COMP90042
Web search and text analysis

Workshop Week 4

xudong.han@unimelb.edu.au
https://github.com/HanXudong/COMP90042_Workshops

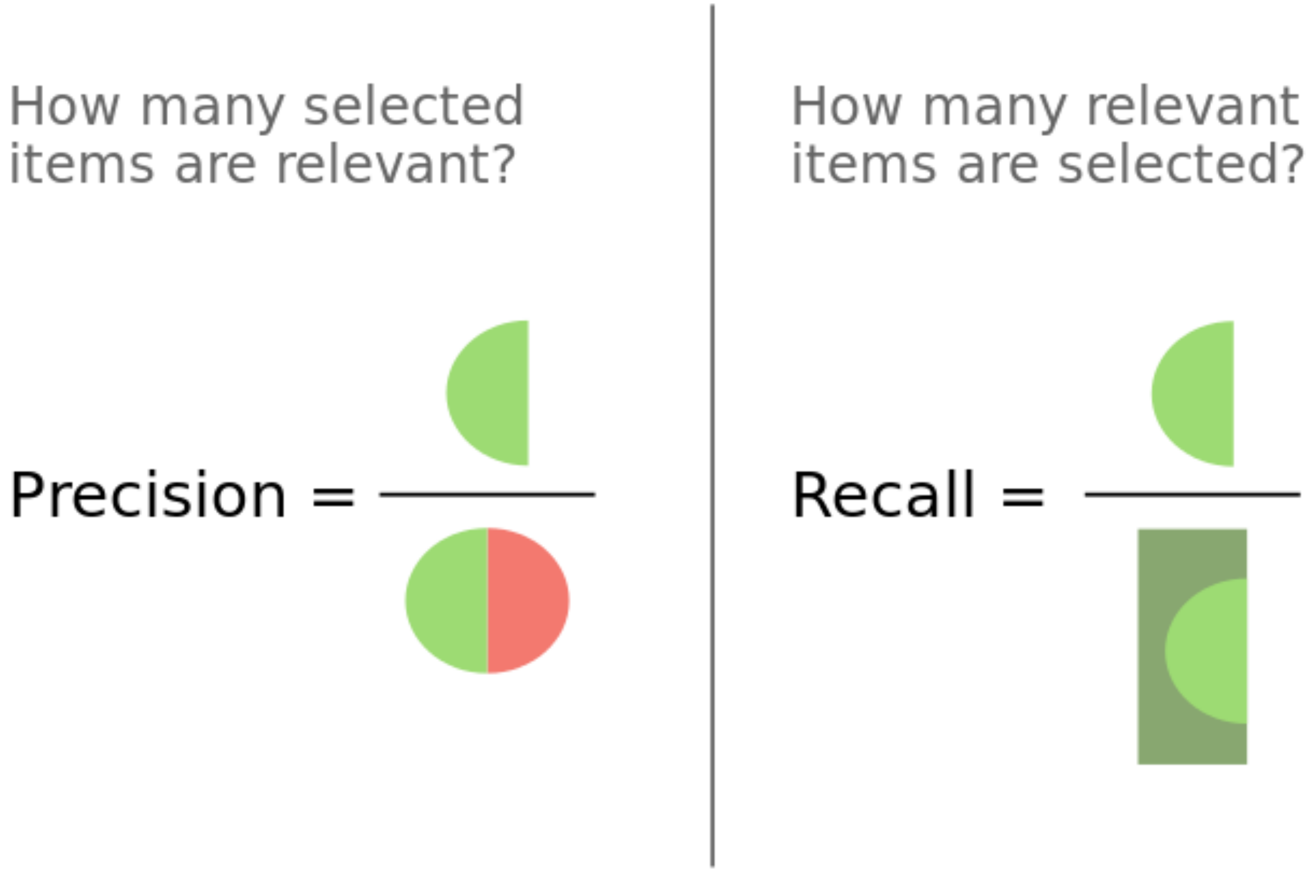
Review

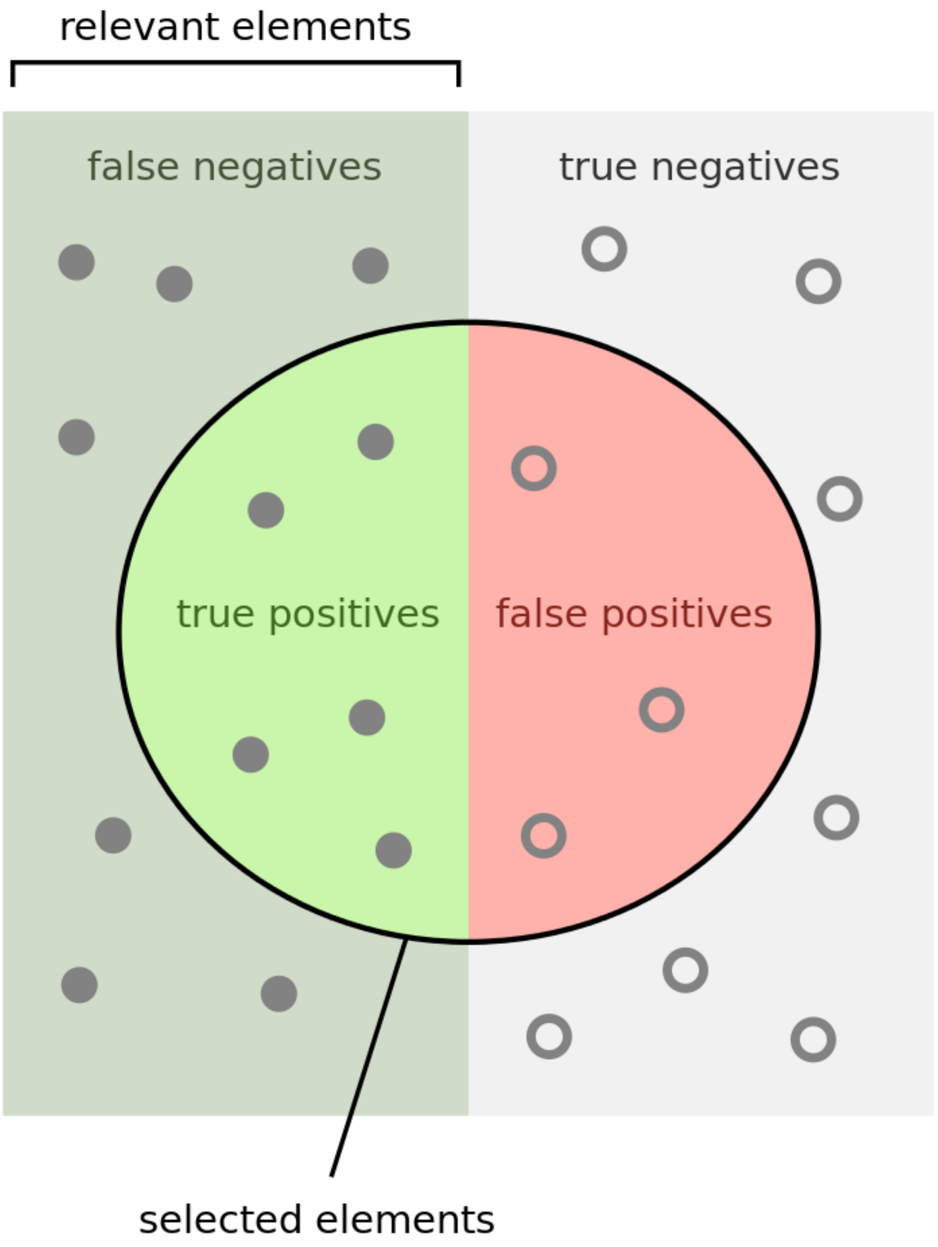
- Variable Byte Compression
- Recall and Precision

