NLP & IR (WMM9MO75)

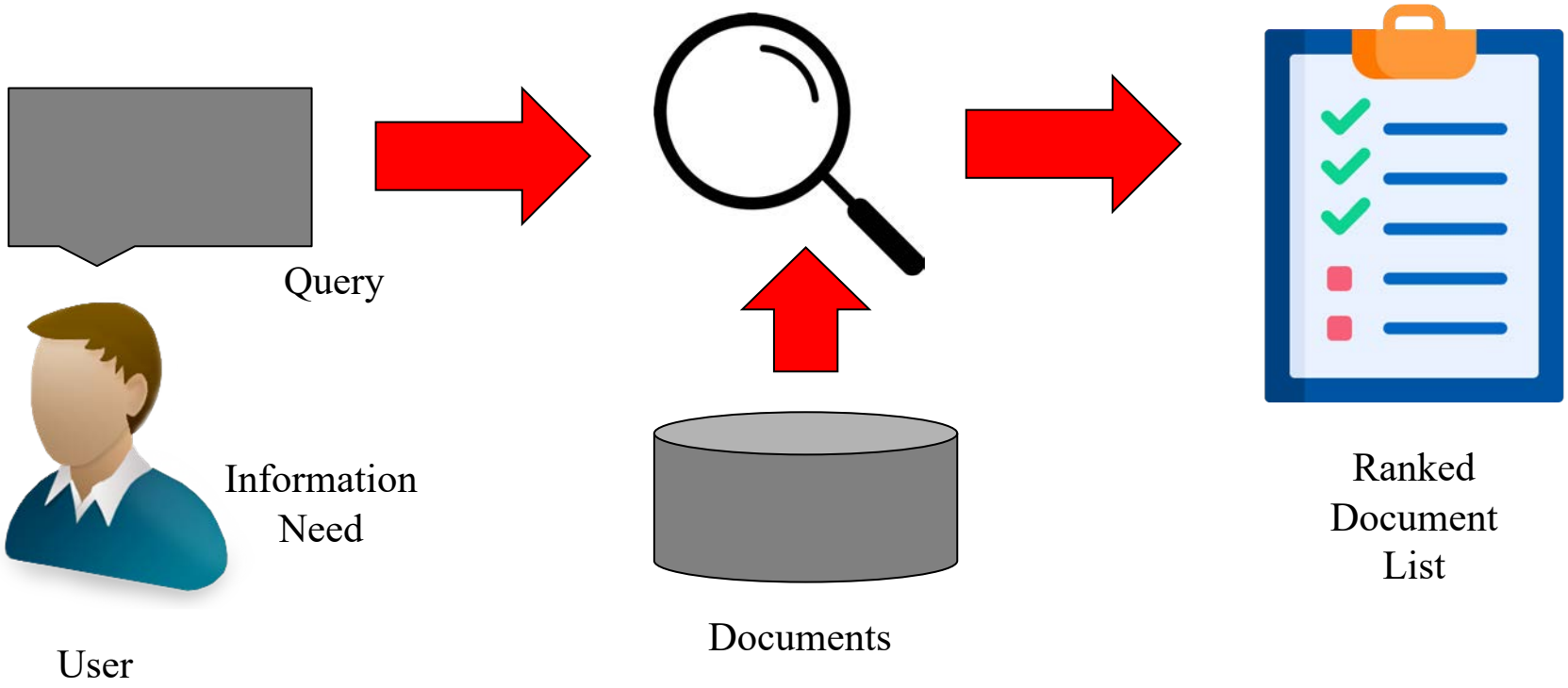
Neural Models in IR - 2023

Petra Galuščáková
Philippe Mulhem
Philippe.Mulhem@imag.fr
MRIM-LIG Research group
http://lig-membres.imag.fr/mulhem/

Outline

1. From probabilistic models to machine learning models
   - Learning to Rank
2. Neural Models – Deep Learning
   1. Pre-BERT neural IR
   2. MS MARCO Collection
   3. BERT in IR
      - MonoBERT / T5
   4. Multi-stage reranking architectures
3. Refining query and document representations
   1. Doc2Query
   2. DeepCT
4. Dense retrieval
   1. ColBERT
   2. COIL

Information Retrieval



The Challenges of Exact Match

$$rsv_{BM25}(d|r,q) =_{rank} \sum_{t \in d \cap q} \log\left(\frac{N - n_t + 0.5}{n_t + 0.5}\right) . \frac{(k_1 + 1) w_t^d}{k_1((1-b) + b . \frac{dl}{avdl}) + w_t^d} . \frac{(k_3 + 1) . w_t^q}{k_3 + w_t^q}$$

N = number of documents in the corpus
n_t = number of documents that contain term t
w_t^d = the number of times term t appears in document d (w_t^q for query q)
k_1, k_3 and b are free parameters
dl: length of document d
avdl = the average document length across all documents in the collection

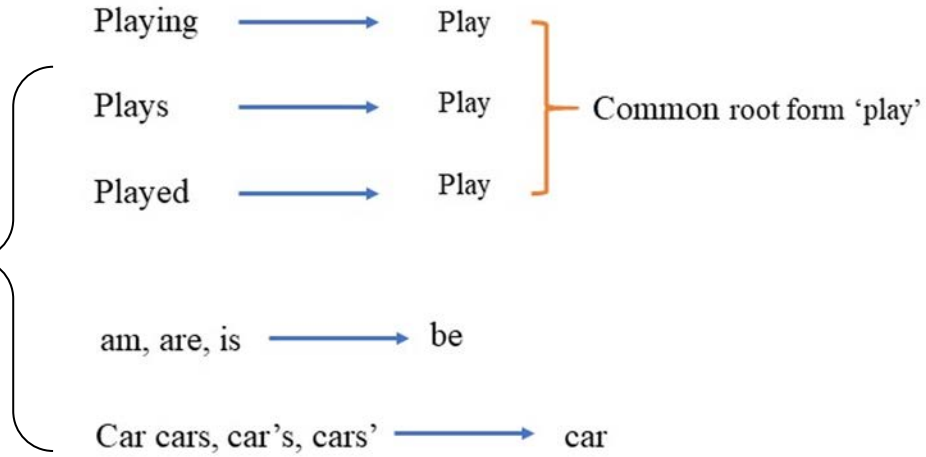
+ inverted file implementation

- synonyms / similar terms ? => vocabulary mismatch

Vocabulary mismatch processing

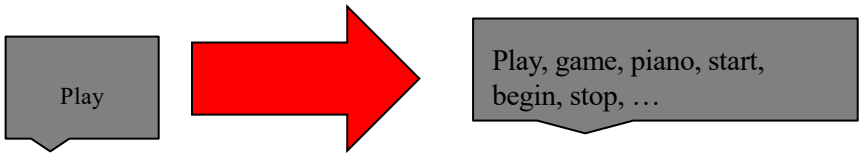
Classical approaches

Stemming and text processing



Enriching query representations

Enriching document representations



Vocabulary mismatch processing

- Is it possible to learn « something » to lower such mismatch ?

From Probabiblistic to learning models (Learning to Rank, LTR)

- Supervised machine-learning techniques to learn ranking models
  - Use additional information (not only terms)
- Use hand-crafted, manually-engineered **features**:
  - **Statistical** properties of terms: functions of term frequencies, document frequencies, document lengths, proximity features
  - **Intrinsic** properties of texts: e.g. the amount of JavaScript code on a web page or the ratio between HTML tags and content, editorial quality, spam score, the count of in/out links, PageRank scores
  - **Interaction**: how many times users issued a particular query or clicked on a particular link

Learning to Rank – Loss function

- **Pointwise:** losses on individual documents, transforming the ranking problem into classification or regression.

- **Pairwise:** losses on pairs of documents, focuses on preferenced: the property wherein A is more relevant than (or preferred over) B.

- **Listwise** approaches consider losses on entire lists of documents, directly optimizing a ranking metric such as nDCG.

Most effectively applied in **decision forest trees** (ensemble of decision trees).

Credit: Keita Kurita, Machine Learning Explained: Learning to Rank Explained (with Code)

Neural Models – Deep-Learning

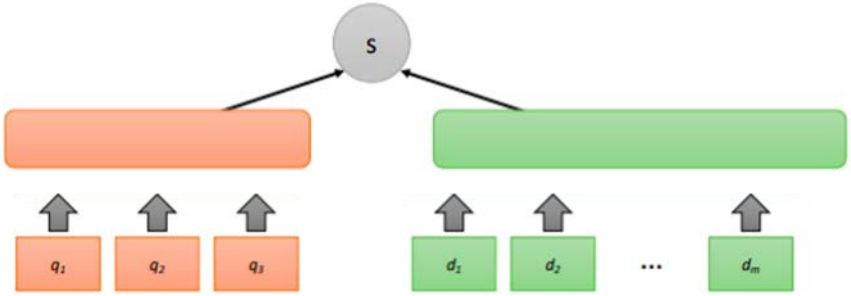
- DL for images since 2013: classification
  - Not obvious tranfert to text and IR
- Freed text retrieval from the bounds of **exact term matching** as in exact-match based IR
- No need for **hand-crafted features** as in LTR
- Pre-BERT neural ranking models: representation-based and interaction-based ("Classical" Feed Forward Networks architectures)
- Large enough dataset availability (MS MARCO)
- BERT revolution ~ 2019 (substantially higher effectiveness)
  - Self-attention based
  - Stacks of encoders/decoders

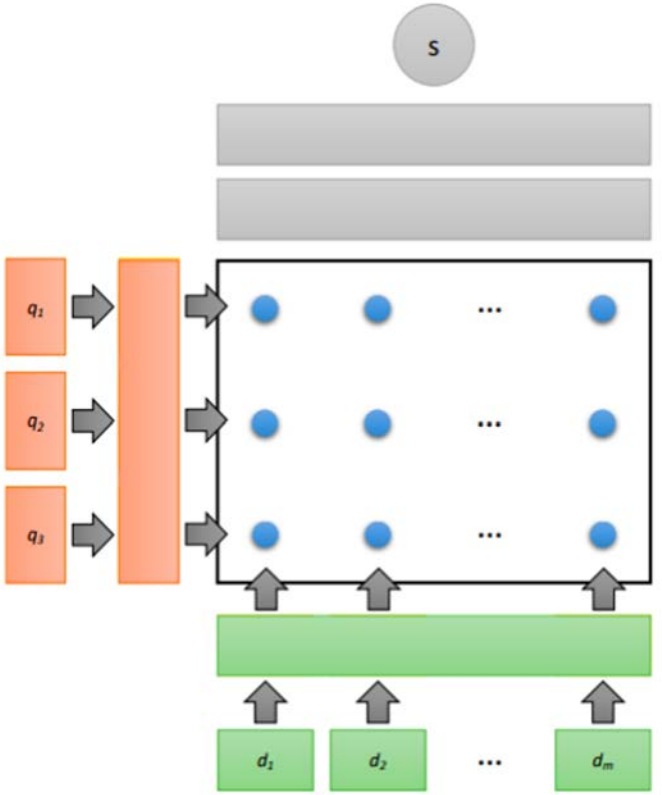
Neural Models – Deep-Learning

- Core questions:
  - What to represent
  - How to represent it
  - What to learn, on which training data

Pre-BERT Neural IR

- Representation-based vs. Interaction-based Models

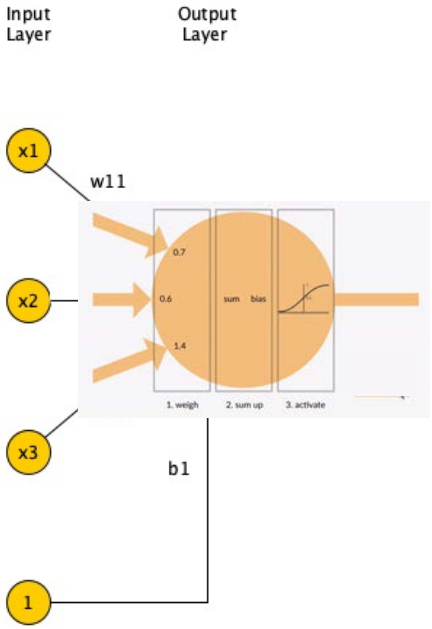




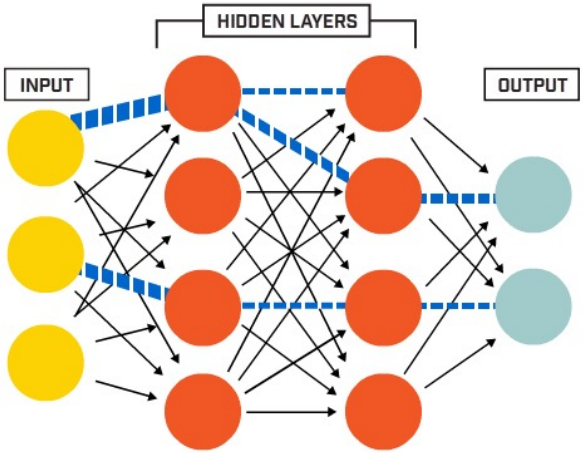
q_i : query terms d_j : document terms

Reminder: Neural Networks

- Simple feed forward architecture
  - From one neuron…

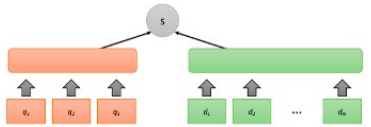


to a network of neurons



https://training.galaxyproject.org/training-material/topics/statistics/tutorials/FNN/tutorial.html
https://medium.com/@b.terryjack/introduction-to-deep-learning-feed-forward-neural-networks-ffnns-a-k-a-c688d83a309d

