344.075 KV: Natural Language Processing
Information Retrieval with Neural Networks

Navid Rekab-Saz

navid.rekabsaz@jku.at

Institute of Computational Perception

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

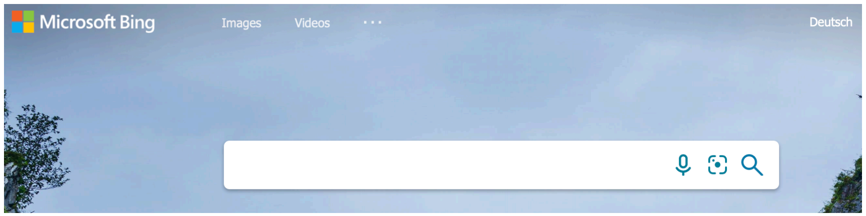
Agenda

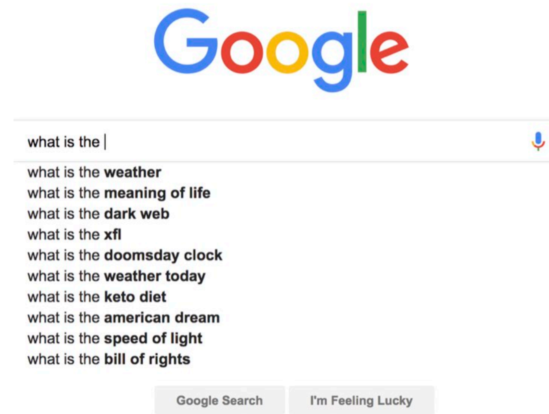
- Introduction to IR
- Scoring, ranking, and indexing
- Evaluation
- Neural IR Models

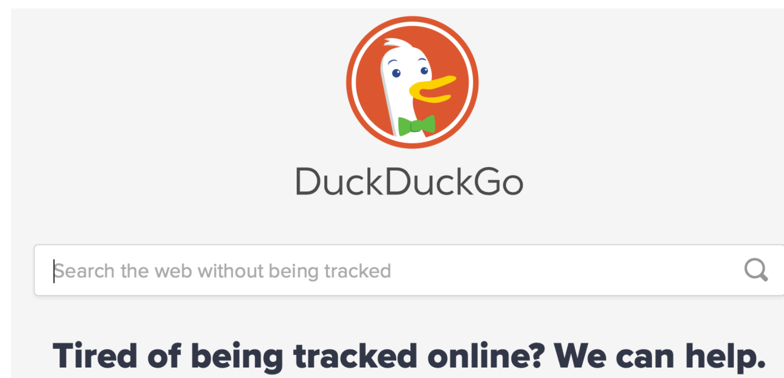
Agenda

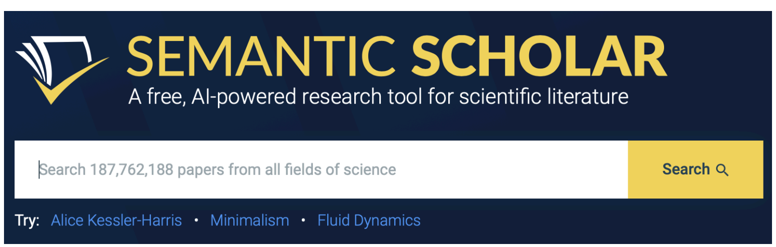
- **Introduction to IR**
- Scoring, ranking, and indexing
- Evaluation
- Neural IR Models

Information Retrieval everywhere!

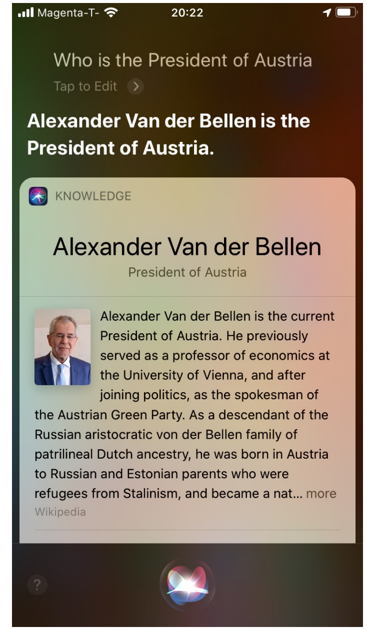


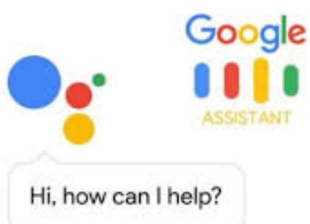






Information Retrieval everywhere!





IBM Watson and Jeopardy

Information Retrieval

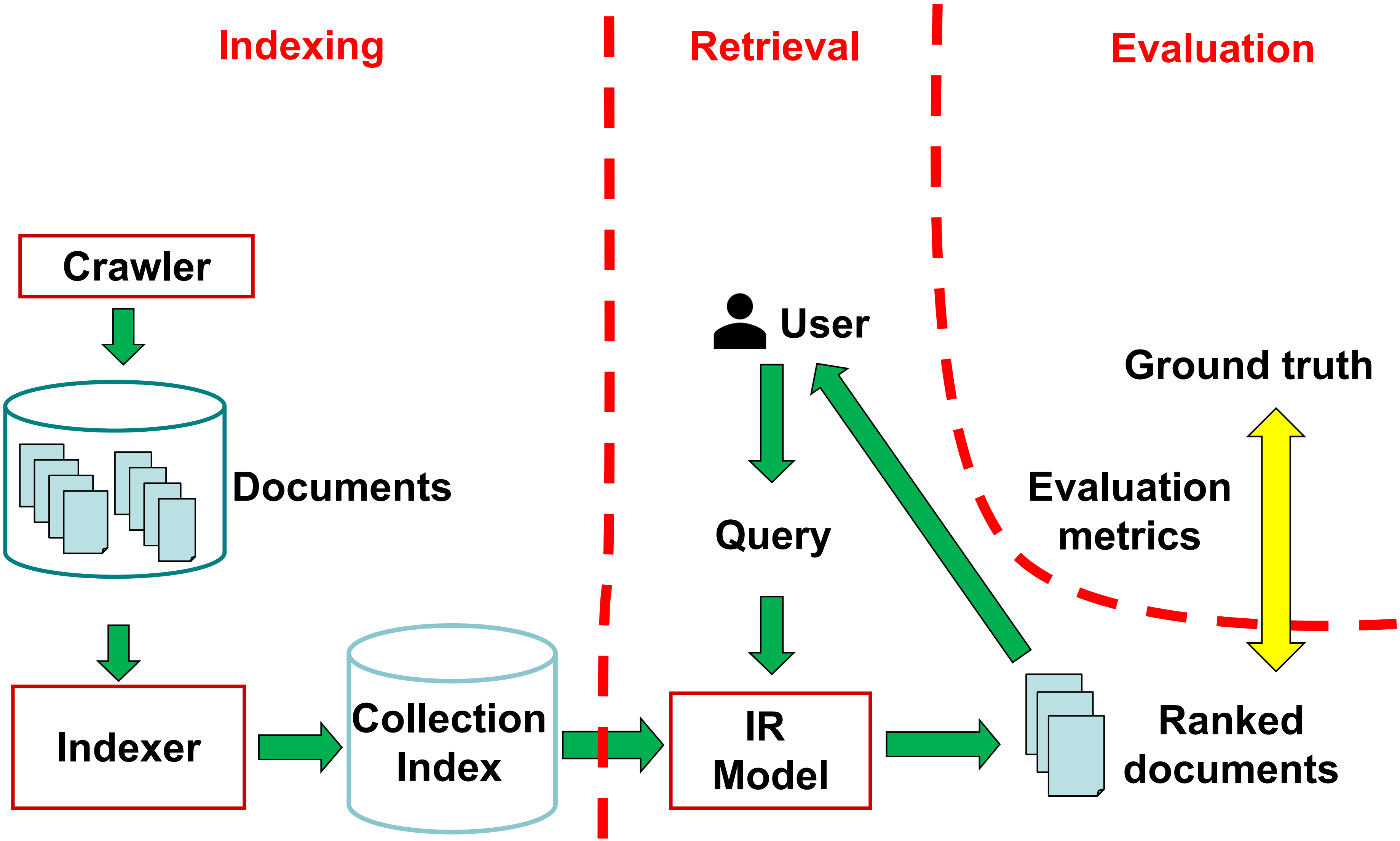
- Information Retrieval (IR) is finding material (usually in the form of documents) of an unstructured nature that satisfies an information need from within large collections

- When talking about IR, we frequently think of web search

- The goal of IR is however to retrieve relevant contents to the user's information need

- IR covers a wide set of tasks such as …
  - Ranking, question/answering, information summarization
  - But also … user behavior/experience study, personalization, etc.

Simplified architecture of an IR system



Terminology

- Information need
  - E.g. *My swimming pool bottom is becoming black and needs to be cleaned*
- Query
  - A designed representation of users’ information need
  - E.g. *pool cleaner*
- Document
  - A unit of data in text, image, video, audio, etc.
- Relevance
  - Whether a document satisfies user’s information need
  - Relevance has multiple aspects: topical, semantic, temporal, spatial, etc.

Ad-hoc IR (all we discuss in this lecture)

