



Neural Models in Information Retrieval

06/12/2022

Petra Galuščáková

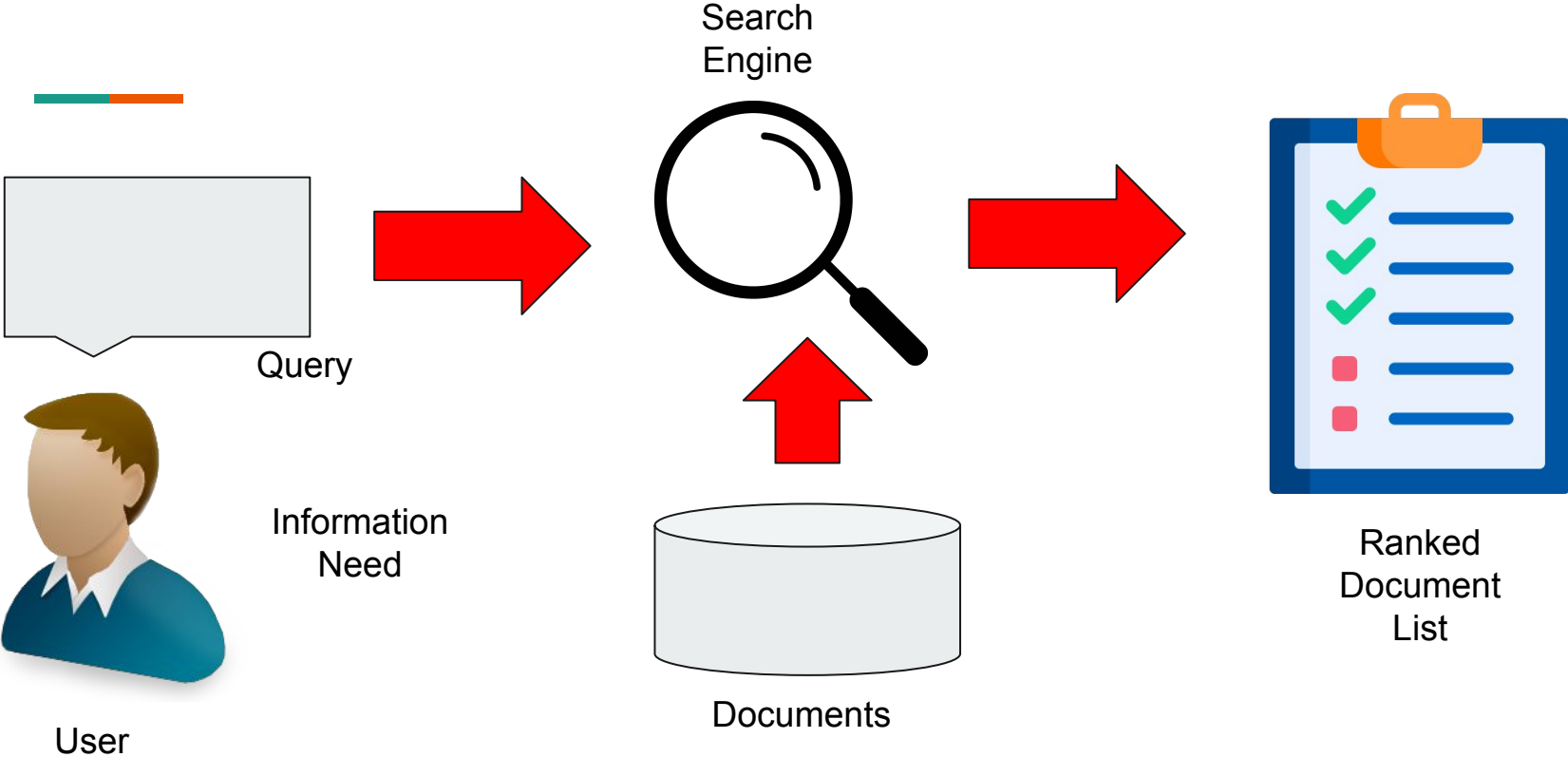
petra.galuscakova@univ-grenoble-alpes.fr



Outline

- From probabilistic models to machine learning models
- Pre-BERT neural models
- BERT in IR
 - MonoBERT
 - Multi-stage architectures
 - Refining query and document representations
 - Dense retrieval

Text Ranking



Probabilistic IR

$$\text{BM25}(q, d) = \sum_{t \in q \cap d} \log \frac{N - \text{df}(t) + 0.5}{\text{df}(t) + 0.5} \cdot \frac{\text{tf}(t, d) \cdot (k_1 + 1)}{\text{tf}(t, d) + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})}$$

inverse document
frequency Term frequency

N = the total number of documents in the corpus

df(t) = the number of documents that contain term t

tf(t, d) = the number of times term t appears in document d

k1 and **b** are free parameters

L = the average document length across all documents in the collection

The Challenges of Probabilistic IR



$$S(q, d) = \sum_{t \in q \cap d} f(t)$$

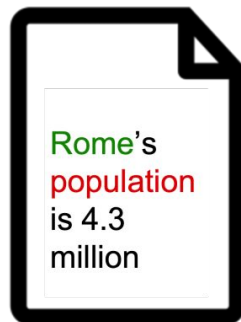
Issues:

Relies exclusively on exact term matching

-> Vocabulary mismatch problem



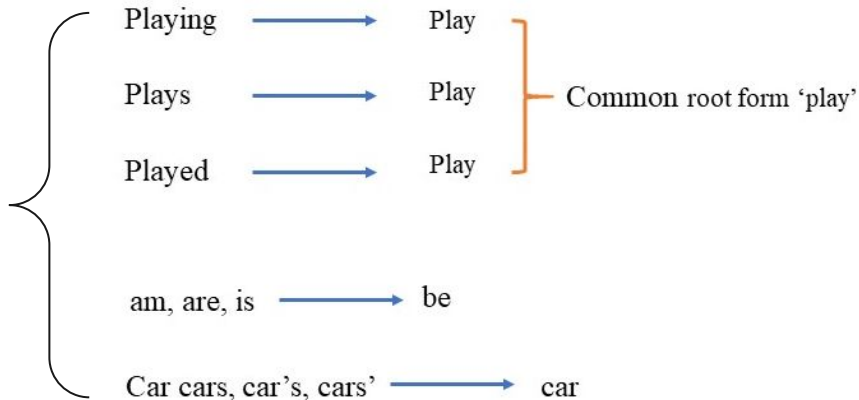
How many people live in Rome?



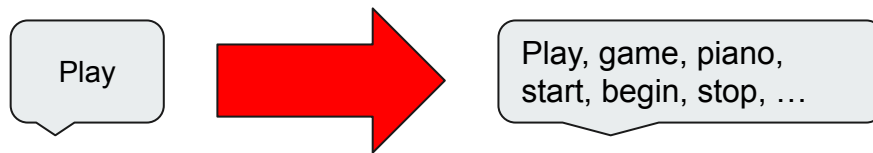
How to Deal with Vocabulary Mismatch Problem?



Stemming and text processing



Enriching query representations

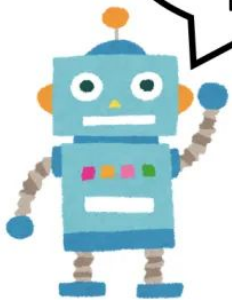


Enriching document representations

Learning to Rank




- Use supervised machine-learning techniques to learn ranking models
- 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 inbound and outgoing links and PageRank scores
 - **Search engines**: how many times users issued a particular query or clicked on a particular link




Click probability is
A: 0.010
B: 0.018

Classification



A: 5 Ways to make a million dollars without working



B: 10 Reasons you Should Drink Milk Every Morning (you won't believe number 7!)



