# Retrieval Metrics & Optimization

## Evaluating & Enhancing RAG Performance

Building reliable, high-performing retrieval systems through quantitative evaluation and strategic optimization techniques.

# Learning Objectives

By the end of this session, you will gain both theoretical understanding and practical experience with RAG evaluation and optimization.

1. Understand retrieval metrics by implementing **MRR, NDCG, and Recall@K**
2. Implement **contextual embedding** strategies to improve semantic search capabilities
3. Learn hierarchical **summary techniques** to optimize context retrieval and reduce latency
4. Apply optimization techniques to build more reliable and efficient RAG systems

# What Are Retrieval Metrics?

Think of retrieval metrics as your **quality control system for RAG**, similar to how a teacher grades tests to measure student performance.

When evaluating retrieval quality, these metrics help you understand:

1. *If your system finds the most relevant information first*
2. *How well it ranks results in order of importance*
3. *Whether it captures all the important context needed*

Because retrieval metrics combine different ways of measuring success, they help you build RAG systems that are both precise and comprehensive. When you measure your retrieval performance, you get **concrete scores to guide optimization** rather than just gut feelings.

# Three Key Retrieval Metrics

1. **Mean Reciprocal Rank (MRR):** Measures how quickly we find the first correct answer. Perfect for single-answer scenarios like factual QA.

2. **Normalized Discounted Cumulative Gain (NDCG):** Evaluates how well we rank all relevant results. Ideal for cases where order and relevance grades matter.

3. **Recall@K:** Shows if we're finding all important information. Critical when comprehensive coverage is needed for vast amounts of information.

GAUNTLET AI

# Multiple Ways to Measure RAG

Think of retrieval metrics as your toolkit for optimization - there's no **single "perfect" metric**, but these three provide a solid foundation:

When You're Just Starting:

- Focus on these metrics for your MVP
- They cover key aspects of retrieval quality
- Easy to implement and understand

Why These Three:

- **MRR**: Quick check of first-result accuracy
- **NDCG**: Overall ranking effectiveness
- **Recall@K**: Retrieval completeness

# Mean Reciprocal Rank (MRR)

**When Should I Use It:**

- Have a single correct answer you need to find?
- Building a factual QA or product search system?
- Need to measure time-to-first relevant result?

**What Does It Measure:**

- Calculates how many tries before first correct answer
- Returns 1.0 if first result is correct, 0.5 if second, 0.33 if third
- Averages this score across all your test queries

**Target MRR Numbers:**

- Above 0.8: Excellent first-result accuracy
- 0.5-0.8: Good but room for improvement
- Below 0.5: Consider returning your retrieval

# MRR: A Visual Breakdown

See how MRR evaluates the position of our first relevant result (green box) across multiple queries.

1. For each query, we take 1/position, then average these values. When relevant results appear earlier (like Query 2), we get higher scores.

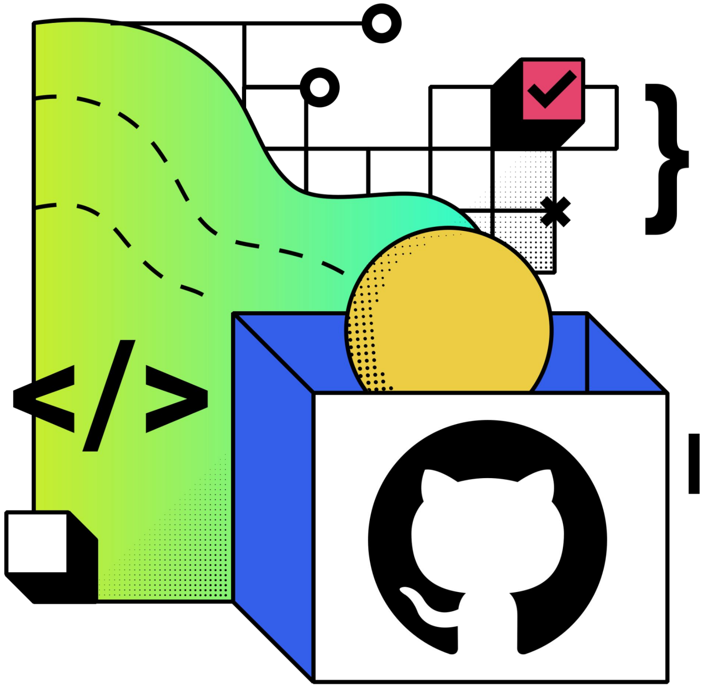
2. When they appear later (like Query 3), the score drops significantly. This position-based penalty helps us understand if our retrieval system is putting the best results first.