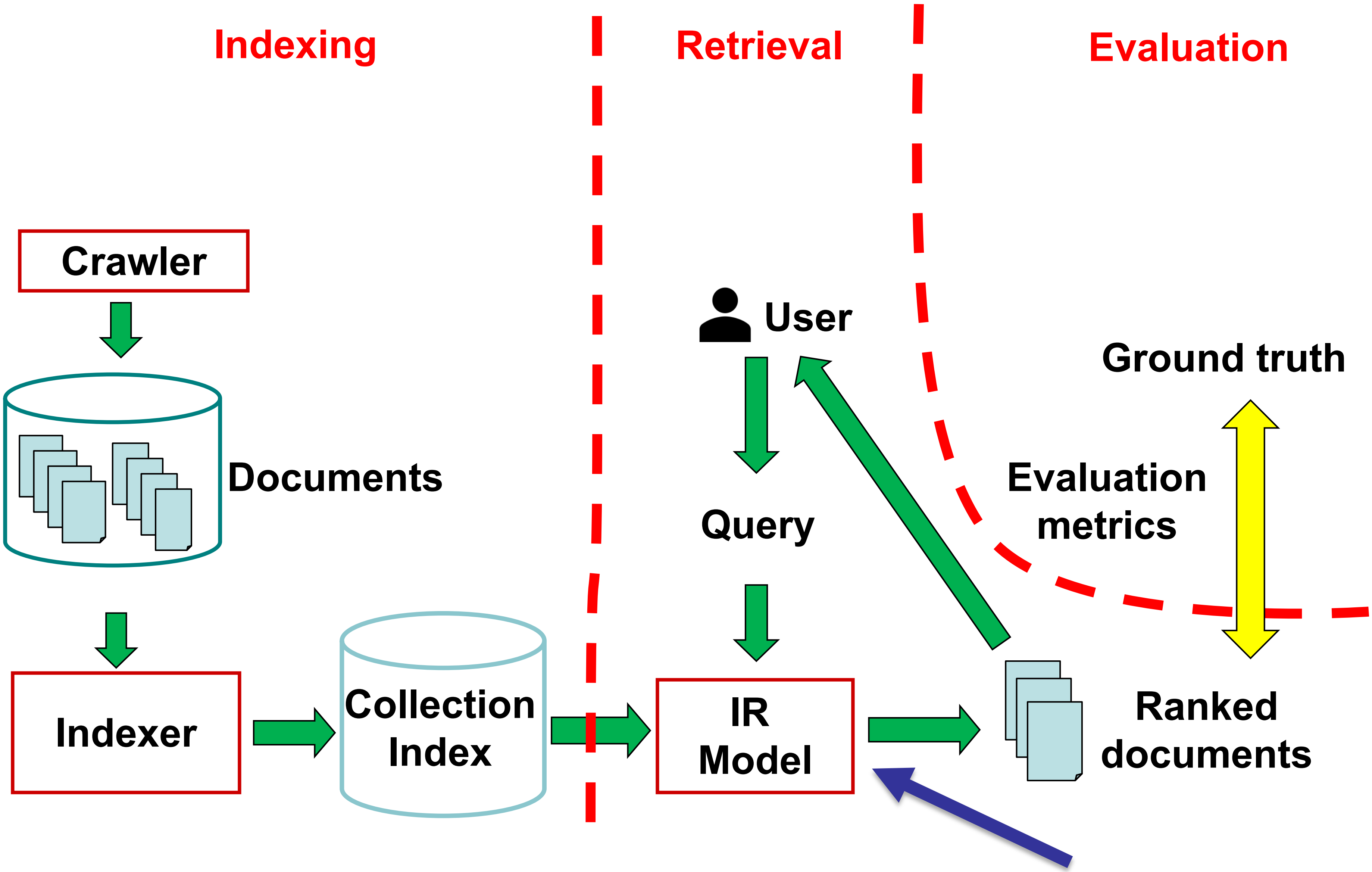
- Studying the methods to estimate relevance, solely based on the contents (texts) of queries and documents
  - In ad-hoc IR, *meta-knowledge* such as temporal, spatial, user-related information are normally taken out
  - The focus of ad-hoc IR is on methods to exploit contents

- Ad-hoc IR is a part of the ranking mechanism of search engines, but there are several other aspects…
  - Diversity of information
  - Personalization
  - Information need understanding
  - SE log files analysis
  - …

Agenda

- Introduction to IR
- **Scoring, ranking, and indexing**
- Evaluation
- Neural Ranking Models

Simplified architecture of an IR system



Relevance scoring & IR models

query: (Q)

wisdom of mountains

$D20$

$D1402$

$D5$

$D100$

Documents are ranked according to their predicted relevance scores to the query, from highest to lowest

How can we calculate relevance of a query to a document? → IR models

Definitions

Definitions

- Collection $\mathbb{D}$ contains $|\mathbb{D}|$ documents

- Document $D \in \mathbb{D}$ consists of terms $d_1, d_2, \ldots, d_m$

- Query Q consist of terms $q_1, q_2, \ldots, q_n$

- An IR model calculates/predicts a relevance score between the query and document:

$$\text{score}(Q, D)$$

Exact-matching IR models – TF-IDF

- Classical (exact-matching) IR models – in their basic forms – assign importance weights to each query term that appears in a document

- <u>Recap:</u> TF-IDF was introduced as a term weighting method
- TF-IDF as an IR model to calculate relevance score:

$$\text{score}(Q, D) = \sum_{q \in Q} \text{tf-idf}_{q,D} = \sum_{q \in Q} \underbrace{\log\left(1 + \text{tc}_{q,D}\right)}_{\textbf{Term matching score}} \times \underbrace{\log\left(\frac{|\mathbb{D}|}{\text{df}_q + 1}\right)}_{\textbf{Term Salience}}$$

$\text{tc}_{q,D}$ number of times query term q appears in document D

df_q number of documents in which query term q appears

Exact-matching IR models – PL

- Pivoted Length Normalization model

$$\mathrm{score}(Q,D) = \sum_{q\in Q} \frac{\overbrace{\log\left(1+\mathrm{tc}_{q,D}\right)}^{\text{Term matching score}}}{\underbrace{1-b+b\,\frac{|D|}{avgdl}}_{\text{Document length normalization}}} \times \underbrace{\mathrm{idf}\,(q)}_{\text{Term Salience}}$$

$\mathrm{tc}_{q,D}$ number of times query term q appears in document D

$avgdl$ average length of the documents in the collection

b a hyper parameter that controls document length normalization

Exact-matching IR models – BM25

- BM25 model *(slightly simplified)*:

$$\mathrm{score}(Q,D)=\sum_{q\in Q}\overbrace{\frac{(k_1+1)\mathrm{tc}_{q,D}}{k_1\underbrace{\left(1-b+b\,\frac{|D|}{avgdl}\right)}_{\text{Length normalization}}+\mathrm{tc}_{q,D}}}^{\text{Term matching score \& normalization}}\times\underbrace{\mathrm{idf}\,(q)}_{\text{Term Salience}}$$

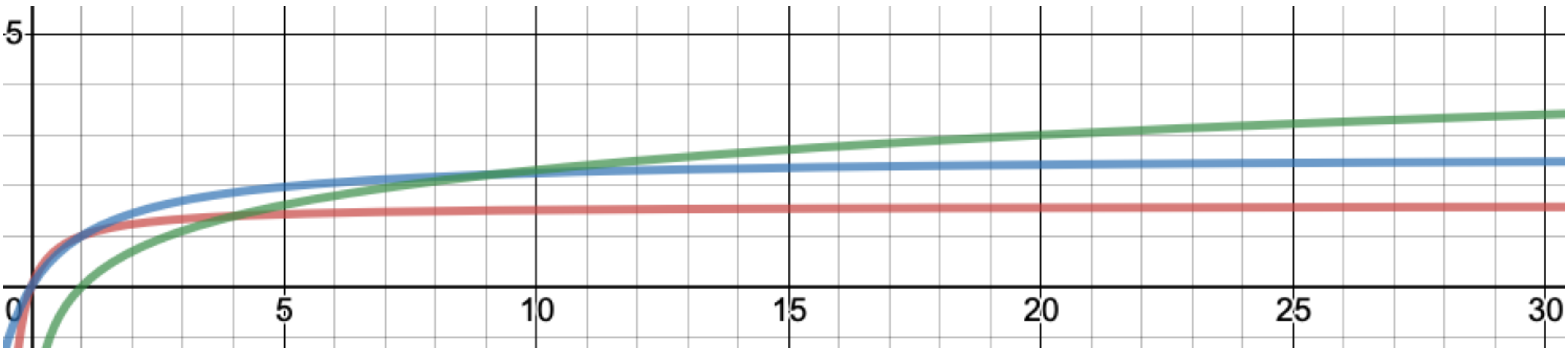
$\mathrm{tc}_{q,D}$ number of times query term q appears in document D

$avgdl$ average length of the documents in the collection

b a hyper parameter that controls length normalization

k_1 a hyper parameter that controls term frequency *saturation*

Exact-matching IR models – BM25

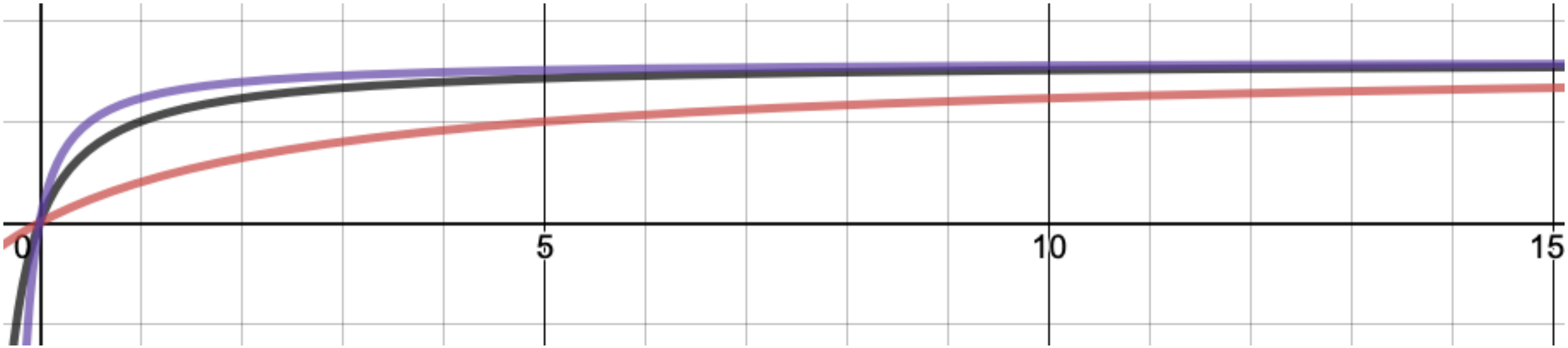


Green: $\log\ \mathrm{tc}_{q,D}$ → TF

Red: $\frac{(0.6+1)\mathrm{tc}_{q,D}}{0.6+\mathrm{tc}_{q,D}}$ → BM25 with $k_1 = 0.6$ and $b = 0$

Blue: $\frac{(1.6+1)\mathrm{tc}_{q,D}}{1.6+\mathrm{tc}_{q,D}}$ → BM25 with $k_1 = 1.6$ and $b = 0$

