GitHub Repository: https://github.com/abhinavbatra06/Gmail-Assistant-

Gmail Assistant

Gmail RAG Assistant with intelligent query routing, hybrid retrieval, and evaluation logging.

Overview

NoticeBoard RAG is a modular retrieval-augmented generation (RAG) system that enables natural language querying over institutional email. The architecture integrates Gmail ingestion, multimodal document processing via Docling, recursive chunking, hybrid retrieval (dense + BM25), and grounded generation with GPT-4o-mini.

Why this project?

Institutional email is one of the richest sources of unstructured data. With modern GenAI models, we can extract precise answers to natural language queries—provided we have high-quality retrieval. However, most AI-powered email tools are closed systems with no visibility into their retrieval or chunking logic. NoticeBoard RAG provides full control over the entire pipeline—chunking strategies, embedding models, retrieval weights, and ranking—enabling experimentation and optimization for academic email workflows.

Quick Start

For detailed installation and setup instructions, see **TUTORIAL.md**.

Quick Setup Checklist

1. **Install dependencies**: `pip install -r requirements.txt`
2. **Set up environment**: Create `.env` with `OPENAI_API_KEY`
3. **Configure Gmail OAuth**: Add `gmail_creds.json` to `creds/` folder
4. **Run pipeline**: Ingest → Process → Chunk → Embed → Query

Features

- **Intent-aware routing**: Directs calendar queries to structured event database
- **Hybrid retrieval**: Combines dense embeddings with BM25 keyword search
- **Query optimization**: LLM-based rewriting and HyDE-generated hypothetical documents
- **Post-retrieval refinement**: LLM-based reranking and Small2Big context expansion
- **Temporal filtering**: Prioritizes recent and upcoming events
- **Multimodal processing**: Handles PDFs, Word docs, images, and calendar files

Documentation

- **TUTORIAL.md**: Complete step-by-step setup and usage guide
- **EXPERIMENTS.md**: Experimental setup, results, and analysis
- **config.yaml**: Configuration file with all system parameters

Experiments & Evaluation

The system includes a comprehensive evaluation framework to assess performance across different query types and scenarios.

Evaluation Approach

- **Query Categories**: Tests cover vague/ambiguous queries, typos/casual language, multi-intent/complex queries, implicit/contextual queries, and negation/edge cases
- **Automated Testing**: Batch query runner executes all test queries and logs results
- **Results Export**: Evaluation logs are exported to CSV format for analysis
- **Manual Accuracy Scoring**: Results are manually evaluated for accuracy (0/1) and analyzed using pivot tables

Quick Start for Experiments

```
# Run all test queries
python Eval/run_test_queries.py

# Export results to CSV
python Eval/export_eval_logs_csv.py
```

Results are saved to `Eval/eval_logs.csv`. For detailed experimental setup and results, see **EXPERIMENTS.md** and **TUTORIAL.md**.

Requirements

- Python 3.10+
- OpenAI API key
- Google Cloud Project with Gmail API enabled
- Virtual environment (recommended)

License

[Add your license here]