Condenser and Adaptive Batch Scheduling for OpenQA Passage Retrieval

Stanford CS224N Custom Project
Mentor:Anna Goldie

Yiting Zhao
Department of MS&E
Stanford University
`yitzhao@stanford.edu`

Xiaoying Yang
Department of Computer Science
Stanford University
`xyang123@stanford.edu`

Zikun Cui
Department of CEE
Stanford University
`cuizk@stanford.edu`

Abstract

Recent research demonstrates the effectiveness of using fine-tuned language models for openQA passage retrieval. However, retrievers are hard to train. In this paper, we tried to improve passage retrieval accuracy by training Condenser model with negative samples generated by Adaptive Batch Scheduling (ABS). We also tried different scheduling algorithms for ABS in order to improve the retrieval accuracy. Our current experiments indicate that Combining Condenser with ABS using dot-product similarity can bring significant improvement in retrieval accuracy.

1 Introduction

Document retrieval for open-domain question answering (OpenQA) is an important research that can be applied to various areas, such as search engine. Thus it has gained extreme popularity among academia and industry. Given a question, most QA systems first search for relevant documents, and then they extract answers from the retrieved documents. To find the relevant documents, dense retrieval architectures are explored. The cross encoder architecture passes questions q_i and passages p_j pairs directly into the model to calculate the relevance score between them, while the dual encoder encodes the given question into dense representation and then compare it with corresponding passage representations.

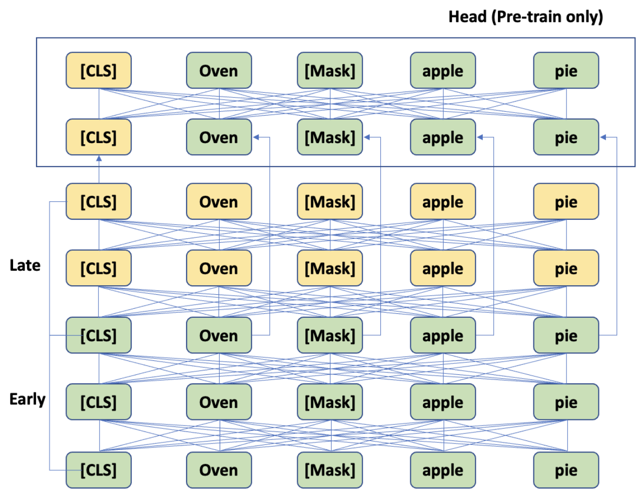

Figure 1: Condenser Architecture

However, neither of these two architectures are trained to aggregate sophisticated information into a single dense representation. To address this issue, [1] introduces a novel transformer pre-training architecture, Condenser (Figure 1), which establishes structural readiness by doing language model pre-training actively CONdition on DENSE Representation.

In-batch negative sampling is widely used to provide extra negative training samples. Figure 2 shows the process of in-batch negative sampling. Given a batch with n pairs of (q_i, p_i), passages other than p_i are considered as negative samples of q_i. The training objective is to choose positive passage among all the samples for a given question q_i.

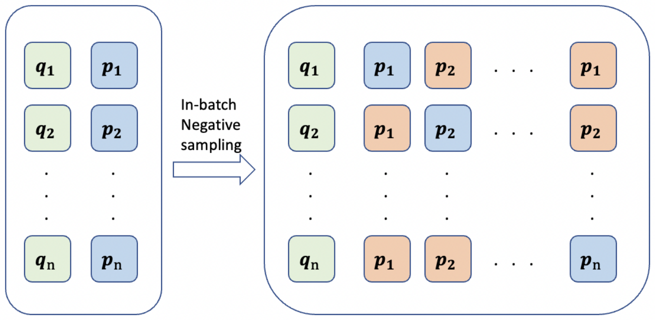

Figure 2: In-batch Negative Sampling

[2] shows that providing hard negatives instead of random negative samples during training could significantly improve the retrieval performance. [3] proposed Adaptive Batch Scheduling (ABS) to provide hard negatives. Its main goal is to maximize the sum of batch hardness scores which is defined as the similarity scores between questions and passages over the batch.

