OpenMatch-v2: An All-in-one Multi-Modality PLM-based Information Retrieval Toolkit

Shi Yu
Tsinghua University
Beijing, China
yus21@mails.tsinghua.edu.cn

Zhenghao Liu
Northeastern University
Shenyang, China
liuzhenghao@mail.neu.edu.cn

Chenyan Xiong
Microsoft Research
Redmond, USA
chenyan.xiong@microsoft.com

Zhiyuan Liu
Tsinghua University
Beijing, China
liuzy@tsinghua.edu.cn

ABSTRACT

Pre-trained language models (PLMs) have emerged as the foundation of the most advanced Information Retrieval (IR) models. Powered by PLMs, the latest IR research has proposed novel models, new domain adaptation algorithms as well as enlarged datasets. In this paper, we present a Python-based IR toolkit OpenMatch-v2. As a full upgrade of OpenMatch proposed in 2021, OpenMatch-v2 incorporates the most recent advancements of PLM-based IR research, providing support for new, cross-modality models and enhanced domain adaptation techniques with a streamlined, optimized infrastructure. The code of OpenMatch is publicly available at https://github.com/OpenMatch/OpenMatch.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; *Evaluation of retrieval results.*

KEYWORDS

PLM-based IR, dense retrieval, re-ranking, open-source toolkit

ACM Reference Format:
Shi Yu, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2023. OpenMatch-v2: An All-in-one Multi-Modality PLM-based Information Retrieval Toolkit. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591813

1 INTRODUCTION

Recent advances in deep learning and neural networks have reshaped traditional bag-of-words Information Retrieval (IR) systems into intelligent agents which better understand users' intent and satisfy their information needs [21, 38, 40]. The advent of Pre-trained Language Models (PLMs) further brings a leap forward in data representation [13, 32, 39] and matching [21, 22, 24] for IR.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan, China.
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3591813

Table 1: Comparison of the functions of OpenMatch-v2 with the previous version. Double check "✓✓" denotes further improvements over the previous version.

Features		OpenMatch	OpenMatch-v2
IR Models	Dense Text Retrieval	✓	✓✓
	Re-ranking	✓	✓✓
	Cross-Modality Retrieval		✓
Domain Adaptation	Query Synthesis	✓	✓✓
	Unsupervised Pre-training		✓
	Re-ranker Score Distillation		✓
Infrastructure	Template-based Data Processing		✓
	Efficient Data Accessing		✓
	Sharded Search		✓

To facilitate research on IR, in early 2021, we proposed OpenMatch [17], an open-source toolkit for developing neural ranking models. OpenMatch was integrated with state-of-the-art neural ranking models and few-shot learning methods for domain adaptation at the time. With the help of OpenMatch, we achieved success in ranking on a set of IR benchmarks like MS MARCO [1] and TREC-COVID [30].

Since then, however, IR research has made significant progress. It has become almostly PLM-centric, learned and tested under larger, more diverse, or even cross-modality datasets [23, 25, 34]. New IR models [6, 22, 32] based on next-generation PLMs [26] are proposed, and algorithms for domain adaptation are further improved [11, 35]. The original design of OpenMatch lacked the flexibility and scalability to adapt to these new changes in IR.

To catch up with the latest trends of IR in 2023, we present OpenMatch-v2, a complete upgrade of OpenMatch. OpenMatch-v2 serves as a comprehensive platform for IR training, inference, and evaluation, and has been fully refactored to accommodate new-generation PLM-based IR models with a unified interface and improved domain adaptation techniques. The infrastructure of OpenMatch-v2 has also been enhanced to support efficient processing and search on data with growing size. Specifically, OpenMatch-v2 encapsulates popular IR models that are based on different PLMs, introducing T5 [26] as a new backbone for IR models in addition to traditional BERT-series models. It also supports CLIP-based text-to-image retrieval [25] and structured data retrieval, including code [10] and product search [27]. For domain adaptation, in addition to query synthesis [19, 33], we add supports for unsupervised pre-training [11] and re-ranker score distillation [9, 29, 35] to enhance model performance in few-shot scenarios. Concerning infrastructure, we propose template-based data processing as a general way to help researchers extract data from different datasets and an efficient way of data accessing and searching to help researchers handle large corpora with limited computational resources. We summarize the key updates of OpenMatch-v2 in Table 1.

