

DeBIR: A Python Package for Dense Bi-Encoder Information Retrieval

Vincent Nguyen^{1,2}, Sarvnaz Karimi², and Zhenchang Xing^{1,2}

¹ Australian National University, School of Computing ² Commonwealth Scientific and Industrial Research Organisation, Data61

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Information Retrieval (IR) is the task of retrieving documents given a query or information need. These documents are retrieved and ranked based on a relevance function or relevance model such as Best-Matching 25 ([Robertson et al., 1995](#)). Although deep learning has been successful in other computer science fields, such as computer vision ([Krizhevsky et al., 2012](#); [Szegedy et al., 2014](#)) and natural language processing ([Devlin et al., 2019](#); [Lee et al., 2019](#); [Yang Liu & Lapata, 2019](#)), success in information retrieval was limited in the literature due to comparisons against weak baselines ([Yang et al., 2019](#)). However, in 2019 ([lin-neural-recentation?](#)), deep learning in information retrieval could surpass less computationally intensive keyword-based statistical models in terms of retrieval effectiveness, sparking the field of dense retrieval. Dense retrieval is the task of retrieving documents given a query or information need using a dense vector representation of the query and documents. The dense vector representation is obtained by passing the query and documents through a neural network. The neural network is usually a pre-trained language model such as BERT ([Devlin et al., 2019](#)) or RoBERTa ([Yinhan Liu et al., 2019](#)). The dense query vector representation is then used to retrieve documents using a similarity function such as cosine similarity.

Unlike statistical learning, tuning deep learning retrieval methods is often costly and time-consuming. This cost makes it essential to automate much of the training, tuning and evaluation processes efficiently.

DeBIR is a library for facilitating dense retrieval research, primarily focusing on bi-encoder dense retrieval where query and documents dense vectors are generated separately ([Reimers & Gurevych, 2019](#)). It allows for expedited experimentation in dense retrieval research by reducing boilerplate code through an interchangeable pipeline API and code extensibility through the inheritance of general classes. It further abstracts standard training loops and hyperparameter tuning into easy-to-define configuration files. This library is aimed at helping practitioners, researchers and data scientists experimenting with bi-encoders by providing them with dense retrieval methods that are easy to use out of the box but also have additional extensibility for more nuanced research.

A brief summary of the pipeline stages ([Figure 2](#)) is:

1. Configuration based on TOML files; these are loaded in a class factory to create pipeline objects.
2. An executor object takes in a query builder object. The purpose of the query builder object is to define the mapping of the documents and which parts of the query to use for query execution.
3. The executor object asynchronously executes the queries.

41 4. Finally, an evaluator object uses the results to list metrics defined by a configuration file
42 against an oracle test set.
43 This pipeline is condensed into a single class that can be built from a configuration file.

44 Statement of Need

45 Dense retrieval has been popular in Information Retrieval for some time ([Guo et al., 2017](#); [Hui](#)
46 [et al., 2017](#); [Yin et al., 2015](#)). In the early 2000s, there had been considerable stagnation
47 in retrieval effectiveness as there needed to be stronger baselines ([Armstrong et al., 2009](#))
48 when proposing new methods. This stagnation repeated with the rise of deep learning, where
49 retrieval performance was again compared against weaker baselines and was not significantly
50 stronger than older non-deep learning statistical models, such as a well-tuned BM25 model
51 ([Yang et al., 2019](#)).

52 However, this was later recanted when transformer models could be used fine-tuned on Natural
53 Language Inference tasks or ms-marco as a cross-encoder (where a query and document pair
54 are encoded at ranking time) ([Lin, 2019](#)), significantly overtaking even the best BM25 models.

55 Today, there are generally two classes of dense retrieval models for IR: the cross-encoder, which
56 encodes queries and documents at query time and the bi-encoder, which can encode documents
57 at index time and queries at query time. The cross-encoder is generally more effective than the
58 bi-encoder model for retrieval. However, this increased effectiveness requires a more substantial
59 computation and can be a bottleneck in production systems. Therefore, a less expensive model
60 such as BM25 is typically used to retrieve smaller candidate lists (first-stage retrieval) for
61 second-stage retrieval re-ranking by a cross-encoder.

