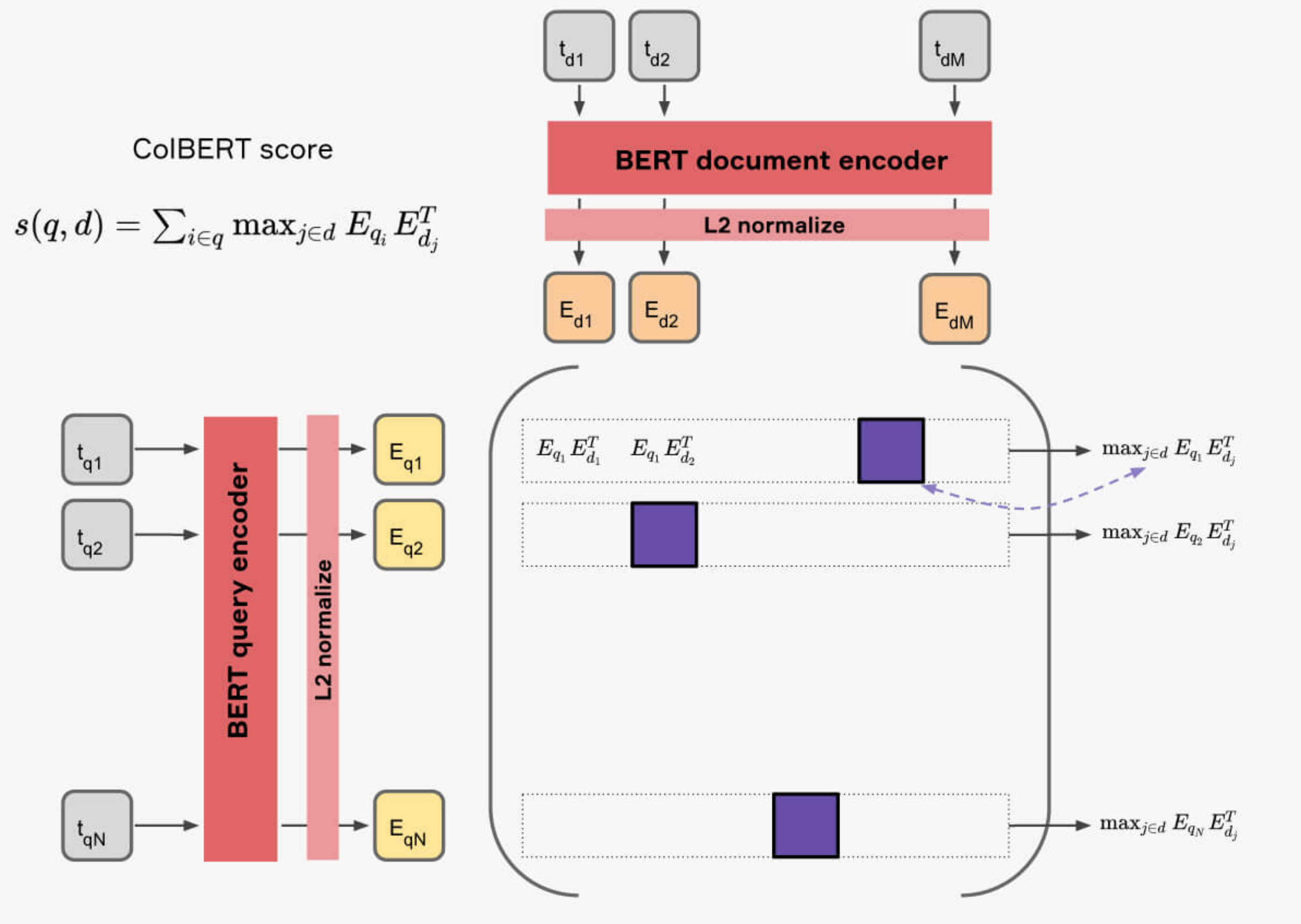
ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction

Thiago Coelho Vieira - thiagotcvieira@gmail.com

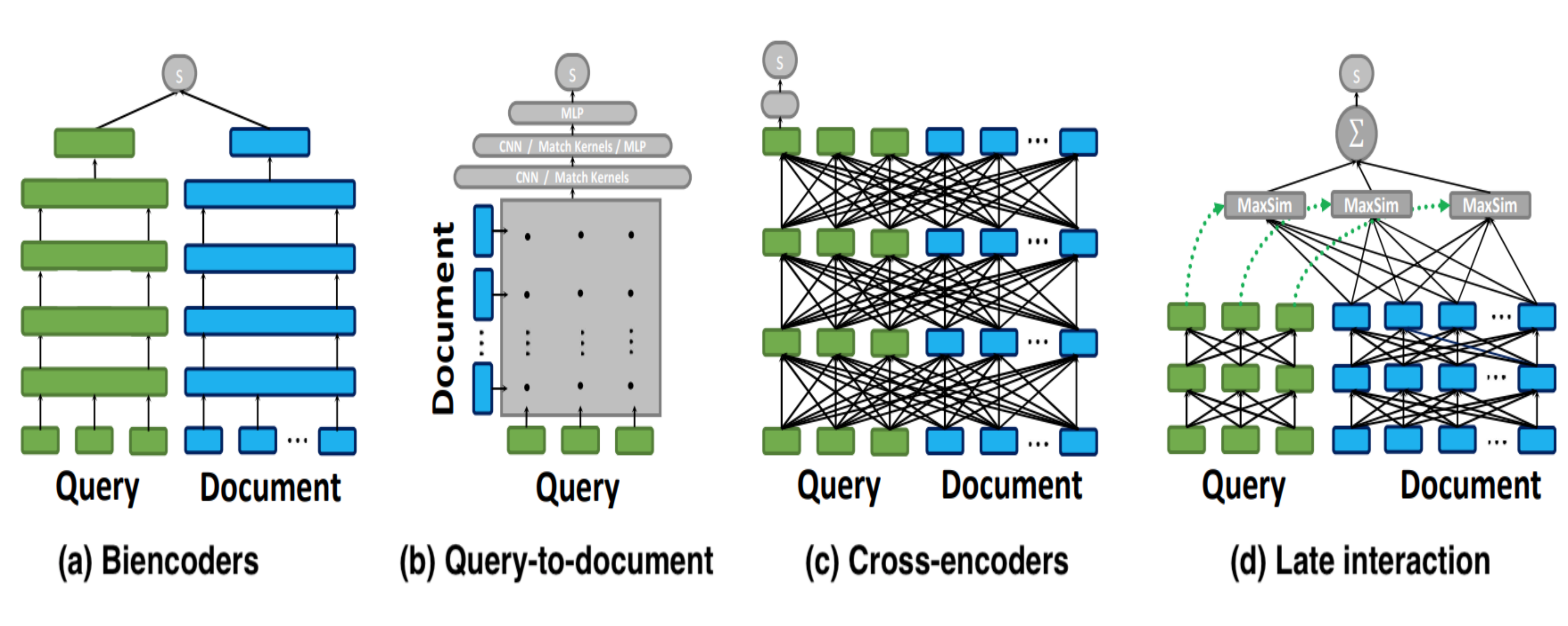
1.1 main concepts

- **ColBERTv1**
  - late interaction as a neural ranking paradigm
  - dense vector representation for each token
  - discussions on exact match vs soft match
  - hypotesis over it promotes exact match where it is more relevant to IR
  - can be use as reranker or end2end top-k retriever
- **single-vector** - a pretrained language model is used to encode each query and each document into a single high-dimensional vector, and relevance is modeled as a simple dot product between both vectors
- **multi-vector**
  - for a query/doc q encoder outputs a matrix nxD, not a vector
- **trade-off** the cost of neural inference for reranking (GPUs) against the cost of large amounts of memory to support efficient nearest neighbor search

1.2 ColBERTv1



1.3 Interactions



1.4 MaxSim - similarity score

$$s_{q,d} = \sum_{i \in \eta(q)} \max_{j \in \eta(d)} \eta(q)_i \cdot \eta(d)_j$$

- largest cosine similarity between each query token matrix and all passages token matrix.

> “ constructs a similarity matrix, performs max pooling along the query dimension, followed by a summation to arrive at the relevance score ”

1.5 how it works

- ColBERTv1
  - "two-stage" retrieval method
    - preprocess representation of each token from the corpus is computed and indexed (FAISS) for nearest neighbor search
    - on query time
      - use each query term vector to retrieve top-k texts from corpus using the index (maximizin in a single query term)
      - these top-k texts for each term query are scored against all query tokens vectors using *MaxSim* for reranking