3. **Final MRR = 0.583**
   The average of 0.5 + 1.0 + 0.25 = 1.75/3


GAUNTLET AI

# MRR: Getting the Right Answer Fast

In question-answering and factual search systems, MRR helps optimize for scenarios where finding the correct answer immediately is crucial - making it perfect for virtual assistants, tech docs, and support.



# Normalized Discounted Cumulative Gain (NDCG)

**When Should I Use It:**
- Working with graded relevance scores?
- Care about the entire ranking order?
- Need to compare different ranking algorithms?

**What Does It Measure:**
- Evaluates quality of entire ranked list
- Rewards relevant documents appearing higher up
- Normalizes scores for fair comparison across queries

**Target MRR Numbers:**
- Above 0.9: Exceptional ranking performance
- 0.7-0.9: Strong real-world performance
- Below 0.7: May need to improve ranking strategy

# NDCG: When Ranking Quality Matters

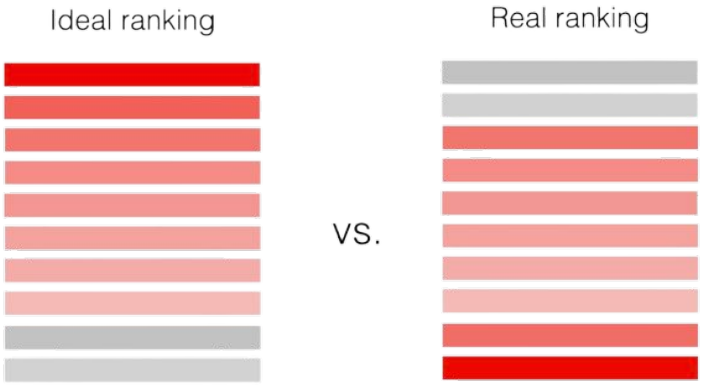
Think of NDCG like a restaurant review scoring system:

- A 5-star review at position 1 gets full value
- Same 5-star review at P5 gets discounted heavily
- 4-5 star reviews ranking high is better than just one
- Poor results (1-2 stars) ranking high hurts your score

**Real-World Example**:

When searching for "data science courses", some results are:

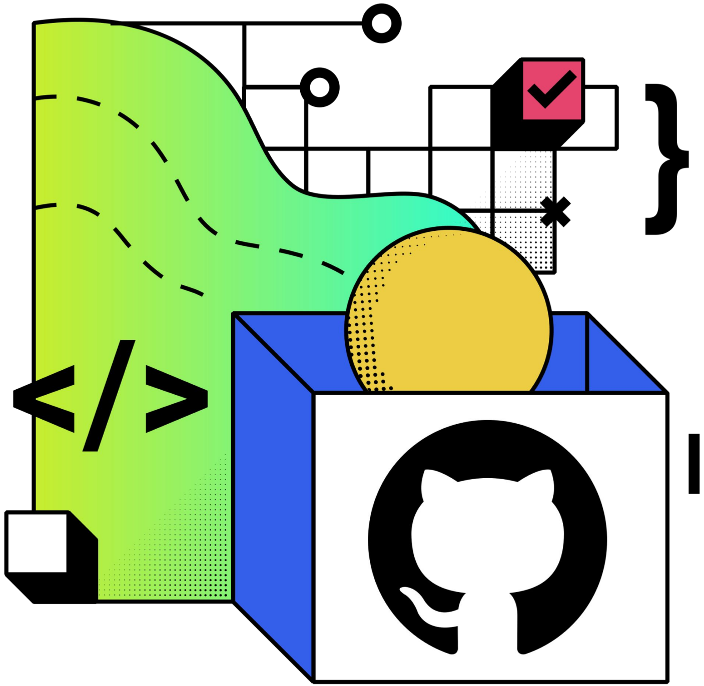
1. Perfect match: "Introduction to Data Science" (rel: 3)
2. Good match: "Python for Data Analysis" (rel: 2)
3. Partial match: "Business Analytics Basics" (rel: 1)
4. Poor match: "Web Development 101" (rel: 0)




