ReNeuIR Workshop on Reaching Efficiency in Neural Information Retrieval

Efficient Machine Learning and Machine Learning for Efficiency in Information Retrieval

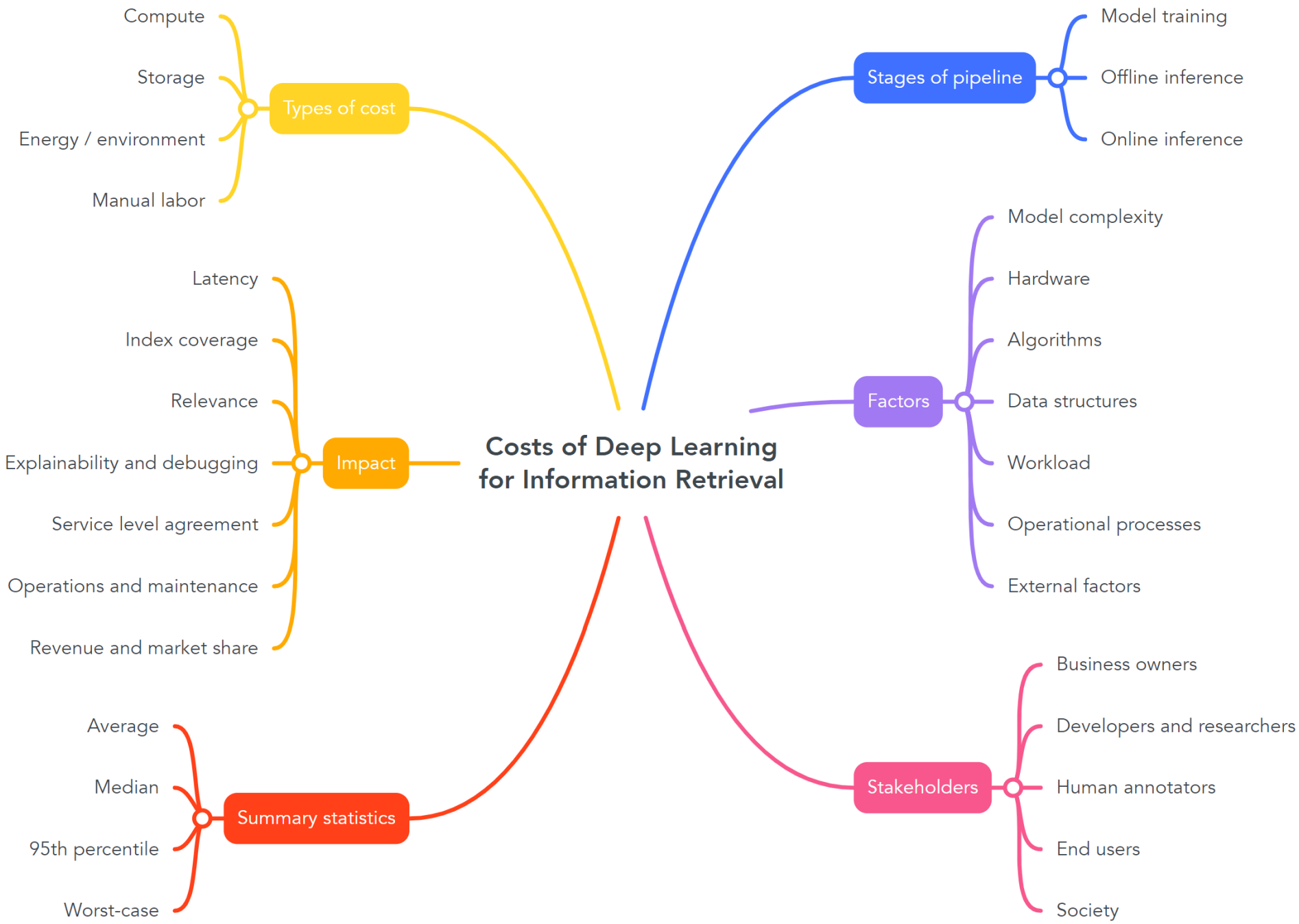
Bhaskar Mitra
Principal Researcher
Microsoft Research

@UnderdogGeek bmitra@microsoft.com

efficient (“working quickly and without waste”)

effective (“having the desired effect”)

Source: https://en.wiktionary.org/wiki/efficient



Key points:

- Complex trade-offs between different costs makes benchmarking challenging
- Typical playbook for efficiency: find cheaper approximation, trade-off other costs, reduce task
- Machine learning can be employed to directly improve efficiency of large-scale IR systems

(Some relevant examples from our NLP peers)

Challenges of benchmarking for efficiency

What do we measure and what do we control for? How do we capture trade-offs between measures?

What stages of the pipeline can we evaluate?

How do we test the benchmark for construct validity and prevent negative externalities?

SustaiNLP (2020) Shared Task
Website: https://sites.google.com/view/sustainlp2020/shared-task
Paper: https://aclanthology.org/2020.sustainlp-1.24.pdf

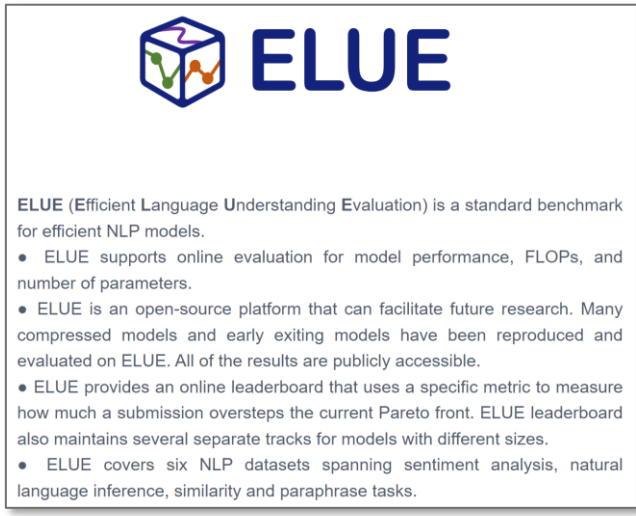
Benchmarks effectiveness and **inference-time energy-consumption**

NeurIPS 2020 EfficientQA Competition
Website: https://efficientqa.github.io/
Paper: http://proceedings.mlr.press/v133/min21a/min21a.pdf

Benchmarks effectiveness while grouping submitted systems by their **inference-time docker image size**

ELUE (Efficient Language Understanding Evaluation)
Website: http://eluebenchmark.fastnlp.top/
Paper: https://txsun1997.github.io/papers/elue_paper.pdf

(Benchmarks effectiveness, **number of model parameters**, and **inference-time FLOPs**)



What would a TREC Neural Efficiency Track look like?

Benchmarking by jointly considering effectiveness, efficiency, and robustness

A preliminary framework:

- Identify key measures of effectiveness, efficiency, and robustness
- Measures can either be traded-off against each other, or act as guardrails
- Apply value-laden and business-informed trade-offs between cost and robustness measures to define aggregate measures, e.g.,

The aggregation of cost factors – such as latency l, indexing time i, and storage s – is highly dependent on the scenario: whether we have a high query workload, a high document update frequency, or other requirements. Here, we instantiate our cost side of `C-Tradeoff` with an exemplary balanced aggregated cost (AC) anchored to BM25 per method m:

$$AC_m = \frac{l_m}{l_{BM25}} * \alpha + \frac{i_m}{i_{BM25}} * \beta + \frac{s_m}{s_{BM25}} * \gamma \quad (11)$$

where α, β, and γ control the importance of each cost component. In Figure 1, we set α to 10, and β & γ to 1.

