td>128</td>
</tr>
<tr>
<td>literature-of</td>
<td>0.35</td>
<td>103</td>
</tr>
<tr>
<td>lives-and</td>
<td>0.25</td>
<td>115</td>
</tr>
<tr>
<td>and-literature</td>
<td>0.24</td>
<td>77</td>
</tr>
<tr>
<td>the-beat</td>
<td>0.06</td>
<td>26</td>
</tr>
<tr>
<td>TOTAL</td>
<td>450</td>
<td></td>
</tr>
</tbody>
</table>
## CommonGrams

### Comparison of Response time (ms)

<table>
<thead>
<tr>
<th></th>
<th>average</th>
<th>median</th>
<th>90th</th>
<th>99th</th>
<th>slowest query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Index</td>
<td>459</td>
<td>32</td>
<td>146</td>
<td>6,784</td>
<td>120,595</td>
</tr>
<tr>
<td>Common Grams</td>
<td>68</td>
<td>3</td>
<td>71</td>
<td>2,226</td>
<td>7,800</td>
</tr>
</tbody>
</table>


Good Response Time with Large Indexes: Tuning Caching and Memory

• Our 10 million document index is about 7 terabytes.
  – About 700 GB per million documents
• Large index means disk I/O is bottleneck
• Tradeoff JVM vs OS memory
  – Solr uses OS memory (disk I/O caching) for caching of postings
  – Tests showed memory available for disk I/O caching has most impact on response time (assuming adequate cache warming)
• Performance tuning
  – Reduce memory needed by Solr (16GB for 3 shards)
    • termInfosIndexDivisor and termIndexInterval
  – Leave a large amount of free OS memory for caching (about 50GB per server)
  – Run cache-warming queries every morning
Full-text search and “search within a book”: Tuning for response time

• Full-text search of 10 million books uses the whole book as a Solr document.
  – Size about 7 TB, so we need to optimize the index for minimal query latency.
  – We index off-line
  – We optimize off-line before putting the index into production for searching

  – Each page has a “book id” field so a search within the book includes a filter query: q1=bacon&fq=book_id=12345
  – It is optimized for minimal latency between the time the book is sent for indexing and the time the book is available for searching
  – We keep the index small by deleting it nightly. Average size is about 4GB
  – We index and serve queries in the same Solr instance
  – We configure the index to minimize merging
  – We store the pages so that we can display and highlight snippets
Good Results and Relevance

• System magic
  – query processing
    • Field weighting 1 (add boosts for matches in author/title/subject metadata)
    • Query normalization (lower casing, diacritic normalization)
    • Query expansion (synonyms, stemming, added phrase queries)
    • Field weighting 2, weighted query expansion/normalization
    • query intent guessing

• Help users refine their search
  – user query reformulation (UX issues)
  – Advanced search including MARC metadata
  – faceting
Relevance: Recall and Precision

• Recall = how many of the relevant results in the collection are returned
• Precision = the ratio of relevant results to the total number of results.
• Normally for relevance ranking these are measured at some number of results for example p@10 is the precision at 10 results
Precision, Recall and Performance

• Generally there is an inverse relationship between precision and recall
• Different types of queries favor recall or precision
• Compare these queries:
  – dog OR food  (4,431,164) hits
  – dog AND food (2,006,995) hits
  – “dog food”  (52,784 ) hits
For large-scale search, favor precision over recall

• With the full text of over 10 million very long documents large result sets can be a problem
  – We changed the default Solr Boolean operator from “OR” to “AND”
  – We allow phrase queries so users can enter very precise queries
  – We don’t use stop words
  – We don’t do stemming or use synonyms*
Weighted query expansion to boost recall without hurting precision. (Scalable?)

• User query: [ dog food]
  – Boost query: ("dog food")^100 OR ("dog food"~200) ^50 OR(dog AND food)^10 OR ( dog OR food) ^1
  – Dismax query: _query_:"{!edismax qf='ocr^500+marcfields^10+ pf =ocr^5000+marcfields^40+ pf2=ocr^2500+marcfields^20}
Usability v. Scalability:
Relevance v.s. performance

• Stopwords
  – (solved with CommonGrams and cache warming)
• Adding weighted phrases (dismax pf,pf2,pf3 )
  – (High disk I/O for OCR phrase searches)
• Proximity (words on same page/within 200 words)
  – (High disk I/O for OCR phrase searches)
Usability v. Scalability:
Relevance v.s. performance complicated by multiple languages