The rest of the paper is organized as follows. Section 2 reviews related work on PLM-based IR and existing IR toolkits. Section 3 gives an overview of OpenMatch-v2. Section 4 presents the main functions and upgrades of OpenMatch-v2 with examples.

2 RELATED WORK

PLM-based IR. In recent years, Pre-trained Language Models (PLMs) have become widely applied in Information Retrieval (IR). Among the earliest PLM-based IR models is the BERT re-ranker [3, 21, 24], which concatenates the query and document with the `[SEP]` token and adds a linear head to obtain the ranking score. T5-based re-rankers [22] are the current state-of-the-art, employing a prompt-based training method for greater effectiveness. Another line of research proposes dense retrieval (DR) [13, 32, 39], which can be described as "dual encoders", where the query and document are encoded separately through their own PLM-based encoders. All documents are encoded as embedding vectors for later approximate nearest neighbor (ANN) search. Research on DR has explored topics including negative mining techniques [32, 39], pre-training approaches [6, 11, 18] for better domain adaptation, and ranking knowledge distillation from re-rankers [9, 29]. In OpenMatch-v2, we focus on PLM-based re-rankers and retrievers since they are the most effective and efficient models to date.

IR Toolkits. Numerous IR toolkits have been developed by researchers with various focuses. MatchZoo [8] implements a series of traditional text-matching models but is not compatible with PLM-based models. Capreolus[1] implements traditional lexical models and neural models including BERT-based re-rankers. Tevatron [7] is a DR-only IR package optimized for efficiency and flexibility. OpenMatch [17] provides support for dense retrieval and re-rankers, along with some data augmentation techniques. Sentence Transformers [28] has full support for PLM-based dense retrieval and re-ranking but lacks an efficient infrastructure. In OpenMatch-v2, we inherit OpenMatch's function but refactor the code to support new features like cross-modality search, DR pre-training, and infrastructure enhancement to satisfy new needs. We start from Tevatron's code base and gradually re-implement its core components to support new features.

3 TOOLKIT OVERVIEW

OpenMatch-v2 is a Python toolkit that can be easily installed via `pip`. It relies on PyTorch, HuggingFace (HF) Transformers [37], HuggingFace Datasets [15], and Faiss [12] to provide model, data, and ANN retrieval support. The toolkit offers a collection of out-of-the-box commands, as well as built-in classes and functions to serve as building blocks.

[1]https://github.com/capreolus-ir/capreolus

Table 2: Possible IR models supported by OpenMatch-v2.

	Dense Retrieval	Reranking
BERT-series	DPR [13], ANCE [39], ANCE-Tele [32]...	BERT Re-ranker [3, 21, 24]
T5-series	GTR [20]	monoT5 [22]
Others	CLIP [25]	–

Out-of-the-box Commands. OpenMatch-v2 provides out-of-the-box training and inference commands in the `driver` folder, which can be quickly accessed after installation. For example, to train a dense retrieval model, the command looks like:

```
python -m openmatch.driver.train_dr
    --output_dir str
    --model_name_or_path str
    --train_path str
    --learning_rate float
    ...
```

All training and inference commands are compatible with PyTorch DistributedDataParallel (DDP) mode, and can be executed in parallel using a simple `torchrun` command.

Classes and Functions. The out-of-the-box commands are implemented with built-in classes and functions of OpenMatch-v2, including dataset classes, model classes, trainer classes, and utility functions like file reading functions. If the user needs more specific functionality than what the out-of-the-box commands provide, they can easily create their own scripts using the classes and functions provided by OpenMatch-v2.