1.6 ColBERTv2

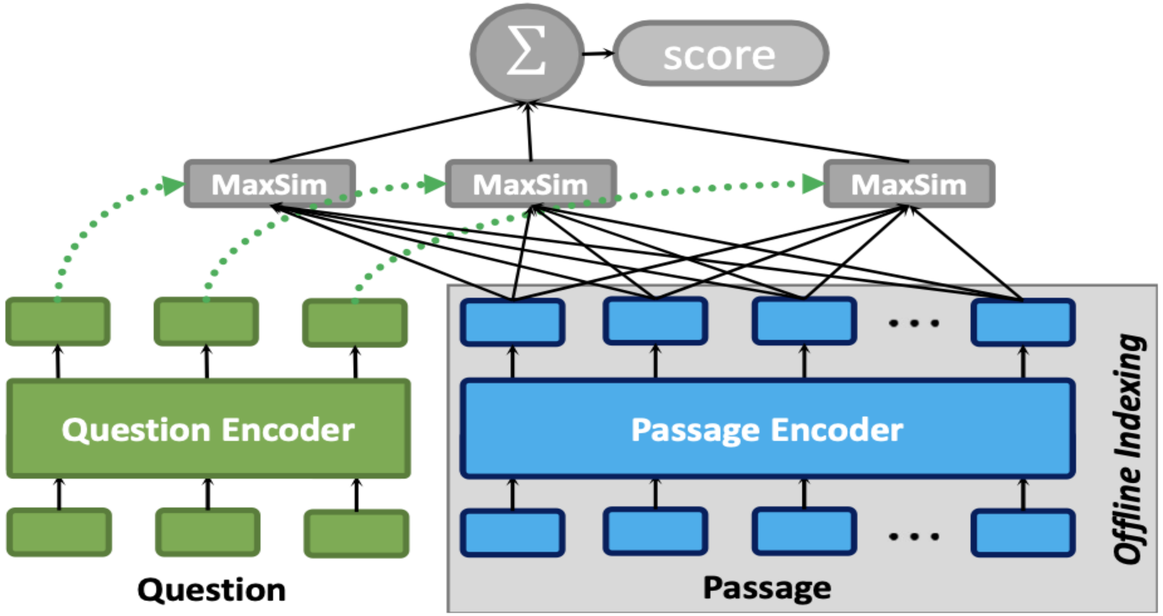


Figure 1: The late interaction architecture, given a query and a passage. Diagram from Khattab et al. (2021b) with permission.

1.7 how it works

- ColBERTv2
  - starting from a index of vectors from the corpus created as ColBERTv1
  - `training the retriever`
    - for each training query, the top-k passages are retrieved and feed each of those query–passage pairs into a cross-encoder reranker
    - use miniLM cross-encoder with distillation to train the reranker
    - after training refresh the index
  - `centroid index`
    - preprocess representation of each token from the corpus, `cluster them and use the term nearest centroids ids + a residual vector as its representation (smaller size)`
  - on query time
    - use each query term vector to retrieve the `nearest centroids from the inverted index` of texts from corpus
    - these top-k texts are scored against all query tokens vectors using *MaxSim*

2.1 contributions

- improvements on ColBERTv2 are mostly implementation
  - residual compression approach significantly reduces index sizes using cluster centroids over the token level vectors space
  - better negative selection (*hard-negative mining*) - in-batch
  - adds knowledge distillation from a cross-encoder miniLM for rerank the initial top-k docs/texts
  - ColBERTv1
    - 128dim vectors with 2 bytes = 256 bytes/vector
  - ColBERTv2
    - dimensionality reduction by arranging vectors in clusters indexed by 4 bytes (2^{32}) clusters)
    - improvement that enable 20-36bytes/vector
    - memory improvement ~6-10x (*residual compression*)
- multi-vectors are stored in cluster based on *MaxSim*
- new dataset *LoTTE (Long-Tail Topic-stratified Evaluation)*

2.2 contributions

- in-domain
  - beats DPR and $SPLADEv2$
- gigantic index
  - ColBERTv1 $154GiB$ 🤯
  - ColBERTv2 $16GiB(1bit)$ and $25GiB(2bit)$
- $MMR@10$
  - 1bit 36.2
  - 2bit 35.5
- success@5 metric
- LoTTE dataset

3.1 interesting/unexpected results

- **exact and soft match** - ColBERT can distinguish terms of which exact match is important
    - for each term check average score in exact and soft cases (how?), if the difference is higher then favors exact match otherwise favors soft match
- **how promote exact match from contextualized embeddings?**
    - hyp: frequent words have contextualized embedding pointing to different directions
        - for important terms, contextual embeddings vary less, thus ColBERT will tend to select same term in docs (*consine sim close to 1*)
        - terms carrying less information (is, the, as...) tend to absorb more the context in sequences, thus their embeddings vary more

4. basic doubts

- *long-tail topics* - `out of domain topics?`
- on ColBERTv1
  - the vector representation of each token is normalized to a unitary L2 norm; this makes computing inner products equivalent to computing cosine similarity.
  - relevance score is the sum of the max similarity between each vector of the query q and all doc d vectors

5. advanced topics

- how to make single vector methods works like ColBERT multi vector where it successed and vice-versa ?
- PLAID: An Efficient Engine for Late Interaction Retrieval same authors

Zeta Alpha Vector - ColBERT + ColBERTv2: late interaction at a reasonable inference cost