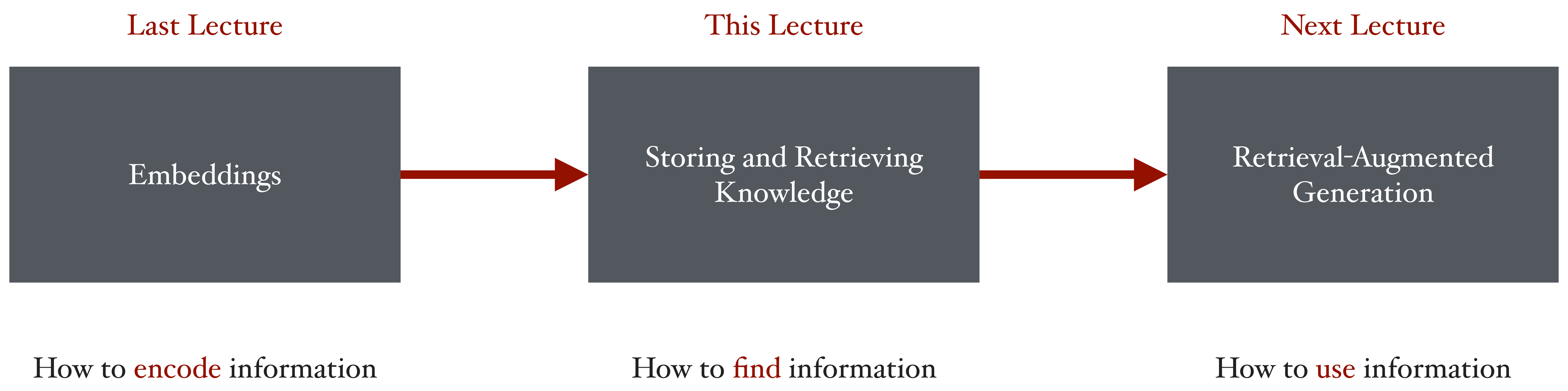
Large Language Model Applications

Storing and Retrieving Knowledge

Daphne Ippolito and Fernando Diaz

January 29, 2026

From Embeddings to Knowledge Access



Two Ways to "Know" Something

Parametric Knowledge

Encoded in model weights

+ Fast inference
+ No dependencies
- Static, expensive to update
- Hallucination-prone
- Unverifiable

Non-Parametric Knowledge

Retrieved from external corpus

+ Updatable, auditable
+ Citable, verifiable
+ Scales independently
- Retrieval latency
- Errors propagate

Information retrieval

given an information need and a corpus of candidate responses, return relevant answers.

Information retrieval

given an information need and a corpus of candidate responses, return relevant answers.

information need: explicit or implicit request for information

Information retrieval

given an information need and a corpus of candidate responses, return relevant answers.

information need: explicit or implicit request for information
corpus: partial, exact, or overcomplete answers

Information retrieval

given an information need and a corpus of candidate responses, return relevant answers.

information need: explicit or implicit request for information
corpus: partial, exact, or overcomplete answers
relevance: satisfying the information need

Information retrieval

web search

given keywords and a web crawl, return relevant URLs.

Information retrieval

image search

given keywords and an image corpus, return relevant images.

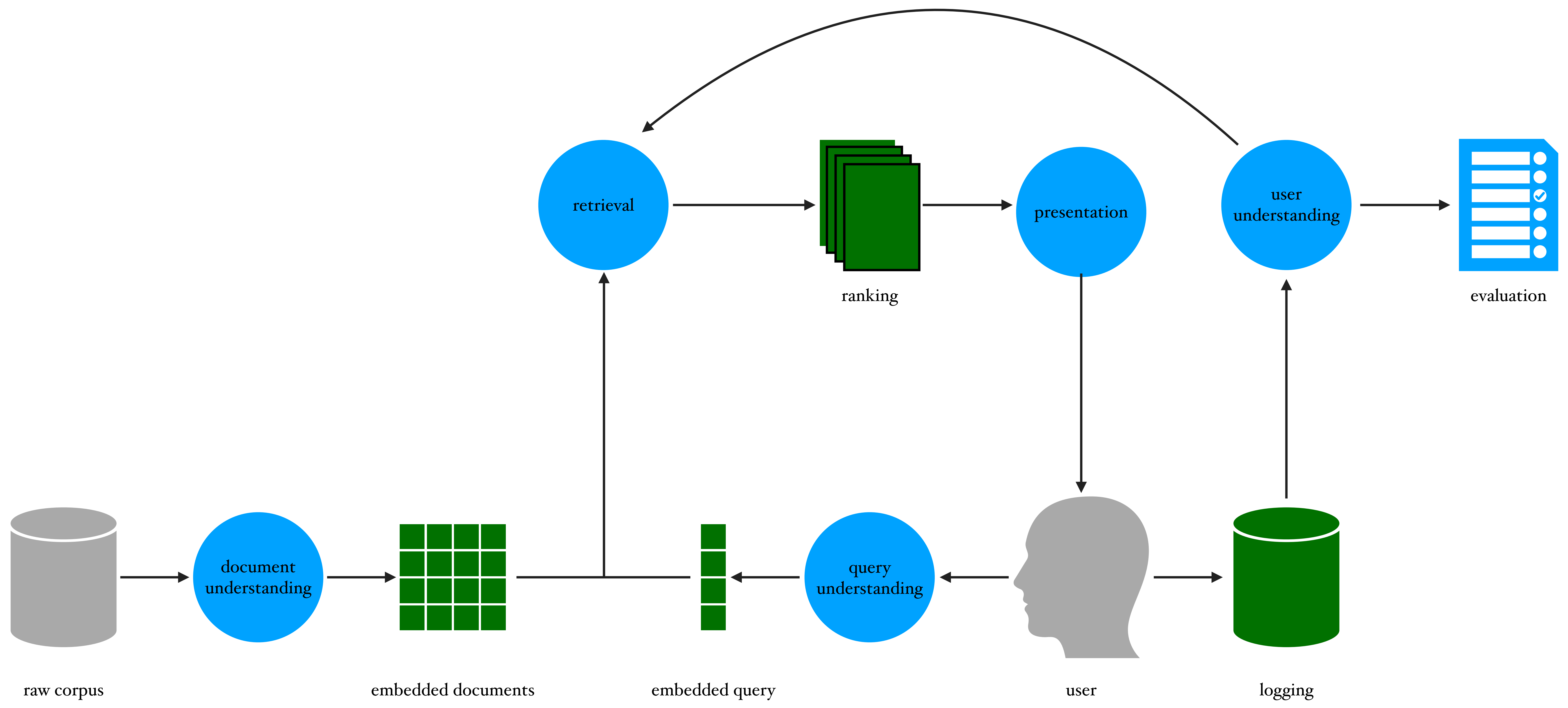
Information retrieval

music recommendation

given a user context and a catalog of music, return relevant tracks.

Information retrieval

Schematic



Classic retrieval

Bag of words: sparse lexical embeddings

- Corpus: set of text documents; can be passages or other text strings.
  - Document representation: sum of one-hot vectors associated with terms in the document
- Query: text string
  - Query representation: sum of one-hot vectors associated with terms in the query
- Ranking function: inner product between query and document representations; sort by cosine similarity
  - can use inverted indices and branch and bound techniques to do highly-efficient top-k ranking
  - think about how quickly Google returns results for a corpus of billions of documents

Classic retrieval

BM25

$$d_w = \overbrace{\frac{|w \cap D| \times (k_1 + 1)}{|w \cap D| + k_1 \times \left(1 - b + b \times \frac{|d|}{\bar{d}}\right)}}^{\text{TF}} \times \overbrace{\ln\left(\frac{n - c_w + \frac{1}{2}}{c_w + \frac{1}{2}} + 1\right)}^{\text{IDF}}$$

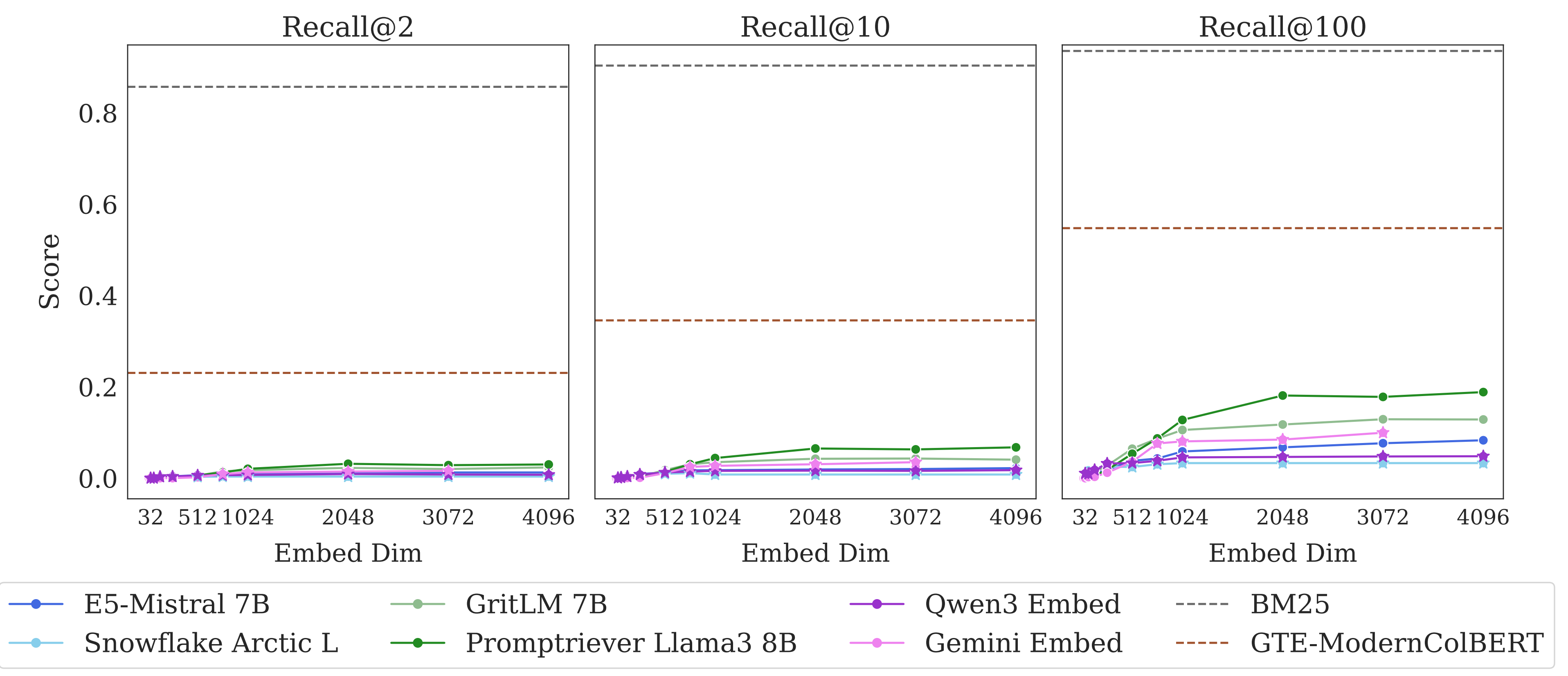
$$q_w = \mathrm{I}\left[w \in Q\right]$$

D	bag of document terms
Q	bag of query terms
c_w	document frequency of w in the corpus
n	corpus size

Classic retrieval

BM25

- Although older, still a highly competitive baseline, especially if you need a reliable technology.
- On relatively simple queries with complex combinations of relevant items, BM25 can have substantially higher effectiveness.
- However, it can still underperform when semantic understanding is necessary.
    - Can be interpreted as "missing terms in the query mean missing non-zeros in the query vector"



Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval. ICLR, 2026.