- Identify the set of acceptable solutions (again) based on value-laden and business-informed trade-offs decision boundaries

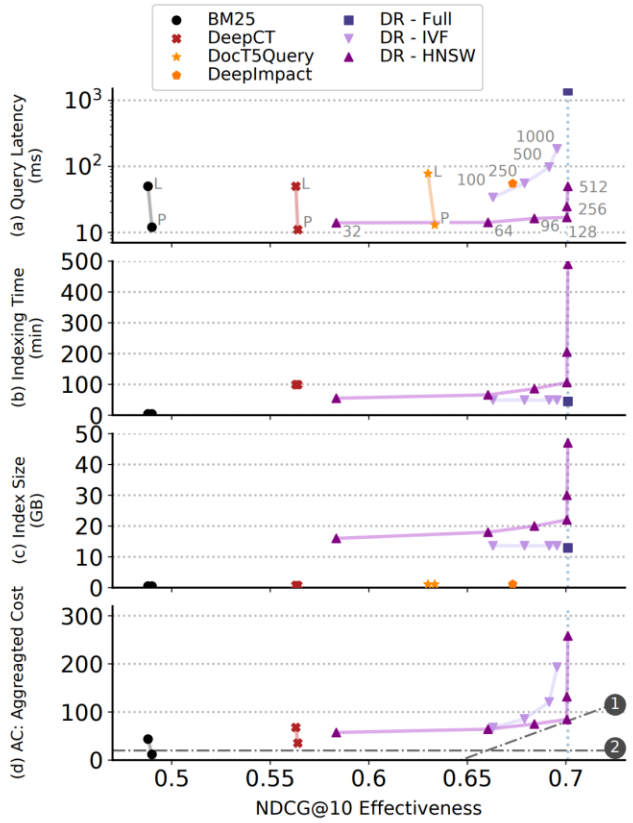


Fig. 1. TREC-DL passage retrieval comparison of the cost-effectiveness tradeoff for term-based and dense retrieval.

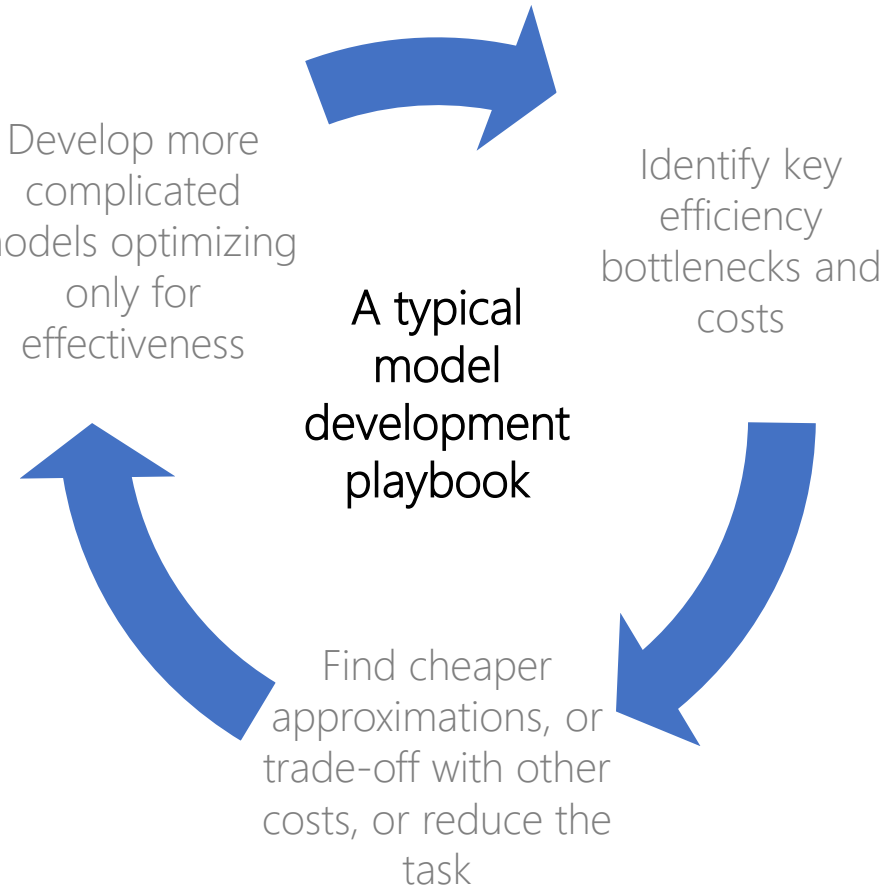
Hofstätter, Craswell, Mitra, Zamani, and Hanbury. Are We There Yet? A Decision Framework for Replacing Term-Based Retrieval with Dense Retrieval Systems. ArXiv preprint. (2022)

Developing more efficient neural IR models

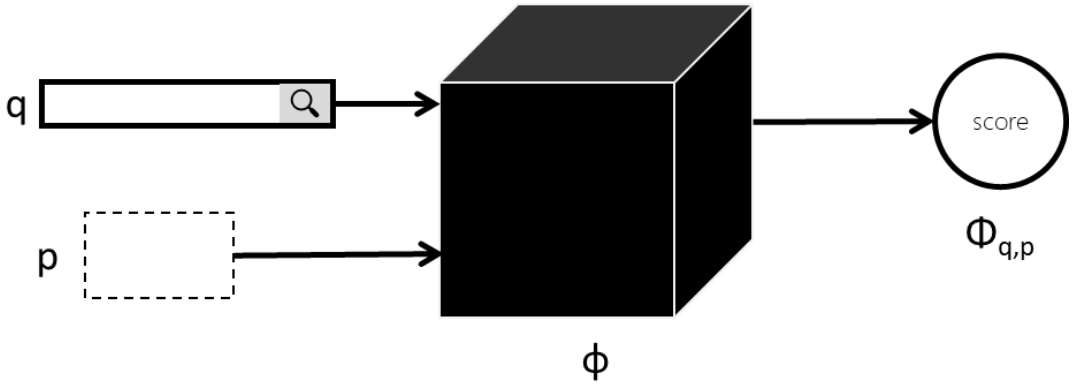
This playbook has been useful in allowing explorations of more ambitious architectures and then reducing their costs / footprints to make them practically deployable in large commercial settings

However, we should ask:

- Would we build different models if we are informed by broader system design (data structures and algorithms) and cost considerations from the very start?
- Are groups with access to large compute resources adequately incentivized to consider efficiency as early as possible in the development process, especially given that large compute resources can be a competitive advantage?
- Are we missing out on use-cases where machine learning is employed specifically to improve efficiency, instead of effectiveness?



Case studies: Scaling BERT-based relevance models to long documents and full retrieval settings



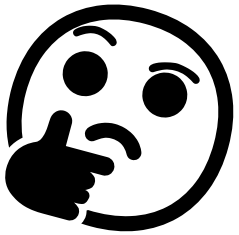
Nogueira and Cho (2019)

How do we scale to longer documents?

Bottleneck: Peak GPU memory during training

How do we scale to full retrieval from large collections?

