CS29S Project Report

Sparse-Dense Hybrid Retrieval

Wentai Xie, Yanze Wu

DEC 2021

1 Objective and Challenges

The present document retrieval methodologies can be categorized into two main divisions: sparse retrieval and dense retrieval. Both these two divisions have different challenges: On one hand, generally dense retrieval has a higher accuracy while suffering from the costly online query latency. On the other hand, the out-of-vocabulary terms and misleading term frequency might damage sparse retrieval.

To find a potential solution given these challenges, we learnt about the state of art model: TCT-Colbert and DeepImpact. We tried to combine these two models as a solution that makes up the deficit in both retrieval methodologies.

2 DeepImpact

2.1 Contribution

The Contribution of DeepImpact can be concluded as follow:

I. Use query to enrich the original document terms, which alleviate the poor representation over the synonym words.
II. Calculate the Impact Score instead of term frequency to have a more capable representation of the document.

2.2 Main Idea

The overall DeepImpact network consists of three parts: Contextualized encoder (e.g Bert), expansion term encoder, and Impact Score Encoder.

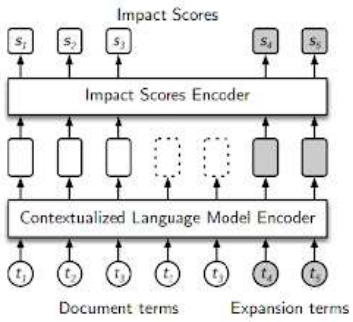


Figure 2.1: Neural network architecture

The paper utilizes DocT5query to generate the expansion term given a document, which enriches the vocabulary in the document. The Contextualized Language Model is a model like Bert. And Impact Score Encoder is a simple 2-layer MLP.

The DeepImpact network will first generate expansion for the given input document. Then the original documents and their expansion will then go through the contextualized language model. This will provide word embedding for each term. After that, an impact score is encoder recalculate the important score of each term.

The first occurrence terms in the input of the intercept between the query term and the expanded document will correspondingly be the input of the Impact Score Encoder. The Impact Score Encoder will then take each one term at a time and give its score.

The document score is the sum of terms impact score which appears in both query and document. And the paper uses pair-wise training, which aims to maximize the difference between positive document score and negative document score.

3 TCT-Colbert

3.1 Contribution

The Contribution of TCT-Colbert can be concluded as follow:

I. Simplify the ColBERT's similarity function MaxSim into a dot-product so that it can solve the expensive storage and computing of Colbert.
II. Proposed Knowledge distillation to train student model.
III. Purpose Hybrid Dense-Sparse Ranking

3.2 Main Idea

As is shown in figure 3.1, the framework of Colbert has a model-teacher model and student model coupling tightly. It uses a fine-tuned Colbert as the teacher model and the student model is based on Colbert.

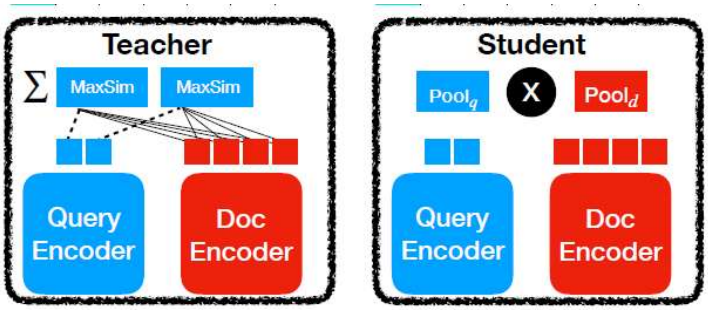


Figure 3.1: TCTColBERT framework

Although Colbert is able to do the passage retrieval efficiently, TCT-Colbert wants to simplify it further. They use an average pooling function over the output embeddings of documents and queries to reduce their dimension. Instead of MaxSim to evaluate their similarity, they use a dot product between the document vector and the query vector.