Exact-matching IR models – BM25



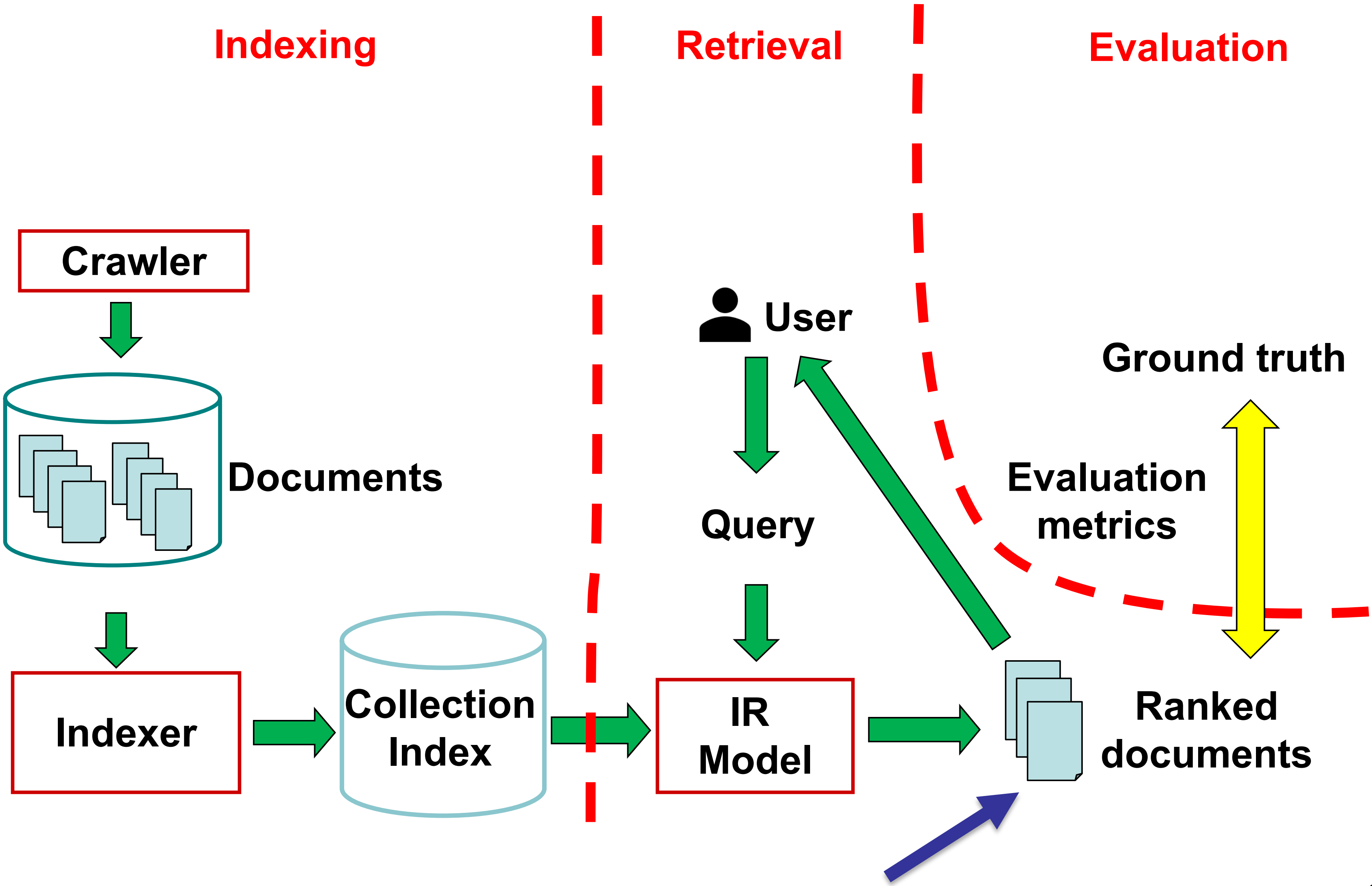
BM25 models with $k_1 = 0.6$ and $b = 1$

Purple: $\frac{(0.6+1)\text{tc}_{q,D}}{0.6(1-1+1(\frac{1}{2}))+\text{tc}_{q,D}}$ → Document length ½ of $avgdl$

Black: $\frac{(0.6+1)\text{tc}_{q,D}}{0.6(1-1+1(\frac{2}{2}))+\text{tc}_{q,D}}$ → Document length the same as $avgdl$

Red: $\frac{(0.6+1)\text{tc}_{q,D}}{0.6(1-1+1(\frac{10}{2}))+\text{tc}_{q,D}}$ → Document length 5 times higher than $avgdl$

Simplified architecture of an IR system



Ranking results as we know!

Google | google daily query volume

Web | News | Videos | Images | Shopping | More | Search tools

About 5,910,000 results (0.42 seconds)

Google Search Statistics - Internet Live Stats
www.internetlivestats.com/**google**-search-statistics/
Historical search **volume**, growth rate, and **Google's** share of global search market. ... launched, **Google** was already answering 3.5 million search **queries daily**.

Insight into **Google** Search **Query** Numbers and What It ...
getstat.com/**google**-search-**queries**-the-numbers/
Jul 27, 2012 - Insights into the true meaning of **Google** Search **Queries** and the numbers behind it. ... **Google** has had an immense impact on how we operate in **everyday** ... Each month, the sheer **volume** of **queries** it answers continues to ...

Google Trends
www.**google**.com/trends/ Google
50,000+ searches. Image Source - New York **Daily** News · David Wilson. 50,000+ searches. Image Source - NBCSports.com · Marilyn Burns. 20,000+ searches.

Google Annual Search Statistics | Statistic Brain
www.statisticbrain.com/**google**-searches/
The first funding for **Google** was an August 1998 contribution of US$100,000 from ... Year, Annual Number of **Google** Searches, Average Searches **Per Day**.

How many search **queries** does **Google** serve worldwide ...
www.quora.com/How-many-search-**queries**-does-**Google**-serve-w... Quora
Answer 1 of 8: This is latest data that Matt Cutts update yesterday - **Google** has seen more than 30 trillion URLs and crawls 20 billion pages **a day**. 3 billion...

Google Trends - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Google**_Trends Wikipedia
Google Trends also allows the user to compare the **volume** of searches between ... the information provided by **Google** Trends **daily**; Hot Trends is updated hourly. ... Because the relative frequency of certain **queries** is highly correlated with the ...

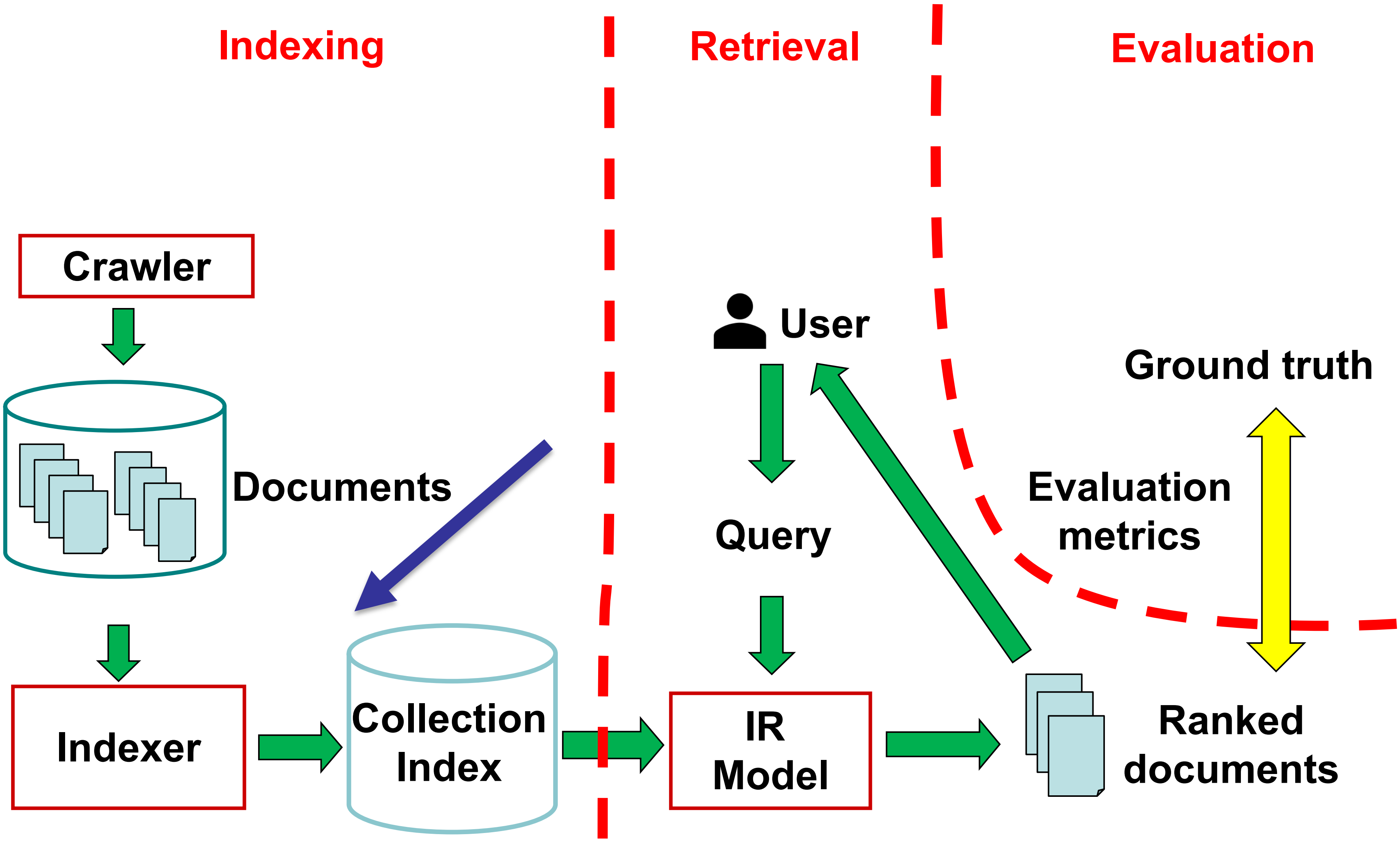
Sample ranking results – format in research!

- TREC run file: standard format to report the ranking results of top-1000 documents for some queries, retrieved by a model

```
qry_id   (iter)   doc_id        rank      score        run_id

  2       Q0      1782337        1       21.656799     cool_model
  2       Q0      1001873        2       21.086500     cool_model
                                 …
  2       Q0      6285819        999     3.43252       cool_model
  2       Q0      6285819        1000    1.6435        cool_model
  8       Q0      2022782        1       33.352300     cool_model
  8       Q0      7496506        2       32.223400     cool_model
  8       Q0      2022782        3       30.234030     cool_model
                                 …
  312     Q0      2022782        1       14.62234      cool_model
  312     Q0      7496506        2       14.52234      cool_model
                                 …
```

Simplified architecture of an IR system



Efficient retrieval with pre-computed Collection Index

query: wisdom of mountains
(Q)

$D20$

$D1402$

$D5$

$D100$

How can we efficiently calculate relevance scores for documents? → Collection Index

☞ Since the IR models so far are based on <u>exact matching</u>, an we can focus on calculating relevance scores only for the <u>documents that contain query terms</u> → done by Inverted Index

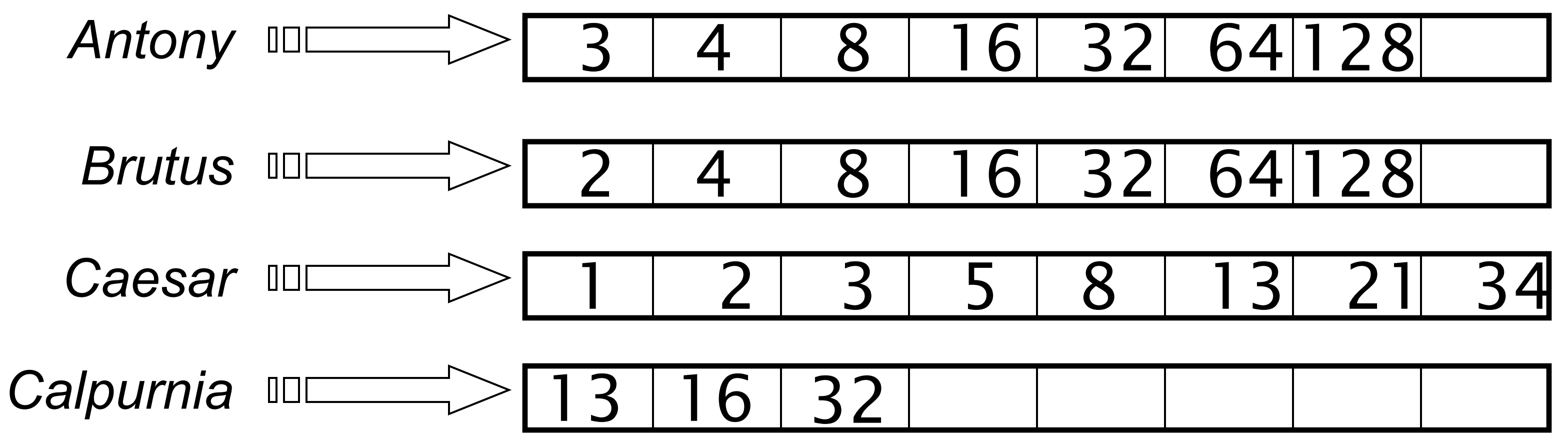
Inverted index

- Inverted index is a data structure for efficient retrieval
  - Inverted index is created once at index time for all documents in the collection, and used for each query during query time
- Inverted index creates a posting list for each unique term in collection
  - A posting list of a term contains the list of the IDs of the documents, in which the term appears

