Contextualized Offline Relevance Weighting for Efficient and Effective Neural Retrieval

Xuanang Chen[1,2], Ben He[1,2], Kai Hui[3], Yiran Wang[1,2], Le Sun[2], Yingfei Sun[1]

[1] University of Chinese Academy of Sciences

[2] Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences

[3] Amazon Alexa

July, 2021

Outlines

- Background
- Method: Offline Relevance Weighting for Online Efficiency
- Experiments: Retrieval and Re-ranking
- Analysis: Efficiency, Offline Computation and Storage Cost
- Summary

Background

- Online search latency:
  - A major bottleneck in deploying large-scale pre-trained language models, like BERT and T5, in retrieval applications.
  - These large-scale models are mainly used to re-rank a shallow-pool.

- Document expansion:
  - doc2query utilizes transformer model to generate queries from documents, and expands the original documents using these queries;
  - BM25 retrieval is done on the expanded documents, with remarkable improvements;
  - It is later upgraded to docTTTTTquery by using more powerful T5 for the query generation.

[doc2query] Rodrigo Nogueira, Wei Yang, Jimmy Lin, Kyunghyun Cho. Document Expansion by Query Prediction. CoRR abs/1904.08375 (2019).
[docTTTTTquery] Rodrigo Nogueira, Jimmy Lin, AI Epistemic. From doc2query to docTTTTTquery. Online preprint (2019).

Background

- ColBERT:
  - Document representations are pre-computed offline and stored on disk;
  - Late interaction with MaxSim scoring for fast online ranking;
  - Can do full retrieval, but latency is linear to the collection size;
  - Huge disk storage cost.

- Can we have an approach that
  - Moves (most) BERT matching offline;
  - Fast online retrieval;
  - Less disk space cost.

[ColBERT] Omar Khattab, Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. SIGIR 2020: 39–48.

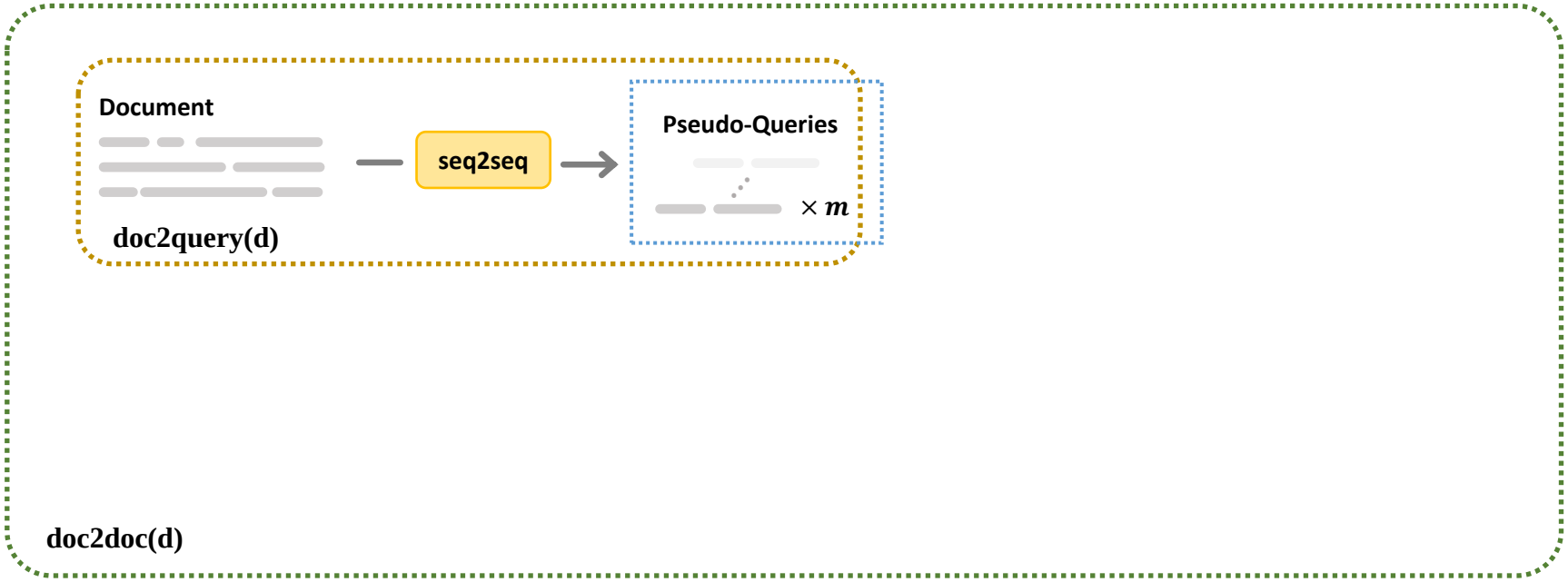
Method: Overview

- Inspired by document expansion method, we propose to **trade offline relevance weighting for online efficiency**.

