IR Topics

PRI 22/23 · Information Processing and Retrieval
M.EIC · Master in Informatics Engineering and Computation

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Outline

- ➔ Learning to Rank
- ➔ Neural Information Retrieval
- ➔ Vector Search

Learning to Rank

Based on Learning to Rank for Information Retrieval, Tie-Yan Liu (2009)

Learning to Rank

- ➔ *Learning to Rank (LtoR) (Liu, 2009)*
  - ➔ *is a task to automatically construct a ranking model*
  - ➔ *using training data, such that*
  - ➔ *the model can sort new objects according*
  - ➔ *to their degrees of relevance, preference, or importance.*

- ➔ Motivated by the large volume of information available
  - ➔ Challenge in ranking millions of documents
  - ➔ Opportunity to use existing data for ranking

Ranking Problems

- Many information retrieval problems are by nature ranking problems, such as:
  - document retrieval,
  - collaborative filtering,
  - key term extraction,
  - definition finding.
- Many different ranking scenarios:
  - Rank document purely according to their relevance with regards to the query.
  - Considering the relationships of similarity and diversity between documents (e.g. relational ranking).
  - Aggregate several candidate ranked lists (e.g. meta search, unified ranking).
  - Find to what degree a document property influences the ranking result.

Ranking in Information Retrieval

- ➔ Conventional document ranking models:
  - ➔ Query-dependent models:
    - ➔ Boolean model, set based model (no degree of relevance)
    - ➔ Vector space model, using term weighting such as TF-IDF.
    - ➔ Probabilistic models, such as language models and BM25.
  - ➔ Query-independent models:
    - ➔ Link-based models for the web, such as PageRank or HITS

Learning to Rank

- ➔ Many of the existing ranking models contain parameters that need to be tuned.
- ➔ Different ranking models can be combined to create a new ranking.
- ➔ In addition, models perfectly tuned using the validation set do not necessary perform well on unseen queries.

- ➔ How to define and combine parameters and models to produce an effective ranking?
- ➔ Machine learning methods have demonstrated their effectiveness in automatically tuning parameters combining multiple evidences.
- ➔ Learning to rank is the application of machine learning methods to information retrieval problems.

Typical Learning to Rank Flow

➔ The training data consists of:

- ➔ n queries (q1, q2, …),
- ➔ their associated documents represented as features vectors (x1, x2, …),
- ➔ and the corresponding relevance judgements.

➔ A learning algorithm is employed to learn the ranking model h.

➔ The ranking model h is then evaluated with standard IR measures by producing ranked lists for the training data.

