Article Presentation

IA368DD_2023S1: Deep Learning aplicado a Sistemas de Buscas

Student: Marcus Vinícius Borela de Castro

DPR (Dense Passage Retrieval for Open-Domain Question Answering) + Colbert (ColBERT Eficient and Efective Passage Search via Contextualized Late Interaction over BERT)

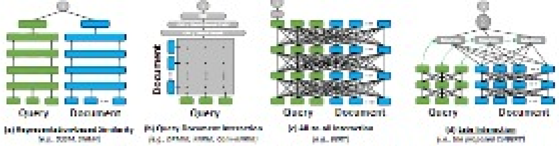

Query
Document
(a) Representation-based Similarity
(e.g., DSSM, SNRM)
Document
Query
(b) Query-Document Interaction
(e.g., DRMM, KNRM, Conv-KNRM)
Query
Document
(c) All-to-all Interaction
(e.g., BERT)
Query
Document
(d) Late Interaction
(i.e., the proposed ColBERT)

Figure 2: Schematic diagrams illustrating query–document matching paradigms in neural IR. The figure contrasts existing approaches (sub-figures (a), (b), and (c)) with the proposed late interaction paradigm (sub-figure (d)).

Main concepts

ColBERT
a ranking model based on
COntextualized Late Interaction
over BERT (figure 2.d)

DPR
a ranking model for Dense Passage Retrieval (figure 2.a)

Unlike rerankers, these 2 architectures allow prior generation of document embeddings, reducing the effort required during the query.

miro

FAISS (Facebook AI Similarity Search, Johnson et al., 2017)

FAISS is an extremely efficient, open-source Python package developed by Facebook AI Research for similarity search and clustering of dense vectors, which can easily be applied to billions of vectors. Faiss can implement algorithms for datasets of any size, including those that can not fit into the RAM

```
import faiss                   # make faiss available
index = faiss.IndexFlatL2(d)   # build the index, d=size of
# here we assume xb contains a n-by-d numpy matrix of type
index.add(xb)                  # add vectors to the index
print index.ntotal
```

This will display the number of indexed vectors. Adding to an `IndexFlat` just means copying them to the internal storage of the index, since there is no processing applied to the vectors.

To perform a search:

```


# xq is a n2-by-d matrix with query vectors
k = 4                          # we want 4 similar vectors
D, I = index.search(xq, k)     # actual search
print I
```

The output embeddings are normalized so each has L2 norm equal to one. The result is that the dot-product of any two embeddings becomes equivalent to their cosine similarity, falling in the [-1,1] range.

Why the range?

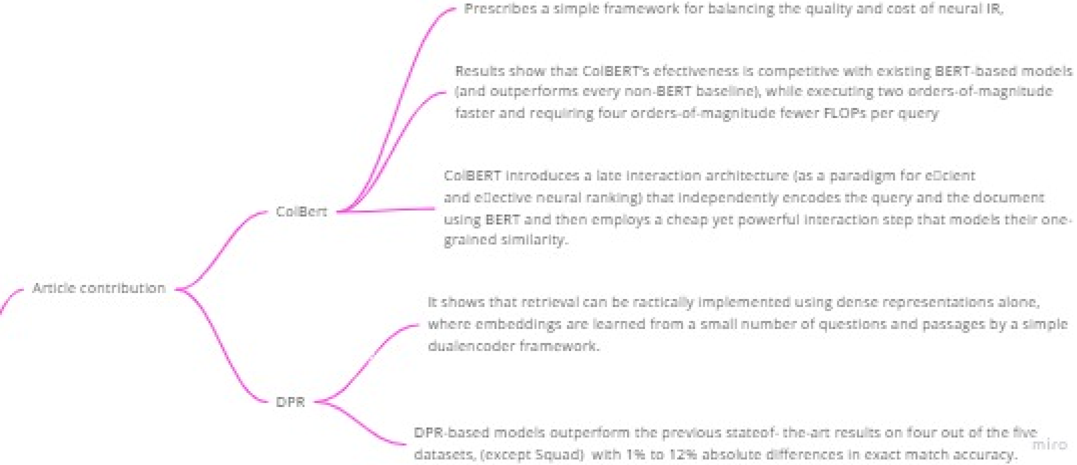
The cosine function has a range of [-1, 1], meaning that it can take values between -1 and 1

Why?