4 KEY FEATURES

In this section, we present the key features of OpenMatch-v2, including its IR models, domain adaptation techniques, and infrastructure.

4.1 IR Models

OpenMatch-v2 includes two sets of PLM-based IR models: dense retrieval models and re-ranking models, which are implemented in the `DRModel` class and `RRModel` class, respectively. Both model classes feature a series of configurable parameters that offer fine-grained control over the models. Unlike its predecessor, OpenMatch (v1), which only supported a limited number of pre-defined PLM-based models, OpenMatch-v2 is designed to be compatible with a wide range of popular PLM-based IR models, thanks to the flexible composition of each parameter.

Model Configurations. PLM-based IR models differ in their PLM backbones, ranking feature extraction methods, and other minor design choices. OpenMatch-v2 offers a variety of options for fitting common architectures and doing customization. Important options include the `type` parameter, which specifies the type of the backbone PLM in the HF Transformers class name, and the `feature` parameter, which specifies the feature used for ranking. Additional parameters, such as `pooling` and `normalize`, can be used to control the post-processing of the extracted feature. T5-specific parameters are also available, allowing users to use its encoder only, like GTR [20], or the entire encoder-decoder model. Those options can be configured by setting command-line arguments or supplying a configuration file, providing users with the flexibility to adapt to new models without having to modify the code.

Supported Models. OpenMatch-v2 supports multiple PLMs as the backbone of dense retrieval models and re-ranking models, including BERT-series (BERT [4], RoBERTa [16], DistilBERT [31], etc.) and T5-series (T5 [26], Flan-T5 [2], etc.). In addition, pre-trained models on structured data, such as CodeBERT [5] and CodeT5 [36] can also be used since they belong to the same architecture family as general models. For text-to-image search, we provide support for CLIP [25]. We encapsulate the various PLMs into the `DRModel` and `RRModel` class and expose uniform interfaces for critical operations such as data encoding and loss computing. Table 2 summarizes possible IR models users can build with OpenMatch-v2.

4.2 Domain Adaptation

PLM-based IR models can demonstrate superior results on datasets when large-scale training signals are available. However, they tend to show sub-optimal transfer ability in new domains where little or no training data is available [34, 41]. To mitigate this issue, we include popular domain adaptation techniques in OpenMatch-v2, providing researchers with tools to augment their trained models. We refer readers to the corresponding papers for more details.

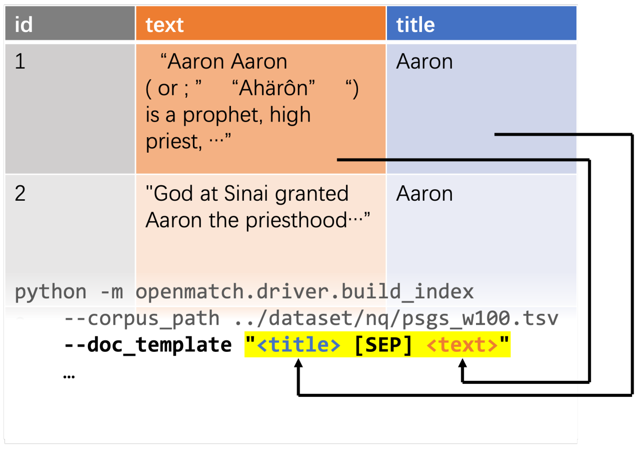
Query Synthesis. The query synthesis approach [19] generates relevant queries given documents on the target domain using a generation model trained on a large-scale source domain. Sun et al. [33] extends this approach by training the generation model to generate a query that is relevant to one given document but not to another in a contrastive fashion. OpenMatch-v2 is integrated with T5-based query generators, which support both types of query generation models, formulated as

$$q \leftarrow \text{T5}\left(d^+\right) \qquad \text{Ma et al. [19]},$$
$$q \leftarrow \text{T5}\left(d^+ \circ [\text{NEG}] \circ d^-\right) \qquad \text{Sun et al. [33]},$$

where d^+, d^- denote the relevant and irrelevant document, respectively, and [NEG] is a special token. OpenMatch-v2 provides users with commands for training query generation models and generating pseudo queries on a specific corpus.