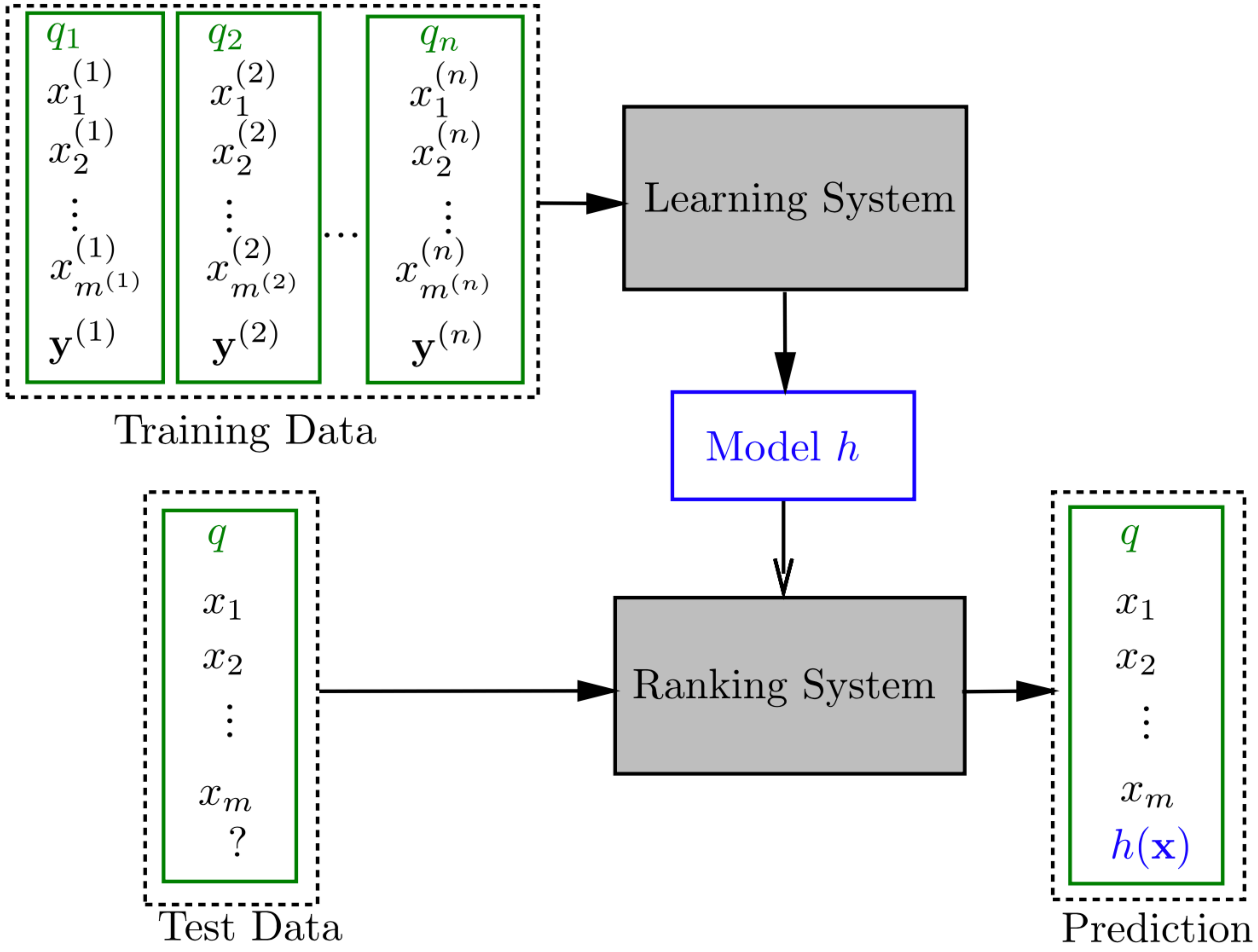


Fig. 1.1 Learning-to-rank framework.

Image from Learning to Rank for Information Retrieval, Tie-Yan Liu (2009)

Popular Features

Table 6.2 Learning features of TREC.

ID	Feature description
1	Term frequency (TF) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title

Table 6.2 (*Continued*)

ID	Feature description
40	LMIR.JM of whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink base score propagation: weighted in-link
44	Hyperlink base score propagation: weighted out-link
45	Hyperlink base score propagation: uniform out-link
46	Hyperlink base feature propagation: weighted in-link
47	Hyperlink base feature propagation: weighted out-link
48	Hyperlink base feature propagation: uniform out-link
49	HITS authority
50	HITS hub
51	PageRank
52	HostRank
53	Topical PageRank
54	Topical HITS authority
55	Topical HITS hub
56	Inlink number
57	Outlink number
58	Number of slash in URL
59	Length of URL
60	Number of child page
61	BM25 of extracted title
62	LMIR.ABS of extracted title
63	LMIR.DIR of extracted title
64	LMIR.JM of extracted title

Table from Learning to Rank for Information Retrieval, Tie-Yan Liu (2009)

Learning to Rank Approaches

- ➔ The learning to rank process can be modeled in different ways.
  - ➔ Pointwise approach
  - ➔ Pairwise approach
  - ➔ Listwise approach

- ➔ Different learning techniques can be employed, including SVM, Boosting, Neural Nets, etc.

- ➔ Effectiveness from literature: Listwise > Pairwise > Pointwise
- ➔ Different methods can be combined with successful results.

Pointwise Approach

- ➔ Input: feature vectors for single documents, e.g. scores for query-document pairs.
- ➔ Output: relevance degree of each single document (e.g., relevant / non-relevant).
- ➔ Methods: OC SVM, McRank, Prank.

- ➔ Treat the problem as a regression problem, i.e. estimate a continuous variable (i.e. each document's score).
- ➔ Straightforward approach.
- ➔ Ambitious goal — produce document scores, but only rankings are needed.

Pairwise Approach

- ➔ Pairwise classification problem, i.e. decide pairwise preferences for documents.
- ➔ Input: pairs of documents, both represented as feature vectors.
  - ➔ Manually annotated data, e.g. (q,d,d') — d is more relevant to q than d' (harder to get, high quality).
  - ➔ Log data, e.g. if, for query q, document d and d' are listed, and user clicked d and not d', then (q,d,d') can be inferred (easy, lower quality).
- ➔ Output: pairwise preferences (ranging from 1 to -1) between document pairs.

- ➔ The problem is treated as a classification problem, i.e. given a query and two documents, determine which is better.
- ➔ Methods: Ranking SVM, LambdaMART, LambdaRank

Listwise Approach

- ➔ Input: entire group of documents associated with a query, i.e. ranking lists.
  - ➔ Offline: obtain relevance judgements (q,d,s), where s is a score indicating the relevance of document d to query q.
  - ➔ Online: present different rankings to users (or interleave lists) and observe (from logs) which users prefer.
- ➔ Output: full document predicted ranking for query.
- ➔ Methods: PermuRank, AdaRank, SoftRank.

- ➔ Problem is modeled in a more natural "IR" way but not directly adaptable to conventional machine learning techniques.
- ➔ Best performing approaches.