B will rank higher with probability 0.7

Learning to Rank

Learning to Rank - Loss Functions

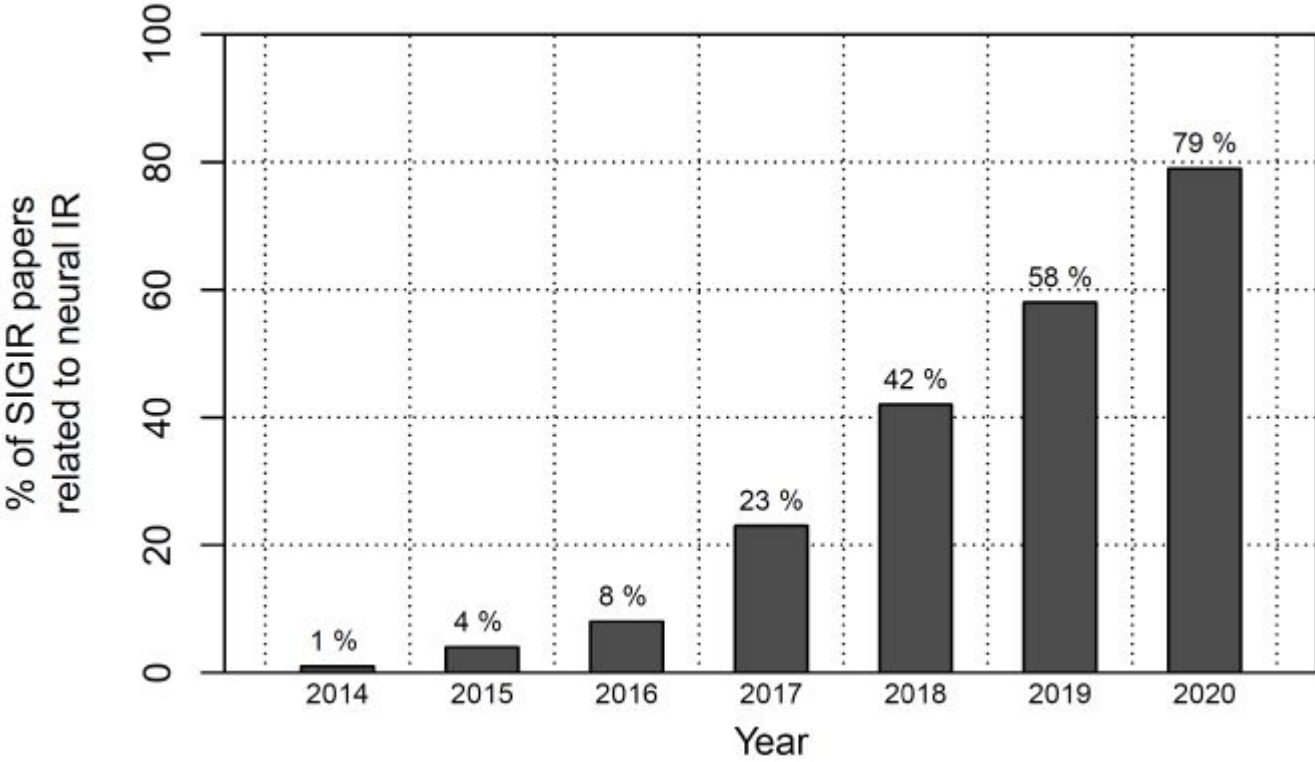


- A **pointwise** approach only considers losses on individual documents, transforming the ranking problem into classification or regression.
- A **pairwise** approach considers losses on pairs of documents, and thus focuses on preferences, that is, the property wherein A is more relevant than (or preferred over) B.
- A **listwise** approach considers losses on entire lists of documents, for example, directly optimizing a ranking metric such as normalized discounted cumulative gain.

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

Pre-BERT Neural Models

#NeuralIR papers at SIGIR





The Advent of Deep Learning

- Freed text retrieval from the bounds of **exact term matching**
- No need for the **hand-crafted features**

- Pre-BERT neural ranking models: representation-based and interaction-based
- BERT revolution ~ 2019 (substantially higher effectiveness)




Representation-based models

- Independently learning **dense vector representations of queries and documents**
- Compared to compute **relevance** via a **simple metric** such as cosine similarity or inner products
- Can be compared at ranking time, allows **document representations** to be computed **offline**

Pre-BERT:

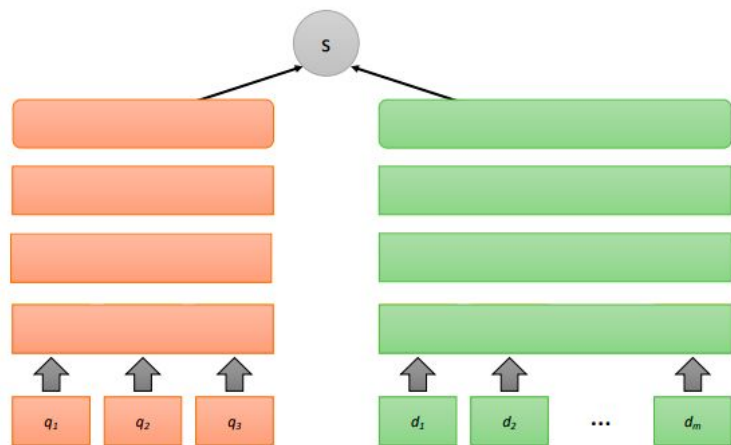
- Deep Structure Semantic Model (DSSM)
 - Constructs character n-grams from an input (i.e., query or document) and passes the results to a series of fully-connected layers to produce a vector representation
- Dual Embedding Space Model (DESM)
 - Represents texts using pre-trained word2vec embeddings and computes relevance scores by aggregating cosine similarities across all query–document term pairs.

Interaction-based models

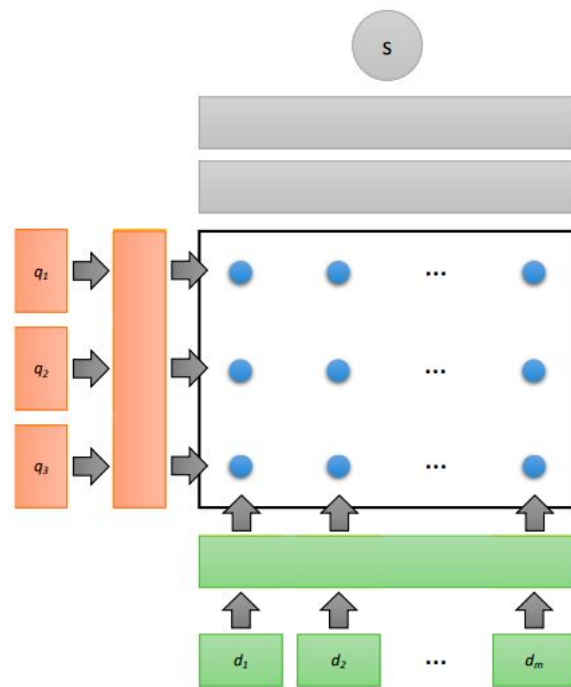
- 
- Compare the representations of terms in the query with terms in a document to **produce a similarity matrix** that captures term interactions.
 - This **matrix** then undergoes further analysis to arrive at a **relevance score**.
 - Each entry $m_{i,j}$ in the matrix is usually populated with the cosine similarity between the **embedding of the i-th query term** and the **embedding of the j-th document term**.

 - Pre-BERT Models: DRMM, KNRM, MarchPyramid, PACRR
 - Pre-BERT interaction-based models were typically more effective but slower than pre-BERT representation-based models.

Representation vs. Interaction-based Models



(a) a generic representation-based neural ranking model



(b) a generic interaction-based neural ranking model



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



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.


BERT in IR



The Arrival of BERT

- BERT [Devlin et al., 2019] coming in October 2018
 - Helpful in many NLP tasks
- First application of BERT on IR in January 2019 [Nogueira and Cho, 2019] -> 30% improvement
- Large amounts of data not necessary, but helpful
- The availability of the MS MARCO collection [Nguyen et al., 2016], further mitigated data availability issues

MS MARCO Collection



At about what age do adults normally begin to lose bone mass?

During childhood and early adulthood, more bone is produced than removed, reaching its maximum mass and strength by the mid-30s. After that, bone is lost at a faster pace than it is formed, so the amount of bone in the skeleton begins to slowly decline.

- Anonymized natural language questions drawn from **Bing's query logs**
- Initially, designed to study **question answering** on web passages, but it was 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 (as assessed by human annotators)
- Prepared "triples": query, relevant passage, non-relevant passage

MS MARCO Collection

Dataset	$ q $	$\bar{L}(q)$	$ J $	$ J /q$	$ \text{Rel} /q$
MS MARCO passage ranking (train)	502,939	6.06	532,761	1.06	1.06
MS MARCO passage ranking (development)	6,980	5.92	7,437	1.07	1.07
MS MARCO passage ranking (test)	6,837	5.85	-	-	-
MS MARCO document ranking (train)	367,013	5.95	367,013	1.0	1.0
MS MARCO document ranking (development)	5,193	5.89	5,193	1.0	1.0
MS MARCO document ranking (test)	5,793	5.85	-	-	-
TREC 2019 DL passage	43	5.40	9,260	215.4	58.2
TREC 2019 DL document	43	5.51	16,258	378.1	153.4
TREC 2020 DL passage	54	6.04	11,386	210.9	30.9
TREC 2020 DL document	45	6.31	9,098	202.2	39.3
Robust04	249	(title) 2.67 (narr.) 15.32 (desc.) 40.22	311,410	1250.6	69.9

Table 3: Summary statistics for select queries and relevance judgments used by many text ranking models presented in this survey. For Robust04, we separately provide average lengths of the title, narrative, and description fields of the topics. Note that for the TREC 2019/2020 DL data, relevance binarization is different for passage vs. documents; here we simply count all judgments that have a non-zero grade.