Unsupervised Pre-training. Recent studies have shown that continuously pre-training the DR models before fine-tuning can improve downstream performance [6, 11, 18]. OpenMatch-v2 is integrated with the state-of-the-art contrastive-learning-based pre-training approach Cropping from Contriever [11] to provide better warm-up before fine-tuning on specific domains. Specifically, during pre-training, OpenMatch-v2 takes a piece of text, and samples two consecutive spans independently to form a positive pair. Negative texts are sampled from other positive pairs in the batch. In addition to simple cropping, the user can configure the training command to add more data augmentation strategies, including random deletion and replacement on document texts. We are planning to add more pre-training approaches in the future.

Re-ranker Score Distillation. As re-rankers are typically more powerful and robust than dense retrievers, and their scores are often used as training signals for retrievers [9, 29]. Inspired by this, OpenMatch-v2 introduces a re-ranker labeler that assigns the



id	text	title
1	“Aaron Aaron (or ; ” “Ahärôn” “) is a prophet, high priest, ⋯”	Aaron
2	"God at Sinai granted Aaron the priesthood⋯”	Aaron

```
python -m openmatch.driver.build_index
    --corpus_path ../dataset/nq/psgs_w100.tsv
    --doc_template "<title> [SEP] <text>"
    …
```

Figure 1: Template-based data processing on the NQ dataset during the dense retrieval index-building phase. Instead of asking users to convert the data into a required form, OpenMatch-v2 directly operates on the raw TSV data, fetching the title and text to assemble them into the final input to the model according to the given template.

(q, d) pairs with re-ranker scores, and a distillation trainer that fine-tunes dense retrieval models with the numeric, continuous training signals obtained from the re-ranker. The distillation trainer can train models in either pairwise (using MarginMSE loss) or listwise (using KLDivLoss) fashion. Note that this method can be used in combination with query synthesis, i.e. to first generate queries and then label the (q, d) pairs using a re-ranker [35].

4.3 Infrastructure

In this section, we present the infrastructure of OpenMatch-v2, which improves efficiency and eventually makes conducting large-scale experiments in resource-limited environments possible.

Template-based Data Processing. IR datasets come in various formats, and previous IR toolkits such as OpenMatch (v1) require users to convert their data into a compatible format for training and inference. To simplify this process and reduce manual effort, we introduce template-based data processing. We observe that TSV and JSONLINES formats are widely used for storing IR data, which can be considered as a collection of key-value objects. For instance, the NQ corpus data from DPR [13, 14] is a TSV file consisting of 20 million lines, with each line containing an identifier, document title, and document text specified by the TSV header (as shown in Figure 1). To prepare the raw data for inference, users only need to specify a template that defines how the input should be built with the data in each field. In the template, the key enclosed in angle brackets “<>” will be replaced with its corresponding value during processing. So the template defined in Figure 1 generates processed inputs that concatenate the document title and text, separated by the BERT sequence separator `[SEP]`. Other datasets like BEIR [34] in JSONLINES format can be processed in a similar template-based approach. Template-based data processing reduces the need for manual data-processing scripts, freeing up researchers to focus on developing their core algorithms.

Table 3: RAM and disk usages of different loading approaches of an 18GB dense retrieval pre-training dataset.

Mode	# Training Processes	RAM	On-disk Cache
Map	4	7GB	22GB
Stream	4	7GB	0
In-memory	4	79GB	0

Table 4: Search time and resources used in different sharding scenarios. Experiments are conducted on A100 GPUs.

