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

---

Experimental Evaluation

This document describes the experimental setup, methodology, and results for evaluating the NoticeBoard RAG system.

Table of Contents

Experimental Setup

System Configuration

The evaluation was conducted with the following system configuration:

- **Embedding Model**: `text-embedding-3-small` (OpenAI)
- **LLM Model**: `gpt-4o-mini` (OpenAI)
- **Vector Database**: ChromaDB with cosine similarity
- **Retrieval Method**: Hybrid (BM25 + Dense, α=0.5)
- **Chunk Size**: 600 tokens with 100 token overlap
- **Top-K Retrieval**: 5 chunks (configurable)
- **Query Optimization**: LLM-based rewriting enabled
- **Reranking**: LLM-based reranking enabled
- **Intent Routing**: Enabled (Router module)

Evaluation Dataset

- **Email Corpus**: Institutional emails from Gmail inbox
- **Date Range**: Configured in `config.yaml` (typically covers academic semester)
- **Total Emails**: Varies based on configuration; see database statistics
- **Total Chunks**: Varies based on email volume and chunking configuration
- **Test Queries**: 85 queries across 5 categories (7 Vague/Ambiguous, 8 Typos/Casual, 26 Multi-intent/Complex, 27 Implicit/Contextual, 1 Negation/Edge case, plus additional queries)

Evaluation Scripts

The evaluation framework consists of:

1. **`Eval/run_test_queries.py`**: Batch query runner that executes all test queries

2. `Eval/export_eval_logs_csv.py`: Exports query results to CSV format
3. `Eval/eval_logs.csv`: Exported results with all query-answer pairs
4. `Eval/eval logs (Concluded).xlsx`: Analysis spreadsheet with manual accuracy scores and pivot tables

---

Query Categories

Test queries are organized into 5 categories to assess system performance across different query types:

1. Vague/Ambiguous Queries

Queries with unclear intent or missing context that require the system to infer meaning.

Examples:

- "Tell me about the party"
- "Who do I need to talk to about registration?"
- "What did Mohammad send me?"
- "Anything about cedar project lately?"

Rationale: Tests the system's ability to handle queries with insufficient context and infer user intent.

2. Typos/Casual Queries

Queries with spelling errors, informal language, or non-standard formatting.

Examples:

- "when is the capston thing due"
- "quiz deadlines pls"
- "whos hosting the holiday thing in december events calendar"
- "gpu stuff for project?"

Rationale: Tests robustness to real-world query variations and typos that users commonly make.

3. Multi-intent/Complex Queries

Queries requiring multiple pieces of information, comparisons, or complex reasoning.

Examples:

- "What's my schedule like and what job applications should I prioritize? events calendar"
- "Show me everything related to capstone projects and internships"
- "Compare salaries across all the job postings"
- "Which jobs require NYU enrollment and which don't?"

Rationale: Tests the system's ability to handle compound queries and extract multiple related pieces of information.

4. Implicit/Contextual Queries

Queries requiring inference, background knowledge, or understanding of implicit relationships.

Examples:

- "Any jobs or research positions I can apply to?"
- "Who's been emailing me most about courses?"
- "What do I need to prepare for the end of semester?"
- "Show me everything about presentations - both capstone and course related"

Rationale: Tests the system's ability to understand implicit user needs and retrieve contextually relevant information.

5. Negation/Edge Cases

Queries with negation or unusual patterns that challenge standard retrieval.

Examples:

- "What courses are NOT being offered in spring?"

Rationale: Tests handling of negation and edge cases that may not be well-handled by standard retrieval methods.