Classic retrieval

BM25

$$d_w = \overbrace{\frac{|w \cap D| \times (k_1 + 1)}{|w \cap D| + k_1 \times \left(1 - b + b \times \frac{|d|}{\bar{d}}\right)}}^{\text{TF}} \times \overbrace{\ln\left(\frac{n - c_w + \frac{1}{2}}{c_w + \frac{1}{2}} + 1\right)}^{\text{IDF}}$$

if we can make this denser, then we can reuse existing logic

$$\boxed{q_w = \mathrm{I}\left[w \in Q\right]}$$

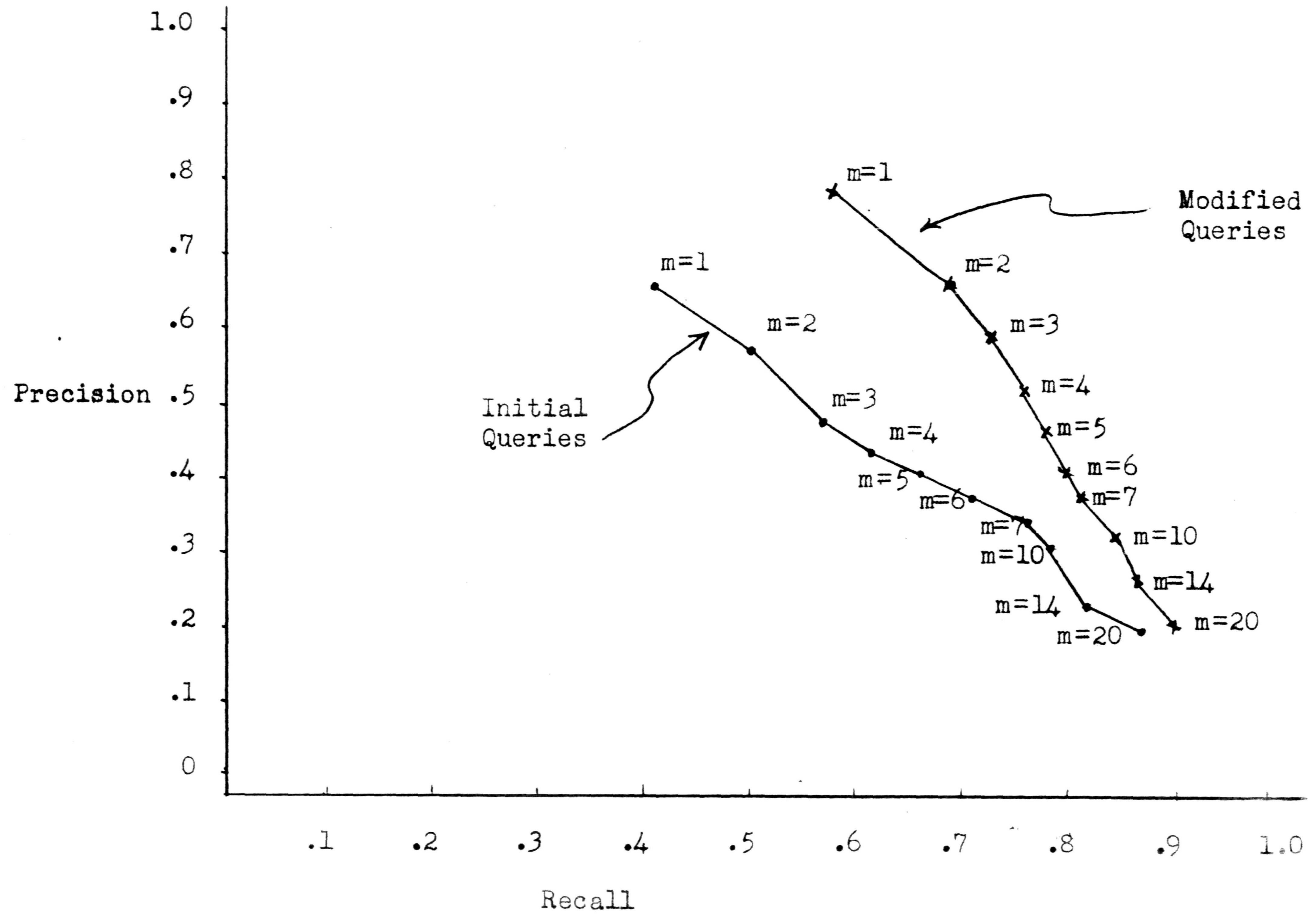
D	bag of document terms
Q	bag of query terms
c_w	document frequency of w in the corpus
n	corpus size

poll: how can we make the query vector denser?

Relevance feedback

The Rocchio algorithm

- One way to do query expansion is to get a small set of relevant documents (few-shot examples).
- The Rocchio algorithm interpolates the original query vector with a linear combination of the relevant document vectors.
- Improves query vector resolution because,
  - documents are denser (reweight)
  - documents have terms not in the query (expand)
- If you have relevant documents, this is a robust result for vector space model algorithms.



$$q' = \lambda q + \frac{(1-\lambda)}{|C^*|} \sum_{d \in C^*} d$$

C^* example relevant documents

poll: what is the problem with relevance feedback?

Pseudo-Relevance feedback

Relevance Models

- The biggest problem with Rocchio, in practice is finding example relevant documents
  - Users do not want to provide with every query
- How can to obtain the relevant documents?
- We can use an existing retrieval system to guess the set!
- Pseudo-relevance feedback with Relevance Models,
  1. score all documents using BM25 (or any ranker)
  2. interpolate the query with a combination of the top *k* documents, weighted by score
  3. score all documents using BM25 and the new query

$$q' = \lambda q + (1 - \lambda) \sum_{d \in C^q} s_d \times d$$

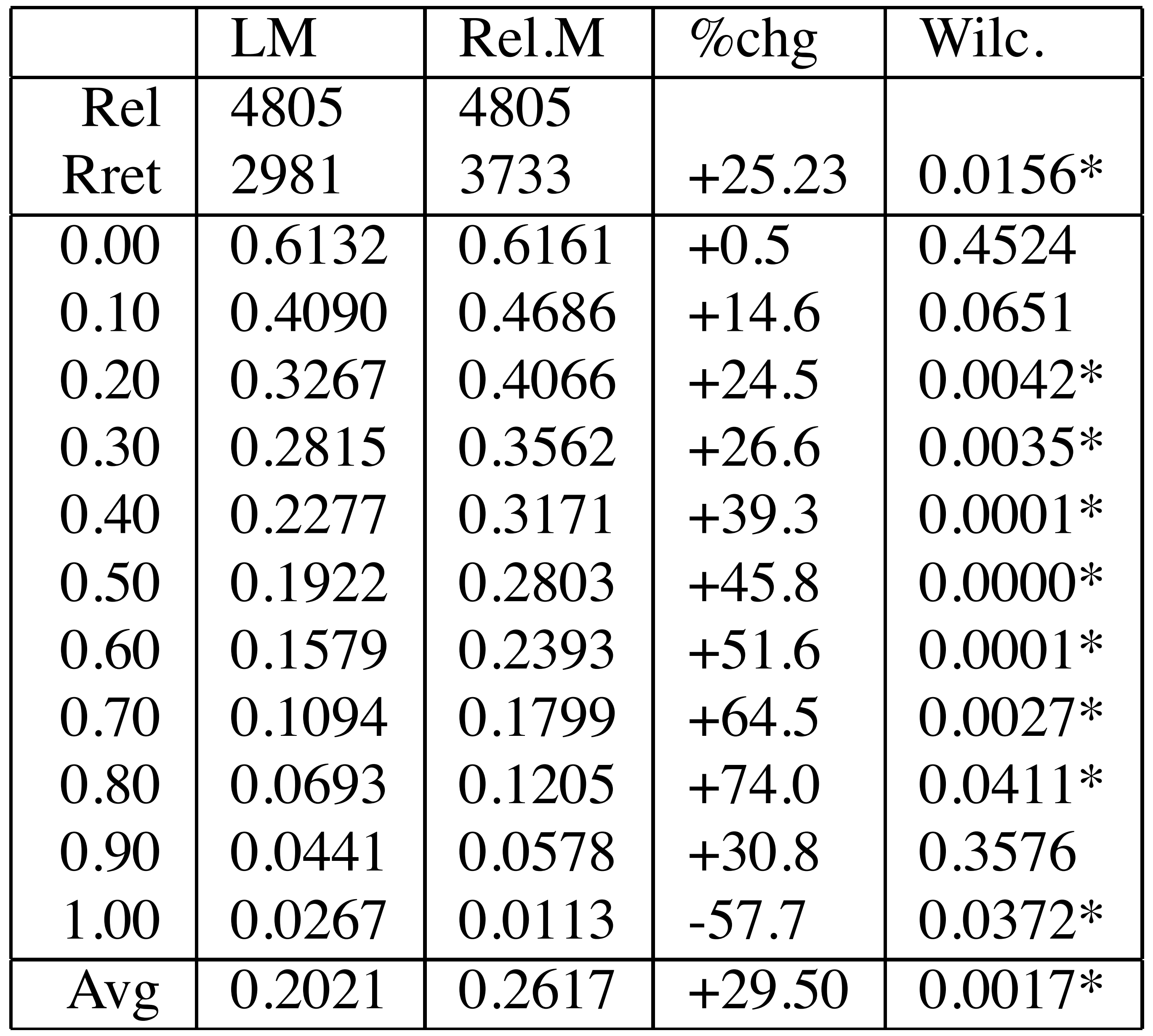
C^q	top k documents from initial retrieval
s_d	ℓ_1-normalized score of document d

Victor Lavrenko and W. Bruce Croft. Relevance based language models. In Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval, 2001.

Pseudo-Relevance feedback

Relevance Models

- The biggest problem with Rocchio, in practice is finding example relevant documents
  - Users do not want to provide with every query
- How can to obtain the relevant documents?
- We can use an existing retrieval system to guess the set!
- Pseudo-relevance feedback with Relevance Models (RM3),
  1. score all documents using BM25 (or any ranker)
  2. interpolate the query with a combination of the top *k* documents, weighted by score
  3. score all documents using BM25 and the new query



	LM	Rel.M	%chg	Wilc.
Rel	4805	4805		
Rret	2981	3733	+25.23	0.0156*
0.00	0.6132	0.6161	+0.5	0.4524
0.10	0.4090	0.4686	+14.6	0.0651
0.20	0.3267	0.4066	+24.5	0.0042*
0.30	0.2815	0.3562	+26.6	0.0035*
0.40	0.2277	0.3171	+39.3	0.0001*
0.50	0.1922	0.2803	+45.8	0.0000*
0.60	0.1579	0.2393	+51.6	0.0001*
0.70	0.1094	0.1799	+64.5	0.0027*
0.80	0.0693	0.1205	+74.0	0.0411*
0.90	0.0441	0.0578	+30.8	0.3576
1.00	0.0267	0.0113	-57.7	0.0372*
Avg	0.2021	0.2617	+29.50	0.0017*

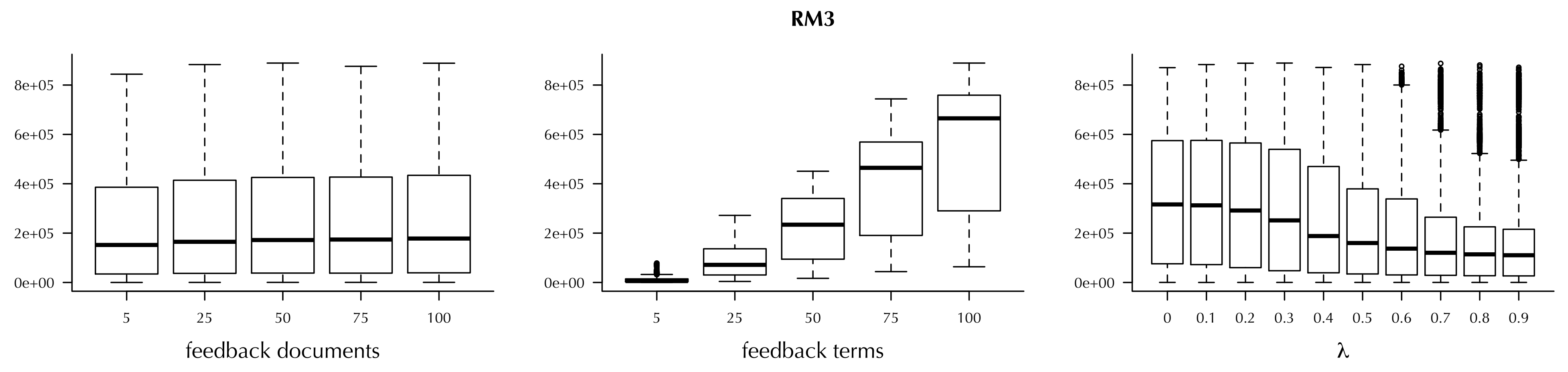
Victor Lavrenko and W. Bruce Croft. Relevance based language models. In Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval, 2001.
Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. Umass at trec 2004: novelty and hard. In Online proceedings of 2004 text retrieval conference, 2004.

poll: what is the problem with pseudo-relevance feedback?