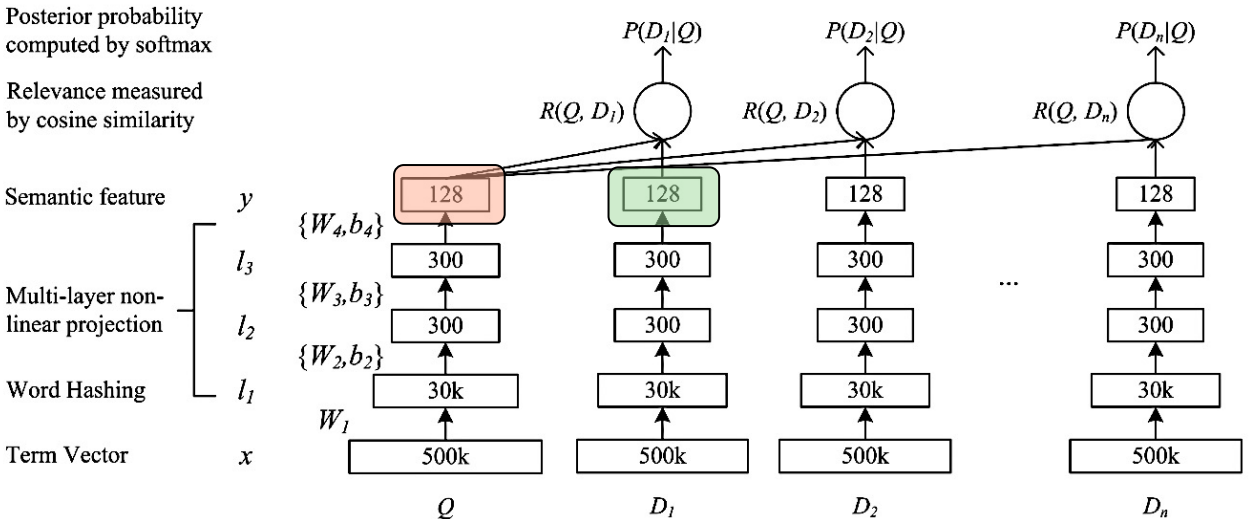
Pre-Bert Representation-Based



- Independently learning **dense vector representations of queries and documents**
- **Relevance** via cosine/inner product between dense vectors

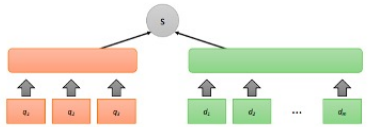
- Matching at ranking time
- **Document representations** computed **offline**… No inverted file implementation

- Deep Structure Semantic Model (DSSM) [Huang et al. 2013]
  - Process of the query/document using word hashing: character 3-grams (compact, few collisions) : "cat" into <#ca, cat, at#>
    - Leading to a 30k number of 3-grams
  - Through fully-connected layers (3) => a 128D vector representation.



Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. ACM CIKM '13, 2333–2338.

Pre-Bert Representation-Based



- Deep Structure Semantic Model (DSSM) [Huang et al. 2013]
  - Learning using clicks (Microsoft) on 100M pairs query-document clicked
  - One query Q associated to a set of documents ***D***
    - ***D***=$\{D^+, D_1^-, D_2^-, D_3^-, D_4^-\}$: D^+ clicked for Q, D^- randomly non-clicked for Q
    - R(Q,D) = cosine(yD, yQ) (yD and yQ the 128D vectors)
    - $P(D|Q) = softmax_{\boldsymbol{D}}(R(D,Q)) = \frac{e^{\gamma R(D,Q)}}{\sum_{D' \in \boldsymbol{D}} e^{\gamma R(D',Q)}}$ (ex.

	R	P
D^+	0,7	0,482
D1-	0,4	0,357
D2-	-0,4	0,161

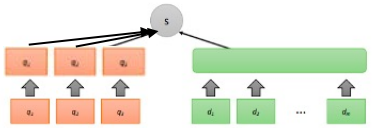
)
    - loss(Q,D) = -*log* $P(D^+ | Q)$

loss
0,317

  - Optimization through *classical* Stochastic Descend Gradient
  - Trained using clicks (Microsoft)
    - on 100M pairs query-document clicked
  - On an internal 16K queries test set:
    - Versus BM25: +17% for NDCG@1, +9% for NDCG@10

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. ACM CIKM '13, 2333–2338.

Pre-Bert Representation-Based



- Dual Embedding Space Model (DESM) [Mitra et al. 2016]
  - Represents texts using pre-trained Word2vec embeddings
    - Continuous Bag Of Words (CBOW): predict a word from its context
    - 1 word => one vector. … No inverted file implementation
  - relevance score:

$$DESM(Q,D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{\mathbf{q}_i^T \overline{\mathbf{D}}}{\|\mathbf{q}_i\| \|\overline{\mathbf{D}}\|}$$

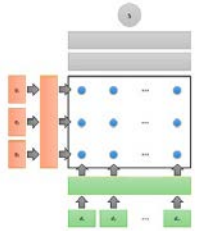
where

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

    - averaged scalar product of normalized vectors
    - across each query term representations $\mathbf{q_i}$ on the document representation $\overline{\mathbf{D}}$
  - CBOW training on 680M queries from BING (Microsoft)
  - On an (internal to Microsoft) 8K test set: +7% for NDCG@1, +7% for NDCG@10 vs BM25

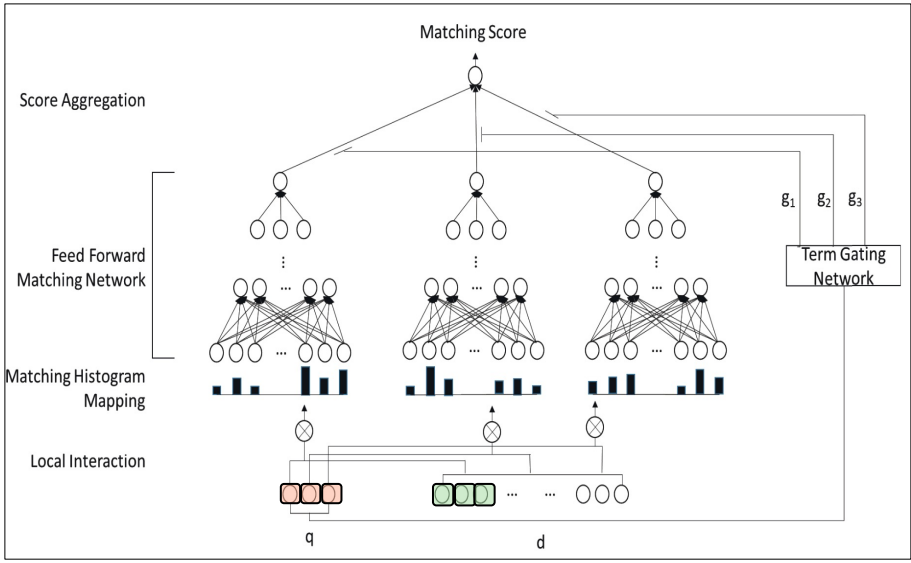
Bhaskar Mitra, Eric Nalisnick, Nick Craswell, Rich Caruana, A Dual Embedding Space Model for Document Ranking, 2016. arXiv:1602.01137

Pre-Bert Interaction-Based



- **A similarity matrix** that captures term-term interactions
  - Each entry $\mathbf{m_{i,j}}$ in the matrix is usually populated with the cosine similarity between the **embedding of the i-th query term** and the **embedding of the j-th document term**.
- **Matrix** analysed to arrive at a final **relevance score**

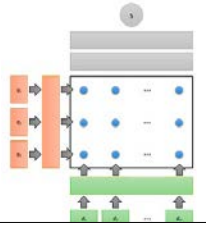
- Ex. DRMM [Guo et. al 2016]
  - Uses one matching histogram doc term x query term generating bins of cosine values (30 bins) pre-trained CBOW embeddings (300D)
  - passed through a 3-layers FC (30/5/1) network lead to one output per query term
  - Gaited (= weighted) by query term weights
  - Training of FFN: pairwise loss (q, d+, d-)
    - Loss = max(0, 1-s(q,d+) + s(q,d-))
  - NDCG@10 +15% vs. BM25 on ClueWeb test collection

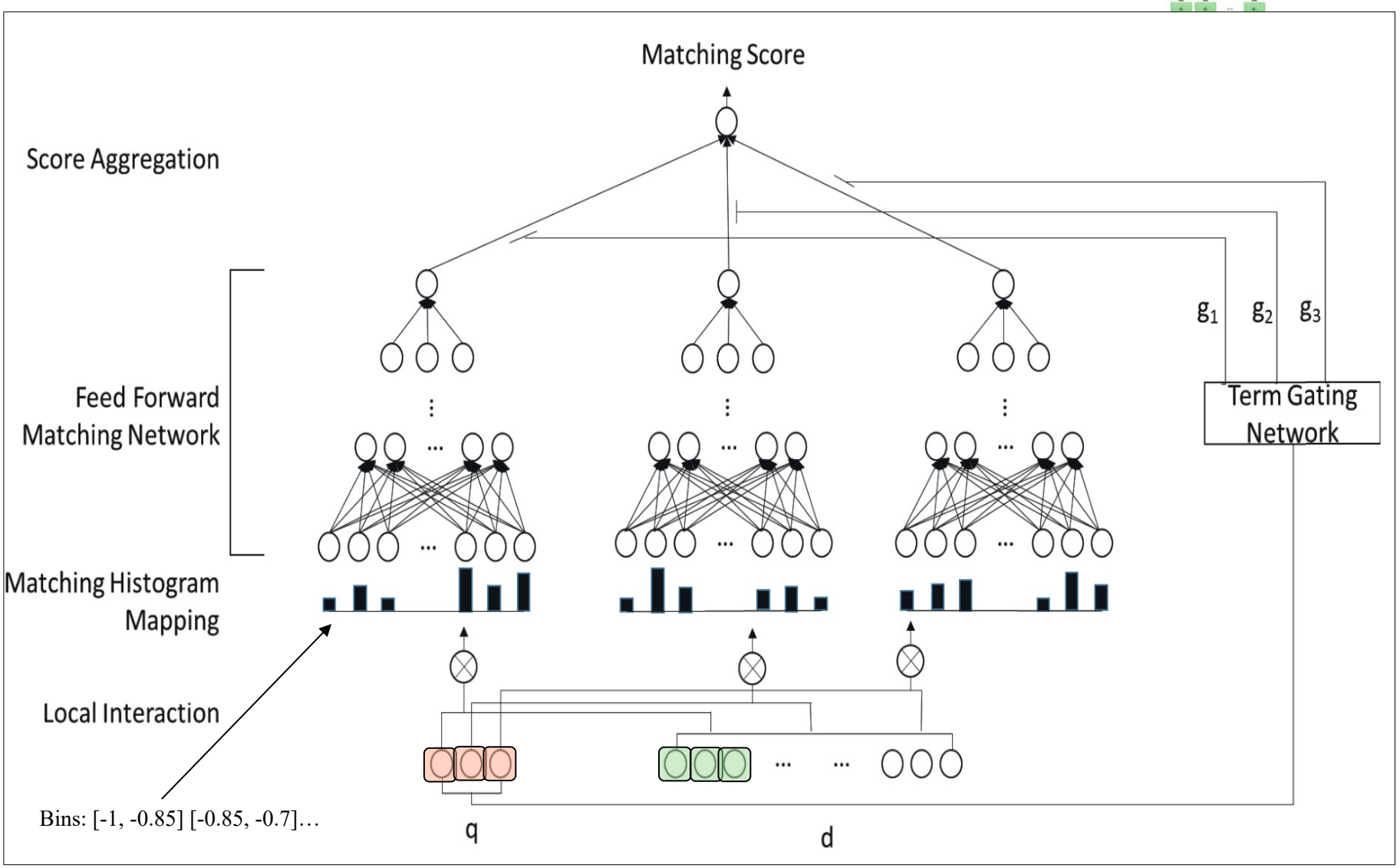


- Pre-BERT interaction-based models more effective but slower than pre-BERT representation-based models.

