AdvisorMatch

A Prototype System for Streamlining Thesis Advisor Discovery

Amit Kumar - 237007553
Archit Panwar - 533000200
Harshith Reddy - 537000715
Sanjit Sahu - 737002703

Motivation

High Stakes, Outdated Process

Critical career decision relying on obsolete, manual navigation of static directories.

The Efficiency Bottleneck

Fragmented workflow; hours wasted cross-referencing university profiles with external databases.

Gap Exists

Semantic search exists for papers, not advisors; directories limited to rigid keyword matching.

Need for Advisor-Centric Search

Current tools find papers, not mentors; need faculty ranking by aggregate expertise.

Current Process

- Visit faculty directory
- Check faculty interests
- Search papers
- Filter by relevance
- Manually rank professors

Existing Systems

Academic Search Engines

e.g., Semantic Scholar

Limitation: Paper-Centric.

Great for documents, bad for people. No "advisor-view." Requires manual aggregation to check active niches.

Generative AI

e.g., Gemini / ChatGPT

Limitation: Reliability & Cost.

Prone to hallucinations. High latency. Expensive. Lacks real-time verification.

Metric-Based Rankings

e.g., CSRankings

Limitation: Volume over Relevance.

Counts publications, ignores topic semantics. Answers "Who publishes most?" not "Who fits my research?"

Our Proposal: Semantic Advisor Search with Relevance Ranking

Intuitive Input

Accepts natural language queries describing research interests (e.g., "privacy in federated learning")—no rigid keywords required.

Semantic Analysis

Uses Vector Search (Sentence-BERT) to understand the meaning of the query and match it against faculty publication abstracts.

Advisor-Centric Rank

Aggregates paper relevance to score the professor, prioritizing recent work and consistent research activity.

Smart Results

Generates an ordered list of faculty members, instantly identifying the best candidates based on overall alignment.

Contextual Evidence

Highlights the specific publications that triggered the match, providing transparency and proof of the advisor's relevance.

Data collection, Annotation and Preprocessing

1. Scrape Faculty Name, Image, Interests, etc from Texas A&M CSE directory

2. Use OpenAlex API: search for faculty → get faculty ID →get papers.

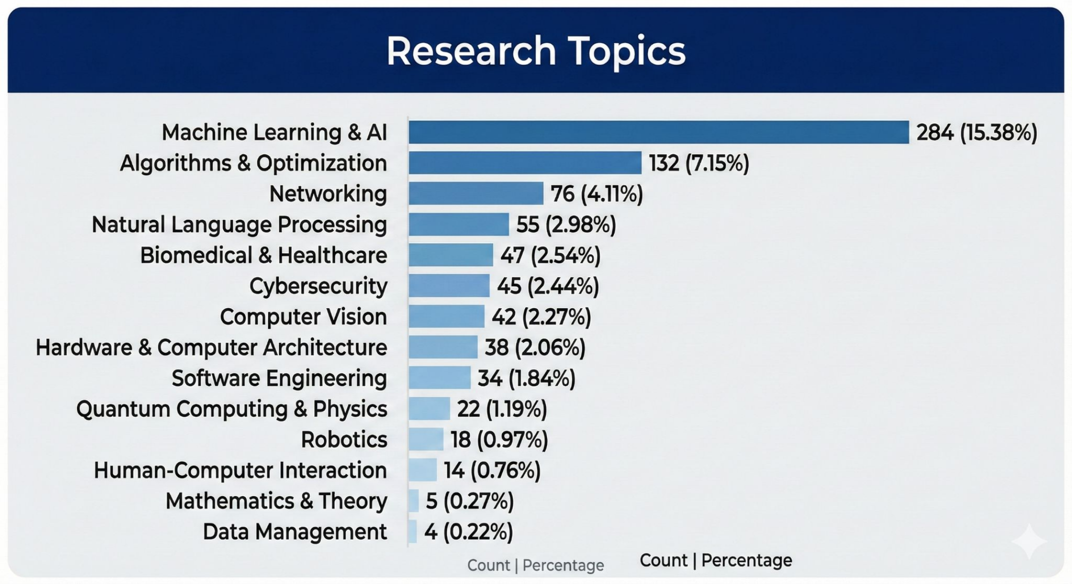
3. Create a database of faculties and paper mappings with metadata (year of publication, citation count, first author, etc.)

