Retrieval/Matching Models

Peitian Zhang

Microsoft Research Asia

July 8, 2021

1. Basics
   - Workflow
   - Datasets
   - Metrics
2. Papers
   - Re-weight Methods
     - DeepCT
     - HDCT
   - Expansion Methods
     - Pseudo-feedback
     - Doc2Query
     - SNRM
     - SparTerm
   - Interaction Methods
     - ColBert
     - COIL
     - PolyEncoder
     - DC-Bert
     - Cascading

Workflow

Sparse Term-based Retrieval

- Create inverted-index for passages
- Traverse documents linked by the inverted list of the words in query and score them by retrieval models such as BM25 and query likelihood

Note

When the inverted index consists of passages, the score of the document is integrated by its passage scores

Workflow

Dense Embedding-based Retrieval

- Express each document and the query as a dense vector
- Retrieve documents that are the nearest k neighbors of the query vector

Datasets

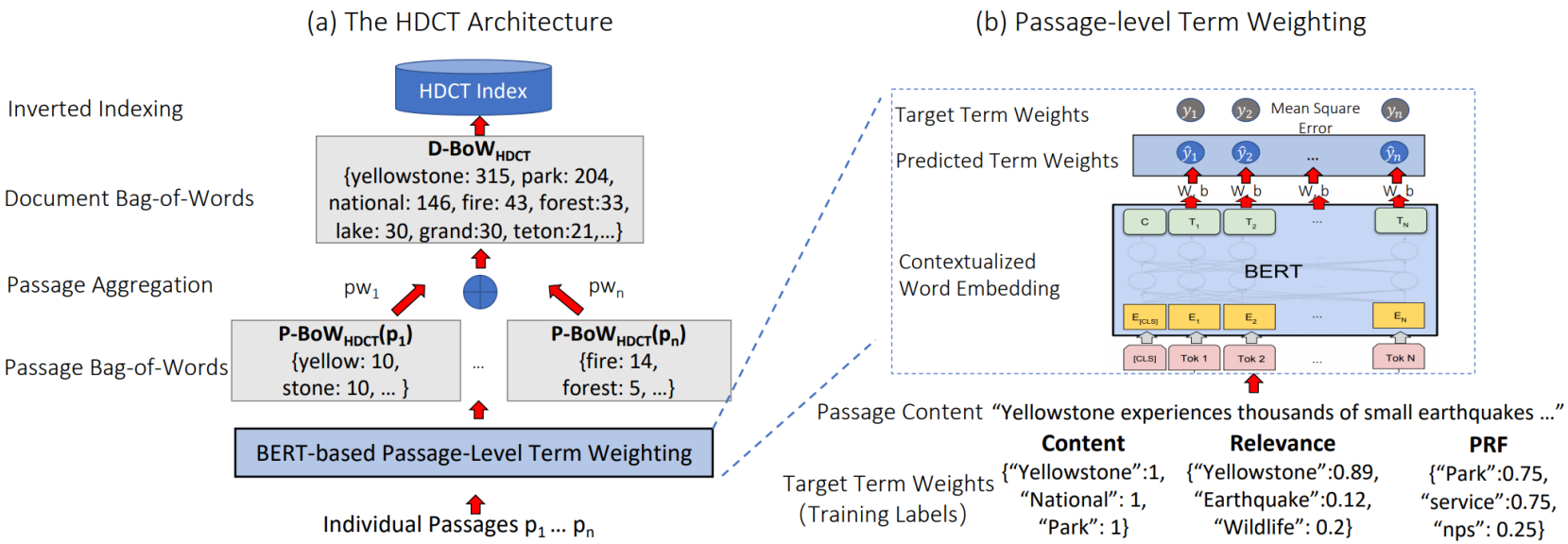
- MS MACRO Retrieval
- MS MACRO Rank

Metrics

- MRR
- Recall@K

Re-weight Methods

Re-weight the word in the document accoding to the contextualized embedding produced by Language Models.