Reference Datasets for Learning to Rank

- ➔ LETOR: Learning to Rank for Information Retrieval
  - ➔ www.microsoft.com/research/project/letor-learning-rank-information-retrieval
- ➔ Microsoft Learning to Rank Datasets
  - ➔ www.microsoft.com/research/project/mslr

Support in Solr

- ➔ Solr includes support for Learning to Rank.
- ➔ https://solr.apache.org/guide/learning-to-rank.html
  - ➔ Includes a step-by-step practical example.

Neural Information Retrieval

Based on An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).
Based on Lecture Notes on Neural Information Retrieval, N. Tonellotto (2022).

Neural Information Retrieval

➔ Neural IR is the application of shallow or deep neural networks to IR tasks.

  ➔ Neural models have been employed in many IR scenarios—including ad-hoc retrieval, recommender systems, multimedia search, and even conversational systems that generate answers in response to natural language questions.

➔ Unlike Learning to Rank approaches that train machine learning models over a set of hand-crafted features, neural models accept the raw text of a query and document as input.

➔ Neural networks are typically used in two different ways:

  ➔ Learn the ranking functions combining the relevance signals to produce an ordering of documents.

  ➔ Learn the abstract representations of documents and queries to capture their relevance information.

Data in Neural Information Retrieval

- ➔ Neural models for IR use vector representations of text, which usually contain a large number of parameters that need to be tuned.
- ➔ ML models with large set of parameters typically benefit from large quantity of training data.
- ➔ Learning suitable representations of text demands large-scale datasets for training.
- ➔ Therefore, unlike classical IR models, these neural approaches tend to be data hungry, with performance that improves with more training data.

Document Ranking

- Document ranking comprises three steps (recall IR models):
  - Generate a representation of the query that specified the information need.
  - Generate a representation of the document.
  - Match the query and the document representations to estimate their mutual relevance.
- Neural approaches can influence one or more of these three steps.
- Unlike traditional learning to rank models, neural architectures depend less on manual feature engineering on more on automatically detecting regularities in matching patterns.