Challenge: Collection size vs. expected online query response time



PASSAGE RE-RANKING WITH BERT

Rodrigo Nogueira
New York University
rodrigonogueira@nyu.edu

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

ABSTRACT

Recently, neural models pretrained on a language modeling task, such as ELMo (Peters et al., 2017), OpenAI GPT (Radford et al., 2018), and BERT (De-

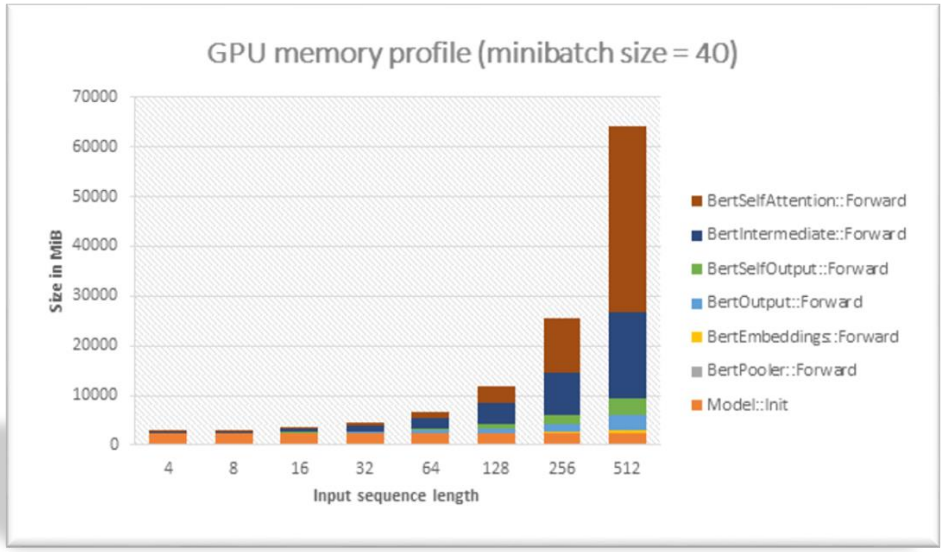
Challenges in scaling BERT to longer inputs

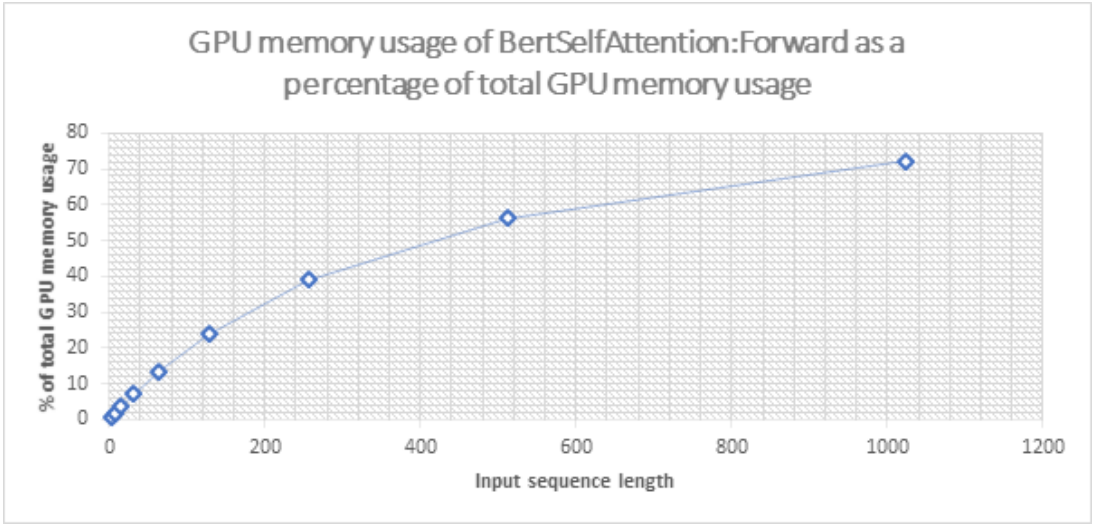
At training time, the GPU memory requirement for BERT's self-attention layers grows quadratically *w.r.t.* to input length

The quadratic complexity is a direct result of storing all the n^2-dimensional attention matrices in GPU memory during training for easier backpropagation

Potential workarounds:

- Trade-off GPU memory and training time by proactively releasing GPU memory during forward pass at the cost of redundant re-computations during backward pass
- Find cheaper approximation to self-attention layers
- Reduce the input space by running BERT on select passages in the document



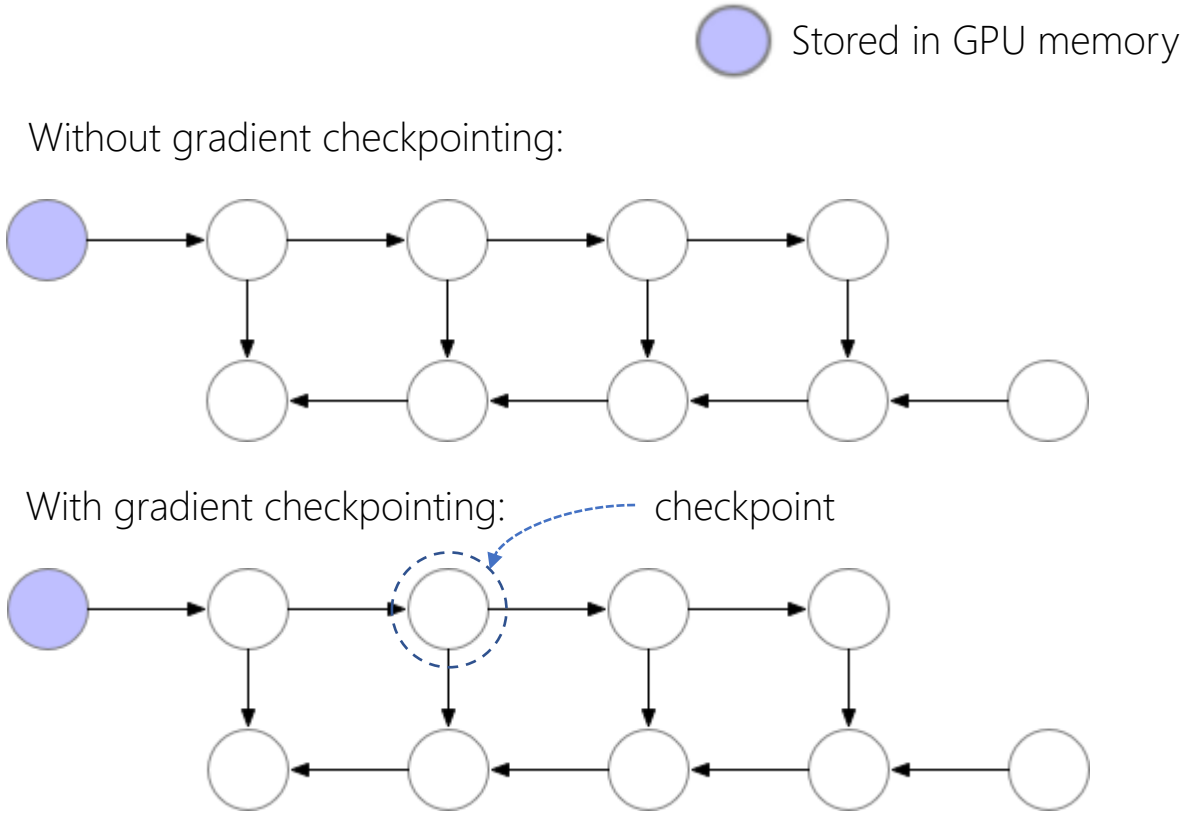


Trade-off GPU memory and training time using gradient checkpointing

At training time, during the forward-pass the model caches all intermediate outputs in GPU memory so that during backward-pass we can easily compute the gradients of a layer's outputs *w.r.t.* its inputs

Under gradient checkpointing (Chen et al., 2016), in contrast, the model only saves intermediate outputs at specific checkpoints; during the backward-pass, missing intermediate outputs are recomputed based on the closed preceding checkpoint(s)

For Transformers, this allows us to store only one n^2-dimensional attention matrix in GPU memory at any given time!



https://medium.com/tensorflow/fitting-larger-networks-into-memory-583e3c758ff9

Cheaper approximation: Transformer → Conformer

Conformer is an alternative to Transformer that employs a separable self-attention layer with linear GPU memory complexity (as opposed to Transformer's quadratic complexity) and is augmented with additional convolutional layers to model short-distance attention

$$\text{Separable-Self-Attention}(Q, K, V) = \Phi(Q) \cdot A$$
$$\text{where, } A = \Phi(K^\intercal) \cdot V$$

