# Retrieving Records from a Gigabyte of Text on a Minicomputer Using Statistical Ranking

Harman & Candela

Berk KARAOĞLU
Can Çağdaş Cengiz

# introduction

previous studies, problems, and aim of this research

# introduction

statistical based ranked retrieval of records using keywords

Not a new concept; even in 1990s.

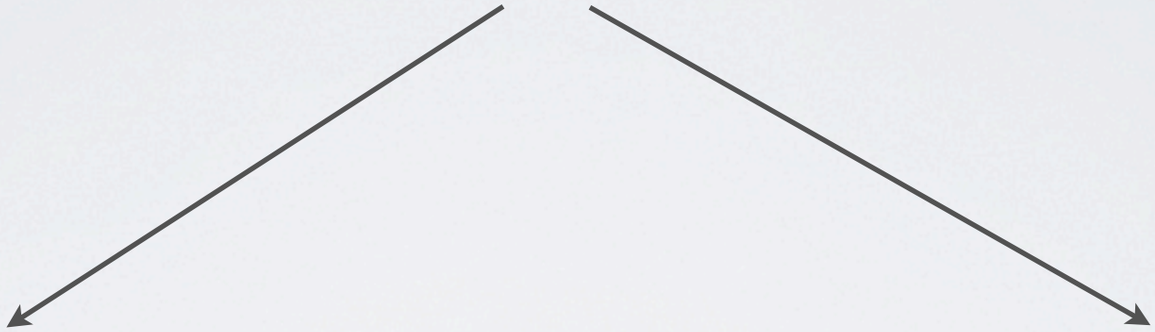
Was a research success.

## But;

not efficient enough for large systems.
no proof that statistical ranking can be done in real-time on large databases

# introduction

areas of this research



producing very efficient indexing and searching algorithms

preliminary design of interfaces to allow user testing.

# introduction

past work in statistical ranking

automatic indexing

automatically retrieving the matched records, and ranking according to their relevance.

# introduction

current approach in statistical ranking (actually, 1990s)

uses same similarity measures and term weighting functions as past experiments.

approach

## assign a ranking weight.

(frequency of query terms and the distribution within the database)

approach

## tf * idf

(term frequency * inverse document frequency)

# Agenda

## Indexing

methods used to create the inverted files

## Searching

search techniques and enhancing response times

## Testing

response time against large databases

# indexing

## Traditional automatic indexing

1- Parse the input text
2- Invert the list
3- (Optional) Post-processing

problem — minicomputers cannot handle sorting large databases

problem — the storage amount is relatively small

solution — skip the sorting step. produce the basic inverted file. add the term weights and reorganize.

# searching

## Traditional retrieval process

1- Retrieve all records containing the search terms
2- Compute total weight for each of those retrieved records
3- Sort the records

problem — process is dependent on the number of records retrieved

solution — create a hash table to accumulate the total record weights

# searching

search engine

Reserve a block of storage

Do a binary search

Get the address of the postings list

Add the term weight for each record id to the storage (do for each term in the query)

Sort each accumulator having non-zero weight

# searching

ways of further improving the search times

approach

## bitmapping

(bitmap the posting lists having more than 44 records.)

approach

## pruning

(eliminate the documents having lower relevance threshold)

# testing

did the search engine perform fast enough to satisfy online use?

did the ranking produce satisfactory results for the users?

what kinds of problems need to be addressed to produce a truly usable retrieval system?

# testing

| Size of the database | 268 MB | 806 MB |
|---|---|---|
| Avg. response time per query | 2.6 | 4.1 |
| Avg. response time (bitmapping) | 1.8 | - |
| Avg. response time (pruning) | 1.1 | 1.6 |

# conclusion

Previous indexing methods were implemented.

Search engine was built.

Bitmapping & Pruning

Create a user interface.

Testing

It is proved that this approach can be used in minicomputers with large databases.