INFORMATION RETRIEVAL

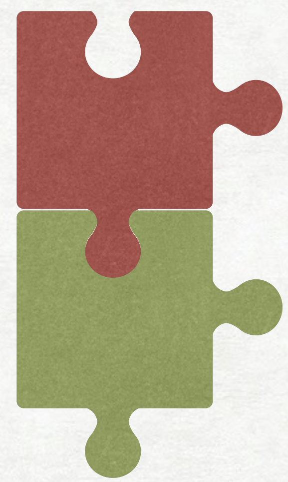
Luca Manzoni
lmanzoni@units.it

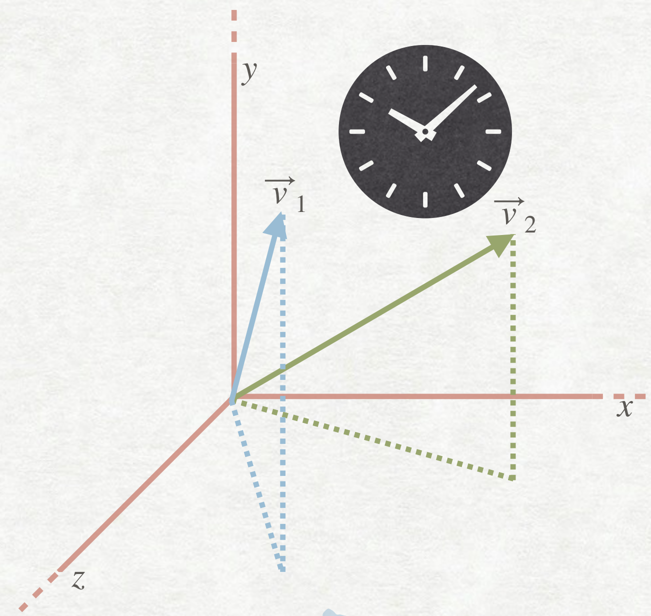
Lecture 6

LECTURE OUTLINE

*SIDE EFFECTS MAY INCLUDE SIDE EFFECTS

Putting Everything Together





Evaluation of IR systems

Computing Similarity Efficiently

COMPUTING SIMILARITY EFFICIENTLY

A FEW INITIAL CONSIDERATIONS

THE LOW-HANGING FRUITS

- We can have an inverted index in which each term has an associated idf_t value (since it depends only on the term).
- Each posting will have the term frequency $\mathrm{tf}_{t,d}$ associated to it (since it depends on both the term and the document).
- We can then compute the score of each document while traversing the posting lists.
- If a DocID does not appear in the posting list of any query term its score is zero.
- To retrieve the K highest scoring documents we can use a *heap* data structure, which is more efficient than sorting all documents.

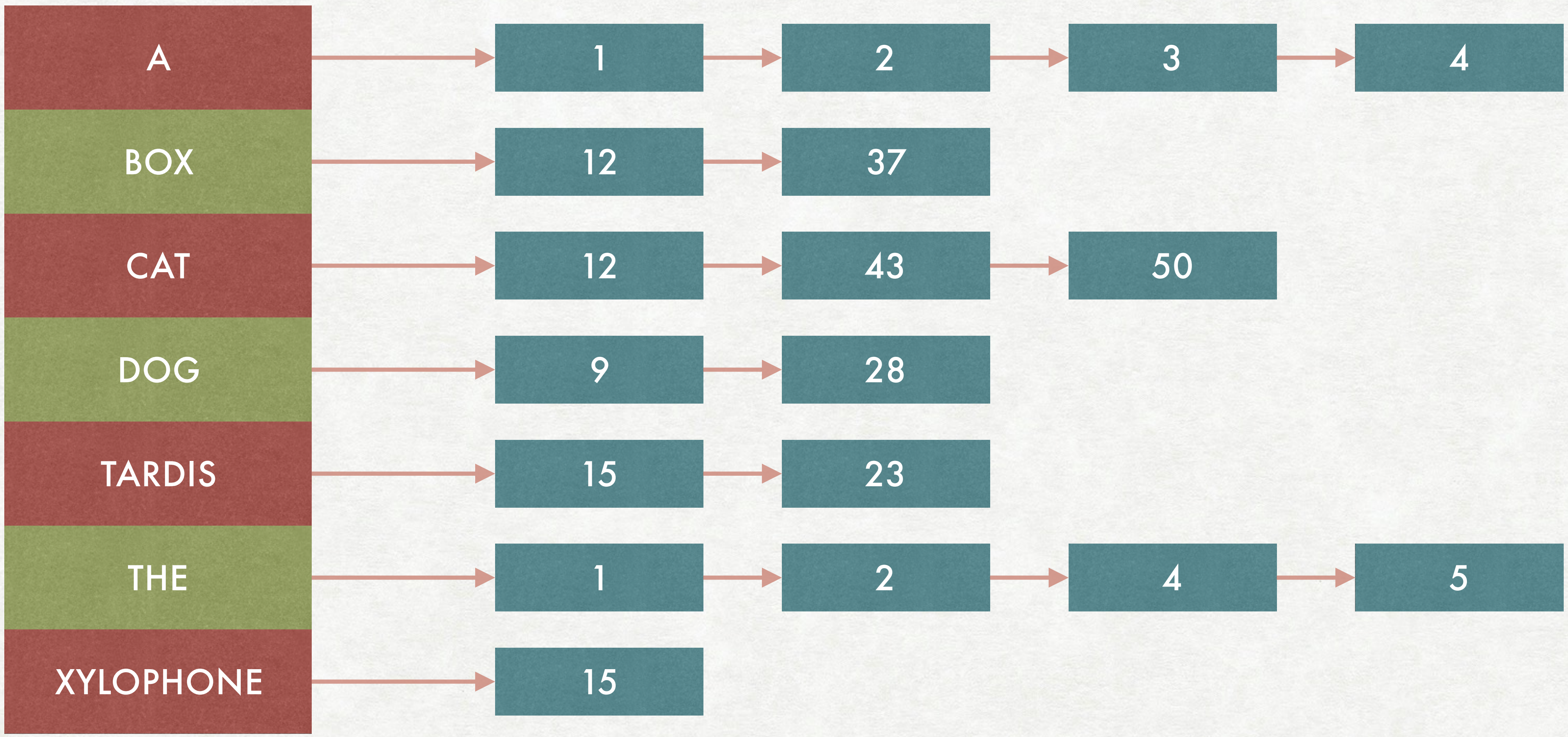
INEXACT TOP K DOCUMENT RETRIEVAL

BEING FAST AND "WRONG"

- Sometimes it is more important to be efficient than to retrieve exactly the K highest scoring documents.
- We want to retrieve K documents that are *likely* to be among the K highest scored.
- Notice that the similarity score is a proxy of the relevance of a document to a query, so we already have some "approximation".
- The main idea to perform an inexact retrieval is:
    - Find a subset A of the documents that is both small and likely to contain documents with scores near to the K highest ranking.
    - Return the K highest ranked documents in A.