BERT in IR

The simplest and most straightforward formulation of text ranking: **convert the task into a text classification problem.**

Sort the texts to be **ranked** based on the **probability** that each item **belongs to the desired class.**

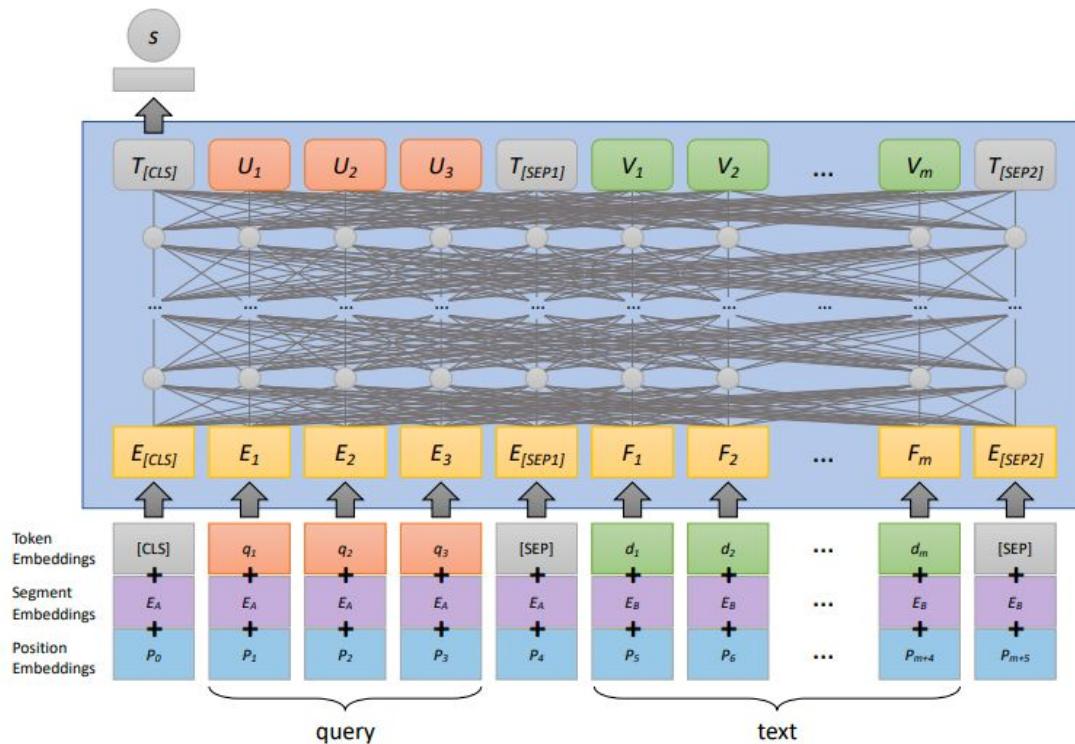
Train a classifier to estimate the probability that each text belongs to the **“relevant” class**, and then at ranking (i.e., inference) time sort the texts by those estimates.

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

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

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

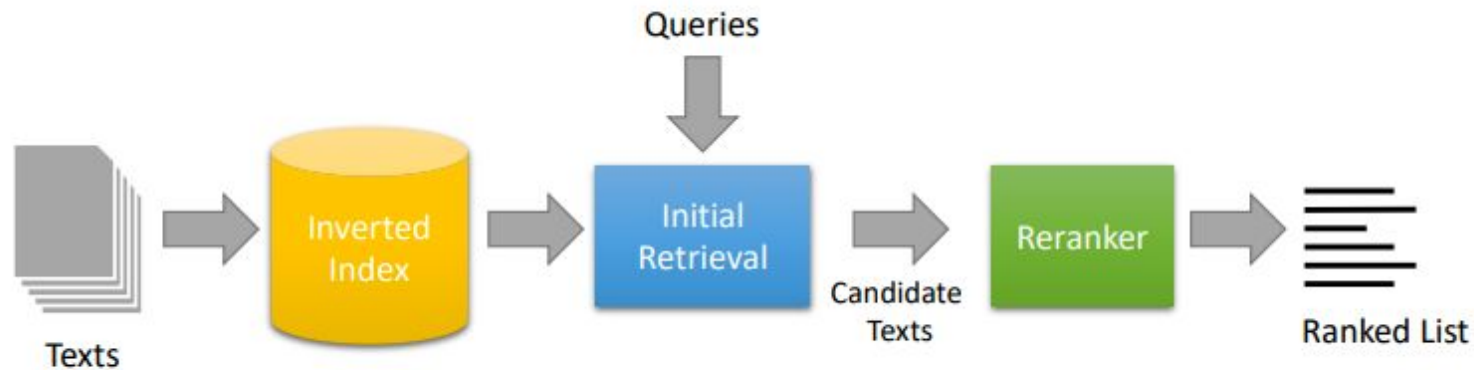
monoBERT



$$P(\text{Relevant} = 1 | d_i, q) = s_i \triangleq \text{softmax}(T_{[CLS]}W + b)_1,$$

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

Reranking using monoBERT



monoBERT: Results

Method		MS MARCO Passage		
		Development		Test
		MRR@10	Recall@1k	MRR@10
(1)	IRNet (best pre-BERT)	0.278	-	0.281
(2a)	BM25 (Microsoft Baseline, $k = 1000$)	0.167	-	0.165
(2b)	+ monoBERT _{Large} [Nogueira and Cho, 2019]	0.365	-	0.359
(2c)	+ monoBERT _{Base} [Nogueira and Cho, 2019]	0.347	-	-
(3a)	BM25 (Anserini, $k = 1000$)	0.187	0.857	0.190
(3b)	+ monoBERT _{Large} [Nogueira et al., 2019a]	0.372	0.857	0.365
(4a)	BM25 + RM3 (Anserini, $k = 1000$)	0.156	0.861	-
(4b)	+ monoBERT _{Large}	0.374	0.861	-

Table 5: The effectiveness of monoBERT on the MS MARCO passage ranking test collection.