Query Expansion

Q3

Query expansion increases query recall

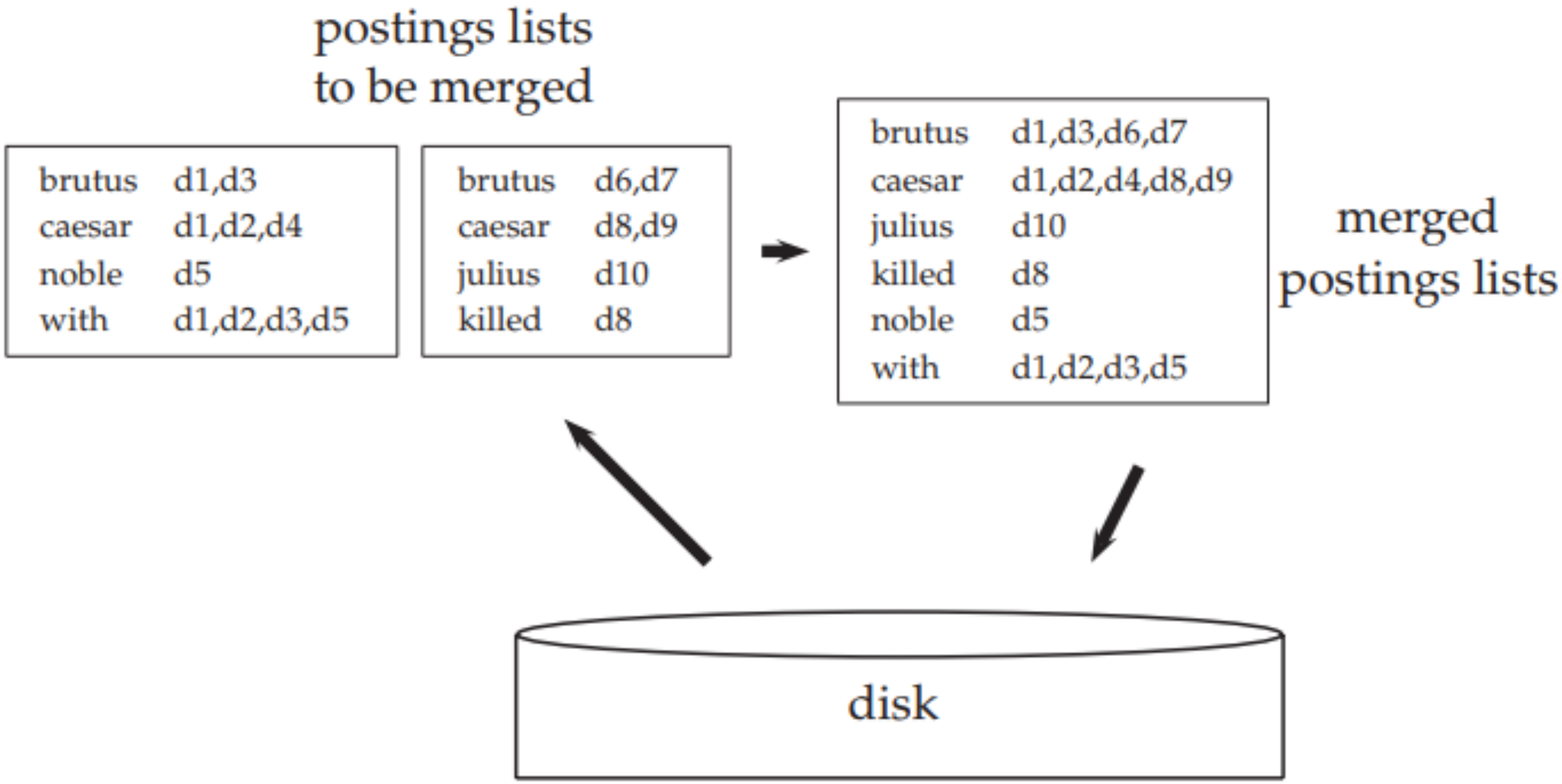

How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =



This workshop

- Inverted index construction
- Logarithmic index layout
- Evaluation metrics
- Supervised machine learning for IR

