Natural Language Processing with Deep Learning
Neural Information Retrieval

Navid Rekab-Saz
navid.rekabsaz@jku.at

Institute of Computational Perception

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

Agenda

- Information Retrieval Crash course
- Neural Ranking Models

Agenda

- **Information Retrieval Crash course**
- Neural Ranking Models

Some slides are adopted from Stanford's *Information Retrieval and Web Search* course http://web.stanford.edu/class/cs276/

Information Retrieval

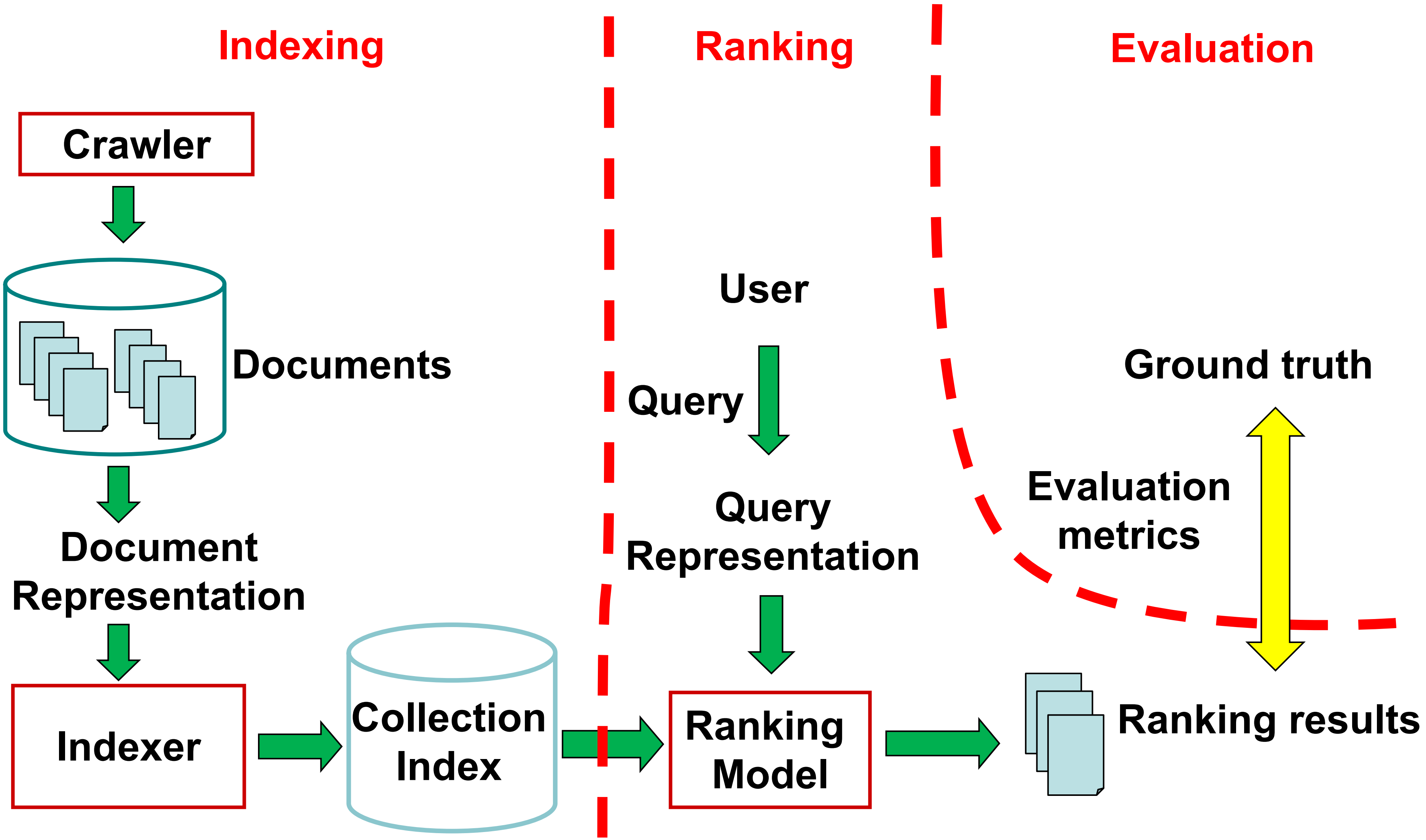
- Information Retrieval (IR) is finding material (usually in the form of documents) of an unstructured nature that satisfies an information need from within large collections

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

- The goal of IR is however to retrieve documents that contain relevant content to the user's information need

- So IR covers a wide set of tasks such as …
  - Ranking, factual/non-factual Q&A, information summarization
  - But also … user behavior/experience study, personalization, etc.

Components of an IR System (simplified)



Essential Components of Information Retrieval

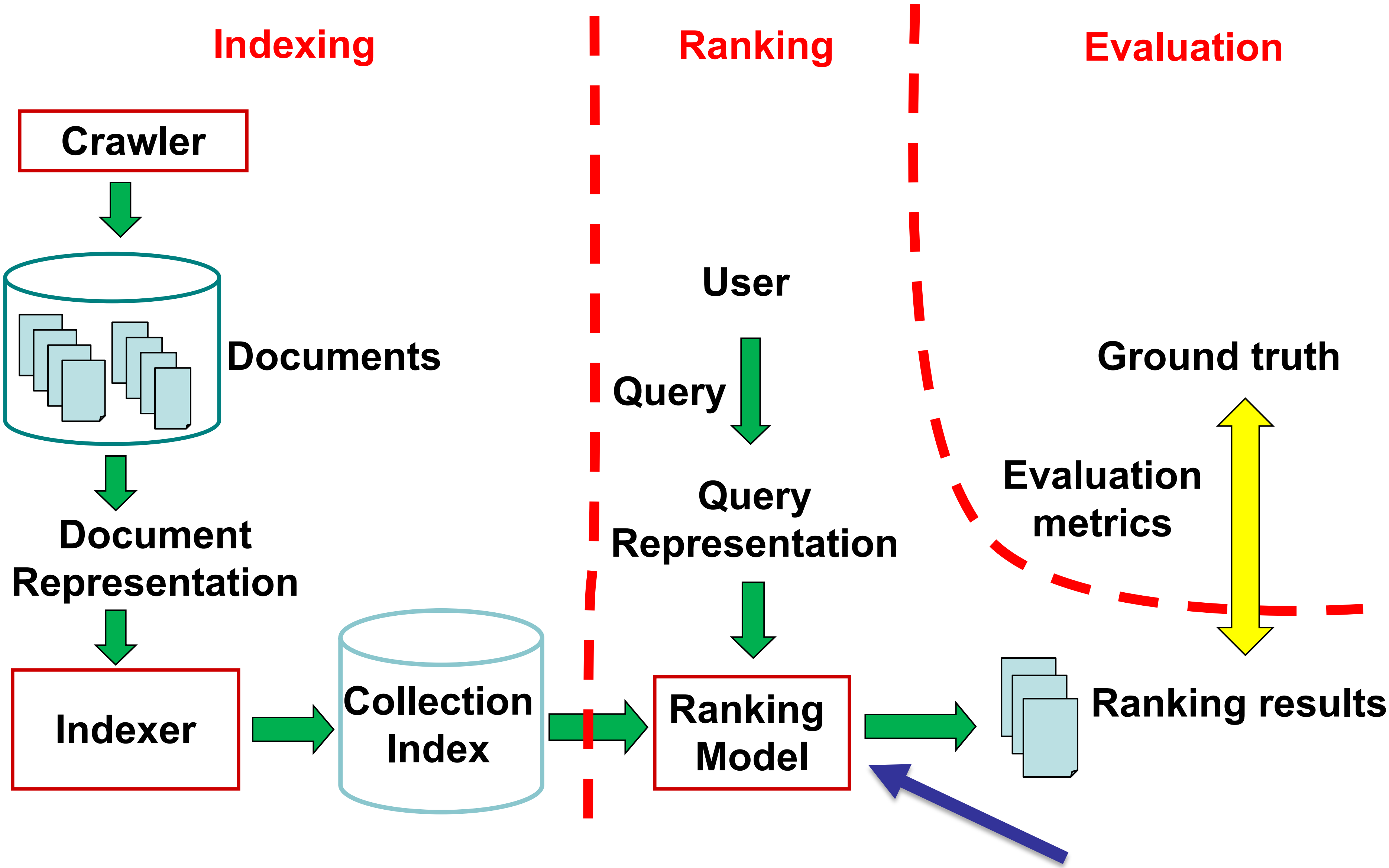
- Information need
  - E.g. *My swimming pool bottom is becoming black and needs to be cleaned*
- Query
  - A designed representation of users' information need
  - E.g. *pool cleaner*
- Document
  - A unit of data in text, image, video, audio, etc.
- Relevance
  - Whether a document satisfies user's information need
  - Relevance has multiple perspectives: 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 ignored
  - The focus is on methods to exploit contents

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

Components of an IR System (simplified)



Ranking Model / IR model

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:

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

Classical IR models – TF-IDF

- Classical IR models (in their basic forms) are based on exact term matching

- Recap: we used TF-IDF as term weighting for document classification
- TF-IDF is also a well-known IR model:

$$\mathrm{score}(Q,D) = \sum_{t \in Q} \mathrm{tf}\,(t,D) \times \mathrm{idf}\,(t) = \sum_{t \in Q} \underbrace{\boxed{\log\left(1+\mathrm{tc}_{t,D}\right)}}_{\text{Term matching score \& normalization}} \times \underbrace{\boxed{\log\left({}^{|\mathbb{D}|}/_{\mathrm{df}_t}\right)}}_{\text{Term Salience}}$$