The Effectiveness of monoBERT

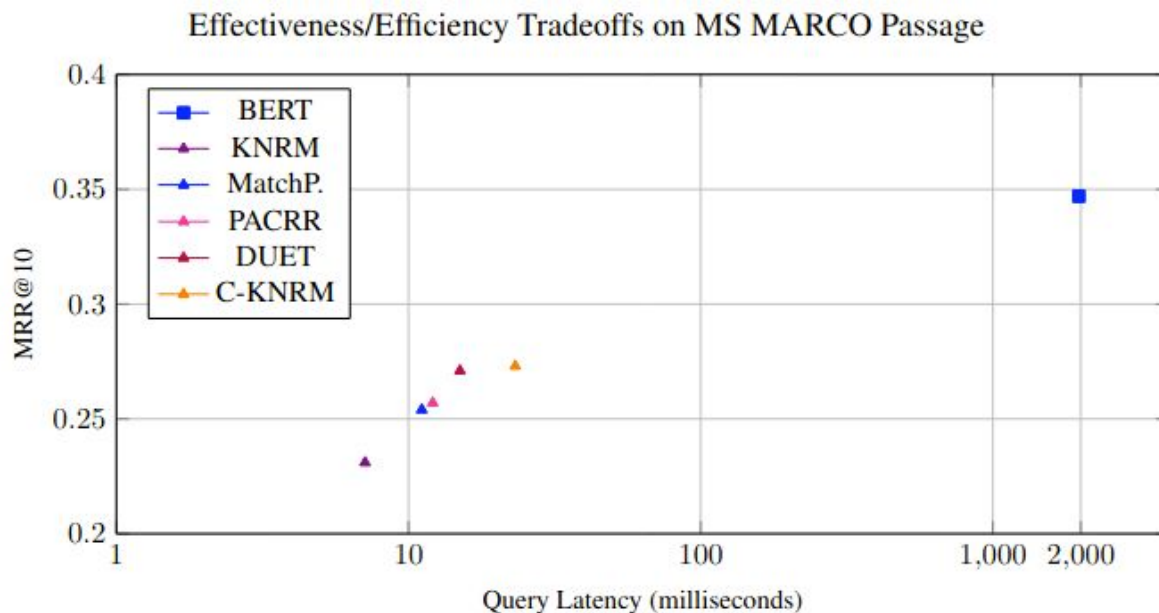
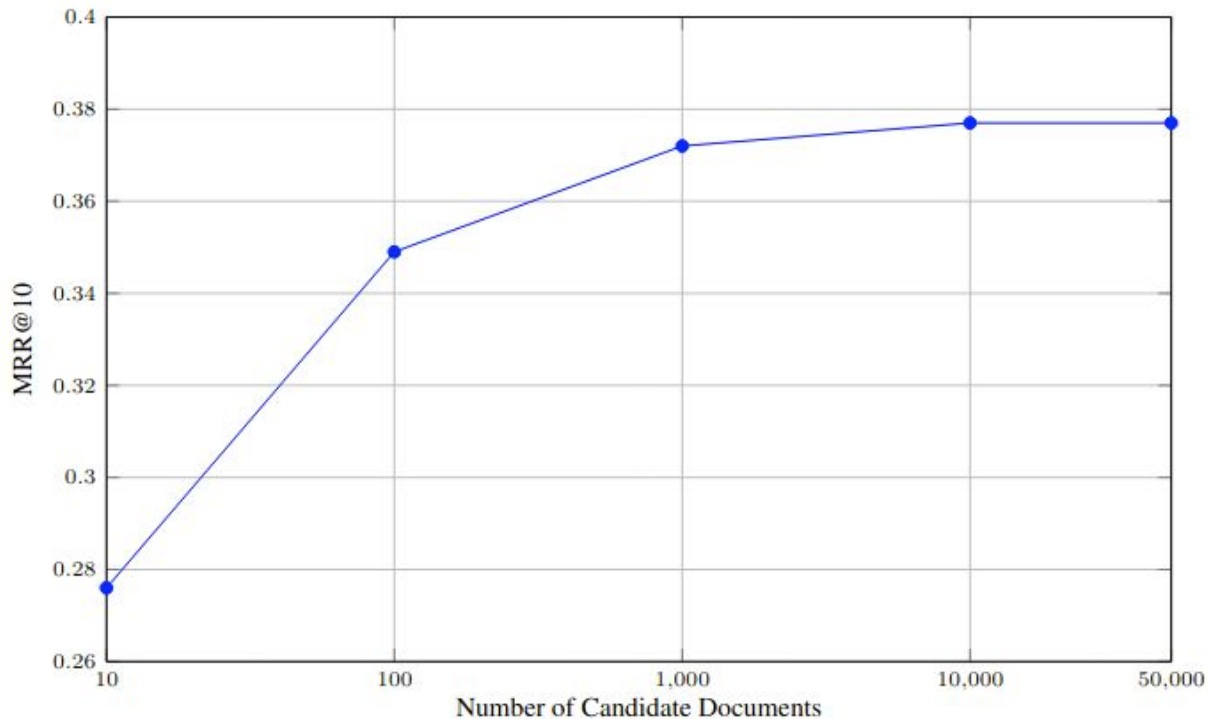


Figure 16: Effectiveness/efficiency tradeoffs comparing BERT with pre-BERT models (using FastText embeddings) on the development set of the MS MARCO passage ranking test collection, taken from Hofstätter and Hanbury [2019]. Note that the x -axis is in log scale.

Effects of Reranking Depth



monoBERT_{Large} Effectiveness vs. Reranking Depth on MS MARCO Passage



The Effectiveness of monoBERT





From Passage to Document Ranking

- monoBERT is limited to ranking paragraph-length passages, not longer documents
- How to deal with this:
 - Only use first n characters/sentences/paragraphs from the document
 - Avoid the problem by using transfer learning
 - Aggregation during the inference:
 - Scores
 - Representations
 - No guarantee that the segments are relevant in training

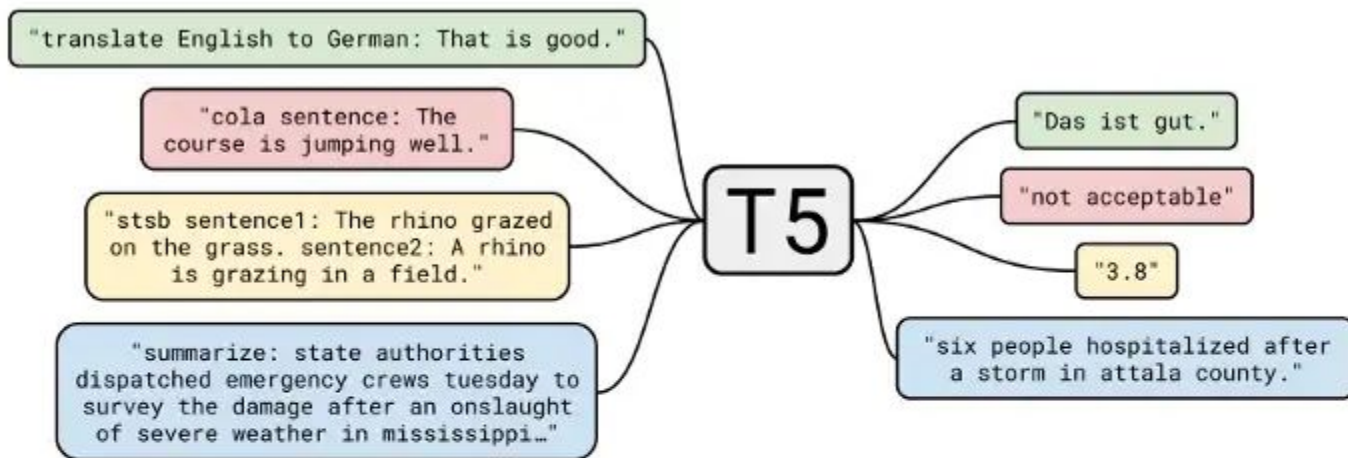
T5 in IR

T5

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

But it is adapted for the **sequence-to-sequence** context.

Just like in BERT, 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.



T5 for IR

[Nogueira et al., 2020]



Use the following input template:

Query: [q] Document: [d] Relevant:

where [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**” depending on whether the document is relevant or not to the query. That is, “true” and “false” are the “target tokens” (i.e., ground truth predictions in the sequence-to-sequence transformation).

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

T5 vs. BERT

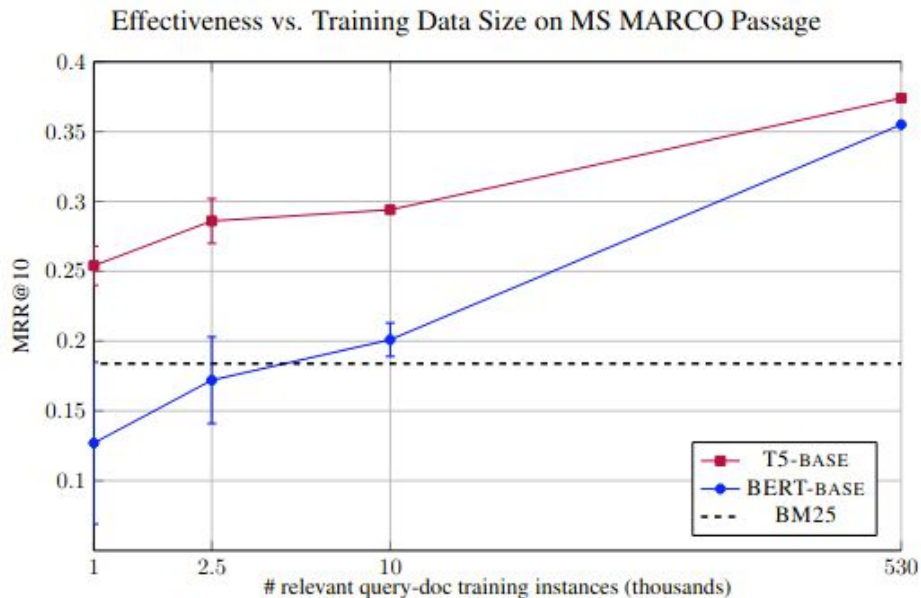


Figure 19: The effectiveness of monoT5-base and monoBERT_{Base} on the development set of the MS MARCO passage ranking test collection varying the amount of training data used to fine-tune the models. Results report means and 95% confidence intervals over five trials. Note that the x -axis is in log scale.