• truncation operator
  – 3+ billion unique words make truncation computationally intractable

• Fuzzy search (helps with dirty OCR)
  – 3+ billion unique words = intractable

• Character ngrams (language independent stemming/helps with dirty OCR)
  – Index size increase of 5-20x not scalable
Stemming and synonyms

• Stemming conflates different forms of the same word to increase recall.
  – cat, cats
  – organized, organizer, organizing, organization

• However over-stemming leads to loss of precision by conflating words with different meanings
  – Organize, Organic, Organ => Organ
Stemming and synonyms

- **Solution:** create separate field and combine unstemmed and stemmed field with lower weight for stemmed field
  - `<copyField source="unstemmed" dest="stemmed">`
  - Boost: \( q = \text{unstemmed}^{1000} + \text{stemmed}^{50} \)

- **Problems**
  - Stemming is language-specific
  - Copying the OCR field would double the index size from 6TB to 12TB

- **Current hack**
  - Use copyfield strategy with English stemmer for MARC fields only.

- **Synonyms** have similar issues
Normalization, diacritics and thé

• We normalize both text and queries. Words are lower-cased and diacritics are removed

• Removing diacritics allows searching for non-English words without special keyboard mappings

• The French word for tea is thé
  – This gets normalized to “the” resulting in millions of hits.
  – The term statistics (idf) and relevance ranking are also affected as thé would normally occur much less frequently than “the”
  – Copyfield solution would double size of index
Normalization, diacritics and thé

- Copyfield solution would double size of index
  - `<copyField source="diacritics" dest="noDiacritics">`
  - Boost: \( q=\text{diacritics}^{1000}+\text{noDiacritics}^{50} \)
- Possible alternative: create filter that outputs token both with and without diacritics as if they were synonyms (only when there is a diacritic in input)
  - Search for \( \text{thé} \) would only match docs containing \( \text{thé} \)
  - Search for \( \text{the} \) would match both \( \text{the} \) and \( \text{thé} \)
  - Index size increase proportional to number of tokens with diacritics
Tokenization and other language-specific issues

• With over 400 languages, using language detection and putting each language in a separate field is not feasible
  – What to do with 500,000 volumes in multiple languages?
• We put all languages in one field
• Need to use analyzer that works ok for all languages: ICUFilters
Tokenization and other language-specific issues

• Some languages such as Chinese, Japanese, Thai, and Vietnamese don’t use spaces to separate words*
  – Language-specific solutions (SmartChineseAnalyzer)
  – Script based solutions (CJK tokenizer, ICUTokenizer)
• For Chinese there are two character sets: Traditional and Simplified. Users expect to enter terms in one character set and get results in both.
• Japanese uses 4 different scripts, the same word may be written in numerous ways

* http://www.hathitrust.org/blogs/large-scale-search/multilingual-issues-part-1-word-segmentation
Tokenization and other language-specific issues

• Arabic has very large number of word forms and needs stemming for effective retrieval.
  – Solr has a very good light stemmer for Arabic
  – Problem is that other languages such as Urdu use the Arabic script
• Other languages (German for example) need decompounding
• We can implement script-specific processing but not yet language specific processing
New Features

• Help users with large result sets
  – Facets based on MARC metadata
  – Advanced search based on MARC metadata

• Improve relevance ranking
  – Use added weight for matches in MARC metadata
  – “Search within a book” back-end moved to Solr: Relevance ranked results when searching within a book
Facets based on MARC Metadata

1.6 million hits for Bacon!

- Bacon the food?
- Francis Bacon?
- Roger Bacon?
Facets: Scalability and Usability

• Facets work best with small value sets of 10-30 values. LCSH provides millions of values!
• Solr will show the top N facet values by facet count
• The facet counts are based on all the results
• In HT full-text search large result sets (over 10,000 records) are common, the facet counts are based on the entire result set, not the most relevant of those 10,000+ records
Irrelevant Facets

Jaguar car, Jaguar animal, Jaguar OS?
Relevant facets

http://lucene.472066.n3.nabble.com/Getting-facet-counts-for-10-000-most-relevant-hits-td3363459.html
Advanced search based on MARC metadata and OCR

Advanced Full-text Search:
- all of these words: bacon
- AND: any of these words: diets cooking food agriculture swine pork