Q1: Discuss the process of static inverted index construction and how it can be used to perform in incremental index construction.



postings lists to be merged

brutus	d1,d3
caesar	d1,d2,d4
noble	d5
with	d1,d2,d3,d5

brutus	d6,d7
caesar	d8,d9
julius	d10
killed	d8

brutus	d1,d3,d6,d7
caesar	d1,d2,d4,d8,d9
julius	d10
killed	d8
noble	d5
with	d1,d2,d3,d5

merged postings lists

▶ **Figure 4.3** Merging in blocked sort-based indexing. Two blocks ("postings lists to be merged") are loaded from disk into memory, merged in memory ("merged postings lists") and written back to disk. We show terms instead of termIDs for better readability.

Static

Merge batches:

1. Open all n files containing batches on disk
2. Read one term(or a few) from each file
3. Perform n-way merge, merging terms with same ids
4. Merge equivalent to appending bytes as document ids are increasing
5. Read the next terms

Dynamic Inverted Index Construction

- a large main index (M postings)
- a small auxiliary index (n postings)
- Searches are run across both indexes and results merged
- A merge requires merging and writing M + n postings to disk (I/O)

Dynamic Inverted Index Construction

Problem:

After one year our index has N = 10 billion postings

At each time we have an inverted index in memory which is merged it has more than n = 1 million postings

- $totalcost = n + 2n + 3n + 4n + ... + (N/n)n = O(N^2/n)$

Q2:
Why is a logarithmic index layout useful?
What are the disadvantages of such an index structure?

- what is a logarithmic index layout?
  Use a logarithmic number(log*N*) of indexes. At each level *i*, store index of size 2^i× *n*

  http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html

- Query all logN indexes at the same time and merge results

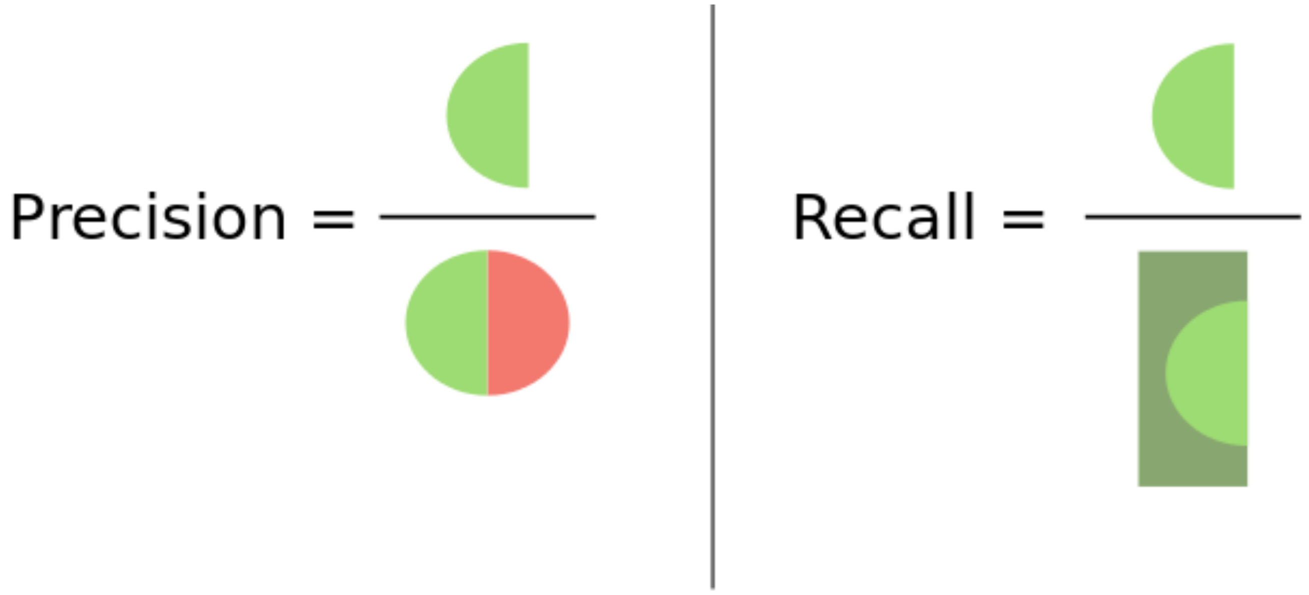
Q2:
Why is a logarithmic index layout useful?
What are the disadvantages of such an index structure?