Multi-Stage Architectures for Reranking

Multi-Stage Architectures for Reranking

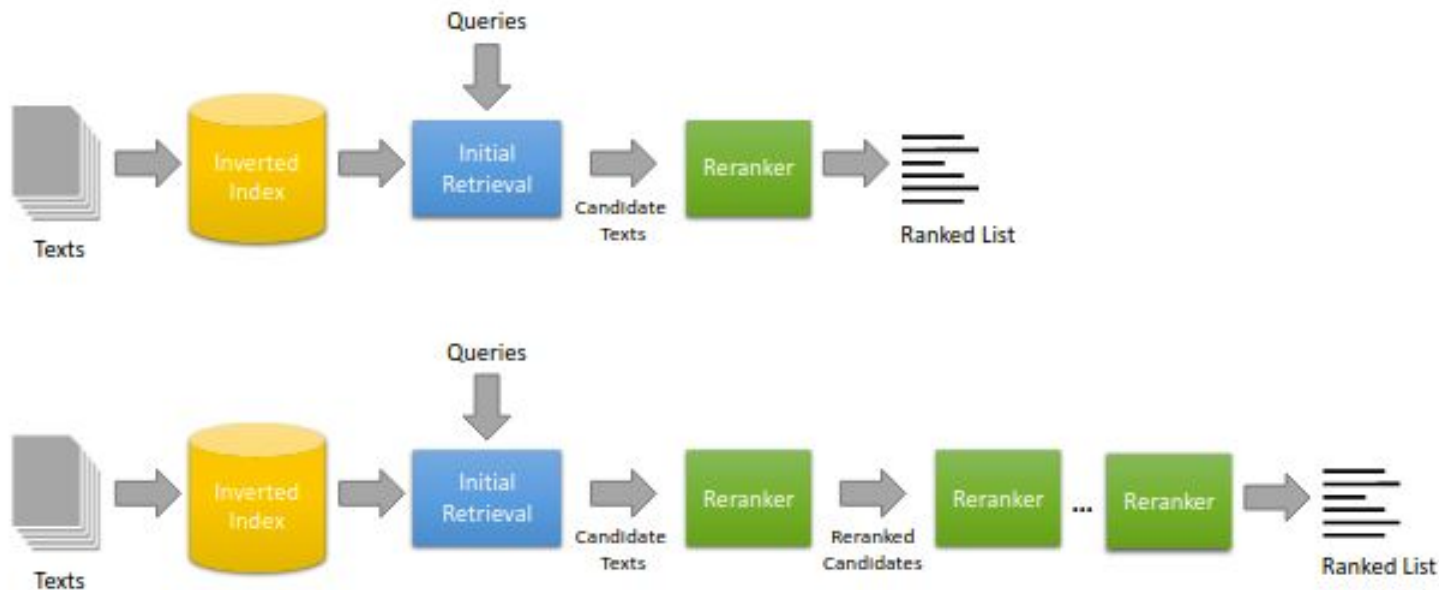


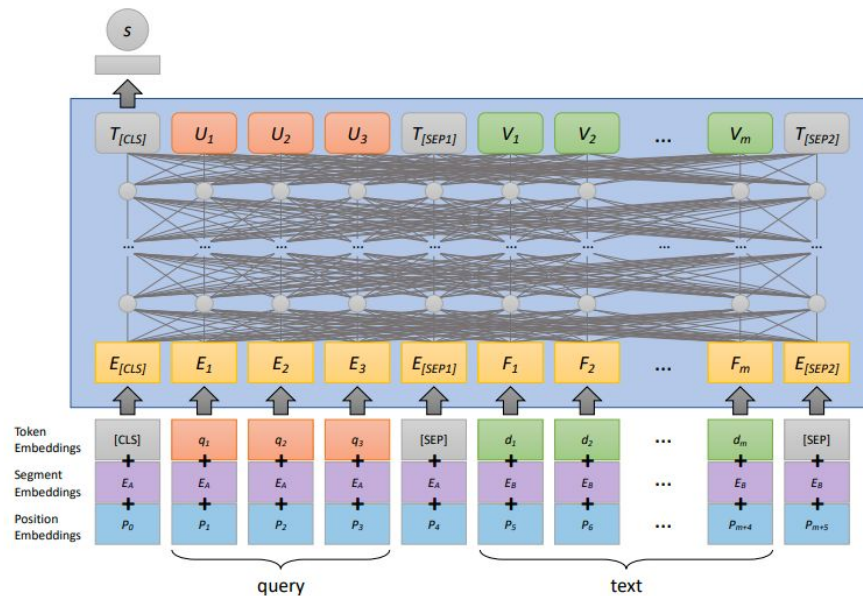
Figure 14: A retrieve-and-rerank design (top) is the simplest instantiation of a multi-stage ranking architecture (bottom). In multi-stage ranking, the candidate generation stage (also called initial retrieval or first-stage retrieval) is followed by more than one reranking stages.



Multi-Stage Architectures for Reranking

- 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.

Pointwise Reranking



$$P(\text{Relevant} = 1 | d_i, q) = s_i \triangleq \text{softmax}(T_{[CLS]}W + b)_1,$$

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

Pairwise Reranking



$$P(d_i \succ d_j | d_i, d_j, q),$$

where $d_i \succ d_j$ is a notation for stating that d_i is more relevant than d_j (with respect to the query 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:

$$[[\text{CLS}], q, [\text{SEP}], d_i, [\text{SEP}], d_j, [\text{SEP}]], \quad (31)$$

duoBERT [Nogueira et al., 2019]



- 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, the **query, candidates d_i and d_j are truncated** to 62, 223, and 223 tokens, respectively.

duoBERT: Scores Aggregation



$$\text{MAX : } s_i = \max_{j \in J_i} p_{i,j},$$

$$\text{MIN : } s_i = \min_{j \in J_i} p_{i,j},$$

$$\text{SUM : } s_i = \sum_{j \in J_i} p_{i,j},$$

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

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

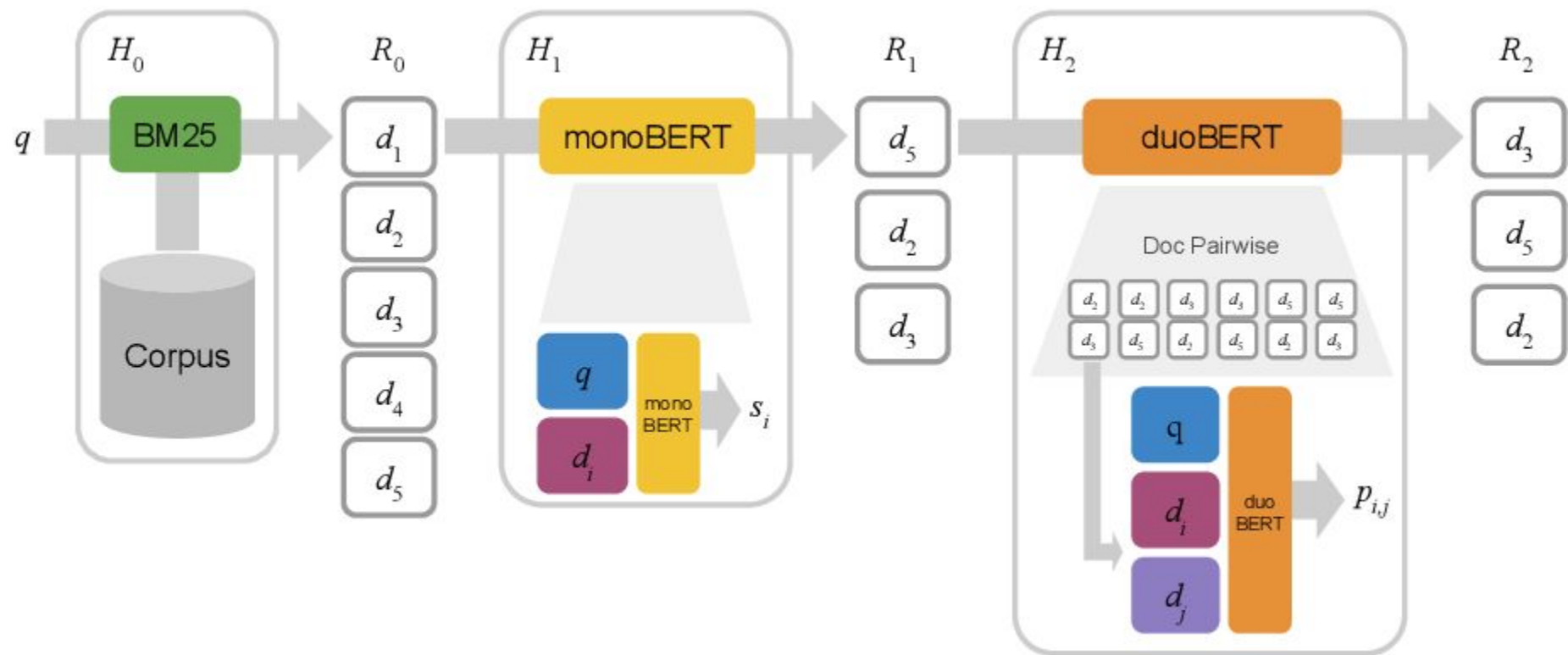
The **BINARY** method is inspired by the Condorcet method

The **MIN (MAX)** method measures the **relevance** of di only against its **strongest (weakest)** “competitor”.