# GPUs	# Big Shards	Retrieval Time	Peak RAM Usage
8	1	37s	198GB
4	2	42s	107GB
2	4	52s	66GB
1	8	72s	39GB

Efficient Data Accessing. OpenMatch-v2 integrates with HF Datasets [15] to provide efficient data accessing during training and inference. In contrast to OpenMatch (v1), which loads all the data into memory, OpenMatch-v2 keeps data memory-mapped by maintaining an on-disk cache in Arrow format[2]. This feature enables OpenMatch-v2 to randomly access large datasets with minimal memory overhead. However, the cache often requires more than 1x the disk space of the original dataset. As presented in Table 3, an 18GB pre-training dataset needs an 22GB additional disk space for the cache. To reduce disk usage, we leverage the *stream* feature of HF Datasets to allow users to directly iterate over the raw dataset from disk during training. We've wrapped the above-mentioned memory mapping and streaming features in the dataset classes.

```
class CustomTrainDataset(TrainDatasetBase):

    def get_process_fn(self, epoch:int, hashed_seed:int):
        """
        Write code here to return a processing function
     that takes a raw example as input and outputs the
     processed result
        """
        ...
# Define this class to expose interfaces for streaming
class StreamCustomTrainDataset(StreamTrainDatasetMixin,
     CustomTrainDataset):
    pass  # No additional code is required!

# Define this class to expose interfaces for random
     access (via memory-mapping)
class MappingCustomTrainDataset(MappingTrainDatasetMixin,
      CustomTrainDataset):
    pass
```

Listing 1: Defining a new dataset class and endowing it with streaming/random access feature.

To define a custom training dataset class, users can inherit `TrainDatasetBase`, the base class for all training dataset classes, and fill in the unimplemented `get_process_fn` function, as shown in Listing 1. To enable the new dataset to be streamed or random-accessed, users can mix in `MappingTrainDatasetMixin` or `StreamTrainDatasetMixin`. With the feature of Python mixin classes, the mixed-in classes will have the corresponding data-access abilities without the need for any additional lines of code.

[2]https://arrow.apache.org/

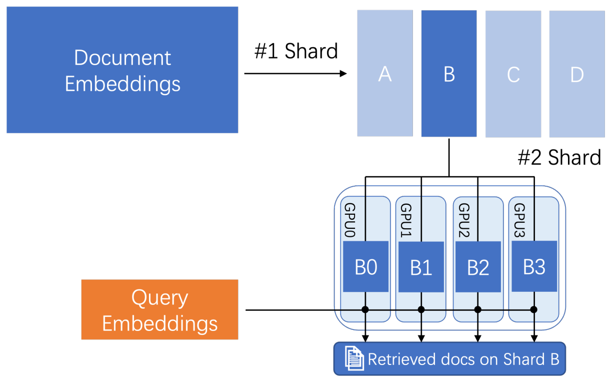


Figure 2: The process of the two-stage sharded search. OpenMatch-v2 iteratively loads and unloads the first-stage big shards (A-D), and searches on the second-stage small shards (B0-B3) in parallel on multiple GPUs. The figure presents the search on shard B.

Sharded Search. Dense retrieval models require all documents to be loaded into CPU/GPU memory for searching. However, as IR datasets continue to grow larger, it becomes increasingly difficult to fit all the data into a single device's memory. Take NQ as an example, its 20M documents can consume 20M $\times$ 768 dimensions $\times$ 4 bytes per dimension $\approx$ 60GB of memory. Typical GPU search is challenging for such a large index due to limited resources.

To address this issue, OpenMatch-v2 introduces a *shard-twice* approach to break down the search process into smaller, more manageable steps, as illustrated in Figure 2. The first round of sharding partitions the document embeddings into several large shards, which are then loaded one by one and further split into smaller shards to fit within the available devices. The smaller shards are then searched in parallel, and the results are aggregated in the final step. We present the trade-off between search time and resource usage of searching on the NQ dataset in Table 4.