In this research, we apply the ABS algorithm to provide training instances for Condenser and use FAISS [4] to find the most relevant passage to the query. We evaluate the trained models against the well-known document retrieval benchmark dataset MS-MARCO [5] for dense passage retrieval. The result shows that our proposed approaches significantly improves the document retrieval performances. In addition, we also propose methods to further improve the performance of the ABS in terms of efficiency and accuracy. The technical details are well explained in section 3.2.

2 Related Work

Dense Retrieval Unlike sparse representation, which utilizes inverted index, dense representation encodes each paragraph and query into vectors. Typically, to measure the relevance of a paragraph and a query, the similarity score between the query vector and the paragraph vector would be used. Extensive researches have been done to explore pre-train language models for encoding the query and passage. [6] augmented pre-train language models with inverse cloze task. [7] showed that providing additional pre-training tasks will improve pre-train models. [8] introduced latent knowledge retrieval into pre-training.

There are also a large amount of researches on fine-tuning. [9] expanded negative samples by retrieving top-n relevant passages via BM25. [2] tried to filter out false negatives among proposed negative samples. Instead of only one encoding vector, [10] generated encoding vectors for each token to better represent each query and passage, and based on this research, [11] calculated the similarity scores between query and passage only with overlapping terms.

In-batch Sampling To feed more negative samples during training, in-batch negative sampling is widely used. [2] showed that even random in-batch negative sampling could significantly improve the system performance. [12] proposed AdaBoost, which weights existing training instances and identifies hard negative samples among those instances. Similar to AdaBoost, [3] proposed the Adaptive Batch Scheduling (ABS) to generate training batches. It finds the new hard training instances by combining different training instances.

In this paper, we implement the ABS algorithm and improve its performance. The original one is proposed with a greedy algorithm to generate training batches while we utilized beam search to find a better solution. We also effort in improving the time and space complexity.

Pre-training Until the very recent, pre-training for dense retrieval has been left unexplored. [13] explored domain matched pre-training and proposed a concurrent architecture DPR-PAQ, which pre-trained dense retrievers with a 65-million-size synthetic QA pair dataset generated by pre-trained Natural Question and Trivia QA pipelines. Instead of improving pre-training experiments, a recently proposed dense retrieval pre-training architecture Condenser [1] is designed, and based on the Condenser architecture, [14] augmented the Condenser MLM loss with a contrastive loss defined over the target search corpus.

In this paper, we tried to combine Adaptive Batch Scheduling with the Condenser architecture and to implement this model with Full-Ranking, in order to see whether introducing hard negatives into Condenser architecture would improve its performance on OpenQA tasks.

3 Approach

3.1 Baseline Pre-trained Language Models

To investigate how the pre-trained model Condenser improves dense retrieval, our group set up the baseline experiment with BERT [15] and RoBERTa [16] as pre-trained language models, which both have given state-of-the-art results on a wide variety of natural language processing tasks. RoBERTa is an optimized version of BERT with more robust design choices like dynamic masking and large mini-batches.

3.2 Adaptive Batch Scheduling

We applied Adaptive Batch Scheduling to generate training batches. Given training instances $T = \{d_1, d_2, ..., d_t\}$, where d_i is the question-passage pair (q_i, p_i), we divide it into m batches $T^B = \{B_1, B_2, ..., B_m\}$. For each batch B_k, we define hardness score as the sum of relevance scores between questions and their negative paragraphs over the batch.

$$h(B_k) = \sum_{d_i \in B_k, d_j \in B_k, i \neq j} s_{ij} \tag{1}$$

Where $s_{ij} = sim(q_i, p_j)$ represents the relevance score between a query q_i and a passage p_j. In our paper, we have explored two methods to calculate the relevance score. The first one is BM25, which is a bag-of-words retrieval function that ranks documents based on the frequency of the query terms in the documents. Given a query Q and a document D, BM25 can generate $score(D, Q)$ based on the term frequency. However, this method may suffer from term mismatch problem. We also apply dot product to the encoded vectors of questions and passages to calculate similarity score between them, where the encoding is given by being-trained models. Since the model learns and updates the encoder in every epoch, which will take a long time if no prior information is provided, to achieve faster training performance, we use BM25 in the first training epoch and then use encoded vector dot product for the remain epoch.

