CS 589: Text Mining and Information Retrieval

Lecture 3 - Complete Study Guide

IR Evaluation & (Pseudo)-Relevance Feedback

Course: CS 589 - Text Mining and Information Retrieval
Institution: Stevens Institute of Technology
Lecture: 3 - IR Evaluation & Relevance Feedback
Topics: Precision/Recall, MAP, MRR, NDCG, TREC, Pooling, A/B Testing, Rocchio Feedback
Prepared for: Midterm Examination

📖 Study Guide Overview

What's in This Guide:

This comprehensive study guide covers all material from Lecture 3 of CS 589, focusing on Information Retrieval evaluation metrics and relevance feedback techniques. The guide includes:

✅ **Complete Topic Coverage**

- All evaluation metrics with worked examples
- Evaluation methodologies (TREC, Pooling, Online)
- Relevance feedback techniques (Rocchio, Pseudo-relevance)

✅ **Exam-Focused Content**

- Step-by-step calculation examples
- Common pitfalls and how to avoid them
- Practice problems with detailed solutions
- Formula quick reference sheet

✅ **Learning Resources**

- Video tutorials (with time stamps)
- Python code examples
- Interactive demos
- Additional reading materials

✅ **Exam Preparation**

- Pre-exam checklist

- Time management strategies
- Last-minute review guide

How to Use This Guide:

1. 📚 **First Pass:** Read through sections 1-4 to understand all concepts
2. ✍️ **Practice:** Complete the practice problems (section 5) by hand
3. 🎥 **Reinforce:** Watch recommended videos for difficult topics
4. 📝 **Review:** Use the quick reference sheet and checklist before exam
5. ⏰ **Final Review:** Follow the "30 Minutes Before Exam" guide

Estimated Study Time: 4-6 hours for complete mastery

📚 Table of Contents

Quick Navigation

Section	Topic	Page Reference
1	Quick Review: Lecture 2 Concepts	RSJ, BM25, Language Models
2	Evaluation Metrics	Precision, Recall, MAP, MRR, NDCG
3	Evaluation Methodologies	TREC, Pooling, A/B Testing
4	Relevance Feedback	Rocchio, Pseudo-relevance
5	Practice Problems	Worked Examples
6	Additional Resources	Videos, Code, Links

Detailed Contents

1. **Quick Review: Lecture 2 Concepts**
   - RSJ Model (Robertson & Spark Jones)
   - BM25 Model
   - Language Model-Based Retrieval
2. **Evaluation Metrics (⭐ Most Important)**
   - 2.1 Precision and Recall
   - 2.2 Mean Average Precision (MAP)
   - 2.3 Mean Reciprocal Rank (MRR)
   - 2.4 Normalized Discounted Cumulative Gain (NDCG)
3. **Evaluation Methodologies**
   - 3.1 The Cranfield Experiment (1958)

- 3.2 Pooling Strategy
- 3.3 Text REtrieval Conference (TREC)
- 3.4 Online Evaluation & A/B Testing
    - Position Bias
    - User Click Logs
    - Decoy Effects
    - A/B Testing & Statistical Tests
    - Multi-Armed Bandit
    - Interleaving

4. **Relevance Feedback**
    - 4.1 Motivation
    - 4.2 Rocchio Feedback (Vector Space Model)
    - 4.3 Pseudo-Relevance Feedback
    - 4.4 Feedback Language Model
    - 4.5 Query Expansion/Reformulation
5. **Practice Problems**
    - Problem 1: Calculate MAP
    - Problem 2: Calculate NDCG@5
    - Problem 3: Calculate MRR
    - Problem 4: Feedback Language Model
    - Problem 5: Understanding IDCG
6. **Additional Resources**
    - Recommended Reading
    - Video Tutorials
    - Practice Tools & Code
    - Comparison Tables
    - Quick Reference Sheet
    - Pre-Exam Checklist

🔑 Symbol Legend

Throughout this guide, you'll see these symbols to help you navigate:

Symbol	Meaning
⭐	**Most Important** - High priority for exam
💡	**Key Concept** - Important insight or principle
🎯	**Exam Tip** - Specific advice for the exam
⚠️	**Common Mistake** - Watch out for this error
📊	**Example/Calculation** - Worked example
🔗	**External Resource** - Link to additional material
✅	**Checklist Item** - Something to verify you know
📝	**Practice Problem** - Try this yourself

📚 PART 1: QUICK REVIEW

---

1. Quick Review: Lecture 2 Concepts

> **Purpose:** Refresh your memory on prerequisite concepts from Lecture 2 that are essential for understanding evaluation metrics.