- The method includes two stages:
  - Offline matching:
    a) pseudo-query generation
    b) seek the nearest neighbor documents for each document based on its pseudo-queries
  - Online retrieval:
    a) the traditional query-document matching is reduced to the **less expensive query to pseudo-query matching**

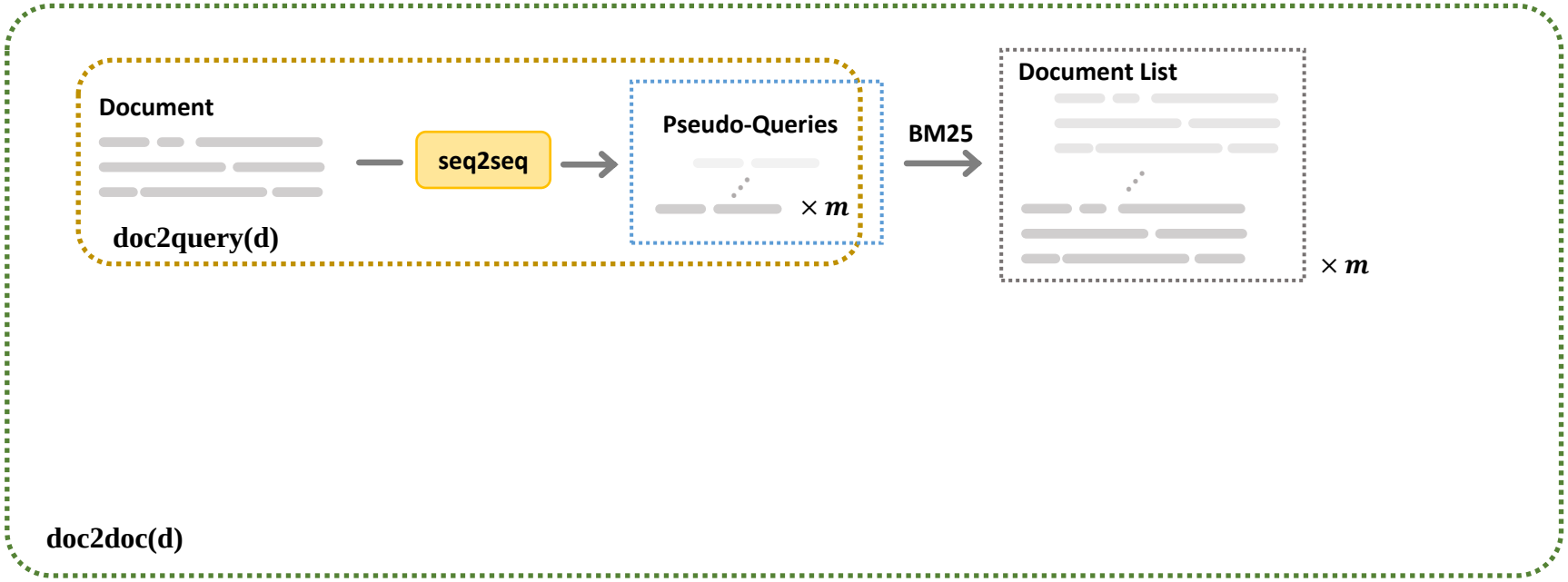
Method: Offline System

- **doc2query(d):** producing ***m*** short text pieces as pseudo-queries for each document in the corpus.
- **doc2doc(d):** seeking the ***n*** nearest neighbours for each document using its pseudo-queries.
- **Offline relevance weighting:** pre-computing relevance scores for pairs between pseudo-queries and neighbour documents.