To train their model. The proposed knowledge distillation, by which their model could learn from the teacher model (Colbert), gives similar predictions while cutting down the computation.

$$\mathcal{L} = -\sum_{i=1}^{|\mathcal{B}|} \left\{ \begin{array}{l} \gamma \cdot \mathbb{1}_{\mathbf{d}_i \in \mathcal{T}_{q_i}^+} \log(P(\mathbf{d}_i|\mathbf{q}_i)) - \\ (1-\gamma) \sum_{\mathbf{d}' \in \mathcal{D}_{\mathcal{B}}} \mathrm{KL}(\hat{P}(\mathbf{d}'|\mathbf{q}_i) || P(\mathbf{d}'|\mathbf{q}_i)) \end{array} \right\},$$

The first term is a softmax cross-entropy loss function which asks students to correctly classify documents. The second term KL is a measure of the difference

between teacher sample distribution and student sample distribution, where they tried to minimize the difference.

4 Margin MSE

Here we only briefly talked about Margin MSE, where we use to train our own TCT-Colbert model, instead of the training method proposed in section 3.2.

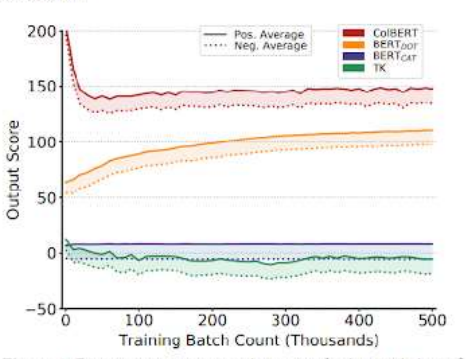


Figure 4.1 margin score discrepancy

From Figure 4.1, these 4 models margins have a very similar margin (difference between the positive score and negative score). This leads to a training idea that only learning the margin knowledge of the teacher model.

To do this, the loss function is proposed as follow:

$$\mathcal{L}(Q, P^+, P^-) = \mathrm{MSE}(M_s(Q, P^+) - M_s(Q, P^-), \\ M_t(Q, P^+) - M_t(Q, P^-))$$

Which minimizes the mean square loss between the Teacher margin (positive score subtract negative score) and the student margin.

5 Experiments

5.1 DeepImpact

We will talk about how we reproduce the DeepImpact and some evaluations in this part. For the Contextualized Language Model Encoder, we use Bert from Hugging Face. We use DocT5query to generate query sentences for the given input document.

The Impact Score is a two-layer perceptron network. We set the dropout rate equals 0.5. the input size is 768 (the embedding size of Bert), and the hidden layers start with a map from a 768 vector to a 768 vector, then follow reduction from 768 dimensions to 1.

Before running through the entire DeepImpact model, we will do an intercept between the document and query, and only the corresponding embedding will be the input of the Impact Score Encoder, as mentioned in Section 2.2.

We use MAMARCO triple dataset to train our model for 100,000 iterations. The learning rate is set to 3×10−6. To solve the GPU memory limitation, we divided our batch into 4 smaller batches, and the overall loss will be the normalization of the average gradient of 4 small batches.

5.2 TCT-Colbert

We first fine-tune our teacher model (Colbert) from Hugging Face. Then we copied the Colbert model to our TCT-Colbert Model, and change its Maxsim into an average pooling follows a dot product.

For the teacher model, we use MSMACRO triple dataset and cross-entropy loss to fine-tune the model. We train the teacher model for 2000 iteration and a learning rate of 3×10−6.

Instead of using the loss function originally presented in TCT-Colbert, we use Margin MSE loss to evaluate the performance between student and teacher. Our maximum iteration for training was set to 100,000 with the same learning rate of 3×10−6. However, we found that the model already converged around 20,000 iterations, hence we did an early stop.

5.3 Hybrid document retrieve

We use a linear combination of these two models as our final output.

We manually set w=0.35 which gives the best classification accuracy on MSMARCO.

6 Evaluations

6.1 Accuracy Evaluation

The simplest evaluation method is to see if the model could correctly classify the triple dataset.

Our validation runs on MSMARCO triple dataset, and we also skip the first 100,000 rows in case of any interception between the training set and the validation set. We also included a TCT-Colbert model from Hugging Face lib as a baseline against our self implemented TCT-Colbert.