Document Ranking

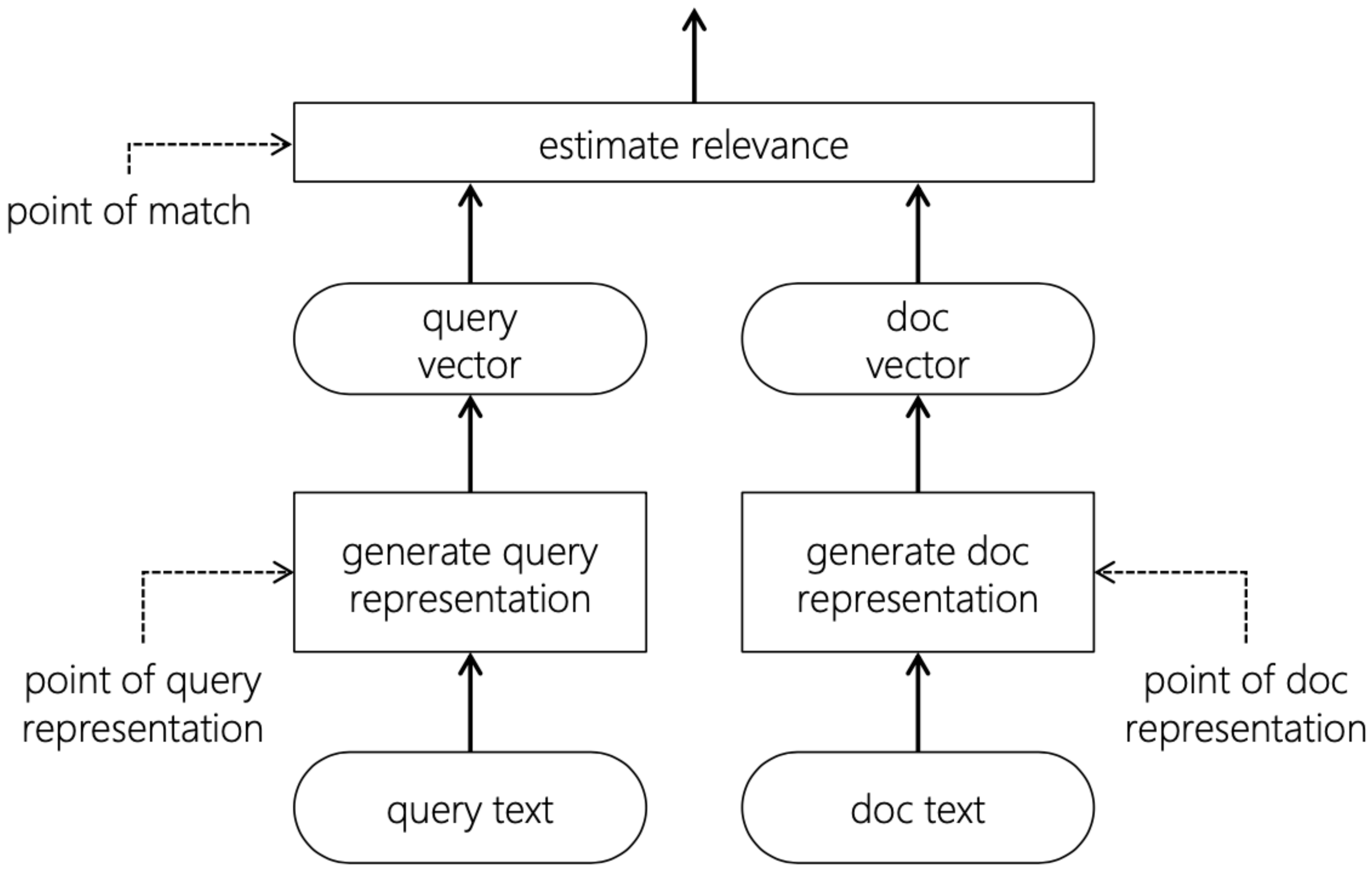


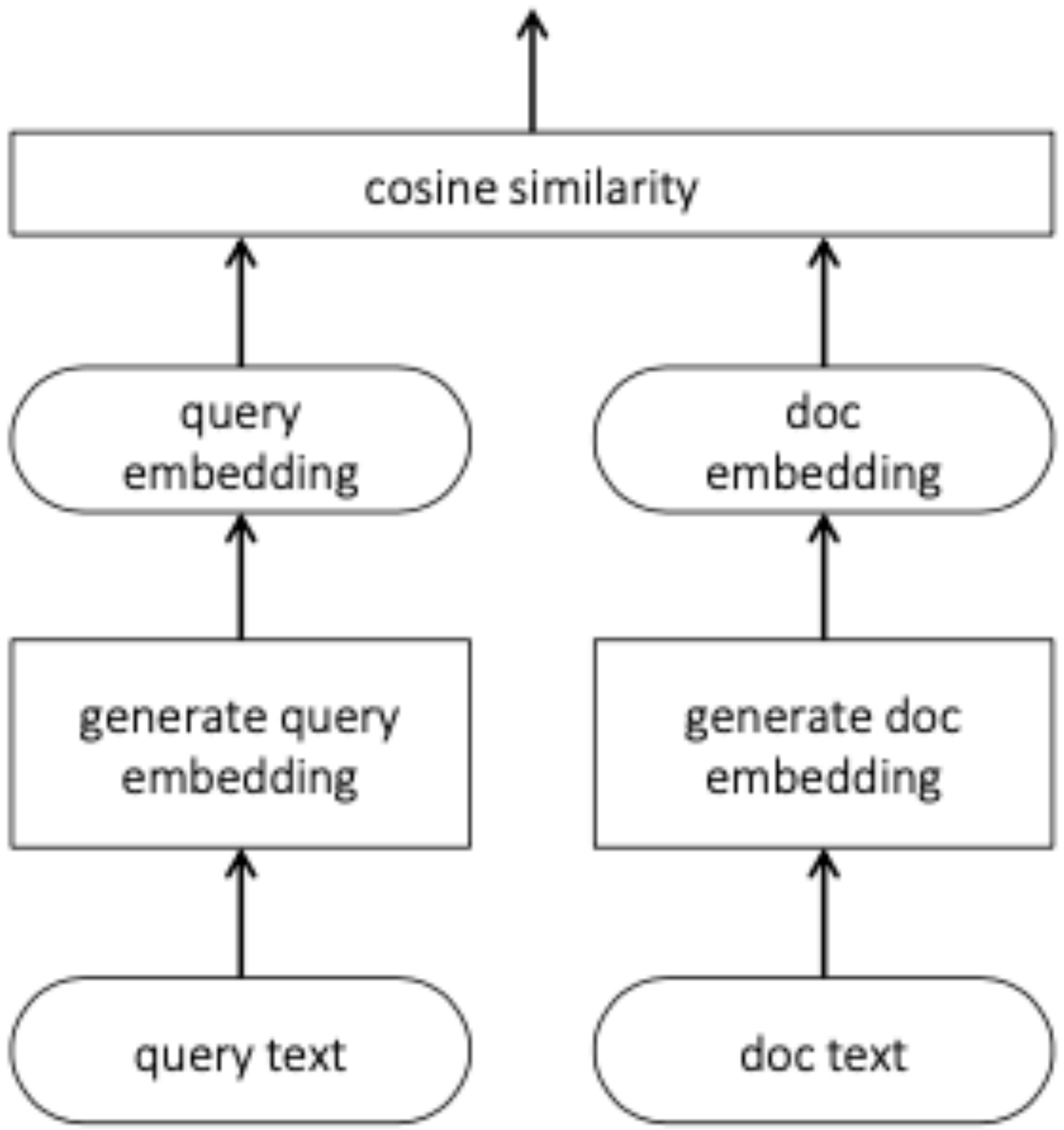
Figure 2.3: Document ranking typically involves a query and a document representation steps, followed by a matching stage. Neural models can be useful either for generating good representations or in estimating relevance, or both.