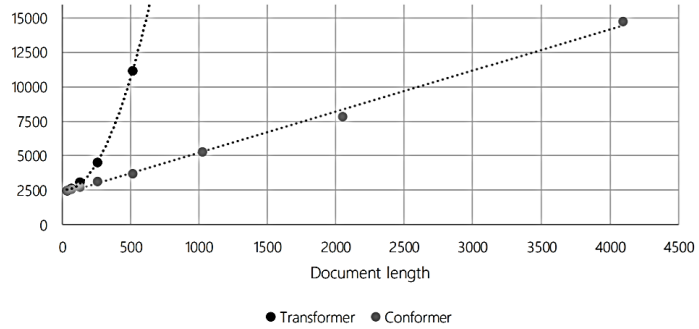
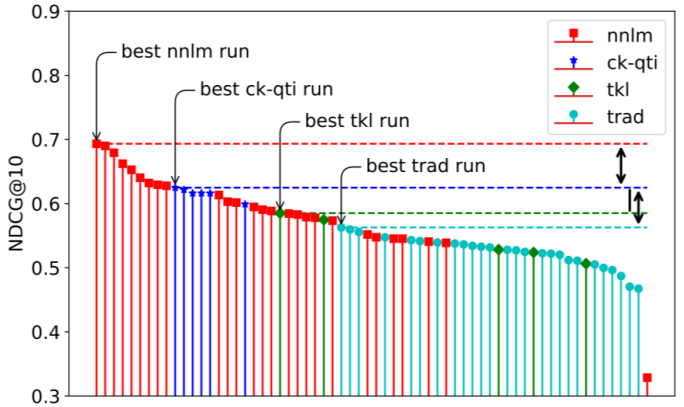


Figure 2: Comparison of peak GPU Memory Usage in MB, across all four GPUs, when employing Transformers *vs.* Conformers.

TREC 2020 Deep Learning Track
(Document Ranking Task)



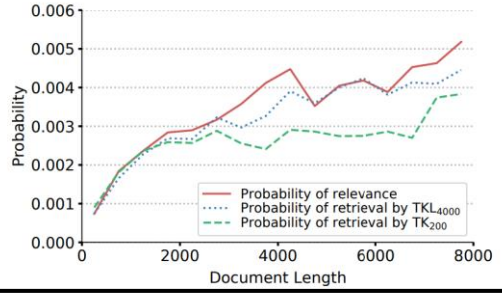
(a) NDCG@10

Mitra, Hofstätter, Zamani, and Craswell. Conformer-Kernel with Query Term Independence for Document Retrieval. ArXiv preprint. (2020)
Mitra, Hofstätter, Zamani, and Craswell. Conformer-Kernel with Query Term Independence at TREC 2020 Deep Learning Track. In Proc. TREC. (2020)
Mitra, Hofstätter, Zamani, and Craswell. Improving Transformer-Kernel Ranking Model Using Conformer and Query Term Independence. In Proc. SIGIR. (2021)

Reduce the task: Passage-based document ranking

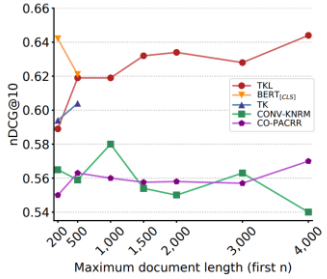
Strategy 1: Run BERT on first-k tokens from the document

Considering only the first-k tokens leads to underestimation of relevance and consequently under-retrieval of longer documents (Hofstätter et al., 2020). Recent studies (Kazai et al., 2022) have also analyzed when single snippets are insufficient for both human and machine learning based relevance estimation.



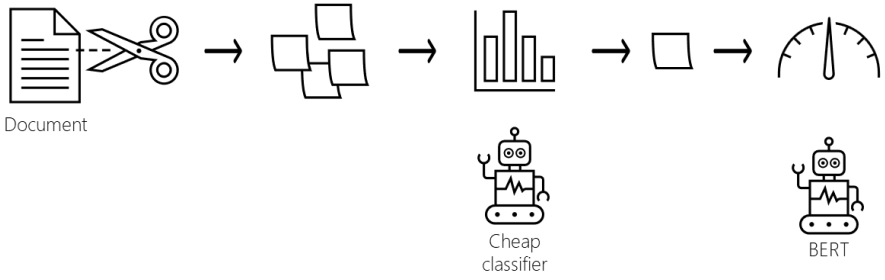
Strategy 2: Run BERT on multiple windows of k-tokens each from the document

This is the approach proposed by Hofstätter et al. (2020). However, the number of windows can be large corresponding to longer documents and running BERT too many times per query-document pair can also be prohibitively costly.



Strategy 3: Run BERT on windows of text pre-selected using cheaper models

This is the approach proposed by Hofstätter et al. (2021). The approach (IDCM) was motivated by cascaded ranking pipelines, but in this case the cascades are employed within-document for passage selection.

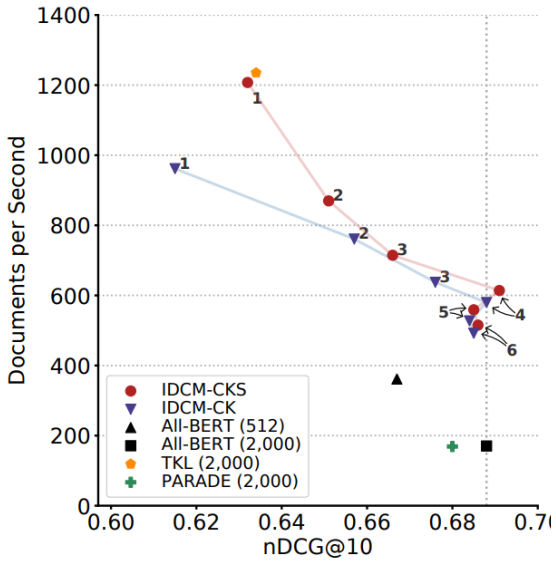


Hofstätter, Zamani, Mitra, Craswell, and Hanbury. Local Self-Attention over Long Text for Efficient Document Retrieval. In Proc. SIGIR. (2020)
Hofstätter, Mitra, Zamani, Craswell, and Hanbury. Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking. In Proc. SIGIR. (2021)
Kazai, Mitra, Dong, Zamani, Craswell, and Yang. Less is Less: When Are Snippets Insufficient for Human vs Machine Relevance Estimation? In Proc. ECIR. (2022)

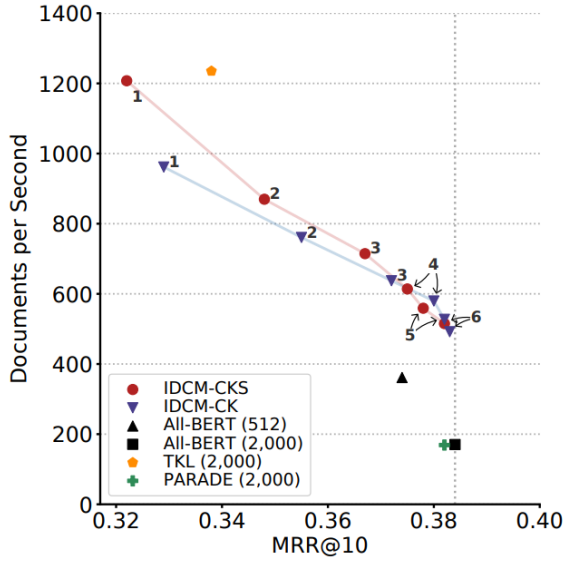
Intra-Document Cascaded Model (IDCM)

We employ a cascaded architecture: a cheaper model ranks-and-prunes candidate passages and costlier BERT model inspects only selected passages from the document

The cheaper model is trained via knowledge distillation from the BERT model



(a) Throughput and nDCG@10 results on TREC DL 2019



(b) Throughput and MRR@10 results on MS MARCO Dev.