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.

duoBERT [Nogueira et al., 2019]



duoBERT [Nogueira et al., 2019]



Method		MS MARCO Passage	
		Development MRR@10	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 _{MAX} ($k_1 = 50$)	0.326	-
(3b)	+ duoBERT _{MIN} ($k_1 = 50$)	0.379	-
(3c)	+ duoBERT _{SUM} ($k_1 = 50$)	0.382	0.370
(3d)	+ duoBERT _{BINARY} ($k_1 = 50$)	0.383	-
(4a)	+ monoBERT + TCP	0.379	-
(4b)	+ monoBERT + duoBERT _{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.

Refining Query and Document Representations



The vocabulary mismatch problem

- The searchers and the authors of the texts to be searched use different words to describe the same concept
- The relevant text that has no overlap with query terms will not be retrieved, and hence will **never be encountered by any of the downstream rerankers**.

- As an example, a text discussing **automobile sales** might be expanded with the term “**car**” to better match the query “**car sales per year in the US**”.

Doc2query [Nogueira et al., 2019]

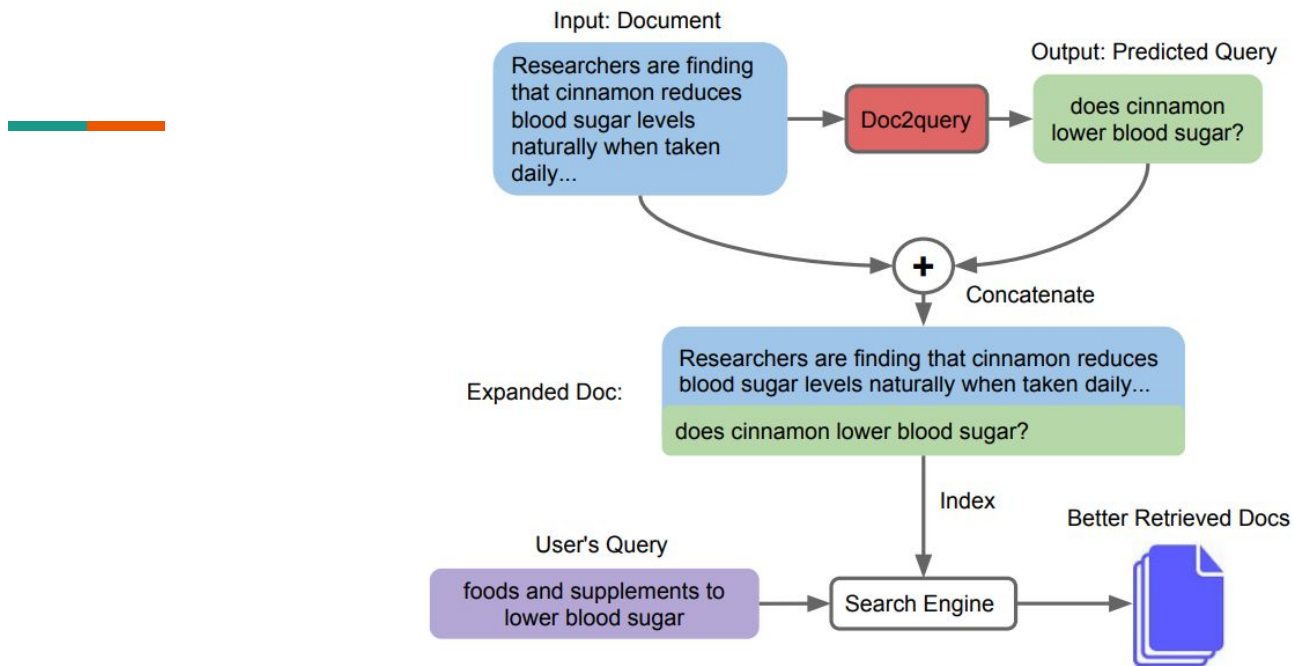


Figure 1: Given a document, our Doc2query model predicts a query, which is appended to the document. Expansion is applied to all documents in the corpus, which are then indexed and searched as before.

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.

Doc2Query: Results



Method	MS MARCO Passage			Latency (ms/query)
	Development MRR@10	Recall@1k	Test MRR@10	
(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	-	-
(3) Best non-BERT [Hofstätter et al., 2019]	0.290	-	0.277	-
(4) BM25 + monoBERT _{Large} [Nogueira et al., 2019a]	0.372	0.853	0.365	3,500

Table 31: The effectiveness of doc2query on the MS MARCO passage ranking test collection.

Term Reweighting as Regression: DeepCT

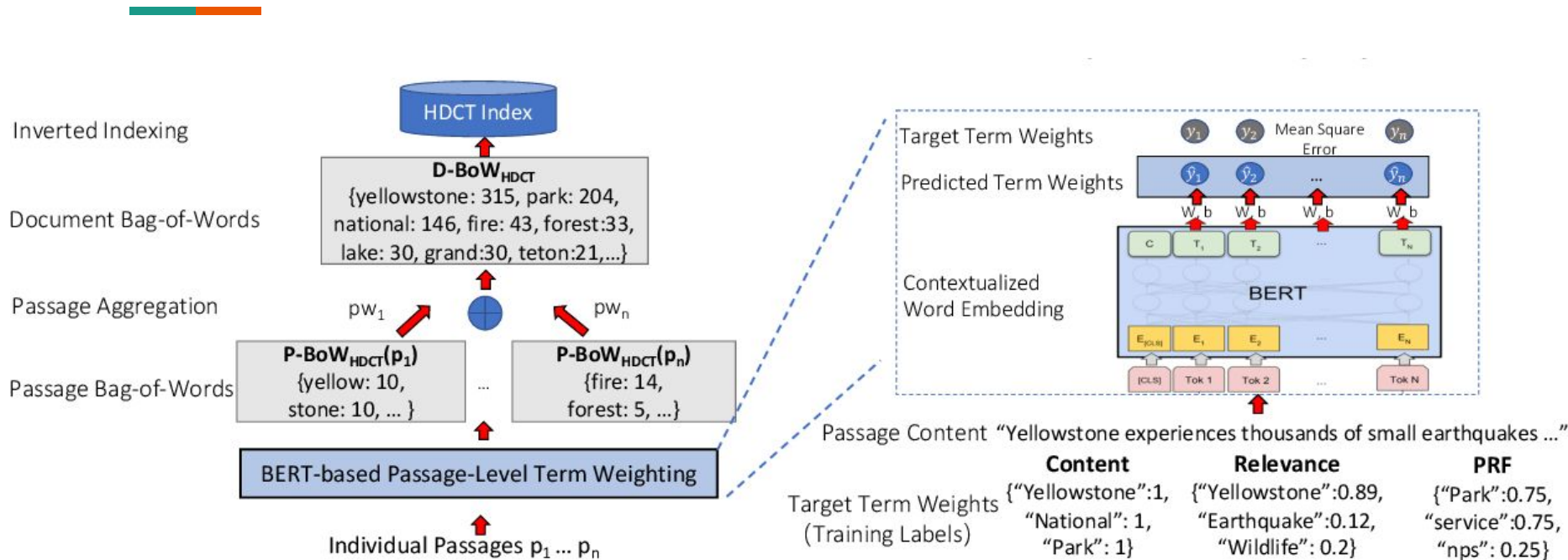
[Dai and Callan, 2019]

What if we were able to directly 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. Its 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 .

Figure 21: Motivating examples for DeepCT, which show passages containing query terms that appear in both relevant and non-relevant contexts, taken from Dai and Callan [2019a].