*Accessed 15 Jan 2025, <https://www.evidentlyai.com/ranking-metrics/ndcg-metric>.*

# NDCG: Ranking Quality Matters

In content recommendation and search systems, NDCG helps evaluate how well results are ranked based on their relevance - making it essential for movie streaming platforms and research databases.




GAUNTLET AI

# Recall@K

**When Should I Use It:**

- Need to find ALL relevant documents?
- Working on research or legal discovery?
- Want to measure retrieval completeness?

**What Does It Measure:**

- Percentage of relevant docs found in top K results
- How comprehensive your retrieval coverage is
- Whether important information is being missed

**Target MRR Numbers:**

- Above 0.95: Comprehensive retrieval (critical for legal)
- 0.8-0.95: Strong coverage for most applications
- Below 0.8: Risk of missing important information

**Note: Common K values are 5, 10, 20, or 100 depending on use case*

GAUNTLET AI

# Recall@K: Understanding Coverage

Think of Recall@K like a legal document review process:

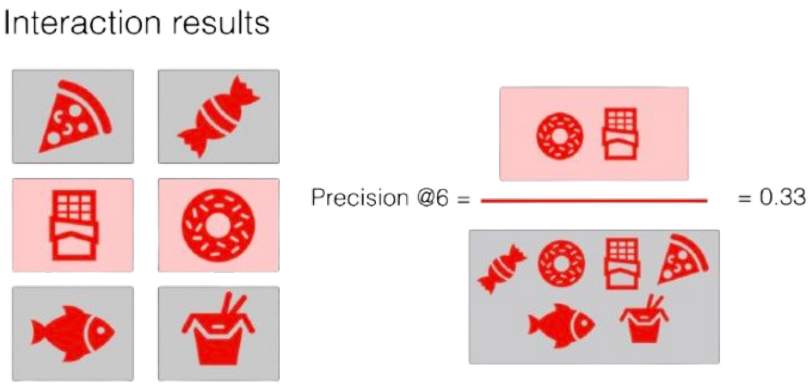
- Finding 9/10 key documents means 90% recall
- Missing even one critical document could be problematic
- Higher K values increase chances of finding everything
- Need to balance comprehensiveness with efficiency

**Real-World Example**:

Patent search for "electric vehicle batteries", results at K=5:

1. Core patent: "Lithium Ion Battery System" (relevant)
2. Related patent: "EV Charging Methods" (relevant)
3. Background: "Battery Manufacturing" (relevant)
4. Missed relevant: "Energy Storage Systems" (not in top K)
5. Missed relevant: "Vehicle Power Management" (not in top K)

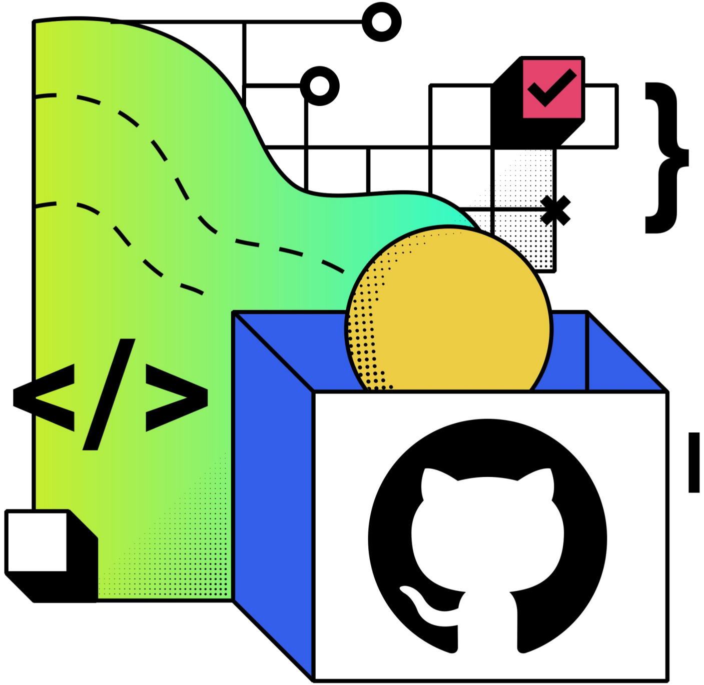
**Recall@5 = 3/5 = 0.60 (found 3 out of 5 relevant documents)**