Jiafeng Guo, Yixing Fan, Qingyao Ai, W. Bruce Croft A Deep Relevance Matching Model for Ad-hoc Retrieval, CIKM 2016.

DRMM (zoom)





Jiafeng Guo, Yixing Fan, Qingyao Ai, W. Bruce Croft A Deep Relevance Matching Model for Ad-hoc Retrieval, CIKM 2016.

Pre-Bert

To what extent do neural ranking models "work" on the **limited amounts of training data** that are publicly available?

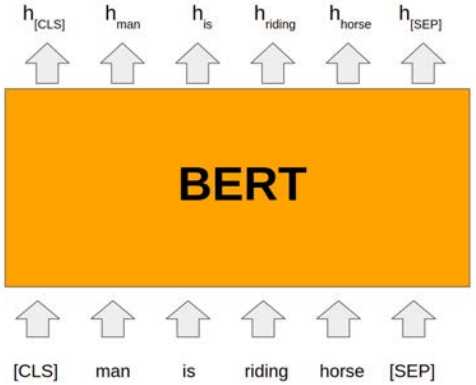
=> Under limited data condition, most of the neural ranking methods **were unable to beat the keyword search baseline.**

Large data only available for the large companies before MS MARCO [Nguyen et al. 2016]

=> BERT arrival did change the game

Arrival of BERT

- Bidirectional Encoder Representation from Transformers [Devlin et al., 2019]



  - Helpful in many NLP tasks
  - Compared to Word2vec: sense of words depend of context
  - Transfert to IR not obvious… but possible !

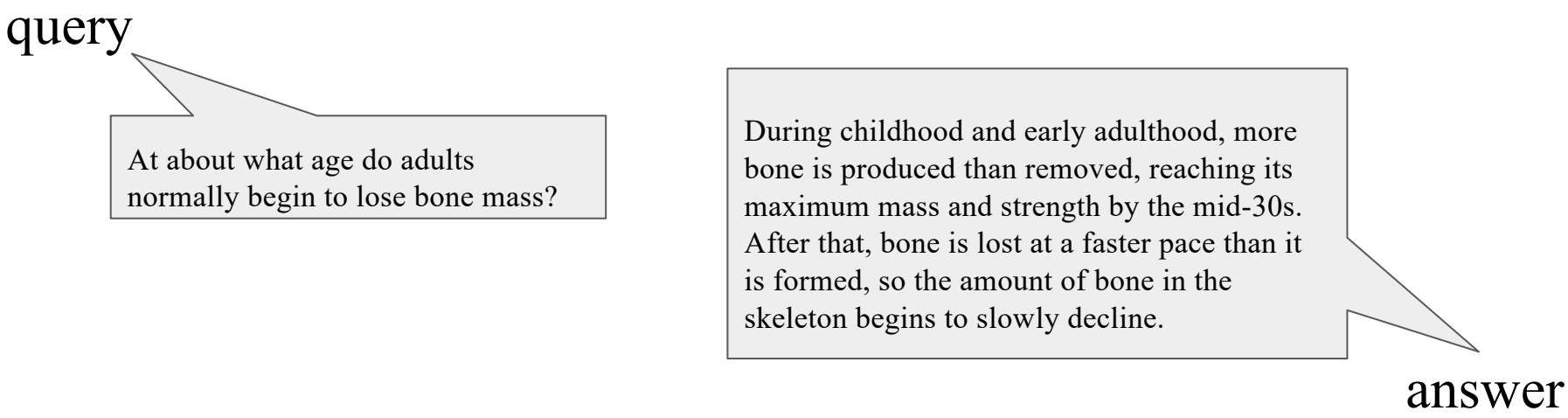
- First application of BERT on IR in January 2019 [Nogueira and Cho, 2019] -> 30% improvement/BM25

- Large amounts of data not necessary (but helpful)

Graphic from https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/

MS MARCO Collection

https://microsoft.github.io/msmarco/TREC-Deep-Learning



- Anonymized natural language questions from **Bing's query logs**
- Initially, designed to study **question answering** on Web passages, later adapted into traditional ad-hoc ranking tasks
- Very natural, often ambiguous, poorly formulated, and may even contain typographical and other errors.
- For each query, the test collection contains, on average, one relevant passage (assessed by human annotators)
- Prepared "triples": query, relevant passage, non-relevant passage

MS MARCO Collection

- Corpus:
  - 3.563M docs
  - 8.841M passages
- Queries and relevance judgements:

Dataset	$\|q\|$	$\overline{L}(q)$	$\|J\|$	$\|J\|/q$	$\|\text{Rel}\|/q$
MS MARCO passage ranking (training set)	502,939	6.06	532,761	1.06	1.06
MS MARCO passage ranking (development set)	6,980	5.92	7,437	1.07	1.07
MS MARCO passage ranking (evaluation set)	6,837	5.85	-	-	-
MS MARCO document ranking (training set)	367,013	5.95	367,013	1.0	1.0
MS MARCO document ranking (development set)	5,193	5.89	5,193	1.0	1.0
MS MARCO document ranking (evaluation set)	5,793	5.85	-	-	-

Table 1: Summary statistics of queries and relevance judgments for the MS MARCO datasets: number of queries $|q|$, mean query length $\overline{L}(C)$, number of judgments $|J|$, judgments per query $|J|/q$, and relevant documents per query $|\text{Rel}|/q$.

Learning data ! ... But what to do with it ?

Table from Jimmy Lin, Daniel Campos, Nick Craswell, Bhaskar Mitra, and Emine Yilmaz. Fostering Coopetition While Plugging Leaks: The Design and Implementation of the MS MARCO Leaderboards. ACM SIGIR '22.

BERT in IR

From the success of BERT for NLP tasks (cf. NLP lessons), how to use it in IR ?

The simplest and most straightforward formulation of text ranking:

convert the task into a text classification problem

Train a classifier to estimate the probability that each text belongs to the **"relevant" class**

Start with a **pretrained model** and then **fine-tune** it further using labeled data from the target task. (transfert learning)

Ranking (i.e. inference) : Sort the texts to be **ranked** based on the **probability** that each item **belongs to the desired class**.

A simple, robust, effective, and widely replicated model for text ranking

Key **limitation** of BERT for text ranking: its inability to handle **long input sequences**

MonoBERT [Nogueira et al. 2019]

- Idea :
  - Based on text input: concatenate query & document

    [CLS] <q> [sep1] <d> [sep2]

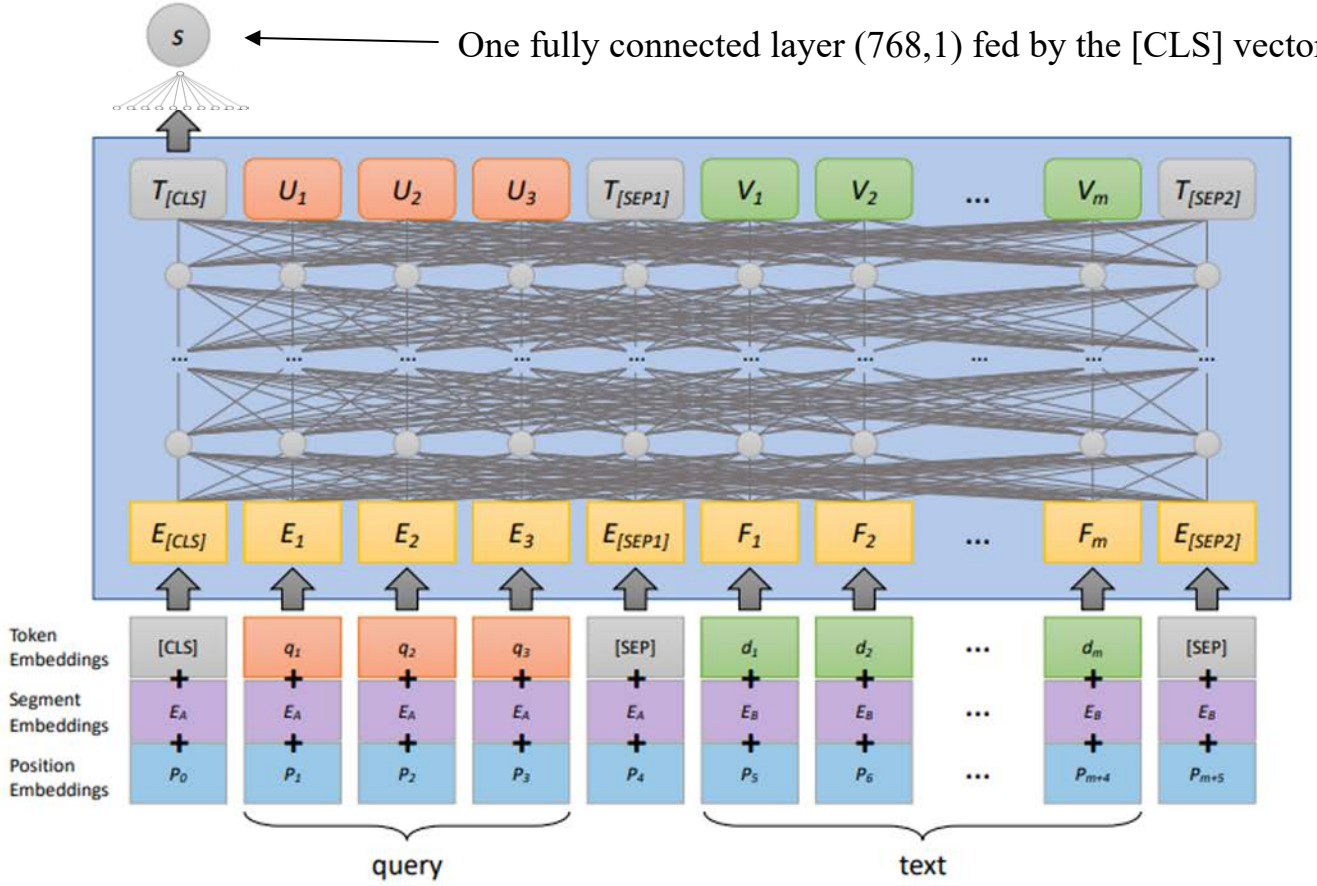
    With [CLS] [sep1] [sep2] specific tokens

    <q> and <d> query and document text

  - Learn the score of d for q
    - 1 for relevant / 0 for non-relevant

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, Jimmy Lin, Multi-Stage Document Ranking with BERT, 2019. arXiv:1910.14424