Pseudo-Relevance feedback

Condensed List Relevance Models

- The biggest drawback of RM3 is the penalty of doing two retrievals.
- For a large search engine with a lot of users, this is 2x the cost!
- However, if we look at the positions of the documents for the first and second retrievals, we find high overlap...



rank	trec12	robust	web
1	1	0.999	0.998
2	0.996	0.996	0.995
3	0.994	0.993	0.996
4	0.996	0.989	0.993
5	0.992	0.987	0.991
⋮	⋮	⋮	⋮
996	0.374	0.416	0.389
997	0.383	0.385	0.388
998	0.368	0.363	0.359
999	0.377	0.385	0.383
1000	0.395	0.387	0.402

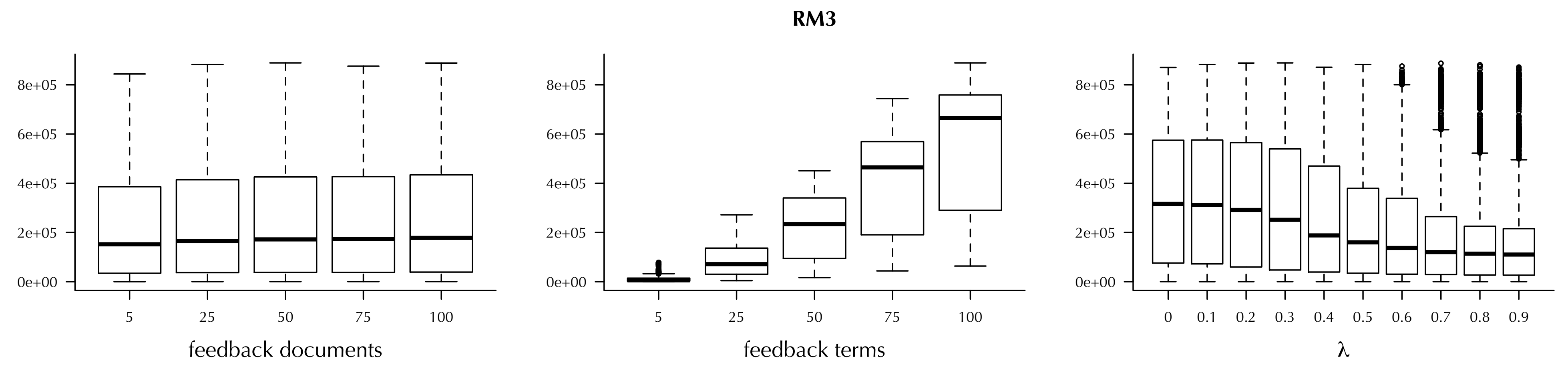
Top n locality of PRF. Probability of the highest ranked documents after PRF occurring in the top 1000 documents of the initial retrieval. Probabilities are computed from 1000 random parameter settings for an RM3 model.

Fernando Diaz. Condensed list relevance models. In Proceedings of the 2015 international conference on the theory of information retrieval, ICTIR '15, 313--316, New York, NY, USA, May 2015. , ACM.

Pseudo-Relevance feedback

Condensed List Relevance Models

- The biggest drawback of RM3 is the penalty of doing two retrievals.
- For a large search engine with a lot of users, this is 2x the cost!
- However, if we look at the positions of the documents for the first and second retrievals, we find high overlap…
- What if the second ranking is a *re*-ranking of the initial retrieval (CLRM3)?
    - effectiveness scores are almost identical!

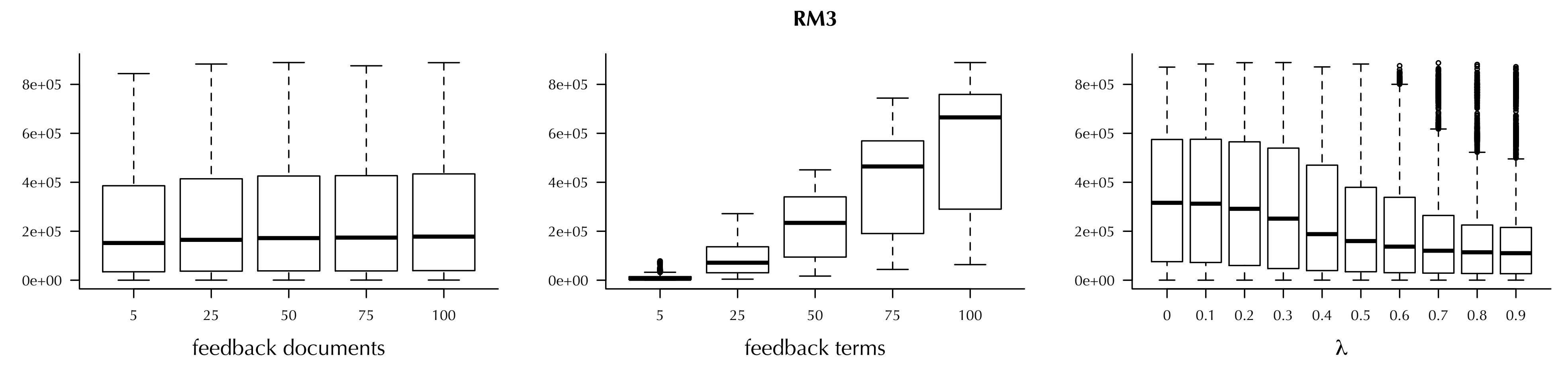


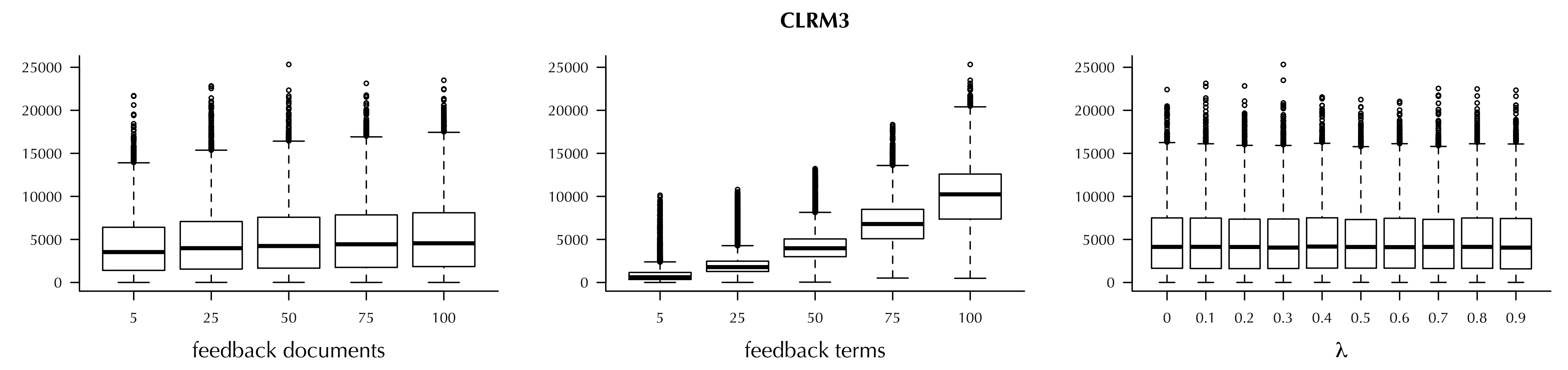
	RR	NDCG5	NDCG10
trec12			
RM3	0.7343	0.6047	0.5629
CLRM3	**0.7599**	0.6091	0.5661
robust			
RM3	0.7100	0.4810	0.4767
CLRM3	0.7100	0.4808	0.4757
cw09b			
RM3	0.4907	0.2161	0.2032
CLRM3	0.4907	0.2161	0.2032

Pseudo-Relevance feedback

Condensed List Relevance Models

- The biggest drawback of RM3 is the penalty of doing two retrievals.
- For a large search engine with a lot of users, this is 2x the cost!
- However, if we look at the positions of the documents for the first and second retrievals, we find high overlap…
- What if the second ranking is a *re*-ranking of the initial retrieval (CLRM3)?
  - effectiveness scores are almost identical!
  - efficiency gains are huge!

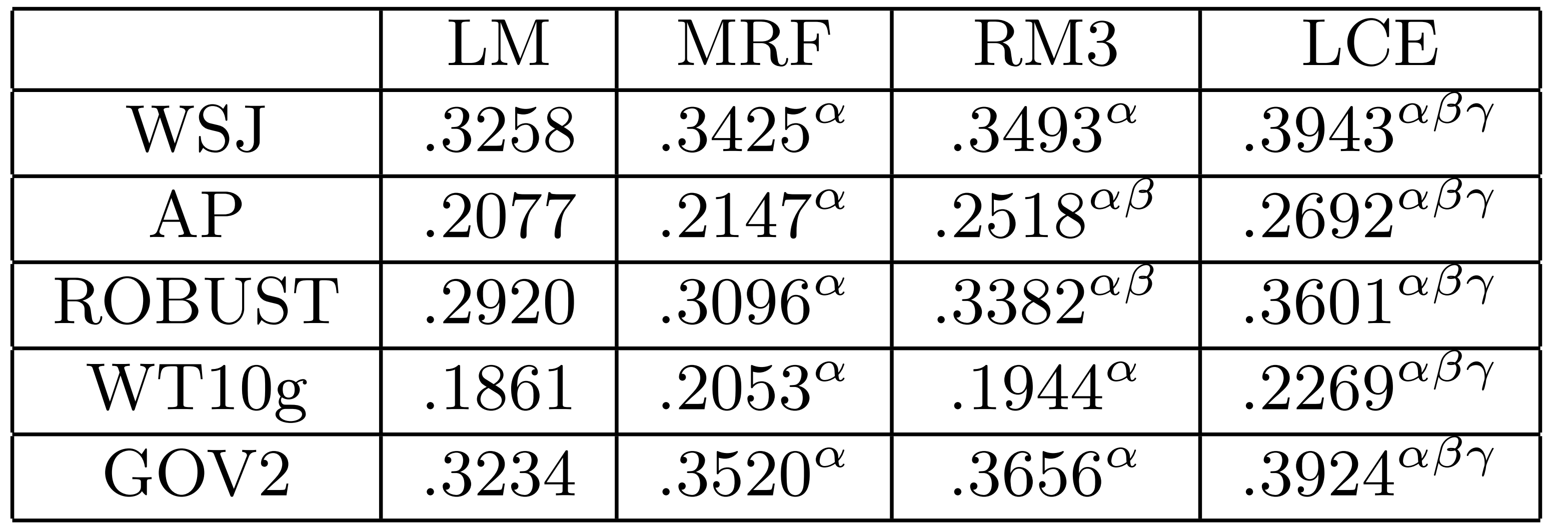




Pseudo-Relevance feedback

- PRF is a good trick to have in your pocket if you are ever in a situation where you need to work with tf.idf vectors

- PRF improves as the initial retrieval improves (e.g., can use a structured query language) [Metzler and Croft 2007]



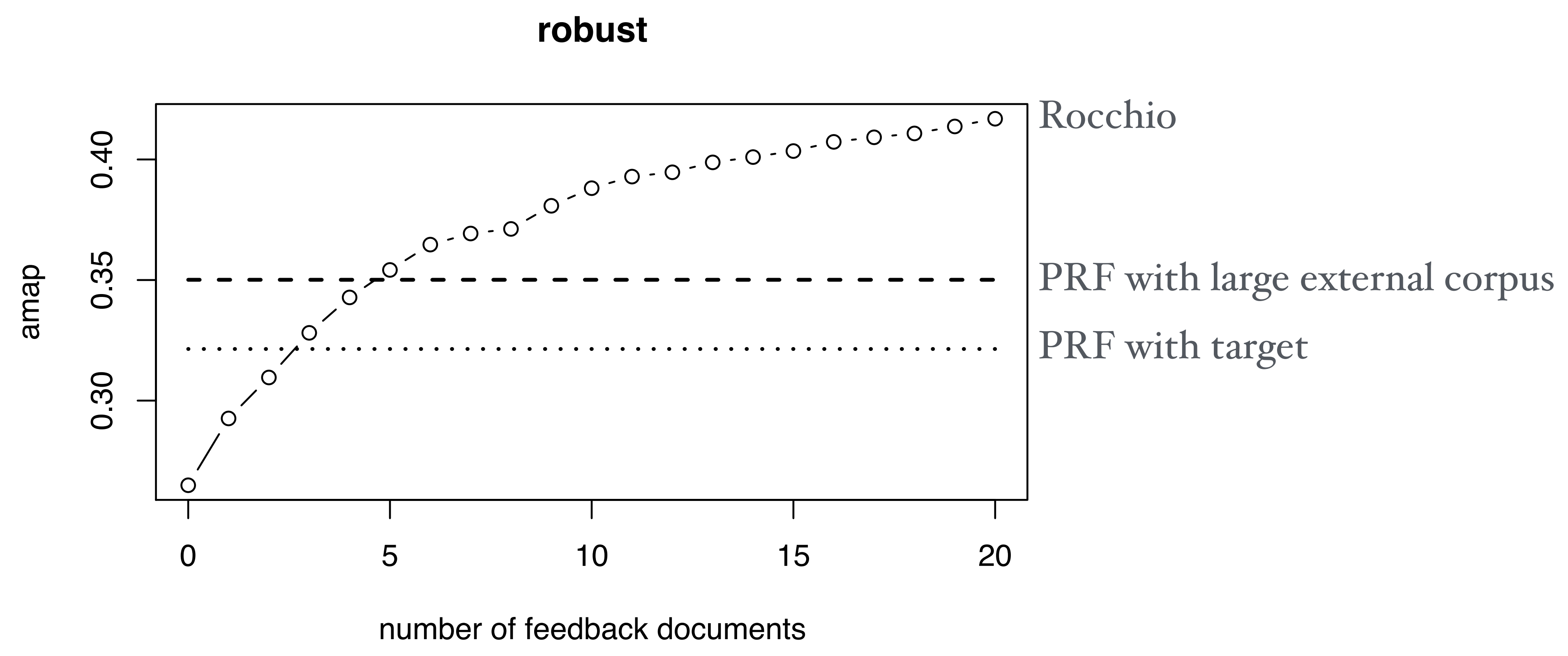
	LM	MRF	RM3	LCE
WSJ	.3258	$.3425^{\alpha}$	$.3493^{\alpha}$	$.3943^{\alpha\beta\gamma}$
AP	.2077	$.2147^{\alpha}$	$.2518^{\alpha\beta}$	$.2692^{\alpha\beta\gamma}$
ROBUST	.2920	$.3096^{\alpha}$	$.3382^{\alpha\beta}$	$.3601^{\alpha\beta\gamma}$
WT10g	.1861	$.2053^{\alpha}$	$.1944^{\alpha}$	$.2269^{\alpha\beta\gamma}$
GOV2	.3234	$.3520^{\alpha}$	$.3656^{\alpha}$	$.3924^{\alpha\beta\gamma}$

Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval, 2007.
Fernando Diaz and Donald Metzler. Improving the estimation of relevance models using large external corpora. In Sigir '06: proceedings of the 29th annual international acm sigir conference on research and development in information retrieval, 154--161, New York, NY, USA, 2006.

Pseudo-Relevance feedback

- PRF is a good trick to have in your pocket if you are ever in a situation where you need to work with tf.idf vectors
- PRF improves as the initial retrieval improves (e.g., can use a structured query language) [Metzler and Croft 2007]
- PRF improves if the initial retrieval is from a much larger external corpus [Diaz and Metzler 2006]
    - equal to getting ~5 labeled documents

	LM	MRF	RM3	LCE
WSJ	.3258	$.3425^{\alpha}$	$.3493^{\alpha}$	$.3943^{\alpha\beta\gamma}$
AP	.2077	$.2147^{\alpha}$	$.2518^{\alpha\beta}$	$.2692^{\alpha\beta\gamma}$
ROBUST	.2920	$.3096^{\alpha}$	$.3382^{\alpha\beta}$	$.3601^{\alpha\beta\gamma}$
WT10g	.1861	$.2053^{\alpha}$	$.1944^{\alpha}$	$.2269^{\alpha\beta\gamma}$
GOV2	.3234	$.3520^{\alpha}$	$.3656^{\alpha}$	$.3924^{\alpha\beta\gamma}$



Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval, 2007.
Fernando Diaz and Donald Metzler. Improving the estimation of relevance models using large external corpora. In Sigir '06: proceedings of the 29th annual international acm sigir conference on research and development in information retrieval, 154--161, New York, NY, USA, 2006.

Modern retrieval

Dense embeddings

- Documents and queries converted to dense vector representations
- Similarity measured via cosine distance or dot product in vector space
- Enables semantic matching beyond lexical matches
- Architecture & Workflow
    - Offline: encode document corpus and store in vector index (FAISS, Pinecone, etc.)
    - Online: encode query and approximate nearest neighbor search to return top-*k* results

Retrieval methods

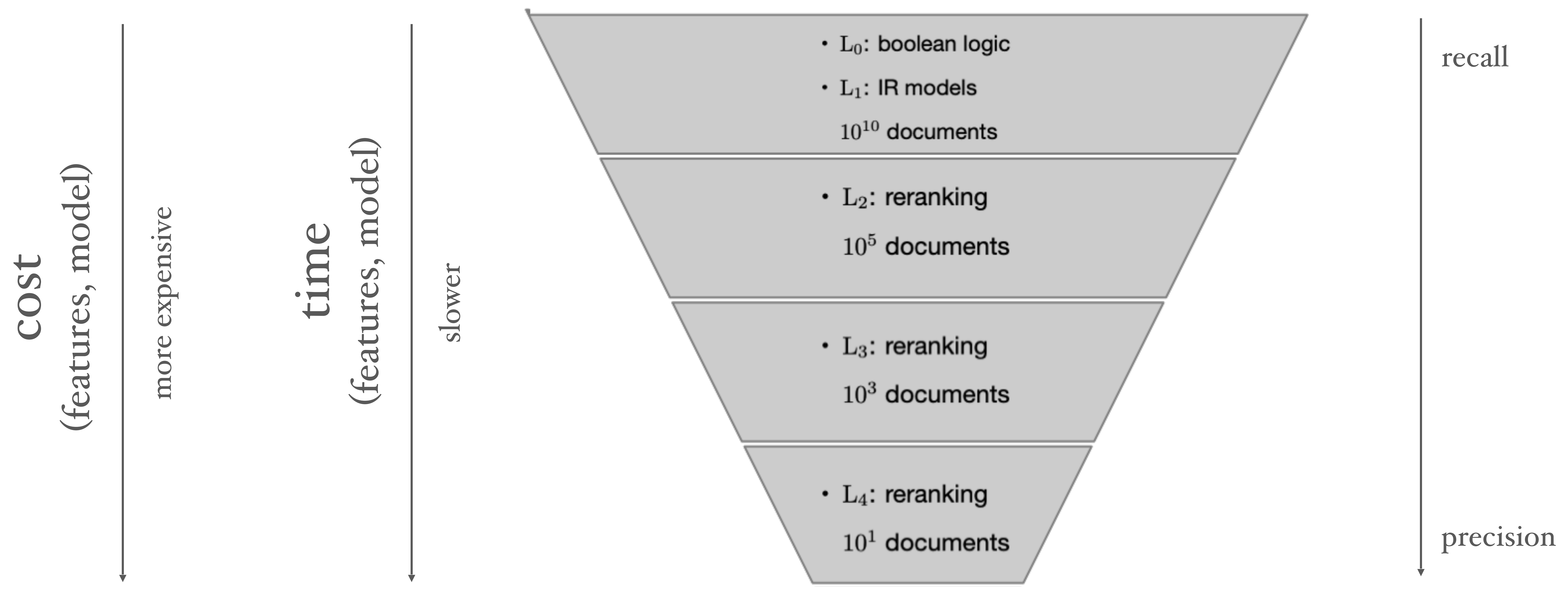
Comparison

Dimension	Sparse	Dense
Matching	Lexical	Semantic
Out-of-domain	Robust	Can degrade
Rare entities	Strong	Weak
Paraphrase	Weak	Strong
Training	None	Required

Multi-stage ranking

- typically retrieval is done by a sequence of retrieval models
  - **candidate generation stage(s):** simple and fast model to retrieve a smaller subset of the corpus that probably contains relevant documents
    - **often focuses on recall**
  - **ranking stage(s):** more accurate but slower model to score candidates
    - **often focuses on precision**
- example
  - **candidate generation:** top 100 documents by BM25
  - **ranking:** score 100 documents using a neural network

Multi-stage ranking



Learning to rank

Introduction

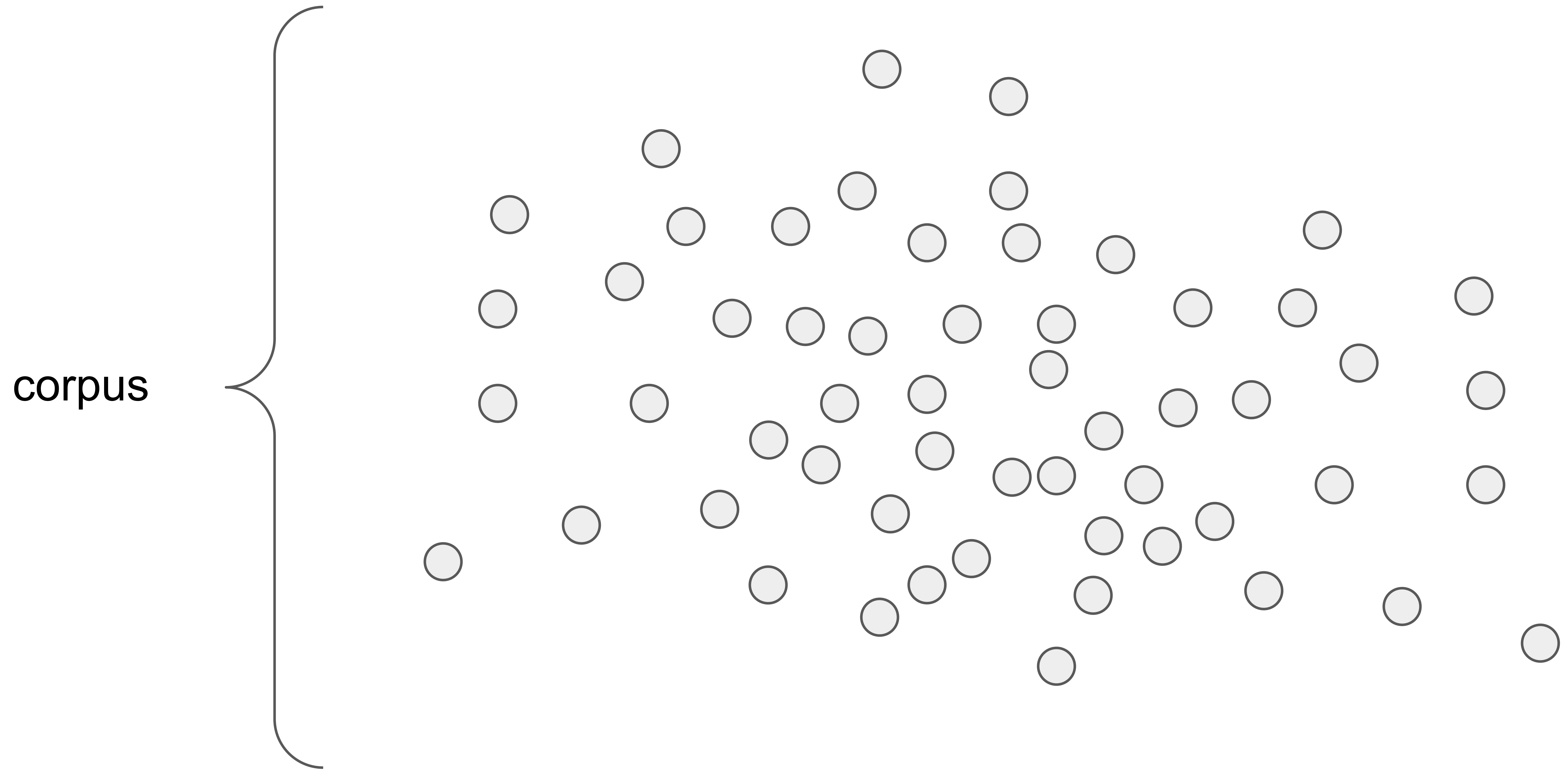
- Given a query, there are many signals correlated with relevance,
    - lexical signals: term matches, phrase matches, ...
    - document representations: title, body, URL, in-links, ...
    - static document information: quality, genre, ...
    - higher-level signals: BM25 score, embedding match, ...
- Different signals can be combined to improve accuracy
    - information is often complementary
- Manually exploring the space of combinations is impractical
    - too many combinations and too many parameters
    - can machine learning instead
    - referred to as learning to rank (LTR)