INDEX ELIMINATION

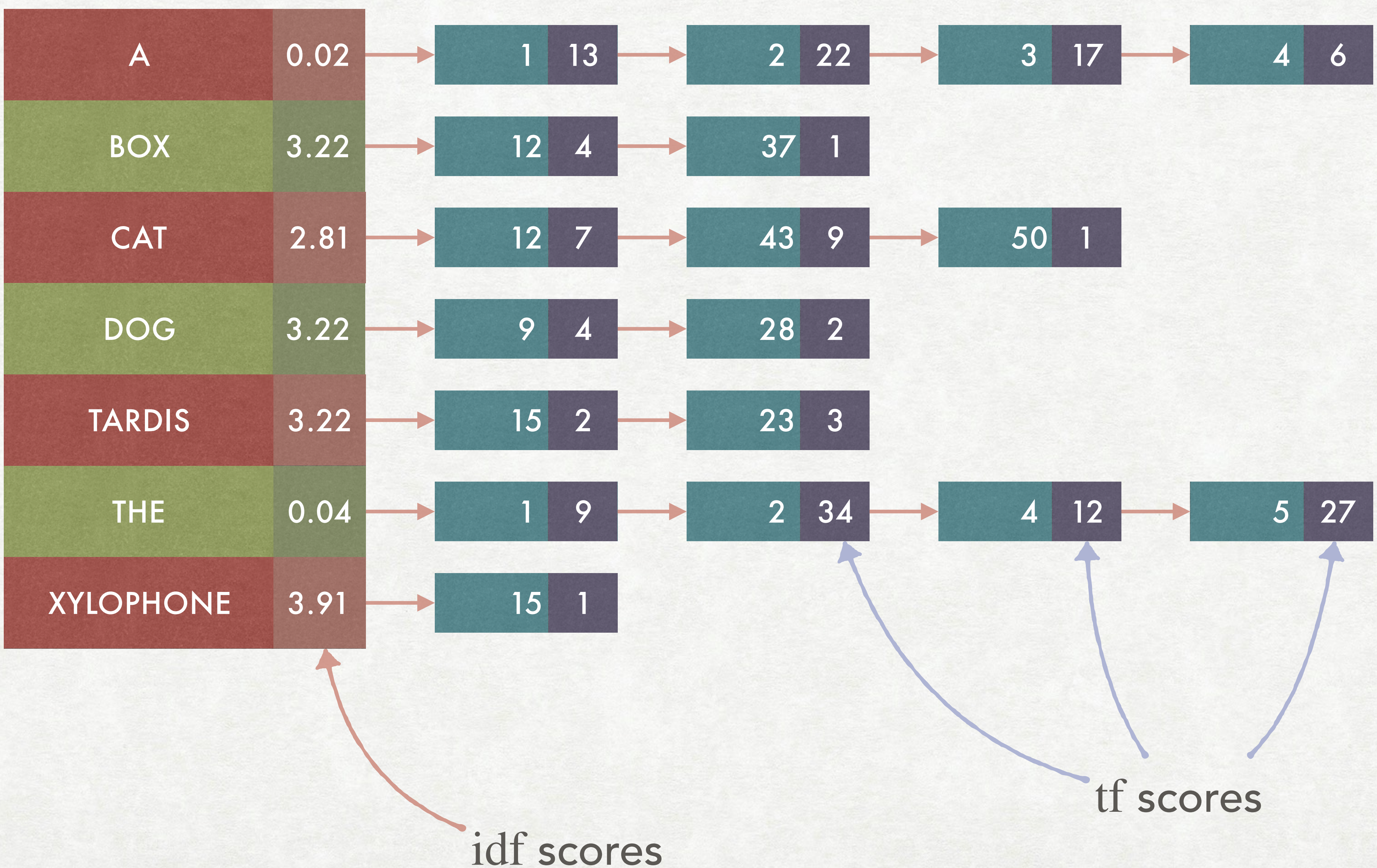
HOW TO IGNORE SOME TERMS



standard inverted index

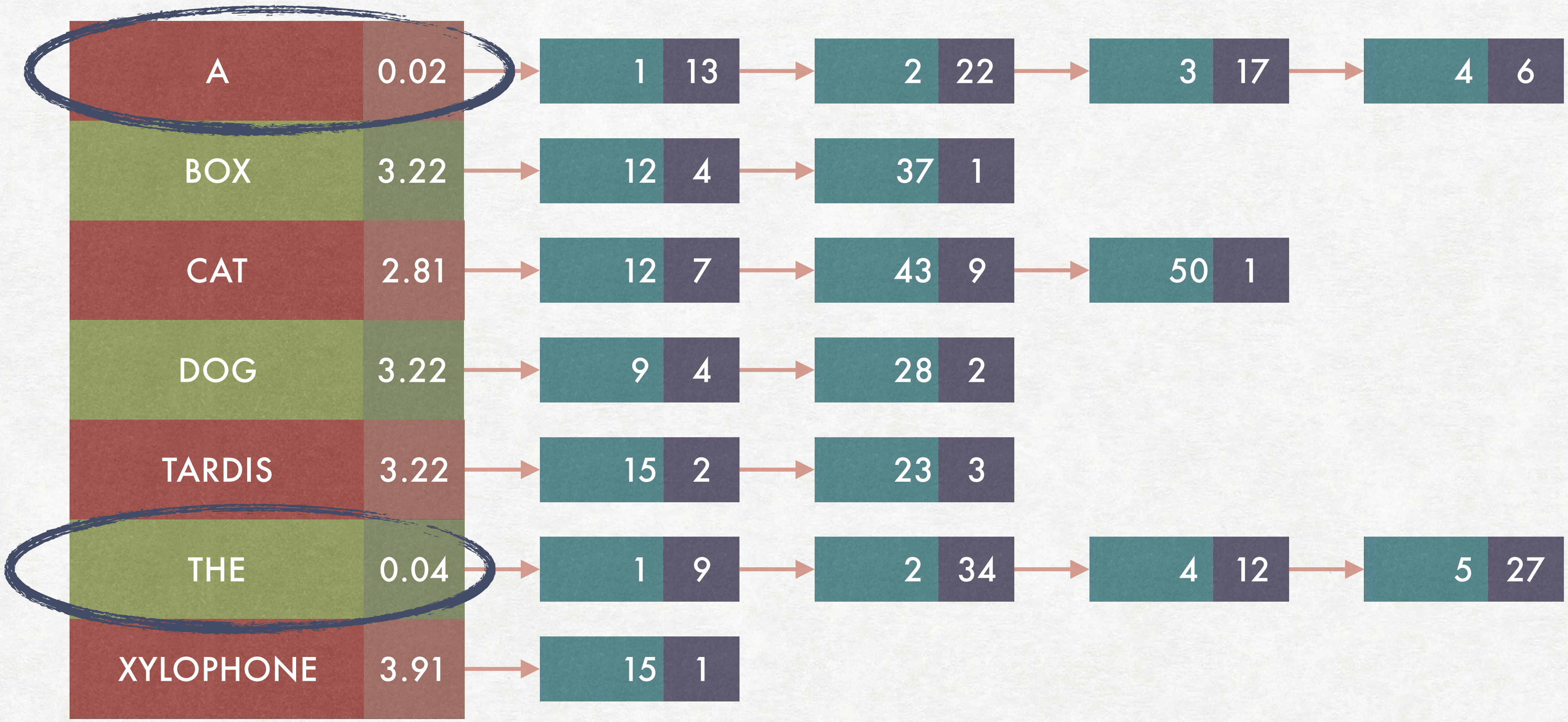
INDEX ELIMINATION

HOW TO IGNORE SOME TERMS



INDEX ELIMINATION

HOW TO IGNORE SOME TERMS



We can remove terms with very low idf score from the search: they are like “stop words” with very long postings list

INDEX ELIMINATION

HOW TO IGNORE SOME TERMS

- By removing terms with low idf value we can only work with relatively shorter lists.
- The cutoff value can be adapted according to the other terms present in the query.
- We can also only consider documents in which most or all the query terms appears…
- …but a problem might be that we do not have at least K documents matching all query terms.

CHAMPION LISTS

OR "TOP DOCS"

- Keep an additional pre-computed list for each term containing only the r highest-scoring documents (usually $r > K$).
- These additional lists are known as *champion lists*, *fancy lists*, or *top docs*.
- We compute the union of the champion lists of all terms in the query, obtaining a set A of documents.
- We find the K highest ranked documents in A.
- Problem: we might have too few documents if K is not known until the query is performed.

STATIC QUALITY SCORES

ADDING A PRE-COMPUTABLE SCORE TO DOCUMENTS

- In some cases we might want to add a score to a document that is independent from the query: a **static quality score**, denoted by $g(d) \in [0,1]$.

- Example: good reviews by users might "push" a document higher in the scoring.

- We need to combine $g(d)$ with the scoring given by the query, a simple possibility is a linear combination: $\text{score}(q, d) = g(d) + \overrightarrow{v}(d) \cdot \overrightarrow{v}(q)$.

- We can also sort posting list by $g(d) + \text{idf}_{t,d}$, to process documents more likely to have high scores first.

IMPACT ORDERING

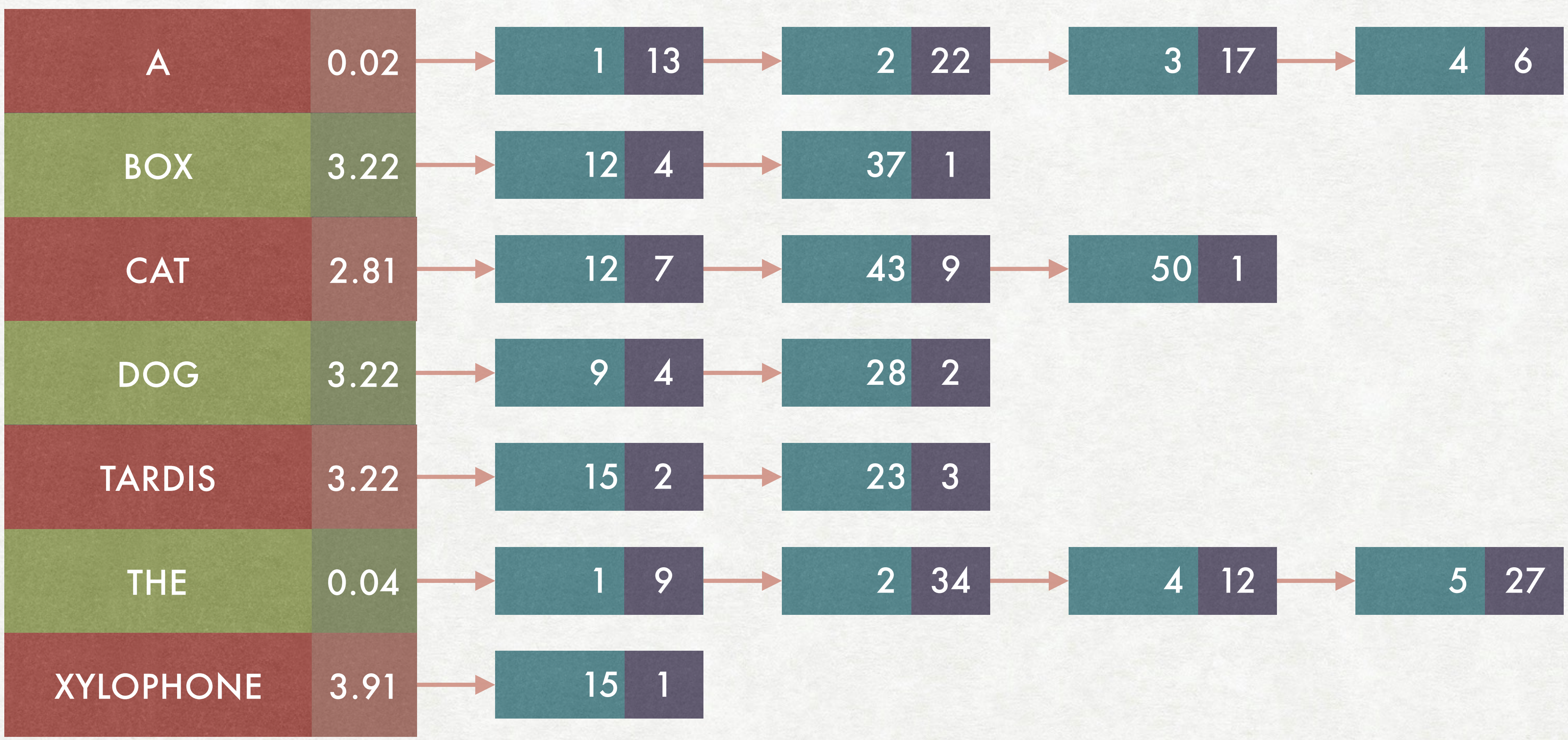
SORTING POSTING LISTS NOT BY DOCID

- Union and intersection for posting lists works efficiently because of the ordering…
- …but everything work as long as they are ordered with some criterium, not necessarily by DocID.
- Idea: Order the documents by decreasing $\mathrm{tf}_{t,d}$. In this way the documents which will obtain the highest scoring will be processed first.
- If the $\mathrm{tf}_{t,d}$ value drops below a threshold, then we can stop.