Limit To:
- Full view only: 
- Year of publication: During or after

Language:
- All
- Abkhazian
- Achi
- Acoli
- Adangme
- Adyghe
- Afrikaans
- Afro-Asiatic
- Akan
- Akkadian

Original Format:
- All
- Archive
- Audio
- Audio (music)
- Audio (spoken word)
- Audio CD
- Audio LP
- Biography
- Book
- Computer File

Find
Clear/reset
Advanced search based on MARC metadata and OCR
Relevance ranking with MARC Metadata

• Example (simplified)
  
  q= _query_:{!edismax
  qf='ocr^5000
  +AnyMARCField^2
  +TitleField^50
  +AuthorField^80
  +SubjectField^50
  +mm='100%'
  tie='0.9' } bacon"

• mm = minimum must match
• tie = weight of all fields other than the field with maximum score

* see: http://wiki.apache.org/solr/DisMaxQParserPlugin
Relevance ranking with MARC Metadata

- Example (actually used)
  - $q =$ _query_: "{"edismax
    qf='ocr^50000+allfieldsProper^2+allfields^1+titleProper^50+title_topProper^30+title_restProper^15+title^10+title_top^5+title_rest^2+series^5+series2^5+
    author^80+author2^50+issn^1+isbn^1+oclc^1+sdrnum^1+ctrlnum^1+id^1+reportnum^1+topicProper^2+topic^1+hlb3^1+fullgeographic^1+fullgenre^1+era^1+
    
    pf='title_ab^10000+titleProper^1500+title_topProper^1000+title_restProper^800+series^100+series2^100+author^1600+author2^800+topicProper^200+fullgenre^200+hlb3^200+allfieldsProper^100+' mm='100%25'
    
    tie='0.9' } bacon"

- qf = query
- pf = query words as a phrase query
- pf2 and pf3 = disjunction of 2 and 3 word combinations as phrases
- mm = minimum must match
- tie = weight of all fields other than the field with maximum score
Tuning Relevance Ranking

• How should we adjust the relative weight between OCR (full-text) and MARC fields?
• What are the Solr settings affecting ranking?
• How do you tell when one group of settings provides more relevant results than another
  – Need to test against some set of queries to insure that improving results for one query doesn’t make results for other queries worse
Solr relevance knobs

- Query expansion
- Field weighting
- Boost queries or dismax queries
- Expert
  - length normalization
    - SweetSpotSimilarity
  - Solr relevance formula tf * idf
    - write custom Similarity class or use Solr 4.0/Trunk
Testing Relevance

• Use test collection and relevance judgements
  – TREC/XML book track*
  – Are queries similar to your user’s queries?
  – Is collection similar to your collection?
  – What is the appropriate metric (P@10) other real-world issues?

• Relevance is complicated and different users will consider different sets of documents relevant
  – Ambiguous queries

*https://inex.mmci.uni-saarland.de/tracks/books/*
Testing Relevance

• Use clickthrough logs and A/B testing*
  – clickthrough not always measure of relevance
  – A/B testing controversial in library settings
  – Use interleaving to overcome order effects

• Usability testing?

• Beta testers?

“Dirty” OCR: Causes

- noise, variable ink, bleedthrough, markings by users
- tight bindings affect scanning, words near binding or edge can be distorted
- background color, fading, water or mold damage/stains
- bad scanning, blurred, operator scanning
- 400+ languages (x volumes in more than one language)

* Feng and Manmatha (2010?)
“Dirty” OCR: Effects on OCR

- OCR engine produces no OCR for page or volume
- OCR engine misrecognizes language/script and produces garbage
- OCR engine interprets images as text and produces garbage
“Dirty” OCR: Effects on Search

- Large number of garbage terms result in over 3 billion unique terms per index.
  - blew up lucene *
  - increased memory requirement to 32GB until we changed termIndexInterval
  - MultiTerm queries (wildcard, prefix/truncation) performance unusable
  - Problem for spelling suggestion
  - Documents with lots of garbage from images have incorrect document length normalization

* http://www.hathitrust.org/blogs/large-scale-search/too-many-words-again
“Dirty” OCR: Effects on Search

- Missing OCR text not retrievable, some volumes not searchable
- Incorrectly recognized text not retrievable
- Term statistics and relevance ranking skewed
- If students underline important parts and underlines cause misrecognition, important parts less retrievable
“Dirty” OCR: Solutions?