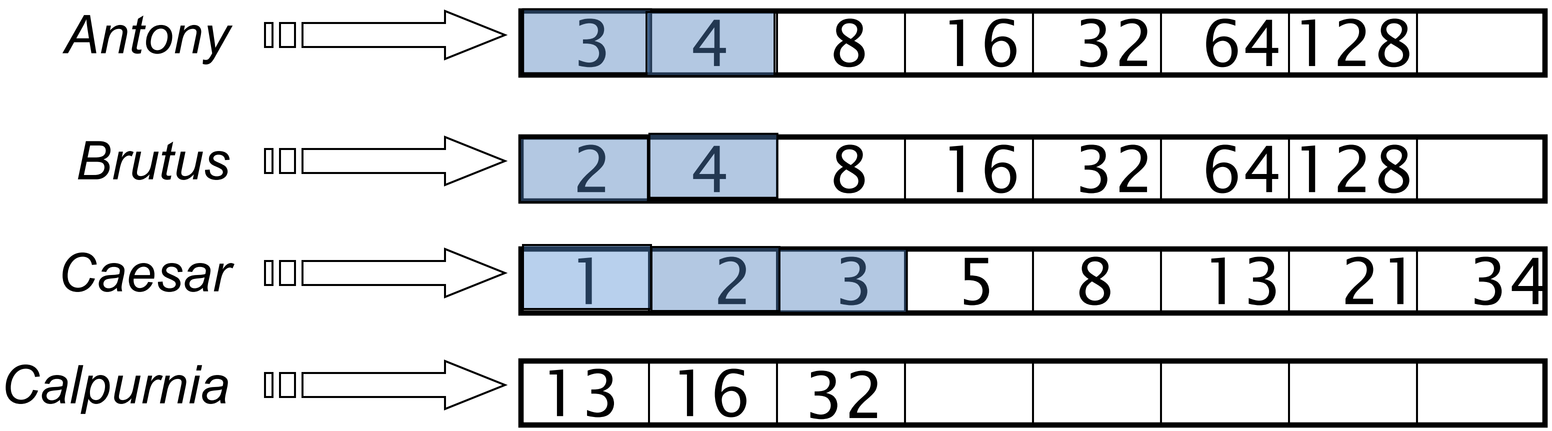
Antony →	3	4	8	16	32	64	128	
Brutus →	2	4	8	16	32	64	128	
Caesar →	1	2	3	5	8	13	21	34
Calpurnia →	13	16	32					

Retrieval process using inverted index

1. Fetch the posting lists of query terms
2. Traverse through posting lists, and calculate the relevance score for each document in the posting lists
3. Retrieve top *n* documents with the highest relevance scores



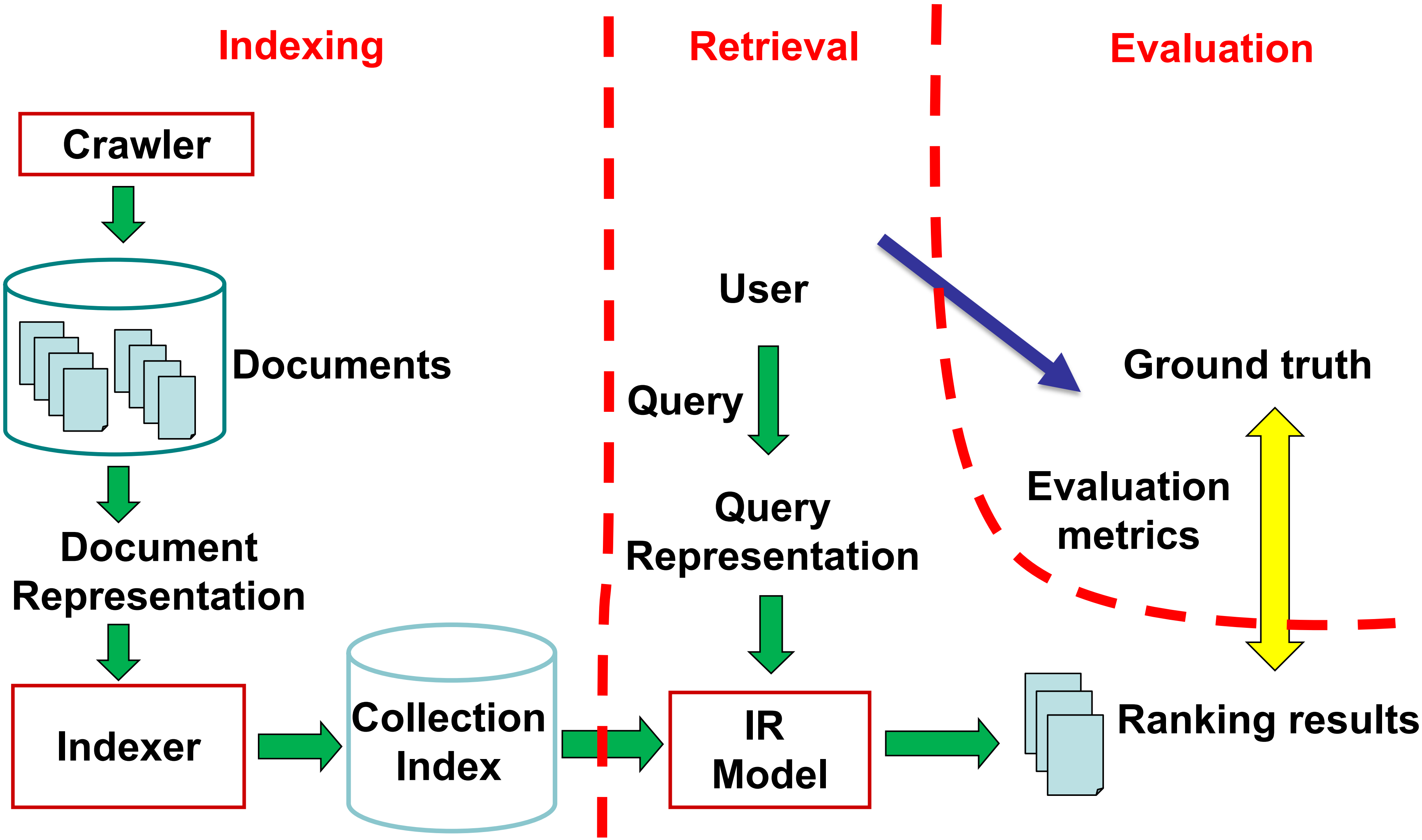
Search with concurrent traversal



Agenda

- Introduction to IR
- Scoring, ranking, and indexing
- **Evaluation**
- Neural IR Models

Components of an IR System (simplified)



IR evaluation

- Evaluation of an IR system requires three elements:
  - A benchmark document collection
  - A benchmark suite of queries
  - Relevance judgements for pairs of query–document
    - Judgements specifies whether the document addresses the underlying information need of the query
    - Ideally done by human, but also through user interactions
    - Relevance judgements appear in forms of …
      - Binary: 0 (non-relevant) vs. 1 (relevant), or …
      - Multi-grade: more nuanced relevance levels, e.g. 0 (non-relevant), 1 (fairly relevant), 2 (relevant), 3 (highly relevant)

Evaluation Campaigns

Text REtrieval Conference (TREC)

...to encourage research in information retrieval from large text collections.

- Text REtrieval Conference (TREC)

https://trec.nist.gov

- Conference and Labs of the Evaluation Forum (CLEF)

http://www.clef-initiative.eu

- MediaEval Benchmarking Initiative for Multimedia Evaluation

http://www.multimediaeval.org

Sample relevance judgement – format in research!

- TREC QRel (QueryRelevance) file: standard format to provide relevance judgements of some queries regarding to some documents

```
qry_id (iter)    doc_id   relevance_grade

 101    0        183294        0
 101    0        123522        2
 101    0        421322        1
 101    0        12312         0
                …
 102    0        375678        2
 102    0        123121        0
                …
 135    0        124235        0
 135    0        425591        1
                …
```

Common IR Evaluation Metrics

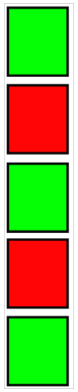
- Binary relevance
  - Precision@*n* (P@*n*)
  - Recall@*n* (P@*n*)
  - Mean Reciprocal Rank (MRR)
  - Mean Average Precision (MAP)

- Multi-grade relevance
  - Normalized Discounted Cumulative Gain (NDCG)

Precision@*n*

- **Precision@*n***: fraction of retrieved docs at top-*n* results that are relevant