62 However, bi-encoders are more effective than BM25 and can complement BM25 for ranking
63 ([Nguyen et al., 2022](#)). Therefore, a gap in the literature in IR is to replace BM25 first-stage
64 retrieval with a bi-encoder or used as the only ranking system in the pipeline if query speed is
65 needed. However, current libraries don't address this use case as it requires integration with
66 the indexing and querying pipeline of the search engine.

67 DeBIR is a library that mainly facilitates bi-encoder research (where query and document can be
68 encoded independently) and provides base classes with flexible functionality through inheritance.
69 Although we provide cross-encoder re-rankers for feature completeness, the library's priority is
70 facilitating bi-encoder research. The strength of bi-encoders lies in the offline indexing of dense
71 vectors. These vectors can then be used for first-stage retrieval and potentially passed to a
72 second-stage retrieval system such as a cross-encoder. Bi-encoders can be used as the sole
73 retrieval system when there is a lack of training data ([Nguyen et al., 2022](#)) and, therefore, can
74 be more useful in areas such as biomedical IR, where training data is scarce. Cross-encoders,
75 however, require large amounts of training data for effectiveness.

76 The DeBIR library exposes an API for commonly used functions for training, hyper-parameter
77 tuning ([Figure 2](#)) and evaluation of transformer-based models. The pipeline can be broken
78 up into multiple stages: parsing, query building, query execution, serialization and evaluation
79 ([Figure 1](#)). Furthermore, we package our caching mechanism for the expensive encoding
80 operations to speed up the pipeline during repeated experimentation.

81 Although similar libraries exist, such as sentence-transformers ([Reimers & Gurevych, 2019](#)),
82 openNIR([MacAvaney, 2020](#)), they have less of a focus on the early stages of the dense retrieval
83 pipeline. This stage involves indexing the textual data from the corpora and indexing dense
84 vector representations, which is only helpful for bi-encoder type models over the traditional
85 cross-encoder. Other limitations include a lack of extendability that restrict the users' options
86 for training customization (we provide base classes that can be inherited) or that the library is
87 tailored to general-purpose machine learning rather than informational retrieval.

88 This library will help facilitate early-stage dense retrieval and rapid experimentation research

89 while being flexible enough for second-stage retrieval from either this library or other libraries.
90 Furthermore, we will continue to improve this tool.

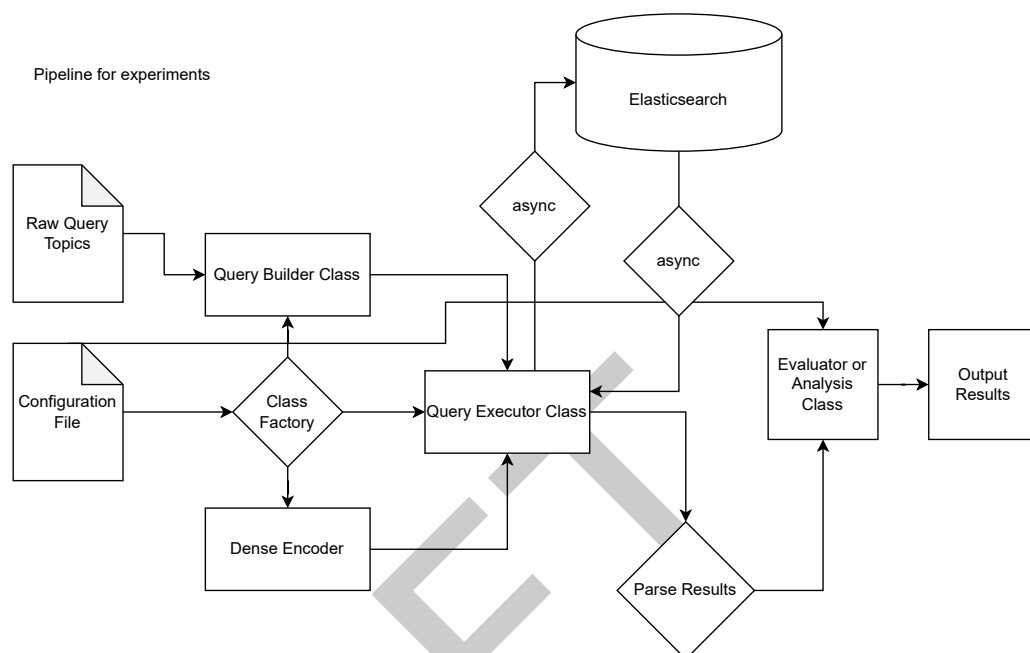


Figure 1: Standard flow of the DeBIR query/evaluation loop.

