

Is searching full text more effective than searching abstracts?

BMC Bioinformatics 2009

By Jimmy Lin

Presented by Nils Schimmelfmann

Comparing text retrieval techniques through different models, metrics, and configurations

Introduction to Text Mining in the Life Sciences

- Text mining is necessary
- The rate of available articles is growing exponentially
- Most current systems search abstracts only
- Effective full-text search methods will become more integral as time goes on

Background

- Open Access is causing more articles to be easily and freely viewable
- More data makes it more difficult to find relevant articles from searches
- More advanced text retrieval algorithms are needed

The Question

- Is full-text mining necessary when there are abstracts?
- Sometimes abstracts are good enough!

Related Work

- Some search techniques use title, abstract, and metadata information
 - Medical Text Indexer (MTI)
 - Such techniques cannot be easily converted to full texts
 - Attempted with full texts giving lackluster results
- Text mining covers more than just the basic 'search engine' functionality
 - For example...

Related Work

- Using surface patterns to extract gene and protein synonyms
 - Higher precision using full texts
- Exploring gene-disease associations
 - Small gain in effectiveness with full texts
- Clinical decision support
 - Abstracts were enough
- Summarization of figures and tables
 - Abstracts can summarize them effectively (80%+)

Comparing Abstracts to Full Text:

Advantages and Disadvantages

Abstracts

- Density of useful information is at its highest
- Less text, allowing for less computational power

Full Text

- Higher coverage of information
- More text, requires clusters of computers
- Noise from more text (conjectures, future work, citations)
- Variety of formats: PDF, HTML, XML, etc
- Required for some uses (image searching)

Method

- Compare two different models (programs/algorithms)
- Use three different evaluation metrics
- Compare the resulting experiments against a base abstract retrieval method

Method

- Test Collection:
 - 162,259 full-text articles from Highwire press
 - 36 topics annotated by humans
- Retrieval Models:
 - Okapi *bm25* ranking algorithm using the MapReduce programming model on Ivory/Hadoop
 - Open-source search engine Lucene's ranking algorithm *tf.idf*

Method

- Indexes:
 - Abstract index: built from only abstracts
 - Article index: made up of all the articles
 - Span index: consisting of the paragraphs in the articles
- Evaluation Metric:
 - Mean Average Precision (MAP)
 - Precision at 20 (P20): Top 20 Articles
 - Interpolated precision at recall of 50% (IP@R50)

“Legal Spans” Retrieval Model

- Articles were divided into 12.6 million “legal spans” (paragraphs)
- Rankings were post-processed for each retrieval model (*bm25* and *tf.idf* for Lucene)
- Two methods were used:
 - Maximum of supporting spans (max): favors highest
 - Sum of supporting spans (sum): favors many potentially relevant spans

Results

Table 1: Effectiveness of *bm25* and the Lucene ranking algorithm on abstracts, full-text articles, and spans from full text.

	MAP ← Mean Average Precision	
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Abstract	0.163	0.129
Article	0.146 (−11%) [◦]	0.235 (+82%)**
Span (max)	0.240 (+47%)**	0.206 (+60%)**
Span (sum)	0.192 (+18%)*	0.198 (+54%)**
	P20 ← Top 20	
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Abstract	0.322	0.293
Article	0.158 (−51%)**	0.353 (+20%)*
Span (max)	0.357 (+11%) [◦]	0.332 (+13%) [◦]
Span (sum)	0.314 (−3%) [◦]	0.317 (+8%)*
	IP@R50 ← Interpolated precision at recall of 50%	
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Abstract	0.110	0.090
Article	0.163 (+48%) [◦]	0.222 (+146%)**
Span (max)	0.212 (+93%)**	0.189 (+109%)**
Span (sum)	0.149 (+36%)*	0.159 (+77%)**

For all metrics, relative improvements over baseline are shown; **= statistically significant ($p < 0.01$); *= statistically significant ($p < 0.05$); [◦]= not significant.

Results

Table 3: Effectiveness of *bm25* and the Lucene ranking algorithm combining evidence from spans with evidence from abstracts and articles.

MAP		
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Span (max)	0.240	0.206
Span (max) + Abstract	0.257 (+7%) [°]	0.216 (+5%) [°]
Span (max) + Article	0.257 (+7%) [°]	0.262 (+27%)**
P20		
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Span (max)	0.357	0.332
Span (max) + Abstract	0.382 (+7%) [°]	0.349 (+5%) [°]
Span (max) + Article	0.343 (-4%) [°]	0.404 (+22%)**
IP@R50		
	Ivory (<i>bm25</i>)	Ivory (<i>Lucene</i>)
Span (max)	0.212	0.189
Span (max) + Abstract	0.215 (+1%) [°]	0.190 (+1%) [°]
Span (max) + Article	0.257 (+21%) [°]	0.244 (+29%)**

← Best method is span (max)

For all metrics, relative improvements over baseline are shown; **= statistically significant ($p < 0.01$); *= statistically significant ($p < 0.05$); [°]= not significant.

Conclusion/Discussion

- The Lucine model along with the "legal spans" method and the "max" evaluation metric provides the best solo result
- Averaging span and article evidence yields the best results
- "max" is doing better than "sum" because of length normalization which would bias the "sum" methodology

Ranking Algorithms

- Standard "bag of words" model from which all ranking algorithms have a base

$$\sum_{t \in q} w_{t,d} \cdot w_{t,q}$$

$w_{t,d}$ = weighted terms in the document
 $w_{t,q}$ = weighted terms in the query

- Okapi *bm25*

$$\sum_{t \in q} \log \left(\frac{N - n + 0.5}{n + 0.5} \right) \frac{(k_1 + 1)tf (k_3 + 1)qtf}{K + tf \quad k_3 + qtf}$$

$$K = k_1 \left((1 - b) - b \frac{dl}{avdl} \right)$$

N = number of documents

n = number of documents with the term

tf = term frequency

qtf = query term frequency

K = normalization factor

$avdl$ = average length of all documents

dl = document length

k_1, k_3, b = tunable parameters

Ranking Algorithms

- Lucene (modified from *tf.idf*)

$$c \sum_{t \in q} \sqrt{tf} \left(1 + \log \frac{N}{n+1} \right)^2 \left(\frac{1}{\sqrt{dl}} \right)$$

N = number of documents

dl = document length

n = number of documents with the terms

tf = term frequency

c = fraction of query terms in the document

- c rewards documents with many terms

Other Difficulties: Computational Power

- Ivory
 - An Open Source implementation of a distributed text retrieval system
 - Built upon Hadoop
- Hadoop is a open source Java implementation of MapReduce
 - Created by Google