5 CONCLUSION

In this paper, we introduce OpenMatch-v2, an all-in-one Python toolkit for experimenting with PLM-based IR. We make a couple of improvements over the previous version of OpenMatch, including novel models, and new domain adaptation techniques, and enhanced infrastructure. We demonstrate that OpenMatch-v2 is able to train and evaluate PLM-based IR models in a more efficient way. We hope that OpenMatch-v2 can benefit the IR community and boost future IR research.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106500) and the National Natural Science Foundation of China (NSFC No. 62236004). We thank Leyu Shan for integrating ANCE-Tele, Zhipeng Xu for implementing the contrastive query synthesis, Yida Lu for integrating parameter-efficient fine-tuning approaches, and Si Sun for proofreading the paper. Our implementation uses Tevatron as the starting point. We thank its authors for their contributions.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).

[2] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416* (2022).

[3] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of SIGIR*. 985–988.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*. 4171–4186.

[5] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of EMNLP*. 1536–1547.

[6] Luyu Gao and Jamie Callan. 2022. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. In *Proceedings of ACL*. 2843–2853.

[7] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. *arXiv preprint arXiv:2203.05765* (2022).

[8] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. Matchzoo: A learning, practicing, and developing system for neural text matching. In *Proceedings of SIGIR*. 1297–1300.

[9] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of SIGIR*. 113–122.

[10] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436* (2019).

[11] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *TMLR* (2022).

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[13] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of EMNLP*. 6769–6781.

[14] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the ACL* 7 (2019), 452–466.

[15] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of EMNLP: System Demonstrations*. 175–184.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[17] Zhenghao Liu, Kaitao Zhang, Chenyan Xiong, Zhiyuan Liu, and Maosong Sun. 2021. OpenMatch: An open source library for Neu-IR research. In *Proceedings of SIGIR*. 2531–2535.

[18] Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong Siamese encoder for dense text retrieval using a weak decoder. In *Proceedings of EMNLP*. 2780–2791.

[19] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2021. Zero-shot Neural Passage Retrieval via Domain-targeted Synthetic Question Generation. In *Proceedings of EACL*. 1075–1088.

[20] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Abrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large Dual Encoders Are Generalizable Retrievers. In *Proceedings of EMNLP*. 9844–9855.

[21] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[22] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of EMNLP*. 708–718.

[23] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *Proceedings of NAACL*. 2523–2544.

[24] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*. 8748–8763.

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* 21 (2020), 140:1–140:67.

[27] Chandan K Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. *arXiv preprint arXiv:2206.06588* (2022).

[28] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP*. 3982–3992.

[29] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of EMNLP*. 2825–2835.

[30] Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R Hersh. 2020. TREC-COVID: rationale and structure of an information retrieval shared task for COVID-19. *Journal of the American Medical Informatics Association* 27, 9 (2020), 1431–1436.

[31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[32] Sun Si, Xiong Chenyan, Yu Yue, Overwijk Arnold, Liu Zhiyuan, and Bao Jie. 2022. Reduce Catastrophic Forgetting of Dense Retrieval Training with Teleportation Negatives. In *Proceedings of EMNLP*. 6639–6654.

[33] Si Sun, Yingzhuo Qian, Zhenghao Liu, Chenyan Xiong, Kaitao Zhang, Jie Bao, Zhiyuan Liu, and Paul Bennett. 2021. Few-Shot Text Ranking with Meta Adapted Synthetic Weak Supervision. In *Proceedings of ACL*. 5030–5043.

[34] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *NeurIPS*.

[35] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. In *Proceedings of NAACL*. 2345–2360.

[36] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *Proceedings of EMNLP*. 8696–8708.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*. 38–45.

[38] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of SIGIR*. 55–64.

[39] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwikj. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*.

[40] Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. 2021. Few-Shot Conversational Dense Retrieval. In *Proceedings of SIGIR*. 829–838.

[41] Kaitao Zhang, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2020. Selective weak supervision for neural information retrieval. In *Proceedings of The Web Conference*. 474–485.