Image from An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).

Document Ranking

➔ Many neural IR models depend on learning low-dimensional vector representations (or embeddings) of query and document text.

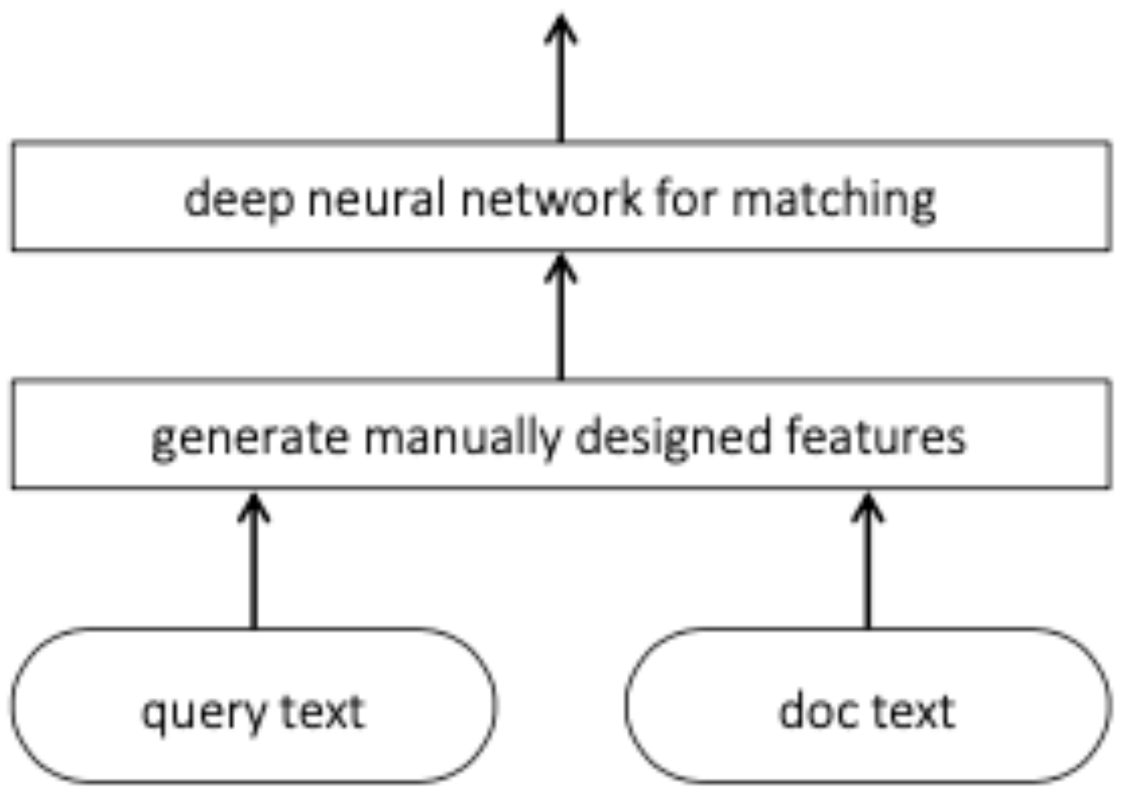
➔ And then use them with traditional IR models in conjunction with simple similarity metrics (e.g., cosine similarity).