A higher $h(B_k)$ indicates that sampled negative passages p_j are very similar to the question q_i. Because the model can improve its performance by learning to select the positive passage among difficult negative samples, the goal is to schedule training instances to maximize the overall hardness scores:

$$T^B_{scheduled} = \underset{T^B}{\text{argmax}} \sum_{B_k \in T^B} h(B_k) \tag{2}$$

Where $T^B_{scheduled}$ is the scheduled training batches $\{B_1^{'}, ..., B_m^{'}\}$.

[3] proposed a greedy algorithm to generate the $T^B_{scheduled}$. Let denote U as the set of remain training instances, which is initialized as training instances T. Given batch size n, scheduler would first randomly select n elements from U to construct an temporary batch B. Then it would iteratively find d_r and d_a which meet the following conditions:

$$d_r \leftarrow \underset{d \in B}{\text{argmax}}\, h(B - d) \tag{3}$$

$$d_a \leftarrow \underset{d \in U, d \notin B}{\operatorname{argmax}} h(B - d_r + d) \tag{4}$$

Then it would replace d_r with d_a if this replacement produces a higher hardness score h(B). The process would stop when no more replacements can be made and the resulting batch B would be added to the set of $T^B_{scheduled}$. This resulting batch B would be removed from the set U and the scheduler would find another B based on the new U following same steps.

We also investigate another approach to generate the $T^B_{scheduled}$. The greedy algorithm may be stuck in a local optimum and results a sub-optimal solution, thus we implement beam search algorithm to generate $T^B_{scheduled}$, which has the potential to maximize the overall hardness score. In each replacement, instead of finding only one d_r and one d_a, we keep track of a list of candidates d_r and d_a. The d_r list is the top k instances d that maximize $h(B-d)$, where k is a beam search range index. While the d_a list is the top k instances d that maximize $h(B - d_{ri} + d)$ for each d_{ri} in d_r list.

To compensate the computational complexity induced by beam search, we use FAISS[4] to index and search relevant queries and passages faster. When identifying potential d_a, we only retrieve the top 100 passages for each question and top 100 queries for each passage in the current batch, instead of searching through the whole corpus. In each step, we have k^2 candidate replacements, and we replace the pair that maximize the $h(B - d_{ri} + d_{aj})$. Other steps are similar to the greedy algorithm.

[3] applied ABS before each epoch to generate all batches. To reduce scheduling time by utilizing parallelism, our implementation generates batches on-demand on a different process during training and updates all passage encodings after each epoch.

3.3 Condenser

As shown in Figure 1, Condenser[1] is a stack of transformer blocks. Given an input $x = [x_1, x_2, ..]$, it will be first prepended with a CLS, then embedded, and finally run through early backbone layers and late backbone layers. Each step can be written as

$$[h^0_{cls}; h^0] = Embed([CLS; x]) \tag{5}$$

$$[h^{early}_{cls}; h^{early}] = Encoder_{early}([h^0_{cls}; h^0]) \tag{6}$$

$$[h^{late}_{cls}; h^{late}] = Encoder_{late}([h^{early}_{cls}; h^{early}]) \tag{7}$$

The head layer takes the CLS representation from the late layers with a short circuit, which is the token representation from early layers. This late-early pair will run through head transformer blocks, which can be written as

$$[h^{cd}_{cls}; h^{cd}] = Head([h^{late}_{cls}; h^{early}]) \tag{8}$$

The head layer outputs are then used to perform masked language model (MLM; [15]) training.