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

df_t number of documents in which term t appears

Classical IR models – PL

- Pivoted Length Normalization model

Term matching score

$$\mathrm{score}(Q,D)=\sum_{t\in Q}\frac{\boxed{\log\left(1+\mathrm{tc}_{t,D}\right)}}{\boxed{1-b+b\,\frac{|D|}{avgdl}}}\times\boxed{\mathrm{idf}\,(t)}$$

Term Salience

Length normalization

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

$avgdl$ average length of the documents in the collection

b a hyper parameter that controls length normalization

Classical IR models – BM25

- BM25 model *(slightly simplified)*:

Term matching score & normalization

$$\text{score}(Q,D) = \sum_{t \in Q} \frac{(k_1+1)\text{tc}_{t,D}}{k_1\left(1-b+b\frac{|D|}{avgdl}\right)+\text{tc}_{t,D}} \times \text{idf}\,(t)$$

Length normalization

Term Salience

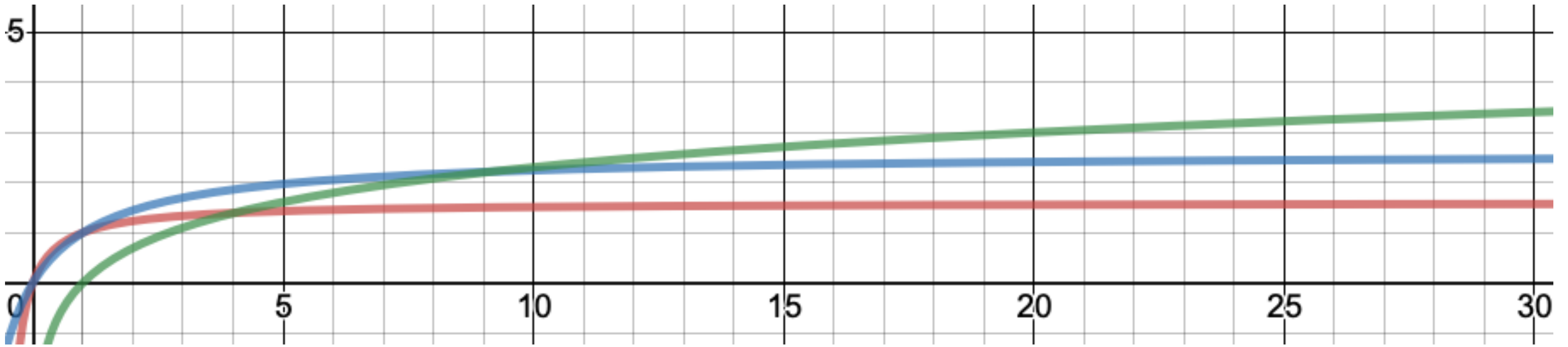
$\text{tc}_{t,D}$ number of times term t appears in document D

$avgdl$ 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*

Classical IR models – BM25

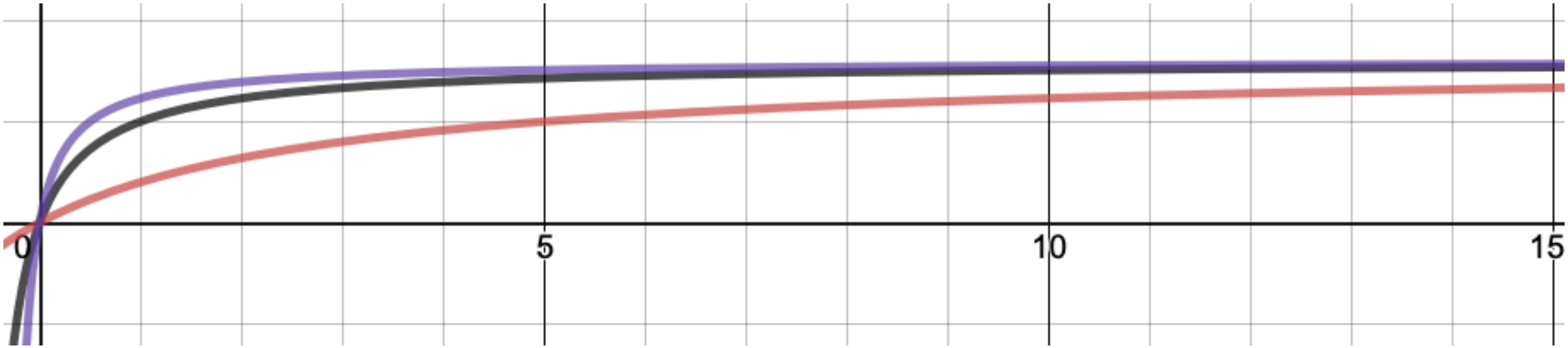


Green: $\log\ \mathrm{tc}_{t,D}$ → TF

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

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

Classical IR models – BM25



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

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

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

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

Scoring & Ranking

query (Q): wisdom of mountains

$D20$

$D1402$

$D5$

$D100$

Documents are sorted based on the predicted relevance scores from high to low

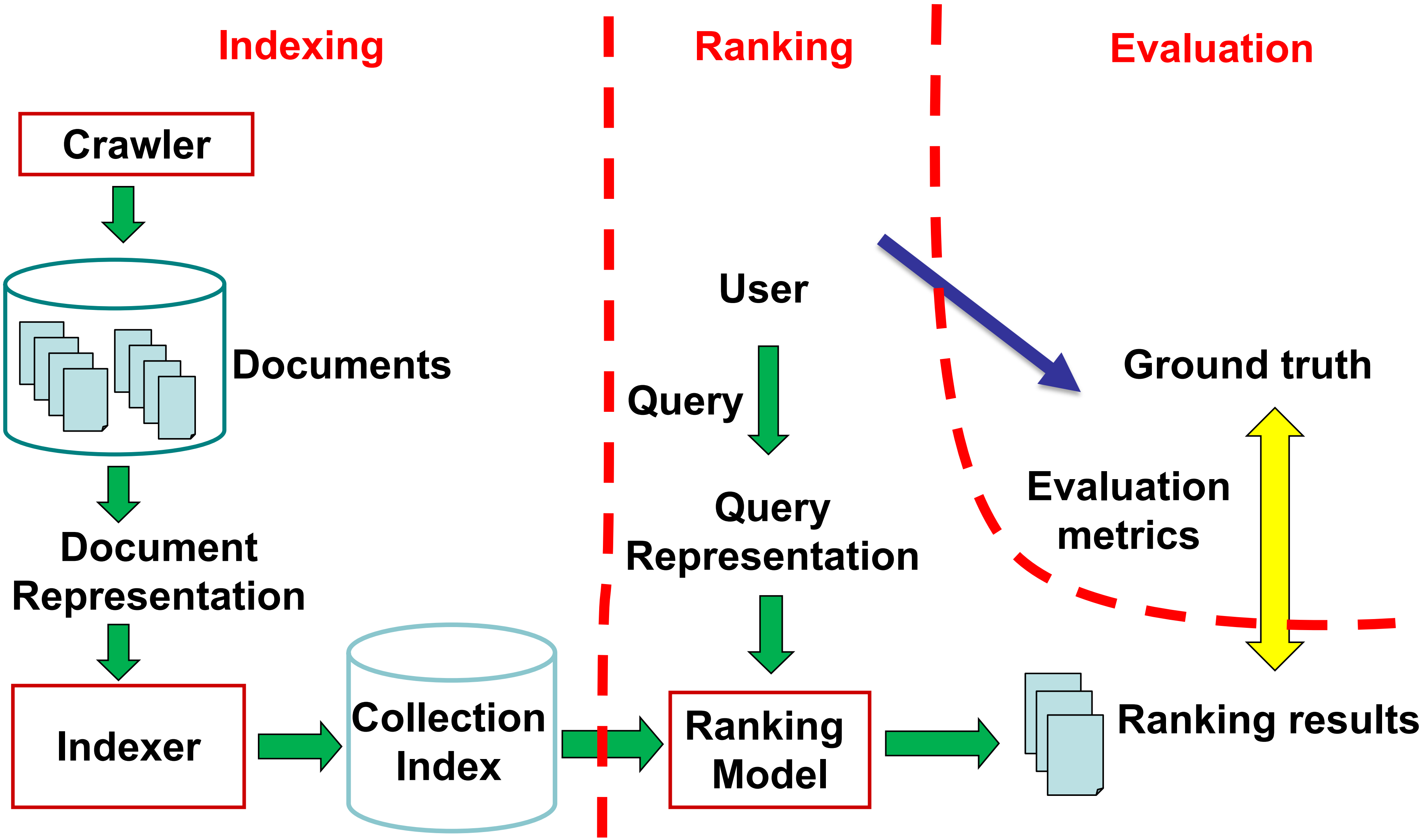
Scoring & Ranking

- TREC run file: standard text format for ranking results of IR models

```
qry_id iter(ignored) 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
…
```

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
  - An assessment for each query and each document
    - Assessment specifies whether the document addresses the underlying information need
    - Ideally done by human, but also through user interactions
    - Assessments are called ground truth or relevance judgements and are provided in …
      - 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)