Term Reweighting as Regression: DeepCT [Dai and Callan, 2019]



Term Reweighting as Regression: DeepCT


[Dai and Callan, 2019]



- 1) Inference is applied to compute a **weight for each term** in each text from the corpus.
 - 2) These weights are then **rescaled** from $[0..1]$ to integers between 0 and 100 so they resemble term frequencies in standard bag-of-words retrieval methods.
 - 3) Finally, the texts are **indexed** using these **rescaled** term weights
-
- New “**pseudo-documents**” are created in which **terms are repeated** the same number of times as their importance weights.
 - For example, if the term “**boyle**” is assigned a weight of **four**, it is repeated four times, becoming “**boyle boyle boyle boyle**” in this new pseudo-document.
 - A **new corpus comprising these pseudo-documents**, in which the repeated terms are concatenated together, is then indexed like any other corpus.
 - Retrieval is performed on this index as with any other bag-of-words query, although it is important to retune parameters in the scoring function.

Term Reweighting as Regression: DeepCT

[Dai and Callan, 2019a]



Method	MS MARCO Passage		
	Development MRR@10	Recall@1k	Test MRR@10
(1a) BM25	0.184	0.853	0.186
(1b) w/ doc2query-base	0.218	0.891	0.215
(1c) w/ doc2query-T5	0.277	0.947	0.272
(1d) w/ doc2query-T5 (copied terms only)	0.221	0.893	-
(2) DeepCT	0.243	0.913	0.239

Table 34: The effectiveness of DeepCT on the MS MARCO passage ranking test collection.



DeepImpact [Mallia et al., 2021]

- Doc2query + DeepCT
- Impact Index: Directly store the quantized weight in the term frequency positions
- “Classic” index might be used -> fast inference

COIL [Gao et al. 2021] and UniCOIL [Lin and Ma, 2021]



- COIL
 - Produces **representations** for each document token that are then directly stored in the inverted index
 - Instead of assigning scalar weights to terms in a query, the “scoring” model **assigns each term a vector “weight”**
 - Query evaluation in COIL involves accumulating inner products instead of scalar weights
 - Best performing sparse model

- uniCOIL
 - Reduce the token **dimension** of COIL to **one** (use scalar weights)

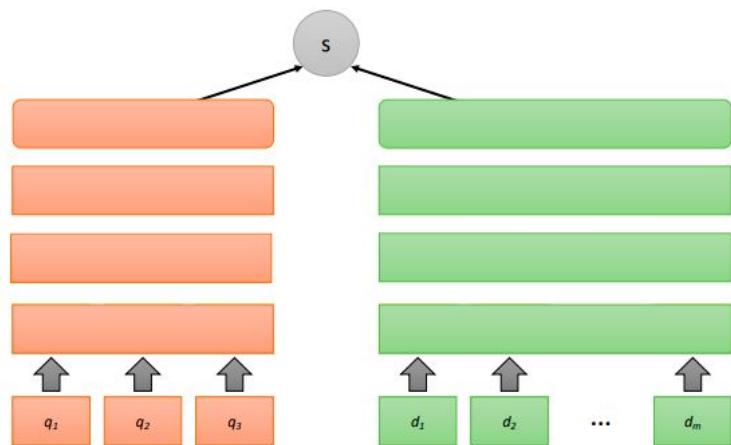
Dense Retrieval



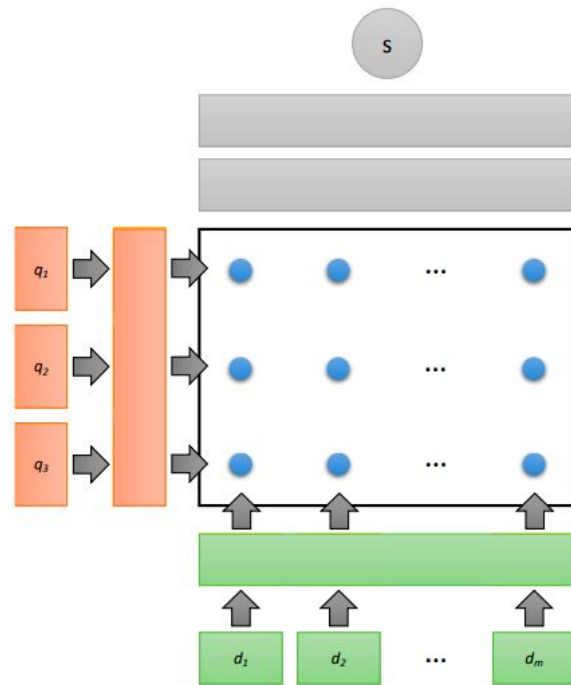
Motivations

- BERT inference is slow.
 - $\eta(d)$ is not dependent on queries, what means that text representations can be precomputed and stored
 - The similarity function ϕ is fast by design and ranking in terms of ϕ over a large (precomputed) collection of dense vectors is typically amenable to solutions based on nearest neighbor search

Representation vs. Interaction-based Models




(a) a generic representation-based neural ranking model



(b) a generic interaction-based neural ranking model

Dense Retrieval



We estimate: $P(\text{Relevant} = 1 | d_i, q) \triangleq \phi(\eta_q(q), \eta_d(d_i)),$

the relevance of a text with respect to a query.

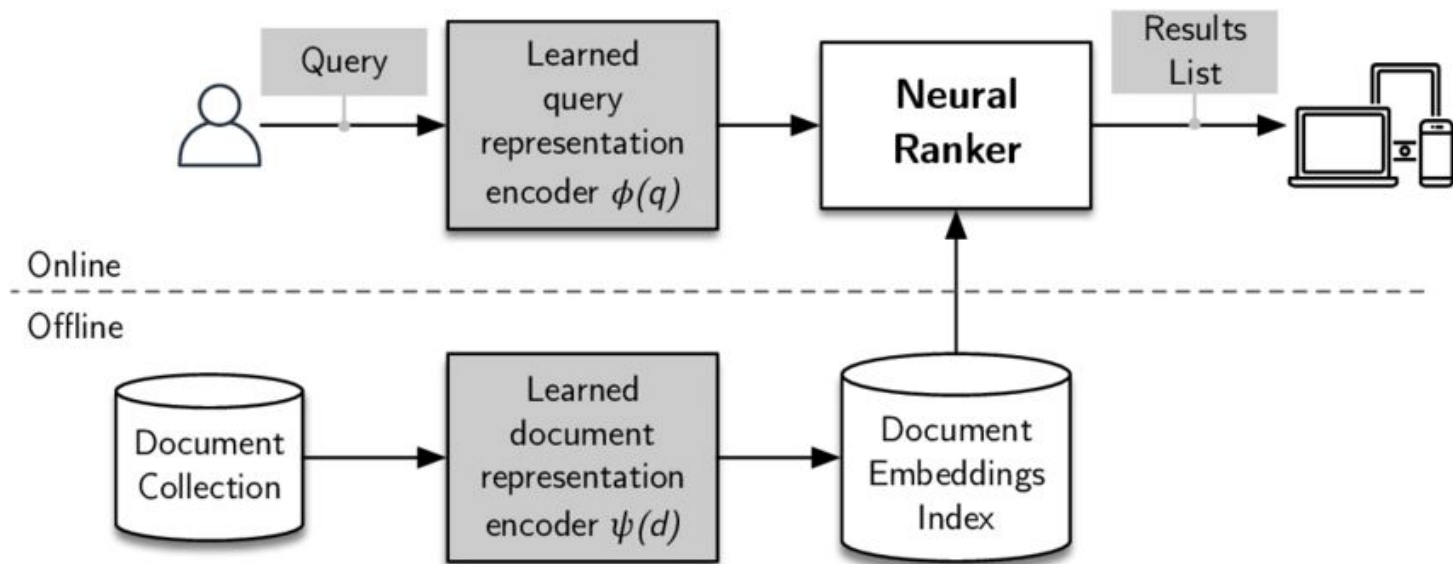
Thus, dense retrieval techniques need to address two challenges:

- 1) the **representation problem**, or the **design of the encoders η** , to accurately capture the “meaning” of queries and texts from the corpus for the purposes of ranking; and
- 2) the comparison problem, or the **design of ϕ** , which involves a balance between what can be **efficiently** computed at scale and what is necessary to **capture relevance** in terms of the dense representations.

The similarity function is most commonly defined to be the **inner product** between the representation vectors

We refer to this as a “**bi-encoder**” design, which contrasts with a “cross-encoder”, which is the standard BERT design that benefits from all-to-all attention across tokens in the input sequence.