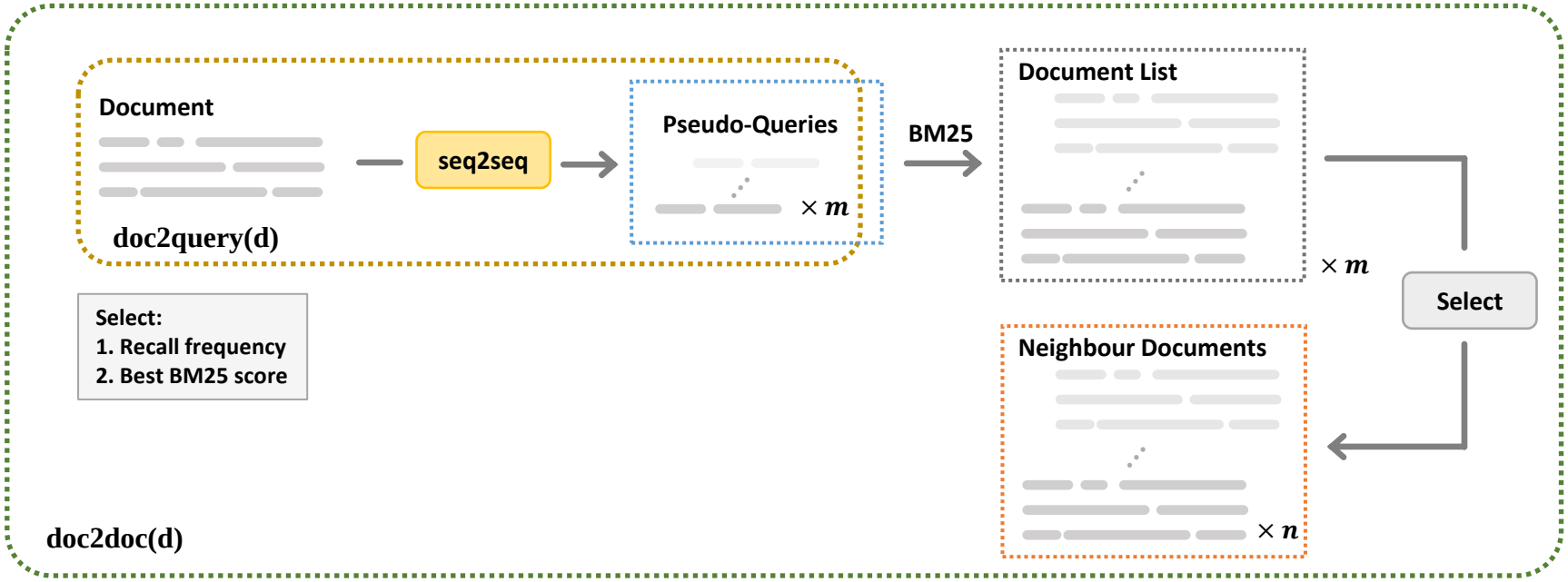
Method: Offline System

- **doc2query(d):** producing ***m*** short text pieces as pseudo-queries for each document in the corpus.
- **doc2doc(d):** seeking the ***n*** nearest neighbours from the corpus using its pseudo-queries.
- **Offline relevance weighting:** pre-computing relevance scores for pairs between pseudo-queries and neighbour documents.



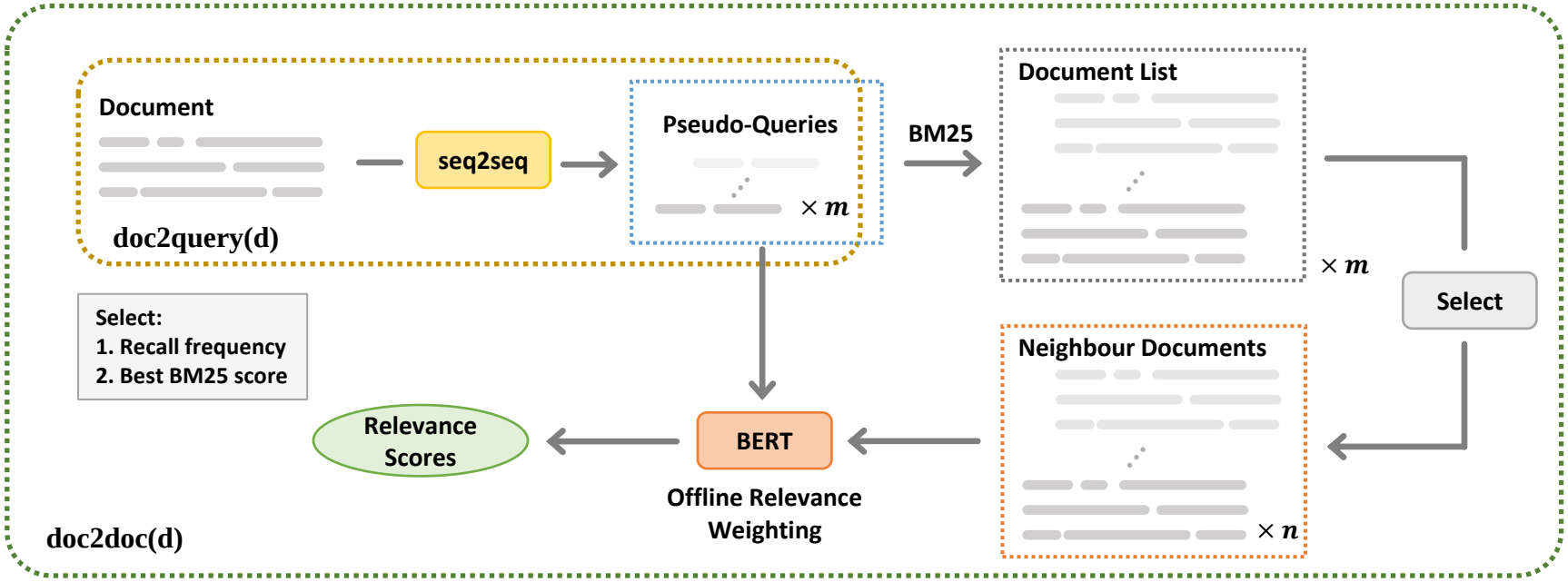
Method: Offline System

- **doc2query(d):** producing m short text pieces as pseudo-queries for each document in the corpus.
- **doc2doc(d):** seeking the n nearest neighbours from the corpus using its pseudo-queries.
- **Offline relevance weighting:** pre-computing relevance scores for pairs between pseudo-queries and neighbour documents.