$$\mathcal{L}^{mlm} = \sum_{i \in masked} CrossEntropy(W h^{cd}_i, x_i) \tag{9}$$

Condenser will generate dense CLS representations to aggregate information. However, the produced embedding space lacks semantics. To solve this, [14] further proposed the CoCondenser, which augmented the Condenser MLM loss with a contrastive loss. Unlike the previous architecture which pretrains model on artificial query-passage pairs in a supervised learning setting, CoCondenser pretrains the passage embedding space in an unsupervised learning setting, which is in a query-agnostic method. Suppose we have a random list of n documents $[d_1, d_2, ..., d_n]$, we can randomly extract from each document to construct pairs of spans, $[s_{11}, s_{12}, ..., s_{n1}, s_{n2}]$, which will be used for training the Condenser. Let the late CLS representation of span s_{ij} to be h_{ij}, we can define its contrastive loss of the target search corpus over the batch as

$$\mathcal{L}^{co}_{ij} = -\log \frac{\exp \langle h_{i1}, h_{i2} \rangle}{\sum_{k=1}^{n} \sum_{l=1}^{2} 1_{ij \neq kl} \exp \langle h_{ij}, h_{kl} \rangle} \tag{10}$$

We can define the loss for this batch as the average sum of MLM and contrastive loss,

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{2} [\mathcal{L}^{mlm}_{ij} + \mathcal{L}^{co}_{ij}] \tag{11}$$

4 Experiments

4.1 Data

We use MS MARCO Passage Ranking [5] dataset to train the model. Samples were collected from Microsoft Bing search logs. The dataset contains 1 million real Bing queries (each query is a question), and 8.8 million candidate documents (each document is a passage). Considering the limited GPU resource, we randomly select 100,000 queries and its positive passages as our training set. 1,500 queries are selected as the validation set. Each query of validation set has 100 candidate passages consisted of 1 positive passage and 99 passages randomly select from its top 1000 passages (MS MARCO dataset does not provide ranking for the top 1000 passages). We use full-ranking task to evaluate the performance of the models. The full-ranking test dataset consists of 4980 queris and 3,128,597 passages.

4.2 Evaluation method

Consistent with MS MARCO Passage Ranking leaderboard, Mean Reciprocal Rank (MRR) is used to evaluate the performance of the models. MRR is a measure to evaluate systems which return a ranked list of answers to queries, in which $rank$ is the position of highest ranked passage in the results and Q is the number of queries. In addition, MRR@100 means the number of candidate passages is 100.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{12}$$

4.3 Experimental details

We first use a smaller sample consisting 10,000 queries for experiment and hyperparameters tuning. BERT, RoBERTa and Condensor are used as query and passage encoders. Adaptive Batch Scheduling using BM25 similarity, dot-product similarity, dot-product similarity plus beam search are applied to generate training batches. For comparison, random and sequential in-batch negative sampling methods are also investigated. In ABS using dot-product similarity, we applied BM25 similarity in the first epoch to warm up. Contrastive loss and Learning-to-Rank (LTR) loss are used during training. For a query q, its positive passage $d+$, and a set of negative passage $[d_1^-, d_2^-, ..., d_l^-]$:

$$\mathcal{L}_{contrastive} = -\log\frac{\exp(q^T d^+)}{\exp(q^T d^+) + \sum_{i=1}^{l}\exp(q^T d_i^-)} \tag{13}$$

$$\mathcal{L}_{LTR} = -\log\frac{\exp(\sigma(W\{q, d^+\} + b))}{\exp(\sigma(W\{q, d^+\} + b)) + \sum_{i=1}^{l}\exp(\sigma(W\{q, d_i^-\} + b))} \tag{14}$$

The final hyperparameter we choose are learning rate as 5e-6, batch size as 8 and epoch as 15. The experiment result also shows that the LTR loss perform much worse than contrastive loss. While contrastive loss can achieve MRR@10 at 0.95 at dev set, LTR can only obtain 0.67. Considering LTR loss is also infeasible for full-ranking task as it requires the whole passages dataset as input, we decide not to move forward in investigating LTR in the following experiment.