Learning to rank

Introduction

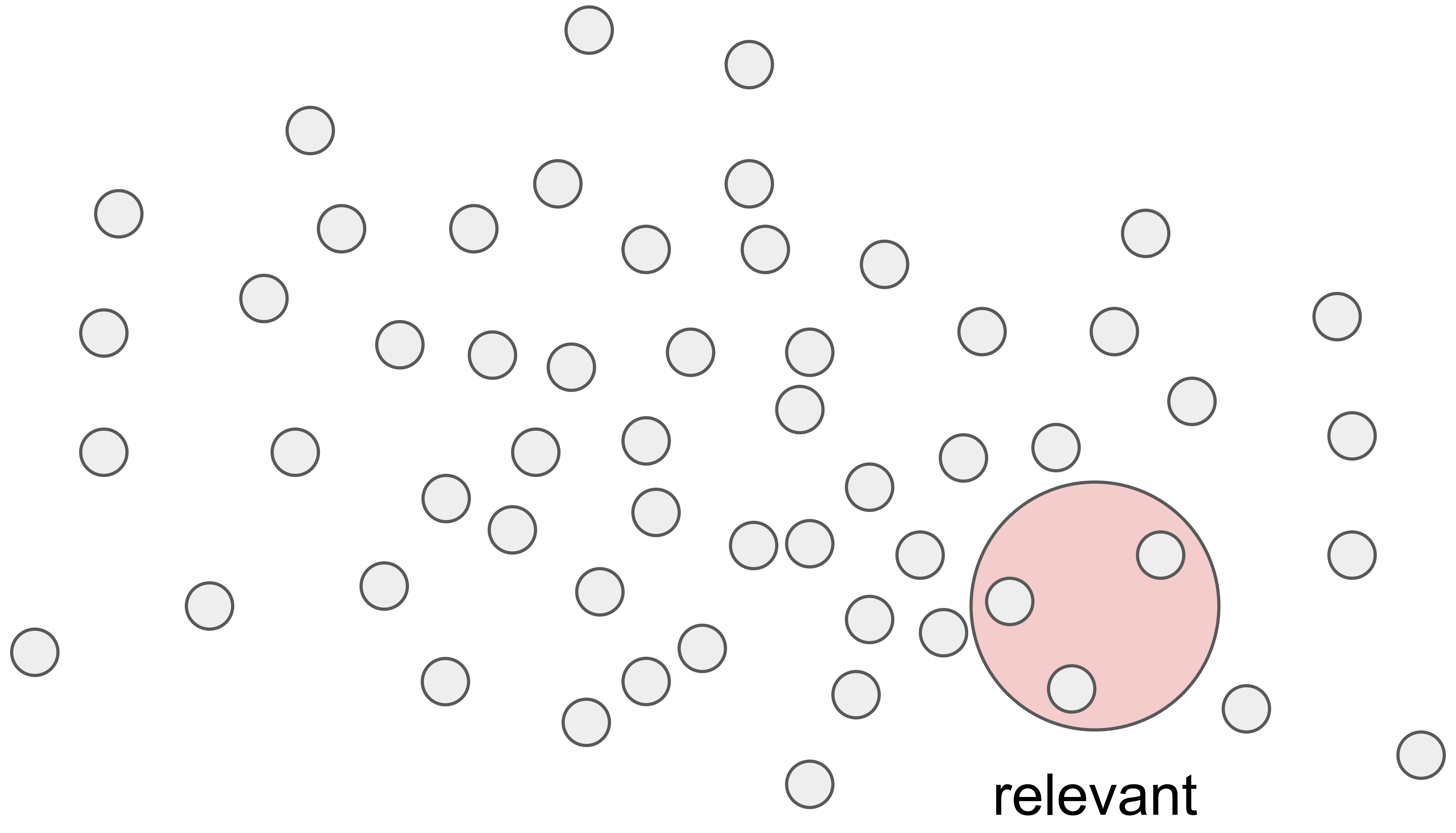
- learning to rank (LTR) refers to the broad set of techniques that optimizes a scorer given a specific ranking loss.
- scorers, also known as **ranking functions**, map a set of query-document signals, also known as **features**, to a scalar value.
- given a query, documents can be ranked using scores computed by a ranking function.
- the parameters of a ranking function are set to optimize performance on a **training set** of labeled query-document pairs.

Learning to rank



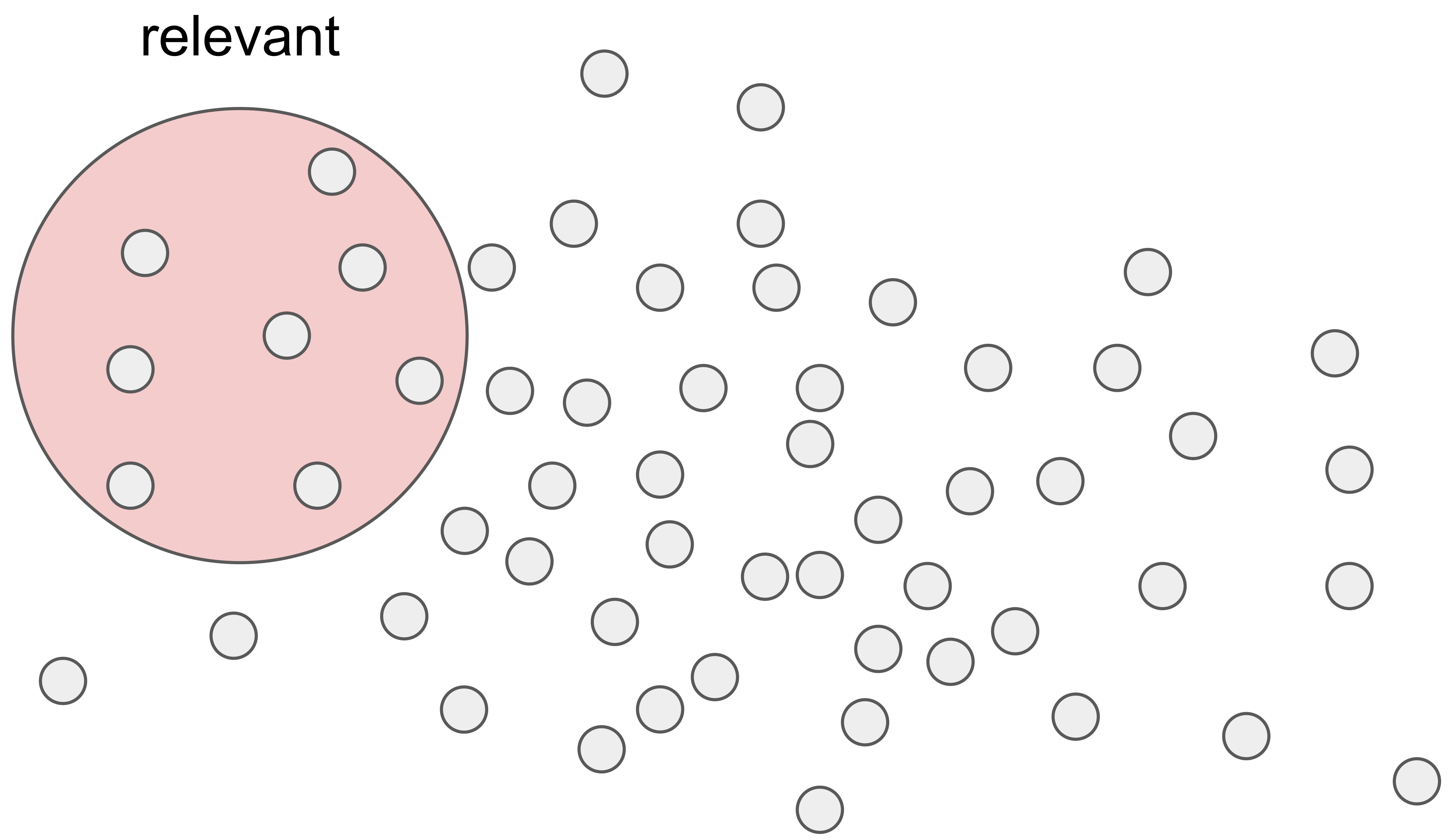
Learning to rank

query: cat



Learning to rank

query: dog



Learning to rank

Core components

- features
  - query
  - document
  - interaction
- optimization
  - pointwise
  - pairwise
  - listwise

Learning to rank

Features

ranking functions produce a document score by using signals from the query, the document, and their interaction.

- features vary in **usefulness,**
  - good features are predictive of document utility or relevance to the information need.
  - bad features are not predictive of document utility or relevance to the information need.
- features vary in **type,**
  - binary features (e.g., “is the term “cat” present in the document?”, “is any query term present in the document?”)
  - integer features (e.g., “count of term “cat” in the document”, “count of query terms in the document”)
  - scalar features (e.g., “BM25 score of the query and document”)
- features vary in **cost**,
  - computing some features are cheap (e.g., “count of query terms in the document”)
  - computing some features are expensive (e.g., “query-document score from a deep matching model”)

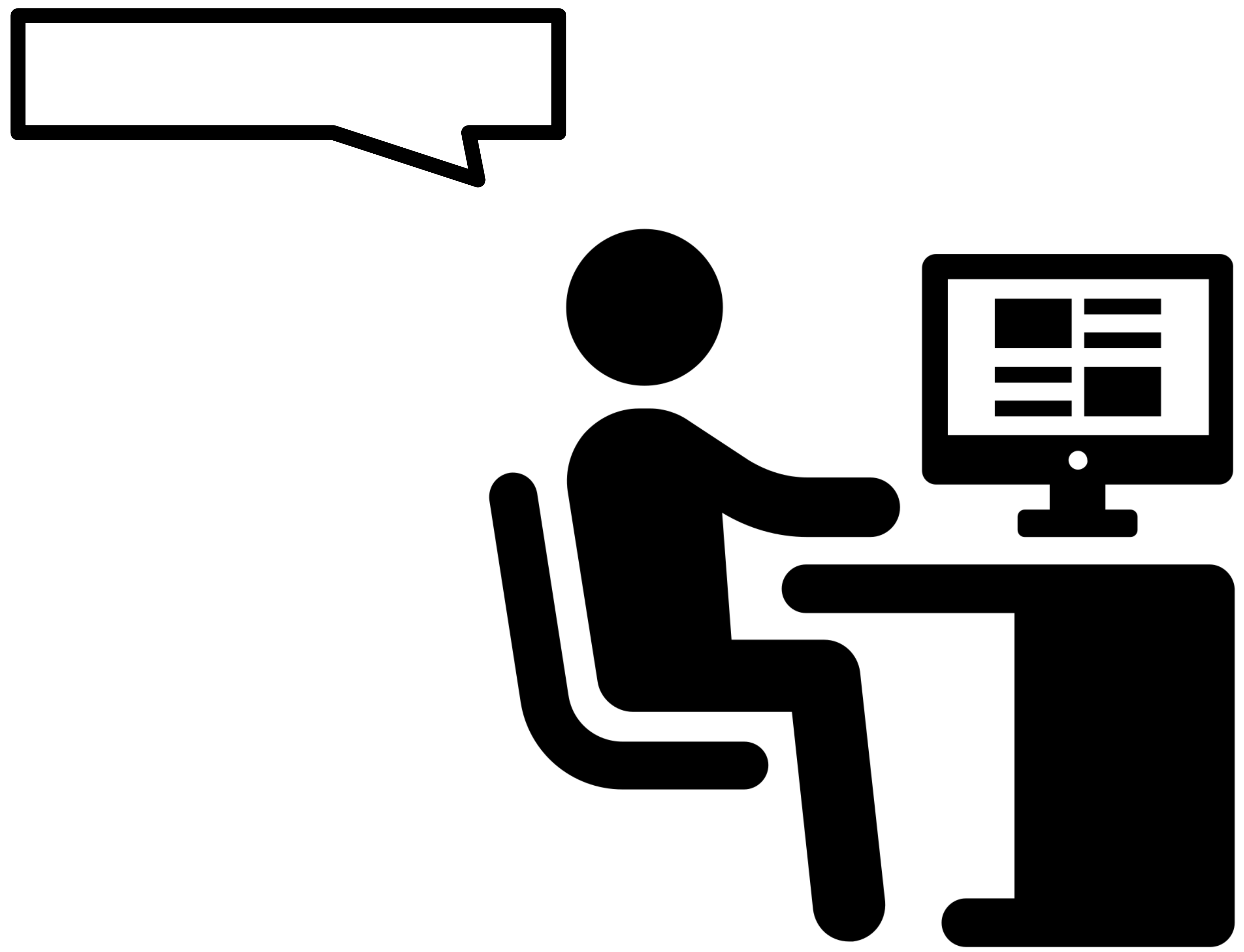
Learning to rank

Features

- features vary in **availability,**
    - computing some features can happen in advance of receiving a query (e.g., “document popularity”)
    - computing some features can only happen when the query is received (e.g.,“count of query terms in the document”)
    - computing some features can only happen after the user sees the results (e.g., “user clicked on the document”); *useful for predicting the relevance during evaluation*
- features vary in **generalizability,**
    - some features are associated with the current query only (e.g., “is the term “cat” present in the document?”)
    - some features are associated with a broader set of queries (e.g., “count of query terms in the document”)