MonoBERT

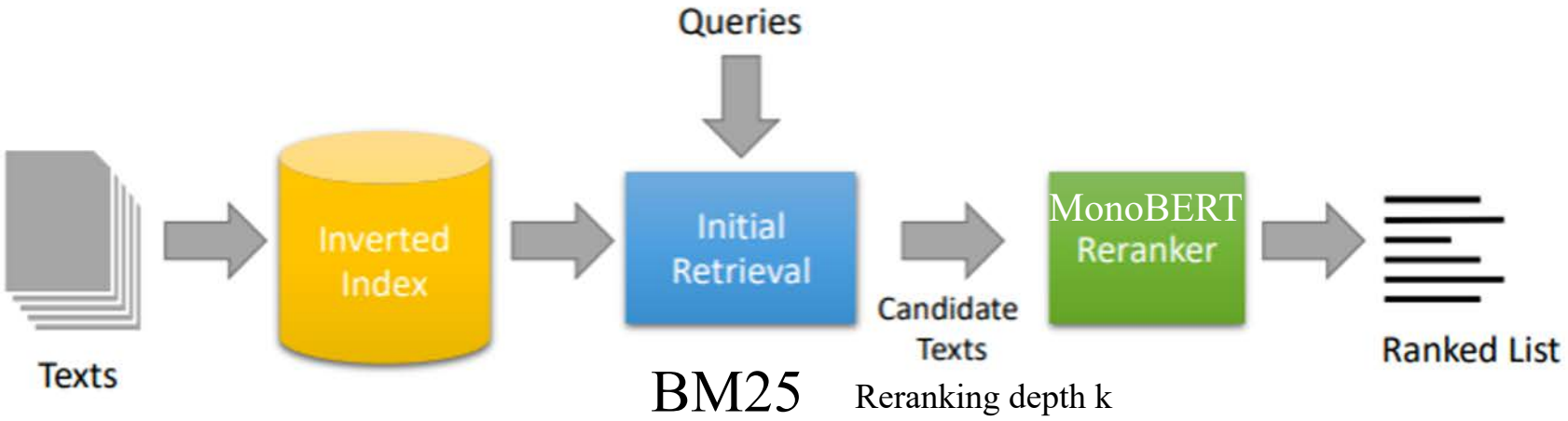


Training loss: $L = -\sum_{j \in J_{\text{pos}}} \log(s_j) - \sum_{j \in J_{\text{neg}}} \log(1 - s_j)$

s_j close to 1: good, close to 0: no good

s_j close to 0: good, close to 1: no good

Reranking using MonoBERT



Reranking using MonoBERT

- Training
  - TCP: Pre-trained BERT
    - Wikipedia + Toronto corpus
  - Reranking monoBERT
    - Training set (from 500k (Q,D_+) and 400M (Q,D_-))
      - 12.8M query-document pairs were used, with equal amount of rel/non-rel *per batch*

monoBERT Results

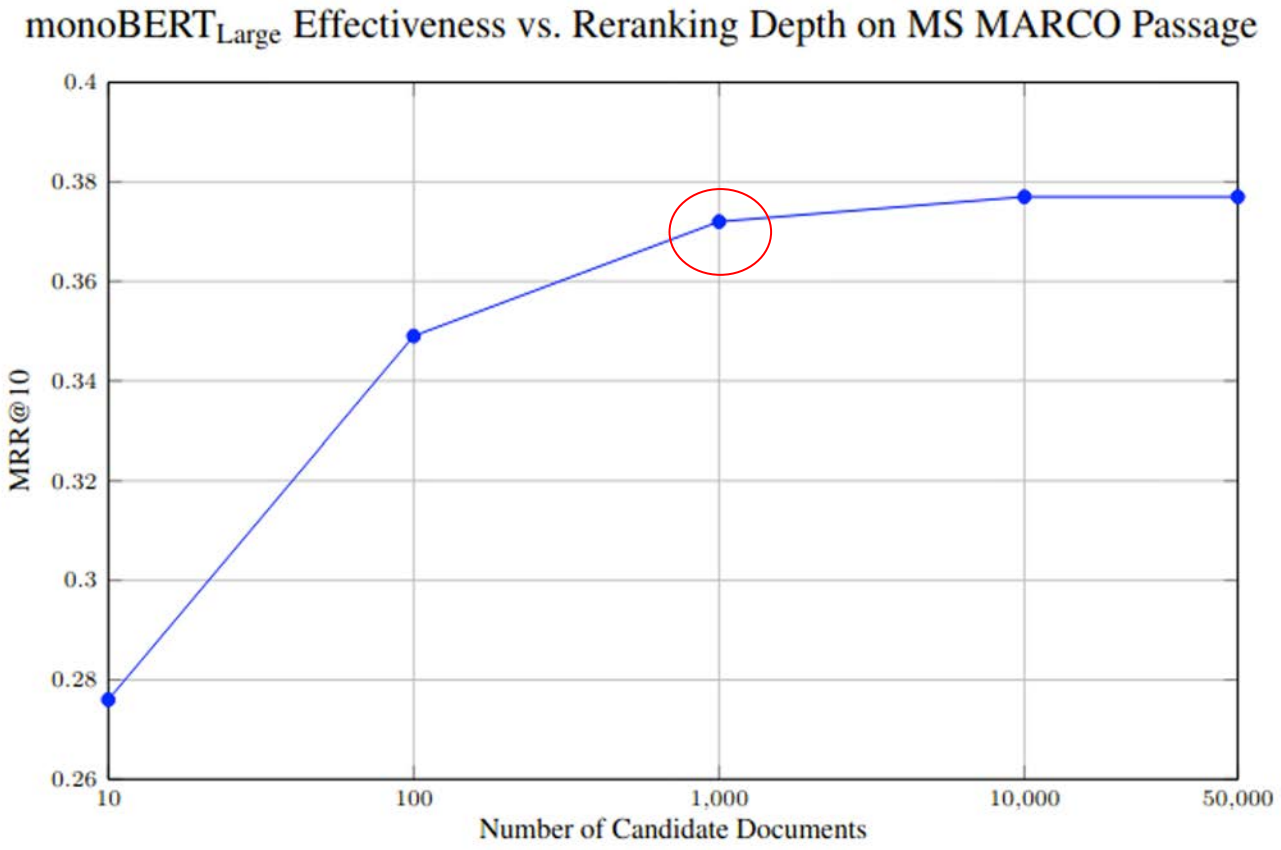
MRR: Mean Reciprocal Rank

Method	MS MARCO Passage		
	Development		Test
	MRR@10	Recall@1k	MRR@10
BM25 (Microsoft Baseline, $k = 1000$)	0.167	-	0.165
+ monoBERT$_{\text{Large}}$ [Nogueira and Cho, 2019]	0.365	-	0.359
+ monoBERT$_{\text{Base}}$ [Nogueira and Cho, 2019]	0.347	-	-
BM25 (Anserini, $k = 1000$)	0.187	0.857	0.190
+ monoBERT$_{\text{Large}}$ [Nogueira et al., 2019a]	0.372	0.857	0.365

Large and Base refer to different versions of BERT

- BERT$_{\text{Large}}$: 24 stacks encoders
- BERT$_{\text{Base}}$: 12 stacks encoders

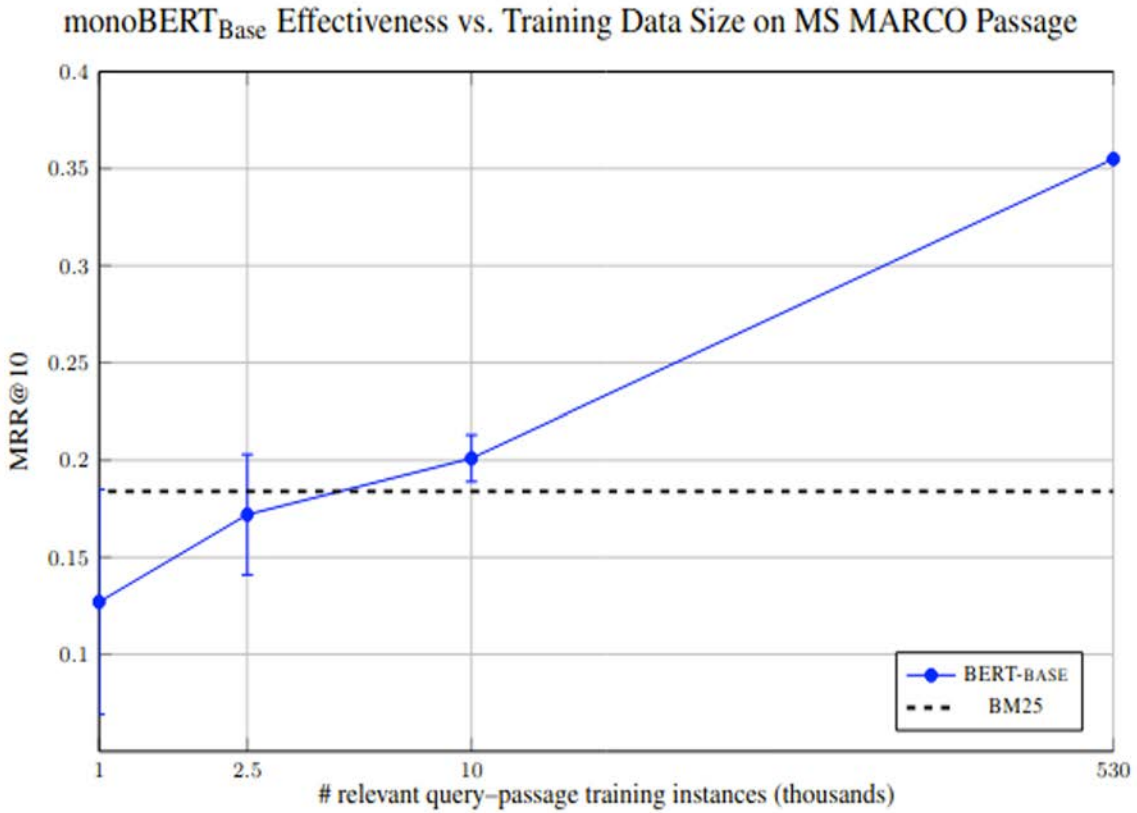
MonoBERT–Effect of reranking depth



⇒ k=1000 a good trade-off between computation time and performance

MonoBERT – Effect of training size

- MonoBERT fine-tuned with 1K, 2.5K, 10K 500K (Q,D_+)
  - Reranking top-1000



=> Outperforms BM25 <u>if training large enough</u>

From short Passages to Document Retrieval

- monoBERT is limited to ranking paragraph-length passages
  - MonoBERT input: 512 tokens for "[CLS] <q> [sep1] <d> [sep2]"
    - 512 tokens ≃ 400 words (this slide: 56 words)

- How to deal with longer documents
  - Only use first n characters/sentences/paragraphs from the document
  - Aggregation during the inference:
    - Scores (*Max* good [Dai and Callan SIGIR 2019])
    - Representations
  - No guarantee that the segments are relevant in training

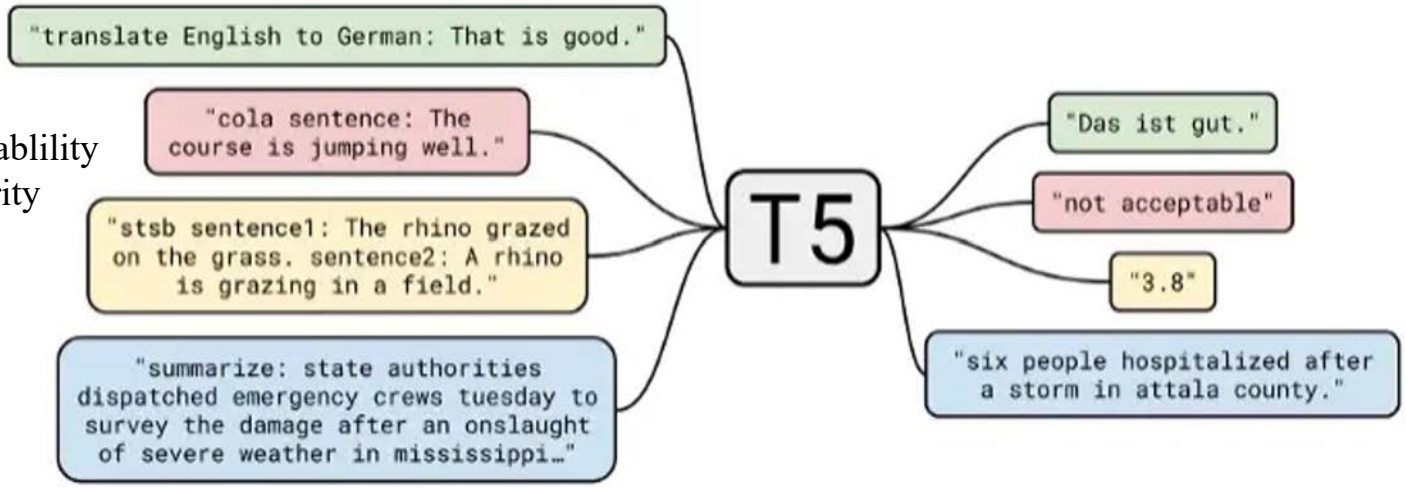
T5 – Text to Text Transformers

T5 [Raffel et al., 2019] is first pretrained on a large corpus of diverse texts using a self-supervised objective similar to masked language modelling in BERT

=> then retrained to text **sequence-to-sequence**: from text to text
trained on 750 GB of text, T5_3B has 3B parameters…

These pretrained models are fine-tuned for **various downstream tasks** using task-specific labeled data, where each task is associated with a specific input template

translation
sentence acceptablility
sentence similarity
sumarization



Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2019. arXiv:1910.10683

Reranking with T5 [Nogueira et al. 2020]