	TCT-Colbert Standard	Colbert (Teacher model)	TCT-Colbert Margin MSE	DeepImpact	Hybrid
Acc	0.9745	0.9896	0.9840	0.9436	0.9843

Table 6.1 Models Accuracy

From Table 6.1, we can claim that Margin MSE is a more efficient way to train the student model. And dense retrieval is consistent with our previous statement that

‘dense has a higher accuracy’. The hybrid model did have some improvements, but not very much, other combination methods might be expected to carry out a more reasonable prediction.

We also implemented MAP evaluations to see how far our models can go. We sampled 50 queries in Marco TOP1000, and for each query, we recalculate their similarity score using different methods and compare it to ground truth (qrels).

	DeepImpact	TCT-Colbert	Hybrid (w=0.35)
MAP	0.344	0.366	0.424
MRR@10	0.339	0.352	0.422

Table 6.2 MAP Score

The DeepImpact MAP score is close to the paper MAP score (0.332), hence we assume we successfully reproduce DeepImpact Model. We also noticed that the hybrid retrieval is a huge improvement that before, even though it doesn’t increase that much in classification accuracy. We discovered that the MAP scores for each query predicted by DeepImpact go extremes: Its output mostly equals 1 or nearly zero. On the other hand, TCT-Colbert outputs mostly are stay between 0.5 to 0.125. A balance between these two features helps to reach a higher MAP score.

7 Indexing

We also tried to build an index that allows us to search our own documents on Pyserini.

Our index is built on top of the MSMARCO Top1000 dataset. Considering that MSMARCO Top1000 dataset has about 6,660,000 documents which are nearly impossible to index all, we sampled 500 queries from the dataset and picked only the documents that only related to those 500 queries. We use them to build our index.

For sparse indexing, we download the official deepimpact structure and mimick the official data structure. Our index structure looks like this:

```
{‘id’:id,’content’:article content,vector={…}}
```

Where the vector looks like:

```
{‘term1’:score1,’term2’:score2}
```

For dense indexing, we use Faiss to index the document vector produced by our TCT-Colbert model. Then same as sparse indexing, we manually index it into Pyserini, and use a dense searcher to retrieve our documents. We follow the Pyserini document says they are using a Faiss index, and IndexSearch did indicate the correct document total number. But we are not able to mimic the official document index this time, because when loading index using Faiss, our memory cannot hold so much index.

Our indexing work has several potential improvements:

First, we are only able to provide index, we are still using an official query tokenizer and pre-processing when doing online query. We find out that the preprocessing and tokenizer of the query are different from our query preprocessing, which makes it inaccurate.

Second, the memory limitation (Faiss index) which makes it impossible for the personal laptop to read the official data structure also adds to the volatility.

8 Our Efforts

Most of our retrieval methods are related to the BERT model introduced in class. And we sum up our efforts into following points:

I. We learned four papers before we started our project

II. We successfully reproduced DeepImpact Model and TCT-Colbert model. And use innovative training methods like MSE instead of the method TCT-Colbert originally introduced. We also use a hybrid combination of DeepImpact and TCT to achieve a better retrieval performance.

III. We learned a lot about MSMARCO data structure, and dive into pyserini different index structures, and manage to search our own document using self-created indexing

IV. The expensive time input, training like DeepImpact takes two days, indexing 500,000 documents takes half-day, and so on. We also solved issues like GPU memory limitations.

V. We implemented MAP and MMR evaluation ourselves. We also did a lot of speculation about how we carried out our evaluations.

