📊 Evaluation Metrics Documentation

Hybrid RAG System with Automated Evaluation

GitHub Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

---

Table of Contents

---

1. Mandatory Metric: MRR (Mean Reciprocal Rank)

1.1 Definition

Mean Reciprocal Rank (MRR) measures how well a retrieval system ranks the first relevant document. It is the average of reciprocal ranks across all queries.

1.2 Mathematical Formulation

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Where:

- $|Q|$ = Total number of queries
- $rank_i$ = Rank position of the first relevant document for query i
- If no relevant document is found, $\frac{1}{rank_i} = 0$

1.3 Example Calculation

Query	Relevant URL Rank	Reciprocal Rank
Q1	1	1/1 = 1.000
Q2	3	1/3 = 0.333
Q3	2	1/2 = 0.500
Q4	Not Found	0.000

$$MRR = \frac{1.0 + 0.333 + 0.5 + 0.0}{4} = 0.458$$

1.4 Why MRR for RAG?

1. **Focus on Top Result**: In RAG systems, the first relevant chunk heavily influences answer quality

2. **User Experience**: Users typically examine top results first
3. **Penalizes Poor Ranking**: Low ranks receive exponentially smaller scores

1.5 Interpretation Guidelines

MRR Score	Interpretation	System Quality
0.90 - 1.00	Excellent	Relevant docs almost always rank first
0.70 - 0.89	Good	Relevant docs typically in top 2
0.50 - 0.69	Moderate	Relevant docs usually in top 3-4
0.30 - 0.49	Fair	Relevant docs often below top 3
0.00 - 0.29	Poor	Significant ranking issues

1.6 Our Results

Method	MRR Score	Interpretation
Dense	0.3025	Fair - relevant docs often below top 3
Sparse (BM25)	**0.4392**	Fair-Moderate - best performance
Hybrid (RRF)	0.3783	Fair - between dense and sparse

---

2. Custom Metric 1: Recall@10

2.1 Justification for Selection

Why Recall@10 was chosen:

1. **Relevance to RAG Context Windows**: Our system retrieves top-10 chunks for answer generation. Recall@10 directly measures how many relevant documents appear in this context window.

2. **Complementary to MRR**: While MRR focuses on rank of first relevant result, Recall@10 measures coverage of all relevant documents.

3. **Answer Quality Correlation**: Higher Recall@10 means more relevant information available to the LLM for answer generation.

4. **Standard in IR Research**: Widely used metric enabling comparison with other systems.

5. **Practical Threshold**: 10 documents represent a reasonable context size for LLM processing without token overflow.

2.2 Mathematical Formulation

$$Recall@K = \frac{|Retrieved_K \cap Relevant|}{|Relevant|}$$

For our implementation:

$$Recall@10 = \frac{\text{Number of relevant URLs in top 10 results}}{\text{Total number of relevant URLs for query}}$$

2.3 Detailed Calculation

Step 1: Retrieve top 10 chunks for query **Step 2:** Extract unique source URLs from retrieved chunks **Step 3:** Compare with ground truth relevant URLs **Step 4:** Calculate intersection ratio

Example:

```
Ground Truth URLs: [url_A, url_B]
Retrieved URLs (top 10): [url_A, url_C, url_D, url_E, ...]

Intersection: {url_A}
Recall@10 = 1/2 = 0.5 (50%)
```

2.4 Implementation

Code Reference: evaluate_chromadb_fast.py

```python
def compute_recall_at_k(retrieved_urls, relevant_urls, k=10):
    """
    Compute Recall@K

    Args:
        retrieved_urls: List of URLs from top K retrieved chunks
        relevant_urls: List of ground truth relevant URLs
        k: Number of top results to consider (default: 10)

    Returns:
        float: Recall score between 0 and 1
    """
    if not relevant_urls:
        return 0.0

    retrieved_set = set(retrieved_urls[:k])
    relevant_set = set(relevant_urls)

    intersection = retrieved_set & relevant_set
    recall = len(intersection) / len(relevant_set)

    return recall
```

2.5 Interpretation Guidelines

Recall@10 Score	Interpretation	Implication
0.90 - 1.00	Excellent	Almost all relevant docs in top 10

Recall@10 Score	Interpretation	Implication
0.70 - 0.89	Good	Most relevant docs retrieved
0.50 - 0.69	Moderate	About half of relevant docs found
0.30 - 0.49	Fair	Significant relevant docs missing
0.00 - 0.29	Poor	Most relevant docs not in top 10

2.6 Our Results

Method	Recall@10	Interpretation
Dense	0.33	Fair - 1/3 of relevant docs retrieved
Sparse (BM25)	**0.47**	Fair-Moderate - best coverage
Hybrid (RRF)	0.43	Fair - improved over dense

---

3. Custom Metric 2: Answer F1 Score

3.1 Justification for Selection

Why Answer F1 was chosen:

1. **End-to-End Quality Assessment**: Unlike retrieval metrics, F1 evaluates the final generated answer quality.
2. **Token-Level Precision and Recall**: Captures both what the system correctly included (precision) and what relevant information was covered (recall).
3. **Handles Partial Matches**: Unlike exact match, F1 gives credit for partially correct answers.
4. **Standard NLP Metric**: Widely used in QA systems (SQuAD, etc.), enabling benchmarking.
5. **Balances Conciseness and Completeness**: Precision prevents over-generation; recall ensures coverage.