DeepCT

Motivation: Generate the contextualized word weight according to its embeddings (produced by BERT).

- $$y = Wx + b,$$

  where x is the BERT embedding of the word, W and b are parameters.
- The weights is used to construct inverted index and applied in the first-stage retrieval.
- The ground-truth weight for each term in the document is estimated by the coocurrance of the term in the doc and the query, then use Mean Square Loss to train the regression model.

HDCT

Motivation: Generate contextualized term importance at the document-level, exceeding the word limitation of BERT.

- weighting terms within a passage (about 300 consecutive word sequence), then aggregate passages of a document.
- document-level evidence (whether a word appears in the important document fields e.g. titles and links, and whether the word appears in the query for the document) provides labels for training HDCT, which can down-weight unimportant documents.
- similar framework as DeepCT.

Expansion Methods

Expand the document/query to contain richer information. Utilize exact match and inverted index to retrieve.

Pseudo-feedback

Motivation: Simulate the first round of user feedback, facilitating the user to get improved retrieval performance without an extended interaction.

$$q^* = q + \lambda \sum_{i=1}^{k} d_i,$$

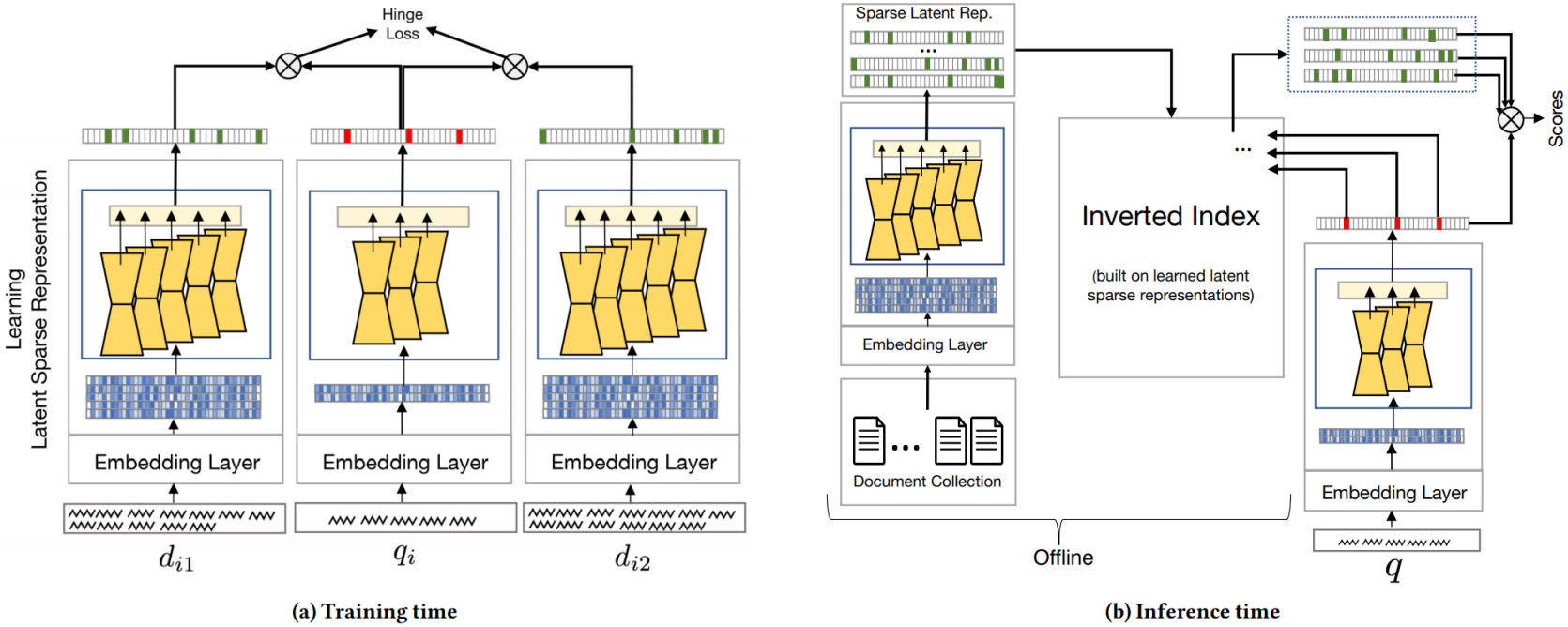
where d_i is the sparse vector of the relevant document with q accoding the retrieval model. q^* is the finalized query vector.

