PROJECT PROPOSAL

Project Title:

Semantic Search Engine Using a Custom Multi-Domain Text Dataset

Author:

Muhammad Haris

Institution:

FAST NUCES

Date:

November 19, 2025

1 Problem Statement

Traditional keyword-based search engines fail to interpret user intent because they rely solely on surface-level word matching. This project aims to build a semantic search engine capable of retrieving documents based on meaning rather than exact keywords, utilizing a multi-domain dataset for improved understanding and relevance.

2 Dataset Description

A custom combined dataset, **UnifiedSearchCorpus 70k**, containing nearly 75,000 documents was constructed by sampling and cleaning multiple publicly available text sources. The goal is to ensure diversity, reduce overfitting, and improve generalization across real-world search scenarios.

2.1 Dataset Components

Dataset	Count	Source	Purpose
OpenWebText2	25,000	General Web Articles	General language understanding
S2ORC Scientific Abstracts	20,000	Academic Papers	Scientific search & domain knowledge
ArXiv Papers	15,000	Scientific Research	Technical search and structured text
C4 Dataset (English subset)	15,000	Web Scraped Text	High-quality diverse data

2.2 UnifiedSearchCorpus 65k — Cleaned Dataset Overview

Total Documents: 79,967

Tokens per Document:

- Average: 325
- Median: 102
- Minimum: 6
- Maximum: 4,997

Vocabulary Size: Reduced from 425,000+ raw tokens to 84,787 unique tokens after cleaning and rare-word filtering.

The final dataset uses the `tokens_final` cleaned set for indexing, ensuring consistency and optimal search performance. The 84,787-token lexicon, enhanced with semantic embeddings, enables broad concept-based query coverage beyond exact words.

Sample Query Coverage:

- `iphone` → 708 documents
- `sunny weather` → 6 documents

These preprocessing steps ensure a compact yet domain-rich lexicon while maintaining strong coverage across general and specialized queries.

3 Languages and Frameworks

- **C++** – Core search engine: indexing, forward/inverted index, barrels, and query processing.
- **Python** – Data preprocessing, text cleaning, and semantic embeddings (Word2Vec, GloVe, FastText).
- **React.js** – Responsive frontend interface with real-time search UI.
- **Flask / FastAPI** – Backend API connecting frontend with the C++ search engine.
- **File System** – Storage for lexicon, inverted index, forward index, and barrels (no external DB).
- **Cloud Deployment** – Heroku, Fly.io, or AWS for hosting; GitHub for version control.