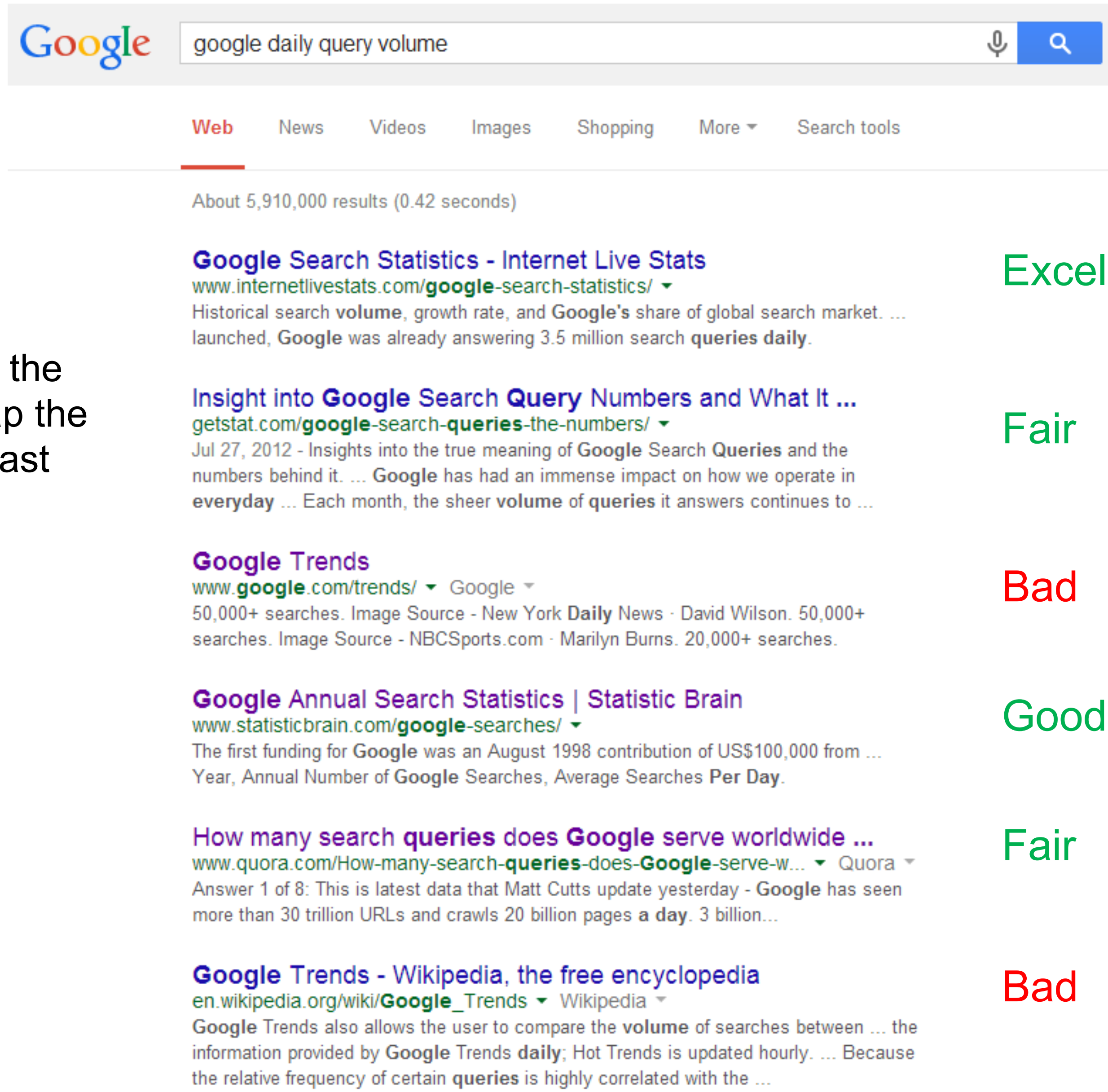
- Example:
  - P@3 = 2/3
  - P@4 = 2/4
  - P@5 = 3/5



- Final evaluation result is the mean of P@*n* across all queries in test set

Rank positions matter!

P@6 remains the same if we swap the first and the last result!



Excellent

Fair

Bad

Good

Fair

Bad

Discounted Cumulative Gain (DCG)

- A popular measure for evaluating web search and other related tasks

- Assumptions:
  - Highly relevant documents are more useful than marginally relevant documents (multi-grade relevance)
  - The lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined
    - This common behavior of users when interacting with ranked lists is known as *position bias*

Discounted Cumulative Gain (DCG)

- Gain: define gain as graded relevance, provided by relevance judgements

- Discounted Gain: gain is reduced as going down the ranking list. A common discount function: ${}^{1}/_{\log_2(\text{rank position})}$
  - With base 2, the discount at rank 4 is 1/2, and at rank 8 it is 1/3

- Discounted Cumulative Gain: the discounted gains are accumulated starting at the top of the ranking to the lower ranks till rank n

Discounted Cumulative Gain (DCG)

- Given the ranking results of a query, DCG at the position n of the ranking list is:

$$\mathrm{DCG@}n = rel_1 + \sum_{i=2}^{n} \frac{rel_i}{\log_2 i}$$

where rel_i is the graded relevance (in relevance judgements) of the document at position i of the ranking results

- Alternative formulation (commonly used):

$$\mathrm{DCG@}n = \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

DCG Example

Rank	Retrieved document ID	Gain (relevance)	Discounted gain	DCG
1	$d20$	3	3	3
2	$d243$	2	2/1=2	5
3	$d5$	3	3/1.59=1.89	6.89
4	$d310$	0	0	6.89
5	$d120$	0	0	6.89
6	$d960$	1	1/2.59=0.39	7.28
7	$d234$	2	2/2.81=0.71	7.99
8	$d9$	2	2/3=0.67	8.66
9	$d35$	3	3/3.17=0.95	9.61
10	$d1235$	0	0	9.61

$$\mathrm{DCG@10} = 9.61$$

Normalized DCG (NDCG)

- DCG results of different queries are not comparable,
  - Based on the relevance judgements of queries, the ranges of *good* and *bad* DCG results can be different between queries

- To normalize DCG at ranking position n:
  - For each query, estimate Ideal DCG (IDCG) which is the DCG for the ranking list, sorted by relevance judgements
  - Calculate NDCG by dividing DCG by IDCG

- Final NDCG@n is the mean across all test queries

Agenda

- Introduction to IR
- Scoring, ranking, and indexing
- Evaluation
- **Neural IR Models**

Learning to predict relevance scores

- Instead of defining a formula as in classical IR models, we can learn to predict relevance scores ($\mathrm{score}(Q, D)$) by training a neural network model

- Such neural IR models can benefit from semantic relations in the embedding space, …
  - Hence do soft-matching between terms (in contrast to exact-matching in classical IR models)

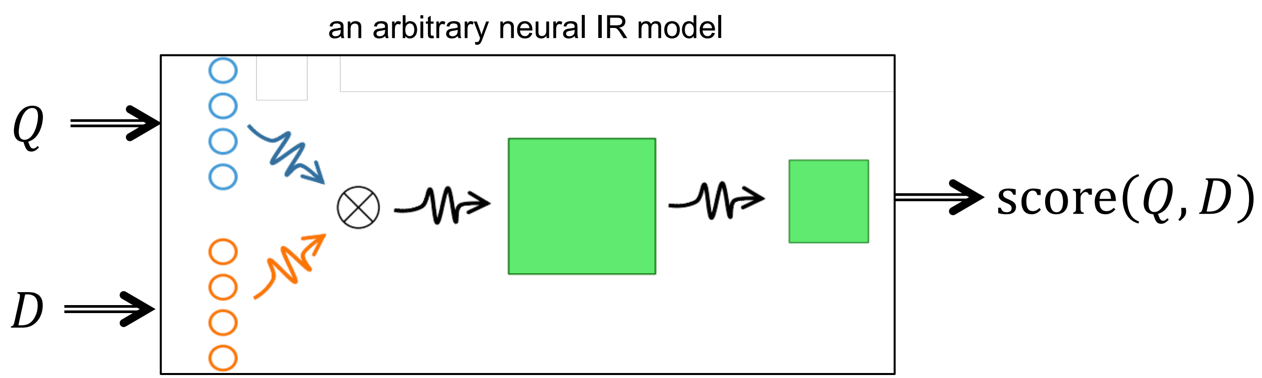


Image source: Pang, Liang, et al. "A deep investigation of deep ir models." *arXiv preprint arXiv:1707.07700* (2017).

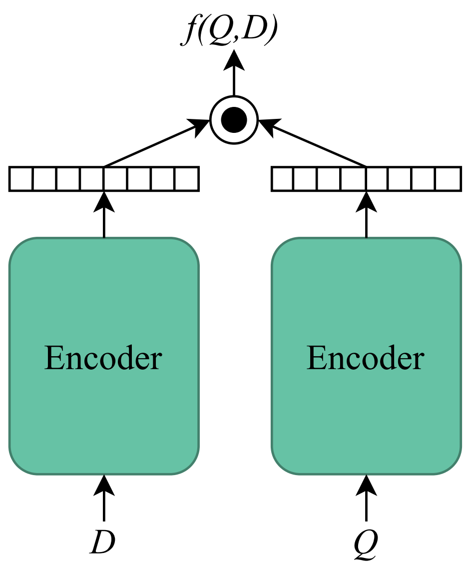
Elements of neural IR models

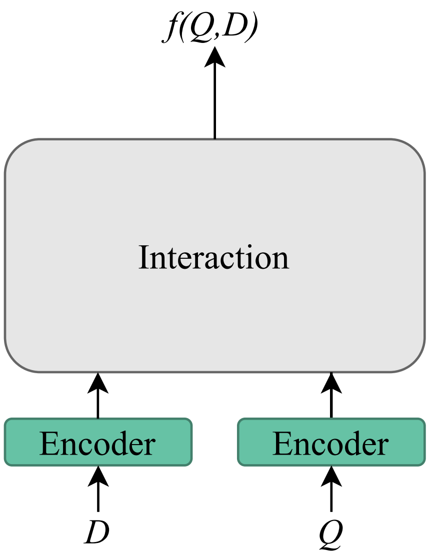
- Model architecture
- Loss function and training
- Inference process (at query time)

Elements of neural IR models

- **Model architecture**
- Loss function and training
- Inference process (at query time)

Architectural categories of neural IR models





Representation-focused models

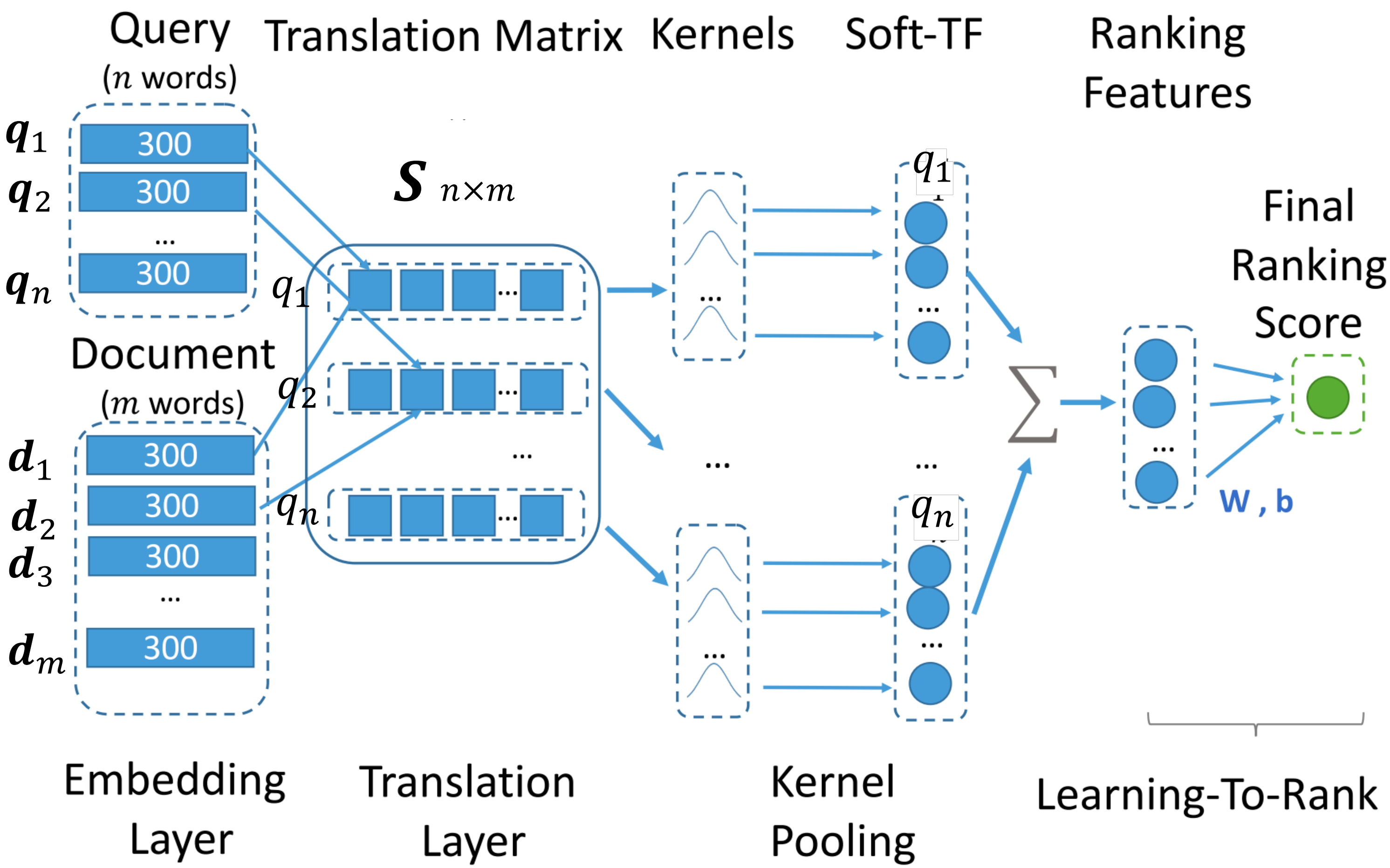
- *Encode* the document in a vector and the query in another vector
- Consider the (cosine) *similarity* between these two vectors as the relevance score of the query to the document

