# Using CHERCHE to Empower Newcomers into Neural Information Retrieval – Extended Abstract

Raphaël Sourty[1,2], Jose G. Moreno[1], Lynda Tamine[1] and Francois-Paul Servant[2]

[1]*University of Toulouse, IRIT, UMR 5505 CNRS, F-31000, Toulouse, France*
[2]*Renault, France*

**Abstract**
In this paper, we briefly[1] present a new open-source python module for building information retrieval pipelines with transformers namely CHERCHE. Our aim is to propose an easy to plug tool capable to execute, simple but strong, state-of-the-art information retrieval models and show their capabilities in small collections. To do so, we have integrated classical models based on lexical matching but also recent models based on semantic matching. CHERCHE is oriented to newcomers into that want to use transformer-based models in small collections without struggling with heavy tools. The code and documentation of CHERCHE is public available at https://github.com/raphaelsty/cherche

**Keywords**
Neural Information Retrieval, Python Library, Information Retrieval Pipelines

**INTRODUCTION** Most of the existing tools for neural IR focus on the training of new models, giving little attention to the use of existing models which makes their integration harder on out of the box Information Retrieval (IR) systems. Indeed, even if an existing pre-trained model for IR is available, its integration on a portable IR system or in a larger system (e.g. in question answering, summarization, entity linking, etc.) is not a straightforward task. Additionally, it is clear that an important number of IR users may not be interested in training their own models while they are still interested in using recent, and publicly available, advances in the field. Currently, more than 29000 public models are available on model hub of huggingface, with more than 8000 focused on the use of BERT and more than 300 finetuned for sentence similarity[1].

Our tool, called CHERCHE [1], was developed as an option to fill this gap and here we test its capabilities in small IR collections. We also aim to empower newcomers in neural IR to explore pipelines with pretrained models without extra effort. Our full architecture is depict in Figure 1. As intended, when using CHERCHE, it is only needed few lines to read, process, and evaluate a neural model. We expect that by reducing the load in the use of these models, more IR users will be motivated to integrate neural IR models into larger systems.

The main contributions of CHERCHE, and thus, of this abstract paper, are:

[1]Full details are given in [1].

*CIRCLE'22: Joint Conference of the Information Retrieval Communities in Europe, July 04–07, 2022, Samatan, Gers, France*

raphael.sourty@irit.fr (R. Sourty); Jose.Moreno@irit.fr (J. G. Moreno); tamine@irit.fr (L. Tamine); francois-paul.servant@renault.com (F. Servant)
0000-0002-8852-5797 (J. G. Moreno)

[1]We arguably suggest that all those models may be useful in neural IR.

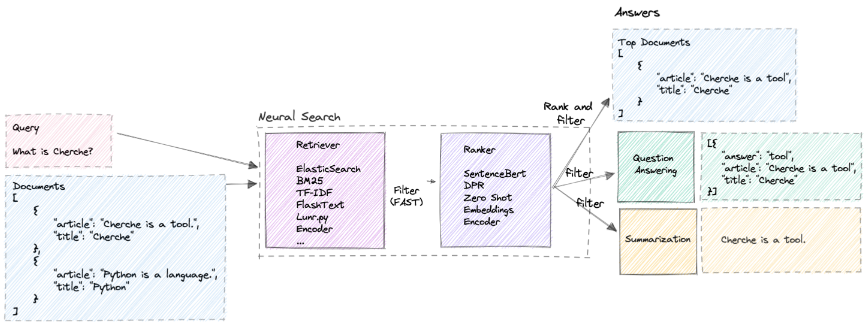

**Figure 1:** Global architecture of CHERCHE.

- A new tool that reduces the load when integrating transformer-based models
- A new option to perform the exploration of new (expert driven) pipelines into larger systems by using neural IR models

**RELATED TOOLS** Multiple python-based IR tools are publicly available nowadays. Some of the most popular, pyterrier [2] and pyserini, are backended by their Java versions Terrier and Anserini [3], because, at their time, they are based on Lucene[2]. Both of them are standard, and well established, alternatives when considering a python-based IR tool. Although they both are developed by strong communities, both tools are "heavy"[3] to install and use as extra steps are needed for their use (e.g., starting a java machine is required in both cases even when only python code is used). This contrasts with the NLP alternatives for similar downstream tasks.