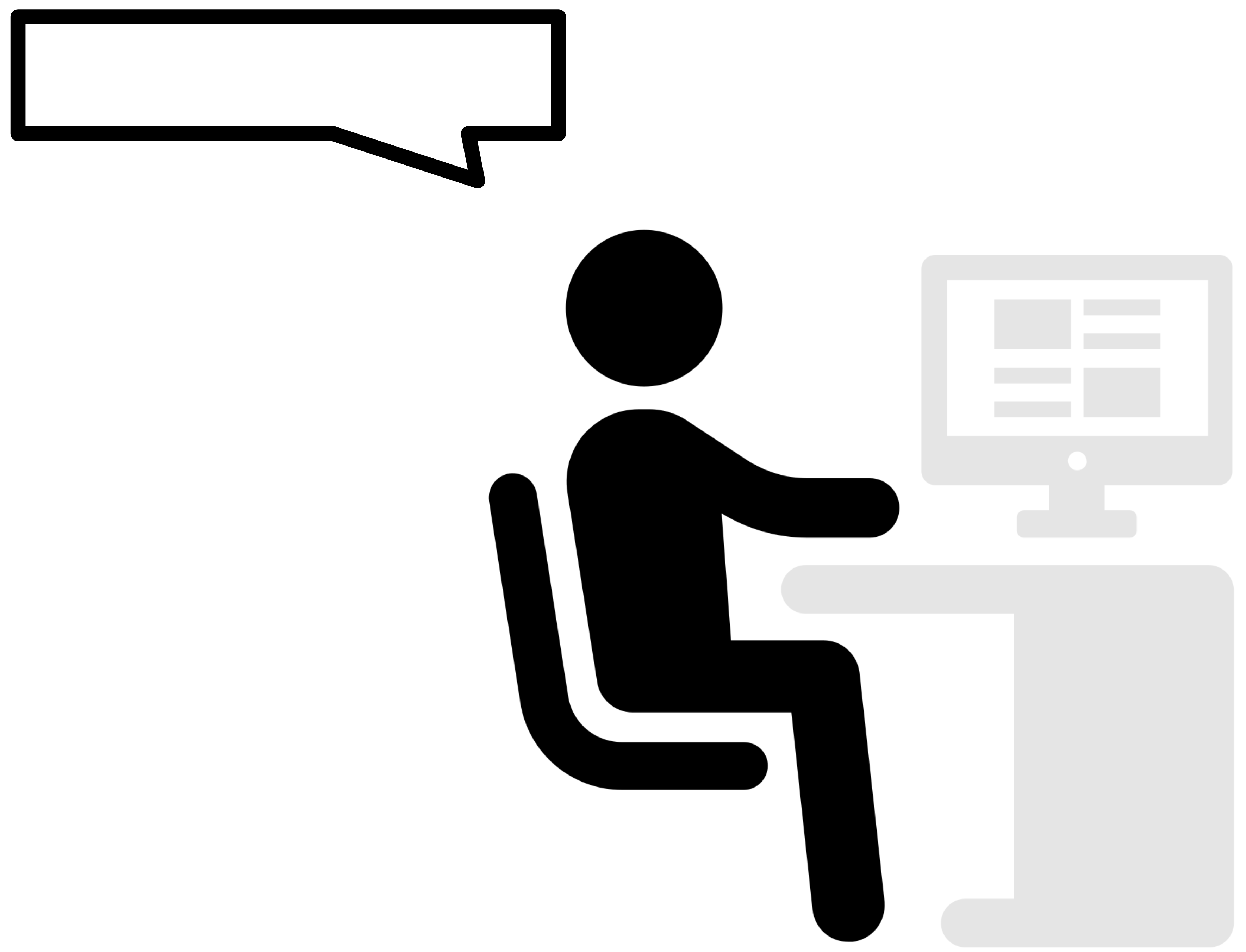
Learning to rank

Feature Types



Learning to rank

Query features



query length

misspellings

query difficulty metrics

user reading level

time of day

signals associated with the query or user, without regard for the document.

Learning to rank

Document features

document length

number of inlinks, number of referring domains, number of referring IP addresses

PageRank

spam score

url depth (pages closer to the root tend to be more relevant)

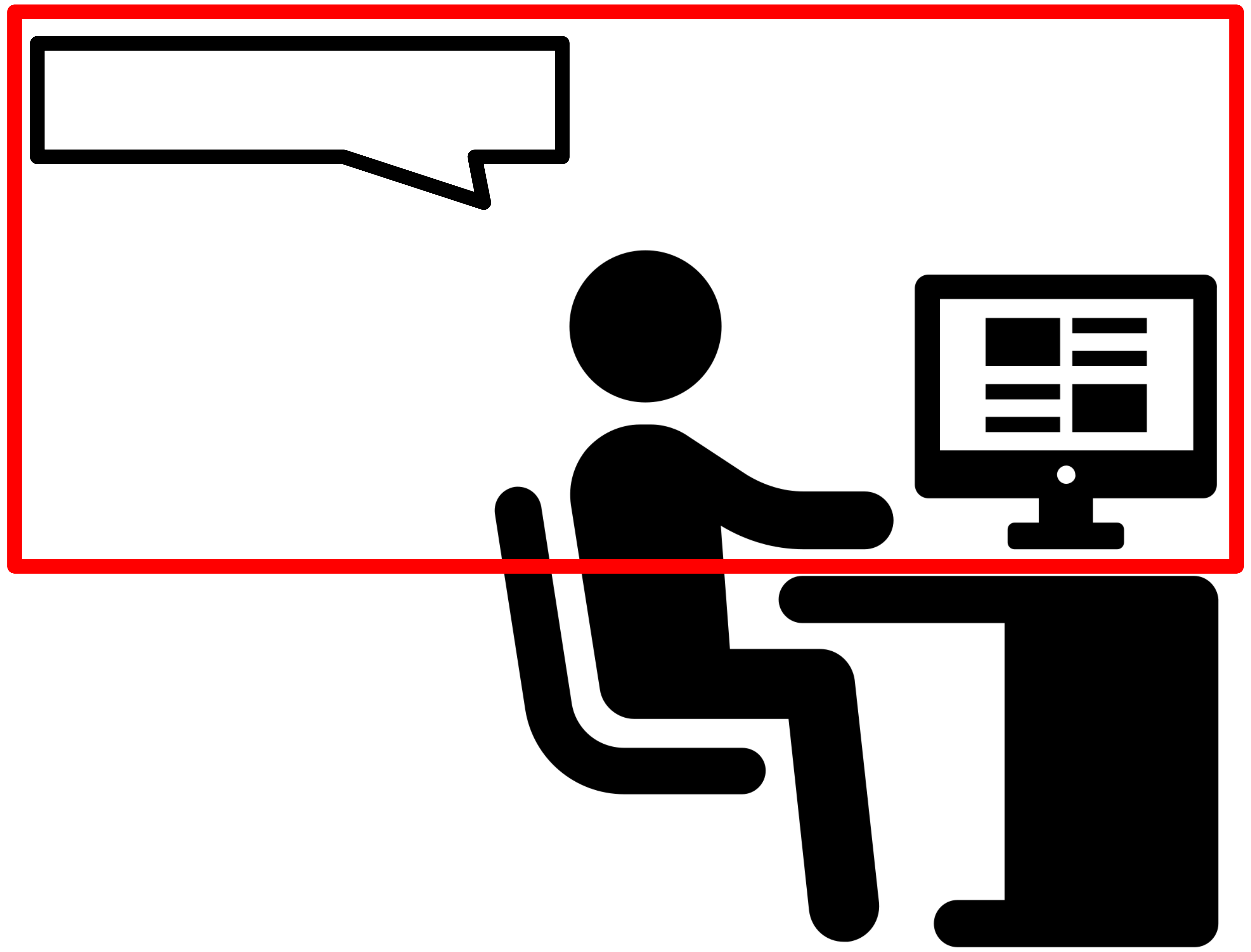
Avg time on site, avg pages / session, bounce rate (% of people who leave quickly)

Website security (https)

signals associated with the document, without regard for the query.

Learning to rank

Query-document features



Traditional retrieval algorithms (e.g., VSM, BM25, Indri, ...)

Embedding-based retrieval algorithms (e.g., BERT, ...)

The number (or percentage) of query terms that match d (coordinate match)

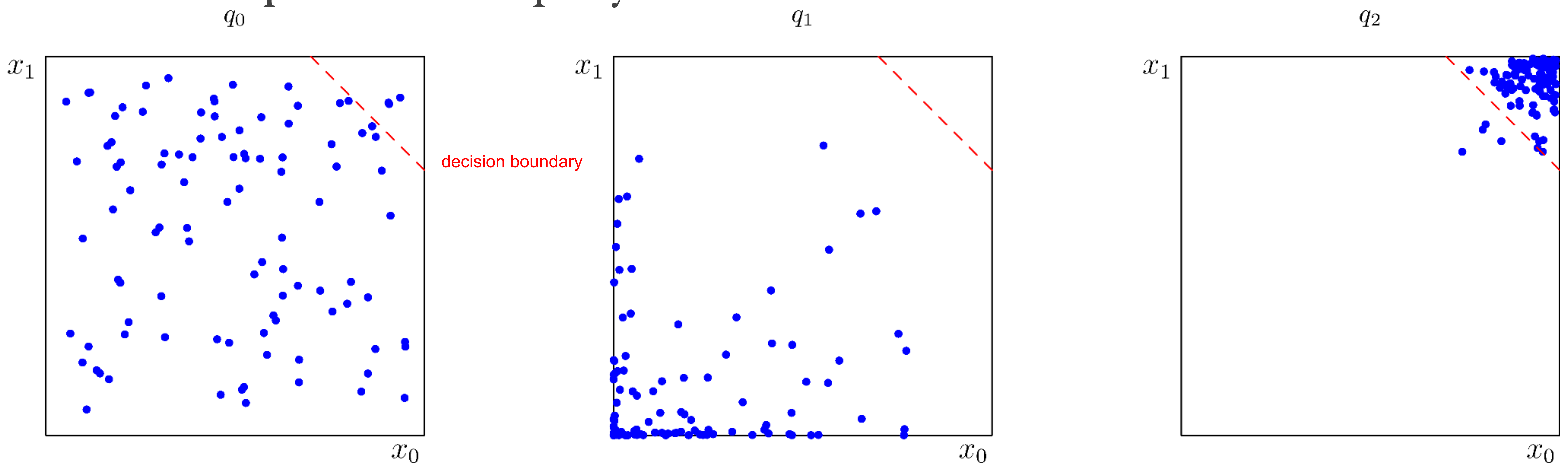
Matching against different parts of d (e.g., title, body, inlink, ...)

Matching with a more powerful query (e.g., SDM)