IMPACT ORDERING

SORTING POSTING LISTS NOT BY DOCID

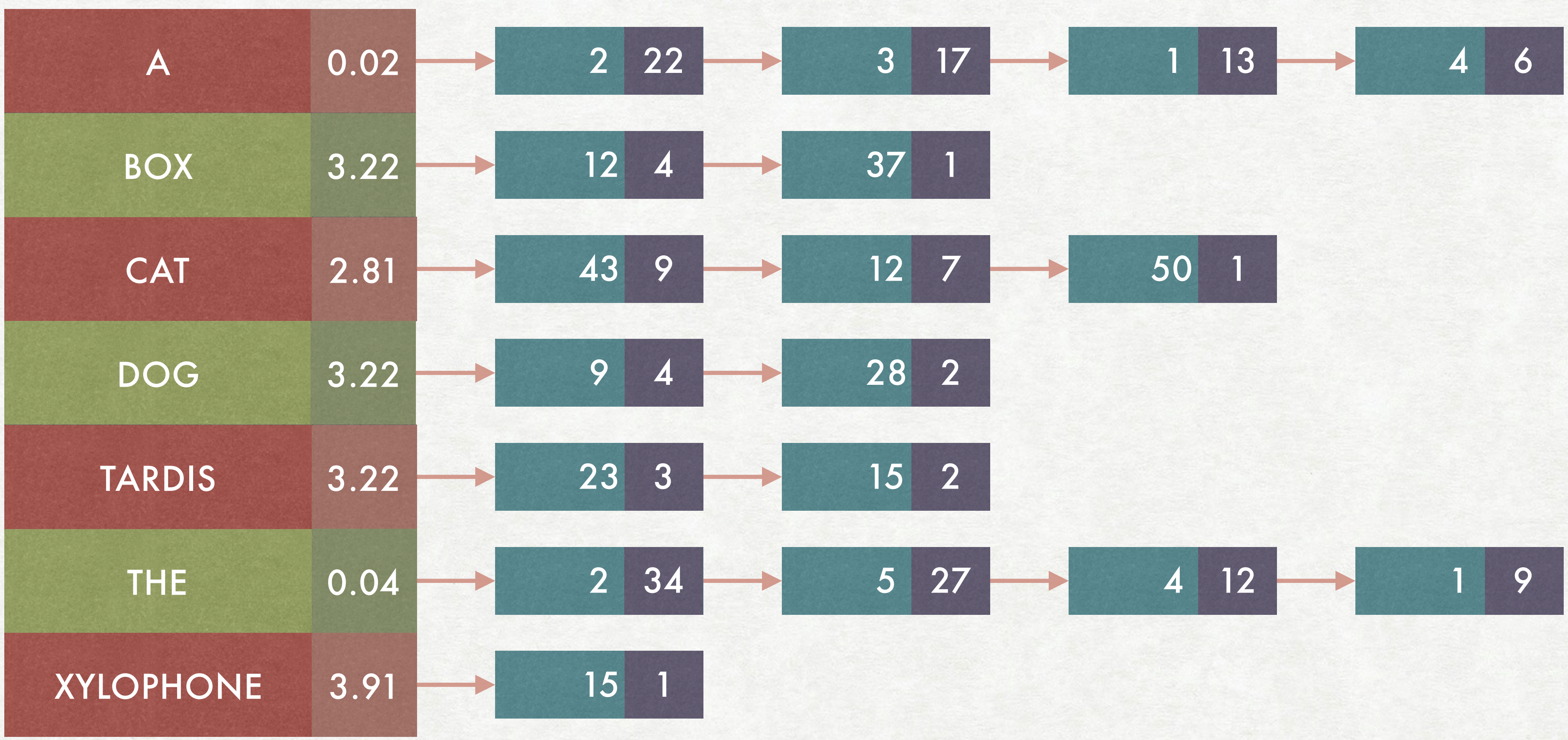
From this...



IMPACT ORDERING

SORTING POSTING LISTS NOT BY DOCID

…to this



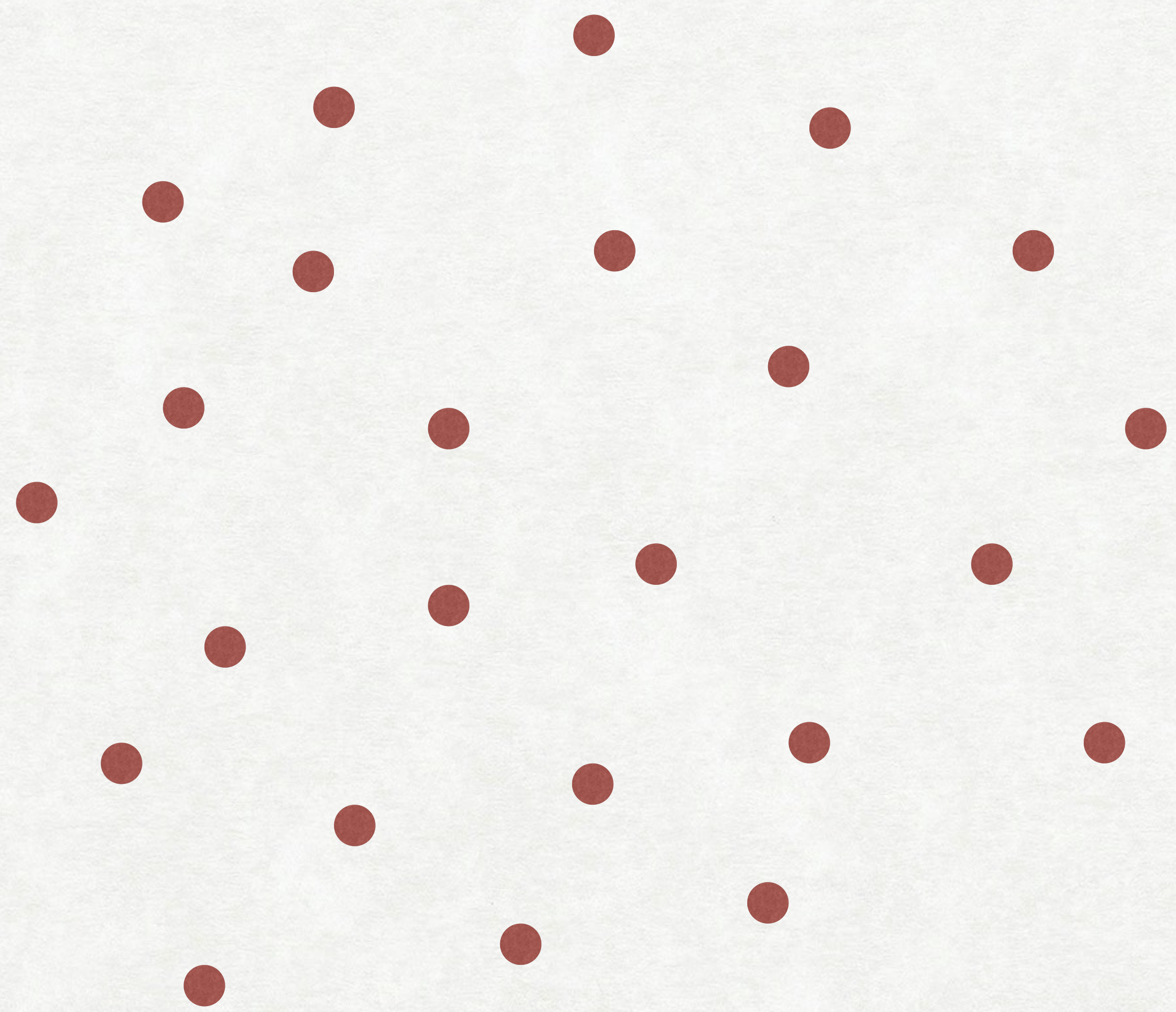
CLUSTER PRUNING

SEARCHING ONLY INSIDE A CLUSTER

- With N document, $M = \sqrt{N}$ are randomly selected as *leaders*. Each leader identifies a cluster of documents.
- For each of the remaining documents, we find the most similar among the M documents selected and we add it to the corresponding cluster.
- For a query q we find the document among the M leaders that is most similar to it.
- The K highest ranked documents are selected among the ones in the cluster of the selected leader.