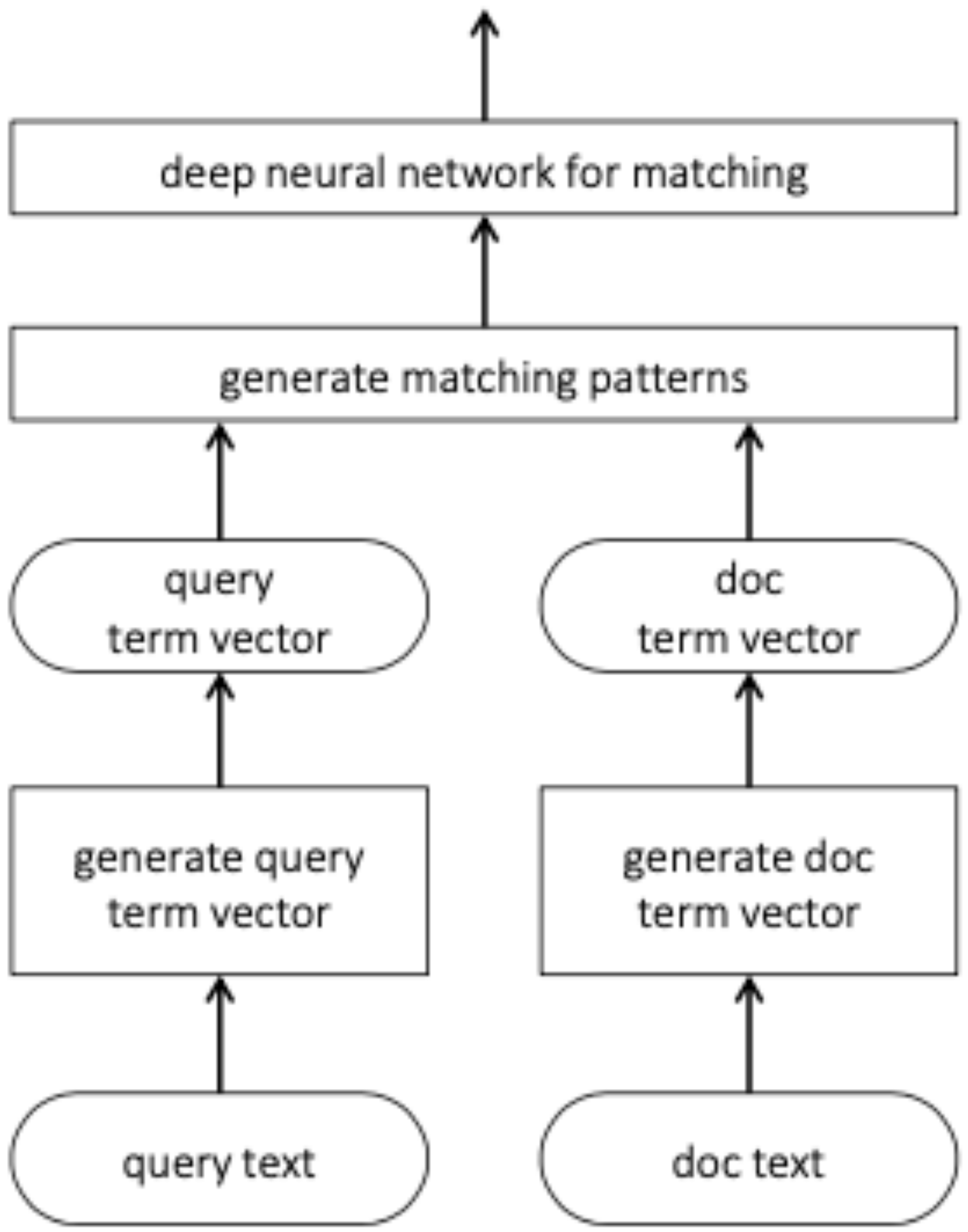
(c) Learning query and document representations for matching (*e.g.*, (Huang *et al.*, 2013; Mitra *et al.*, 2016a))

Image from An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).