Use the following input template (compatible with seq2seq):

Query: <q> Document: <d> Relevant: <relevance>

- <q> and <d> are replaced with the query and document texts, respectively

- and the other parts of the template are verbatim string literals

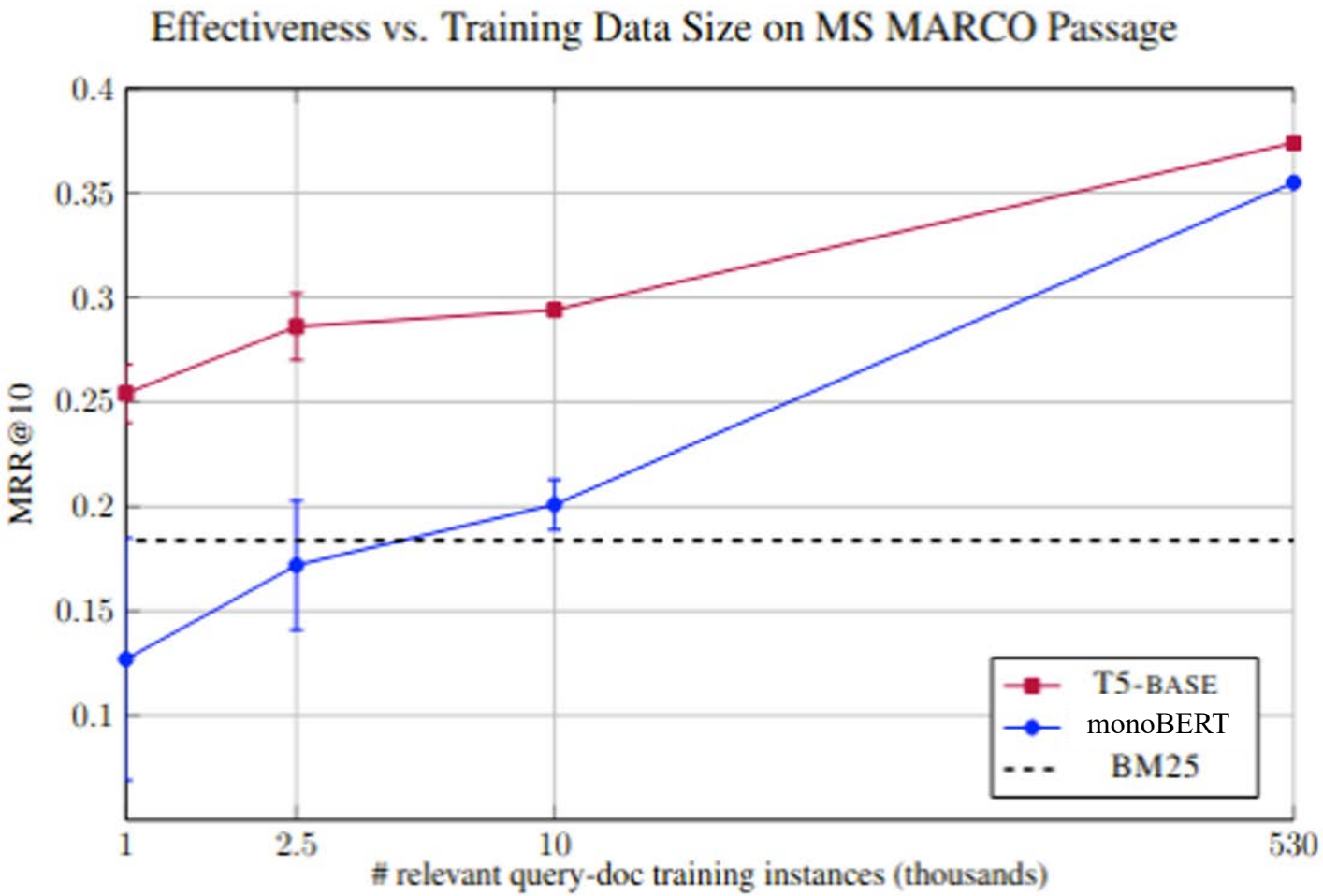
The model is **fine-tuned** to produce the tokens "**true**" or "**false**" for <relevance> depending on whether the document is relevant or not to the query using MS MARCO

$\Rightarrow$ the strings "true" and "false" are the "target tokens" (i.e., ground truth predictions in the sequence-to-sequence transformation)

$\Rightarrow$ So, similar to the classification problem, training as monoBERT

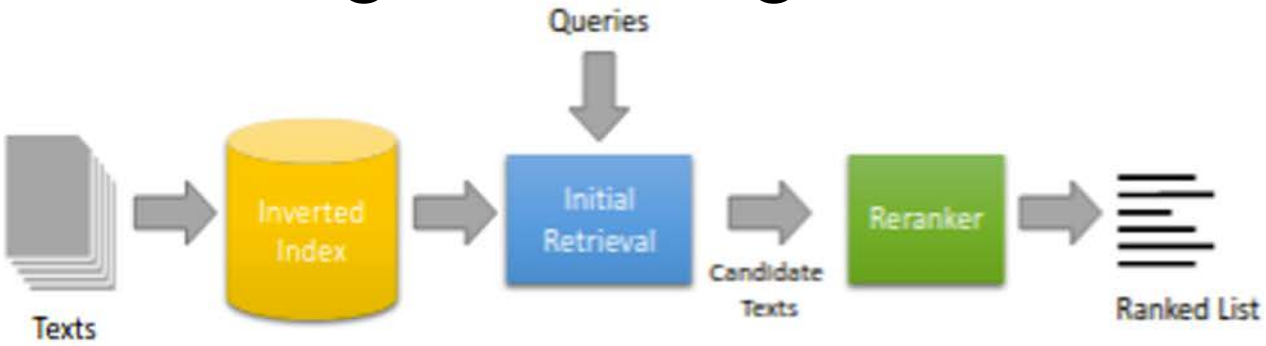
Similar to monoBERT, monoT5 is deployed as a **reranker**

T5 versus BERT

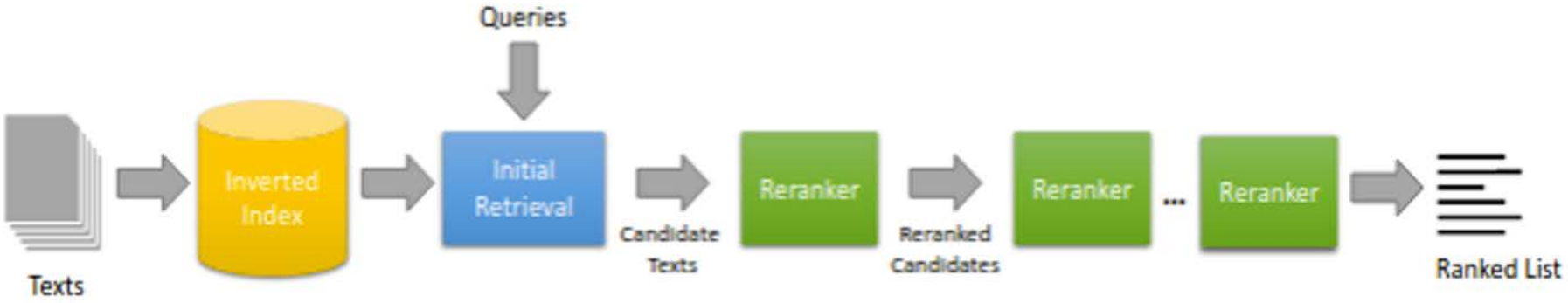


Multi-Stage Reranking Architectures

From single reranking…



…To multiple rerankings



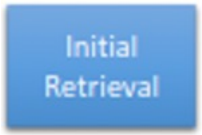
Multi-Stage Reranking Architectures

- Better balance **tradeoffs** between **effectiveness** (quality of the ranked lists) and **efficiency** (e.g. retrieval latency or query throughput).
- Exploit expensive features only when necessary

- **Earlier stages** in the reranking pipeline can use "**cheap**" features to discard candidates that are easy to distinguish as not relevant.
- "**Expensive**" **features** can then be brought to bear after the "easy" non-relevant candidates have been discarded.

Multi-Stage Reranking Architectures

- [Nogueira et al., 2019]
  - 3 stages

1. BM25 

2. monoBERT (already discussed): pointwise loss 

3. DuoBERT: paiwise loss (next slide) 

Multi-Stage Reranking Architectures

- DuoBERT
  - Estimation of $P(d_i \succ d_j | d_i, d_j, q)$,

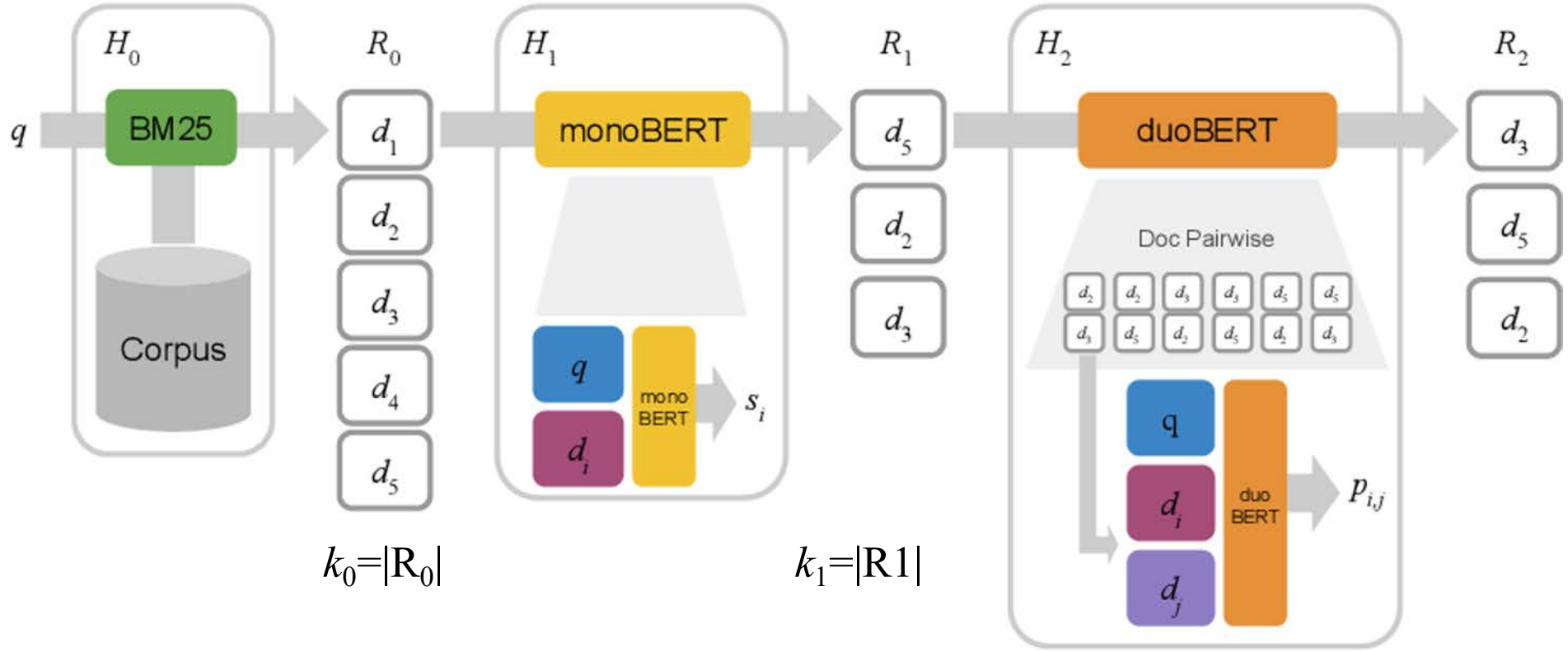
    where $d_i \succ d_j$ denotes that d_i is more relevant than d_j (with respect to q)

The duoBERT model is trained to estimate $p_{i,j}$, the probability that $d_i \succ d_j$, i.e., candidate d_i is more relevant than d_j. It takes as input a sequence comprised of a query and two texts, comprising the input template:

$$[\texttt{[CLS]}, q, \texttt{[SEP]}, d_i, \texttt{[SEP]}, d_j, \texttt{[SEP]}],$$

Multi-Stage Reranking Architectures

- DuoBERT detailed architecture



Multi-Stage Reranking Architectures

- duoBERT
  - The result of model inferences comprises **a set of pairwise comparisons between candidate texts**. Evidence from these pairs still need to be **aggregated** to produce a final ranked list
  - Compare each candidate to every other candidate (e.g., from first-stage retrieval), and thus the **computational costs increase quadratically** with the size of the candidate set.
  - Due to the length limitations of BERT of 512 tokens, the **query, candidates d_i and d_j are truncated** to 62, 223, and 223 tokens, respectively.

