

# Project 1: Text Retrieval

Information retrieval is the process of searching and returning relevant documents for a query from a collection. This project focuses on text retrieval task, and includes two subtasks:

## Task 1: Document Full Ranking

In this task, you are expected to rank documents based on their relevance to the question, where documents can be retrieved from the full document collection provided. It models a scenario where you are building an end-to-end retrieval system.

## Task 2: top-100 re-ranking

We provide you with an initial ranking of documents per question, and you are expected to re-rank the documents in terms of their relevance to the question. This is a very common real-world scenario since many end-to-end systems are implemented as retrieval, followed by top-k re-ranking. The re-ranking subtask allows you to focus on re-ranking only without needing to implement an end-to-end system.

## Dataset

We provide you a corpus of 1,471,405 documents with the following information:

Split	# of examples
Task 1 train	532,751
Task 2 train	10
Task 1 test	6980
Task 2 test	33

In the dataset folder you find the following files:

- `"corpus.jsonl"`: This file contains a collection of documents we want to retrieve from. Each document has a unique id (`"_id"`), and its content (`"text"`).
- `"queries.jsonl"`: This file contains a collection of queries that we want to retrieve relevant documents for. Each query has a unique id (`"_id"`), and its content (`"text"`).

- `task1_train.tsv`: This file is the “train” split of the data for Task 1 containing relevant documents for a set of queries and their relevance scores (always 1).
- `task2_train.tsv`: This file is the “train” split of the data for Task 2 containing relevant documents for a set of queries and their relevance scores (between 0 to 4).
- `task1_test.tsv`: This file is the “test” split of the data for Task 1 containing the queries you need to retrieve documents for.
- `task2_test.tsv`: This file is the “test” split of the data for Task 2 containing a set of queries and their retrieved candidates where you need to rank them based on their relevance.

## Evaluation

**Task 1:** For each query given in the Task 1 test set (`task1_test.tsv`), you have to retrieve 10 relevant documents from the given corpus.

We evaluate the performance of your retrieval system using **Recall@10** metric.

**Task 2:** For each query given in the Task 2 test set (`task2_test.tsv`), you have to re-rank the retrieved the documents in terms of their relevance to the query.

We evaluate the performance of your re-ranking system using **NDCG**<sup>1</sup> metric.

Final evaluation score is computed as a weighted average of the scores computed for these two tasks:

$$w_1 * recall@10 + w_2 * NDCG$$

where  $w_1 = 0.8$  and  $w_2 = 0.2$ .

**Computation time:**

## Requirements:

- You can use any supervised or unsupervised methods in the project. However, for the supervised methods you have to train the model yourself (using the given labeled data) and you are not allowed to use any trained models publicly available.

---

<sup>1</sup> Normalized Discounted Cumulative Gain (NDCG)

## Submission

Data distribution, running codes, and submission are all done using the Kaggle platform. Please create a Kaggle account with your **EPFL email**. Similar to colab, you can run notebooks (with or without GPU) on Kaggle for investigating data, as well as the final **submission** notebook. This notebook should write the submission file as one of the outputs. We will give more details about the final deliverables during the Thursday exercise session. In order to join the competition (with your **EPFL email Kaggle**), please use this [invitation link](#).

Note: Throughout the competition, please don't make your notebooks public, and keep them **private** throughout the competition!