Scoring & Ranking

- TREC qrel file: a standard text format for relevance judgements of some queries and documents

```
qry_id iter(ignored) 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 and Recall

- **Precision**: fraction of retrieved docs that are relevant
- **Recall**: fraction of relevant docs that are retrieved

	Relevant	Nonrelevant
Retrieved	TP	FP
Not Retrieved	FN	TN

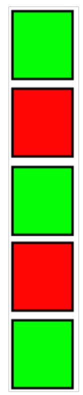
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision@*n*

- Given the ranking results of a query, compute the percentage of relevant documents in top *n* results

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



- Calculate the mean P across all test queries

- In similar fashion we have Recall@*n*

Mean Reciprocal Rank (MRR)

- MRR supposes that users are only looking for one relevant document
  - looking for a fact
  - known-item search
  - navigational queries
  - query auto completion

- Consider the rank position K of the first relevant document

$$\text{Reciprocal Rank (RR)} = \frac{1}{K}$$

- MRR is the mean RR across all test queries

Rank positions matter!

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

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**.

Excellent

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 ...

Fair

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.

Bad

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**.

Good

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...

Fair

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 ...

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 (graded 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 (*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 position K is:

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

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

- Alternative formulation (commonly used):

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

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 rank 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

Evaluation Campaigns

- Text REtrieval Conference (TREC)

Text REtrieval Conference (TREC)

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

https://trec.nist.gov

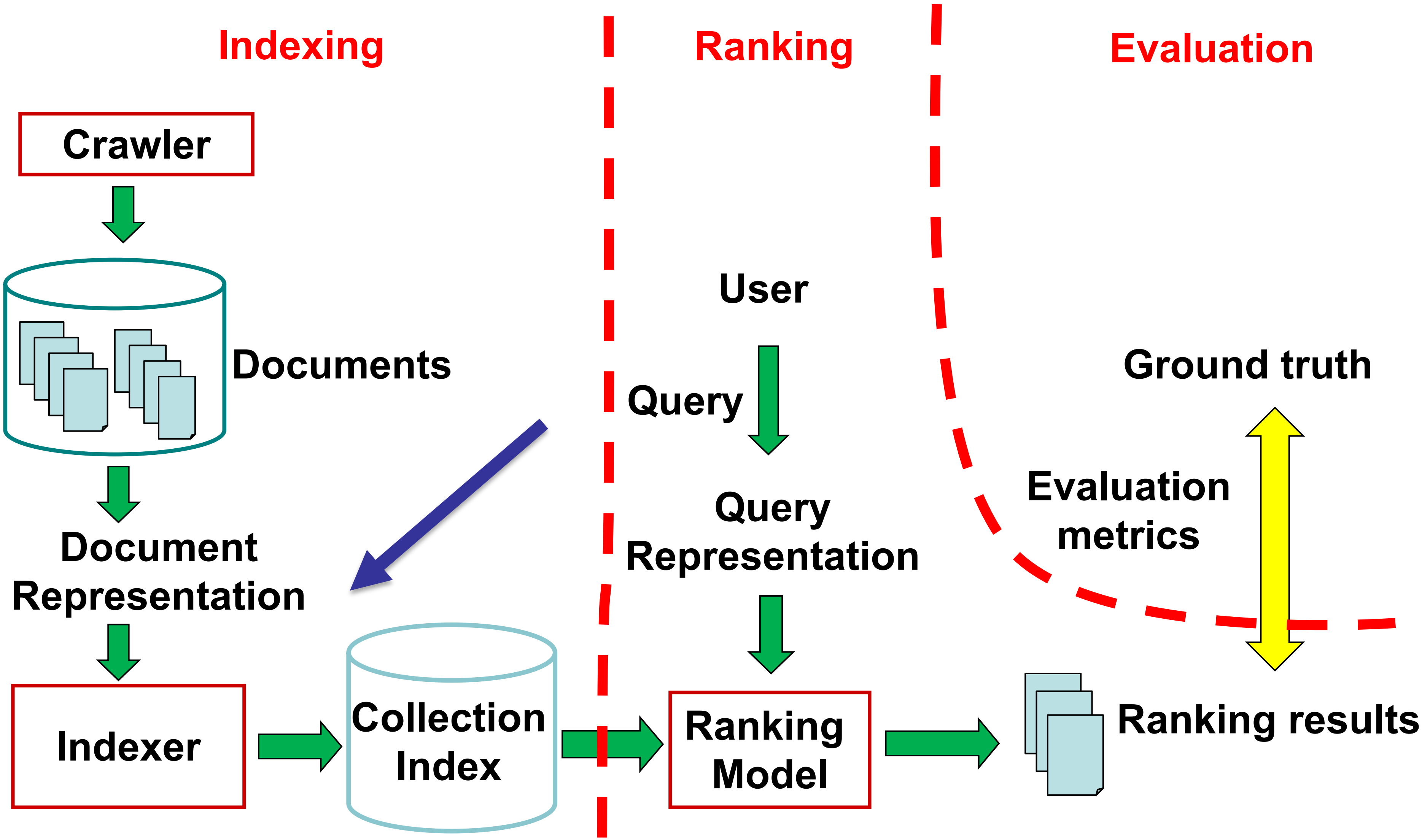
- Conference and Labs of the Evaluation Forum (CLEF)

http://www.clef-initiative.eu

- MediaEval Benchmarking Initiative for Multimedia Evaluation

http://www.multimediaeval.org

Components of an IR System (simplified)



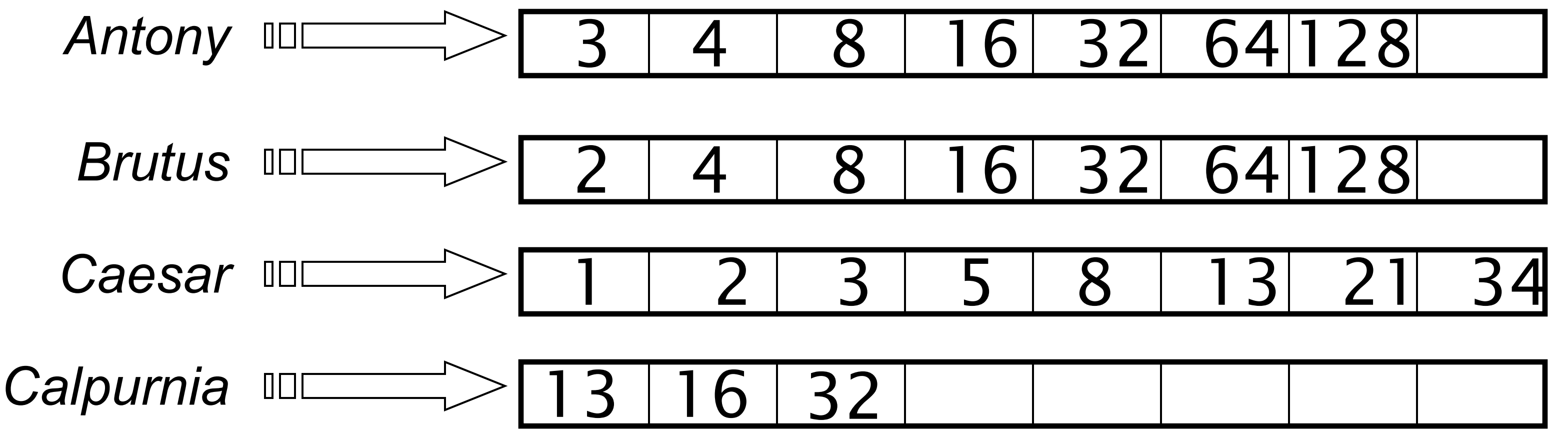
Inverted index

- Inverted index is a data structure for efficient document retrieval
- Inverted index consists of posting lists of terms
- A posting list contains the IDs of the documents in which the term appears

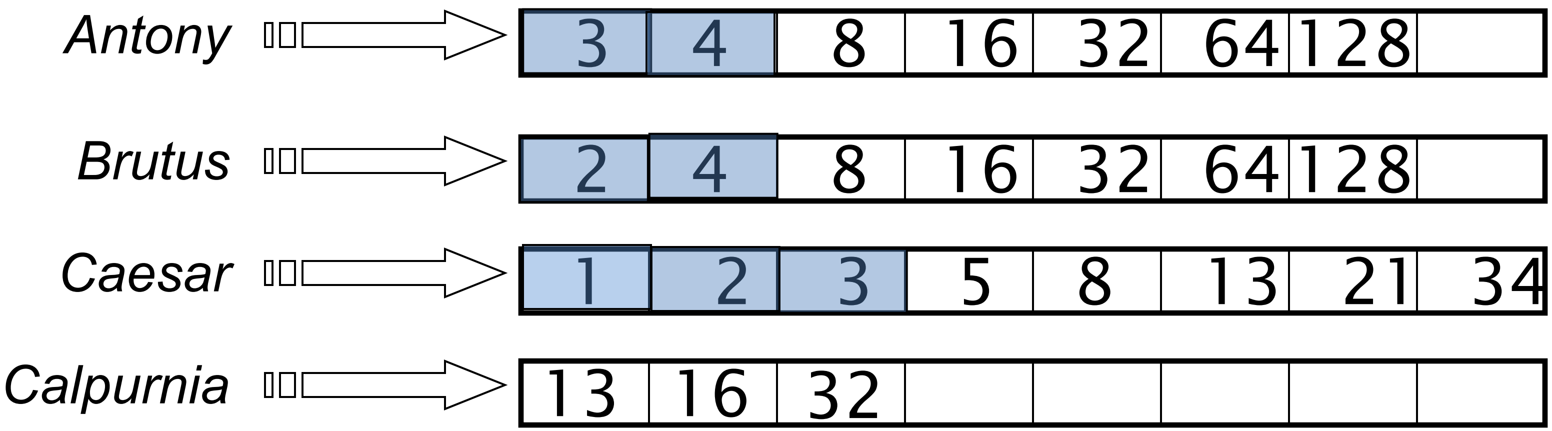
Antony	3	4	8	16	32	64	128	
Brutus	2	4	8	16	32	64	128	
Caesar	1	2	3	5	8	13	21	34
Calpurnia	13	16	32					

Search with inverted index

1. Fetch 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 *K* documents with the highest relevance scores



Search with concurrent traversal



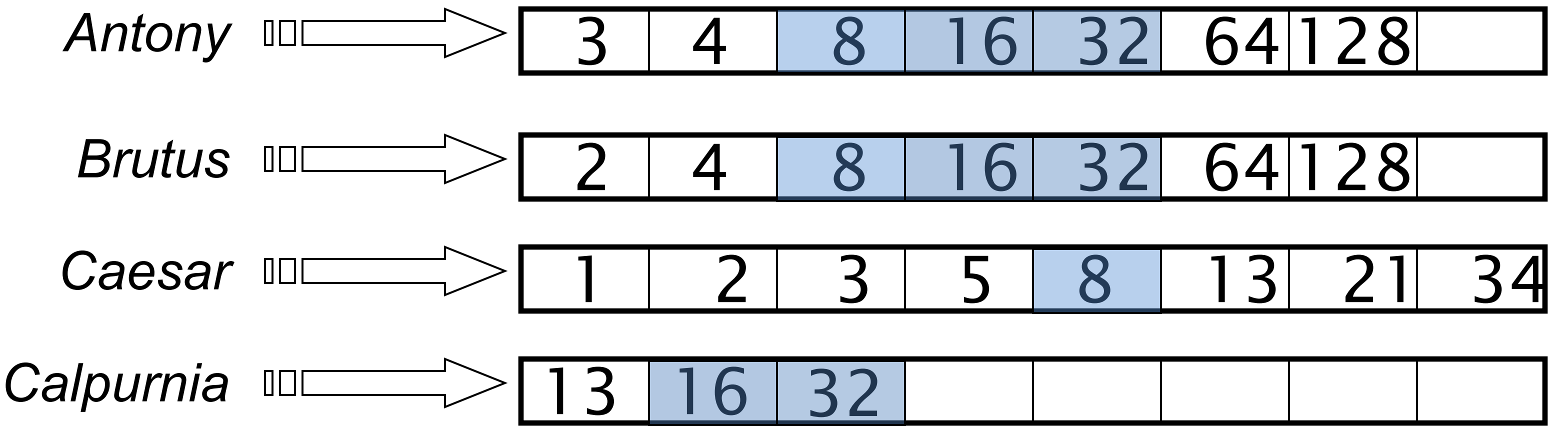
More efficient search – inexact top *K* retrieval

- Instead of processing all the documents in the posting lists, find top *K* documents that are likely to be among the top *K* documents with exact search

- For the sake of efficiency!

- One sample approach: only process the documents, containing *several* query terms

Inexact top *K* retrieval



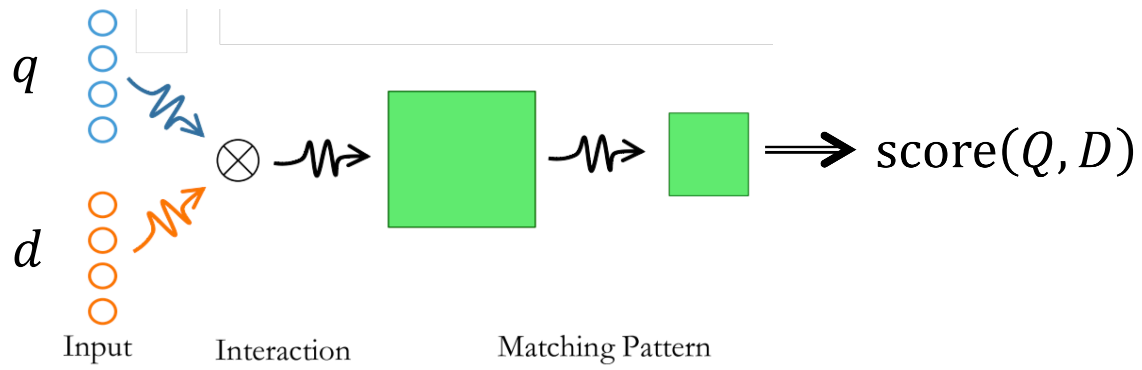
Scores only computed for docs 8, 16 and 32.

Agenda

- Information Retrieval Crash course
- **Neural Ranking Models**

