Data Science Project: An Information Retrieval Engine

Manos Chatzakis (emmanouil.chatzakis@etu.u-paris.fr), Teaching Assistant
Themis Palpanas (themis@mi.parisdescartes.fr), Instructor

Phase 1 Deadline: March 2, 2026, 11:59 PM
Phase 2 Deadline: April 6, 2026, 11:59 PM

Introduction

Information retrieval is a core problem in data science and machine learning. Given a collection of documents and a user query, the goal is to rank the documents so that the most relevant ones appear first. This task appears in many real-world systems, such as **search engines**, **recommendation systems,** and **question answering**.

In this project, you will create your very own information retrieval pipeline, using classic retrieval methods as well as deep learning embeddings. You will implement different parts of the pipeline, and you will develop methods to evaluate the quality of your system.

You will work with a dataset that contains documents and queries in the form of text. Your task is to build document embeddings, retrieve relevant documents for each query, and evaluate the quality of your rankings using standard similarity and evaluation measures. The project is divided into two main phases. During the first phase, you will create the basic skeleton of a retrieval pipeline, while during the second phase, you will implement optimizations for it.

The project evaluation will be based on

1. your **report, the quality of your code, and your understanding of the concepts**
2. a **Kaggle competition**, where your systems will be evaluated automatically based on their retrieval quality. Join using https://www.kaggle.com/t/f383bc6c1f194226bb43d21ab3d65418.

Please read the whole report before starting your project work.

Dataset

The dataset is available on Kaggle.

Documents (documents.json)
Contains the documents available in the document collection (or corpus). Each document has a unique id, and its content is divided into title (str), text (str), and tags (list of str). It also contains a category, denoted by the origin of the document (e.g., whether it's originating from a website or forum dedicated to programmer questions, gaming questions, etc.).

Not all documents contain a title, text, and tags, and documents from different categories might have different contents.

Queries
You are also provided a set of queries for training and testing. The train queries will be used to evaluate your system yourselves and make design decisions, while the test set is a set of queries to be used for the Kaggle competition.

For the training queries, you are also provided a dictionary (gts.json) that contains, for each query ID, the list of the relevant document IDs.

Logistics

For this project, you will work in teams of **four** students (submit your team here). The project must be implemented using **Python notebooks.** Your final grade will be given on a scale of **0 to 10,** and it will be given by:

- **50%:** quality of your report, **written in English**. Your report should describe the overall process you followed to finish this project, and contain the answers to the questions you will find later in the text.
- **20%:** the quality of your code (explanations, cleanliness, docstrings)
- **30%:** Kaggle competition (leaderboard-based, see last part of the report)

Both phases have the same weight in the final grade.

About AI usage: You are free to use any information you might find online, including forums, manuals, documentation, tutorials, and LLMs. However, if you use LLMs to help you write code or prepare your report, you should clearly declare it in your report and say specifically how you used it.

Phase 1 Deadline: March 2, 2026, 11:59 PM
Phase 2 Deadline: April 6, 2026, 11:59 PM

Phase 1: Retrieval Basics

Load and prepare the data

1. The first step of the project is to understand the dataset you are working with. Using the JSON module, load the documents, (train) queries, and ground-truth relevance data. Examine the fields provided for each document and query, and briefly describe what information they contain. Report basic statistics such as the total number of documents, the number of queries, and the number of relevance judgments.

   In addition, compute and report other information that helps you understand the dataset. For example, you may look at document length statistics, query length statistics, or how many relevant documents exist per query, and how many document categories we have in total. The goal of this step is to gain familiarity with the data and identify characteristics that may influence retrieval performance later in the project. Explain in your report what conclusions you made from your data, what they look like, what they represent, etc.

2. For both the documents and the queries, create a new text field named **`content`** by merging the information from the available **title**, **text**, and **tags** fields. There is no single correct way to perform this merge. You may choose how to combine these fields, for example, by simple concatenation. Optionally, you may apply any text preprocessing/cleaning steps that you believe could improve retrieval performance. Describe in your report how you perform your merge.

Implement the retrieval process pipeline.