3.2 Mathematical Formulation

$$Precision = \frac{|Generated \cap Reference|}{|Generated|}$$

$$Recall = \frac{|Generated \cap Reference|}{|Reference|}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where:

- $Generated$ = Set of tokens in generated answer (after normalization)
- $Reference$ = Set of tokens in ground truth answer

3.3 Detailed Calculation

Preprocessing Steps:

1. Convert to lowercase
2. Remove punctuation
3. Tokenize into words
4. Remove stopwords (optional)

Example:

```
Generated Answer: "The capital of France is Paris"
Reference Answer: "Paris is the capital city of France"

Generated Tokens: {the, capital, of, france, is, paris}
Reference Tokens: {paris, is, the, capital, city, of, france}

Common Tokens: {the, capital, of, france, is, paris} = 6
Generated Count: 6
Reference Count: 7

Precision = 6/6 = 1.00
Recall = 6/7 = 0.857
F1 = 2 × (1.00 × 0.857) / (1.00 + 0.857) = 0.923
```

3.4 Implementation

Code Reference: evaluate_chromadb_fast.py

```
def compute_answer_f1(generated_answer: str, reference_answer: str) -> 
float:
    """
    Compute token-level F1 score between generated and reference answers.

    Args:
        generated_answer: Model's generated answer
        reference_answer: Ground truth answer

    Returns:
        float: F1 score between 0 and 1
    """
    def normalize(text):
        """Normalize text: lowercase, remove punctuation, tokenize"""
        text = text.lower()
        text = re.sub(r'[^\w\s]', '', text)
        tokens = text.split()
        return set(tokens)

    gen_tokens = normalize(generated_answer)
    ref_tokens = normalize(reference_answer)

    if not gen_tokens or not ref_tokens:
```

```
        return 0.0

    common = gen_tokens & ref_tokens

    precision = len(common) / len(gen_tokens) if gen_tokens else 0
    recall = len(common) / len(ref_tokens) if ref_tokens else 0

    if precision + recall == 0:
        return 0.0

    f1 = 2 * (precision * recall) / (precision + recall)
    return f1
```

3.5 Interpretation Guidelines

Answer F1 Score	Interpretation	Answer Quality
0.80 - 1.00	Excellent	Nearly identical to reference
0.60 - 0.79	Good	Most key information present
0.40 - 0.59	Moderate	Partial overlap with reference
0.20 - 0.39	Fair	Some relevant tokens, major gaps
0.00 - 0.19	Poor	Minimal overlap with reference

3.6 Our Results

Method	Avg Answer F1	Interpretation
Dense	~0.05	Poor - answers don't match reference
Sparse (BM25)	~0.05	Poor - answers don't match reference
Hybrid (RRF)	~0.05	Poor - answers don't match reference

Analysis of Low F1 Scores: The low F1 scores indicate a mismatch between generated answers and reference answers. This is often due to:

1. **Answer Style Differences**: LLM generates verbose answers vs. concise references
2. **Paraphrasing**: Same meaning but different words
3. **Context Window Issues**: Retrieved context doesn't contain answer information

---

4. Metric Comparison Summary

4.1 Metric Properties

Property	MRR	Recall@10	Answer F1
Measures	Ranking Quality	Coverage	Answer Quality

Property	MRR	Recall@10	Answer F1
Focus	First relevant result	All relevant results	Generated text
Range	0 to 1	0 to 1	0 to 1
Higher is	Better	Better	Better
Stage	Retrieval	Retrieval	Generation

4.2 Combined Insights

Method	MRR	Recall@10	Answer F1	Overall
Dense	0.30	0.33	0.05	Weak retrieval, poor answers
Sparse	**0.44**	**0.47**	0.05	**Best retrieval**, poor answers
Hybrid	0.38	0.43	0.05	Moderate retrieval, poor answers

4.3 Key Findings

1. **BM25 (Sparse) dominates**: Outperforms dense embeddings on Wikipedia-style factual content
2. **Hybrid provides balance**: RRF fusion gives intermediate performance
3. **Answer generation is the bottleneck**: Low F1 scores across all methods indicate LLM generation needs improvement
4. **Retrieval ≠ Answer Quality**: Good retrieval (Recall@10 = 0.47) doesn't guarantee good answers (F1 = 0.05)

---

5. Implementation References

5.1 Code Files

Component	File	GitHub Link
Evaluation Script	`evaluate_chromadb_fast.py`	View
RAG System	`chromadb_rag_system.py`	View
Metrics Library	`evaluation/comprehensive_metrics.py`	View

5.2 Result Files

File	Purpose	GitHub Link
`evaluation_results_chromadb.csv`	Detailed results (300 rows)	View
`evaluation_summary_chromadb.json`	Summary statistics	View