Neural Ranking models

- Instead of a ranking formula, we can train a neural ranking model to calculate $\mathrm{score}(Q, D)$



- Neural ranking models benefit from semantic relations or soft matching (vs. 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).

Learning to Rank

- The Learning problem in ranking models:
  - Given a query, the model learns to provide a *good* ranking of documents: Learning to Rank

- Three families of Learning to Rank models:
  - Point-wise,
  - Pair-wise,
  - List-wise

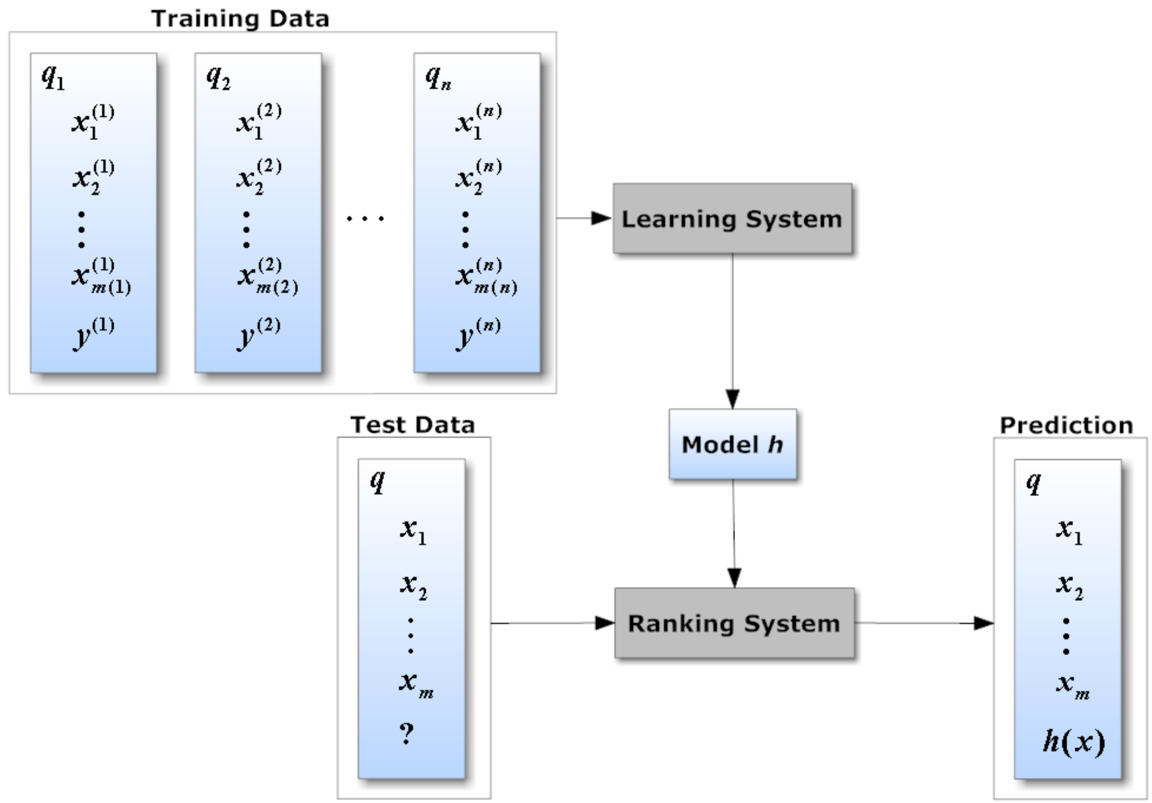
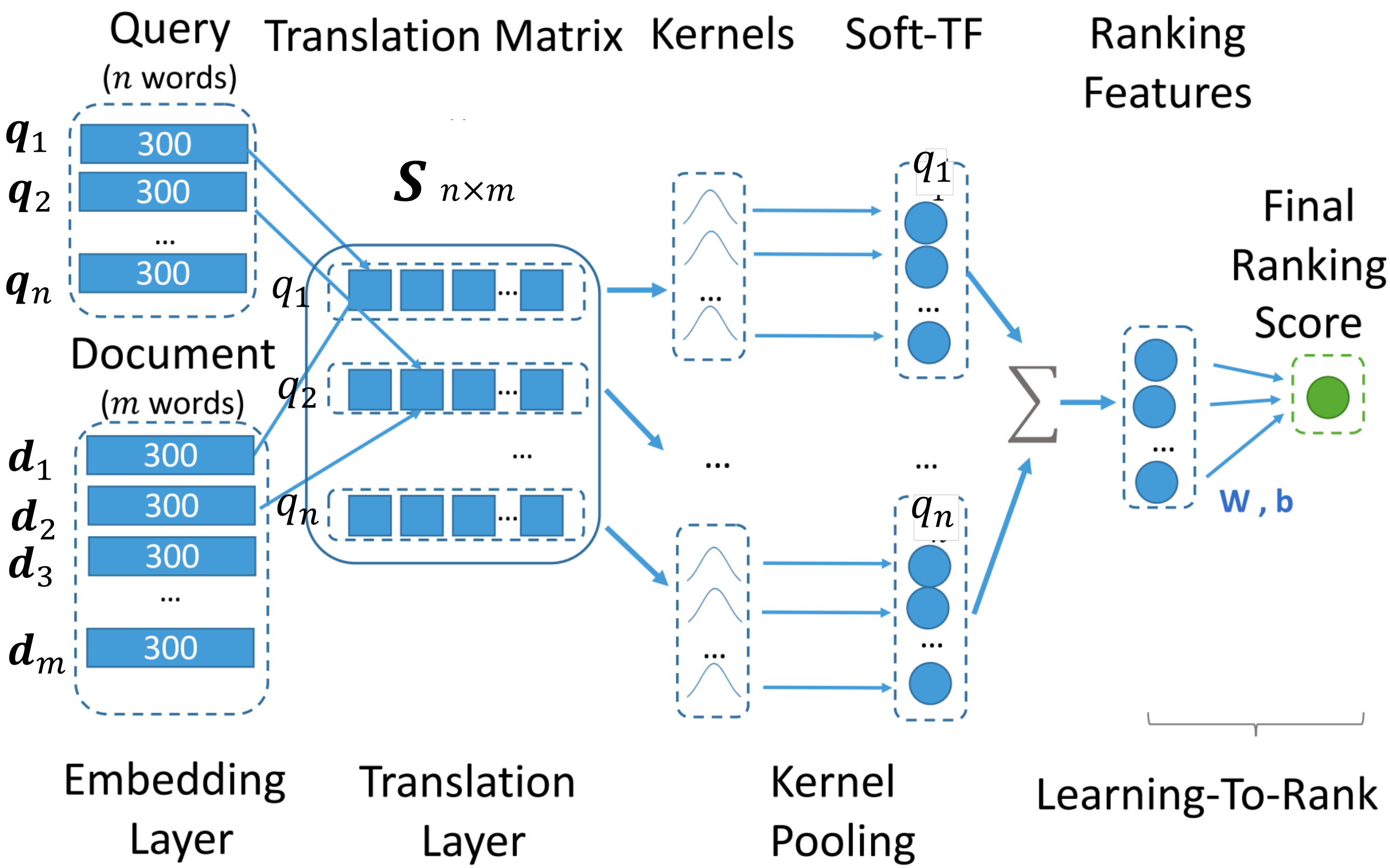


Image source: https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418

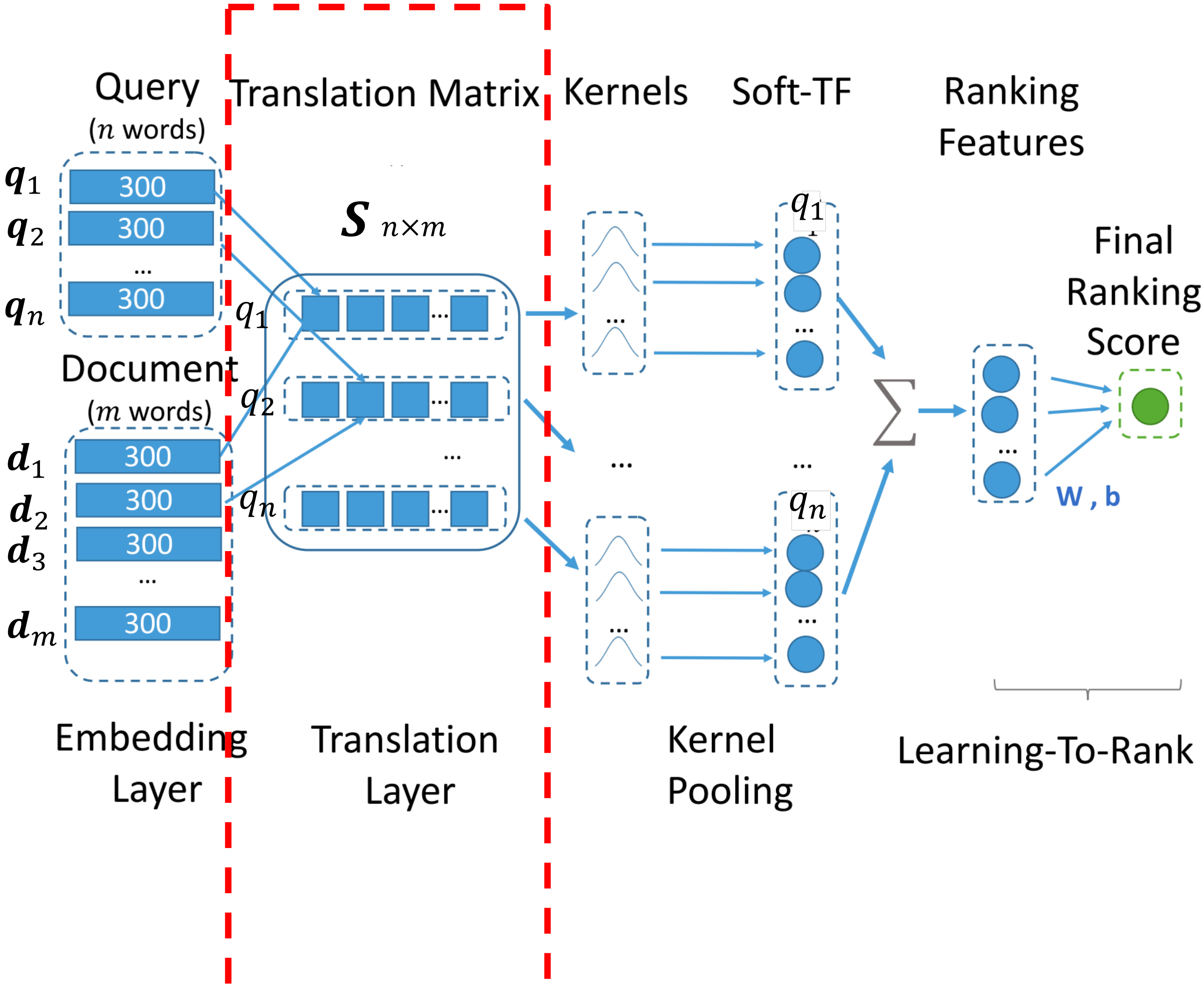
A sample neural ranking model



Kernel-based Neural Ranking Model (K-NRM)

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

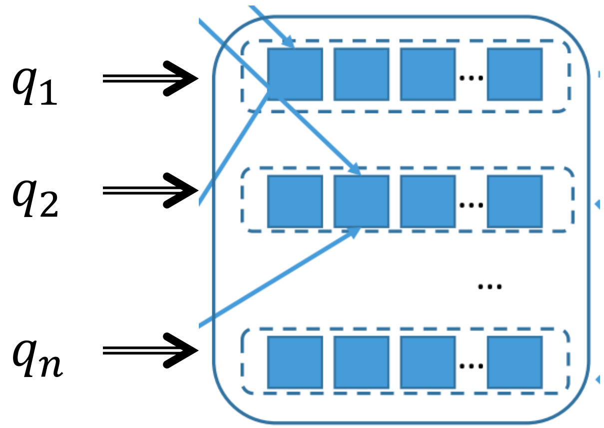
KNRM


Query
(n words)
q_1
q_2
q_n
300
300
...
300
Document
(m words)
d_1
d_2
d_3
d_m
300
300
300
...
300
Translation Matrix
$S_{n \times m}$
q_1
q_2
q_n
Kernels
Soft-TF
q_1
q_n
Ranking
Features
$\sum$
Final
Ranking
Score
W , b
Embedding
Layer
Translation
Layer
Kernel
Pooling
Learning-To-Rank

KNRM – Translation Matrix

- n query terms and m document terms
- Embedding of ith query term $\boldsymbol{q}_i$
- Embedding of jth document term $\boldsymbol{d}_j$
- Term-to-Term similarity scores:

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

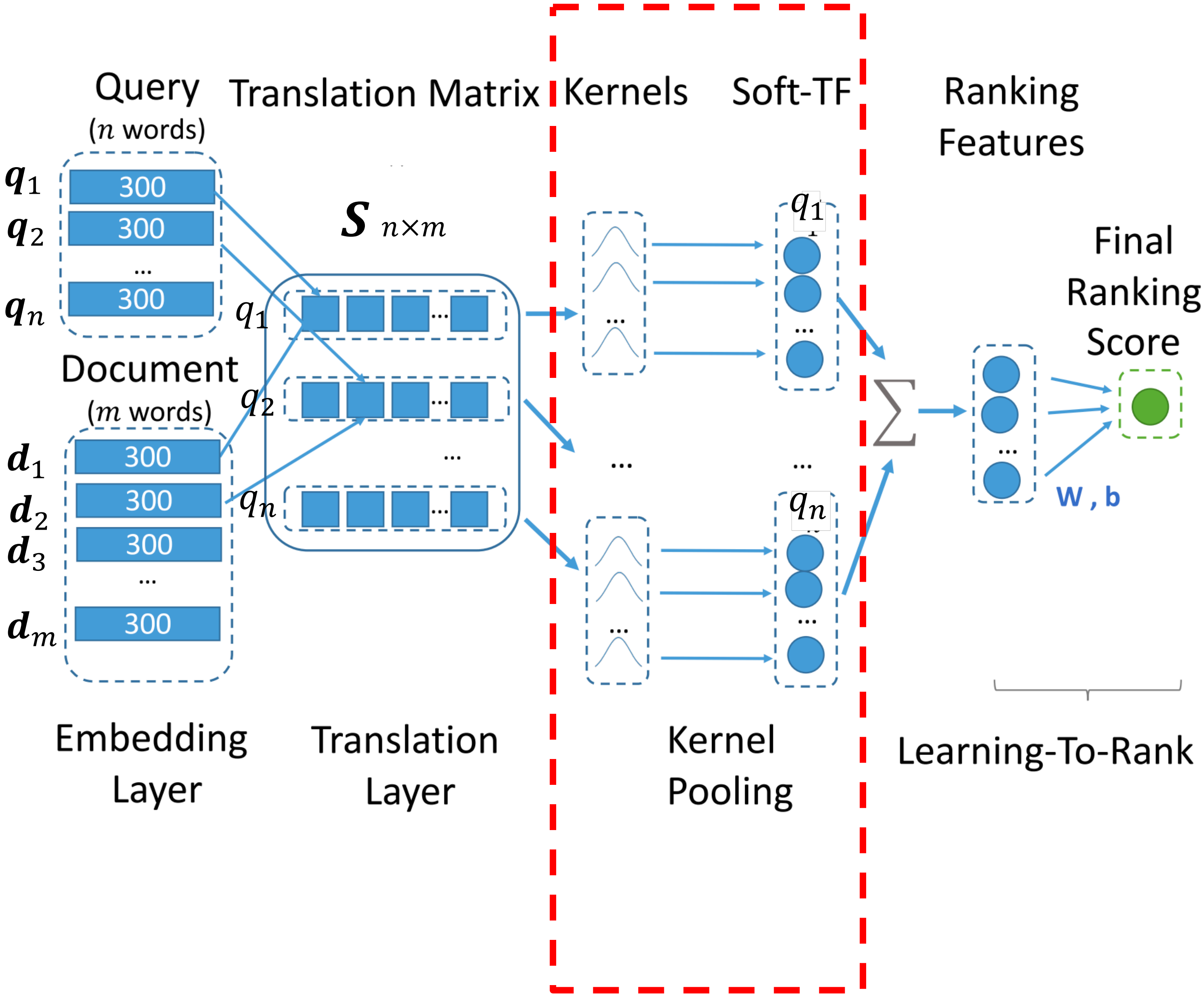


Matrix $\boldsymbol{S}$ with
n rows (queries) and
m columns (documents)

An example of a vector of similarity scores for q_i, denoted as $\boldsymbol{s}_i$:

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

KNRM


Query
(n words)
q_1
q_2
q_n
300
300
...
300
Document
(m words)
d_1
d_2
d_3
d_m
300
300
300
...
300
Translation Matrix