- As the sizes of at the different level increases merging becomes cheaper as less IOs are required in total.
- At query time a logarithmic number of indexes have to be queried.

Q3:

Based on the following top-6 retrieval results from a collection of 100 documents, and the accompanying binary relevance judgements

doc	score	relevance
a	0.4	0
b	1.2	0
c	2.2	1
d	0.5	1
e	0.1	1
f	0.8	0



Q3:
Based on the following top-6 retrieval results from a collection of 100 documents, and the accompanying binary relevance judgements

doc	score	relevance
a	0.4	0
b	1.2	0
c	2.2	1
d	0.5	1
e	0.1	1
f	0.8	0

precision@3

$$Precision@k = \frac{\sum_{i=1}^{k} relevance_i}{k}$$

1. Rank docs
2. Take the first 3 docs
3. Compute

Q3:

Based on the following top-6 retrieval results from a collection of 100 documents, and the accompanying binary relevance judgements

$$\langle c,b,f,d,a,e \rangle = \langle 1,0,0,1,0,1 \rangle$$

Average precision

$$AP = \frac{\sum_{k=1}^{n} precision@k \times relevance_k}{\sum_{k=1}^{n} relevanc_k}$$

Q3:
Based on the following top-6 retrieval results from a collection of 100 documents, and the accompanying binary relevance judgements