CLUSTER PRUNING

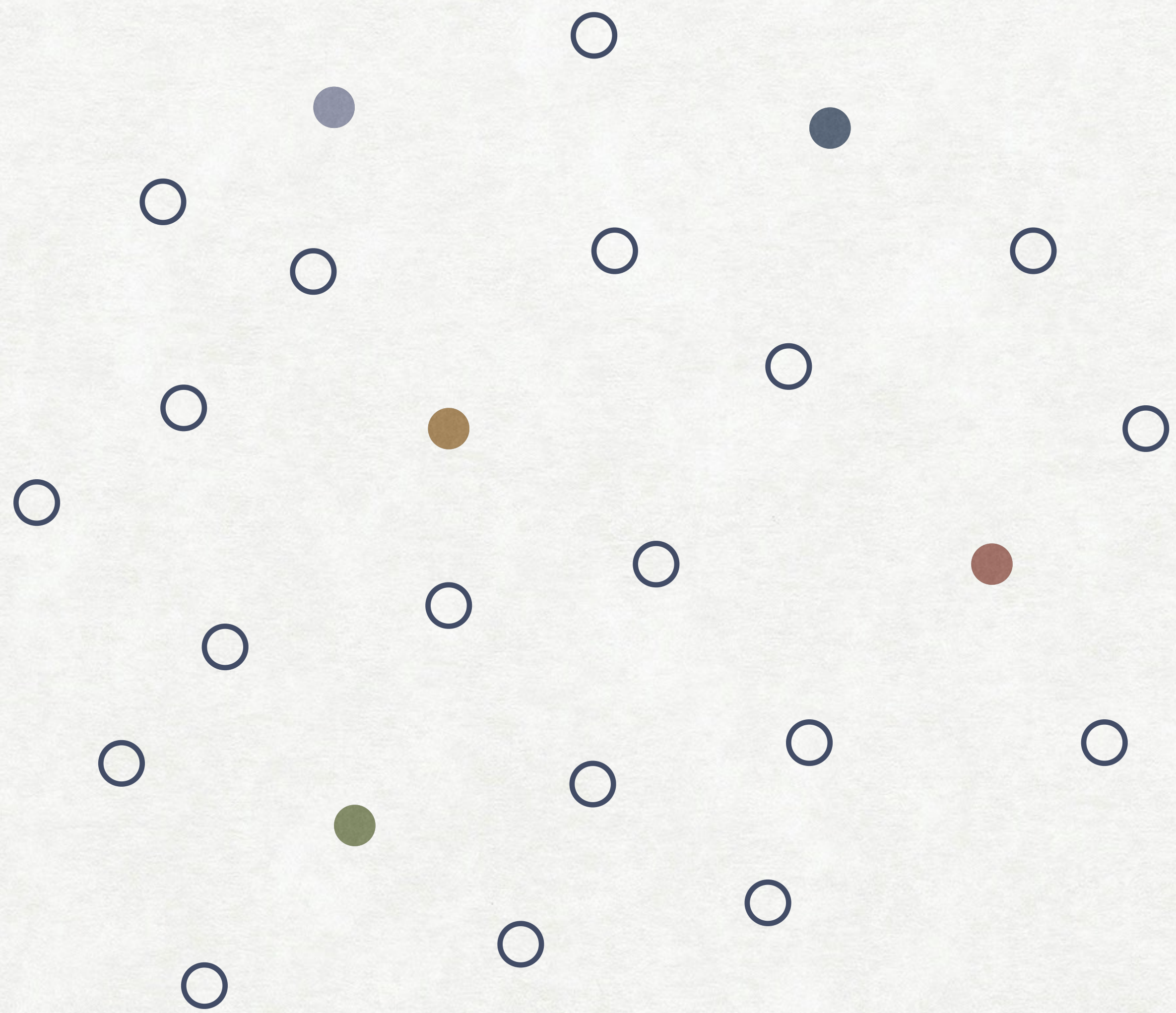
AN EXAMPLE



Documents represented as points in space

CLUSTER PRUNING

AN EXAMPLE



Documents represented as points in space

Selection of the leaders

CLUSTER PRUNING

AN EXAMPLE

Documents represented as points in space

Selection of the leaders

Assigning documents to clusters

CLUSTER PRUNING

AN EXAMPLE

Documents represented as points in space

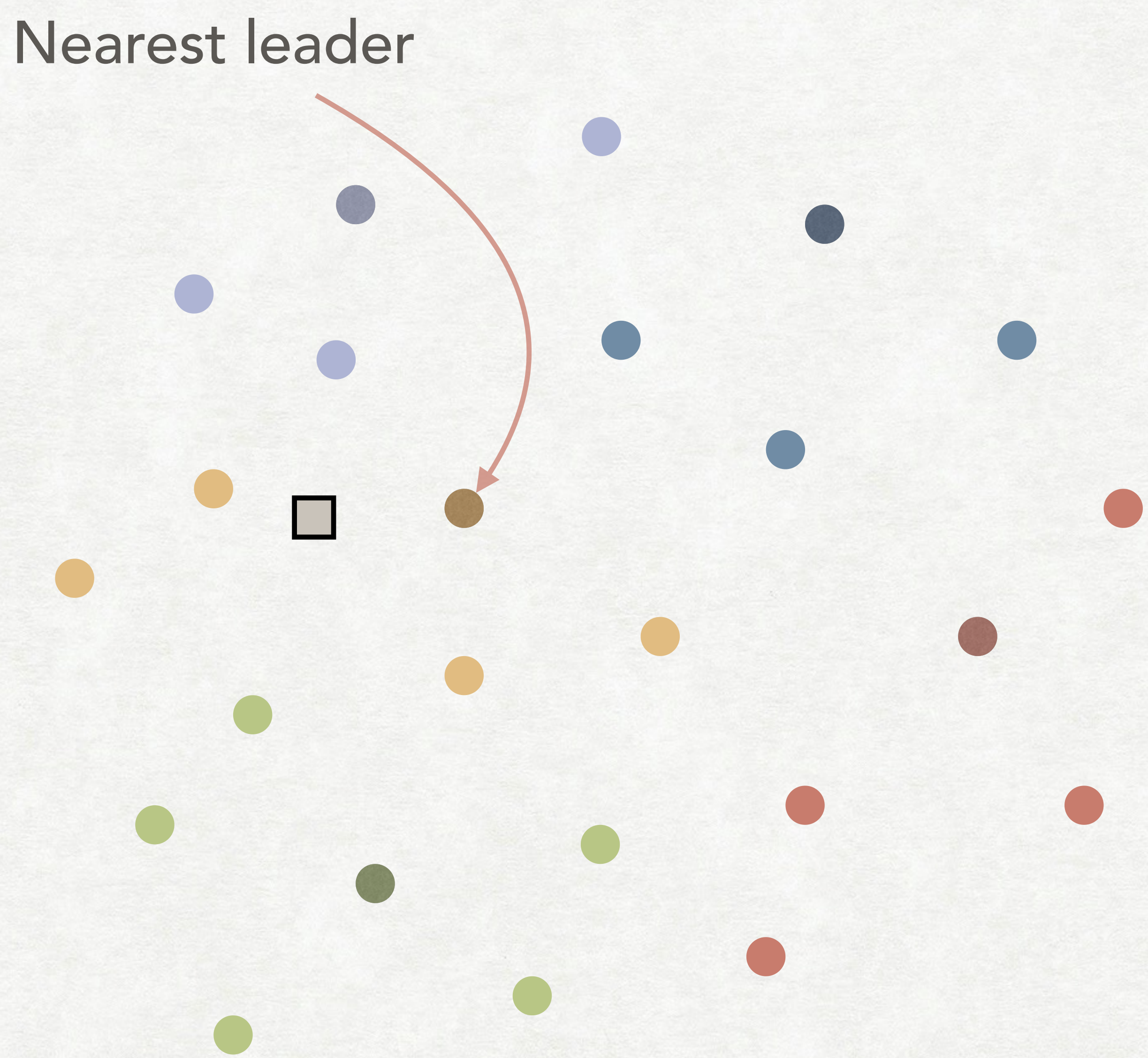
Selection of the leaders

Assigning documents to clusters

A query arrives

CLUSTER PRUNING

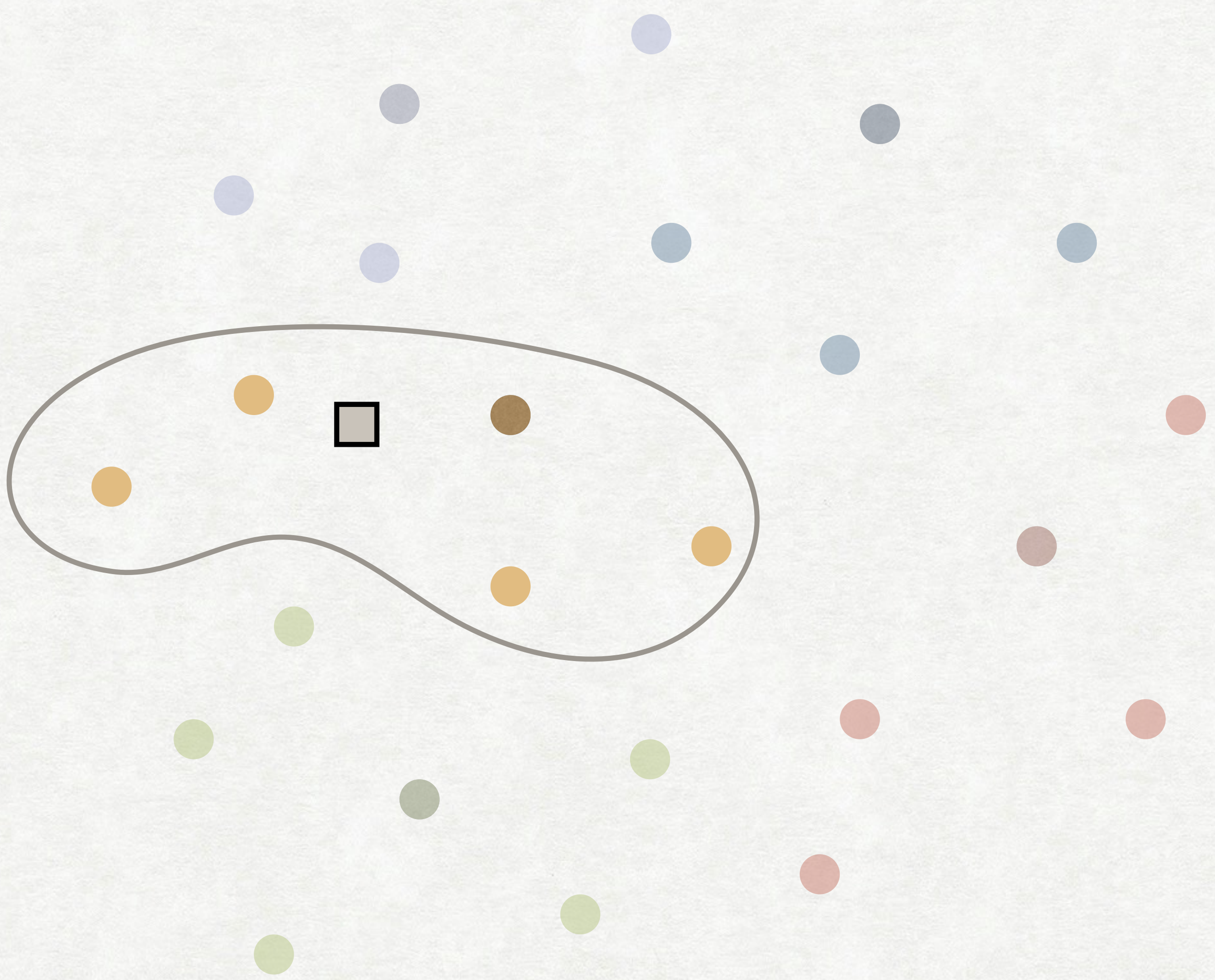
AN EXAMPLE



- Documents represented as points in space
- Selection of the leaders
- Assigning documents to clusters
- A query arrives
- The nearest leader is found

CLUSTER PRUNING

AN EXAMPLE



Documents represented as points in space

Selection of the leaders

Assigning documents to clusters

A query arrives

The nearest leader is found

The similarity is computed only in one cluster

CLUSTER PRUNING

ADDITIONAL CONSIDERATIONS

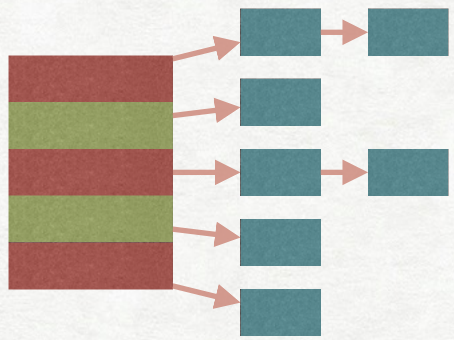
- The selection of $\sqrt{N}$ leaders randomly likely reflects the distribution of documents in the vector space: the most crowded regions will have more leaders.
- A variant more likely to return the "real" K highest ranked document is the following:
    - When creating clusters, each document is associated to b_1 leaders (i.e., it is part of more than one cluster).
    - When a query is received the clusters of the b_2 nearest leaders are considered.

INTEGRATING EVERYTHING

TIERED INDEXES

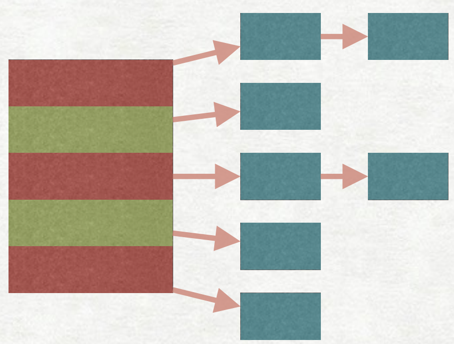
GENERALISATION OF CHAMPION LISTS

Rank 1



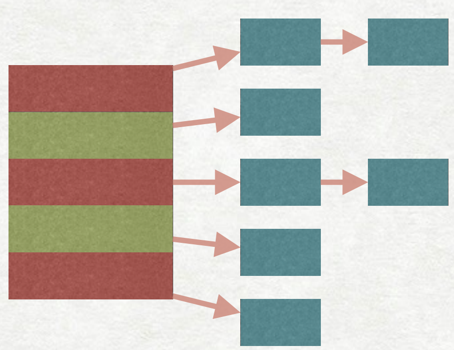
Index for documents with tf over 20

Rank 2



Index for documents with tf between 10 and 20

Rank 3



Index for documents with tf below 10

We search for K documents in the rank 1 index,
if we have less than K we continue in the rank 2 index, and so on

QUERY TERM PROXIMITY

TOWARDS A "SOFT CONJUNCTIVE" SEMANTICS

- If we have a query $q = t_1\ t_2\ \ldots, t_k$ we might want to give a higher score to documents in which the three terms appears close to each other.

- This is not a phrase query, but if the terms appears in close proximity the documents might be an indication that the document is more relevant.