Reference

1. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers
Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin arXiv:2010.11386

2. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation
Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, Allan Hanbury arXiv:2010.0266

3. Learning Passage Impacts for Inverted Indexes
Antonio Mallia, Omar Khattab, Nicola Tonellotto, Torsten Suel arXiv:2104.1206

4. Colbert: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT
Jimmy Huang, Yi Chang, Xueqi Cheng SIGIR '20

2021
Project Presentation

Wentai Xie，Yanze Wu

01

Introduction

TCT-ColBERT: an approach to ranking with dense representations

1. Adopt the “late interaction” **ColBERT** model(Khattab and Zaharia,2020)

2. Use knowledge distillation (Hintonet al., 2015) to simplify the ColBERT’s similarity function **MaxSim** into a dot-product (Motivation: Solve the existing problem of ColBERT)

3. **Hybrid Dense-Sparse Ranking**

References

[1] Sheng-Chieh Lin，Jheng-Hong Yang. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers

[2] Antonio Mallia, Omar Khattab, Torsten Suel，Nicola Tonellotto . 2021. Learning Passage Impacts for Inverted Indexes

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

02

Methodology

TCT-ColBERT famework

TCT-ColBERT: A bi-encoder—Tight coupling between teacher and student models

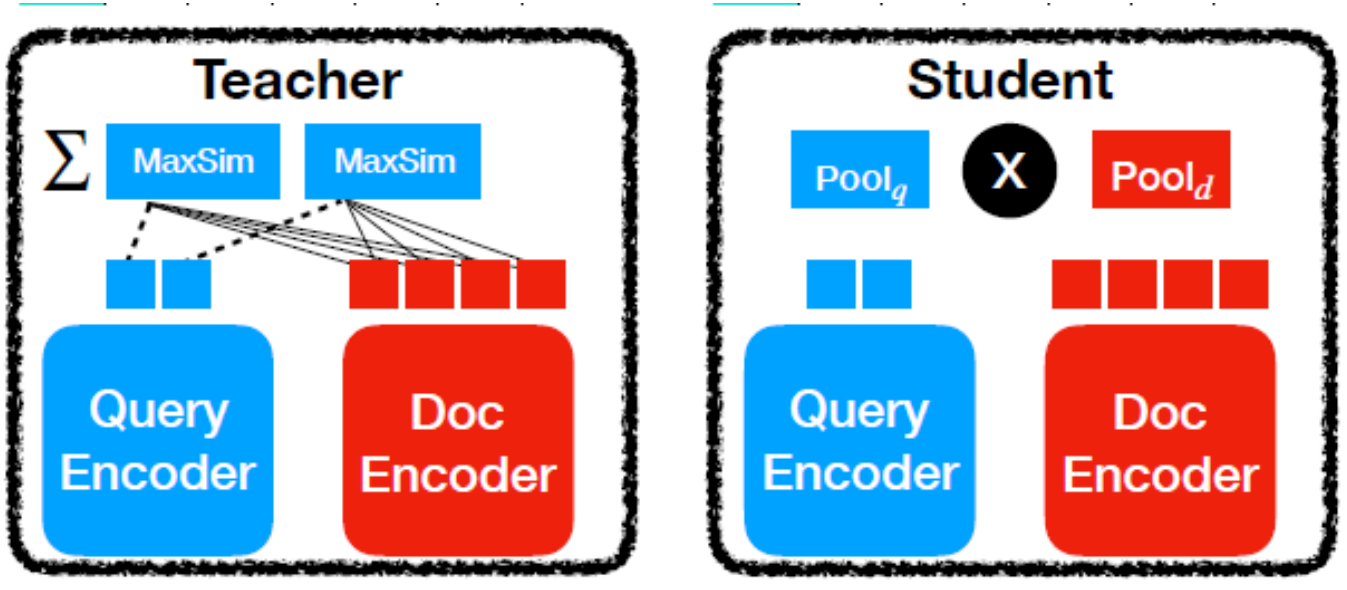


Figure1 : TCT-ColBERT famework

ColBERT

Query and document encoder

ColBERT is a highly-effective model that employs novel BERT-based query and document encoders within the **late interaction** paradigm.