signals associated with query-document pair

Learning to rank

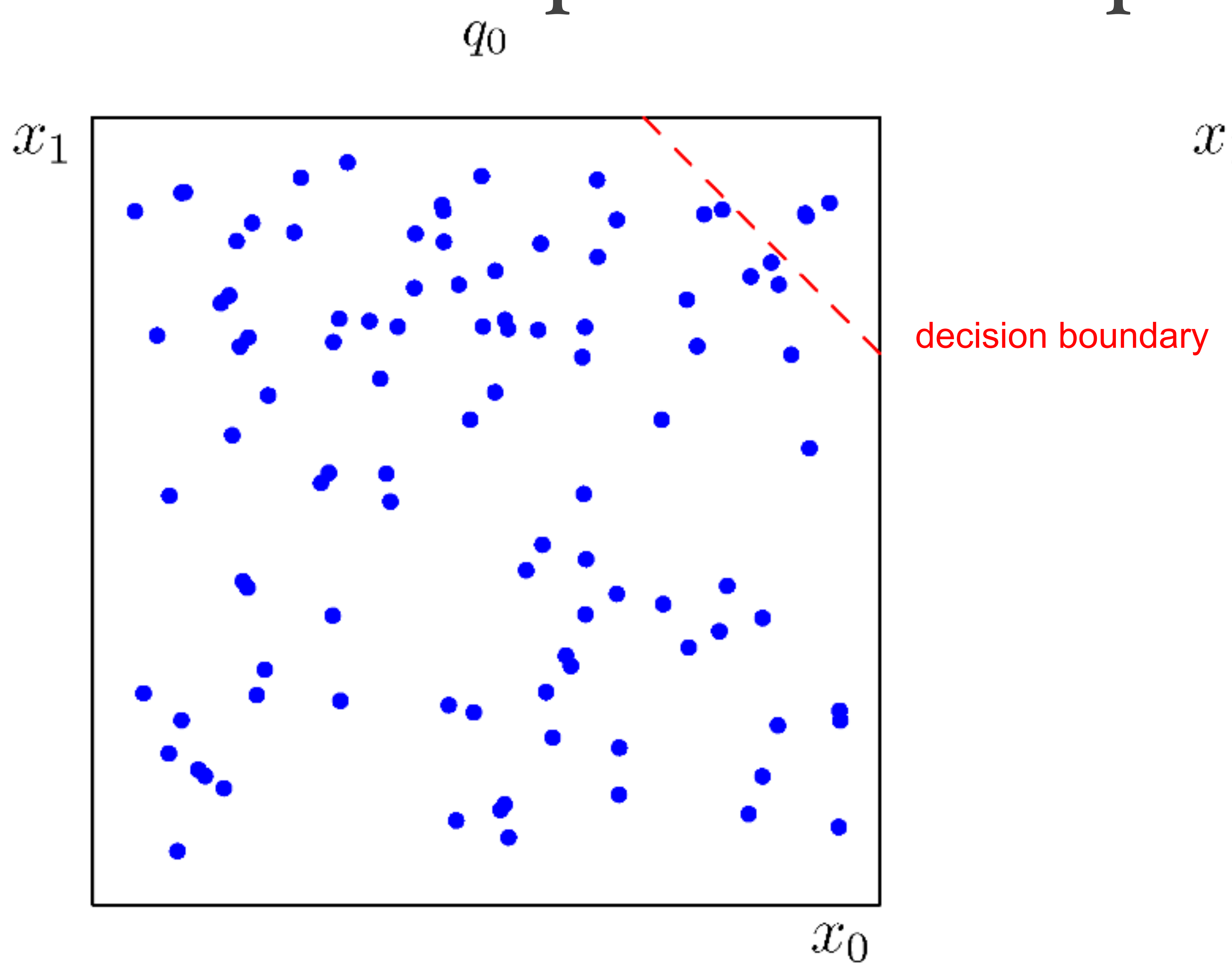
Features depend on the query

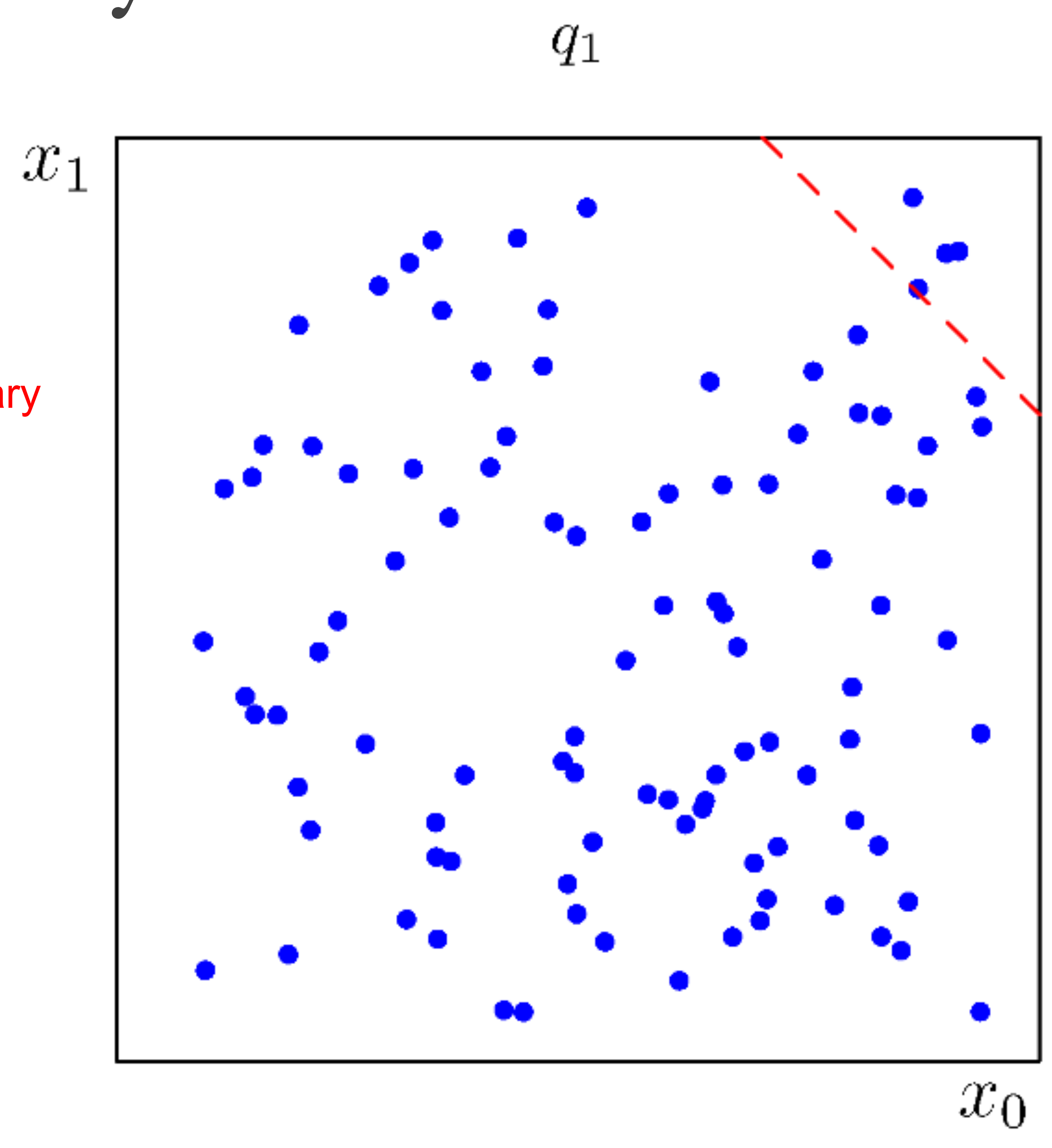


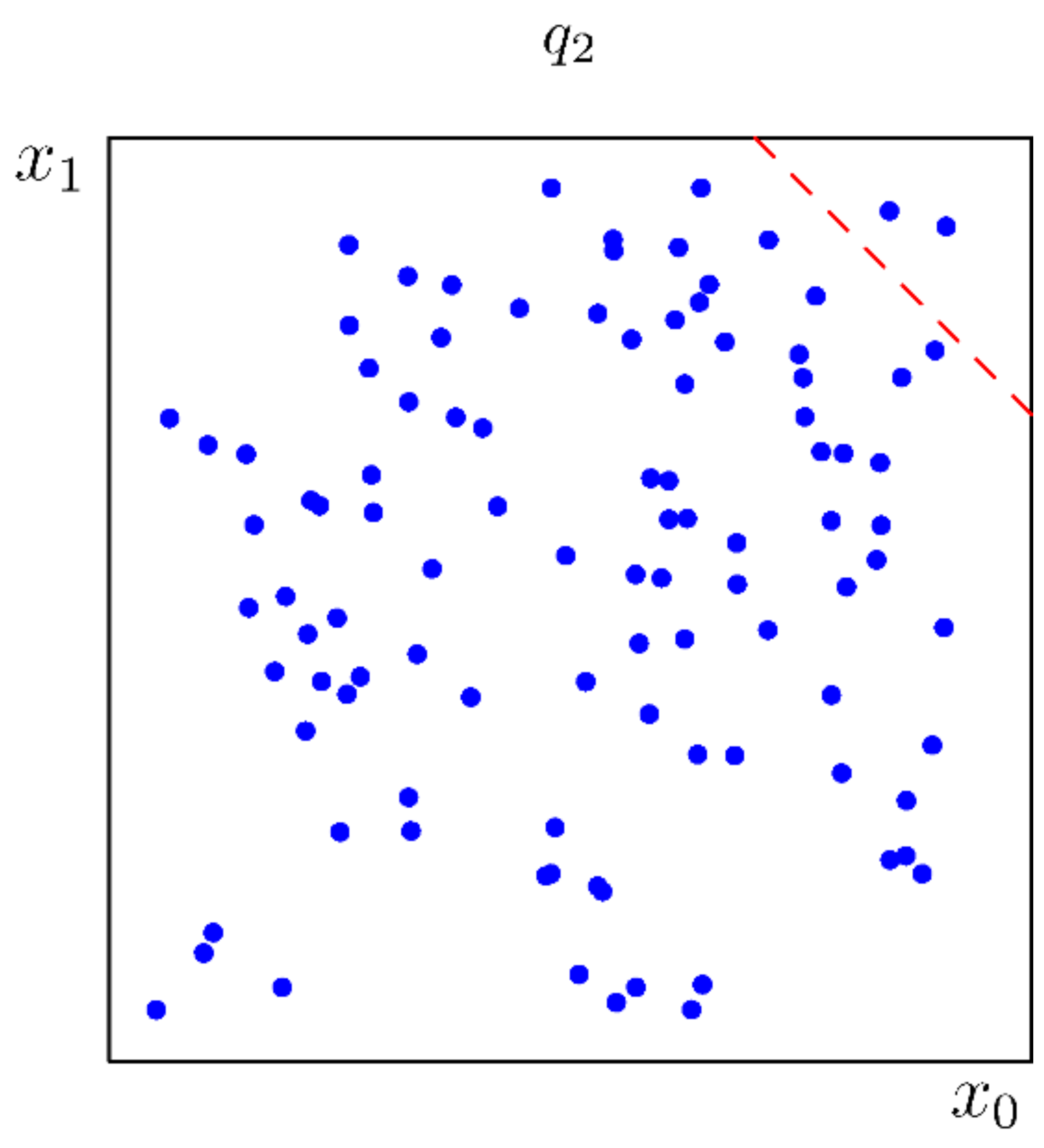
ranges of features will depend on the query → normalize features per-query to address

Learning to rank

Features depend on the query







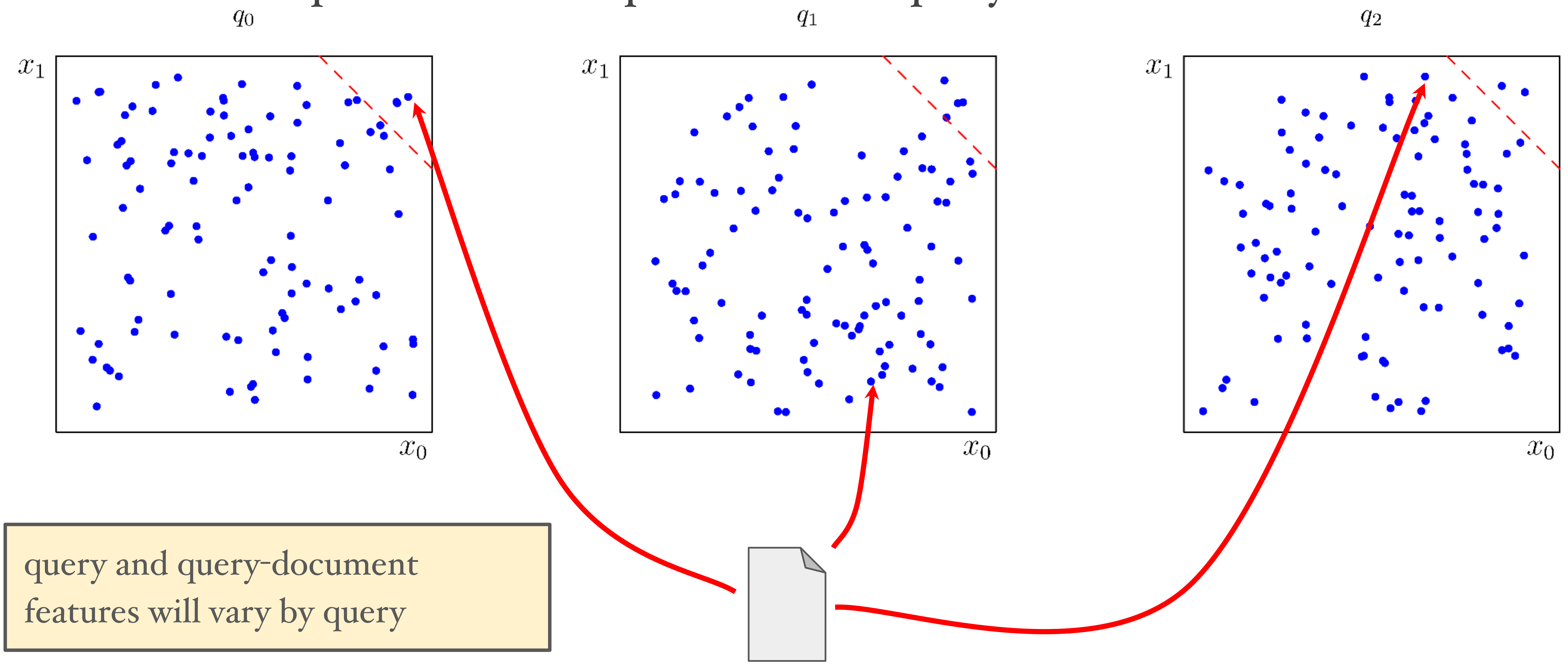
ranges of features will depend on the query

$\longrightarrow$

normalize features per-query to address

Learning to rank

Document representation depends on the query



query and query-document
features will vary by query

Learning to rank

Optimization

optimization refers to, given a set of labeled training data, and a parameterized function class, learn good weights with respect to a loss function.

