

From Passages Into Elements in XML Retrieval

Kelly Itakura
Charles L. A. Clarke

What Users Want

- Human assessors perform relevance judgments based on passages
- What we really need to return is passage or elements based on passages.



Reality



- Trotman and Geva (2006):
 - In INEX adhoc track, returned results are XML elements.
 - We confuse XML retrieval with XML-element retrieval.

Turning Reality into Ideal

- To transition into passage-based XML retrieval, we need ways to convert

passages \longleftrightarrow elements

• TG+



• TG-



Our Implementation of TG+

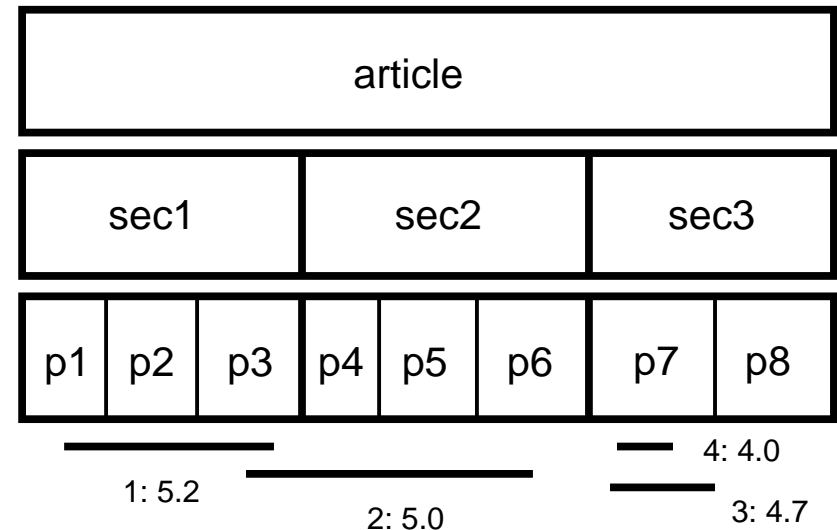
- Compute the scores of all possible passages.

- Score:

$$score(P) = \sum_{t \in Q} W_t \frac{f_{P,t}(k_1 + 1)}{f_{P,t} + k_1(1 - b + b \frac{|P|}{avgdl})}$$

$$W_t = \log \frac{total\#documents}{\#documents_containing_t}$$

- Assign the scores of elements as follows.



Other Algorithms for Comparison

- **Element Retrieval**

- score each elements, get rid of nesting, and get the top 1500 results.

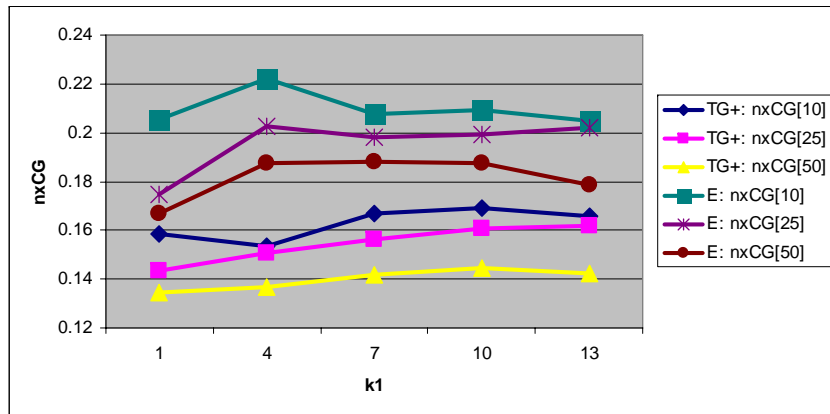
- **Passage Retrieval**

- get the positions of all query term occurrences, score all possible passages defined by these positions, get rid of nesting, return the top 1500 results.

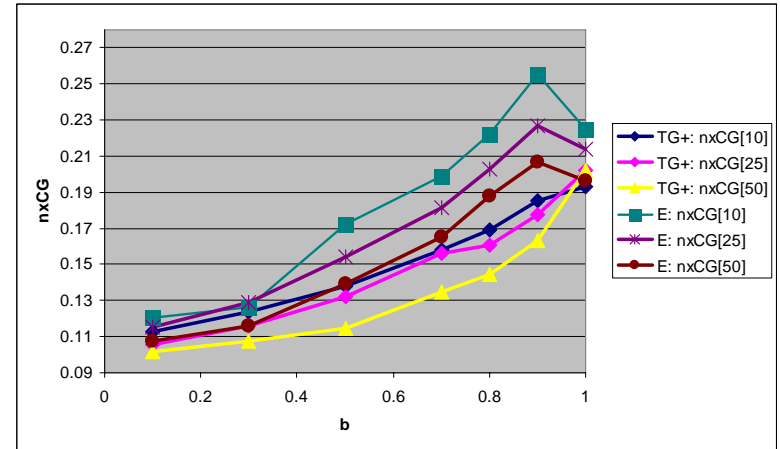
Experimental Setup

- **Corpus:** INEX2005 IEEE collection
 - 16,819 files from IEEE journals from 95-04.
- **Queries:** 39 topics from INEX2005 adhoc focused task.
- **Evaluation:** nxCG metric in generalized quantization.
- **Others:** Wumpus search engine to retrieve positions of query terms.