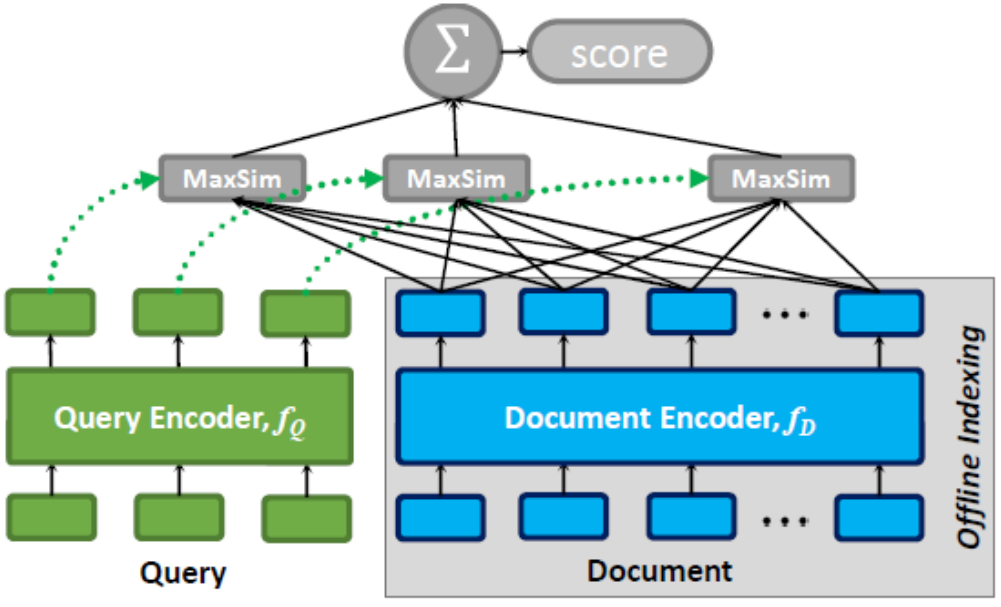


Figure2 ColBERT famework

fQ： a query encoder, encodes query q into a bag of fixed-size embeddings Eq

Eq := Normalize(CNN(BERT(“[Q]Where is UCSB?”)))

fD： a document encoder, encodes document d into a bag of fixed-size embeddings

Ed := Filter(Normalize(CNN(BERT(“[D]I am studying at UCSB”))))

Query and document encoder

Example:
Given a textual query q and document d .
1.tokenize them into their BERT-based Word Piece tokens[1]

Query q “[Q],q0,q1,q2,q3,……qn”

Document d “[D],d0,d1,d2,d3,……,dn”

Eq :=
Normalize(CNN(BERT(“[Q]$q0q1…ql$”)))
Ed :=
Filter(Normalize(CNN(BERT(“[D]$d0d1…dn$”))))

[Q]:a special token distinguish from document token [D]
[D]:a special token distinguish from query token [Q]

2.The sequence will go through the BERT and subsequent linear layer

3.Normalize so each has L2 norm=1
Filter out the punctuation symbols

[1] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi,Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al.2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016).

