Maria Castro, Seyla Wickramasinghe, Salma Yousry

Deliverable 2

Problem Statement

We aim to build an AI-powered research article recommender using the ArXiv dataset. Given a paper's title, abstract, or keywords, the model recommends related papers using similarity-based methods.

Data Preprocessing

The dataset we are working with is the arXiv Metadata OAI Snapshot from Kaggle, which contains 1.7 million articles, with relevant features such as article titles, authors, categories, abstracts, and full-text PDFs. For the preprocessing, we worked with a subset of 5000 samples of the dataset. To clean the data, all text was converted to lowercase, special characters, numbers, punctuation, and stopwords were removed, and single-character words were excluded to reduce noise. Rows with missing abstracts and duplicate abstracts were removed to avoid bias in similarity calculations. These modifications of the data allowed for clean versions of the titles and abstracts, which were saved to a Parquet file for efficient storage and processing. This preprocessing ensures that the data is consistent, normalized, and ready for similarity-based retrieval.

Machine learning model

We are using a KNN-based retrieval model in an unsupervised context. Instead of predicting a numeric value, the model identifies the 5 (k=5) most similar articles based on cosine similarity. Libraries used include scikit-learn, pandas, and joblib. We also used Parquet files. The architecture of the model is straightforward:

1 - The raw text from each paper (including title and abstract) is preprocessed and cleaned.

2 - A TF-IDF vectorizer converts the cleaned text into numeric vectors, creating embeddings that represent the content of each paper.

3 - The KNN model uses the embeddings to find the nearest neighbors in the TF-IDF vector space so that it can recommend the most similar papers to a given query.

There is no train/test split because this model does not involve supervised learning. The KNN model is used to identify the nearest neighbors based on embedding similarity. No explicit regularization techniques were used in this project. Regularization techniques are used in supervised models to prevent overfitting when training on labeled data. Again, since this is an unsupervised KNN search on precomputed embeddings, this training process is not optimized. This is appropriate for this project because the purpose of the project is to retrieve similar documents based on their embeddings. Also, since the TF-IDF + KNN approach always finds the closest papers, overfitting is not really an issue, and underfitting is unlikely. The main challenges faced when implementing the model were preprocessing (particularly removing the noise from abstracts), finding the idea of using TF-IDF vectors and cosine similarity for the model, and evaluating the relevance of the results obtained from the model, since one of the members had to manually go through each sample result, look at the articles and decide if it was relevant (1) or not relevant (0) for the Precision@k.

Preliminary Results

We evaluate our model using Precision@5, which measures how many of the top 5 recommended papers are actually relevant to each query.

So far, the TF-IDF and KNN setup gives decent early results. Precision@5 scores range from 0 to 1, averaging 0.65 across all queries. Topics like "Dark matter and Superconductivity" scored a perfect 1.0, showing that the top 5 suggestions were all on point. Others like "Transformer models" or "Symmetry and Numerical Solutions" were lower, landing closer to 0 - 0.4. That drop likely comes from overlapping terminology or broader topic scope, which makes it harder for TF-IDF to capture the real meaning.

The cosine similarity distribution (Figures 1 and 2) shows that most article pairs fall between 0.25 and 0.45 similarity, which means the model is finding moderately close connections. A few higher-scoring outliers (up to 0.7) appear for tightly focused topics like "cancer and free radicals", where vocabulary overlap is stronger.

The cosine-vs-precision scatter plot (Figure 3) shows a positive trend: queries with higher average cosine values tend to have higher Precision@5. So, cosine similarity is doing what we expect, higher similarity usually means more relevant recommendations. This confirms that cosine similarity is a good signal for relevance in our project.

It's also worth noting that at this point of the project we only worked with a very small subset of the dataset (about 5,000 papers out of over 1 million). Even with this limited data, the model still manages to produce meaningful recommendations, which is a good sign for scalability. Using the full dataset would definitely boost both cosine similarity and precision scores.