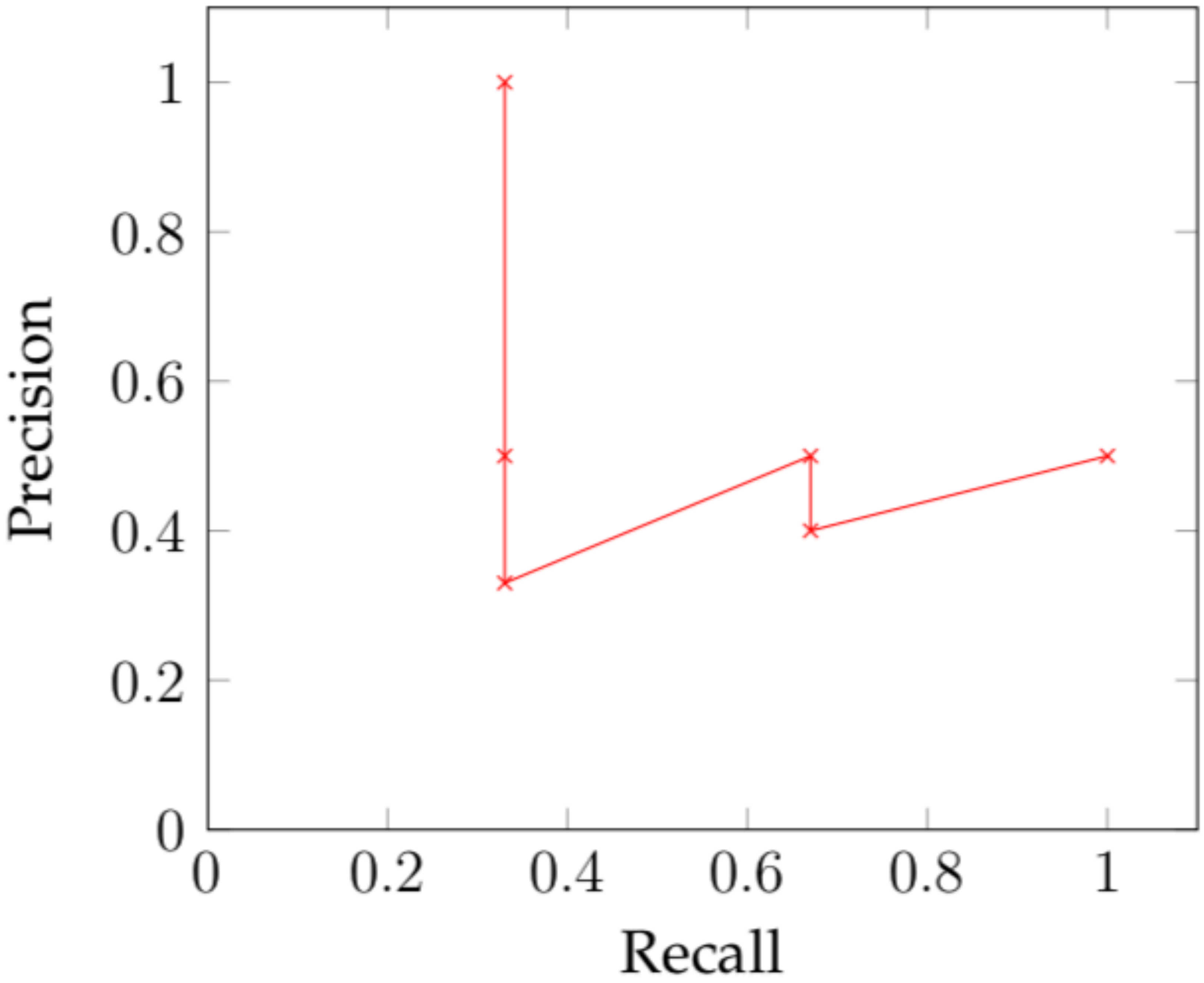
$$\langle \mathrm{c,b,f,d,a,e} \rangle = \langle 1,0,0,1,0,1 \rangle$$

$$\text{rank-biased precision (RBP), with p} = 0.5$$

$$RBP = (1 - p) \times \sum_{i=1}^{n} r_i \times p^{i-1}$$

Q3-d:

plot the precision-recall graph, where you plot (precision, recall) point for the top k documents, k = 1,2,...6.



Q3-e:
what are the strengths and weaknesses of the methods above for evaluating IR systems?

$$Precision@k = \frac{\sum_{i=1}^{k} relevance_i}{k}$$

Precision@k

- Easy to evaluate and understand
- But no differentiation by rank for ranked document 1, 2, …, k
- But no adjustment for the size of the relevant documents

$$AP = \frac{\sum_{k=1}^{n} precision@k \times relevance_k}{\sum_{k=1}^{n} relevanc_k}$$

Average precision

- Differentiation by rank
- Adjustment for the size of the relevant documents
- But need to know the size of the relevant documents