Some examples

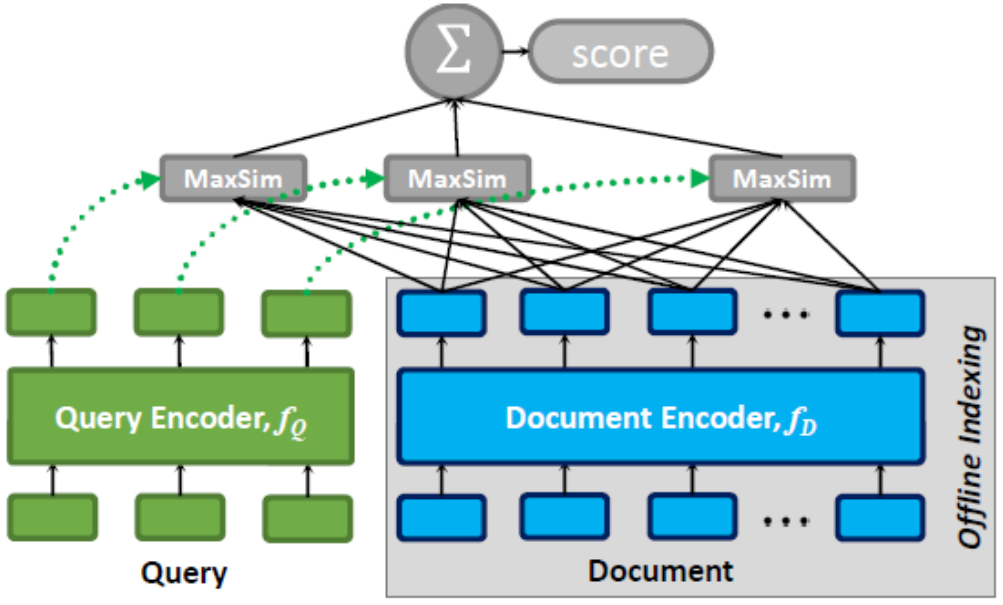
```
text = "Where is UCSB?"
token = bertTokenizer.encode(text,return_tensors="pt").to(device)
print(token)
```

```
tensor([[  101,  2073,  2003, 15384, 19022,  1029,   102]], device='cuda:0')
```

```
outputs = bertModel(token)
states = outputs.last_hidden_state
print(states.shape)
print(states)
#the each term is converted into a 768 vector
```

```
torch.Size([1, 7, 768])
tensor([[[-0.1036,  0.2285, -0.0987,  ..., -0.5736,  0.3942,  0.5410],
         [-0.5306, -0.5094,  0.2621,  ..., -0.4189,  0.4367,  0.0743],
         [-0.2126, -1.0201,  0.1537,  ..., -0.6653,  0.5390,  0.6819],
         ...,
         [-0.8555, -0.6644, -0.3412,  ...,  0.4427,  0.6662,  0.3402],
         [ 0.0547, -0.4813, -0.8176,  ..., -0.0174,  0.2172, -0.0255],
         [ 0.8550,  0.0265, -0.2815,  ...,  0.2419, -0.5727, -0.1818]]],
       device='cuda:0', grad_fn=<NativeLayerNormBackward0>)
```

MaxSim



Eq： the output embeddings from query encoder fq

Ed： the output embeddings from document encoder fD

Maxsim: maximum similarity (e.g., cosine)

Score: The relevance score of d to q

$$S_{q,d} := \sum_{i \in [|E_q|]} \max_{j \in [|E_d|]} E_{q_i} \cdot E_{d_j}^T$$

MaxSim example

Score: The relevance score of d to q

$$S_{q,d} := \sum_{i \in [|E_q|]} \max_{j \in [|E_d|]} E_{q_i} \cdot E_{d_j}^T$$

Example:

For query q, the output embeddings from query encoder-Eq:

Query Term 1 (Eq_1): [0.1 , 0.4 , 0.5,……]

Query Term 2 (Eq_2): [0.4 , 0.2 , 0.3,……]

……

Query Term n (Eq_n): [0.2 , 0.5 , 0.6 ,……]

For document d, the output embeddings from document encoder-Dq:

Document Term 1 (Ed_1): [0.2 , 0.3 , 0.7,……]

Document Term 2 (Ed_2): [0.1 , 0.4 , 0.6,……]

……

Document Term m(Ed_m):[0.4 , 0.2 , 0.7 ,……]

$Sq,d=\max\{ Eq_1 \cdot Ed_1^T , Eq_1 \cdot Ed_2^T , Eq_1 \cdot Ed_3^T ,……, Eq_1 \cdot Ed_m^T\}$

$+\max\{ Eq_2 \cdot Ed_1^T , Eq_2 \cdot Ed_2^T , Eq_2 \cdot Ed_3^T ,……, Eq_2 \cdot Ed_m^T\}$

$+\max\{ Eq_3 \cdot Ed_1^T , Eq_3 \cdot Ed_2^T , Eq_3 \cdot Ed_3^T ,……, Eq_3 \cdot Ed_m^T\}$

+……

$+\max\{ Eq_n \cdot Ed_1^T , Eq_n \cdot Ed_2^T , Eq_n \cdot Ed_3^T ,……, Eq_n \cdot Ed_m^T\}$

TCT-ColBert optimization

Student model

Assume the kernel dimension of BERT is 756, a document d contains 100 terms

Teacher encoder
ColBERT encoder outputs:E_d
(100,756)
->Large storage cost,high latency