$S_{n \times m}$
q_1
q_2
q_n
Kernels
Soft-TF
Ranking
Features
Final
Ranking
Score
W , b
Embedding
Layer
Translation
Layer
Kernel
Pooling
Learning-To-Rank

KNRM – Kernels

- Apply k Gaussian kernels to the vector of similarity scores, corresponding to a query term:

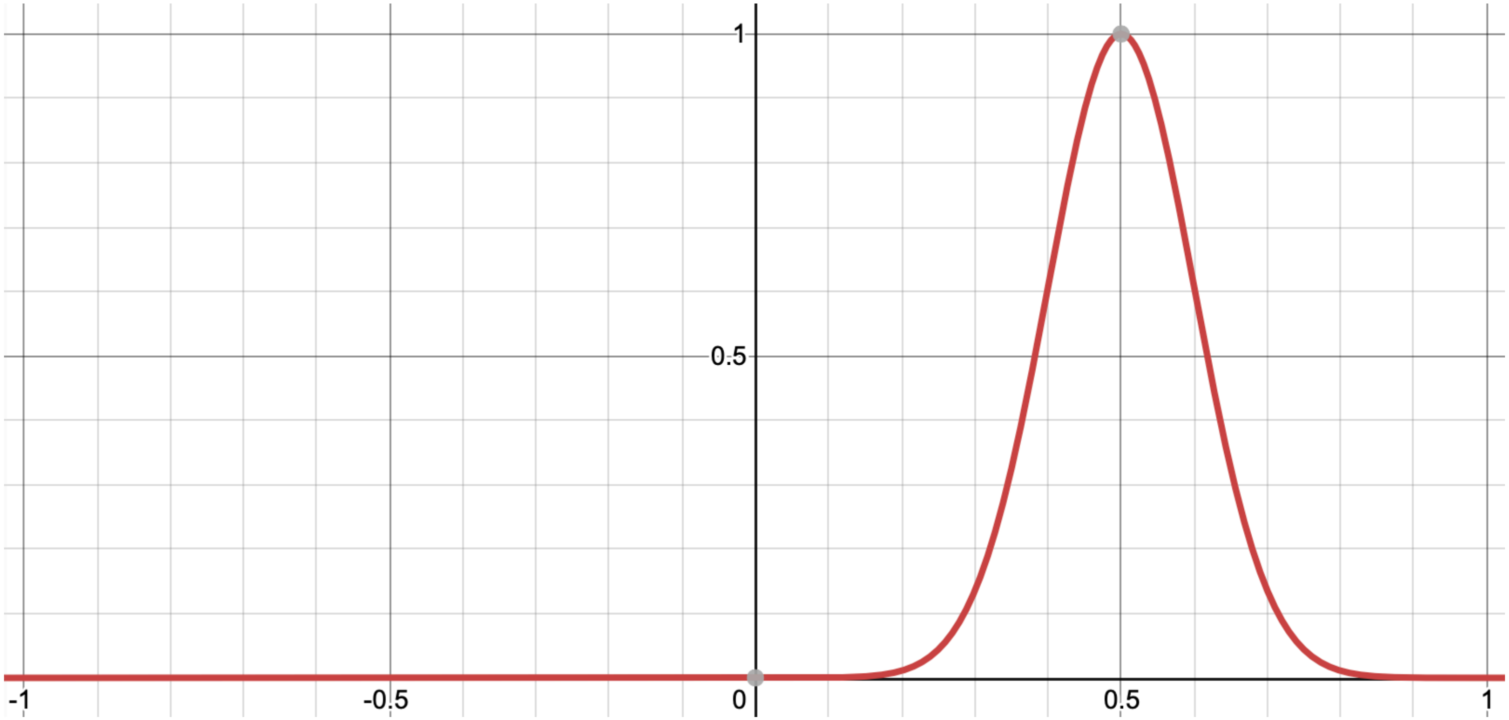
$$K_k(\boldsymbol{s}_i) = \sum_{j=1}^{m} e^{\left(-\frac{(s_{i,j}-\mu_k)^2}{2(\sigma_k)^2}\right)}$$

μ_k and σ_k are mean and standard deviation of the kth kernel, set as hyper-parameters

- Each kernel result $K_k(\boldsymbol{s}_i)$ is a soft term count for q_i
  - $K_k(\boldsymbol{s}_i)$ is the sum of the results of applying a Gaussian function with mean and std μ_k and σ_k to the similarity scores

KNRM – Kernels

A Gaussian kernel at $\mu_k = 0.5$ and $\sigma_k = 0.1$: $e^{(-\frac{(x-0.5)^2}{2(0.1)^2})}$



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

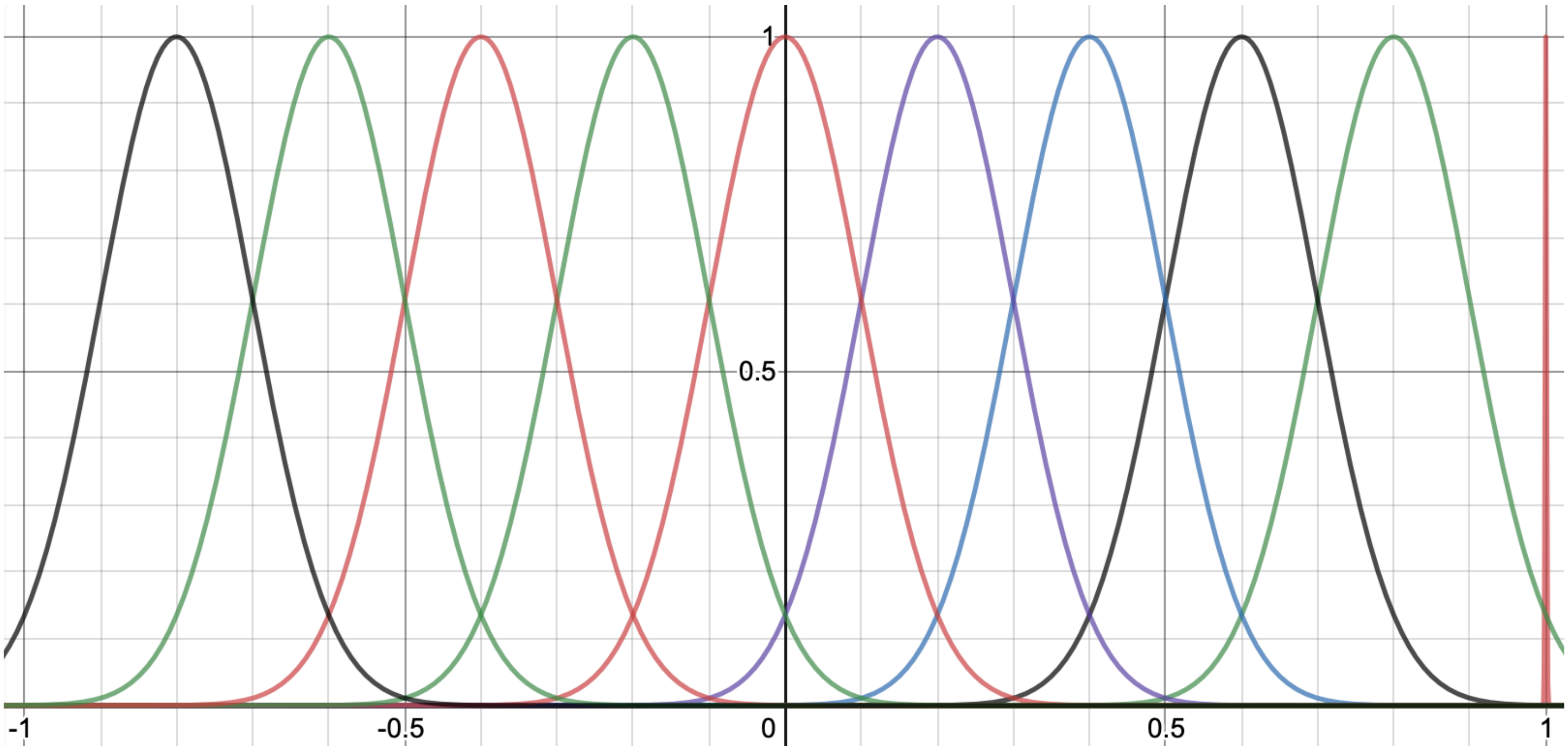
Applying the kernel $\rightarrow [0.011 \quad 0.882 \quad 0.135 \quad 0.0]$

$$K_k(\boldsymbol{s}_i) = 1.028$$

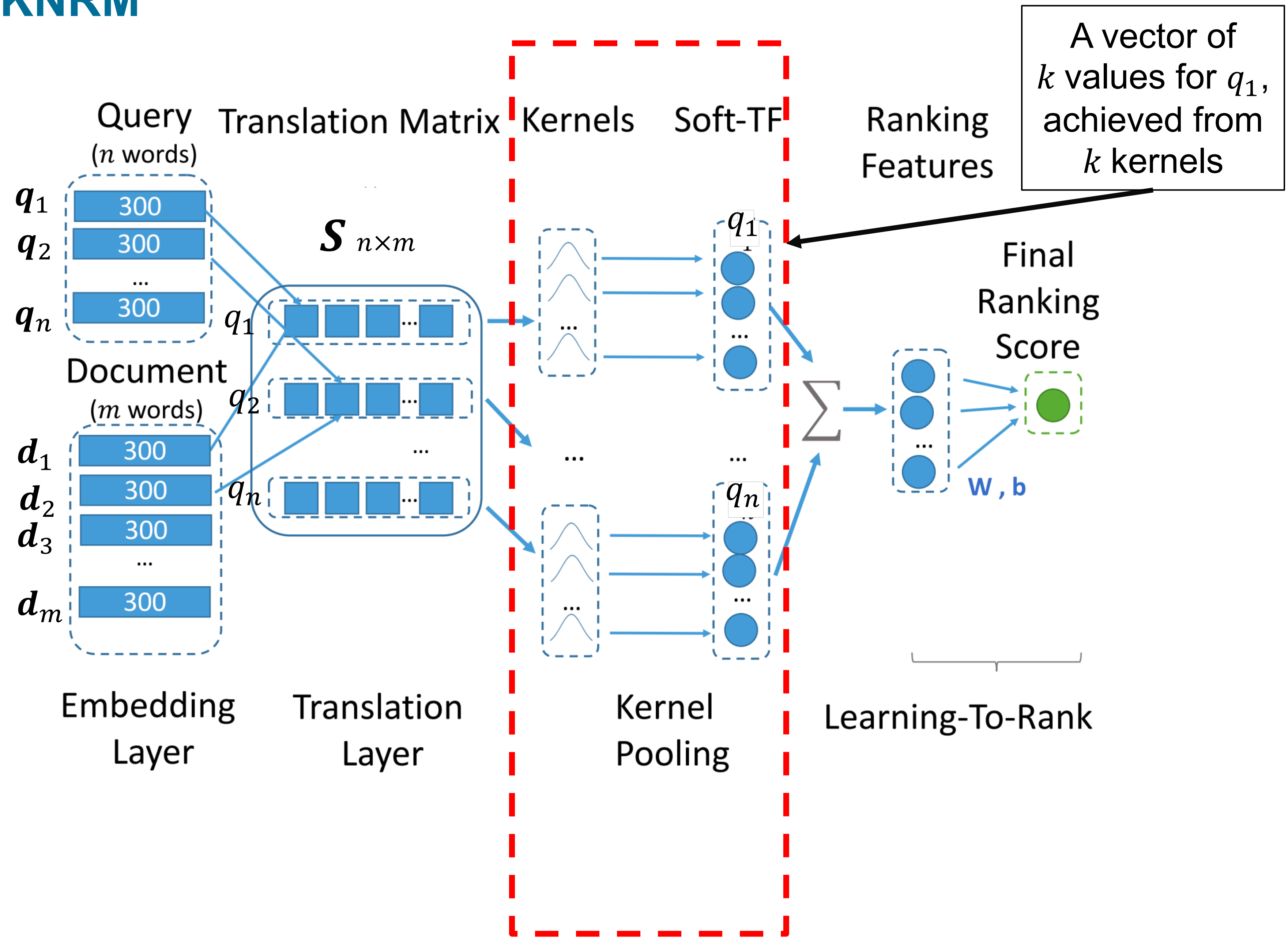
$K_k(\boldsymbol{s}_i)$ is a soft term count for similarity scores of q_i

KNRM – Kernels

k Gaussian kernels with different mean values μ_k. Standard deviation of all are the same $\sigma_k = 0.1$

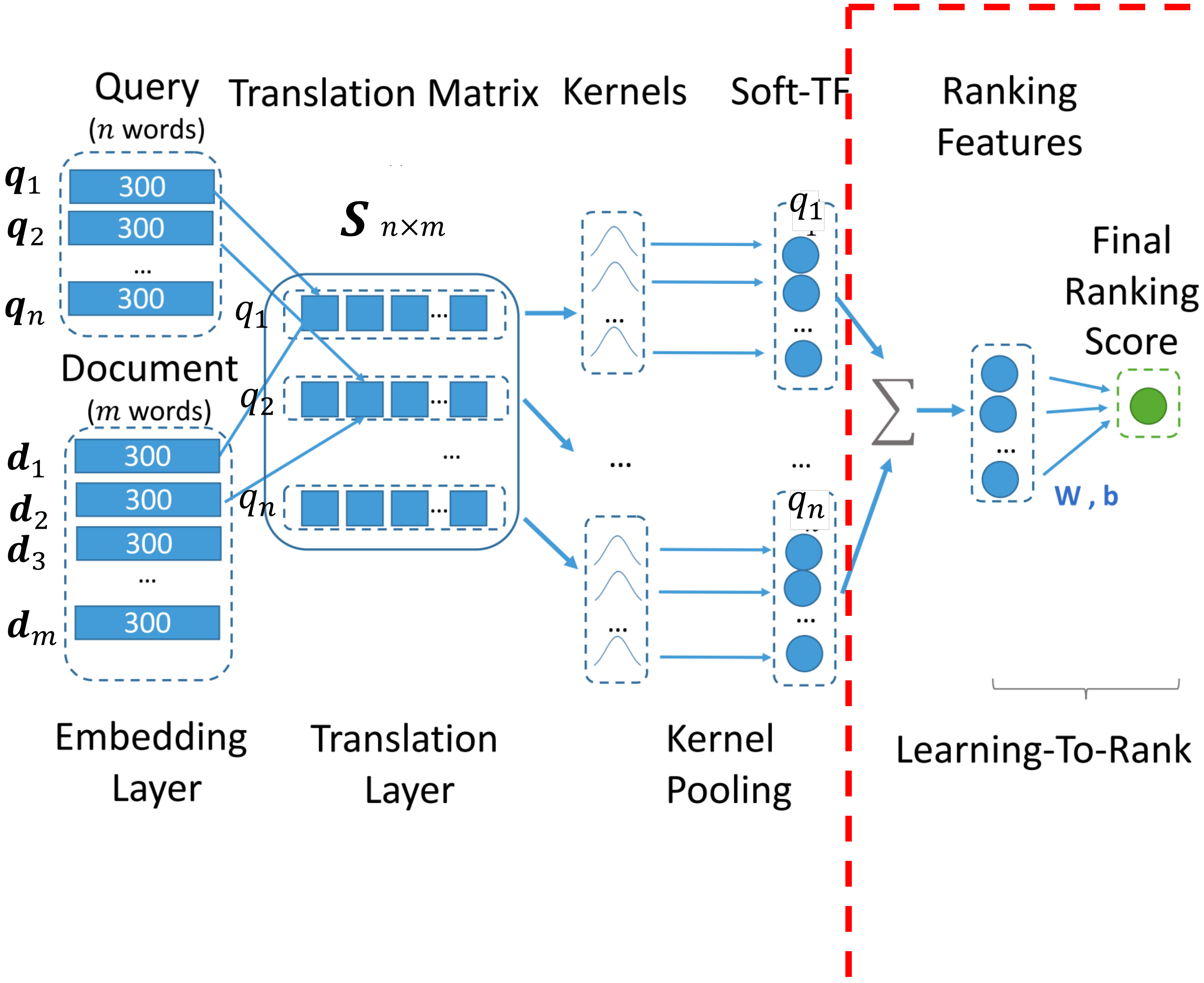


KNRM



A vector of k values for q_1, achieved from k kernels
Query (n words)
Translation Matrix
Kernels
Soft-TF
Ranking Features
Final Ranking Score
q_1
q_2
q_n
300
300
...
300
Document (m words)
d_1
d_2
d_3
d_m
300
300
300
...
300
$S_{n \times m}$
q_1
q_2
q_n
W , b
Embedding Layer
Translation Layer
Kernel Pooling
Learning-To-Rank

KNRM



Query
(n words)
Translation Matrix
Kernels
Soft-TF
Ranking Features
q_1
q_2
q_n
300
300
300
$S_{n\times m}$
Document
(m words)
d_1
d_2
d_3
d_m
300
300
300
300
q_1