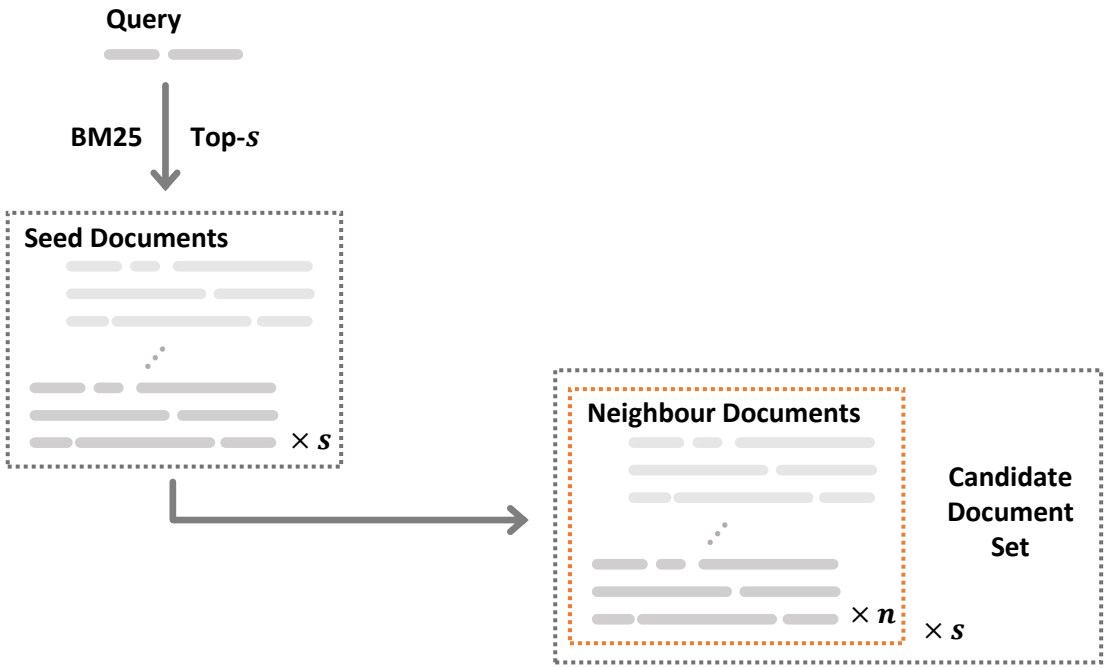
Method: Offline System

- **doc2query(d):** producing m short text pieces as pseudo-queries for each document in the corpus.
- **doc2doc(d):** seeking the n nearest neighbours from the corpus using its pseudo-queries.
- **Offline relevance weighting:** pre-computing relevance scores for pairs between pseudo-queries and neighbour documents.



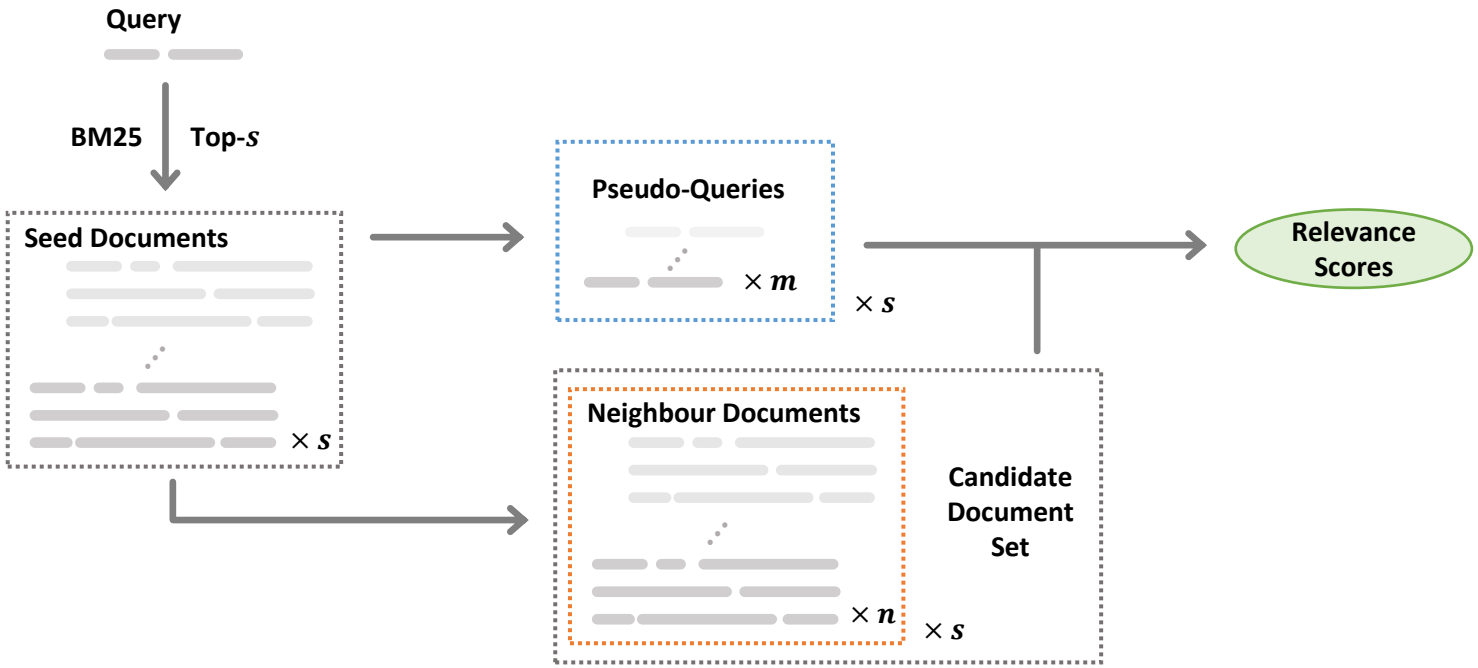
Method: Online System

- Creation of the candidate document set
  - **Seed documents:** getting top-s ranked documents using BM25 to form a seed document set.
  - The neighbor documents of seed documents are all collected to form the final candidate document set for re-ranking.