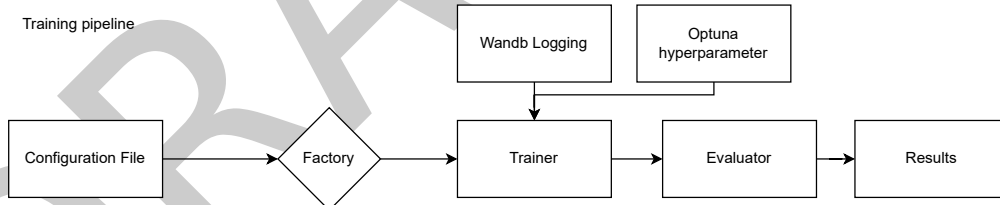


Figure 2: Standard flow of the DeBIR training loop.

Acknowledgments

92 The DeBIR library uses sentence-transformers, huggingface's transformers and datasets,
93 allRank, optuna, elasticsearch and trecTools python packages.

94 This search is supported by CSIRO Data61, an Australian Government agency through the
95 Precision Medicine FSP program and the Australian Research Training Program. We extend
96 thanks to Brian Jin (Data61) for providing a code review.

Examples

Pipeline

99 The pipeline is a single class that can be built from a configuration file. The configuration
100 file is a TOML file that defines the pipeline stages and their parameters. The pipeline is built
101 using a class factory that takes in the configuration file and creates the pipeline stages. The
102 pipeline stages are then executed in order.

```

from debeir.interfaces.pipeline import NIRPipeline
from debeir.interfaces.callbacks import (SerializationCallback,
                                         EvaluationCallback)

from debeir.evaluation import Evaluator

p = NIRPipeline.build_from_config(config_fp="./tests/config.toml",
                                 engine="elasticsearch",
                                 nir_config_fp="./tests/nir_config.toml")

# Optional callbacks to serialize to disk
serial_cb = SerializationCallback(p.config, p.nir_settings)

# Or evaluation
evaluator = Evaluator.build_from_config(p.config, metrics_config=p.metrics_config)
evaluate_cb = EvaluationCallback(evaluator,
                                config=p.config)

p.add_callback(serial_cb)
p.add_callback(evaluate_cb)

# Asynchronously execute queries
results = await p.run_pipeline()

# Post processing of results can go here

```

103 Training a model

```

import wandb

from debeir.training.hparam_tuning.trainer import SentenceTransformerTrainer
from debeir.training.hparam_tuning.config import HparamConfig
from sentence_transformers import evaluation

# Load a hyper-parameter configuration file
hparam_config = HparamConfig.from_json(
    "./configs/training/submission.json"
)

# Integration with wandb
wandb.wandb.init(project="My Project")

# Create a trainer object
trainer = SentenceTransformerTrainer(
    dataset=get_dataset(), # Specify some dataloading function here
    evaluator_fn=evaluation.BinaryClassificationEvaluator,
    hparams_config=hparam_config,
    use_wandb=True
)

# Forward parameters to underlying SentenceTransformer model
trainer.fit(
    save_best_model=True,
    checkpoint_save_steps=179
)

```

104 Hyperparameter tuning

```

105 from sentence_transformers import evaluation
106 from debeir.training.hparm_tuning.optuna_rank import (run_optuna_with_wandb,
                                                         print_optuna_stats)

107 from debeir.training.hparm_tuning.trainer import SentenceTransformerHparamTrainer
108 from debeir.training.hparm_tuning.config import HparamConfig

# Load a hyper-parameter configuration file with optuna parameters
hparam_config = HparamConfig.from_json(
    "./configs/hparam/trec2021_tuning.json"
)

trainer = SentenceTransformerHparamTrainer(
    dataset_loading_fn=data_loading_fn,
    evaluator_fn=evaluation.BinaryClassificationEvaluator,
    hparams_config=hparam_config,
)

# Run optuna with wandb integration
study = run_optuna_with_wandb(trainer, wandb_kwargs={
    "project": "my-hparam-tuning-project"
})

# Print optuna stats and best run
print_optuna_stats(study)

```

105 More information on the library is found on the github page, DeBelR . Any feedback and
 106 suggestions are welcome at issues .