4. Generate embeddings for paper abstract.

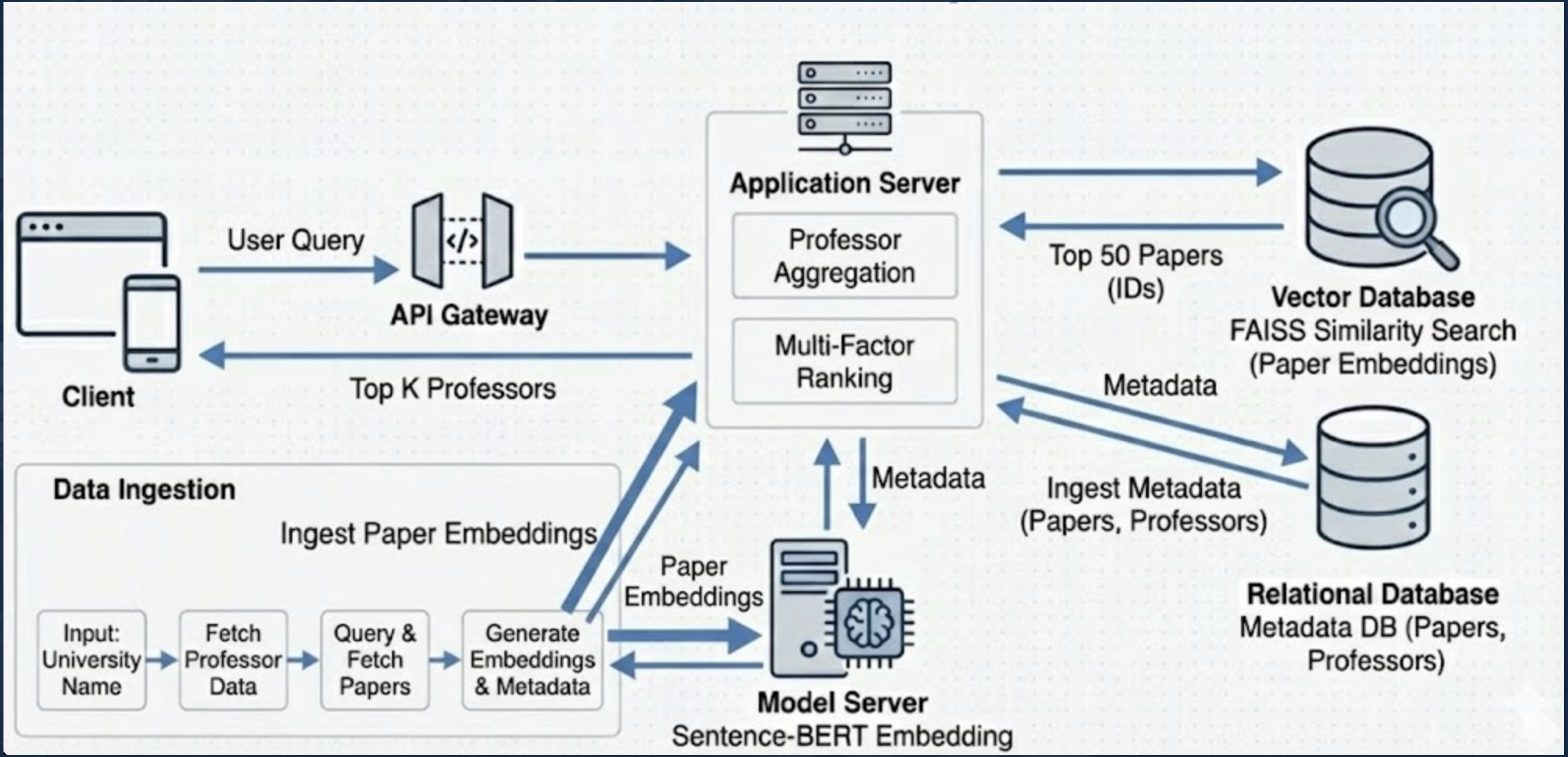
5. Build FAISS Index → Create search index.

6. Manually curate a small list of annotated results for testing. We used MRR and NDCG to analyze most relevant professor per query.