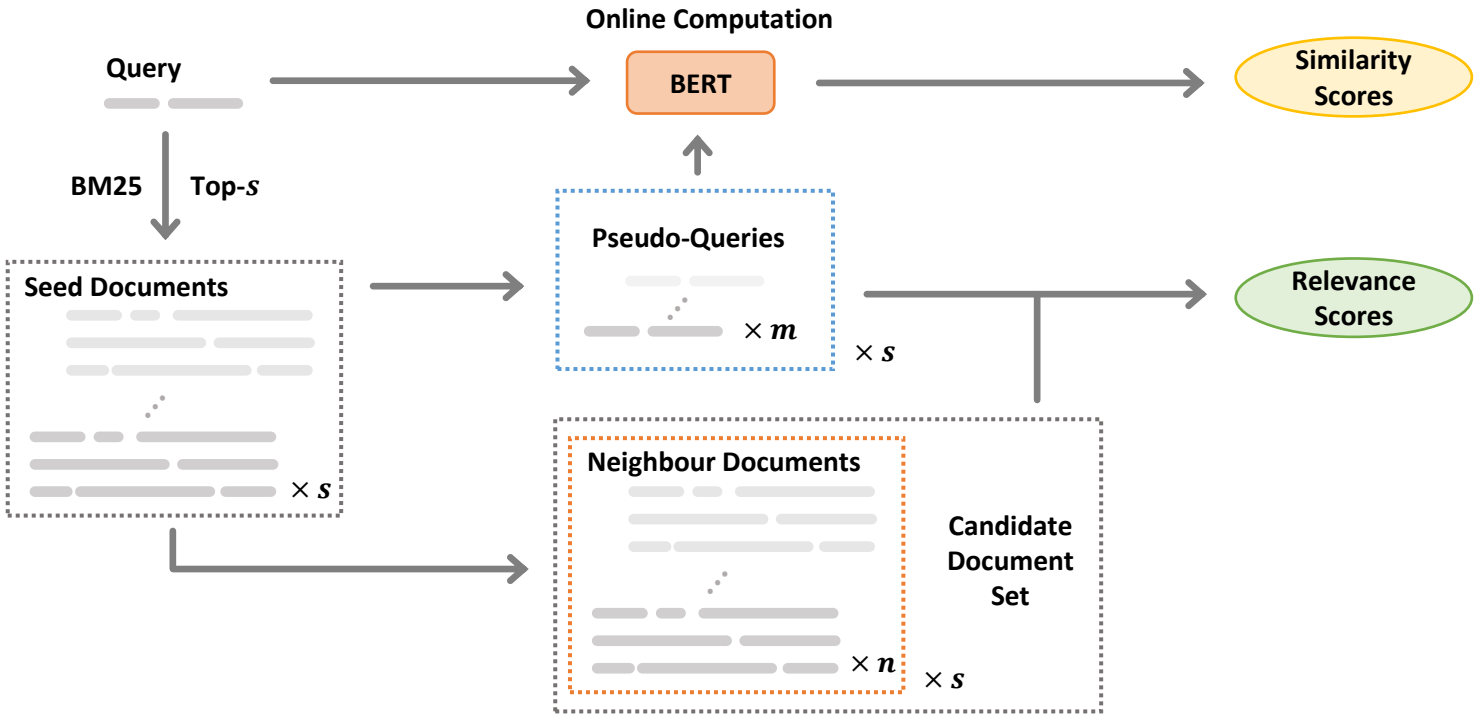
Method: Online System

- Relevance for the candidate documents
  - For each seed document, its pseudo-queries and pre-computed relevance scores are used.