Multi-Stage Reranking Architectures

- DuoBERT scores agregation

$$\textsc{Max}: \quad s_i = \max_{j \in J_i} p_{i,j},$$

$$\textsc{Min}: \quad s_i = \min_{j \in J_i} p_{i,j},$$

$$\textsc{Sum}: \quad s_i = \sum_{j \in J_i} p_{i,j},$$

$$\textsc{Binary}: \quad s_i = \sum_{j \in J_i} \mathbb{1}_{p_{i,j} > 0.5}.$$

i \ j	A	B	C	D
A	-	0.4	0.5	0.7
B	0.8	-	0.7	0.9
C	0.6	0.3	-	0.7
D	0.2	0.1	0.4	-

The **MIN (MAX)** method measures the **relevance** of d_i only against its **strongest** (weakest) "competitor".

The **SUM** measures the pairwise agreement that **candidate d_i is more relevant than the rest** of the candidates.

The **BINARY** method is inspired by the Condorcet method →

Runner \ Opponent	A	B	C	D
A	—	0	0	1
B	1	—	1	1
C	1	0	—	1
D	0	0	0	—

A '1' indicates that the runner is preferred over the opponent; a '0' indicates that the runner is defeated.

Multi-Stage Reranking Architectures

- DuoBERT evaluation results

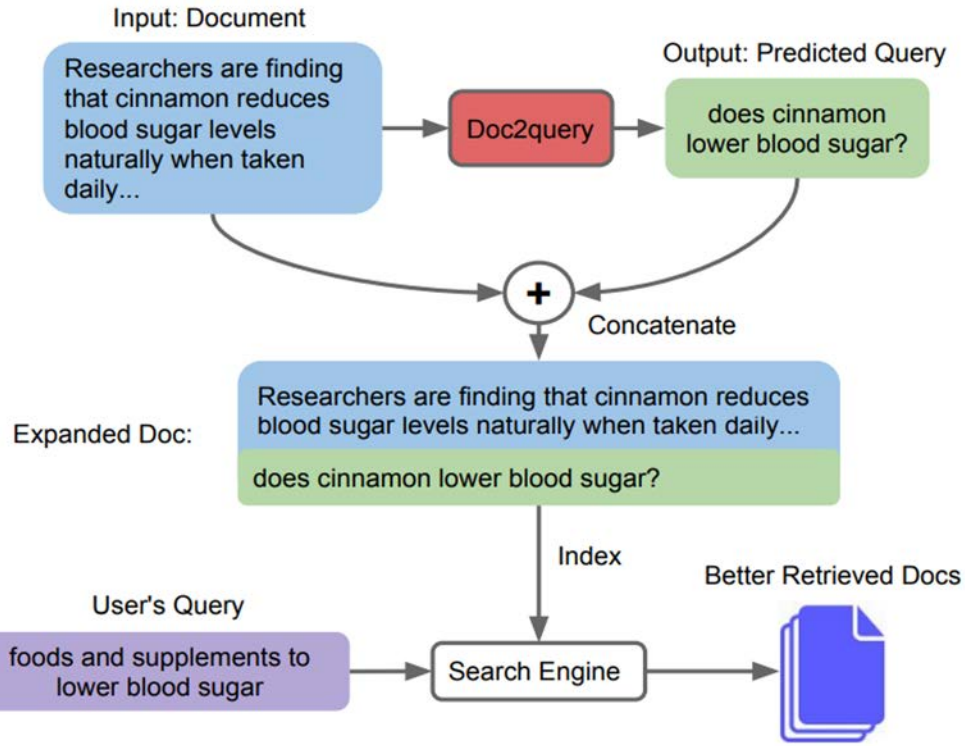
	Method	MS MARCO Passage Development MRR@10	MS MARCO Passage Test MRR@10
(1)	Anserini BM25 = Table 5, row (3a)	0.187	0.190
(2)	+ monoBERT ($k_0 = 1000$) = Table 5, row (3b)	0.372	0.365
	+ monoBERT ($k_0 = 1000$)		
(3a)	+ duoBERT$_{\text{MAX}}$ ($k_1 = 50$)	0.326	-
(3b)	+ duoBERT$_{\text{MIN}}$ ($k_1 = 50$)	0.379	-
(3c)	+ duoBERT$_{\text{SUM}}$ ($k_1 = 50$)	0.382	0.370
(3d)	+ duoBERT$_{\text{BINARY}}$ ($k_1 = 50$)	0.383	-
(4a)	+ monoBERT + TCP	0.379	-
(4b)	+ monoBERT + duoBERT$_{\text{SUM}}$ + TCP	0.390	0.379

Table 21: The effectiveness of the monoBERT/duoBERT pipeline on the MS MARCO passage ranking test collection. TCP refers to target corpus pretraining.

TCP: Target Corpus Pre-training = fine-tuning

Refining documents representations

- Doc2query
  - given a document, Doc2query predicts a query, which is appended to the document

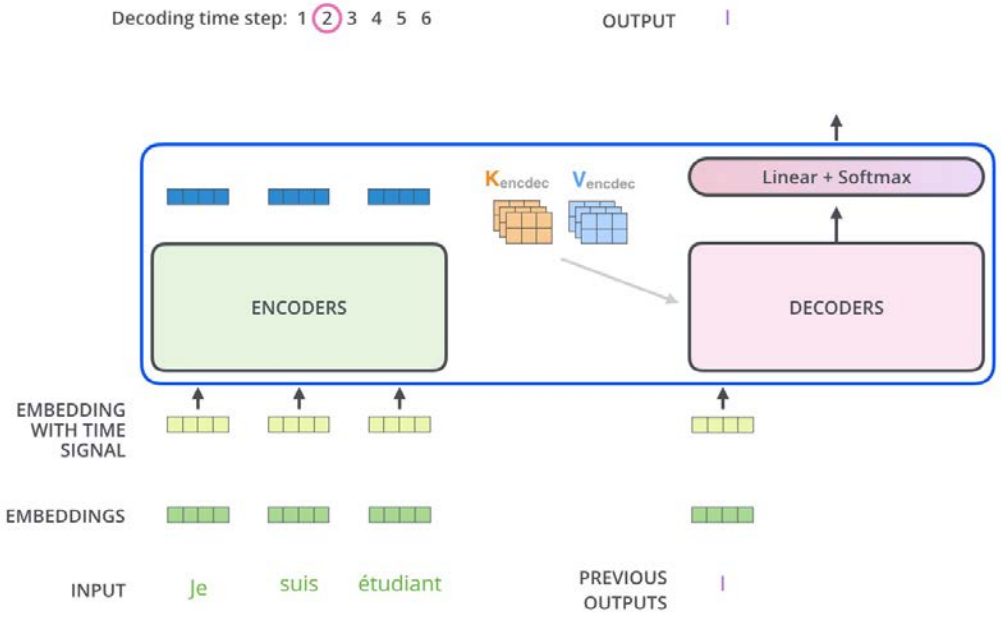


  - Then classical IR on the expanded docs

Rodrigo Nogueira, Wei Yang, Jimmy Lin, Kyunghyun Cho Document Expansion by Query Prediction, 2019. arXiv:1904.08375

Refining documents representations

- Doc2query
  - Needs to generate text: Uses a sequence to sequence encoder-decoder architecture [Vaswani et al. 2017]
    - From an input, it generates probabilities for each token and then outputs the token with higher probability



from https://jalammar.github.io/illustrated-transformer/

Refining documents representations

- Doc2query
  - Expansion of documents not straightforward, must be diverse
    - Can not use only BERT to predict: no diversity
    - Usage of top-k random sampling scheme [Fan et al. 2018]
      - At each timestep, the models generates the probability of each word in vocabulary to be the next word
      - Randomly select one from the k most probable word t
      - Iterate.
    - Generate 10 queries that are appended to the document

Refining documents representations

- Doc2query examples

Input: July is the hottest month in Washington DC with an average temperature of 27°C (80°F) and the coldest is January at 4°C (38°F) with the most daily sunshine hours at 9 in July. The wettest month is May with an average of 100mm of rain.

Target query: what is the temperature in washington

doc2query–base: weather in washington dc
doc2query–T5: what is the weather in washington dc

Input: The Delaware River flows through Philadelphia into the Delaware Bay. It flows through and *(sic)* aqueduct in the Roundout Reservoir and then flows through Philadelphia and New Jersey before emptying into the Delaware Bay.

Target query: where does the delaware river start and end

doc2query–base: what river flows through delaware
doc2query–T5: where does the delaware river go

Input: sex chromosome - (genetics) a chromosome that determines the sex of an individual; mammals normally have two sex chromosomes chromosome - a threadlike strand of DNA in the cell nucleus that carries the genes in a linear order; humans have 22 chromosome pairs plus two sex chromosomes.

Target Query: which chromosome controls sex characteristics

doc2query–base: definition sex chromosomes
doc2query–T5: what determines sex of someone

Figure 20: Examples of predicted queries on passages from the MS MARCO passage corpus compared to user queries from the relevance judgments.

Refining documents representations

- Doc2query results

	Method	MS MARCO Passage			
		Development		Test	Latency
		MRR@10	Recall@1k	MRR@10	(ms/query)
(1a)	BM25	0.184	0.853	0.186	55
(1b)	w/ doc2query–base [Nogueira et al., 2019b]	0.218	0.891	0.215	61
(1c)	w/ doc2query–T5 [Nogueira and Lin, 2019]	0.277	0.947	0.272	64
(2a)	BM25 + RM3	0.156	0.861	-	-
(2b)	w/ doc2query–base	0.194	0.892	-	-
(2c)	w/ doc2query–T5	0.214	0.946	-	-

  - Increase results: tackles some term-matching problems
  - Not as good as MonoBERT, but used on top of MonoBERT outperforms MonoBERT

Rodrigo Nogueira and Jimmy, From do2query ro docTTTTTquery, 2019.
https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-latest.pdf

Refining documents representations

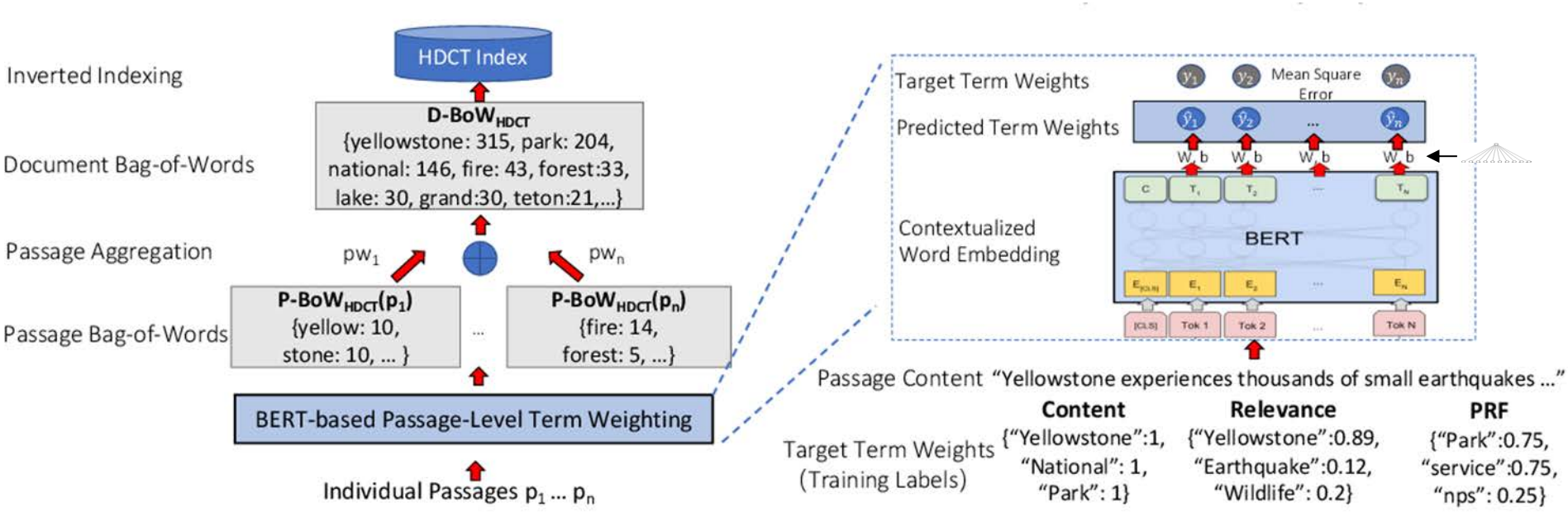
- Term Reweighting as regression: DeepCT
  - **estimate the importance of a term** in the context that the term appears in