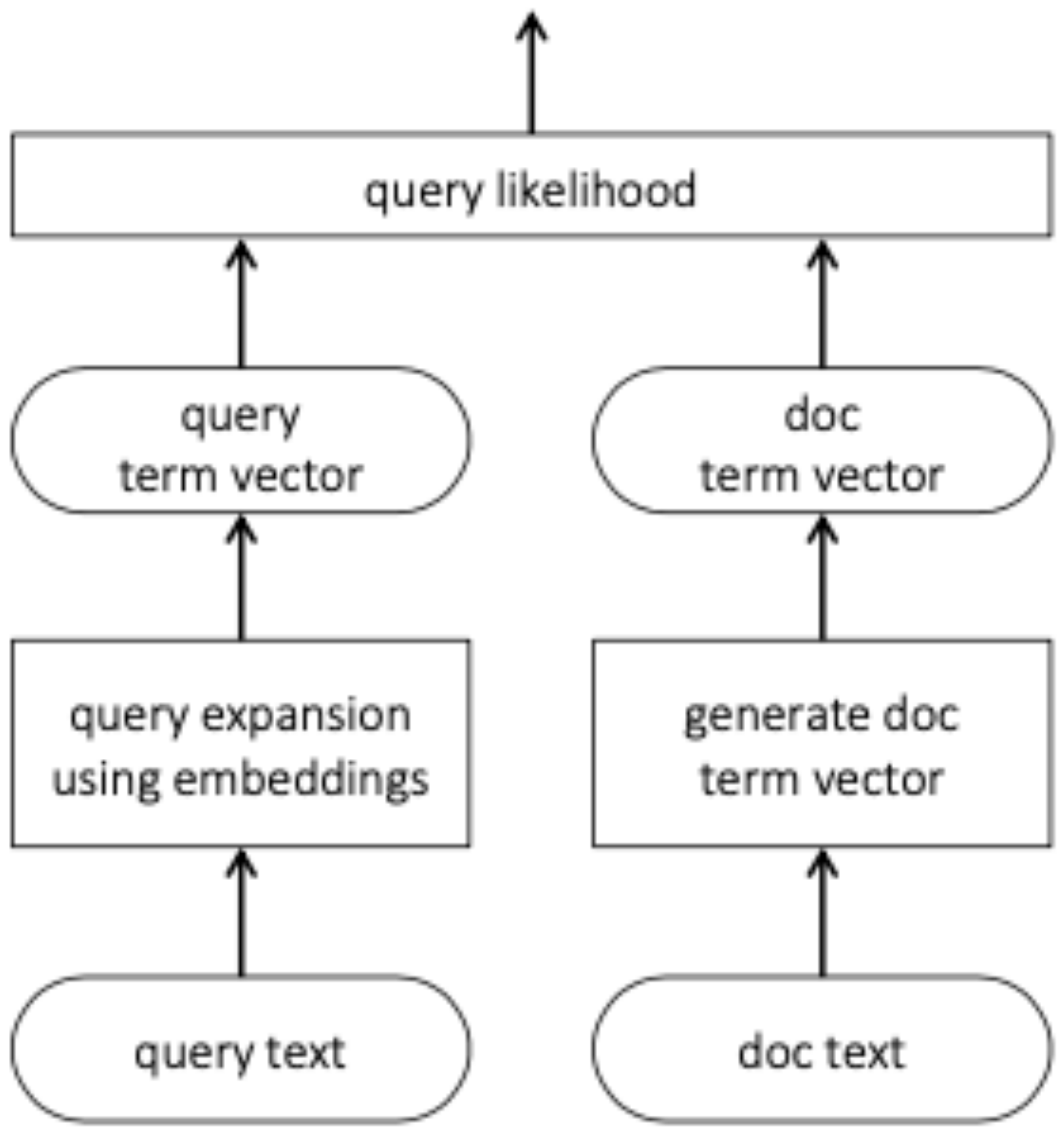
Document Ranking



(a) Learning to rank using manually designed features (*e.g.*, Liu (2009))



(b) Estimating relevance from patterns of exact matches (*e.g.*, (Guo *et al.*, 2016a; Mitra *et al.*, 2017a))



(d) Query expansion using neural embeddings (*e.g.*, (Roy *et al.*, 2016; Diaz *et al.*, 2016))

Image from An Introduction to Neural Information Retrieval, B. Mitra and N. Craswell (2018).

Word Embeddings

- ➔ In the 1950s, many linguists formulated the distributional hypothesis: words that occur in the same contexts tend to have similar meanings.
- ➔ According to this hypothesis, the meaning of words can be inferred by their usage together with other words in existing texts.
- ➔ “You shall know a word by the company it keeps.”, John Firth (1957)
- ➔ In recent years, the distributional hypothesis has been applied to create semantic understandings of terms and term sequences through word embeddings.
- ➔ A word embedding is a numerical vector that represents the semantic meaning of a given term sequence.

Recall Sparse Vectors Representation

query | exact term lookup in inverted index

	apple	caffeine	cheese	coffee	drink	donut	food	juice	pizza	tea	water
latte	0	0	0	0	0	0	0	0	0	0	0
cappuccino	0	0	0	0	0	0	0	0	0	0	0
apple juice	**1**	0	0	0	0	0	0	**1**	0	0	0
cheese pizza	0	0	**1**	0	0	0	0	0	**1**	0	0
donut	0	0	0	0	0	**1**	0	0	0	0	0
soda	0	0	0	0	0	0	0	0	0	0	0
green tea	0	0	0	0	0	0	0	0	0	**1**	0
water	0	0	0	0	0	0	0	0	0	0	**1**
cheese bread sticks	0	0	**1**	0	0	0	0	0	0	0	0
cinnamon sticks	0	0	0	0	0	0	0	0	0	0	0

Figure 2.10 Vectors with one dimension per term in the inverted index. Every query on the left maps to a vector on the right, with a value of "1" for any term in the index that is also in the query, and a "0" for any term in the index that is not in the query.

Image from AI Powered Search, T. Grainger et al. (2022).

Word Embeddings / Dense Vectors

	food	drink	dairy	bread	caffeine	sweet	calories	healthy
apple juice	0	5	0	0	0	4	4	3
cappuccino	0	5	3	0	4	1	2	3
cheese bread sticks	5	0	4	5	0	1	4	2
cheese pizza	5	0	4	4	0	1	5	2
cinnamon bread sticks	5	0	1	5	0	3	4	2
donut	5	0	1	5	0	4	5	1
green tea	0	5	0	0	2	1	1	5
latte	0	5	4	0	4	1	3	3
soda	0	5	0	0	3	5	5	0
water	0	5	0	0	0	0	0	5

Figure 2.11 Word embeddings with reduced dimensions. In this case, instead of one dimension per term (exists or missing), now higher-level dimensions exist that score shared attributes across items such as "healthy", contains "caffeine" or "bread" or "dairy", or whether the item is "food" or a "drink".

Image from AI Powered Search, T. Grainger et al. (2022).

Transformers

- ➔ Static word embeddings map words with multiple senses into an average or most common-sense representation based on the training data used to compute the vectors.
  - ➔ The vector of a word does not change with the other words used in a sentence around it.
- ➔ A transformers is a neural network designed to explicitly take into account the context of arbitrary long sequences of text.
  - ➔ Transformer compute contextualized word embeddings, where the representation of each input token is conditioned by the whole input text.
  - ➔ The most popular contextualized word embeddings are learned with deep neural networks such as the Bidirectional Encoder Representations from Transformers (BERT).
- ➔ With fine-tuning, the parameters of a pre-trained language model can be updated for the domain data and target task.
  - ➔ Pre-training typically requires a huge general-purpose training corpus, and long and expensive computation resources.
  - ➔ On the other side, fine-tuning requires a small domain-specific corpus focused on the downstream task, affordable computational resources and few hours or days of additional training.