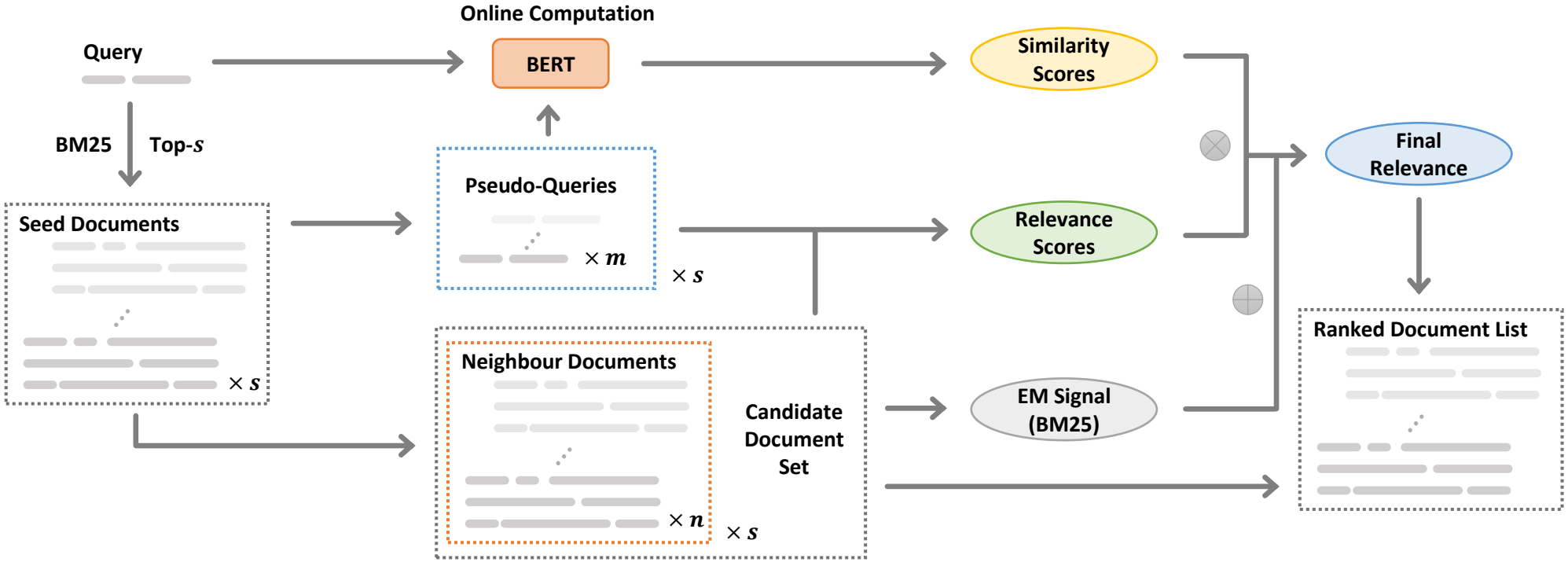
Method: Online System

- Relevance for the candidate documents
  - **Online computation:** the similarity scores between the query and the pseudo-queries (which is much cheaper than the query-document matching).



Method: Online System

- Relevance for the candidate documents
  - Using offline pre-computed relevance scores and online computed similarity scores, we can score all candidate documents;
  - It can also be interpolated with exact matching signals, e.g. BM25.



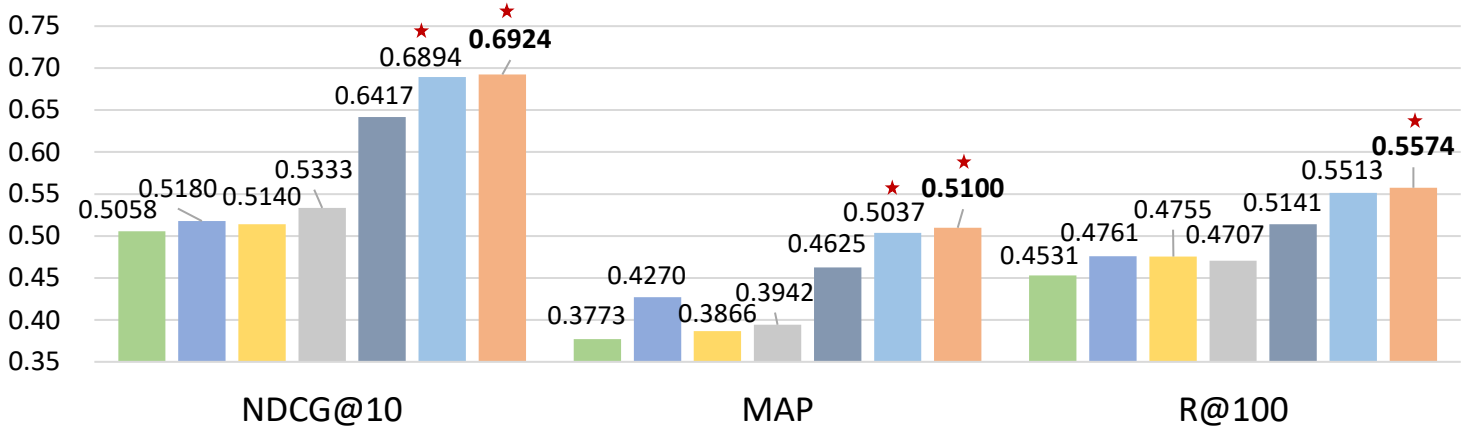
Experiments

- Dataset: MS MARCO Passage and Document collections
- Benchmarks: TREC 2019 and 2020 Deep Learning (DL) Track
  - Metrics: Use the main official evaluation metrics NDCG@10 and MAP, and also report the retrieval Recall metric R@100.