107 References

- 108 Armstrong, T., Moffat, A., Webber, W., & Zobel, J. (2009). Improvements that don't add up:
 109 Ad-hoc retrieval results since 1998. *CIKM*, 601–610.
- 110 Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep
 111 bidirectional transformers for language understanding. *Proceedings of the Conference of*
 112 *the North American Chapter of the Association for Computational Linguistics: Human*
 113 *Language Technologies*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- 114 Guo, J., Fan, Y., Ai, Q., & Croft, B. (2017). A deep relevance matching model for ad-hoc
 115 retrieval. *Computing Research Repository, abs/1711.08611*, 55–64. [http://arxiv.org/abs/](http://arxiv.org/abs/1711.08611)
 116 [1711.08611](http://arxiv.org/abs/1711.08611)
- 117 Hui, K., Yates, A., Berberich, K., & Melo, G. de. (2017). A position-aware deep model for rele-
 118 vance matching in information retrieval. *Computing Research Repository, abs/1704.03940*.
 119 <http://arxiv.org/abs/1704.03940>
- 120 Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep
 121 convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger
 122 (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran
 123 Associates, Inc.
- 124 Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2019). BioBERT: a pre-
 125 trained biomedical language representation model for biomedical text mining. *Bioinformatics*,
 126 *36*(4), 1234–1240. <https://doi.org/10.1093/bioinformatics/btz682>

- 127 Lin, J. (2019). Neural hype, justified! A recantation. *ACM SIGIR Forum*, 53. [http:](http://sigir.org/wp-content/uploads/2019/december/p088.pdf)
128 [//sigir.org/wp-content/uploads/2019/december/p088.pdf](http://sigir.org/wp-content/uploads/2019/december/p088.pdf)
- 129 Liu, Yang, & Lapata, M. (2019). Text summarization with pretrained encoders. *Proceedings*
130 *of the 2019 Conference on Empirical Methods in Natural Language Processing and the*
131 *9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*,
132 3730–3740. <https://doi.org/10.18653/v1/D19-1387>
- 133 Liu, Yinhan, Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer,
134 L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach.
135 *Computing Research Repository*, abs/1907.11692. <http://arxiv.org/abs/1907.11692>
- 136 MacAvaney, S. (2020). OpenNIR: A complete neural ad-hoc ranking pipeline. *Proceedings*
137 *of the 13th International Conference on Web Search and Data Mining*, 845–848. [https:](https://doi.org/10.1145/3336191.3371864)
138 [//doi.org/10.1145/3336191.3371864](https://doi.org/10.1145/3336191.3371864)
- 139 Nguyen, V., Rybinski, M., Karimi, S., & Xing, Z. (2022). Search like an expert: Reducing
140 expertise disparity using a hybrid neural index for COVID-19 queries. *Journal of Biomedical*
141 *Informatics*, 127, 104005. [https://doi.org/https://doi.org/10.1016/j.jbi.2022.104005](https://doi.org/10.1016/j.jbi.2022.104005)
- 142 Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese
143 BERT-networks. *EMNLP*, 3982–3992. <https://doi.org/10.18653/v1/D19-1410>
- 144 Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1995, January).
145 Okapi at TREC-3. *TREC*. https://trec.nist.gov/pubs/trec3/t3/_proceedings.html
- 146 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V.,
147 & Rabinovich, A. (2014). Going deeper with convolutions. *IEEE Conference on Computer*
148 *Vision and Pattern Recognition*, 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
- 149 Yang, W., Lu, K., Yang, P., & Lin, J. (2019). Critically examining the “neural hype” weak
150 baselines and the additivity of effectiveness gains from neural ranking models. *Proceedings of*
151 *the 42nd International ACM SIGIR Conference on Research and Development in Information*
152 *Retrieval*, 1129–1132. <https://dl.acm.org/doi/10.1145/3331184.3331340>
- 153 Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2015). ABCNN: Attention-based convo-
154 lutional neural network for modeling sentence pairs. *Computing Research Repository*,
155 abs/1512.05193. <http://arxiv.org/abs/1512.05193>