Interaction-focused models

- Calculate the *interactions* (i.e. cosine similarities) between each word embedding of the document to each word embedding of query
- Create a *feature vector* from the results of these interactions
- Estimate query-to-document relevance using the feature vector

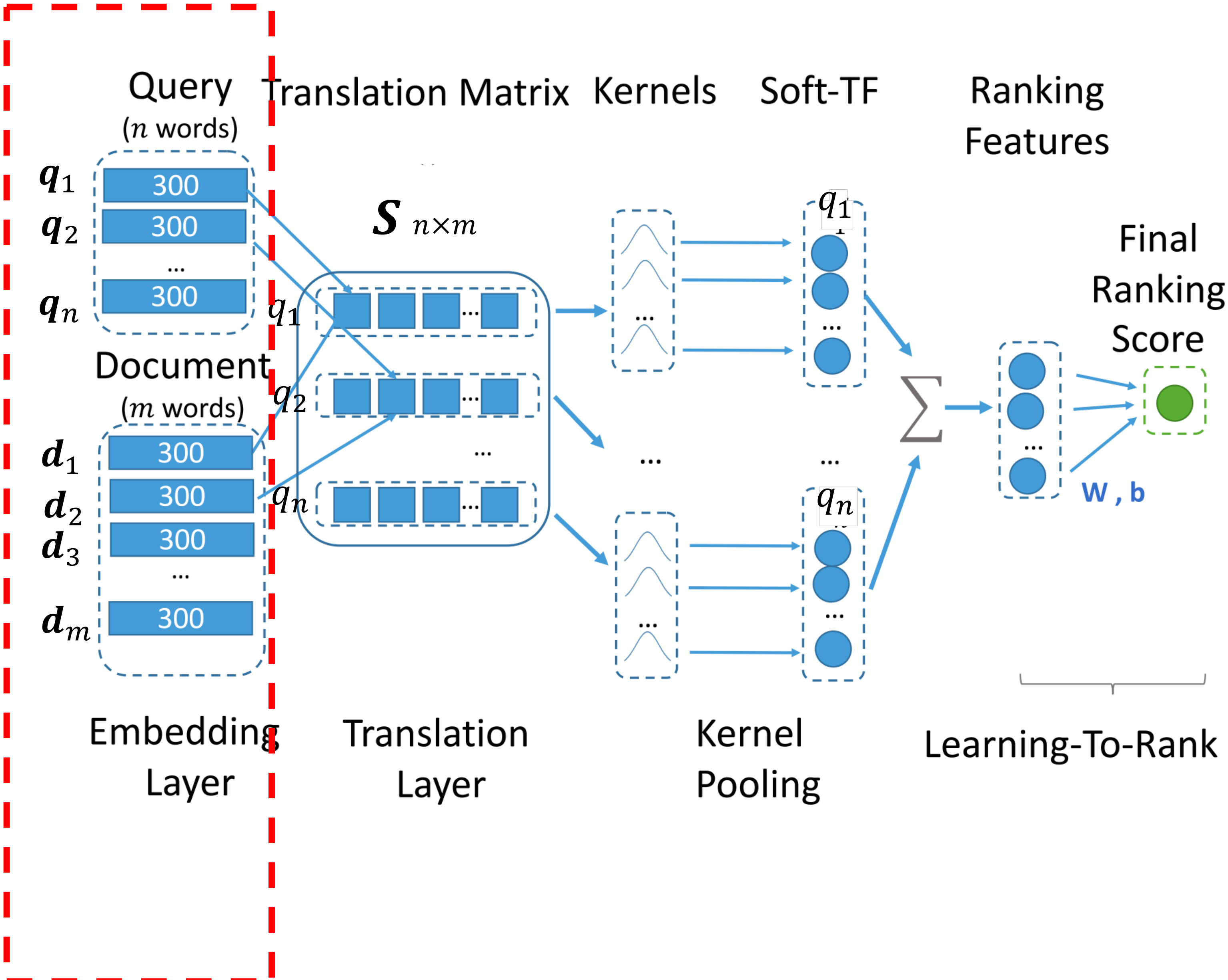
A sample interaction-focused model:

A neural IR model with kernels



Xiong, Chenyan, et al. "End-to-end neural ad-hoc ranking with kernel pooling." *Proceedings of SIGIR*. 2017.

A neural IR model with kernels

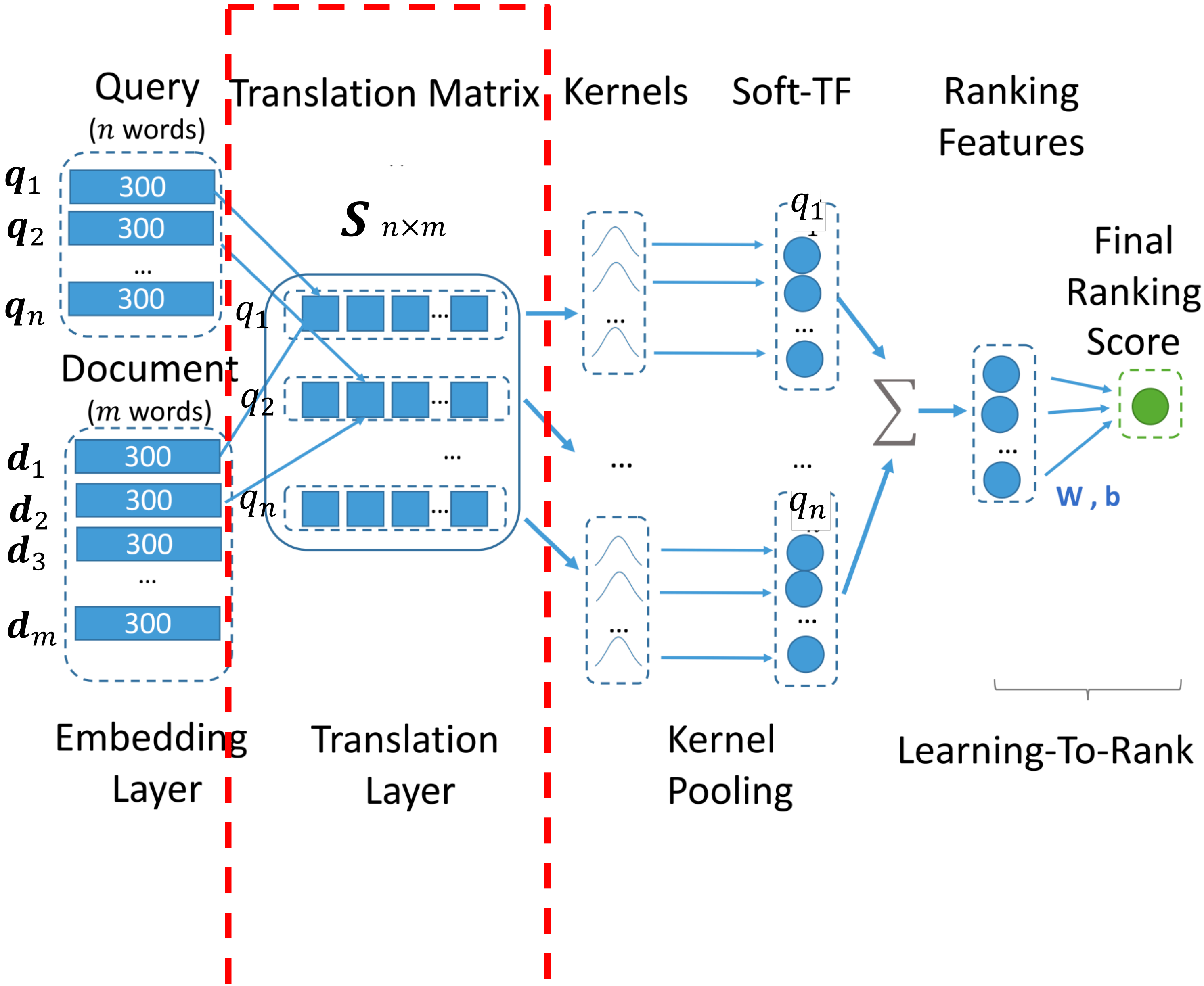


Translation Matrix

- n : number of query terms
- m : number of document terms
- $\boldsymbol{q}_i$: embedding of ith query term
- $\boldsymbol{d}_j$: embedding of jth document term

- Word embeddings can be …
  - fetched from a pre-trained word embedding model (like GloVe or word2vec) or …
  - randomly initialized and get trained together with the other parameters of the IR model

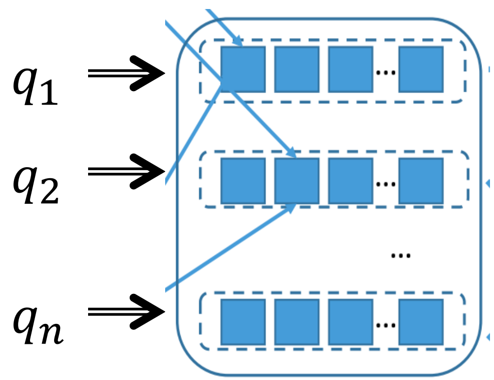
A neural IR model with kernels



Translation (Similarity) Matrix

$$s_{i,j} = \cos(\boldsymbol{q}_i, \boldsymbol{d}_j)$$

Matrix $\boldsymbol{S}$ with
n rows (queries) and
m columns (documents)