Experimental Results #1



Experimental Results #1

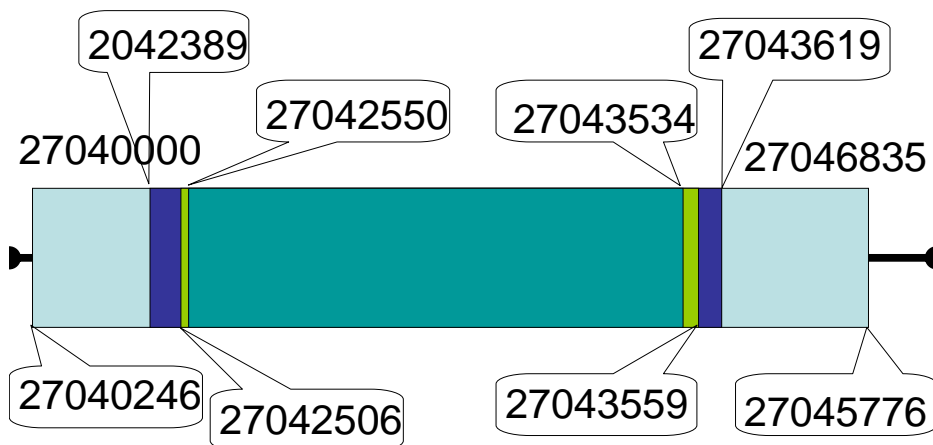


39.44 /article/bdy

42.53 Best in /article/bdy

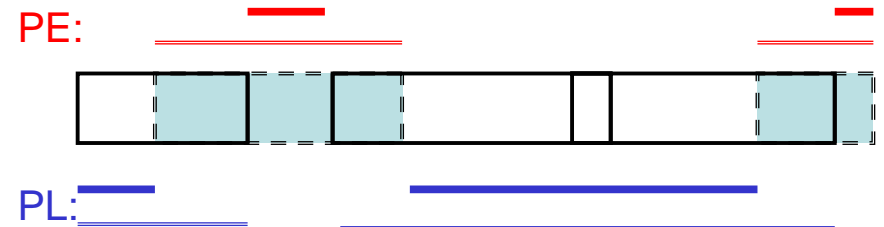
40.90 /article/bdy/sec[4]

41.02 Best In /article/bdy/sec[4]

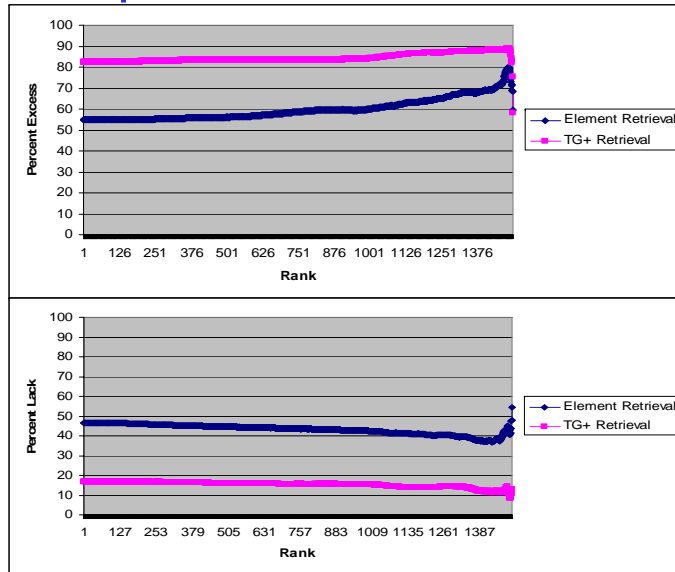


Experimental Results #2

- Measures: **Percent Excess (PE)** and **Percent Lack (PL)** of results returned by Passage Retrieval.



Experimental Results #2



Conclusions



- TG+ performs worse than element retrieval because it returns a lot of extra terms.
- TG- may miss a lot of terms but because Focused task prefers specificity over exhaustivity, the performance will be better.
- Ultimately, **the best elements to return are those that best fit the best passages,**
- **Or, return multiple elements that cover the passages just enough.**

Happily



Ever After