For investigating the effect of dataset size on different models and ABS strategies, the whole training set with 100,000 queries is used to train the models. We use the same hyperparameters while setting the batch size to 5. Full-ranking task is used to evaluate the performance of different models. All 4980 queries and 3,128,597 passages are encoded by the fine-tuned models. FAISS is used for retrieving top 100 candidate passages for each query. We used MRR@100 as evaluation metric in the full-ranking task.

4.4 Results

In the experiment of 10,000 queries sample dataset, Condenser model outperforms all baseline models, achieving the highest MRR@100 in dev set as 0.592 and the highest full-ranking task as 0.253. Adaptive Batch Scheduling improves performance in all models. Figure 3 shows the performance of the Condenser model with different training methods. For Condenser model, the highest MRR@100 achieved at dev set of ABS and non-ABS training are 0.592 and 0.568, respectively. For RoBERTa, the highest MRR@100 for ABS and non-ABS training are 0.52 and 0.43. Different types of ABS have similar improvement in the performance, as shown in Table 2. The ABS using dot-product similarity has better performance than that using BM25 similarity. ABS using dot-product similarity with and without beam search have similarity performance.

	Dev Set Loss	Dev Set MRR@100	Full-Ranking MRR@100
BERT	4.640	0.422	0.137
RoBERTa	3.491	0.494	0.198
Condenser	**2.376**	**0.592**	**0.253**

Table 1: The best performance of different models using ABS in 10,000 dataset

	Dev Set MRR@100	Full-Ranking MRR@100
Non-ABS-Sequential	0.535	0.227
Non-ABS-Random	0.568	0.233
ABS-BM25	0.578	0.233
ABS-Encode	0.586	0.249
ABS-BeamSearch	0.592	0.253

Table 2: The performance of different ABS with Condenser Model in 10,000 dataset

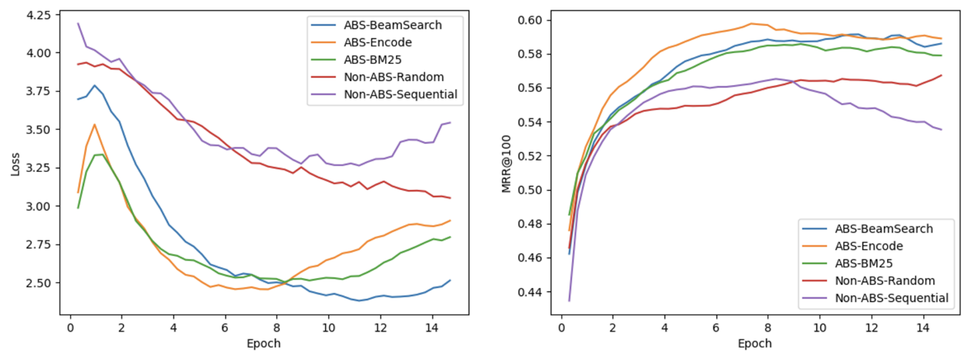

Figure 3: The performance of different ABS with Condenser Model in 10,000 dataset during training

In the experiment of 100,000 queries dataset. The Condenser model still outperforms baseline models as shown in Table 3. ABS exhibits significant improvement in model performance and regularization. While non-ABS training quickly overfits after 2 epoch, ABS can continue to learn from the training data. ABS using dot-product similarity achieves highest MRR@100 in both dev set and full-ranking task

	Dev Set Loss	Dev Set MRR@100	Full-Ranking MRR@100
BERT	2.514	0.503	0.186
RoBERTa	2.256	0.592	0.235
Condenser	**1.946**	**0.627**	**0.281**