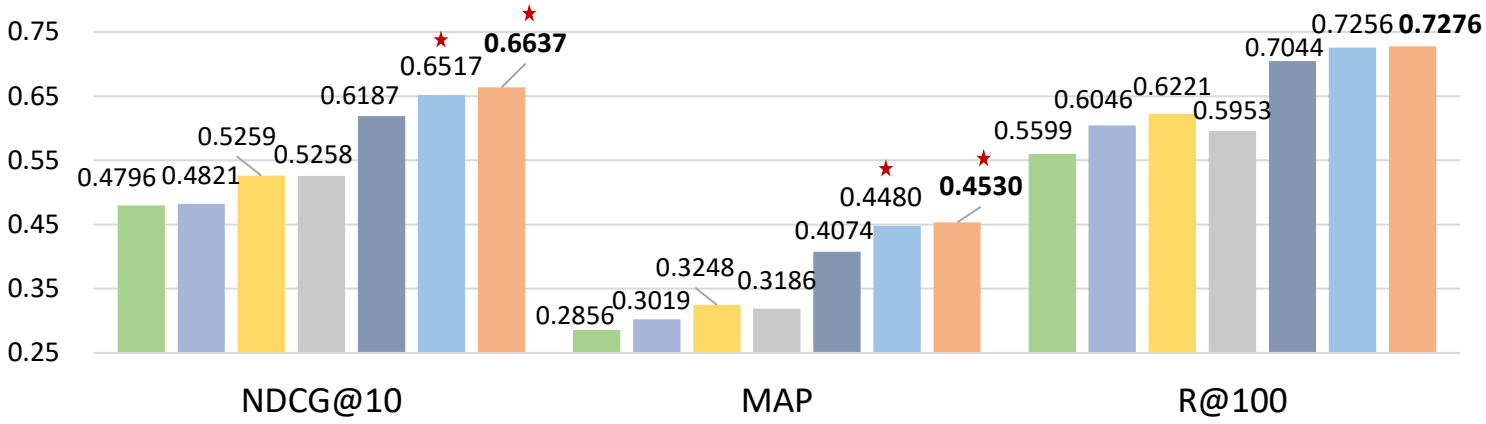
- Baselines:
  - BM25: a classical unsupervised probabilistic ranking model based on exact term matching.
  - BM25+RM3: applies query expansion based on relevance language model.
  - DeepCT: produces the term frequency component of BM25 using BERT-based contextualized word representations during indexing.
  - doc2query and docTTTTTquery: expand each document with queries generated by a seq2seq model. Both rely on the BM25 index for retrieval over the expanded documents.

Passage Retrieval Results

- TREC 2019 DL Passage Ranking



- TREC 2020 DL Passage Ranking

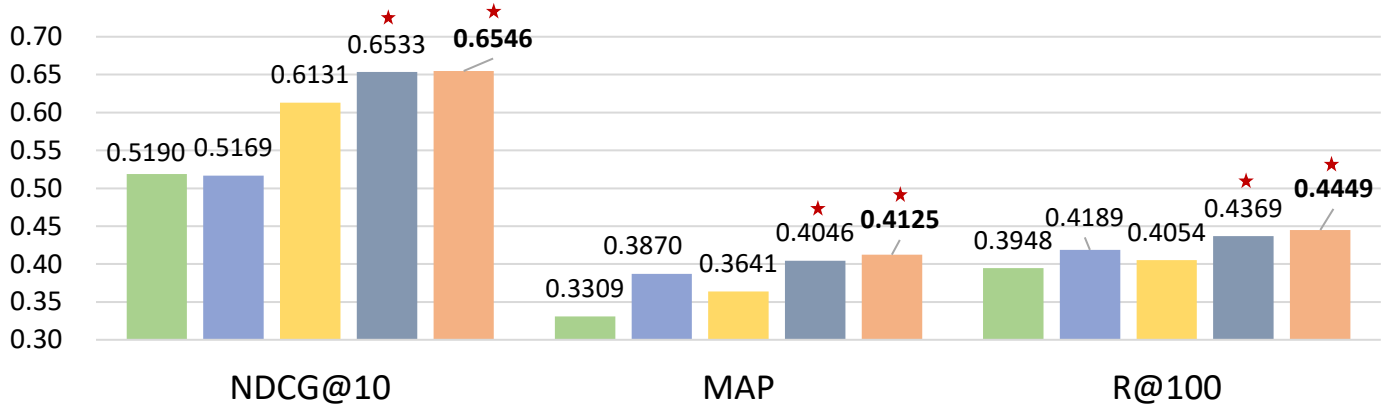


Our method returns significantly better candidate ranking list.

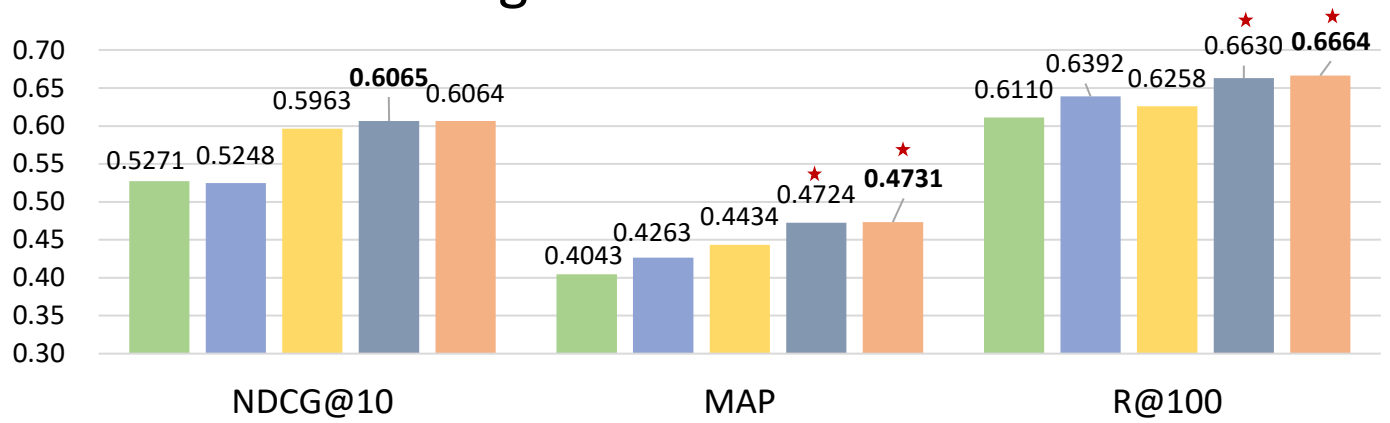
- Statistical significant improvements relative to docTTTTTquery at p-value < 0.05 is denoted by ⋆.

