Proof of concept and core component review

Student(s): Aritra Ray

Aspect	Description
Dataset info: Dataset creation / Data scraping (e.g. number of labeled queries, number of documents, tool used for scraping)	Custom Python crawler built using requests + BeautifulSoup. Extracted ~520 unique English-language Help Center pages from SurveyMonkey. Scraped {url, title, body} consistently and normalized into articles.csv. Built a labeled Golden Dataset (78 queries) with expected URLs + acceptable alternatives. Created an unseen evaluation set (40 queries) automatically derived from corpus pages excluded from the Golden set.
Data processing: basic data cleaning and normalization applied	Removed HTML tags, scripts, repeated navigation items. Normalized everything to lowercase, trimmed whitespace, and sanitized titles. Combined title + body into a unified searchable document field. Normalized URLs to a canonical form (lowercase, trailing slash removal). Cleaned Golden dataset fields and split acceptable_urls by ;.
Data representation: indexing strategy for storing and retrieving data (query + documents)	Stored each page as a structured document: { "url", "title", "text" } ● Indexed corpus using three independent retrieval representations: 1. BM25 lexical index (tokenized corpus) 2. TF-IDF unigram + bigram vector space 3. MiniLM-L6-v2 dense embeddings (384-dimensional) ● Dense embeddings were precomputed offline, saved to disk, and loaded instantly during inference to eliminate compute overhead. ● At retrieval time:

	1. BM25 gives lexical recall. 2. TF-IDF vector gives semantic-lite ranking. 3. Dense embeddings support semantic reranking with very low latency.
Core retrieval mechanism: retrieval/ranking model mechanism	1. Multi-representation hybrid retrieval For each query + its LLM variants: ● BM25 score ● TF-IDF cosine similarity score ● Fuzzy title match (token_set_ratio + WRatio) 2. Reciprocal Rank Fusion (RRF): All rankings are merged using RRF to give a balanced hybrid rank list. 3. Semantic reranking (fast CPU path) Instead of the heavy CrossEncoder (DeBERTa), final top-N candidates are reranked via: ● Precomputed MiniLM-L6-v2 (384-dim) embeddings ● Query → embedded at runtime using same encoder ● Reranking uses pure dot product cosine similarity → extremely fast 4. Query Reformulation (QPP-triggered LLM) ● If clarity (KL-divergence based) is low → Flan-T5 generates reformulated variants ● Injected rule-based variants optionally ● All variants fed into RRF-stage for maximum recall
Evaluation: metrics used for the evaluation	● success@1, success@3 ● recall@10, recall@20 ● MRR, NDCG@3, NDCG@10 ● count_999 (no hit in top 20) ● Metrics computed against *expected_url* ∪ *acceptable_urls* to reflect real-world acceptable targets.
Evaluation: baselines methods	1. Lexical baseline → BM25 only 2. RRF baseline → BM25 + TF-IDF + Fuzzy

	3. Hybrid + MiniLM reranker 4. Hybrid + LLM rewrites (QPP) 5. Strict/no-boost evaluation mode 6. Unseen-evaluation on auto-generated test set
Results: Preliminary/final results	Golden set (clean queries, known content) ● Nearly 1.0 across all metrics with hybrid + semantic reranking. ● Zero misses (count_999 = 0). Unseen evaluation set (new queries from corpus) ● success@1 ≈ 0.775 ● success@3, recall@10, recall@20 = 1.0 ● MRR ≈ 0.88 ● No 999s → semantic reranker generalizes well. ● Shows robustness while avoiding “memorizing” Golden data.
Any Blockers?	1. CrossEncoder instability & slowness (CPU) ● DeBERTa-v3 CrossEncoder was extremely slow, hitting 40–60 seconds per batch. ● Led to system freezes — impractical for 78–500 queries. ● Replaced with MiniLM embeddings, which brought near-instant reranking. 2. Memory and inference latency ● CrossEncoder input tokens are too long (full article bodies). ● Had to truncate bodies, then ultimately replace them with embeddings. 3. Crawling inconsistencies ● Some SM pages contained repeated template blocks → caused giant text blobs. ● Required cleaning to prevent irrelevant keyword dominance. 4. URL mismatch issues

	● Golden URLs sometimes differed in minor normalization from corpus URLs. ● Solved via URL normalization & acceptable_urls field. 5. Perfect scores can mislead stakeholders ● Full 1.0 metrics on Golden must be contextualized as a controlled environment.
Main challenges:	● Balancing speed vs. accuracy under CPU-only constraints. ● Making LLM rewrites meaningful despite clean Golden queries. ● Preventing retrieval inflation due to giant pages or repeated boilerplate. ● Designing a fair unseen-evaluation pipeline without deep learning overkill. ● Ensuring acceptable_urls are incorporated correctly in scoring. ● Avoiding overfit behaviour while still maximizing Golden coverage.
References:	● SurveyMonkey Help Center, public documentation ● Robertson & Zaragoza, "BM25 and Beyond" ● Salton et al., "TF-IDF and Vector Space Model" ● "Sentence-BERT (MiniLM Embeddings)" ● HuggingFace Transformers library ● "Reciprocal Rank Fusion" ● Emory IR Lab papers on query reformulation & generative PRF ● Flan-T5 model documentation