q_2
q_n
Final Ranking Score
W , b
Embedding Layer
Translation Layer
Kernel Pooling
Learning-To-Rank

KNRM – Features and final relevance score

- Feature vector $\boldsymbol{v}$ with k values. Each value v_k corresponds to the sum of the results of all queries on one kernel:

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

Logarithm normalizes soft term count (similar to TF) → it is therefore called soft-TF

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

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

Collection for Training

- MS MARCO (Microsoft MAchine Reading Comprehension)
- 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 in the form of triples:

`(query, a relevant document, a non-relevant document)`

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

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

Training

- Training data provides relevant but also non-relevant judgements → pair-wise training

- Margin Ranking loss
  - A widely used loss function for pair-wise training
  - Also called *Hinge loss*, *contrastive loss*, *max-margin objective*
  - It "punishes" until a margin C is held between the scores

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

Example: For $C = 1$,
If $f(Q,D^+) = 2$ and $f(Q,D^-) = 1.8$ then loss is 0.8
If $f(Q,D^+) = 2$ and $f(Q,D^-) = 3.8$ then loss is 2.8
If $f(Q,D^+) = 2$ and $f(Q,D^-) = 0.8$ then loss is 0

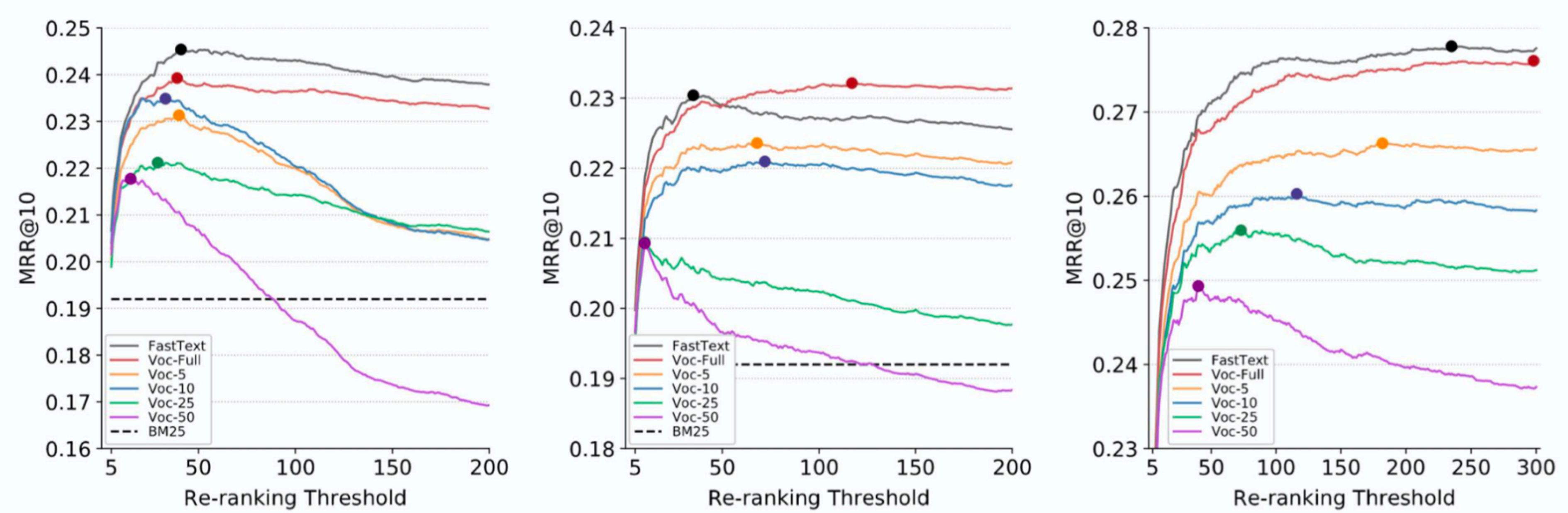
Inference (Validation/Test)

- Since neural ranking models are based on soft matching, we can't simply exploit inverted index to select a set of candidate documents

What to do?

- Option 1: calculate relevance score $f(Q, D)$ for all documents in the collection for each query → full ranking
  - Highly expensive to calculate!
- Option 2: Re-ranking top results of a fast ranker