Overall, the model performs well given the small data slice and simple approach. The current results show that the project is feasible and that our content-based recommender can already surface contextually relevant research papers.

Figure 1: Cosine distribution boxplot over all recommendations

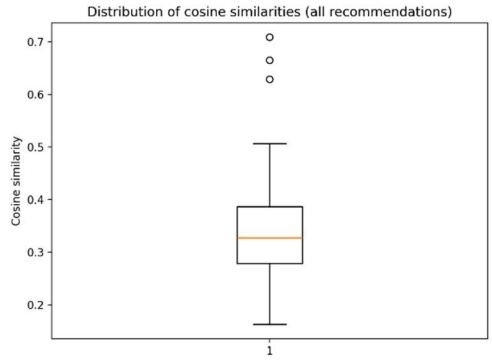

Figure 2: Cosine similarities histogram over all recommendations

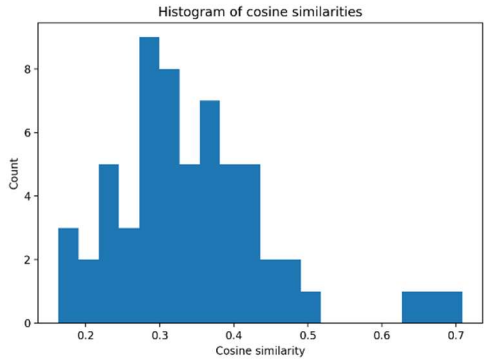

Figure 3: Cosine vs Precision@5 scatter

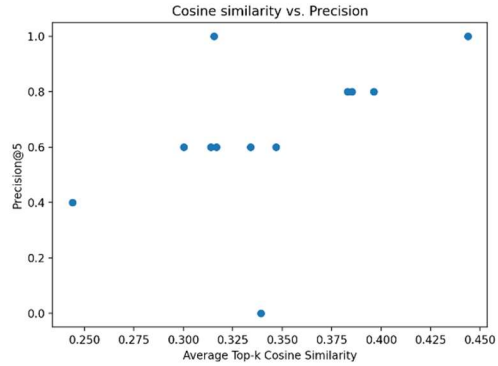

Figure 4: Precision@5 per query histogram

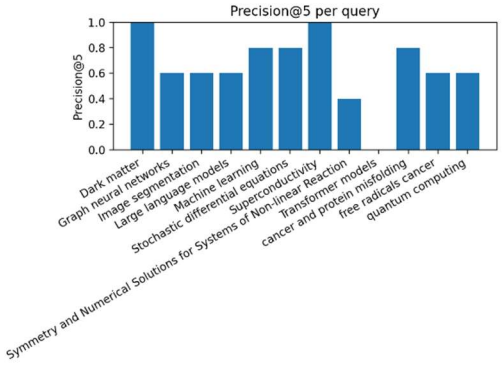

Figure 5: Results summary table

Query	Average cosine	Max cosine	Precision@5
Dark matter	0.44382	0.5068	1
Machine learning	0.39626	0.6653	0.8
Stochastic differential equations	0.38516	0.4128	0.8
cancer and protein misfolding	0.38302	0.629	0.8
Image segmentation	0.34694	0.4433	0.6
Transformer models	0.33926	0.3823	0
free radicals cancer	0.3339	0.7088	0.6
Large language models	0.31674	0.4229	0.6
Superconductivity	0.31556	0.363	1
Graph neural networks	0.31384	0.3584	0.6
quantum computing	0.3002	0.415	0.6
Symmetry and Numerical Solutions for Systems of Non-linear Reaction	0.24392	0.3122	0.4

Next steps

Pros: Straightforward architecture and good baseline performance with little tuning. Cons: Limited scalability due to brute-force similarity search and no personalization (if 2 users use the same query, they get the same recommendation).

The next steps could be implementing FAISS for efficient approximate nearest neighbor (ANN) retrieval and adding category-based filtering to improve relevance.