Evaluation Methodology

Execution Process

1. **Run Test Queries**:

   ```
   python Eval/run_test_queries.py
   ```

   - Executes all queries through the RAG pipeline
   - Logs results to `db/memory.db`
   - Displays progress and results in terminal

2. **Export Results**:

   ```
   python Eval/export_eval_logs_csv.py
   ```

   - Extracts query-answer pairs from database
   - Exports to `Eval/eval_logs.csv`

3. **Manual Evaluation**:

   - Review each query-answer pair
   - Score accuracy: 0 (incorrect) or 1 (correct)
   - Add accuracy scores to spreadsheet

4. **Analysis**:
    - Create pivot tables by category, intent, and module
    - Calculate accuracy rates and performance metrics
    - Identify patterns and failure modes

Evaluation Metrics

- **Accuracy**: Binary score (0/1) for each query-answer pair
- **Intent Classification Accuracy**: Correctness of intent routing
- **Retrieval Quality**: Relevance of retrieved chunks
- **Response Latency**: Time to generate answer
- **Chunk Retrieval Count**: Number of chunks used per query
- **Module Usage**: Frequency of Predict vs. RAG retrieval

Results

Summary Statistics

Metric	Value
Total Queries	85
Overall Accuracy	74.1%
Average Latency	12.27s
Average Chunks Retrieved	5.4

Accuracy by Category

Category	Queries	Accuracy	Avg Latency	Avg Chunks	Notes
Typos/Casual	8	87.5%	13.73s	6.2	Excellent performance despite spelling errors
Implicit/Contextual	27	81.5%	14.42s	5.0	Strong performance on context-dependent queries
Multi-intent/Complex	26	73.1%	12.13s	5.2	Moderate performance on compound queries
Vague/Ambiguous	7	71.4%	12.32s	5.0	Moderate performance; struggles with unclear intent
Negation/Edge Cases	1	100.0%	13.99s	5.0	Limited sample size; handled negation well

Performance by Intent Type

Intent	Count	Accuracy	Avg Latency	Module Used
Information	34	85.3%	13.96s	RAG (Retriever)
General	10	80.0%	14.50s	RAG (Retriever)
Deadline	13	69.2%	13.82s	RAG (Retriever)
Sender	6	66.7%	12.55s	RAG (Retriever)
Calendar	21	57.1%	7.06s	Predict
Summarization	1	100.0%	20.05s	RAG (Retriever)

Module Performance Comparison

Module	Queries Handled	Accuracy	Avg Latency	Avg Chunks
RAG (Retriever)	64	79.7%	13.98s	5.0
Predict	21	57.1%	7.06s	6.8

Key Observation: The Predict module is significantly faster (7.06s vs 13.98s) but has lower accuracy (57.1% vs 79.7%). This suggests that while structured event retrieval is fast, it may miss relevant information or struggle with complex calendar queries that require additional context from email content.

Detailed Results

For complete results, see:

- `Eval/eval_logs.csv`: Raw query-answer pairs with all metadata
- `Eval/eval logs (Concluded).xlsx`: Analysis spreadsheet with accuracy scores and pivot tables

---

Analysis

Key Findings

1. **Category Performance**:
    - **Typos/Casual queries** achieved the highest accuracy (87.5%), demonstrating the system's robustness to spelling errors and informal language. The hybrid retrieval (BM25 + Dense) effectively handles lexical variations.
    - **Implicit/Contextual queries** performed well (81.5%), showing the system can infer user intent and retrieve contextually relevant information.
    - **Multi-intent/Complex queries** achieved moderate accuracy (73.1%), indicating the system can handle compound queries but may miss some information when queries require multiple pieces of data.
    - **Vague/Ambiguous queries** had lower accuracy (71.4%), as expected, since these queries lack sufficient context for precise retrieval.
2. **Intent Routing**:

- Calendar queries were successfully routed to Predict module 100% of the time (21/21 queries)
- However, calendar queries had the lowest accuracy (57.1%) among all intent types, suggesting that while routing works correctly, the Predict module may need improvement or calendar queries may benefit from hybrid Predict+RAG approach
- Information queries achieved the highest accuracy (85.3%), demonstrating strong performance of the RAG retriever for factual queries
- Intent classification appears accurate based on routing behavior