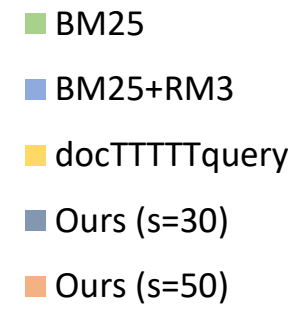
Document Retrieval Results

- TREC 2019 DL Document Ranking



- TREC 2020 DL Document Ranking





Our method returns significantly better candidate ranking list.

- Statistical significant improvements relative to docTTTTTquery at p-value < 0.01 is denoted by ⋆.

Passage Re-ranking Results

- Results of passage ranking task after re-ranked (rr.) by BERT-Base or ColBERT. Top-20 / 1000 passages are re-ranked. NDCG@10 is reported.

Method	TREC 2019 DL Track	TREC 2020 DL Track
BM25 rr. BERT-Base	0.6235 / 0.6945	0.5814 / 0.6677
BM25 rr. ColBERT	0.6276 / 0.7164	0.5943 / 0.6913
BM25+RM3 rr. BERT-Base	0.6014 / 0.6963	0.5664 / 0.6608
BM25+RM3 rr. ColBERT	0.5998 / 0.7281	0.5900 / 0.6814
DeepCT rr. BERT-Base	0.6348 / 0.6929	0.5855 / 0.6883
DeepCT rr. ColBERT	0.6534 / 0.7128	0.5970 / **0.7050**
doc2query rr. BERT-Base	0.6478 / 0.6949	0.6049 / 0.6793
doc2query rr. ColBERT	0.6501 / 0.7233	0.6205 / 0.6920
docTTTTTquery rr. BERT-Base	0.7269 / 0.7100	0.6820 / 0.6784
docTTTTTquery rr. ColBERT	0.7399 / 0.7328	0.6931 / 0.6935
Ours ($s = 30$) rr. BERT-Base	0.7237 / 0.6990	0.6808 / 0.6749
Ours ($s = 30$) rr. ColBERT	0.7454 / 0.7244	0.7086 / 0.6842
Ours ($s = 50$) rr. BERT-Base	0.7271 / 0.7043	0.6860 / 0.6791
Ours ($s = 50$) rr. ColBERT	**0.7556** / **0.7340**	**0.7129** / 0.6916

Only a small number of candidates need to be re-ranked using our method, which means our method can achieve competitive performance with improved online efficiency.

Analysis

- Efficiency of Passage Retrieval

Method	CPU (ms/query)	GPU (ms/query)	Latency (ms/query)
BM25	93	-	93
BM25+RM3	186	-	186
DeepCT	-	-	93 (est.)
doc2query	116	-	116
docTTTTTquery	139	-	139
ColBERT	-	-	687 (est.)
BERT-Base (re-rank)	-	11777	11777
Ours ($s = 30$)	120	363	483
Ours ($s = 50$)	174	605	779

- The 43 queries in TREC 2019 DL Track are used. Each method retrieves top-1000 passages per query. s is the number of seed document.
- In our method, when 30 seed documents are used, only 30×5 = 150 pseudo-queries need to be matched for a query, wherein the latency is lower than ColBERT.

Analysis

- The offline computing cost (MS MARCO - 8.8 M Passages)
  - TPU v3 is 20-30× faster than Titan RTX 24G GPU from our experience.

Device	Sequence Length	Model	Est. Cost
Titan RTX 24G GPU × 1	512	BERT-Base	≈ 6,631 day
TPU v3 × 100	256	BERT-Base	< 1 day

- The storage space cost (one document)

Method	Doc. Embeddings	Pseudo-Query	Rel. Scores	Storage Space
ColBERT	200×128×4	-	-	102,400 bytes (5×)
Ours	-	5×34×1	5×1000×4	20,170 bytes (1×)

  - ColBERT: suppose each document includes on average 200 tokens (128-dimensional vectors, 4 bytes/dimension).
  - Ours: each document has 5 pseudo-queries (34 letters/pseudo-query, 1 byte/letter) and 1,000 neighbour documents (5×1000 relevance scores, 4 bytes/score).

Summary

- Pros
  - Move most expensive BERT matching offline
  - Online computation is mainly query to pseudo-query matching
    - Using BERT, short text matching is much cheaper
  - Our approach has similar latency with ColBERT full ranking on MS MARCO
    - But ColBERT's latency is linear to collection size
  - Storage cost is 20% of ColBERT
    - Mainly scores of neighbour documents and pseudo-queries
- Cons
  - The offline computation can be VERY expensive
  - Takes efforts to train the query generation model

Thanks~