Student encoder
TCT-ColBERT encoder outputs: AvgPool(E_d)
(1,756)

Saving storage, compute with a dot product

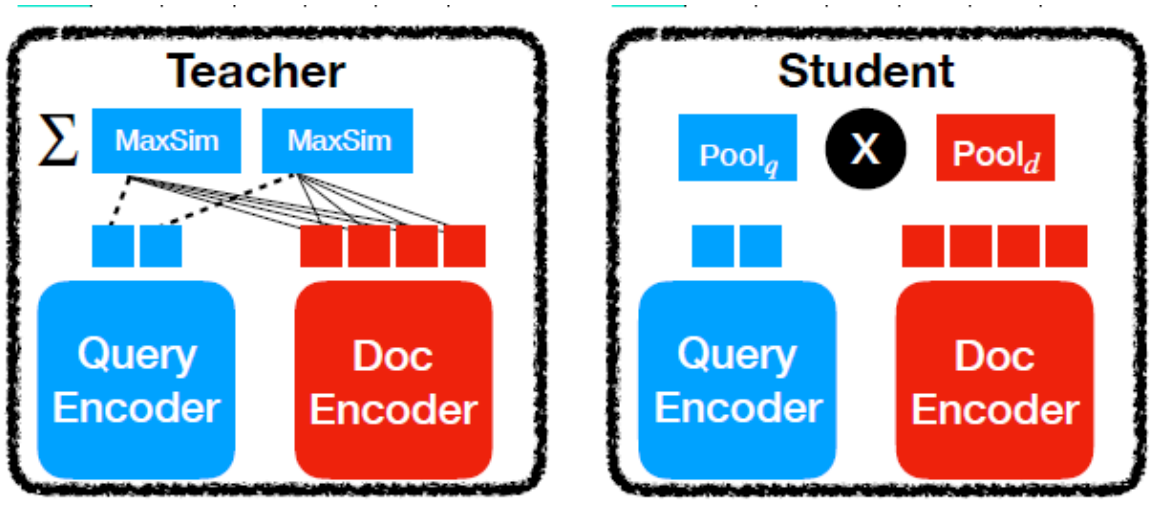


Figure1 : TCT-ColBERT famework

Using Tightly coupled teacher to minimize the lost

Teacher: ColBert
Student: TCT-ColBert (identical)

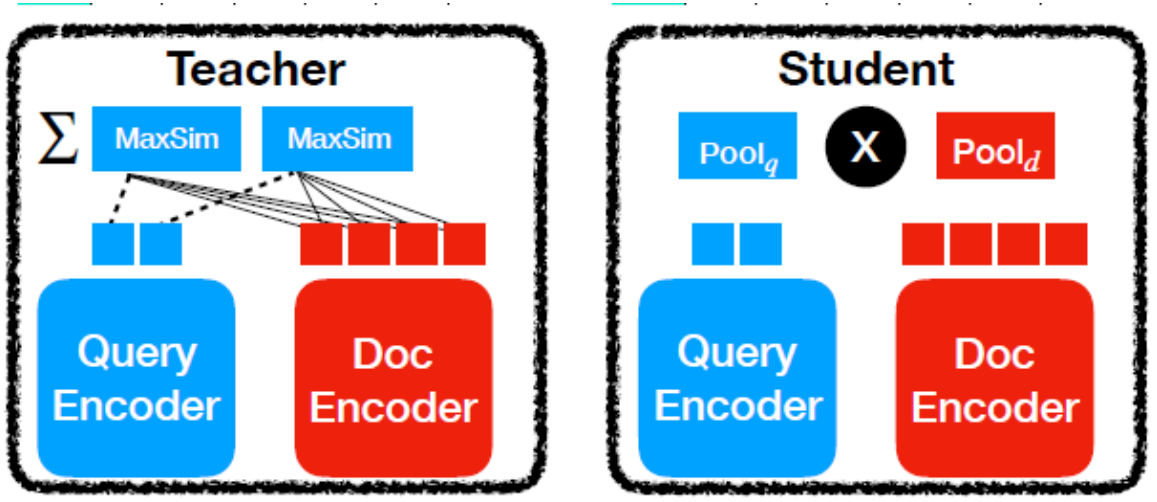
Train the student model encoder to output a well-behaved pooled embedding