3. **Retrieval Quality**:

- Average chunks retrieved per query: 5.4 (within the configured top_k=5 range)
- Hybrid retrieval (BM25 + Dense) was used for all queries, providing both semantic and lexical matching
- Calendar queries retrieved more chunks on average (6.8) when using Predict module, suggesting the module may be retrieving multiple events
- Deadline queries retrieved fewer chunks (4.8), potentially indicating more focused retrieval

4. **Error Patterns** (based on manual evaluation comments):

- **Missing context/information**: Several failures noted "missed the stuff related to..." or "missing all the opportunities", indicating incomplete retrieval. This was particularly common in multi-intent queries where the system addressed part of the query but missed other aspects.
- **Confusion between similar items**: Calendar queries sometimes confused cancelled vs. active events ("Got confused between 2 invites - one of which was cancelled"), suggesting the need for better event status tracking.
- **Misunderstanding query intent**: Some queries were not understood correctly ("Did not understand the question"), particularly for vague or ambiguous queries.
- **Incomplete responses**: Responses sometimes lacked comprehensive information ("There is more information than this in my inbox"), indicating retrieval may need to be more exhaustive for certain query types.
- **Context gaps**: Some queries missed relevant context ("Missed the context completely"), suggesting retrieval scope may need expansion for implicit queries.

5. **Query Category Distribution**:

- The evaluation included 85 queries, with Implicit/Contextual queries being the largest category (27 queries, 31.8%)
- Multi-intent/Complex queries were the second largest (26 queries, 30.6%)
- Some queries in the dataset did not match the predefined categories exactly (16 queries classified as "Unknown"), which achieved 56.2% accuracy and may represent edge cases or queries added during evaluation

Strengths

- **Robustness to typos and informal language**: Achieved 87.5% accuracy on typos/casual queries, demonstrating that hybrid retrieval (BM25 + Dense) effectively handles spelling variations and informal language patterns.

- **Strong performance on information queries**: Achieved 85.3% accuracy on information intent queries, showing the RAG pipeline excels at factual information extraction.
- **Effective intent classification**: Intent routing worked correctly, with calendar queries consistently routed to Predict module and other queries to RAG retriever.
- **Fast calendar query processing**: Predict module processes calendar queries in 7.06s on average, nearly 2x faster than RAG retrieval (13.98s), providing a speed advantage for structured event queries.
- **Good handling of implicit queries**: 81.5% accuracy on implicit/contextual queries shows the system can infer user needs and retrieve relevant information without explicit keywords.
- **Hybrid retrieval effectiveness**: The combination of BM25 and dense embeddings provides both semantic understanding and lexical precision, contributing to overall strong performance.

Limitations

- **Calendar query accuracy**: Despite fast processing, calendar queries achieved only 57.1% accuracy when routed to Predict module. This suggests that structured event retrieval may miss relevant context or struggle with complex calendar queries that require additional email content.
- **Incomplete information retrieval**: Several failure cases noted missing information ("missed the stuff related to...", "missing all the opportunities"), indicating that the system may not always retrieve comprehensive results, especially for queries requiring multiple pieces of information.
- **Confusion with similar items**: Calendar queries sometimes confused cancelled vs. active events, suggesting the system needs better handling of event status and deduplication.
- **Vague query handling**: Vague/ambiguous queries achieved 71.4% accuracy, indicating the system struggles when queries lack sufficient context or have unclear intent.
- **Multi-intent query completeness**: Multi-intent/complex queries achieved 73.1% accuracy, with failures often related to missing parts of compound queries rather than complete failures.
- **Context gaps**: Some queries missed relevant context entirely, suggesting retrieval may need improvement in understanding query scope and related information.