The retrieval process aims to find the top-k most relevant documents of a query. Your goal is to return, for each query, the top-k most similar document IDs. There are several methods and retrieval models to do this. You will implement 3 of them. Note that for all methods, you are not expected to implement them from scratch. You can use Python modules such as scikit-learn, sentence-transformers, etc.

Classic information retrieval models: TF-IDF and BM25.

Retrieval methods such as **TF-IDF** and **BM25+** can be used to rank documents. For this part of the project, you will implement the following:

1. TF-IDF Retrieval
   - In a markdown cell and in your report, explain how TF-IDF works.
   - Represent your documents and queries using a TF-IDF vectorizer.
   - Compute cosine similarity between the TF-IDF vector of each query and each document.
   - Return the top-k most similar documents for each query (`topk_indices_tfidf` and `topk_scores_tfidf`).
2. BM25+ Retrieval
   - In a markdown cell and in your report, explain how BM25+ works.
   - Use the BM25+ algorithm on the documents.
   - Retrieve the top-k documents for each query (`topk_indices_bm25` and `topk_scores_bm25`).

Representation Learning: Text Embeddings.

In modern data science, one of the most common ways to perform information retrieval is by representation learning. This means converting the text of documents and queries into high-dimensional vectors, also called embeddings.

The key idea is that these embeddings capture the semantic meaning of text. Two texts that are similar in meaning will have vectors that are close to each other in the high-dimensional space. This allows us to retrieve documents by computing similarity measures (like cosine similarity) between the query vector and all document vectors. The top-k most similar documents are returned as the search results:

1. In a markdown cell of your notebook and in your report, answer the following: **What is the purpose of high-dimensional embeddings? How can they help in information retrieval compared to simpler models?**

   Hint: Think about how embeddings capture meaning, synonyms, and relationships between words. You can look up examples of SentenceTransformers or other embedding models to get ideas.

2. Using **SentenceTransformers**, generate embeddings for your **documents** and **queries**. After generating embeddings:
   - Examine the structure of the embeddings
   - Print the **shape** of your vectors (it should be `[#documents, embedding_dim]` for documents and `[#queries, embedding_dim]` for queries)
3. To better understand your data, visualize the embeddings in **2D** using a dimensionality reduction method:
   - **t-SNE** or **UMAP** are recommended: Plot the 2D projection and look for patterns
   - Answer: *Do you notice any clustering or separation? Why might this happen? Answer in a markdown cell and in your report.*

   Hint: If it is not clear to you, try to color the embeddings of the same category with a different color.

4. Now that you understand the embeddings, implement the **retrieval process**:
   1. For each query vector, compute the **cosine similarity** with all document vectors.
   2. Identify the **top-k most similar documents**.
   3. Return two arrays:
      - `topk_indices` → shape `[#queries, k]`, containing the indices of the most similar documents for each query
      - `topk_scores` → shape `[#queries, k]`, containing the corresponding similarity scores (e.g., the cosine similarities)

   Note**:** The value of k is up to you. You will need to experiment to see which k gives the best retrieval performance. You may use Google Colab if you need GPU access.

Create a retrieval evaluation suite.

As we discussed, the output of your retrieval engine should be `[Topk_indices, Topk_scores]`

To evaluate the performance of your retrieval engine, you will use standard evaluation measures. Each measure takes values in the range **[0.0, 1.0]**, where values closer to **1.0** indicate better retrieval performance.

Recall

Recall measures how many of the relevant documents were successfully retrieved. It answers the question: Did the system find the documents that matter? A high recall means that most relevant documents appear somewhere in the retrieved results.

Recall = # of correctly retrieved documents / # of total relevant documents of the query

Precision

Precision measures how many of the retrieved documents are actually relevant. It answers the question: Are the top results useful? High precision means that a large fraction of the retrieved documents are relevant.

Precision = # of correctly retrieved documents / # of total retrieved documents

Mean Reciprocal Rank (MRR)

MRR measures how early the first relevant document appears in the ranked list. For each query, it looks at the rank of the first relevant document and takes the reciprocal of that rank. The final score is the average over all queries. A high MRR means that relevant documents tend to appear very early in the ranking.