- Let ω the length of the window (in term of number of words) in which $t_1, t_2, \ldots, t_k$ all appear.

QUERY TERM PROXIMITY

TOWARDS A “SOFT CONJUNCTIVE” SEMANTICS

Query: CAT XYLOPHONE

$\omega = 5$

Document 1: THE CAT JUMPED ON THE XYLOPHONE

$\omega =$ a lot more than 5

Document 2: CAT: NOUN, A FELINE […] XYLOPHONE: NOUN, AN […]

How can we use ω in out scoring function?

- Hand-coding a scoring function using ω
- As an additional linear term whose weight we can learn from training samples

BOOLEAN RETRIEVAL
HOW TO PERFORM IT IN THE VECTOR SPACE MODEL

- We can use the vector space representation to perform Boolean retrieval:

- A document d is inside the set of documents denoted by t iff $\overrightarrow{v}(d)_t > 0$ (i.e., if the entry t of the vector of d is positive).

- The reverse is not true: the Boolean model does not keep trap of frequencies.

- The two models are different in a more fundamental way: in the Boolean model the queries are written to *select documents*, in the vector space model queries are a form of *evidence accumulation*.

WILDCARD QUERIES

CAN WE IMPLEMENT IT IN THE VECTOR SPACE MODEL?

- In most cases wildcard queries need an additional (and separate) index.
- We can return, from that index, the set of terms that satisfy the wildcards present in the query.
- Suppose that we have CAT* as a query. We obtain the terms “CAT”, “CATASTROPHE”, and “CATERPILLAR”.
- How can we score a document?
- We simply consider the three terms as “normal” query terms: if a document contains all three of them then it will probably be more relevant.

PHRASE QUERIES

PHRASES IN A “BAG OF WORDS” MODEL

- In the vector space model our documents are “bags of words”, without any ordering, while in phrase queries the ordering is important.
- The two models are, in some sense, incompatible: a bag of words model cannot be directly used for phrase queries.
- They can still be combined in some meaningful way:
  - Perform the phrase query and rank only the documents returned by the query.
  - If less than K documents are present then “reduce” the share query and start again.

EVALUATION OF IR SYSTEMS

STANDARD TEST COLLECTIONS

STANDARD BENCHMARKS

CRANFIELD COLLECTION

ONE OF THE OLDEST, NOW TOO SMALL.
1398 ABSTRACTS OF AERODYNAMICS JOURNAL ARTICLES AND 225 QUERIES.

TREC
(TEXT RETRIEVAL CONFERENCE)