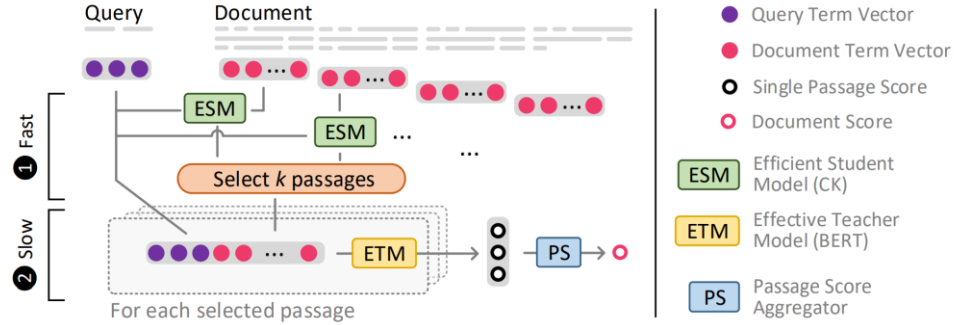


Figure 1: The IDCM architecture that consists of two cascading stages: ❶ To allow for long-document input, the first stage is a lightweight and fast selection model. ❷ Only the top k passages from the selection model are scored with a costly BERT-based scoring module to form the final document score.

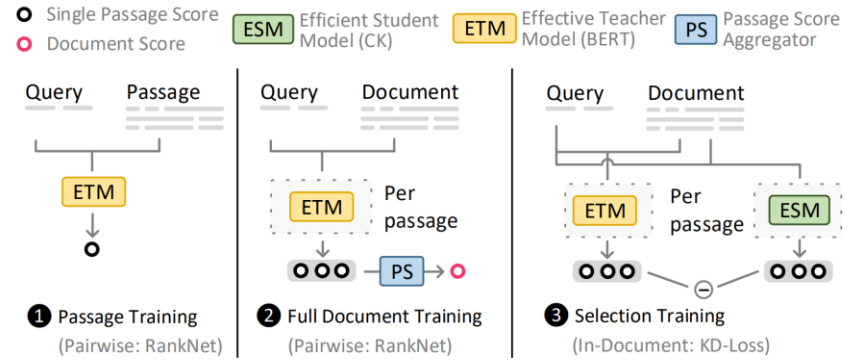


Figure 2: The staged training workflow of IDCM: ❶ Training the ETM (BERT) passage module ❷ Training the full model on a document collection without selection (all available passages of a document are scored with ETM). ❸ The ESM (CK) selection module is now trained via knowledge distillation using the ETM (BERT) scores as labels.

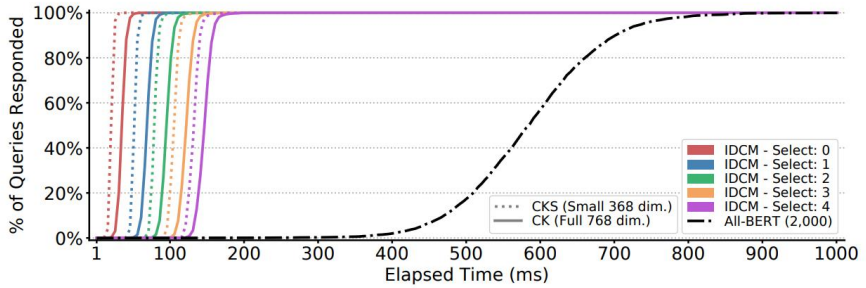
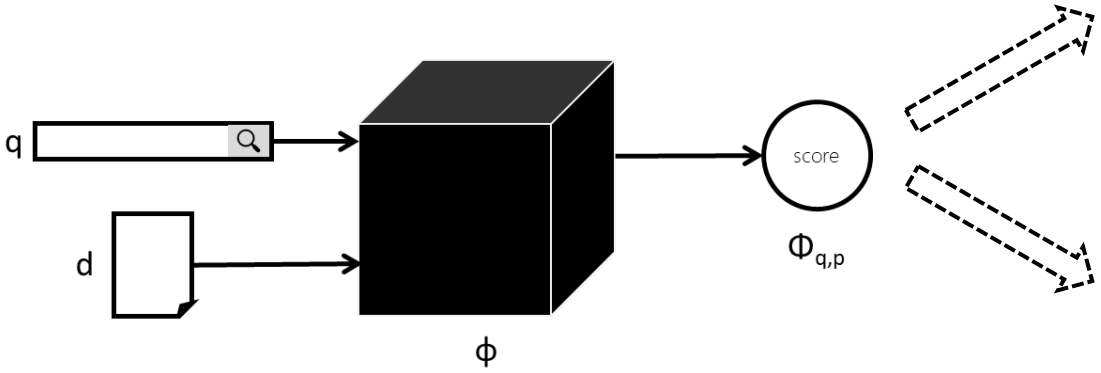


Figure 4: Fraction of queries that can be answered in the given time-frame for re-ranking 100 documents with up to 2,000 tokens on MSMARCO. Select 0 means only CK timing without BERT cascading.

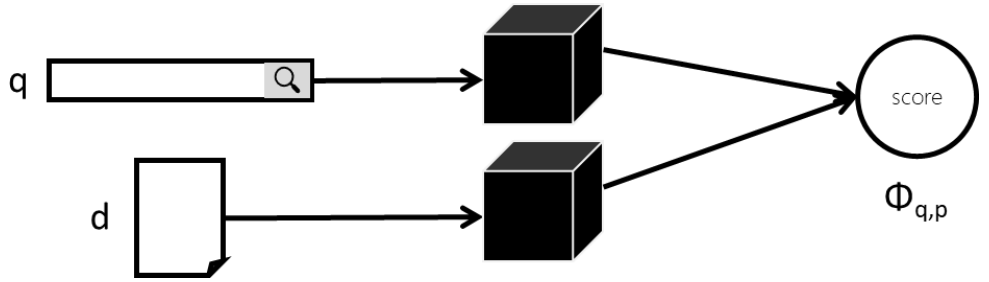
Hofstätter, Mitra, Zamani, Craswell, and Hanbury. Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking. In Proc. SIGIR. (2021)

Challenges in scaling BERT to full retrieval

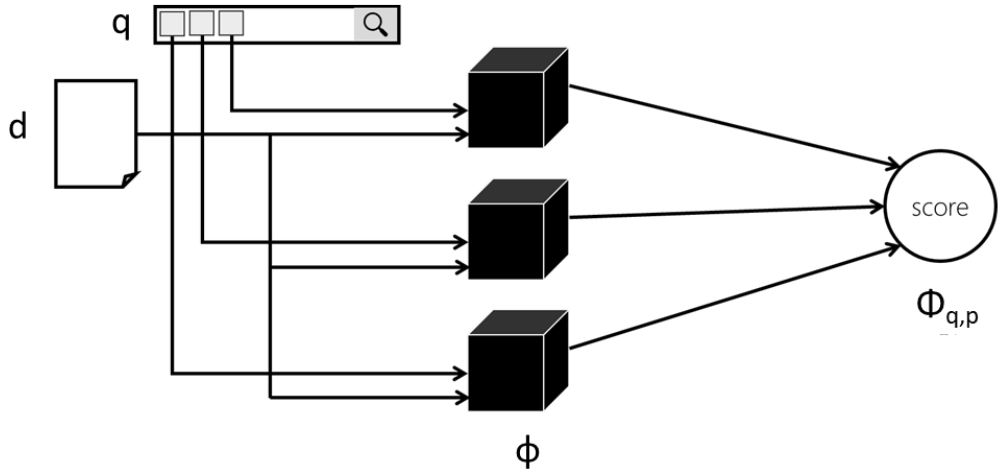


Nogueira and Cho (2019)

Broadly two sets of approaches have emerged: Dense retrieval and Query Term Independent (QTI) models; both precompute document representations at indexing time and require very little computations at query response time



Dense retrieval: Xiong et al. (2021), Qu et al. (2021), Hofstätter et al. (2021), and others