Doc2Query

Motiation: Generate a set of possible queries for each document using a seq-to-seq model and expand the document with these queries, in order to expand the *receptive field* of the document and facilitate retrieval models e.g. BM25.

SNRM

Motivation: Create a latent representation that aims to capture meaningful semantic relations while still efficiently matching documents.



(a) Training time

(b) Inference time

- The learnt representation of the document and the query should be sparse, and be in the same semantic space. The representation of the query should be sparser than that of the document, to ensure few computation when retrieval.

$$s(q,d) = \phi_Q(q)^\top \phi_D(d),$$

$$\phi_*(x) = \text{ReLU}\left(\frac{1}{|d|-n+1}\sum_{i=1}^{|d|-n+1}\phi(w_i,\cdots,w_{i+n-1})\right),$$

$$\phi(w_i,\cdots,w_{i+n-1}) = \text{MLP}\left(\bigoplus_{j=i}^{i+n-1} E_{w_j}\right).$$

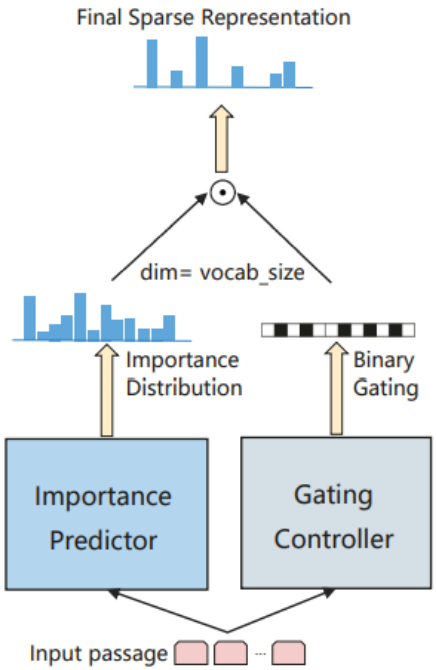
Insight

N-grams are used as w_i, I think Bert-based embeddings would be better.

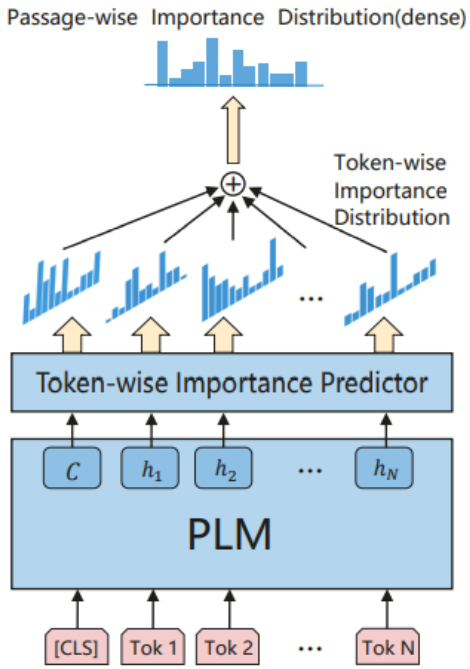
- In order to train the model to be effective in retrieval and sparse in representation, a unified loss composed of a hinge-loss and a L1-norm of the representation vector are proposed. The training signal (which one is more relevant to q between d_1 and d_2) is given by a pretrained query likelihood model.