NOT A SINGLE COLLECTION. THERE IS A RANGE OF TEXT COLLECTIONS ON DIFFERENT TOPICS.
SEE : HTTPS://TREC.NIST.GOV

REUTERS

REUTERS-21578 (21578 DOCUMENTS) AND REUTERS-RCV1 (806791 DOCUMENTS) COLLECT A LARGE NUMBER OF NEWSWIRE ARTICLES

Also see: http://ir.dcs.gla.ac.uk/resources/test_collections/

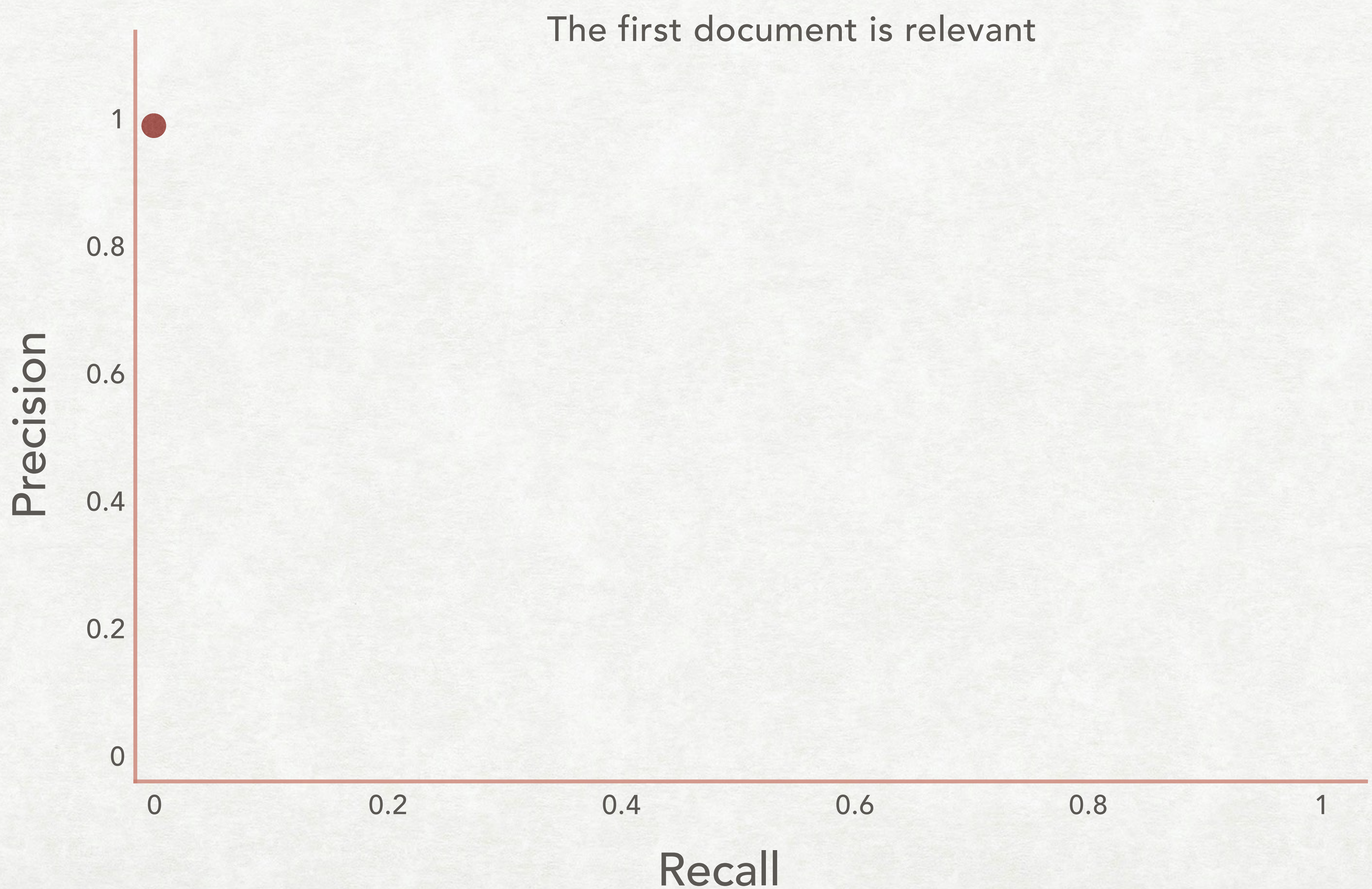
RANKED RETRIEVAL

HOW TO COMPUTE PRECISION AND RECALL?

- We usually evaluate the effectiveness of a IR system with precision and recall (other measures are also possible)…
- …and this works well with *unranked* results.
- How can we extend it to *ranked* results, where position is important?
  - Precision-recall curve and interpolated precision
  - Eleven-point interpolated average precision
  - Mean average precision (MAP)
  - Precision at k and R-precision

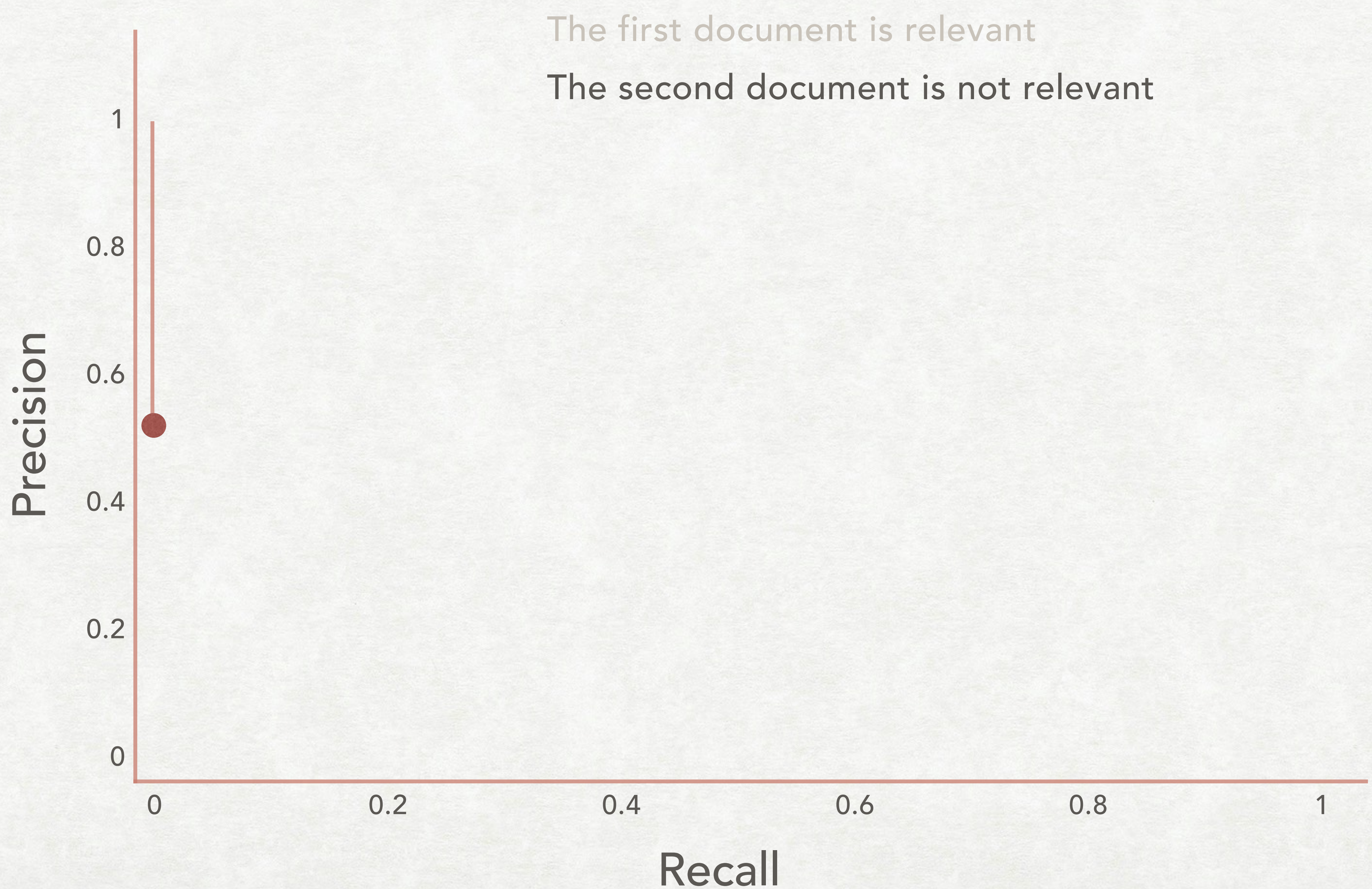
PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



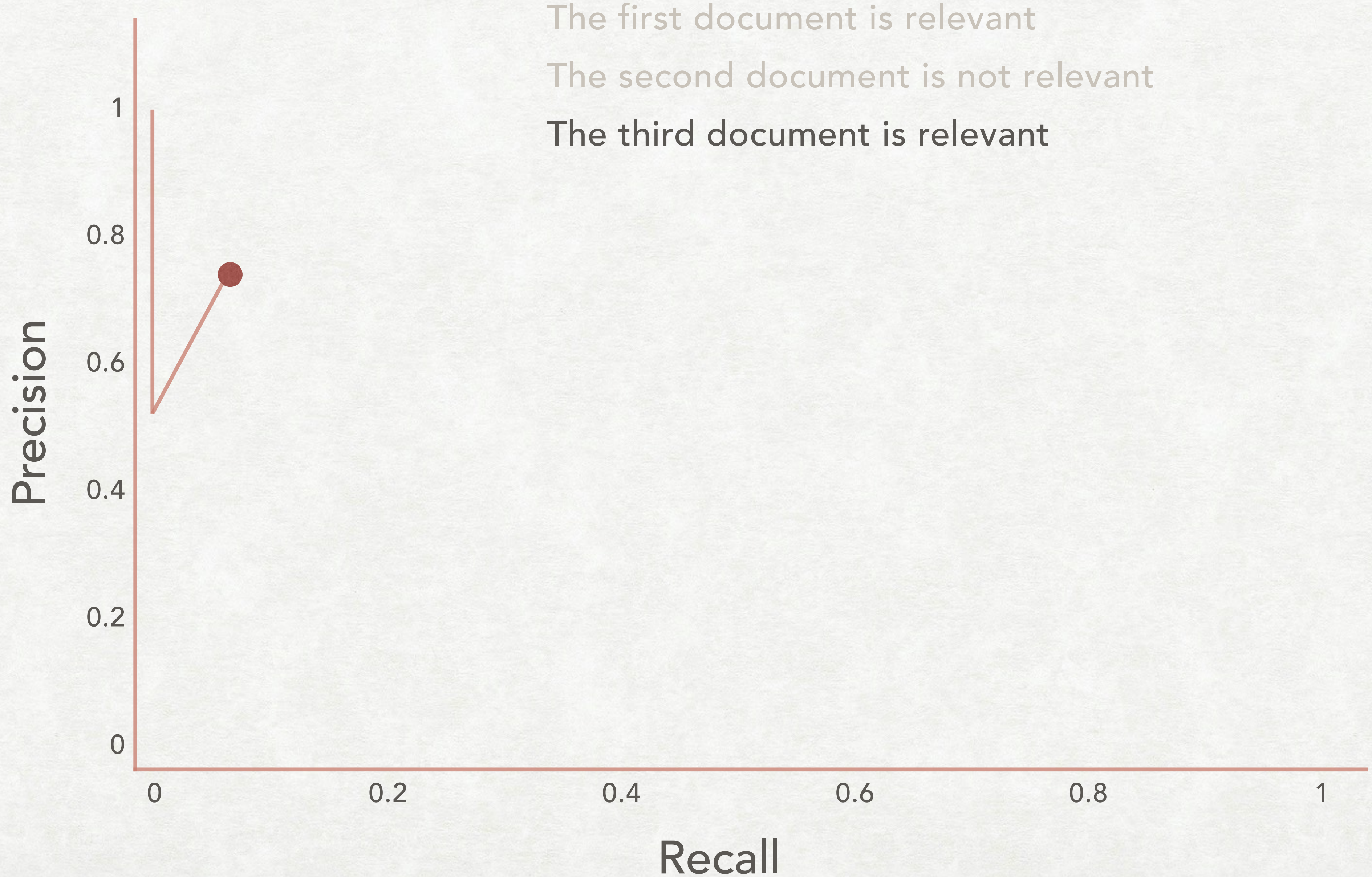
PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