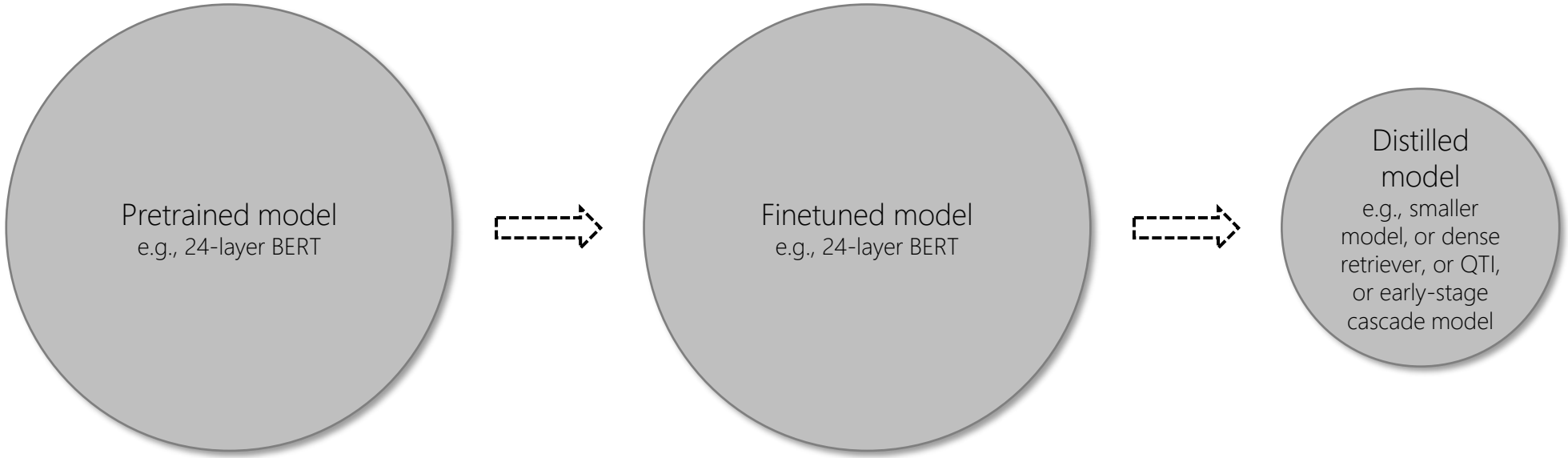


QTI: Mitra et al. (2019), Nogueira et al. (2019), Dai and Callan (2020), and others

Mitra, Rosset, Hawking, Craswell, Diaz, and Yilmaz. Incorporating Query Term Independence Assumption for Efficient Retrieval and Ranking Using Deep Neural Networks. ArXiv preprint. (2019)
Mitra, Hofstätter, Zamani, and Craswell. Conformer-Kernel with Query Term Independence for Document Retrieval. ArXiv preprint. (2020)
Mitra, Hofstätter, Zamani, and Craswell. Conformer-Kernel with Query Term Independence at TREC 2020 Deep Learning Track. In Proc. TREC. (2020)
Mitra, Hofstätter, Zamani, and Craswell. Improving Transformer-Kernel Ranking Model Using Conformer and Query Term Independence. In Proc. SIGIR. (2021)

A note about distillation

A popular recipe involves pretraining/finetuning large models and then knowledge distillation to smaller models that can be deployed in real-world retrieval systems



Machine Learning for Retrieval Efficiency

In IR, predictive machine learning has largely been employed for relevance estimation

Kraska et al. (2018) were one of the earliest to propose learned index structures where predictive machine learning is employed to speed up search over classical data structures

Opinion: I believe there's a significant opportunity to employ deep learning and other machine learning approaches to directly optimize for efficiency in our search and recommendation stacks

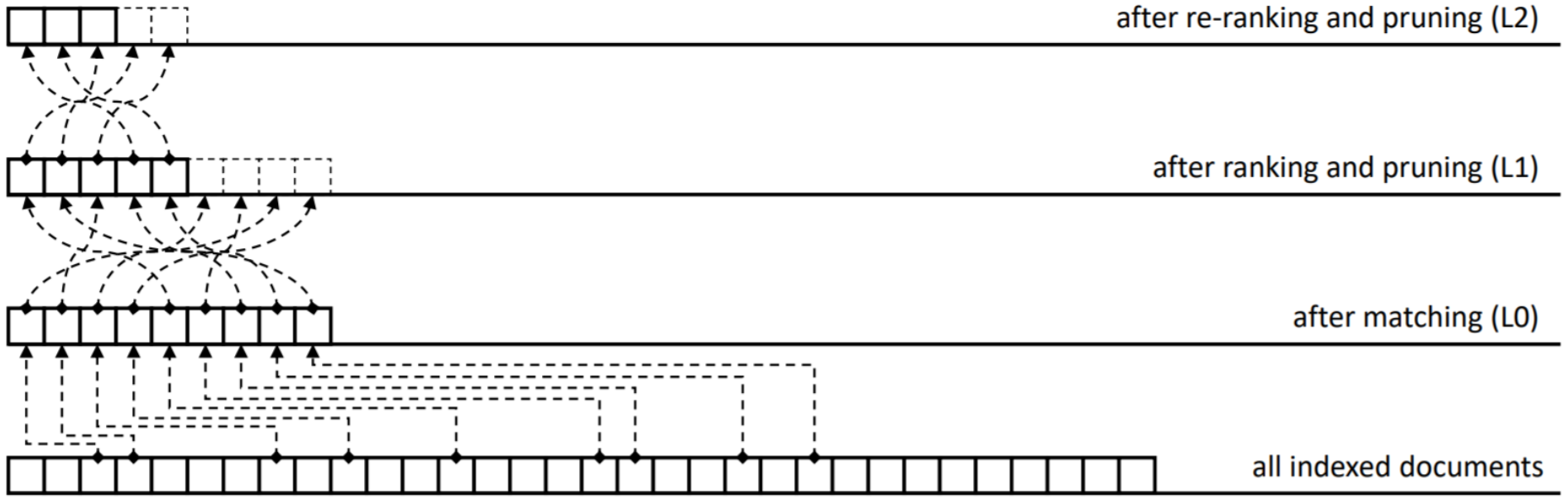
Let's look at an example…

Large scale IR systems trade-off search result quality and query response time

In Bing, we have a candidate generation stage followed by multiple rank and prune stages

