Semantic Similarity Search and Ranking of Research Papers Using arXiv Abstract Data

Ravindra Kumar Saini, Keshav Mishra
Missouri State University
{rs626s, km462}@missouristate.edu

Dataset Requirement

This project uses the official arXiv metadata snapshot. Download the dataset from the following link:

```
https://www.kaggle.com/datasets/Cornell-University/arxiv/data
```

After downloading, place the file in the project directory (same level as `main.py`).

```
Filename:  arxiv-metadata-oai-snapshot.json
```

Environment Setup (Using Conda)

To ensure reproducibility and consistent package versions, this project uses a Conda environment defined in the `environment.yml` file. This file contains all dependencies including NumPy, PyTorch, HuggingFace libraries, FAISS, and KaggleHub.

1. Create the Environment

Open a terminal (or Command Prompt on Windows), navigate to the project directory, and run:

```
conda env create -f environment.yml
```

This will create an environment named `semantic-search-env`.

2. Activate the Environment

Once created, activate it using:

```
conda activate semantic-search-env
```

3. Verify Installation (Optional)

To confirm that all required packages were installed correctly:

```
python -c "import torch, faiss, sentence_transformers, kagglehub; print('Environment OK!')"
```

If this prints `Environment OK!`, you are ready to run the system.

How to Run the System

Run the following commands from the `code/` directory.

1. Basic Execution (Default)

The default command loads the **local JSON dataset**, evaluates all models, and starts the interactive search:

```
python main.py
```

This performs:

- Loads the entire local JSON dataset (no category restrictions).
- Builds TF–IDF, BM25, and BERT indices.
- Runs full evaluation (Precision@K).
- Starts interactive semantic search using BERT.

2. Using the arXiv API Instead of Local JSON

Add the `--use-arxiv-api` flag:

```
python main.py --use-arxiv-api
```

This downloads papers live from the arXiv API based on configured categories.

3. Limit the Number of Papers (Optional)

To load only a subset (useful for speed/testing):

```
python main.py --max-papers 50000
```

Works for both JSON and API modes.

4. Choose a Model for Interactive Search

Supported models: `tfidf`, `bm25`, `bert` (default: `bert`)

```
python main.py --model tfidf
python main.py --model bm25
python main.py --model bert
```

5. Skip Evaluation (Interactive Search Only)

Use this option to skip Precision@K evaluation:

```
python main.py --no-eval
```

6. Combine Options (Examples)

Use JSON + limit papers + run only BM25 search

```
python main.py --model bm25 --max-papers 20000 --no-eval
```

Use arXiv API + BERT + full evaluation

```
python main.py --use-arxiv-api --model bert
```

Load full JSON + TF–IDF search only

```
python main.py --model tfidf --no-eval
```

Expected Output

- Summary of dataset loading (total papers, duplicates removed).
- Indices for TF–IDF, BM25, and BERT.
- Evaluation plots and metrics saved in the `results/` directory.
- Interactive search prompt:

```
Enter an abstract or description:
```

Notes

- The full JSON dataset is large (several GB); loading all papers may take several minutes.
- Using `--max-papers` is recommended during debugging.
- All randomness is controlled using a fixed seed for reproducible results.