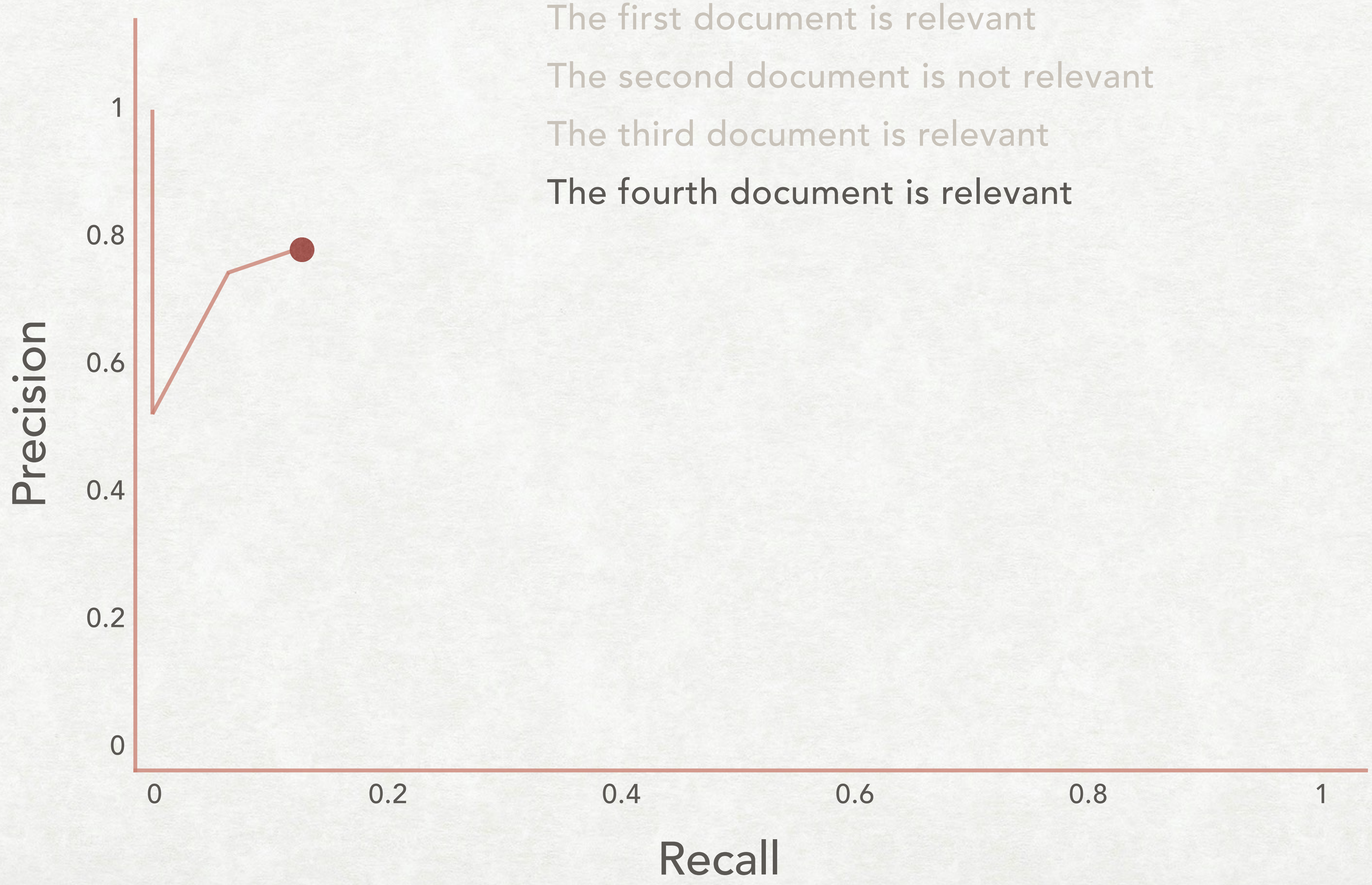
The first document is relevant

The second document is not relevant

The third document is relevant

PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



The first document is relevant

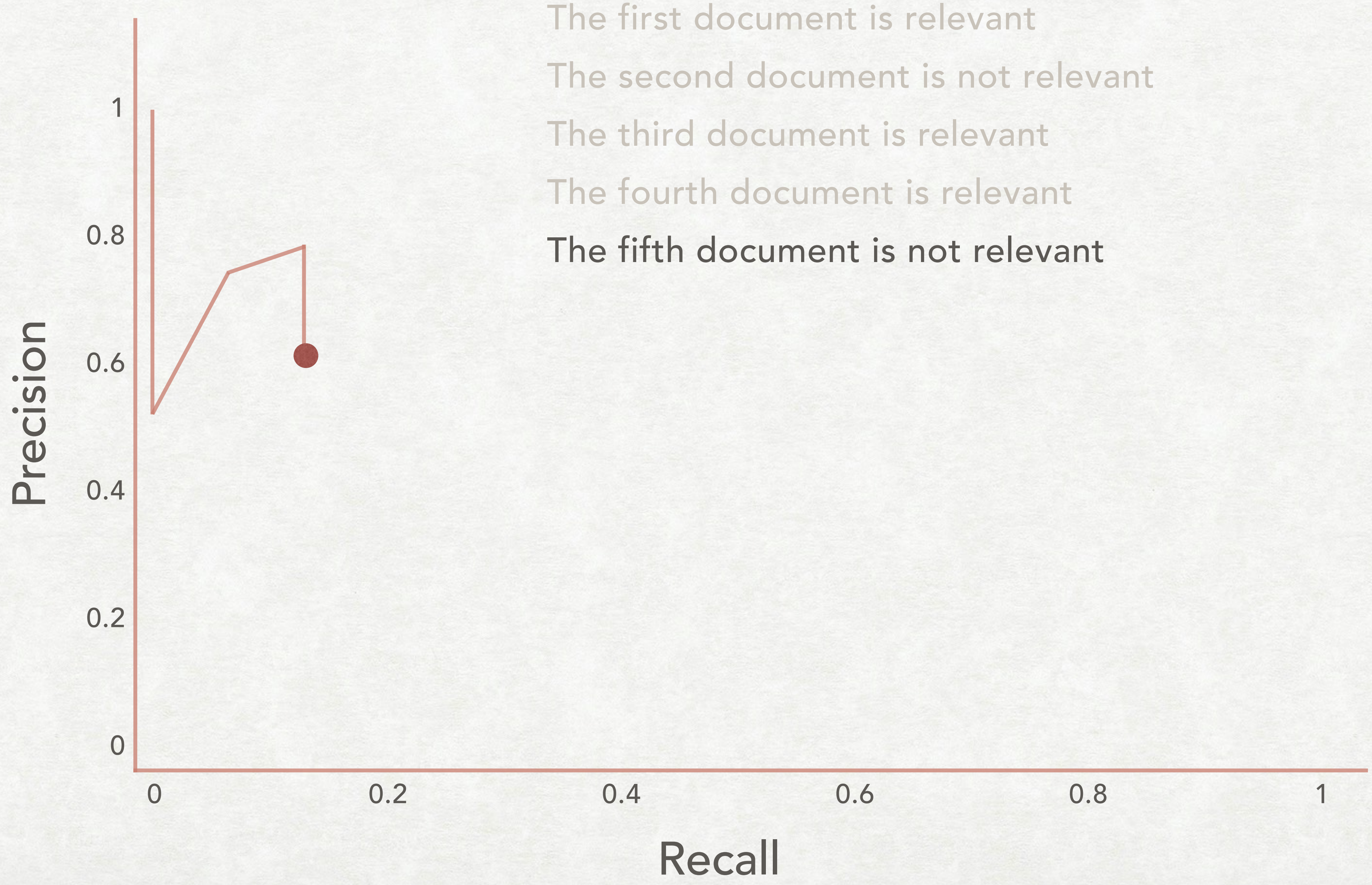
The second document is not relevant

The third document is relevant

The fourth document is relevant

PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



The first document is relevant

The second document is not relevant

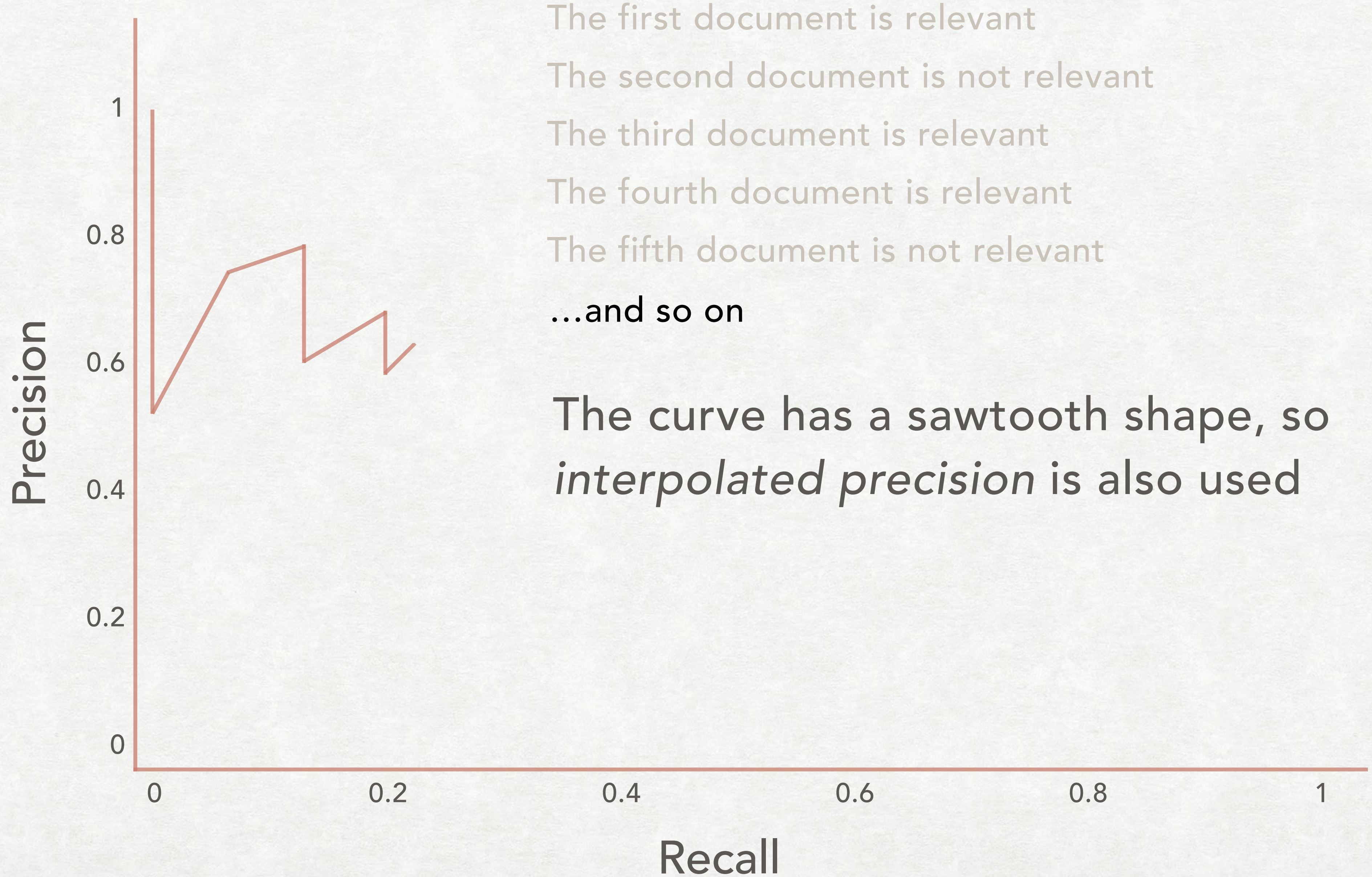
The third document is relevant

The fourth document is relevant

The fifth document is not relevant

PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



The first document is relevant

The second document is not relevant

The third document is relevant

The fourth document is relevant

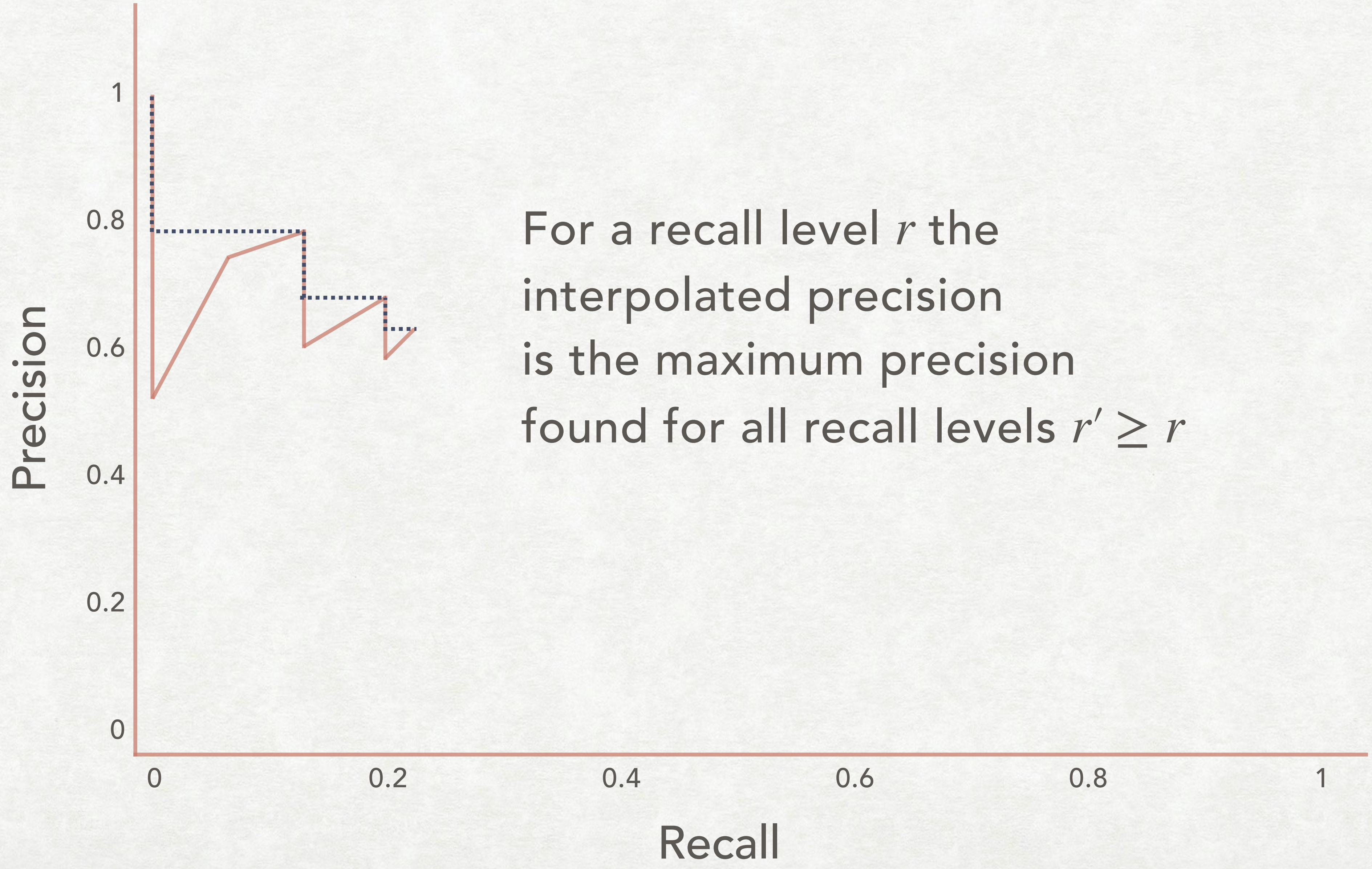
The fifth document is not relevant

…and so on

The curve has a sawtooth shape, so *interpolated precision* is also used

PRECISION-RECALL CURVE

We compute precision and recall for the first 1, 2, 3, 4, etc. retrieved documents:



For a recall level r the interpolated precision is the maximum precision found for all recall levels $r' \geq r$

ELEVEN POINT INTERPOLATED PRECISION

PRECISION AT ELEVEN RECALL LEVELS