Typically, we apply machine learning in the re-ranking stages

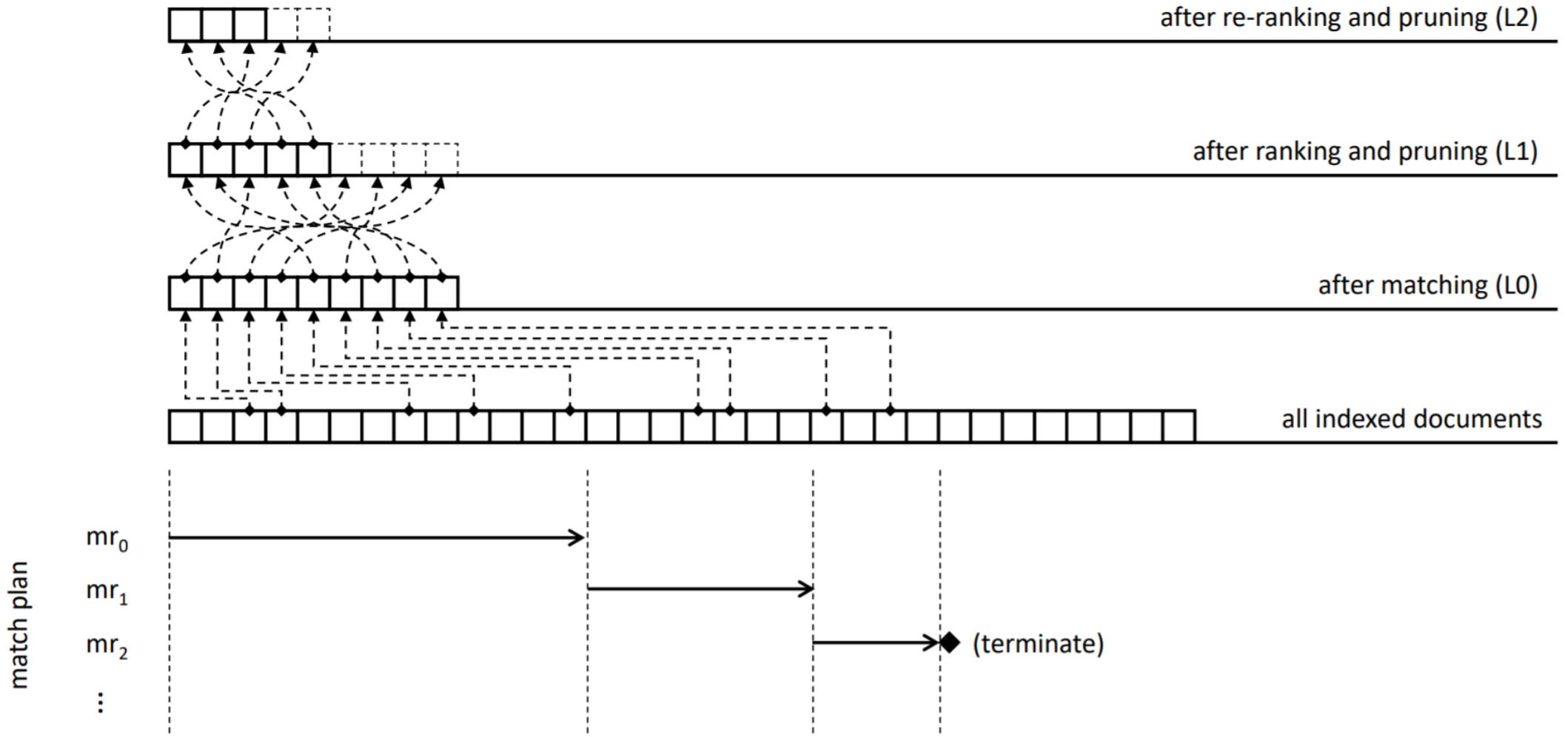
In this work, we explore reinforcement learning for effective and efficient candidate generation



Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)

In Bing, the index is distributed over multiple machines

For candidate generation, on each machine the documents are linearly scanned using a match plan



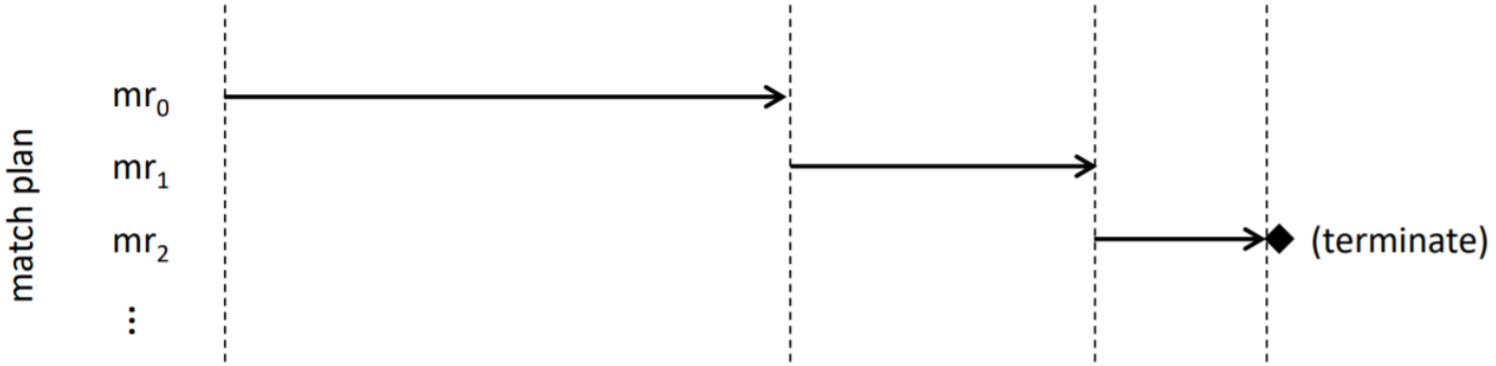
Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)

When a query comes in, it is automatically categorized, and a pre-defined match plan is selected

A **match rule** defines the condition that a document should satisfy to be selected as a candidate

A **match plan** consists of a sequence of match rules, and corresponding stopping criteria

The **stopping criteria** decides when the index scan using a particular match rule should terminate—and if the matching process should continue with the next match rule, or conclude, or reset to the beginning of the index

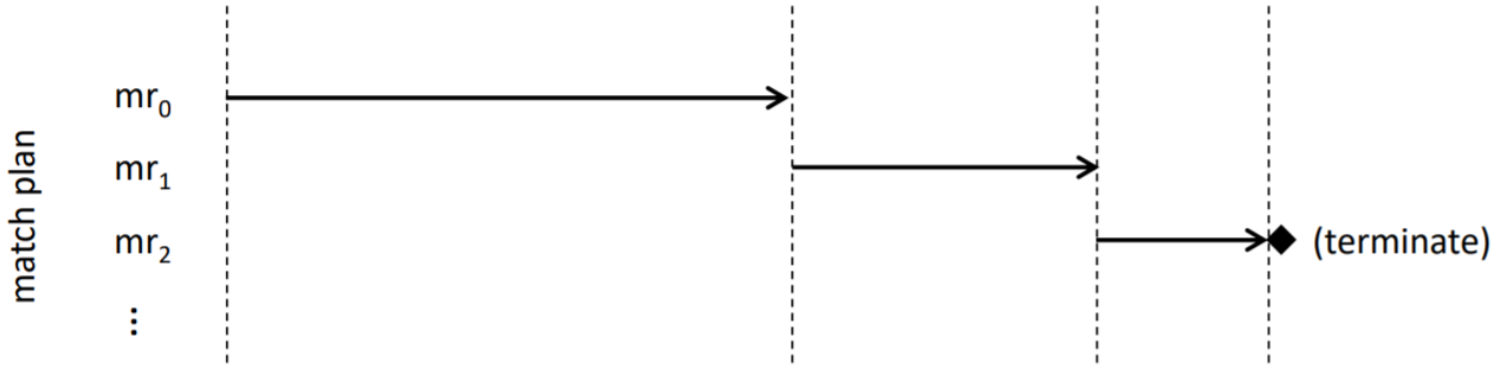


Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)

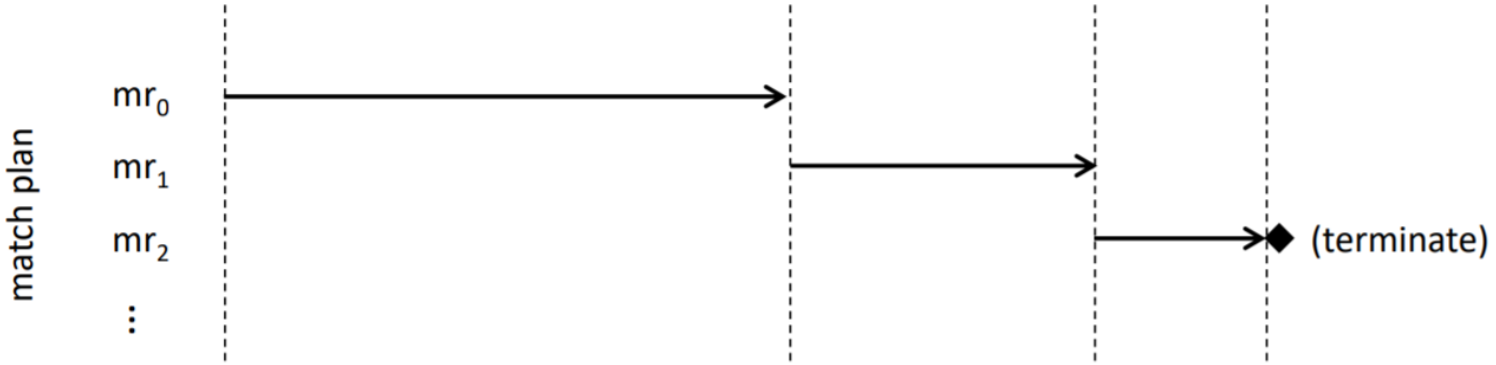
Match plans influence the trade-off between effectiveness and efficiency