  - First retrieve a set of documents using a fast model (like BM25)
  - Select top-K documents from the retrieved results
    - K is re-ranking threshold
  - Calculate relevance score $f(Q, D)$ for the top-K documents using the neural ranking model
  - Re-order (re-rank) the top-K documents in the original retrieved results using the scores of the neural ranking model

Effect of Re-ranking Threshold

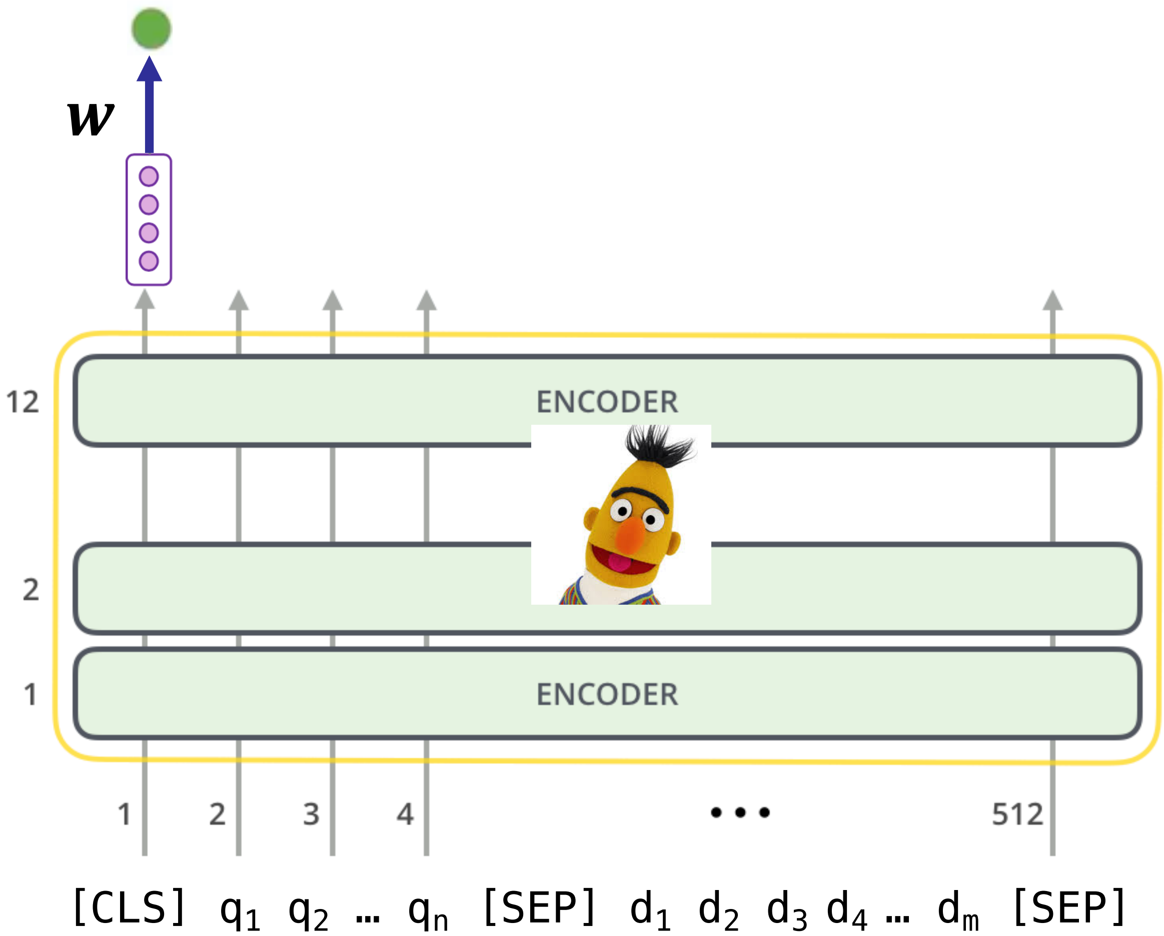


(a) ***MatchPyramid*** (b) ***KNRM*** (c) ***CONV-KNRM***

Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. *On the Effect of Low-Frequency Terms on Neural-IR Models*. In Proceedings of SIGIR 2019

BERT Fine-tuning for ranking

$$\mathrm{score}(Q, D) = f(Q, D)$$



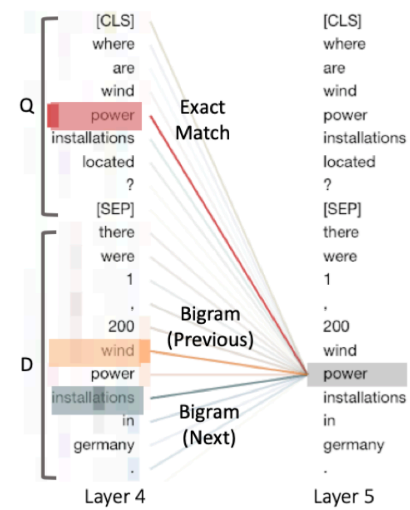
Some results

Ranking Model	Model Parameters		Evaluation	
	All	Transferred	MRR	Recall
BM25	–		0.192	0.398
KNRM_{RND}	109,481,411	none	0.213	0.390
KNRM		GloVe	0.230	0.439
$\text{MatchPyramid}_{RND}$	109,539,960	none	0.232	0.424
MatchPyramid		GloVe	0.240	0.445
PACRR_{RND}	109,875,938	none	0.228	0.426
PACRR		GloVe	0.242	0.451
ConvKNRM_{RND}	110,022,399	none	0.243	0.443
ConvKNRM		GloVe	0.268	0.488
BERT-Base	109,483,778	all	0.342	0.585
BERT-Large	335,143,938	all	0.353	0.596

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

Some open challenges in neural rankings

- Ranking instead of re-ranking
- Green and energy-efficient neural ranking models
- Interpretability and understanding of neural ranking models
- Reinforcement learning for learning to rank
- Measurement of and debiasing reflected societal biases in search engines (next lecture)



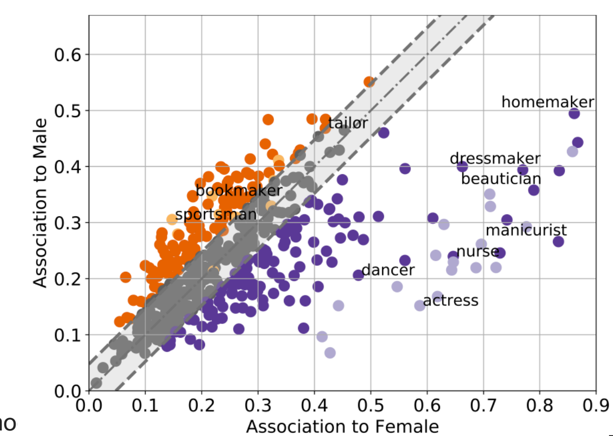


Figure source: Dai, Zhuyun, and Jamie Callan. "Deeper text understanding for IR with contextual neural language mo *Proceedings of SIGIR* 2019.