- IR techniques for retrieving dirty OCR such as ngrams, fuzzy matching, error correction and query garbling have disappointing performance or are not scalable (TREC 4&5 and Savoy 20xx)
- Removing hapax (terms that occur only once) will likely remove huge amount of real words including proper names
- Correction algorithms are designed for single language
- Heuristics will remove some fraction of bad words but must work with 400 languages
- Google improves OCR engine and re-OCR’s in a 2 year cycle
Collection Builder

A screenshot of the Hathi Trust Digital Library showing a search for "library of universal knowledge". The search results page displays a list of items, including:

- General encyclopedias in print, 1963
- History by the topical method
- Erasmus & Henry VIII
- General encyclopedias in print, 1964
- Anglo-American general encyclopedias: a historical bibliography, 1703-1967
# Collection Builder

### Login

To create and save permanent collections or see your private collections.

### Collections Filter

- **All**
- **Recently Updated**
- **Featured**
- **My Collections**

### Search Bar

Find a collection

### Collections Display

<table>
<thead>
<tr>
<th>Collection Title</th>
<th>Items</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generations</td>
<td>2</td>
<td>10/11/10</td>
</tr>
<tr>
<td>1850s-1870s Japan</td>
<td>20</td>
<td>10/18/10</td>
</tr>
<tr>
<td>19-20th C. Psychology Texts-Gen</td>
<td>127</td>
<td>09/05/11</td>
</tr>
<tr>
<td>19th c cookbooks</td>
<td>184</td>
<td>07/28/11</td>
</tr>
<tr>
<td>19th c periodicals</td>
<td>23</td>
<td>03/16/10</td>
</tr>
<tr>
<td>19th c Ottoman binding</td>
<td>1</td>
<td>05/27/10</td>
</tr>
</tbody>
</table>

### Featured Collection

**Adventure Novels: G.A. Henty**

Novels from the Henty Collection and books about George A. Henty.

![Henty Series Cover](image)
Collection Builder
[The relationships of the eastern North American crayfishes, with a revision of the genus Faxonius]
by Creaser, Edwin P.
Published 1933
- Catalog Record
- Full view

"Eastern magnificence & European ingenuity" : clocks of late imperial China / Catherine Pagani.
by Pagani, Catherine.
Published 2001
- Catalog Record
- Full view

The "Goldhagen effect" : history, memory, Nazism--facing the German past / edited by Geoff Eley.
Published 2000
- Catalog Record
- Limited (search-only)

1886. Crashaw; a study in baroque sensibility.
Published 1957
- Catalog Record
- Full view

9226 Kercheval; the storefront that did not burn.
by Millo, Nancy.
Published 1970
- Catalog Record
- Full view
Collection Builder

- http://babel.hathitrust.org/cgi/mb
- Need to update one field, i.e. add user’s collection number to the collection field
- Solr needs to re-index whole record including 800MB+ of OCR
- Solution #1 Keep collection-unique id table in database send query with ids in collection to Solr
  - q1=ocr:bacon AND id:(2 OR 5 ...or 1023)
  - not scalable
- This is one of many use cases for the ability to merge a list of ids from the result of a database or other external system search into a Solr query.
  - See: http://lucene.472066.n3.nabble.com/filter-query-from-external-list-of-Solr-unique-IDs-tc1709060.html#a1709716
  - https://issues.apache.org/jira/browse/SOLR-139
  - https://issues.apache.org/jira/browse/LUCENE-1879
Collection Builder

• Solution 2
• Bite the bullet and update the entire record every time someone add a record to their collection. i.e. update the coll_ids field
• To minimize the latency between updating a record and having it be available for searching we needed a separate small Solr index configured/optimized for minimal update latency
Collection Builder

• New requirement: medium to large collections i.e. 10,000 – 100,000 volumes

• Solution 3/Current hack
  • Eliminate small Solr and use same Solr as full-text search
  • If collection is under 1000 (1024) items in size store list of unique ids in database and send id list to Solr. Collection update just changes a row in the database and requires no Solr re-indexing so immediate collection update
  • If collection is over 1000 items, add the items to the nightly indexing queue with an update to the coll_ids field. Message to user saying they will have to wait overnight to be able to search new items
Thank You!

Tom Burton-West
tburtonw@umich.edu

www.hathitrust.org/blogs/large-scale-search