**CHERCHE** Here, we summarize CHERCHE. An extended presentation of CHERCHE can be found in [1].

*Installation* We opted for a light installation from pip repository as many other python libraries. So the single line "*pip –install cherche*" allows the full installation of CHERCHE.

*Retriever* Retrievers allow the speed up of a neural search pipeline by filtering out documents that may be not relevant. We implemented most common retrievers based on lexical matching between the query and the documents. However, recent models also use semantic similarity combined with approximate search, based on faiss [4], to speed up the process. Here is the list of available retrievers using CHERCHE: TfIdf, BM25L and BM25Okapi, Elastic, Lunr, Flash, Encoder, DPR, and Fuzz. Currently, only the retriever *Elastic* is recommended with large corpora in CHERCHE. The other retrievers are adapted to small size corpora.

*Reranker* Rerankers will then be able to pull up documents based on semantic similarity. Rerankers are models that measure the semantic similarity between a document and a query.

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

[3]It is clear that current installation process is clean but we refer to heavy to the fact that a Java virtual machine is needed.

**Table 1**
Statistics of the vaswani dataset.

| Dataset | Type | Total | Stats |
|---|---|---|---|
| vaswani | Documents | 11429 | 41.92 tokens/doc |
| | Qrels | 93 | 22.39 docs/query |
| scifact | Documents | 5183 | 201.81 tokens/doc |
| | Qrels | 30 | 1.3 docs/query |

The reranker allows to reorder the documents retrieved by the retriever based on the semantic similarity between the query and the documents retrieved. Rerankers are compatible with all the retrievers in CHERCHE. To enhance the integration of new models, CHERCHE supports SentenceTransformers models available in the model hub of huggingfaces. This opens a multiple of models that could be used. Additionally, local models can be also specified as the system supports the standard class loader of huggingfaces models. Here is the list of available rerankers using CHERCHE: Encoder and DPR.

*Pipelines* CHERCHE overcharges the operators '+' (plus), '|' (union) and '&' (intersection) to build pipelines efficiently.

*Evaluation* Evaluate a pipeline using pairs of query and answers. The pipeline objects allow evaluation with three different metrics including F1, Precision, Recall, and P-Recall. However, we used external evaluation libraries in our experiments.

**EXPERIMENTS WITH CHERCHE** In order to highlight the characteristics of CHERCHE, we performed a set of experiments using two small IR datasets. A simple pipeline strategy was used, e.g. a pipeline composed by a retriever followed by a reranker.
*Datasets* We used the vaswani[4] and scifact[5] datasets in all experiments. In both cases, we opted for the python libraries that offer an easy access to the datasets, BEIR [5] and IR_datasets [6], to highlight the flexibility of opting for CHERCHE.

Details of both dataset are presented in Table 1. Note that both dataset are small as experiments were performed without Elastic[6].
*Metrics* Although CHERCHE proposes an internal evaluation module, we opted for standard IR evaluations metrics including MAP, NDCG@5, NDCG@10, and NDCG@20. The public available pytrec_eval[7] [7] implementation was used as evaluation tool.
*Models* We used a set of the most downloaded models on huggingfaces hub for the sentence encoders. The full list of models is presented in Table 2.
RESULTS

Table 3 presents the summary of our results on both datasets. Note that we included pyterrier

[4]http://ir.dcs.gla.ac.uk/resources/test_collections/npl/
[5]https://scifact.apps.allenai.org/
[6]Elastic is needed for larger datasets.
[7]https://github.com/cvangysel/pytrec_eval

**Table 2**
SentenceTransformers models used in our experiments. Models were selected based on their number of downloads only.

| Model | Huggingfaces name |
|---|---|
| Model A | all-mpnet-base-v2 |
| Model B | all-MiniLM-L6-v2 |
| Model C | paraphrase-albert-small-v2 |
| Model D | paraphrase-MiniLM-L6-v2 |
| Model E | paraphrase-mpnet-base-v2 |
| Model F | paraphrase-multilingual-MiniLM-L12-v2 |
| Model G | bert-base-nli-mean-tokens |
| Model H | LaBSE |

**Table 3**
Results on the two used datasets. AVG(A-H) correspond to the average performance of the models A to H. Best model correspond to the best results, the A model. '*' values were taken from https://github.com/terrier-org/pyterrier/blob/master/examples/notebooks/ltr.ipynb.