$\gamma \cdot 1_{di \in T} + q_i \log(P(d_i|q_i))$ $(1-\gamma) \times d0 \in DB\ KL(P\hat{}(d\ 0\ |q_i)||P(d\ 0\ |q_i))$

Means the relative document of the given query must be correctly classified as relative

The non-relative documents are expected have a similar output score

γ is a weight



Hybrid Dense-Sparse Ranking

01

Dense Retrieval

Features：

- simple but effective
- don't need to be trained
- work on any language

Limitation：

- **High online processing latency**

ColBERT

02

Sparse Retrieval

Features：

- powerful but computationally more expensive especially during indexing
- trained using labelled datasets
- language specific

Limitation：

- **Imprecise term frequency**
- **Cannot handle out-of-vocabulary**

DeepImpact

Hybrid Dense-Sparse Ranking

A single dense embedding **cannot sufficiently represent** passages(Luan et al. (2020); Gao et al. (2020))

So we use a **linear combination** of sparse and dense retrieval--**Hybrid Dense-Sparse Ranking:**

$$\Phi(q,d) = \alpha \cdot \Phi_{sp}(q,d) + \Phi_{ds}(q,d)$$

$\Phi(q,d)$: The relevance scores of d to q

α： weight

We are using TCT-Colbert as dense ranking, deep Impact for sparse ranking

For example, we can evaluate the relevance between the query and document as shown below

The relevance of sparse representation	$\Phi_{sp}(q,d)_1=0.7$	$\Phi_{sp}(q,d)_2=0.5$	$\Phi_{sp}(q,d)_3=0.2$,……
The relevance of dense representation	$\Phi_{ds}(q,d)_1=0.5$	$\Phi_{ds}(q,d)_2=0.8$	$\Phi_{ds}(q,d)_3=0.9$,……
If α=3 highlight sparse	$\Phi(q,d)_1= 2.6$	$\Phi(q,d)_2= 2.3$	$\Phi(q,d)_3= 1.5$,……
If α=0.3 highlight dense	$\Phi(q,d)_1= 0.71$	$\Phi(q,d)_2= 0.95$	$\Phi(q,d)_3= 0.96$,……

03

Datasets and result

Datasets and result

Datasets : MS MARCO ,TREC-2019

Result :

Table 1: Main results on passage retrieval tasks.

	MS MARCO dev		TREC2019 DL		latency
	MRR@10	R@1000	NDCG@10	R@1000	(ms/query)
Sparse retrieval (Single Stage)					
BM25	0.184	0.853	0.506	0.738	55
DeepCT (Dai and Callan, 2020)	0.243	0.913	0.551	0.756	55
doc2query-T5 (Nogueira and Lin, 2019)	0.277	0.947	0.642	0.802	64
Dense retrieval (Single Stage)					
ANCE (Xiong et al., 2020)	0.330	0.959	0.648	-	103
Bi-encoder (PoolAvg)	0.310	0.945	0.626	0.658	103
Bi-encoder (TCT-ColBERT)	0.335	0.964	0.670	0.720	103
Multi-Stage					
ColBERT (Khattab and Zaharia, 2020)	0.360	0.968	-	-	458
BM25 + BERT-large (Nogueira and Cho, 2019)	**0.365**	-	0.736	-	3,500
Hybrid dense + sparse (Single Stage)					
CLEAR (Gao et al., 2020)	0.338	0.969	0.699	0.812	-
Bi-encoder (PoolAvg) + BM25	0.342	0.962	0.701	0.804	106
Bi-encoder (TCT-ColBERT) + BM25	0.352	0.970	0.714	0.819	106
Bi-encoder (PoolAvg) + doc2query-T5	0.354	0.970	0.719	0.818	106
Bi-encoder (TCT-ColBERT) + doc2query-T5	0.364	**0.973**	**0.739**	**0.832**	106

Thanks for listening