	Term weight: 0.0 0.1 0.2 0.3 0.4 > 0.5
Query	**who is susan boyle**
Relevant	Amateur vocalist **Susan Boyle** became an overnight sensation after appearing on the first round of 2009's popular U.K. reality show Britain's Got Talent.
Non-Relevant	Best Answer: a troll is generally someone who tries to get attention by posting things everyone will disagree, like going to a **susan boyle** fan page and writing **susan boyle** is ugly on the wall. they are usually 14-16 year olds who crave attention.
Query	**what values do zoos serve**
Relevant	**Zoos serve** several purposes depending on who you ask. 1) Park/Garden: Some **zoos** are similar to a botanical garden or city park. They give people living in crowded, noisy cities a place to walk through a beautiful, well maintained outdoor area. The animal exhibits create interesting scenery and make for a fun excursion.
Non-Relevant	There are NO purebred Bengal tigers in the U.S. The only purebred tigers in the U.S. are in AZA **zoos** and include 133 Amur (AKA Siberian), 73 Sumatran and 50 Malayan tigers in the Species Survival Plan. All other U.S. captive tigers are inbred and cross bred and do not **serve** any conservation **value**.
Query	**do atoms make up dna**
Relevant	**DNA** only has 5 different **atoms** - carbon, hydrogen, oxygen, nitrogen and phosphorous. According to one estimation, there are about 204 billion **atoms** in each **DNA**.
Non-Relevant	Genomics in Theory and Practice. What is Genomics. Genomics is a study of the genomes of organisms. It main task is to determine the entire sequence of **DNA** or the composition of the **atoms** that make up the **DNA** and the chemical bonds between the **DNA atoms**.

Zhuyun Dai, Jamie Callan, Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval, 2019. arXiv:1910.10687

Refining documents representations

- DeepCT



Zhuyun Dai, Jamie Callan, Context-Aware Document Term Weighting for Ad-Hoc Search, 2020. WWW 2020.

Refining documents representations

- DeepCT
  - Target weight (in [0,1]) from feature vector for term t in context of a passage p T_{BERT}(t,p) (FeedForwardN):

$$\hat{y}_{t,p} = w \cdot T_{\text{BERT}}(t,p) + b$$

  - Scale to a tf-like integer: (ex. N=100)

$$tf_{\text{BERT}}(t,p) = round(N * \sqrt{\hat{y}_{t,p}})$$

  - Generate bag of words representation for passages

$$\text{P-BoW}_{\text{HDCT}}(p) = [tf_{\text{BERT}}(t_1,p), .., tf_{\text{BERT}}(t_m,p)]$$

  - Aggregate passages into documents

$$\text{D-BoW}_{\text{HDCT}}(d) = \sum_{i=1}^{n} pw_i \times \text{P-BoW}_{\text{HDCT}}(p_i)$$

Refining documents representations

- DeepCT
  - Learning: minimize Mean Square Error between predicions and ground truth

$$MSE = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2$$

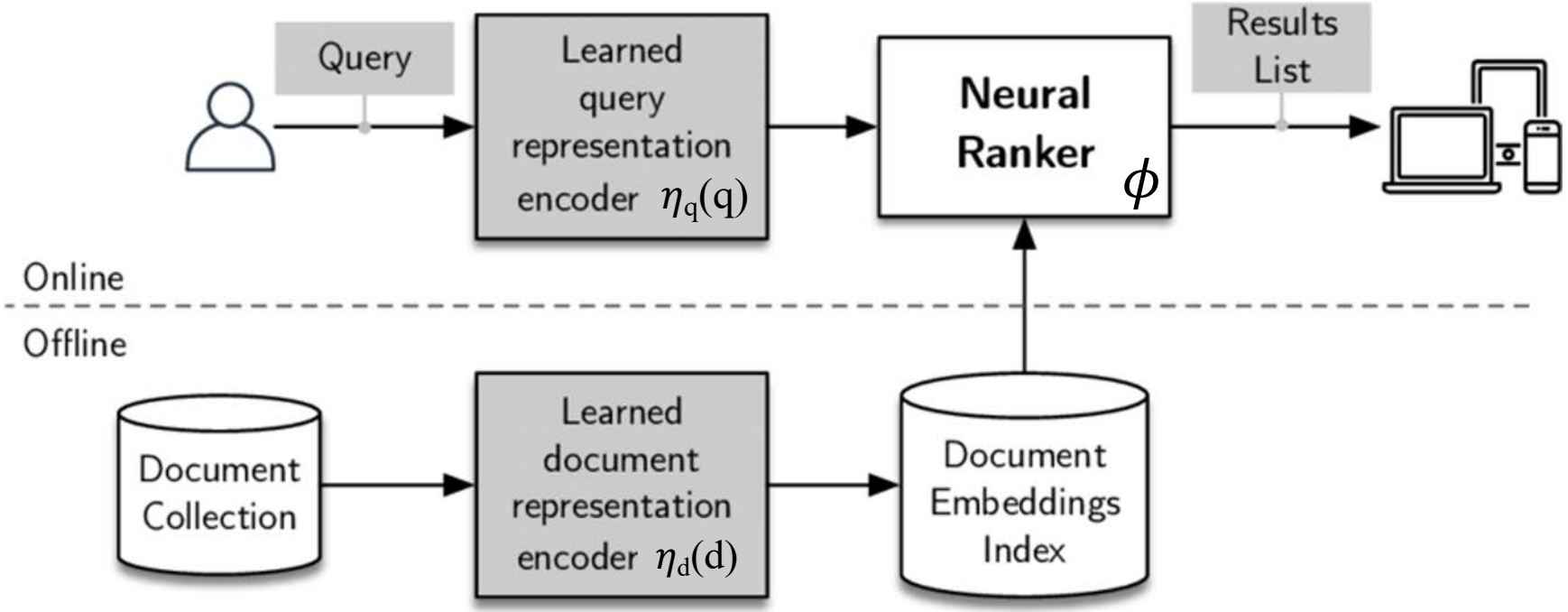
  - Ground truth for training
    - content: occurrence of word in document (typically *title*)
    - PRF: pseudo relevance feedback labels

Refining documents representations

- Term Reweighting as regression: DeepCT

MS-MARCO-Doc		Dev Query	
Retrieval Model	Indexing Term Weight	MRR	
BM25	*tf*	0.254	–
	HDCT-title	0.287*	13%
BM25+RM3	*tf*	0.250	–
	HDCT-title	0.288*	15%

Dense Retrieval

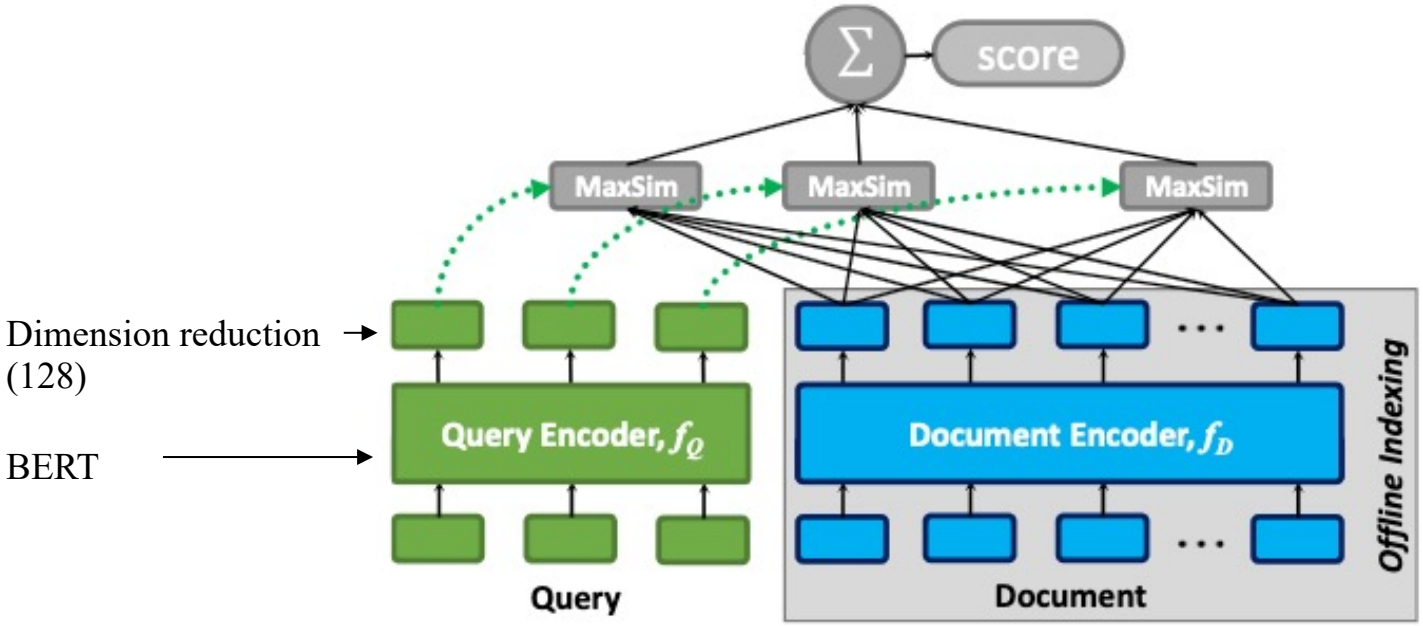


Dense Retrieval

- Objective $P(\text{Relevant} = 1 | d_i, q) \stackrel{\Delta}{=} \phi(\eta_q(q), \eta_d(d_i))$
  - With dense representations $\eta_q(.)$ and $\eta_d(.)$ through encoders
- We can not use of the shelf BERT
  - BERT inference is slow for IR purpose
  - Need $\eta_d(.)$ independent from queries => text representations must be precomputed and stored
  - The similarity function ϕ must be fast by design => ranking in terms of ϕ over a large (precomputed) collection of dense vectors needed => solutions based on nearest neighbor search

Dense Retrieval - ColBERT

- Contextualized Late interaction over BERT



Omar Khattab, Matei Zaharia, ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. ACM SIGIR 2020. arXiv:2004.12832

Dense Retrieval - ColBERT

- Multistage ranking
  - Step 1: for each query token embedding, fast access to the k-nearest full documents embeddings ([CLS] output vector)
  - Step 2 : document-query matching (next slide)