Data Statistics



Implementation



Technology Stack

Embedding Model:
Sentence-BERT
(all-MiniLM-L6-v2)

→

Vector Search:
FAISS
(Facebook AI
Similarity Search)

→

Backend:
FastAPI
(Python)

→

Database:
SQLite

→

Frontend:
HTML/CSS/JavaS
cript

Pre - Processing Step

Scrape Professors - Ingest script scraping from university websites

OpenAlex Author Search - Finding and matching professors with Author names and with tamu affiliations

Fetch Papers from OpenAlex - Getting all papers and metadata

Database Storage - Storing everything in SQLite

Generate Embeddings - Creating semantic embeddings with Sentence-BERT

Build FAISS Index - Building the search index

Implementation - Initial Paper Ranking

FAISS Similarity Search

Index Type: FAISS IndexFlatIP (Inner Product)
Method: Cosine similarity on normalized vectors
Retrieval: Top 50 papers (TOP_K_PAPERS = 50)
Performance: Sub-100ms search time

Advantages:
Fast: Sub-100ms for 1,847 papers
Scalable: Efficient approximate nearest neighbor search
Semantic: Understands meaning, not just keywords

Search Process

Query Embedding (384-dim)

↓

FAISS Index Search

↓

Top 50 Papers + Similarity Scores (0.0 to 1.0)

Implementation - Re ranking

Paper Aggregation
- Group top 50 papers by professor
- Extract: similarity, year, author_position

Multi-Factor Scoring

Weighted Similarity
- Formula: similarity × recency_weight`
- First-author bonus: × 1.2 (20% boost)
- Second-author bonus: × 1.1 (10% boost)
- Average across top 10 papers

Recency Weight (Exponential Decay)
- Formula: `exp(-0.1 × years_ago)
- Recent papers weighted higher
- Example: 2024 paper = 0.905, 2020 paper = 0.607

Activity Bonus
- Formula: min(recent_papers × 0.05, 0.20)
- Rewards professors with recent publications (2022-2025)
- Capped at 0.20 (4+ recent papers)

Citation Impact
- Formula: $\log_{10}(1 + \text{citations}) / 3.0 \times 0.15$
- Log-normalized citation scores
- Weighted contribution: 15% of final score

Final Score = avg(weighted_scores) + activity_bonus + citation_impact

Implementation - Re ranking

$\text{Score}_p = \text{AvgSim}_p + \text{Activity}_p + \text{Citation}_p$

1. Average Weighted Similarity (AvgSim_p)

$\text{AvgSim}_p = (1 / 10) \cdot \Sigma_{i=1}^{10} (\text{sim}_i \cdot \exp(-0.1 \cdot \text{yearsAgo}_i) \cdot \text{bonus}_i)$

- i : one of professor p's top 10 papers
- sim_i : semantic similarity between the query and paper i
- yearsAgo_i : how many years ago paper i was published
- $\exp(-0.1 \cdot \text{yearsAgo}_i)$: recency weight (recent papers > old papers)
- bonus_i : 1.2 if first author, 1.1 if second author, 1.0 otherwise

2. Activity Bonus (Activity_p)

$\text{Activity}_p = \min(0.05 \cdot \text{recentPapers}_p , 0.20)$

- recentPapers_p : number of papers by p from 2022–2025
- Adds 0.05 per recent paper, capped at 0.20 (4+ recent papers)

3. Citation Impact (Citation_p)

$\text{Citation}_p = [\log_{10}(1 + \text{citations}_p) / 3.0] \cdot 0.15$

- citations_p : total citations for p's papers
- $\log_{10}(1 + \text{citations}_p)$: compresses very large citation counts
- Division by 3.0 normalizes; × 0.15 => about 15% of final score

Variable Legend

- p : a professor
- i : one of p's papers considered in scoring
- Score_p : final advisor score used for ranking

Results and Model Comparison

For Queries- Use complex semantic queries that test conceptual understanding

Metric	BM25	Embedding	Improvement
MRR	0.3611	**0.3750**	**+3.8%**
NDCG@3	0.3977	**0.4401**	**+10.7%**
NDCG@4	0.3977	**0.4401**	**+10.7%**

Demo

Thank You