Table 3: The best performance of different models using ABS in 100,000 dataset

	Dev Set MRR@100	Full-Ranking MRR@100
Non-ABS-Sequential	0.561	0.200
Non-ABS-Random	0.577	0.177
ABS-BM25	0.623	0.270
ABS-Encode	0.629	0.281
ABS-BeamSearch	0.611	0.250

Table 4: The performance of different ABS with Condenser Model in 100,000 dataset

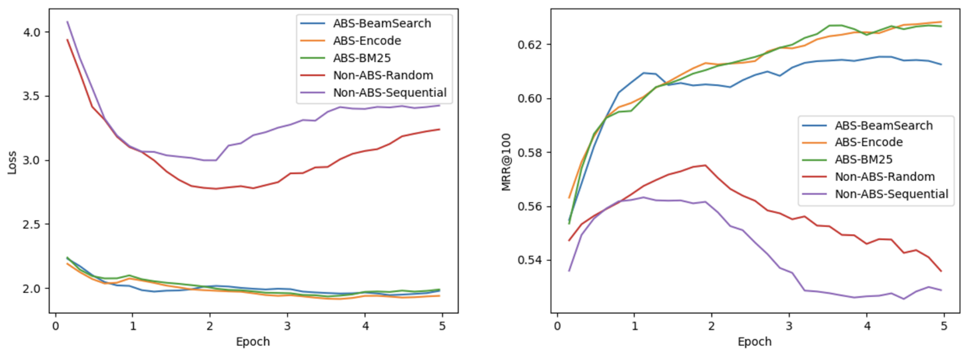

Figure 4: The performance of different ABS with Condenser Model in 100,000 dataset during training

5 Analysis

Condenser is built upon the idea that the typical pretrained LM cannot actively aggregate sentence information into CLS token in the middle transformer blocks. It divides transformer blocks into early, late and head layers and forces late layer to only passing information to head layer through the CLS token. The information of the early layer is passed to head layer by short circuit. In this way, all the transformer blocks can actively aggregate sentence information during training. The experiment results show that Condenser model outperforms BERT and RoBERTa with different training strategies and dataset sizes. It proves that the Condenser architecture can generate better sentence encoding than baseline models.

The result shows that Adaptive Batch Scheduling can improve model performance, especially in large dataset. As Figure 5 shown, the training loss and MRR@100 increase at the beginning of each epoch, at which time the ABS starts a new scheduling round, whereas the training loss of non-ABS training quickly drops near to zero and remains stable. This is because ABS finds new hardness negative samples for each query. In addition, the training loss of ABS training is always larger than that of the non-ABS training. It indicates that ABS generates batches always have harder negative samples. Consistently and adaptively feeding hard negative samples to models can help better distinguish positive passages from negative passages. Larger dataset can help ABS find harder negative samples for each query, so the effective of ABS is more obvious.

Besides improving the model retrieval accuracy, ABS can also prevent model from overfitting as shown in Figure 4. Since ABS restart a new scheduling round at each epoch, it each query will have different negative samples at different epochs, preventing model from overfitting to fixed batched. Therefore, ABS can be regarded as a regularization method.

We investigate three types ABS with different similarity metric and scheduling algorithm in our experiment. BM25 similarity is word-frequency based and commonly used in negative sample selections. It can generally find good negative samples. However, BM25 similarity is fixed at start and cannot adaptively change during training. Dot-product similarity is based on encoded query and passage vectors. It can adaptively change during traning process as model continuously learn to provide better encoding. However, it may suffer from cold start problem, i.e. it cannot find out real negative samples at the start of training. Hence, we use BM25 similarity for the first epoch of training instead. The experiment shows that ABS based on dot-product similarity has better model performance in the latter training process, which is in line with our theory.

At the aspect of ABS scheduling algorithm, beam search has better performance in small dataset while greedy algorithm exceed beam search in larger dataset. In theory, beam search should find better negative samples than greedy algorithm. However, due to the consideration of computation complexity, we limit the search range of beam search to top 100 similar queries and passages. For smaller dataset, this search range is sufficient to find good negative samples. However, this became a bottleneck for large dataset. The trade off between computation complexity and accuracy is the key to choose appropriate search range. It requires further experiment for finding suitble search range for different size of datasets.