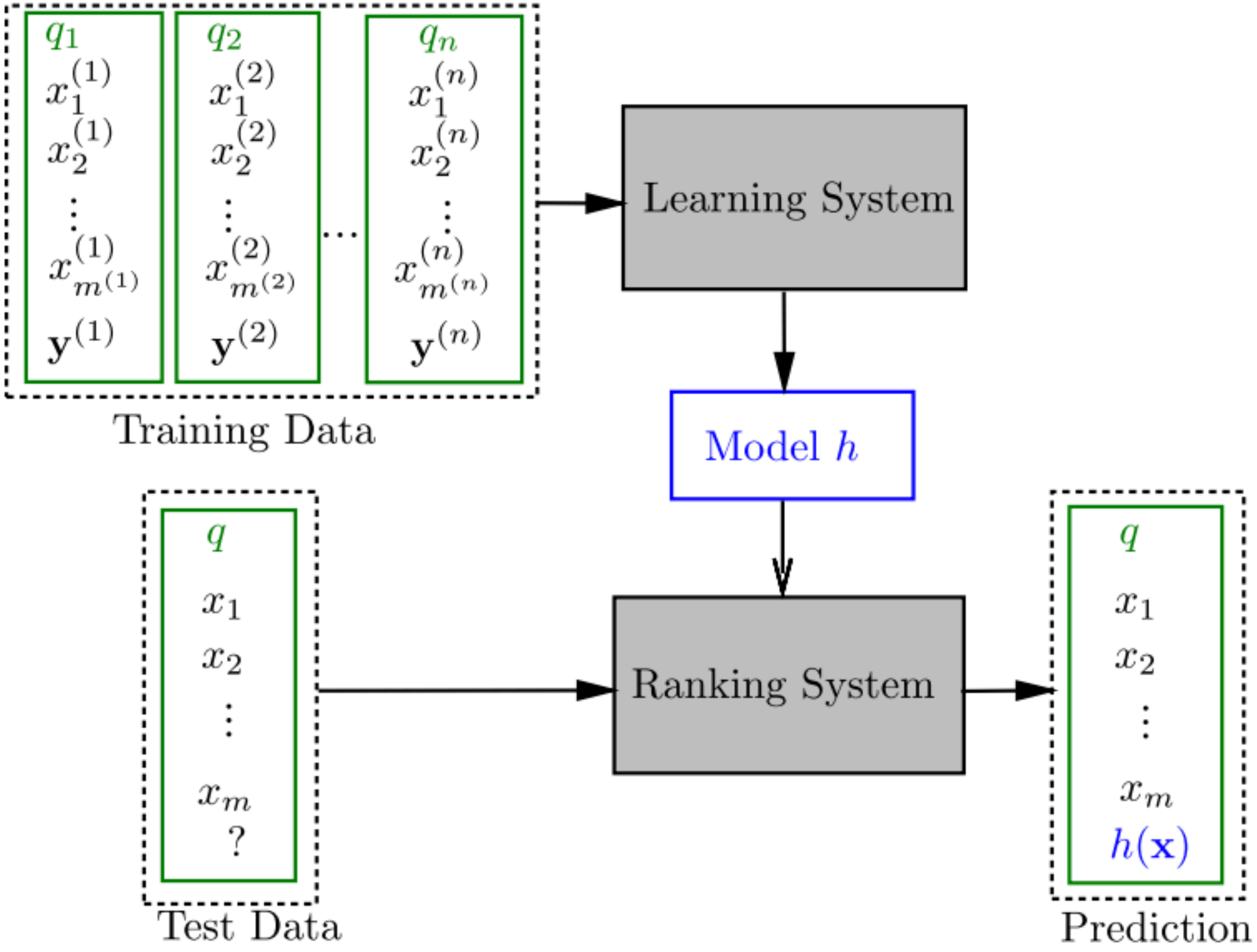
$$RBP = (1-p) \times \sum_{i=1}^{n} r_i \times p^{i-1}$$

Rank biased precision

- Differentiation by rank
- Adjustment for the size of the relevant documents
- But need to decide on the persistence probability *p*

Q4:

How can a retrieval method be learned using supervised machine learning methods? Consider how to frame the learning problem, what data will be required for supervision, and what features are likely to be useful.



Q4:
How can a retrieval method be learned using supervised machine learning methods? Consider how to frame the learning problem, what data will be required for supervision, and what features are likely to be useful.

- User features e.g. search history of the user, location
- Document features e.g. page rank, quality score, topics
- Query features e.g. number of query terms, popularity of the query