Dot Product: A · B = |A| * |B| * cos(θ)

When vectors are normalized, their magnitudes become 1, which means they lie on the surface of a unit sphere in the multi-dimensional space. In this normalized space, the magnitudes |A| and |B| become 1, and the dot product simplifies to:

Normalized Dot Product: A · B = cos(θ) = A1 * B1 + A2 * B2 + ... + An * Bn
(by chatGPT, adjuted)



Article contribution

ColBert

Prescribes a simple framework for balancing the quality and cost of neural IR,

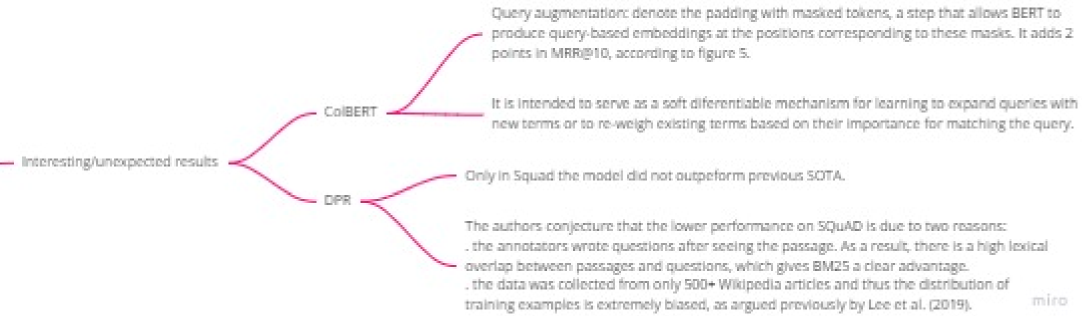
Results show that ColBERT's efectiveness is competitive with existing BERT-based models (and outperforms every non-BERT baseline), while executing two orders-of-magnitude faster and requiring four orders-of-magnitude fewer FLOPs per query

ColBERT introduces a late interaction architecture (as a paradigm for e□cient and e□ective neural ranking) that independently encodes the query and the document using BERT and then employs a cheap yet powerful interaction step that models their one-grained similarity.

DPR

It shows that retrieval can be ractically implemented using dense representations alone, where embeddings are learned from a small number of questions and passages by a simple dualencoder framework.

DPR-based models outperform the previous stateof- the-art results on four out of the five datasets, (except Squad) with 1% to 12% absolute differences in exact match accuracy.

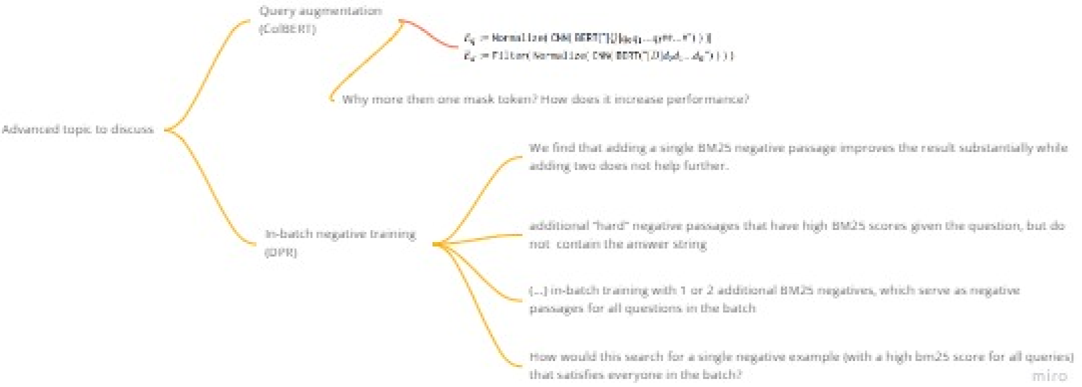


- Interesting/unexpected results
  - ColBERT
    - Query augmentation: denote the padding with masked tokens, a step that allows BERT to produce query-based embeddings at the positions corresponding to these masks. It adds 2 points in MRR@10, according to figure 5.
    - It is intended to serve as a soft diferentiable mechanism for learning to expand queries with new terms or to re-weigh existing terms based on their importance for matching the query.
  - DPR
    - Only in Squad the model did not outpeform previous SOTA.
    - The authors conjecture that the lower performance on SQuAD is due to two reasons:
      . the annotators wrote questions after seeing the passage. As a result, there is a high lexical overlap between passages and questions, which gives BM25 a clear advantage.
      . the data was collected from only 500+ Wikipedia articles and thus the distribution of training examples is extremely biased, as argued previously by Lee et al. (2019).

Basic doubts that may arise

"We use the batch size of 16 for large (NQ, TriviaQA, SQuAD) and 4 for small (TREC, WQ) datasets" (DPR)

Why should a small batch size be used for a small dataset (and vice versa)?



Advanced topic to discuss

Query augmentation (ColBERT)

$E_q := \text{Normalize}(\text{CNN}(\text{BERT}(\text{"[Q]}q_0q_1...q_l\#\#...\#\text{"})))$

$E_d := \text{Filter}(\text{Normalize}(\text{CNN}(\text{BERT}(\text{"[D]}d_0d_1...d_n\text{"}))))$

Why more then one mask token? How does it increase performance?

In-batch negative training (DPR)

We find that adding a single BM25 negative passage improves the result substantially while adding two does not help further.

additional "hard" negative passages that have high BM25 scores given the question, but do not contain the answer string

(...) in-batch training with 1 or 2 additional BM25 negatives, which serve as negative passages for all questions in the batch

How would this search for a single negative example (with a high bm25 score for all queries) that satisfies everyone in the batch?