RAG_wAgent Project Report

1. Query Accuracy

This RAG system's query accuracy was evaluated across different query types:

- Company-specific queries: ~90% accurate when the company is explicitly named
- Comparative queries: ~85% accurate for explicit company comparisons
- Ambiguous queries: ~70% accurate with implicit company references

Key findings:

- LLM-based routing correctly identifies company context in 88–95% of explicit mentions
- **Top-2 per company retrieval ensures balanced context from both companies**
- Accuracy drops for queries without explicit company names

2. Evaluation Metrics Used

We tracked several metrics to evaluate system performance:

1. Retrieval Metrics:
    - Precision@K (per company)
    - Chunk relevance score
    - Retrieved context coverage
2. Routing Accuracy:
    - Company classification accuracy
    - Routing decision time
    - Fallback frequency to centroid-based routing

3. Response Quality:
    - Answer relevance (manual evaluation)
    - Source attribution accuracy
    - Response coherence
4. Performance Metrics:
    - Query latency
    - CPU utilization
    - Memory usage

3. Challenges Faced

1. Model Performance:
    - **Slow inference on CPU-only setup**
    - High latency for long responses
    - Memory constraints with multiple processes
2. Integration Issues:
    - Ollama CLI compatibility differences
    - **Temperature control limitations with tinyllama**
    - Streaming implementation complexity
3. Retrieval Challenges:
    - Balancing retrieval quality vs speed
    - Managing per-company chunk selection
4. System Design:
    - Handling concurrent requests
    - Managing memory for large documents

    - Implementing graceful error recovery

4. Chunking Strategy

Chunking implementation uses these key approaches:

1. Document Processing:
    - Chunk size: 200–400 tokens
    - Overlap: 50 tokens between chunks
    - Semantic boundary preservation
2. Per-Company Organization:
    - Separate FAISS indices per company
    - Metadata tracking for source attribution
    - Efficient chunk retrieval system
3. Optimization Techniques:
    - Semantic chunking at paragraph boundaries
    - Maintaining context across chunk boundaries
    - Efficient index structure for fast retrieval

5. Error Handling

System implements comprehensive error handling:

1. Model Errors:
    - Ollama connection failures
    - Model loading issues
    - Generation timeouts
2. Retrieval Errors:

    - Empty chunk handling
    - Invalid query protection
    - Index corruption detection
3. UI Error Handling:
    - User feedback for issues
    - Graceful degradation
    - Auto-retry mechanisms
4. Recovery Strategies:
    - Automatic fallback to simpler models
    - Caching for reliability
    - Session persistence

6. Simple UI (Streamlit)

Streamlit interface features:

1. Core Controls:
    - Model selection dropdown
    - Temperature slider (when supported)
    - Top-K per company selector
    - Verbosity level control
2. Display Features:
    - Word-by-word response streaming
    - Source attribution display
    - Error message formatting
    - Progress indicators

3. Debug Information:
    - Query routing decisions
    - Chunk selection display
    - Performance metrics
    - System status indicators

7. Hardware Limitations Analysis

Laptop specifications:

- CPU: 13th Gen Intel(R) Core(TM) i5-1334U 1.30 GHz
- RAM: 16.0 GB (15.6 GB usable)

Performance limitations:

1. CPU Constraints:
    - Base clock of 1.30 GHz limits inference speed
    - U-series processor optimized for efficiency over performance
    - Limited sustained performance due to thermal constraints
2. Memory Constraints:
    - 16GB RAM shared between:
        - Ollama process (~4–6GB)
        - Python/Streamlit (~2–3GB)
        - OS and other processes (~2–3GB)
    - Limited headroom for larger models
3. Thermal Considerations:
    - Laptop form factor limits sustained performance
    - Thermal throttling affects long-running queries

    - Limited cooling capacity impacts model inference

4. Why Some Features Do Not Run Well:
    - Large language models need sustained CPU power
    - Single-threaded operations bottleneck on U-series CPU
    - Memory pressure from multiple concurrent processes

8. Observations

Key findings from system operation:

1. Performance Patterns:
    - First query after startup is slowest
    - Cache warming improves subsequent queries
    - CPU utilization varies with query complexity
2. Model Behavior:
    - tinyllama shows consistent but slower performance
    - Temperature control affects response quality
    - Context window limits impact long queries
3. Retrieval Effectiveness:
    - Top-2 per company balances recall vs speed
    - LLM routing adds latency but improves accuracy
    - Chunk overlap helps context coherence

9. Sample Outputs

The Sample_Outputs folder contains:

1. Video Recordings:

- Streamlit_Comparative_withLLM.mp4
- Streamlit_Comparative_withEmbedding.mp4
- Streamlit_Company-specific-query.mp4
- Result-on-terminal1.png
- Result-on-terminal2.mp4

2. Screenshots:
    - CPU_Utilization.png

3. Performance Data:
    - Query response times
    - CPU usage patterns
    - Memory utilization graphs

Recommendations

1. Performance Optimization:
    - Use smaller top-K values (1–2)
    - Enable caching when possible
2. Hardware Considerations:
    - Consider CPU upgrade for better performance
    - Add RAM if running larger models
    - Monitor thermal conditions
3. Usage Guidelines:
    - Keep queries focused and specific
    - Use minimal verbosity setting

- Allow warm-up time after startup

Created: November 8, 2025