Retrieval Architectures

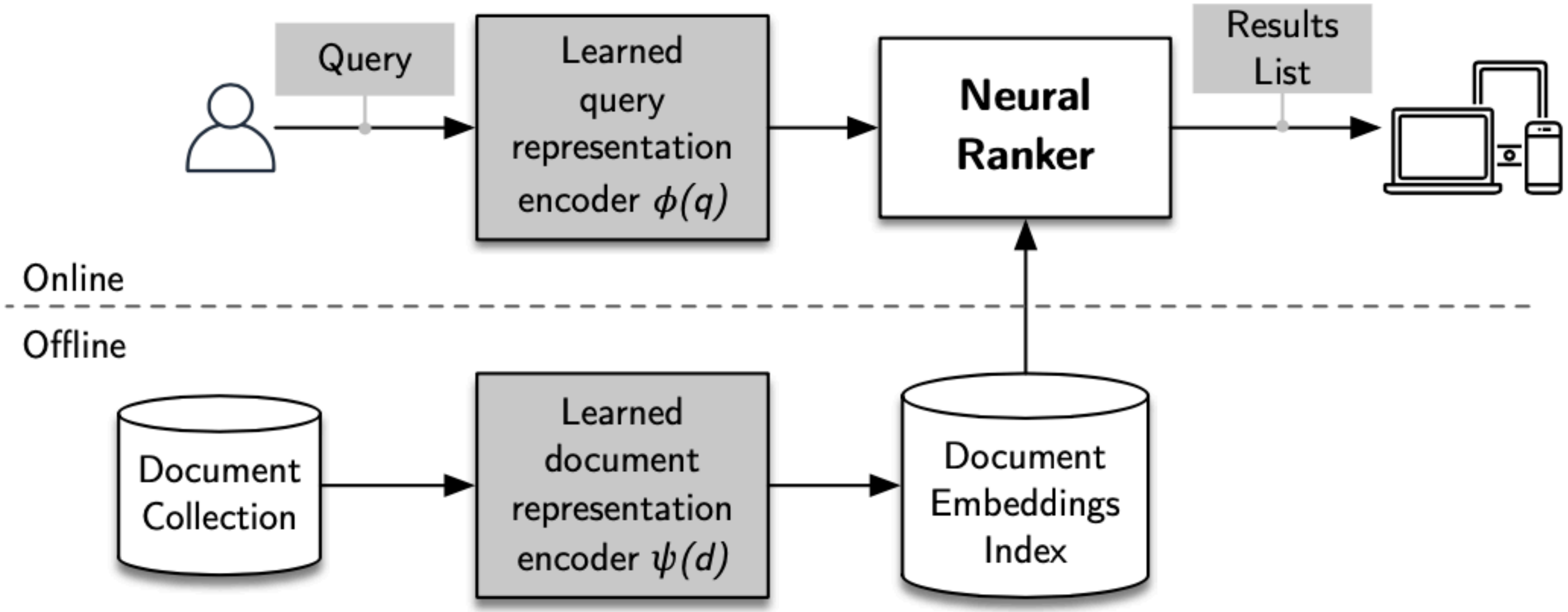


Figure 8: Dense retrieval architecture for representation-focused neural IR systems.

Image from Lecture Notes on Neural Information Retrieval, N. Tonellotto (2022).

Vector Search Systems

- ➔ Also called vector databases.
- ➔ In dense retrieval systems, document embeddings are pre-computed, thus the need for storing and searching through these document embeddings.
- ➔ Given a query, represented as a dense vector, instead of computing the distance between this vector and all document dense vectors (too expensive), the strategy is to use approximate nearest neighbor search.

- ➔ Solr (since version 9) includes support for Dense Vector Search.
- ➔ https://solr.apache.org/guide/solr/latest/query-guide/dense-vector-search.html

Vector Search Open-Source Frameworks

- ➔ Milvus, https://milvus.io
- ➔ Vespa, https://vespa.ai
- ➔ Qdrant, https://qdrant.tech

- ➔ Vector Search benchmarks
  - ➔ http://ann-benchmarks.com

References

- ➔ Liu, Tie-Yan. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3.3 (2009): 225-331.
- ➔ Mitra, Bhaskar, and Nick Craswell. An introduction to neural information retrieval. Foundations and Trends in Information Retrieval 13.1 (2018): 1-126.
- ➔ Tonellotto, Nicola. Lecture Notes on Neural Information Retrieval. 2022.
- ➔ Grainger, Trey, Doug Turnbull, and Max Irwin. AI Powered Search. Manning (2022).