SparTerm

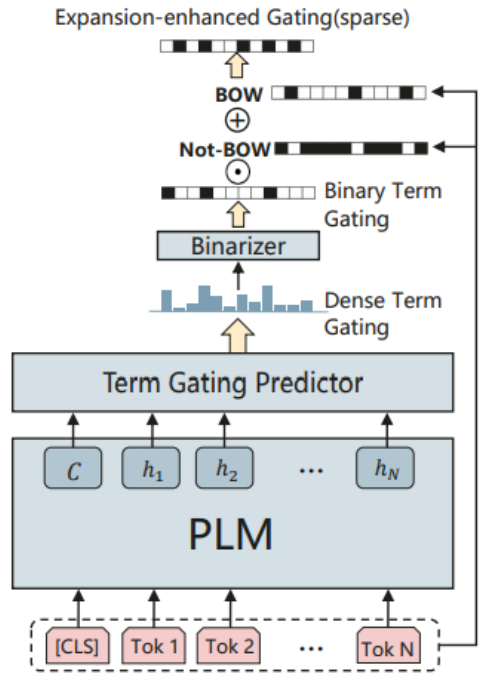
Motivation: Learn deep term-based sparse representation for each passage in the whole vocabulary space rather than the given text.



(a) SparTerm Model



(b) Importance Predictor



(c) Gating Controller

- Importance predictor: output the dense importance distribution of the whole passage.

$$I = \sum_{i=1}^{n} \text{transform}(r_i)E + b \in \mathbb{R}^{|V|},$$

where r_i is the Bert representation of the i-th word in the given passage, and E is the embedding matrix composed of the representation of every word, $|V|$ denotes the size of the vocabulary and transform is a function to normalize r_i.

- Gating Controller: output binarizer to impose the importance distribution to be sparse.

$$\hat{G} = G_{exp} + \text{BoW}(p),$$
$$G_{exp} = (G > \gamma) \odot \neg\text{BoW}(p),$$

where $\text{BoW}(p)$ is the binary BoW vector, G is the gating

distribution obtained by another *importance predictor* with independent parameters. γ is the threshold for G to be included in the expansion.

- Combine the two module to produce the sparse representation (term-based) for passage:

$$s(p) = \hat{G} \odot I(p)$$

- End-to-end training, Maximum log likelihood for *Importance predictor* and Cross entropy for *gating controller*.

Insight

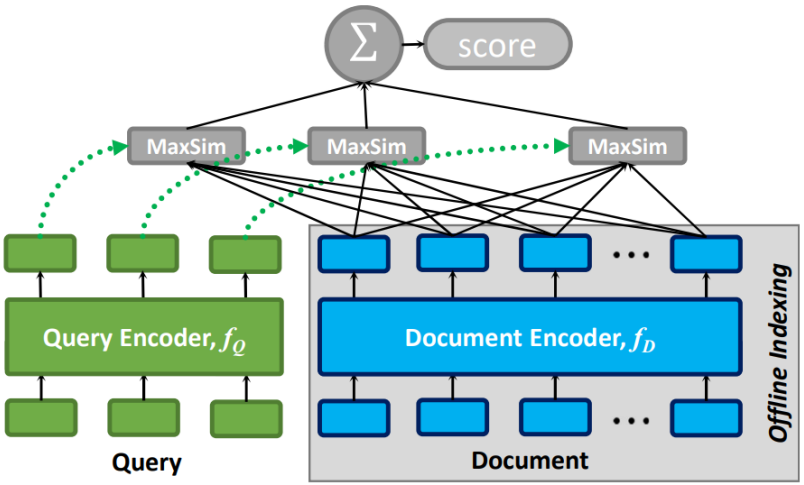
Equals to DeepCT when G_{exp} is $\mathbf{0}$

Interaction Methods

Use neural models to learn representation for each word in the document and the query, which interacts online.