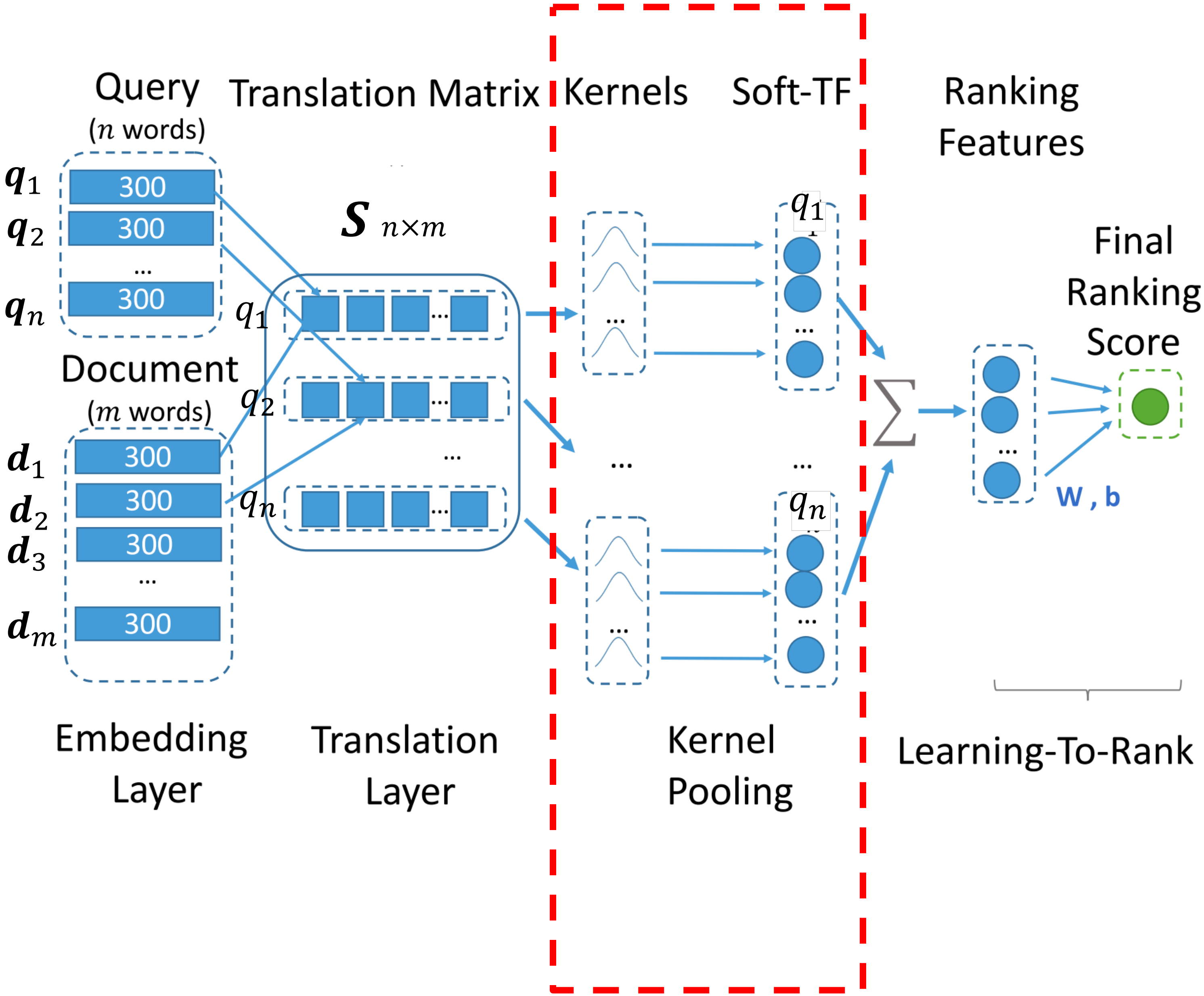


- A sample Translation matrix $\boldsymbol{S}$ between a query with 2 terms and a document with 4 words:

$$\boldsymbol{S} = \begin{bmatrix} -0.1 & 0.04 & 0.3 & 0.11 \\ 0.1 & 0.2 & 0.45 & 0.7 \end{bmatrix}$$

$$\boldsymbol{s}_2 = [0.1 \quad 0.2 \quad 0.45 \quad 0.7]$$

A neural IR model with kernels



Kernels

- Apply K Gaussian kernels to $\boldsymbol{s}_i$ (vector of similarity scores, corresponding to the ith query term):

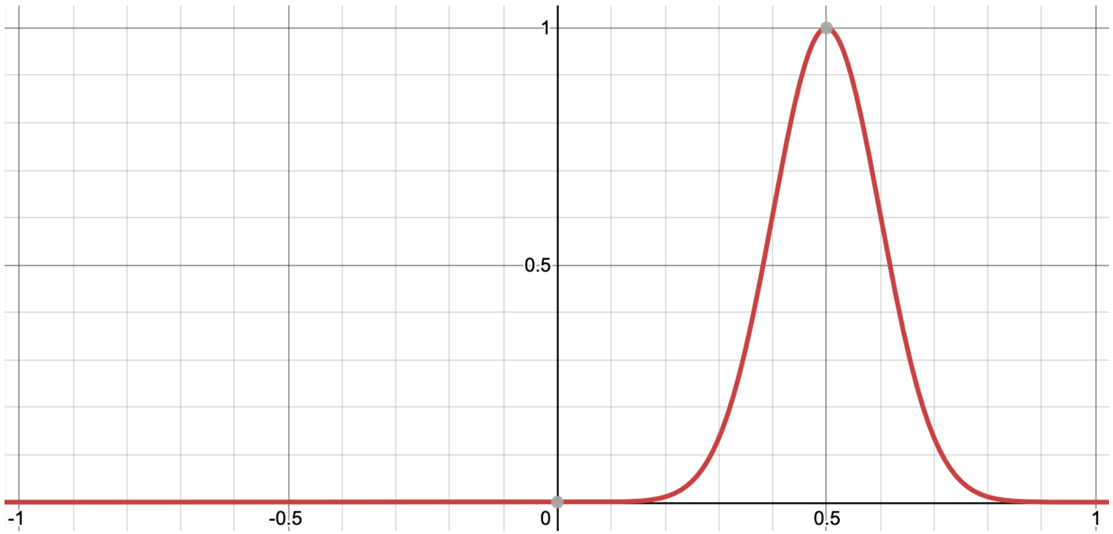
$$g_k(\boldsymbol{s}_i) = \sum_{j=1}^{m} e^{\left(-\frac{(s_{i,j}-\mu_k)^2}{2(\sigma_k)^2}\right)}$$

μ_k is mean and σ_k is standard deviation of the kth kernel (hyper-parameters)

- $g_k(\boldsymbol{s}_i)$ provides a soft term-frequency (Soft-TF) for the similarity values related to the ith query term

Kernels

The Gaussian kernel k with $\mu_k = 0.5$ and $\sigma_k = 0.1$: $e^{\left(-\frac{(x-0.5)^2}{2(0.1)^2}\right)}$



$$\boldsymbol{s}_2 = [0.1 \quad 0.2 \quad 0.45 \quad 0.7]$$

Results of applying this Gaussian kernel ($\mu_k = 0.5, \sigma_k = 0.1$) to $\boldsymbol{s}_2$:

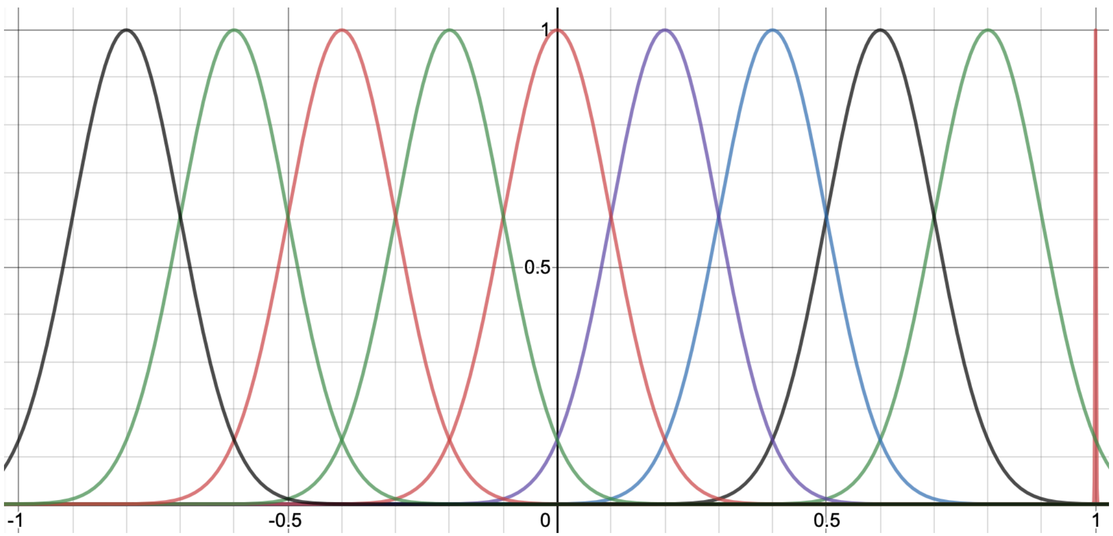
$$[0.00 \quad 0.01 \quad 0.88 \quad 0.13]$$