Dense Retrieval - ColBERT

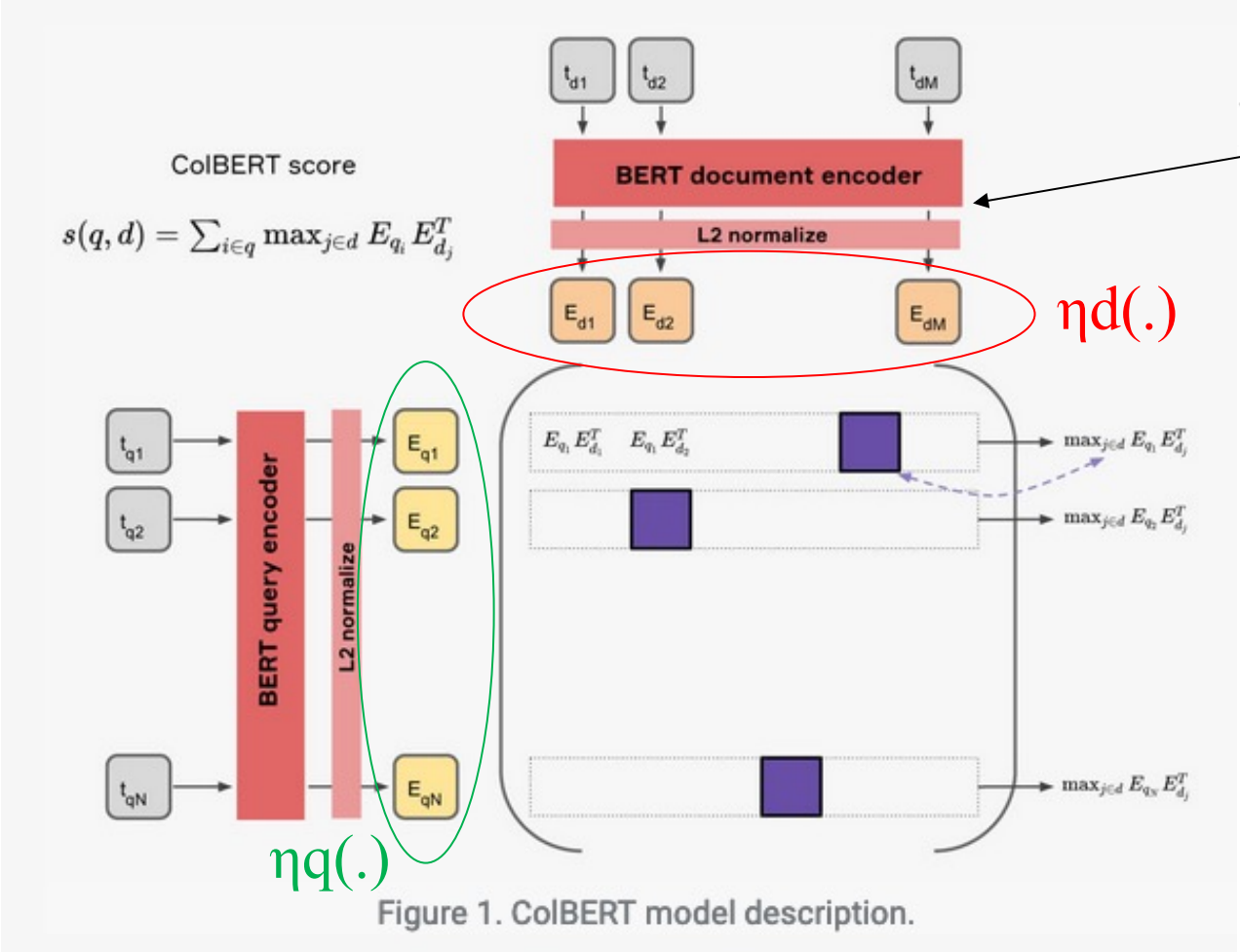


Figure 1. ColBERT model description.

Figure from Navers Labs.

Dense Retrieval - ColBERT

- Results

Method	MS MARCO Passage (Dev)		
	Development		Latency
	MRR@10	Recall@1k	(ms)
BM25 (Anserini, top 1000)	0.187	0.861	62
+ monoBERT$_{Large}$	0.374	0.861	32,900
doc2query–T5	0.277	0.947	87
ColBERT (with BERT$_{Base}$)	0.360	0.968	458

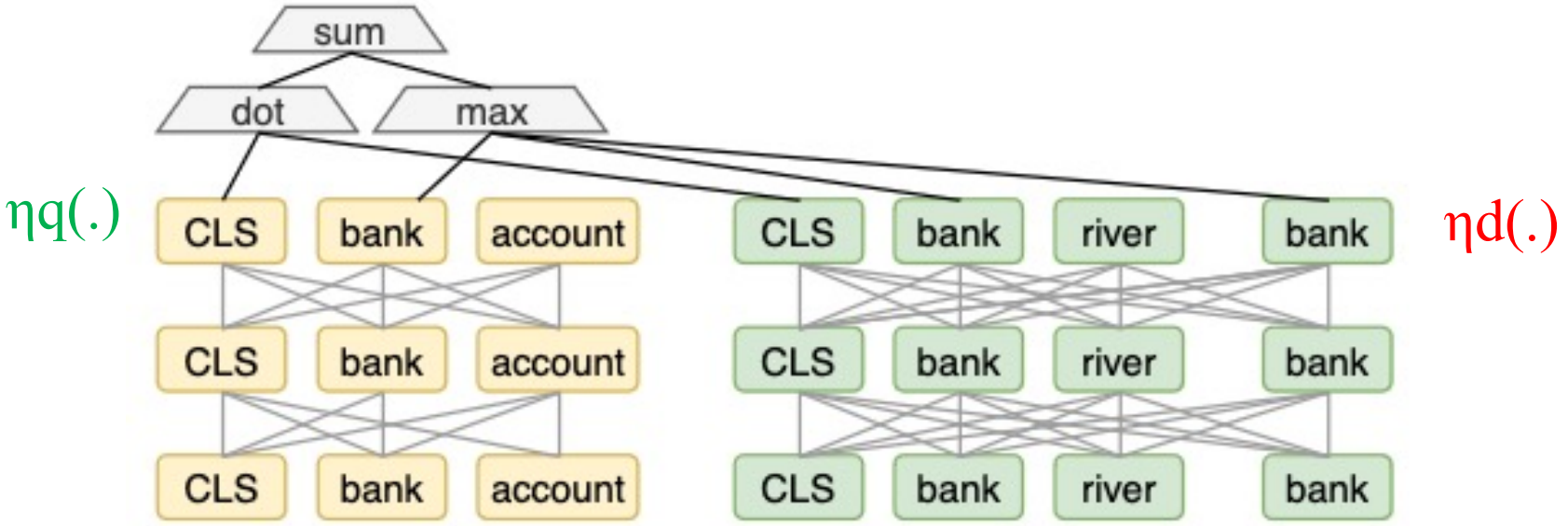
The effectiveness of ColBERT on the development set of the MS MARCO passage ranking test collection. Query latencies for ColBERT and monoBERT$_{Large}$ are measured on a V100 GPU.

Dense Retrieval - ColBERT

- Steps to lower the gap between monoBERT and pre-BERT neural ranking models
- It is able to accomplish this with only modest degradation in effectiveness compared to monoBERT reranking
- One major drawback of ColBERT: the space needed to store the per-token representations of texts from the corpus
  - The use of Fully Connected layer to lower the dimensionality

Dense Retrieval - COIL

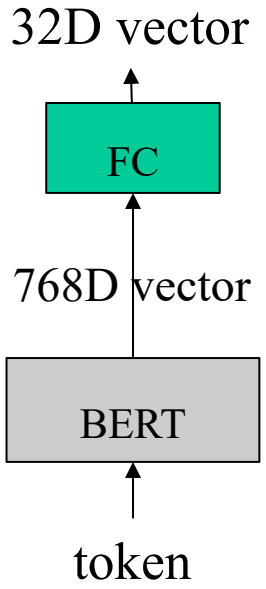
- Contextualized Inverted Lists
  - Propose a retrieval based on token embeddings from BERT, and passage embedding (from [CLS] token)
  - Extend classical inverted files with deep words (token)



Luyu Gao, Zhuyun Dai, Jamie Callan, COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List arXiv:2104.07186

Dense Retrieval - COIL

– The « trick »: to project classical 768D embeddings into lower dimension spaces (typically 32D)

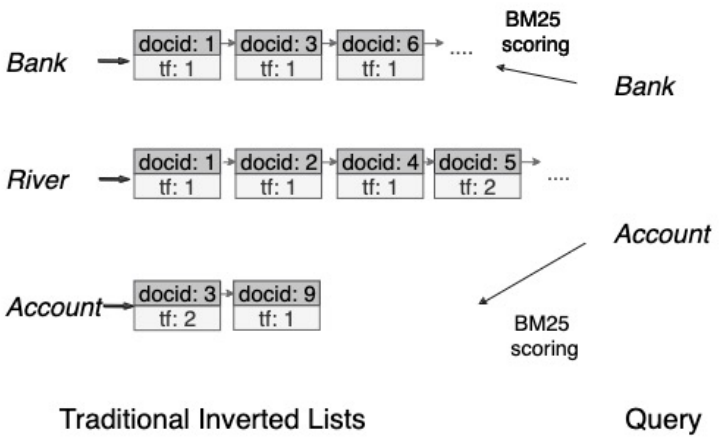


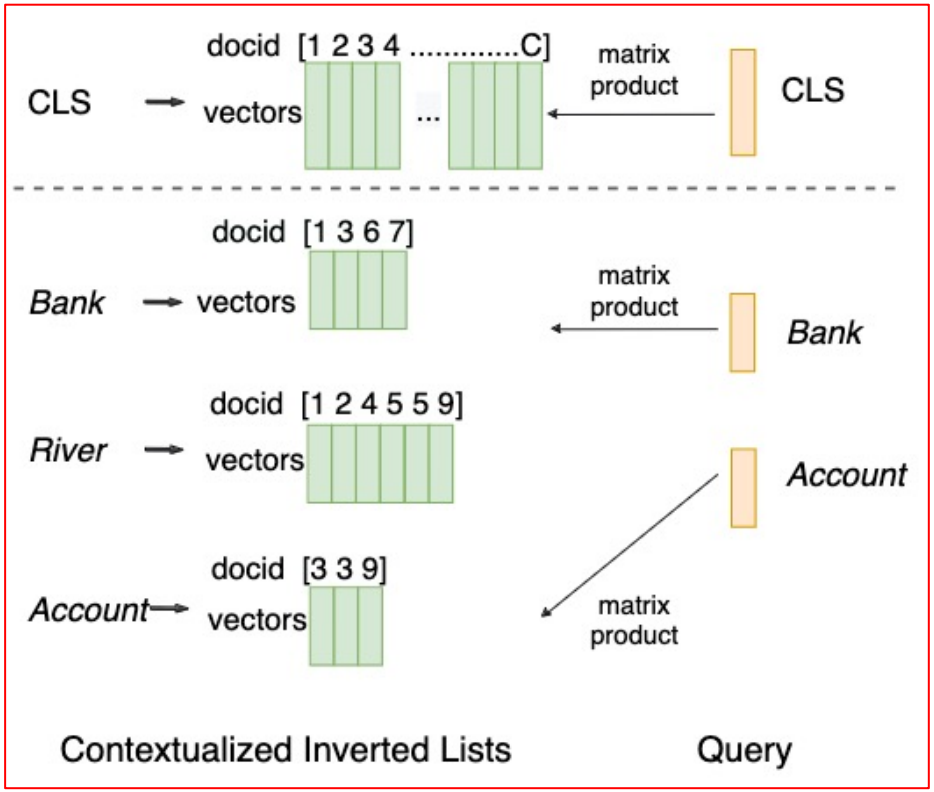
⇒less storage space

- Same idea than ColBERT

Dense Retrieval - COIL

– Inverted files with deep words (token) representations
  - Speed during retrieval





Dense Retrieval - COIL

– Representation of query

$$\boldsymbol{v}_i^q = \boldsymbol{W}_{tok}\text{LM}(q,i) + \boldsymbol{b}_{tok}$$

$$\boldsymbol{v}_{cls}^q = \boldsymbol{W}_{cls}\text{LM}(q,\text{CLS}) + \boldsymbol{b}_{cls}$$

– Representation of document

$$\boldsymbol{v}_j^d = \boldsymbol{W}_{tok}\text{LM}(d,j) + \boldsymbol{b}_{tok}$$

$$\boldsymbol{v}_{cls}^d = \boldsymbol{W}_{cls}\text{LM}(d,\text{CLS}) + \boldsymbol{b}_{cls}$$

– Matching

- Token-only: $s_{\text{tok}}(q,d) = \sum_{q_i \in q \cap d} \max_{d_j = q_i} (\boldsymbol{v}_i^{q\intercal} \boldsymbol{v}_j^d)$
- Full (with CLS): $s_{\text{full}}(q,d) = s_{\text{tok}}(q,d) + \boldsymbol{v}_{cls}^{q\intercal} \boldsymbol{v}_{cls}^d$

- Note: *max* is used also by ColBERT, but on selected terms

Dense Retrieval - COIL

- Results (effectiveness + efficiency)

Model		Dev Retrieval		Latency/ms	
		MRR@10	Recall@1K	CPU	GPU
BM25		0.184	0.853	36	n.a.
ColBERT		0.360	0.968	458*	–
COIL					
n_c	n_t				
768	32	0.355	0.963	380	41
128	32	0.350	0.953	125	23
128	8	0.347	0.956	113	21
0	32	0.341	0.949	67	18
0	8	0.336	0.940	55	16

[CLS] Tokens
Embeddings dimension

Dense Retrieval - COIL

- COIL
  - Good results with lower storage an higher efficiency
  - Based on augmented inverted files using words

- UniCOIL
  - Unicoil reduces the COIL token dimension to 1 => scalar
    - COIL avec n_c=0 et n_t=32 : 0.341
    - uniCOIL n_c=0 et n_t=1 : 0.315

Conclusion

- Still room for higher effectiveness
- Training questions: tranfert learning for specific domains
- Multi-stage retrieval still open
- Better efficiency still open
- +/- Explainability

- Integration with dialogs (chatGPT) : Retrieval Oriented Machine Learning