ColBert

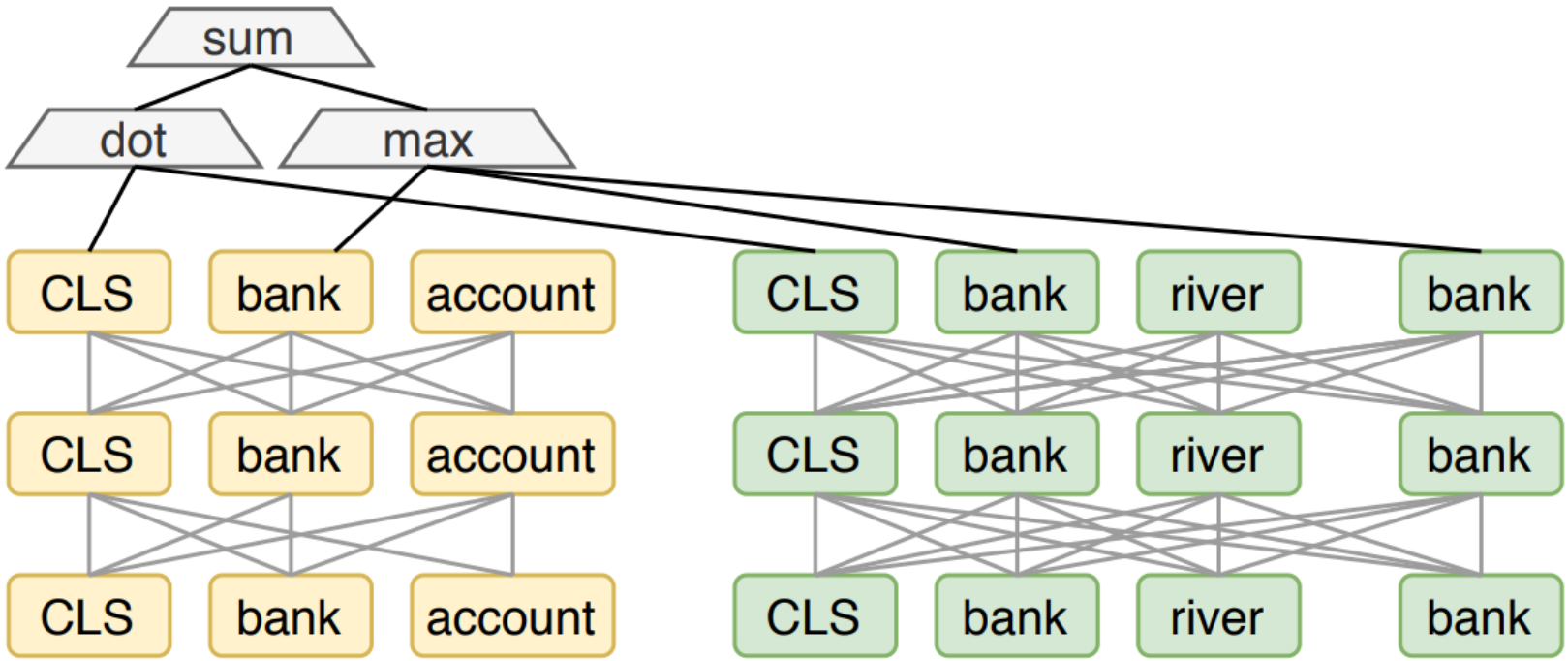
Motivation: Bert-based model is effective but slow, thus use *late interaction* (i.e. online interaction) to speed up inference.



- each query term interacts with each document term, the max score of the document term is kept, then the score of every query term is summed to form the final document score.
- MaxSim is compatible with approximate indexing, so can be used for ANN search (retrieval).

COIL

Motivation: Performs exact query-document token matching but compute matching signals with contextualized token representations instead of heuristics.