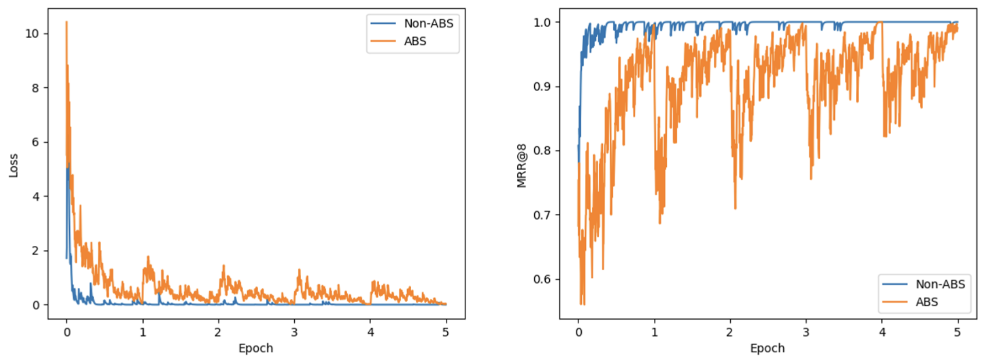

Figure 5: Training Loss of Condenser Model with and without ABS

6 Conclusion

Our experiments show that Condenser model can generate better query and passage encoding than BERT and RoBERTa. Its architecture allows model learn to encode more sentence information into the CLS token than other transformer encoders. Besides, Adaptive Batch Scheduling can greatly improve model performance and prevent overfitting, especially in large dataset. Adaptive Batch Scheduling using dot-product similarity can adaptively find better negative samples during training, thus has performance than BM25 similarity. As the aspect of scheduling algorithm, beam search outperforms greedy algorithm in small dataset, while greedy algorithm has better performance in large dataset. The possible cause is that we limit the search range of beam search to top 100 queries and passages in order to control computation complexity. The trade off between accuracy and computation complexity should be further investigated in the future.

Although our experiments show that combining Adaptive Batch Scheduling and Condenser has great potential in improving the OpenQA performance, we fail to achieve high score in MS MARCO Passage Ranking leaderboard. The current best submission achieved MRR@100 of 0.439 in full-ranking task. This is mainly due to lack of hardware resources. ABS is heavy in computation. [2] trained ABS over 500,000 training instances with batch size of 2048 in 30 epoch, while we only train our model using 100,000 queries with batch size of 8 in 5 epoch. If we could have sufficient hardware resources, our future work will be trying more variants of ABS, such as expanding its batch size and search range and training our model on larger dataset.

References

[1] Luyu Gao and Jamie Callan. Is your language model ready for dense representation fine-tuning? *CoRR*, abs/2104.08253, 2021.

[2] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *CoRR*, abs/2010.08191, 2020.

[3] Donghyun Choi, Myeongcheol Shin, Eunggyun Kim, and Dong Ryeol Shin. Adaptive batch scheduling for open-domain question answering. *IEEE Access*, 9:112097–112103, 2021.

[4] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017.

[5] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016.

[6] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *CoRR*, abs/1906.00300, 2019.

[7] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. *CoRR*, abs/2002.03932, 2020.

[8] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[9] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020.

[10] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. *CoRR*, abs/2004.12832, 2020.

[11] Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: revisit exact lexical match in information retrieval with contextualized inverted list. *CoRR*, abs/2104.07186, 2021.

[12] Yoav Freund and Robert E. Schapire. A short introduction to boosting. 1999.

[13] Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick S. H. Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen-tau Yih, Sonal Gupta, and Yashar Mehdad. Domain-matched pre-training tasks for dense retrieval. *CoRR*, abs/2107.13602, 2021.

[14] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval, 2021.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.