MRR = 1 / rank of the first correctly retrieved document

Compare the results of retrieval models.

1. Retrieve the top-k documents of each one of the queries using the embeddings, TF-IDF, and BM25+.
2. Print the results on all quality metrics. You should report, for example, the average recall, precision, etc., for each one of your retrieval methods.
3. Comment on the results. Why does a model have better results than others? At which metrics does each model seem to do well? What are the reasons for that? What are the tradeoffs between using each one of the models? How does the value of k change the results? How is the latency (speed) of retrieval different under different models? Answer in the corresponding markdown cells and in your report.

Kaggle competition: Phase 1.

Once you have compared the methods and decided which one gives you the best retrieval quality, you must submit the output of your model to Kaggle.

For information on how to submit to Kaggle and other useful stuff, please refer to the end of this report (Section Kaggle).

Phase 1 Deadline: March 2, 2026, 11:59 PM

Phase 2: Classification and Reranking

Document and Query Classification.

Remember that your data (documents and train queries) come with a category that defines the origin of the data. Your goal is to create a classifier that, given some input (document of query), it predicts its category.

1. What model could you use as a document classifier? How does it work? Write your answer in a markdown cell and in your report.

2. What data could you use to train your classifier? Will you use both documents and queries? Explain your decisions.
3. What input features could you use for your classifier?
4. Train your classifier and evaluate its classification quality.
    a. What is the accuracy and classification report of your classifier? How is the accuracy computed?
    b. How is it different for each document category?
    c. Does it work equally well on documents and queries?

   Answer in markdown cells and in your report.

Apply the classifier to your retrieval system.

At this point, you have available your classifier, which is a tool to predict the category of a given document or query. Since you have this tool at your fingertips, how could you use it to improve your retrieval system? Answer the following with code and text in your document.

1. Update your retrieval pipeline. Now, instead of only the top-k documents and their scores, your system should return the predicted category of the queries, using the classifier.
2. Update your evaluation metrics: Now, instead of the average recall, precision, and mrr, you should also report the average prediction accuracy of your model in predicting the category of the query.
3. In a markdown cell and your report, explain how the classifier could have been used to optimize retrieval quality.
    a. Hint: Remember that your system will be evaluated using testing queries, for which you only have their text.
    b. Example: You could predict the classification probability for each category that a query might belong to. Then, you could somehow “rescore” or “filter out” the documents to be retrieved based on whether they belong to the same category as your query.
4. Evaluate the performance of your system that uses the classifier to improve the results. You should implement the addition of a classifier to the pipeline and evaluate it with all the models (TF-IDF, BM25+, and Embeddings). How could you explain your conclusions? Explain in a cell and your report why the results improve (if they do) or why they do not (if they don’t).
    a. Hint: This is an open-ended question, and there is no correct way to solve it. We do not necessarily expect you to have better results, but you should show us that you understand why the results look like they do.

Kaggle competition: Phase 2.

The final part of the project involves submitting the results of your system to your Kaggle competition.

You should use all the information you have learned at this point (retrieval, classification, etc.) to optimize your retrieval routine to obtain better results.

This part is open-ended: You can do whatever you think makes sense to improve the retrieval quality. For example, you may use your system from the previous question. You can create cooperative pipelines using both BM25+, TFIDF, and embeddings. You can use different SentenceTransformer models to embed your documents. You can filter our resulting documents using the classifier to perform reranking. It is totally up to you to improve the quality of your method.

Kaggle Information and Submission (Phase 1 & 2)

You have been provided an additional query set (test queries), for which you do not know the relevant documents or their category.

The goal of the competition is to evaluate your retrieval systems on this query workload and climb the leaderboard in Kaggle.

For phase 1, you are expected to submit your outputs with code written only in phase 1, while for phase 2, you are expected to submit the output of the retrieval process you optimized. The code that generates the score of your submission should be uploaded to Kaggle, and your results must be reproducible.

Please use the following link to join the Kaggle competition. All the information about how to submit results, how the scoring works, and how you will be graded is there.

Please read the information on the competition page carefully.
Kaggle Competition Invite Link

Phase 2 Deadline: April 6, 2026, 11:59 PM