*Accessed 15 Jan 2025, <https://www.evidentlyai.com/ranking-metrics/precision-recall-at-k>.*

# Recall@K: Finding All Relevant Information

Recall@K measures how comprehensively your system finds relevant information within a specified result set - making it crucial for medical research, patent searches, and legal documents.



GAUNTLET AI

# Sample Metric Calculators

***MRR, NDCG, and Recall@K Calculation***: *See Examples On GitHub*

The Python code demonstrates how to calculate MRR, NDCG, and Recall@K metrics, providing a practical framework that can be adapted to evaluate and improve RAG system performance in production environments.

# Contextual Embeddings

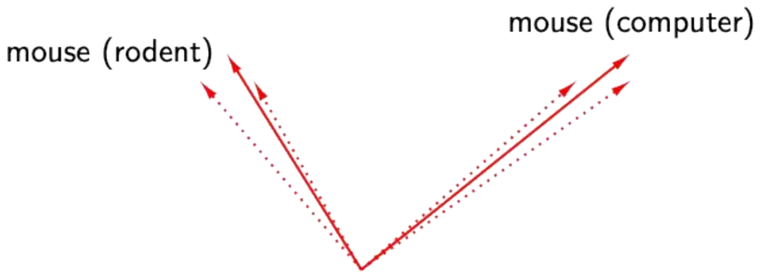
Contextual Embeddings are **meaning-aware vectors** that understand how words change in different situations:

Text → Vector: Words get encoded differently based on their surrounding context. Understanding Context – **Same word, different meanings:**

"Bank account" → [0.8, 0.2, 0.6]
*Different from*: "River bank" → [0.1, 0.7, 0.9]
"Bank turn" → [0.3, 0.9, 0.2]

**Why It Matters** – This approach enables:

- More accurate meaning captured in vectors
- Better handling of ambiguous terms
- Understanding context-dependent relationships



# Contextual Information

**Types of Context to Consider:**

- Document hierarchy (chapters, sections, subsections)
- Metadata (title, author, timestamps)
- Neighboring text (previous/next paragraphs)
- Category labels and classifications

**How to Apply Context:**

- Include hierarchy markers in the text
- Prepend metadata to chunks
- Use sliding windows for text context
- Add document relationships

**Impact on Search Quality:**

- Improved domain-specific matching
- Better handling of ambiguous terms
- More accurate relevance ranking
- Reduced false positives

**Note: Balance context amount with computational cost - more context isn't always better*

# What Is Summary Lookup?

Think of summary lookup like a book's table of contents that helps you quickly find the right section before reading the details:

How It Works – Building layers of information:
**Document** → **Chapter Summary** → **Section Summary** → **Paragraph**

Example: Technical Manual

Level 1: "API Documentation"
Level 2: "Authentication Methods"
Level 3: "OAuth Implementation Steps"

**Why It Matters** – This approach enables faster initial filtering of relevant content, reduced context window loading, and more efficient use of token limits.

# Summary Lookup In Action

***Summary Lookup RAG Using Python***: *See Repo On GitHub*

Here's a practical example implementation of summary lookup using Python, integrating Pinecone for vector storage and LangChain with LangSmith for monitoring, demonstrating how to build efficient hierarchical document retrieval.