| | | vaswani | | scifact | |
|---|---|---|---|---|---|
| | | MAP | ndcg@10 | MAP | ndcg@10 |
| pyterrier | PL2 | 0.2060* | 0.4245 | 0.4087 | 0.4438 |
| | LambdaMART | 0.2043* | - | - | - |
| CHERCHE | retriever only | **0.2590** | 0.4316 | 0.5169 | 0.5619 |
| | AVG (A:H) | 0.1827 | 0.3456 | 0.4494 | 0.4943 |
| | BEST | 0.2508 | **0.4741** | **0.6047** | **0.6484** |

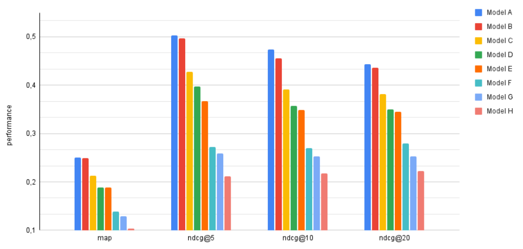

**Figure 2:** map, ndcg@5, ndcg@10, and ndcg@20 performances for models A, B, C, D, E, F, and H using vaswani dataset and a simple pipeline with CHERCHE.

as baseline, and if available, we reported the results from their official repository[8] or performed experiments to obtain its results. In both datasets, we used Lunr as retriever as it performs similarly than pyterrier in terms of ndcg@10 when using vaswani dataset[9]. As a main result, note that based on Table 3, the best reranker strongly outperforms the retriever, but it is not the case

[8]https://github.com/terrier-org/pyterrier
[9]Also note that as mentioned before, CHERCHE is clearly an option for small datasets but standards libraries, such as pyterrier, will be more adapted for large datasets.

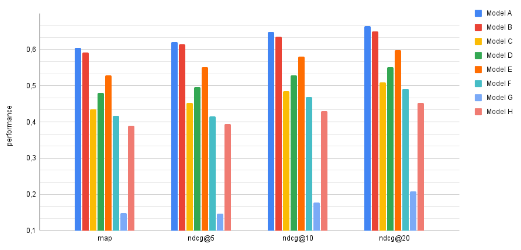

**Figure 3:** map, ndcg@5, ndcg@10, and ndcg@20 performances for models A, B, C, D, E, F, and H using scifact dataset and a simple pipeline with CHERCHE.

when averaging the performance of all models (AVG). This result highlight the importance of selecting an adapted transformer model, which can be easily performed when using CHERCHE.
*vaswani* Results of the performances for the eighth rerankers mentioned in Table 2 are presented in Figure 2. Note that the model A outperforms other alternatives and clearly outperforms the single retriever. Indeed, models A, B, and C outperform the retriever performances. Other models underperformed the retriever. This trend was observed for most of the used metrics.
*scifact* Results using scifact are presented in Figure 3 and follow the vaswani results, e.g., the retriever performance is outperformed by a clear margin for models A and B. However, model C did not manage to obtain a good performance, but model E does across multiple metrics. Finally, model G is clearly not an option for this dataset. This shows one of the feature of CHERCHE, the rapidly identification of candidate models to integrate a production pipeline.

**CONCLUSION** This paper briefly presents CHERCHE a library for neural pipelines definition. Our library was developed to be light and portable to new environments as a tool to quickly evaluate/integrate neural IR models into multiple text-related tasks including question answering. Although it can be used to develop new models, CHERCHE targets newcomers to the neural search that want to verify the pipelines based on transformers within their collections. Our results show that state of the art performance can be easily implements using CHERCHE.

## References

[1] R. Sourty, J. G. Moreno, L. Tamine, F.-P. Servant, Cherche: A new tool to rapidly implement pipelines in information retrieval, in: Proceedings of SIGIR 2022, 2022.

[2] C. Macdonald, N. Tonellotto, Declarative experimentation ininformation retrieval using pyterrier, in: Proceedings of ICTIR 2020, 2020.

[3] P. Yang, H. Fang, J. Lin, Anserini: Enabling the use of lucene for information retrieval research, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 2017, p. 1253–1256.

[4] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with GPUs, IEEE Transactions on Big Data 7 (2019) 535–547.

[5] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models, in: Thirty-fifth Conference

on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[6] S. MacAvaney, A. Yates, S. Feldman, D. Downey, A. Cohan, N. Goharian, Simplified data wrangling with ir_datasets, in: SIGIR, 2021.

[7] C. Van Gysel, M. de Rijke, Pytrec_eval: An extremely fast python interface to trec_eval, in: SIGIR, ACM, 2018.