Dense Retrieval



Sentence-BERT [Reimers and Gurevych, 2019]

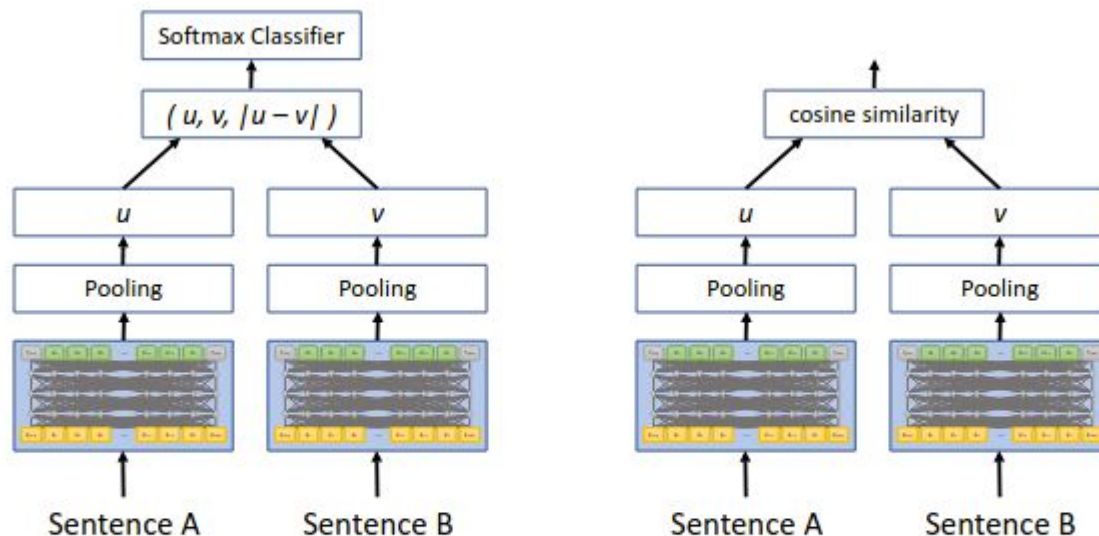


Figure 23: The architecture of Sentence-BERT, redrawn from Reimers and Gurevych [2019]. The training architecture for the classification objective is shown on the left. The architecture for inference, to compute similarity scores, is shown on the right.

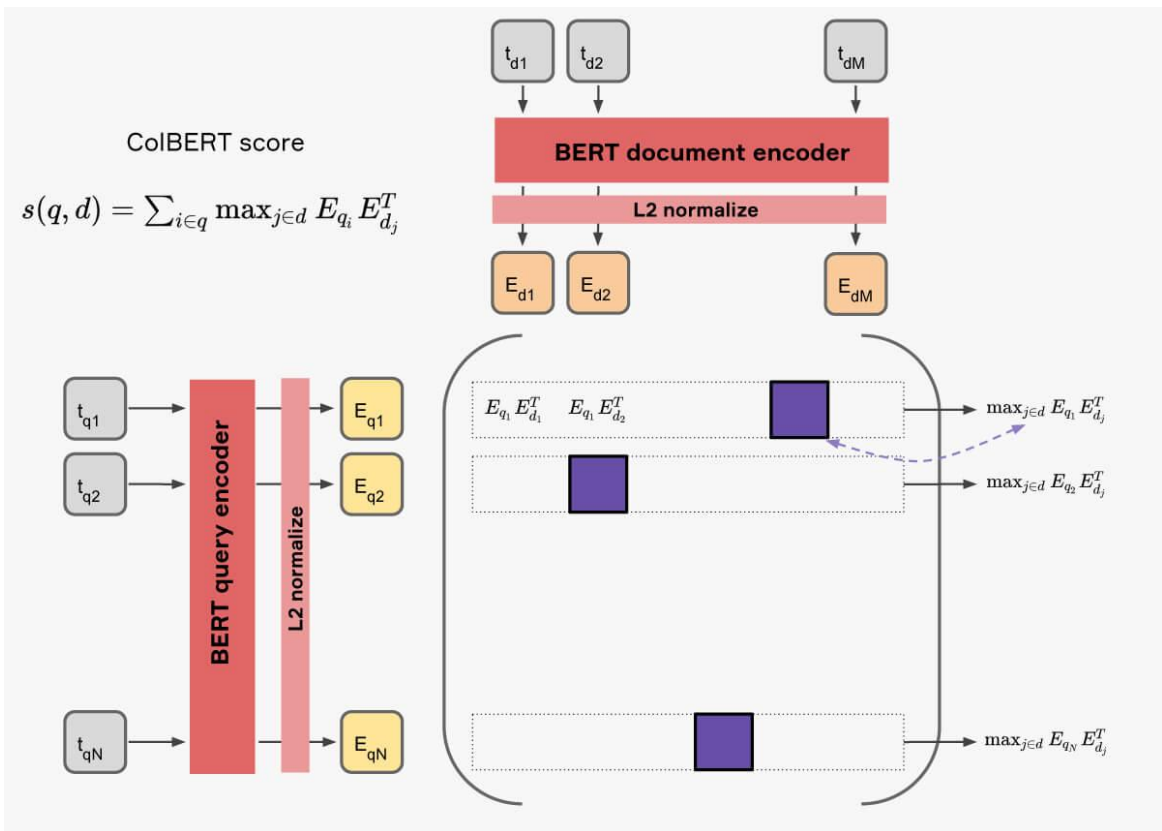


SentenceBERT Results

- **Without any fine-tuning**, average pooling of BERT's contextual representations appears to be **worse** than average pooling of static GloVe embeddings, based on standard metrics for semantic similarity datasets.
- Out-of-domain fine-tuning leads to large gains
- Further in-domain fine-tuning provides an additional boost in effectiveness, consistent with the multi-step fine-tuning approaches

Per-Token Representations and Late Interactions: ColBERT

[Khattab and Zaharia, 2020]



Per-Token Representations and Late Interactions: ColBERT


[Khattab and Zaharia, 2020]



- As a preprocessing step, the **representation** of each token from the corpus is indexed using Facebook's **Faiss** library for nearest neighbor search, where each vector **retains a pointer back to its source** (i.e., the text from the corpus that contains it).
- At query time, ranking proceeds as follows:
 - In the **first stage**, each query term embedding $\eta(q_i)$ is issued concurrently as a query and the top k texts from the corpus are retrieved by following the pointer of each retrieved term vector back to its source.
 - In the **second stage**, these candidate texts are scored using all query token representations according to the MaxSim operator

Per-Token Representations and Late Interactions: ColBERT

[Khattab and Zaharia, 2020]



Method		MS MARCO Passage (Dev)		
		Development		Latency
		MRR@10	Recall@1k	(ms)
(1a)	BM25 (Anserini, top 1000)	0.187	0.861	62
(1b)	+ monoBERT _{Large}	0.374	0.861	32,900
(2)	FastText + ConvKNRM	0.290	-	90
(3)	doc2query-T5	0.277	0.947	87
(4)	ColBERT (with BERT _{Base})	0.360	0.968	458

Table 44: 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.

Per-Token Representations and Late Interactions: ColBERT

[Khattab and Zaharia, 2020]



- ColBERT has **closed** much of 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



Further Reading

Jimmy Lin, Rodrigo Nogueira, Andrew Yates: **Pretrained Transformers for Text Ranking: BERT and Beyond**, Synthesis Lectures on Human Language Technologies, Morgan & Claypool, October 2021.

Thesis Proposal / Internship

Supervisors: Philippe Mulhem, Petra Galuscakova
Team : MRIM group, LIG laboratory <https://lig-mrim.imag.fr/>
Duration : 5 months
Masters: MOSIG DSAI, MSIAM

Currently, many state of the art IR approaches stack multiple retrieval processes. For instance, the best performing models at the MSMARCO Deep Learning Track rely on reranking, i.e. the first retrieval stage is done on a large corpus composed of millions of documents, and then the second retrieval stage is applied on the top-results of the first retrieval. Implicitly, such stacked retrieval process is based on the following hypotheses:

H1-Efficiency: Some retrieval models are much faster, and thus much more able to process large sets of documents in a reasonable time, than others ;
H2- Effectiveness: Some retrieval models perform better than others.

With respect to these hypotheses, stacking employs efficient and somewhat effective models in its first retrieval stage, and the most effective systems in the second stage, to achieve good results on large sets of documents. Of course, a simple sequence stacking is not the only possible combination being applied. Different retrieval approaches may be applied in parallel and merged, as we did in previous experiments. Moreover, these models may be combined with the recent approaches such as Colbert and Unicoil, which tend to be efficient while still being highly effective.

The goal of this internship is to define a framework able to propose stackings generation based on the features of the models considered, and then experiment the proposed stacking to verify the improvement achieved.

Formally, the goals of this work is thus following:

- i) Formulate a set of hypotheses which may define a stacked retrieval
- ii) Formalize the stacking processes according to the hypotheses formulated in i)
- iii) Perform experiments with different stacking retrieval setups using several common IR test collections, such as TREC-DL 2021