Recommendations

1. **For Calendar Queries (Predict Module)**:
    - **Hybrid approach**: Consider combining Predict module results with RAG retrieval for calendar queries to capture both structured events and related email context. This could improve accuracy from 57.1% while maintaining speed benefits.
    - **Event status handling**: Improve handling of cancelled events and event updates to avoid confusion between similar events.
    - **Context expansion**: When Predict module finds events, also retrieve related email chunks that mention the event to provide more comprehensive answers.
2. **For Vague/Ambiguous Queries**:

- **Query clarification**: Implement a query clarification step that asks users to refine vague queries before retrieval.
- **Expanded retrieval**: Increase top_k for vague queries to retrieve more candidate chunks, then use reranking to select the most relevant.
- **Context-aware expansion**: Use conversation history or user preferences to disambiguate vague queries.

3. **For Multi-intent/Complex Queries**:
   - **Query decomposition**: The system already supports sub-query decomposition; ensure it's enabled and tuned for complex queries.
   - **Comprehensive retrieval**: Increase retrieval depth for multi-intent queries to ensure all parts of the query are addressed.
   - **Answer synthesis**: Improve LLM prompts to explicitly check that all parts of compound queries are answered.
4. **For Information Completeness**:
   - **Iterative retrieval**: Use the iterative retrieval feature more aggressively for queries that may require multiple pieces of information.
   - **Small2Big expansion**: Enable Small2Big expansion to retrieve more context around initially matched chunks, especially for queries that failed due to incomplete information.
   - **Post-processing validation**: Add a validation step that checks if the answer addresses all aspects of the query before returning results.
5. **General Improvements**:
   - **Reranking enhancement**: Improve reranking to better identify and prioritize chunks that comprehensively address the query.
   - **Query understanding**: Enhance query understanding to better identify when queries require multiple pieces of information or comparisons.
   - **Error recovery**: Implement fallback mechanisms when initial retrieval fails, such as expanding the search scope or trying alternative query formulations.

Reproducibility

Prerequisites

- Complete system setup (see TUTORIAL.md)
- Email corpus processed and indexed
- OpenAI API key with sufficient quota

Reproducing Results

1. **Ensure system is ready**:

```
# Verify pipeline is complete
python -c "from src.db_helper import DBHelper; db =
```

```
DBHelper('db/emails.db'); print(f'Emails: {db.get_chunking_stats()
[\"total_emails\"]}')"
```

2. **Run evaluation**:

```
python Eval/run_test_queries.py
python Eval/export_eval_logs_csv.py
```

3. **Compare results**:
   - Compare `Eval/eval_logs.csv` with previous runs
   - Note: Results may vary slightly due to:
     - LLM non-determinism
     - Changes in email corpus
     - API response variations

Configuration for Reproducibility

To reproduce exact results, use the following configuration in `config.yaml`:

```yaml
rag:
  top_k: 5
  query_optimization: rewrite
  enable_hybrid_retrieval: true
  hybrid_alpha: 0.5
  enable_reranking: true
  rerank_method: llm
  enable_routing: true
  enable_predict: true
```

Notes on Variability

- **LLM Responses**: GPT-4o-mini may produce slightly different answers for the same query
- **Retrieval Order**: Chunk ordering may vary due to similarity score calculations
- **Intent Classification**: Router may classify queries differently based on context

For consistent evaluation, consider:

- Setting random seeds where possible
- Running multiple trials and averaging results
- Using deterministic retrieval settings

Additional Resources

Evaluation Files

- `Eval/run_test_queries.py`: Script to run all test queries
- `Eval/export_eval_logs_csv.py`: Script to export results
- `Eval/eval_logs.csv`: Exported results (generated)
- `Eval/eval logs (Concluded).xlsx`: Analysis spreadsheet with accuracy scores

Related Documentation

- **TUTORIAL.md**: Step-by-step instructions for running experiments
- **README.md**: Quick overview of evaluation approach

Last Updated: [Date] **Evaluation Version**: [Version number if applicable]