- labeled training data: rater judgments on query-document pairs.
- parameterized function class: how feature values are combined into a prediction of relevance.
- loss function: how the quality of ranking function predictions of relevance are measured.

Learning to rank

Training data

- machine learning algorithms learn to predict labels or scores
    - typically, LTR systems predict scores
- The most common choices for a query-document pair

label	values	when used...
binary	{0,1}	navigational queries, exposure measurement
graded	{0, 1, 2, 3, 4}	web search, large sets of relevant documents
scalar	float	derived from model or behavioral data (e.g., clickthrough rate)

Training data

raw training data from human raters

$$\begin{gathered}
\langle q_0, d_0, y_{0,0} \rangle \\
\langle q_0, d_1, y_{0,1} \rangle \\
\langle q_0, d_2, y_{0,2} \rangle \\
\vdots \\
\langle q_i, d_0, y_{i,0} \rangle \\
\langle q_i, d_1, y_{i,1} \rangle \\
\langle q_i, d_2, y_{i,2} \rangle \\
\vdots \\
\langle q_n, d_0, y_{n,0} \rangle \\
\langle q_n, d_1, y_{n,1} \rangle \\
\langle q_n, d_2, y_{n,2} \rangle
\end{gathered}$$

Training data

apply feature generator to each query-document pair.

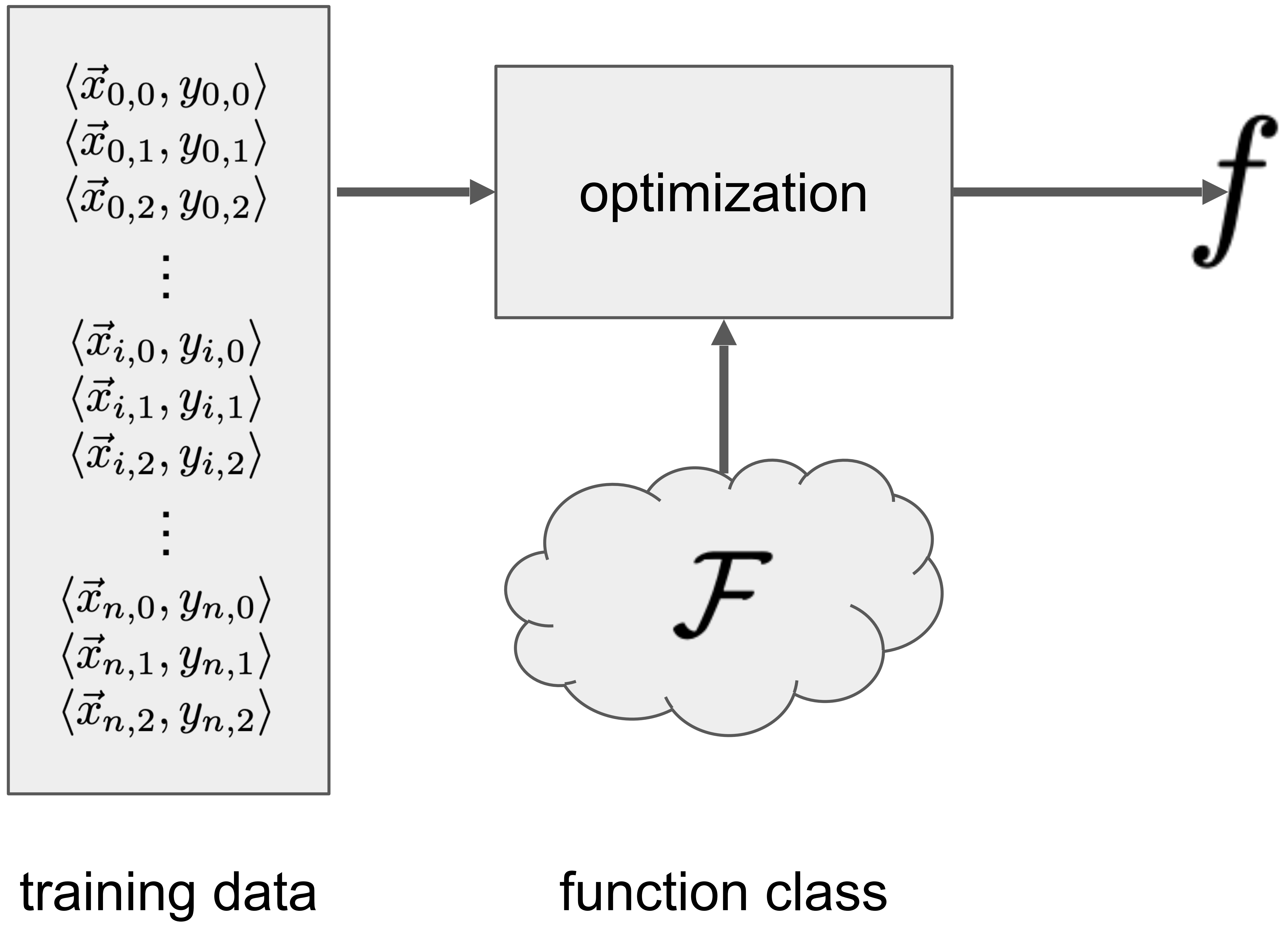
$$\begin{array}{cc}
\langle q_0, d_0, y_{0,0} \rangle & \rightarrow_\phi \\
\langle q_0, d_1, y_{0,1} \rangle & \rightarrow_\phi \\
\langle q_0, d_2, y_{0,2} \rangle & \rightarrow_\phi \\
\vdots & \vdots \\
\langle q_i, d_0, y_{i,0} \rangle & \rightarrow_\phi \\
\langle q_i, d_1, y_{i,1} \rangle & \rightarrow_\phi \\
\langle q_i, d_2, y_{i,2} \rangle & \rightarrow_\phi \\
\vdots & \vdots \\
\langle q_n, d_0, y_{n,0} \rangle & \rightarrow_\phi \\
\langle q_n, d_1, y_{n,1} \rangle & \rightarrow_\phi \\
\langle q_n, d_2, y_{n,2} \rangle & \rightarrow_\phi
\end{array}$$

Training data

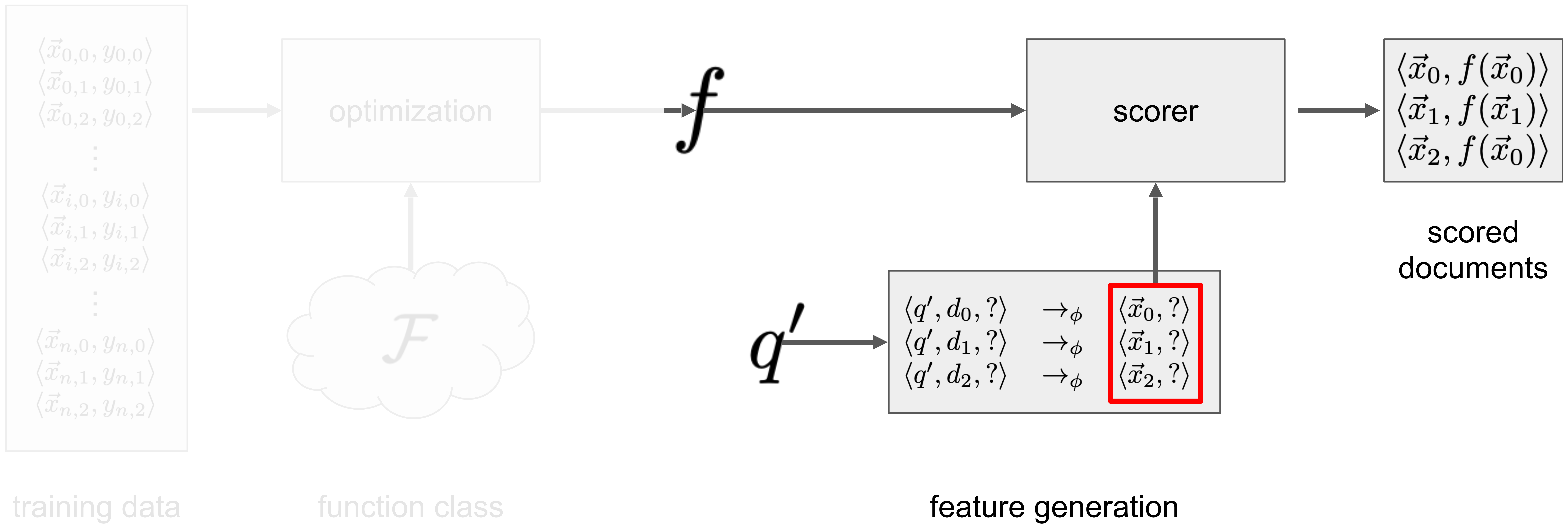
$$\begin{array}{ccc}
\langle q_0, d_0, y_{0,0} \rangle & \rightarrow_\phi & \langle \vec{x}_{0,0}, y_{0,0} \rangle \\
\langle q_0, d_1, y_{0,1} \rangle & \rightarrow_\phi & \langle \vec{x}_{0,1}, y_{0,1} \rangle \\
\langle q_0, d_2, y_{0,2} \rangle & \rightarrow_\phi & \langle \vec{x}_{0,2}, y_{0,2} \rangle \\
\vdots & \vdots & \vdots \\
\langle q_i, d_0, y_{i,0} \rangle & \rightarrow_\phi & \langle \vec{x}_{i,0}, y_{i,0} \rangle \\
\langle q_i, d_1, y_{i,1} \rangle & \rightarrow_\phi & \langle \vec{x}_{i,1}, y_{i,1} \rangle \\
\langle q_i, d_2, y_{i,2} \rangle & \rightarrow_\phi & \langle \vec{x}_{i,2}, y_{i,2} \rangle \\
\vdots & \vdots & \vdots \\
\langle q_n, d_0, y_{n,0} \rangle & \rightarrow_\phi & \langle \vec{x}_{n,0}, y_{n,0} \rangle \\
\langle q_n, d_1, y_{n,1} \rangle & \rightarrow_\phi & \langle \vec{x}_{n,1}, y_{n,1} \rangle \\
\langle q_n, d_2, y_{n,2} \rangle & \rightarrow_\phi & \langle \vec{x}_{n,2}, y_{n,2} \rangle
\end{array}$$

final training data for learning to rank model

Training (offline)


$\langle \vec{x}_{0,0}, y_{0,0} \rangle$
$\langle \vec{x}_{0,1}, y_{0,1} \rangle$
$\langle \vec{x}_{0,2}, y_{0,2} \rangle$
$\vdots$
$\langle \vec{x}_{i,0}, y_{i,0} \rangle$
$\langle \vec{x}_{i,1}, y_{i,1} \rangle$
$\langle \vec{x}_{i,2}, y_{i,2} \rangle$
$\vdots$
$\langle \vec{x}_{n,0}, y_{n,0} \rangle$
$\langle \vec{x}_{n,1}, y_{n,1} \rangle$
$\langle \vec{x}_{n,2}, y_{n,2} \rangle$
optimization
f
$\mathcal{F}$
training data
function class

Inference (online)



$\langle \vec{x}_{0,0}, y_{0,0} \rangle$
$\langle \vec{x}_{0,1}, y_{0,1} \rangle$
$\langle \vec{x}_{0,2}, y_{0,2} \rangle$
$\vdots$
$\langle \vec{x}_{i,0}, y_{i,0} \rangle$
$\langle \vec{x}_{i,1}, y_{i,1} \rangle$
$\langle \vec{x}_{i,2}, y_{i,2} \rangle$
$\vdots$
$\langle \vec{x}_{n,0}, y_{n,0} \rangle$
$\langle \vec{x}_{n,1}, y_{n,1} \rangle$
$\langle \vec{x}_{n,2}, y_{n,2} \rangle$
training data
optimization
$\mathcal{F}$
function class
f
scorer
$\langle \vec{x}_0, f(\vec{x}_0) \rangle$
$\langle \vec{x}_1, f(\vec{x}_1) \rangle$
$\langle \vec{x}_2, f(\vec{x}_0) \rangle$
scored documents
q'
$\langle q', d_0, ? \rangle \rightarrow_\phi \langle \vec{x}_0, ? \rangle$
$\langle q', d_1, ? \rangle \rightarrow_\phi \langle \vec{x}_1, ? \rangle$
$\langle q', d_2, ? \rangle \rightarrow_\phi \langle \vec{x}_2, ? \rangle$
feature generation