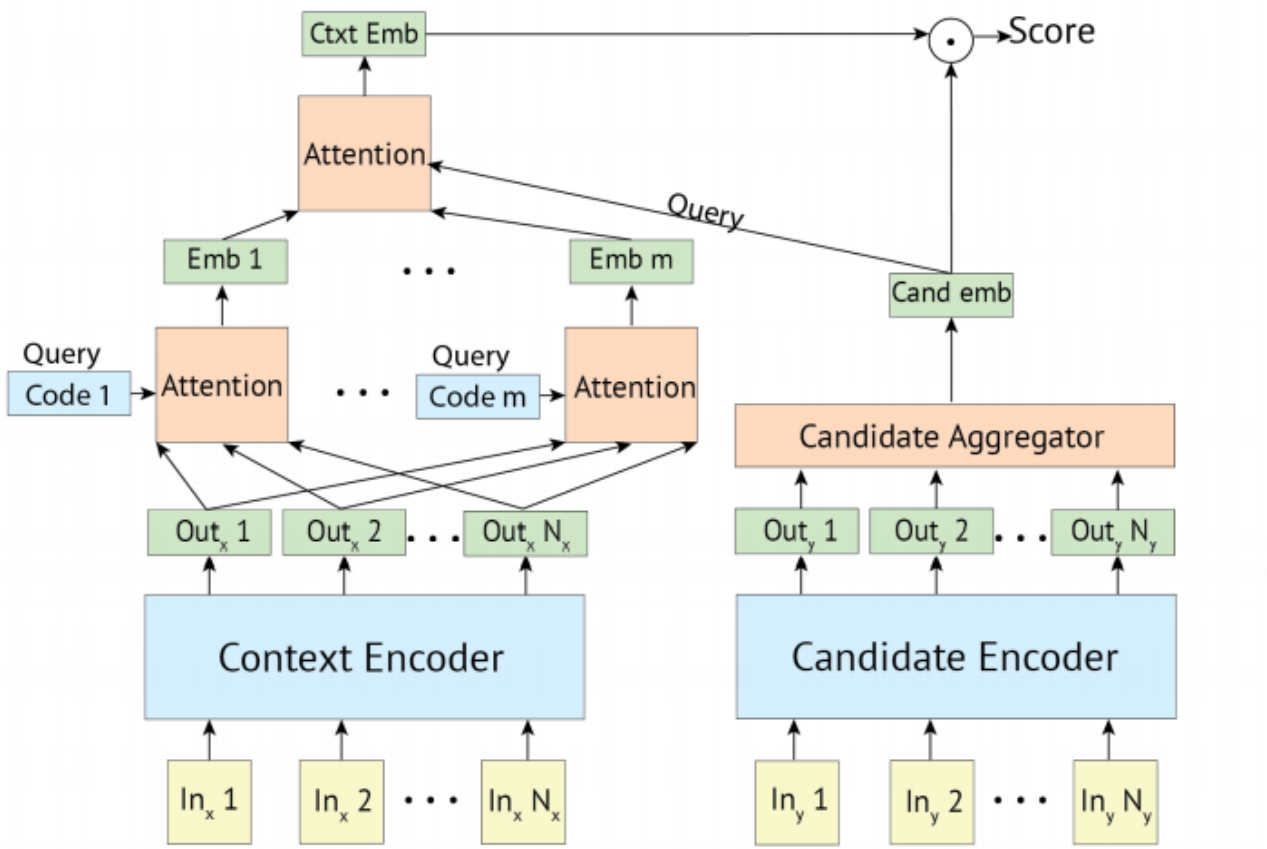


- $$s(q,d) = \sum_{t \in q \cap d} h_q(q,t)^\top h_d(d,t) + \mathsf{CLS}_q^\top \mathsf{CLS}_d$$

  where q is a query, d is a document and t is a co-occuring term between them. $h_q(q,t)$ gives contextualized embedding for t in q, $h_d(d,t)$ gives that of t in d.
- The contextualized vector of the same word $h_q(q,t), h_d(d,t)$ can be stored in an inverted list to be used for retrieval. Further, the vector can be structured by approximate index to speed up searching.

PolyEncoder

Motivation: Get the best of both worlds from the Bi- and Cross-encoder, where the formmer is efficient and the latter is effective.



DC-Bert

Motivation: Similar to polyencoder.