Recall	Precision
0.0	1.0
0.1	0.73
0.2	0.64
0.3	0.58
0.4	0.51
0.5	0.45
0.6	0.38
0.7	0.27
0.8	0.21
0.9	0.13
1.0	0.09

The recall levels are fixed and for each recall level the corresponding precision is recorded.

MEAN AVERAGE PRECISION

A SINGLE FIGURE

We have a set of queries $Q = \{q_1, \ldots, q_n\}$

For each q_j we know the set of documents $\{d_1, \ldots, d_{m_j}\}$ that are relevant

Let R_{jk} the set of ranked documents retrieved for the j^{th} query that we get to obtain k relevant documents

Then the mean average precision $\mathrm{MAP}(Q)$ is:

$$\frac{1}{n}\sum_{j=1}^{n}\left(\underbrace{\frac{1}{m_j}\sum_{k=1}^{m_j}\mathrm{Precision}(R_{jk})}_{\text{Average precision of the } j^{\text{th}} \text{ query}}\right)$$

PRECISION AT K AND R-PRECISION

OTHER SINGLE FIGURES

- Precision at k simply means that we record the precision of the first k retrieved documents. Like “precision at 10”.
- If there are less than k relevant documents then the value cannot be one. Its value is highly dependant on the number of relevant documents that exists.
- A solution to this is the R-precision. If there are R relevant documents for a query, the R-precision is the precision of the top R ranked documents returned by the query.
- R-precision can be averaged across queries.