E.g., long queries with rare intents may require expensive match plans that consider body text and search deeper into the index

In contrast, for popular navigational queries a shallow scan against URL and title metastreams may be sufficient



Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)



E.g.,

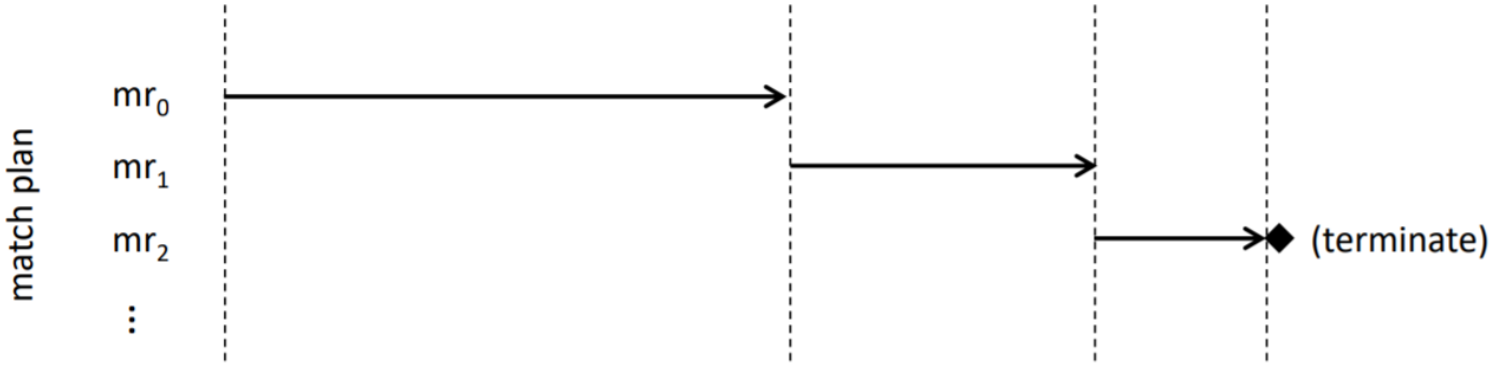
Query: halloween costumes

Match rule: $mr_A \rightarrow (\text{halloween} \in A|U|B|T) \wedge (\text{costumes} \in A|U|B|T)$

Query: facebook login

Match rule: $mr_B \rightarrow (\text{facebook} \in U|T)$

Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)



During execution, two accumulators are tracked

u: the number of blocks accessed from disk

v: the cum. number of term matches in all inspected documents

A stopping criteria sets thresholds for each – when either thresholds are met, the scan using that particular match rule terminates

Matching may then continue with a new match rule, or terminate, or re-start from beginning

Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)

Optimizing query evaluations using reinforcement learning

Learn a policy $\pi_\theta : S \rightarrow A$ which maximizes the cumulative discounted reward R, where γ is the discount rate

$$R = \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \quad , \quad 0 < \gamma \leq 1$$

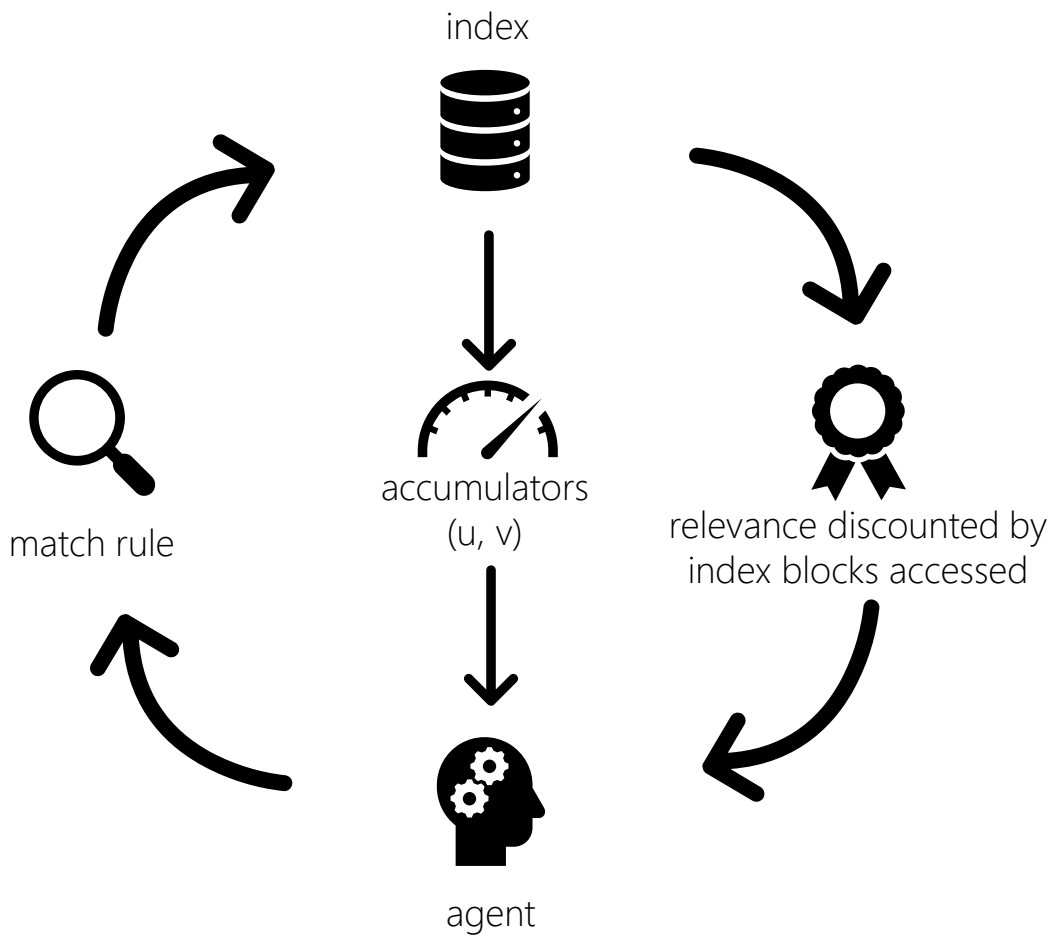
We employ table-based Q learning

State space: index blocks accessed (u_t) and term matches (v_t)

Action space: $\mathcal{A} = \{\mathrm{mr}_1, \ldots, \mathrm{mr}_k\} \cup \{a_{\mathrm{reset}}, a_{\mathrm{stop}}\}$

Reward function: $$r_{\mathrm{agent}}(s_t, a_t) = \frac{\sum_i^{m_{t+1}} g(d_i)}{n \cdot u_{t+1}}$$

$g(d_i)$ is the relevance of the i^{th} document estimated based on the subsequent L1 ranker score—considering only top n documents



Rosset, Jose, Ghosh, Mitra, and Tiwary. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In Proc. SIGIR. (2018)

In summary…

- Complex trade-offs between different costs makes benchmarking challenging
- Typical playbook for efficiency: find cheaper approximation, trade-off other costs, reduce task
- Machine learning can be employed to directly improve efficiency of large-scale IR systems

Questions to the audience…

- How can we create a shared task to encourage more efficient deep learning approaches for IR?
- What would deep learning models for IR look like if designed with specific retrieval data-structures in mind from the start?
- What are the key opportunities to employ predictive machine learning to speed up large-scale retrieval systems?

Thank you!

@UnderdogGeek bmitra@microsoft.com