Sum of these values: $g_k(\boldsymbol{s}_2) = 1.02$

Kernels

$K = 10$ Gaussian kernels, each with a different mean value

- Standard deviations are the same: $\sigma = 0.1$, except for the last one: $\sigma_{10} \approx 0.0$



$$\boldsymbol{s}_2 = [0.1 \quad 0.2 \quad 0.45 \quad 0.7]$$

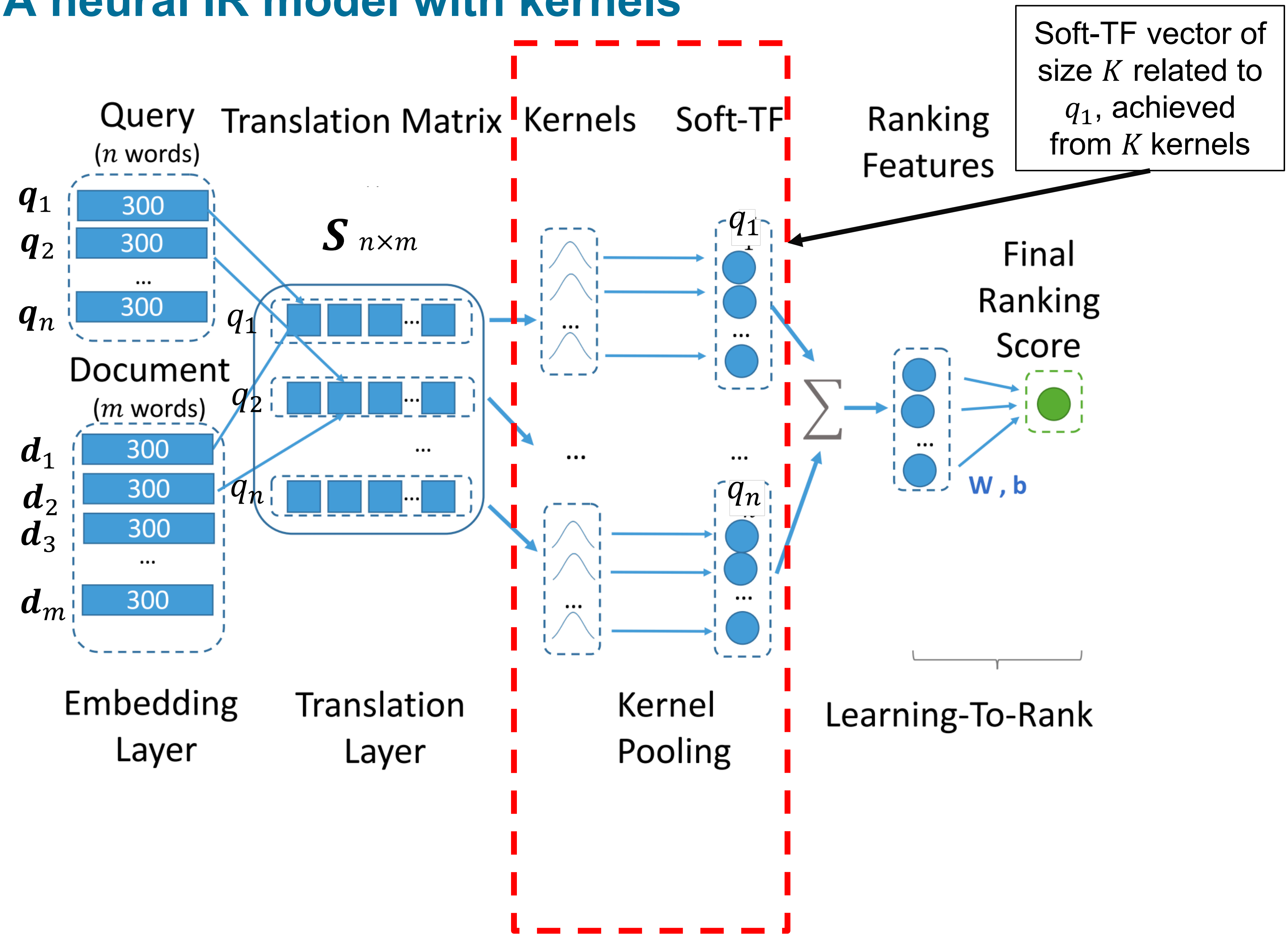
$$\mu_1 = -0.8 \rightarrow [0.00 \quad 0.00 \quad 0.00 \quad 0.00] \rightarrow g_1(\boldsymbol{s}_2) = 0.00$$

$$\mu_5 = 0.0 \rightarrow [0.60 \quad 0.13 \quad 0.00 \quad 0.00] \rightarrow g_5(\boldsymbol{s}_2) = 0.73$$

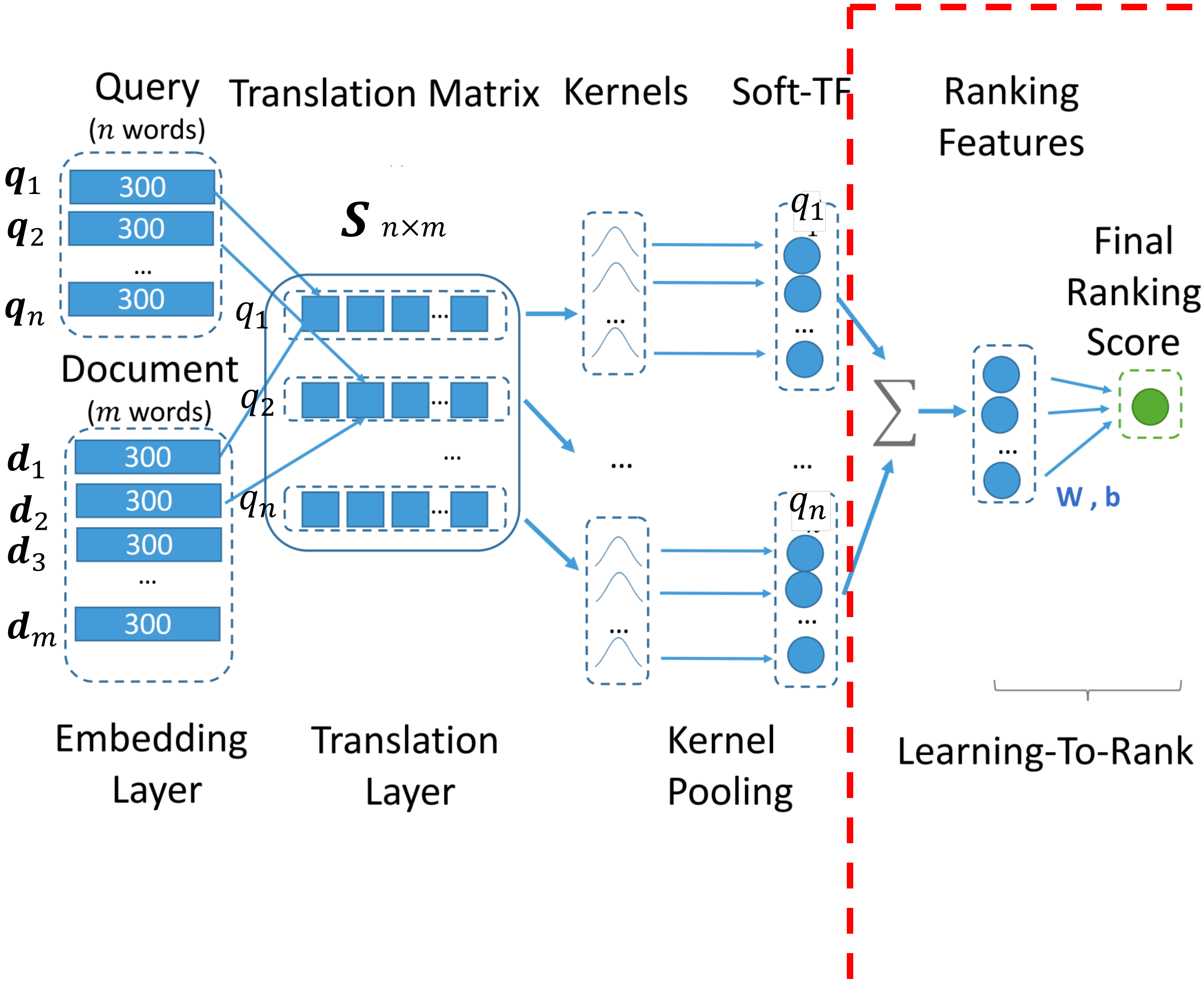
$$\mu_7 = 0.4 \rightarrow [0.01 \quad 0.13 \quad 0.88 \quad 0.01] \rightarrow g_7(\boldsymbol{s}_2) = 1.03$$

$$\mu_9 = 0.8 \rightarrow [0.00 \quad 0.00 \quad 0.00 \quad 0.60] \rightarrow g_9(\boldsymbol{s}_2) = 0.60$$

A neural IR model with kernels



A neural IR model with kernels



Feature vector and relevance score

- Final feature vector $\boldsymbol{v}$ with K values: for every kernel k, we sum over the results of all query terms:

$$v_k = \sum_{i=1}^{n} \log g_k(\boldsymbol{s}_i)$$

Logarithm smoothens Soft-TF values (as in the original TF)

- Final predicted relevance score is a linear transformation of $\boldsymbol{v}$

$$\text{score}(Q, D) = f(Q, D) = \boldsymbol{w}\boldsymbol{v} + b$$

Elements of neural IR models

- Model architecture
- **Loss function and training**
- Inference process (at query time)

Collection for Training

- MS MARCO (Microsoft MAchine Reading Comprehension)
  - Commonly used collection for training large-scale neural IR models
  - Queries and retrieved passages of BING, annotated by human

	MS MARCO [28]
# of documents	8,841,822
Average document length	58.8 ± 23.5
Average query length	6.3 ± 2.6
# of training data points	39,780,811
# of validation queries	6,980
# of test queries	48,598

- Training data is provided in the form of triples:

```
[query, a relevant document, a non-relevant document]
```

$$[Q, D^+, D^-]$$

https://microsoft.github.io/msmarco/

Training with a pair-wise objective

- Pair-wise learning-to-rank optimizes the network such that the predicted relevance scores of a query to a relevant document be higher than the one to a non-relevant document

- Given a training data point $[Q, D^+, D^-]$, a pair-wise objective aims to satisfy the criterion:

$$f(Q, D^+) > f(Q, D^-)$$

- This means that the IR model learns to give a higher relevance score to D^+, and therefore rank D^+ in a higher position than D^-. This (hopefully) leads to a better overall ranking results for the given query.

Margin Ranking loss

- A widely used loss function for pair-wise training
- Also called *Hinge loss*, *contrastive loss*, *max-margin objective*

- Margin ranking loss "punishes" the network until a margin C is held between the predicted scores for the relevant and non-relevant documents:

$$\mathcal{L} = \mathbb{E}_{(Q,D^+,D^-)\sim\mathcal{D}}[\max(0, C - (f(Q,D^+) - f(Q,D^-)))]$$

Examples when $C = 1$:

If $f(Q,D^+) = 2$ and $f(Q,D^-) = 1.8 \rightarrow \mathcal{L} = 0.8$

If $f(Q,D^+) = 2$ and $f(Q,D^-) = 3.8 \rightarrow \mathcal{L} = 2.8$

If $f(Q,D^+) = 2$ and $f(Q,D^-) = 0.8 \rightarrow \mathcal{L} = 0.0$

Elements of neural IR models

- Model architecture
- Loss function and training
- **Inference process (at query time)**

Inference process at query time (Validation/Test)

- Since neural IR models are based on soft matching (semantically similar terms are also involved), we can’t simply use inverted index!
  - Retrieval will not be so efficient as the classical IR models

What do we do then?!

- Two common but non-optimal approaches:
  - Full-ranking: given a query, calculate relevance scores for all documents
    - Very expensive!
  - Re-ranking: re-rank top-t results of another IR model
    1. Pass the query to an efficient IR model (e.g. BM25) and retrieve a first set of documents
    2. Select the top-t documents of this first set
    3. Re-calculate relevance scores for these documents using the neural IR model
    4. Update the original ranking results by re-ordering (re-ranking) the top-t documents based on the newly calculated scores
- An active area of research!

Summary

- Ad-hoc retrieval components
  - Exact-matching IR models
  - Collection index
- Evaluation of IR models
  - Position in the ranking list matters!
- Neural IR models
  - An interaction-focused IR model with Gaussian kernels