Key Models You Should Know:

RSJ Model (Robertson & Spark Jones)

- Ranks by probability of relevance: p(rel=1|q,d)
- Uses Bayes rule but doesn't leverage TF information
- Relies on relevance judgments

BM25 Model

- Approximates 2-Poisson model
- Formula:

```
c_i^BM25(tf_i) ≈ log(N/df_i) × [tf_i(k_1 + 1)] / [k_1(1 - b + b|d|/avgdl) + tf_i]
```

- Parameters: b = 0.75, k_1 ∈ [1.2, 2.0]

Language Model-Based Retrieval

- Ranks by probability of generating query from document: p(q|d)

- Dirichlet smoothing formula provided in slides

⭐ PART 2: EVALUATION METRICS (MOST IMPORTANT)

---

2. Evaluation Metrics

> ⚠️ **Critical for Exam:** This section contains the most heavily tested material. Practice all calculations by hand!

2.1 Precision and Recall

Definitions:

- **Precision** = (# relevant AND retrieved) / (# retrieved)
- **Recall** = (# relevant AND retrieved) / (# relevant)

Example from Lecture: Query: "cs 589 stevens"

- Top 3 results: 2 relevant, 1 not relevant
- **Precision@3 = 2/3**

💡 **Key Point:** Precision measures accuracy of what you returned; Recall measures completeness of retrieval.

2.2 Mean Average Precision (MAP)

Step-by-Step Calculation:

1. **Find positions of relevant documents** in your ranking (K_1, K_2, ... K_R)
2. **Calculate Precision@K** for each relevant position
3. **Average these precision values**
4. **Divide by total # of relevant documents** (NOT # retrieved)

Formula:

```
AveP = Σ(P(k) × rel(k)) / (number of relevant documents)
```

where the sum is over all retrieved documents n

Worked Example from Slides:

Ranking #1: + + - - + - - + - -

- Relevant docs at positions: 1, 2, 5, 8

- P@1 = 1/1 = 1.0
- P@2 = 2/2 = 1.0
- P@5 = 3/5 = 0.6
- P@8 = 4/8 = 0.5
- **Average Precision = (1.0 + 1.0 + 0.6 + 0.5) / 5 = 0.62**
  - Note: Divided by 5 because there are 5 total relevant documents

Ranking #2: - + - - + - + - - -

- Relevant docs at positions: 2, 5, 7
- P@2 = 1/2 = 0.5
- P@5 = 2/5 = 0.4
- P@7 = 3/7 = 0.43
- **Average Precision = (0.5 + 0.4 + 0.43) / 5 = 0.266**

Mean Average Precision = (0.62 + 0.266) / 2 = 0.443

🎯 **Common Mistake:** Don't divide by number of retrieved documents! Always divide by total number of relevant documents.

2.3 Mean Reciprocal Rank (MRR)

Use Case: When users only need ONE relevant document (e.g., navigational queries)

Formula:

```
RR = 1 / (rank of first relevant document)
MRR = average of RR across all queries
```

Example from Slides:

- Ranking #1: First relevant at position 1 → RR = 1/1 = 1.0
- Ranking #2: First relevant at position 2 → RR = 1/2 = 0.5
- **MRR = (1.0 + 0.5) / 2 = 0.75**

⚠️ **Note:** In the slides, they use `RR = 1.0 / (1.0 + rank_1)` where rank starts from 0, so adjust based on indexing convention.

2.4 Normalized Discounted Cumulative Gain (NDCG)

Use Case: When you have **multiple levels of relevance** (not just binary)

- Example: 2 = click > 10s, 1 = click < 10s, 0 = no click

Step 1: Calculate DCG

Two formulas (use the one your professor prefers):

Formula 1:

```
DCG_p = Σ(rel_i / log₂(i+1)) for i=1 to p
```

Formula 2:

```
DCG_p = Σ((2^rel_i - 1) / log₂(i+1)) for i=1 to p
```

Step 2: Calculate IDCG (Ideal DCG)

- Sort relevance scores in descending order
- Calculate DCG for this ideal ranking

Step 3: Calculate NDCG

```
NDCG = DCG / IDCG
```

Worked Example from Slides:

Ranking #1: [2, 0, 1, 2, 2, 1, 0, 0, 0, 2]

DCG@4 using Formula 1:

```
= 2/log₂(2) + 0/log₂(3) + 1/log₂(4) + 2/log₂(5)
= 2/1 + 0 + 1/2 + 2/2.32
= 2 + 0 + 0.5 + 0.86
= 3.3613
```

IDCG@4: Ideal ranking = [2, 2, 2, 2]

```
= 2/log₂(2) + 2/log₂(3) + 2/log₂(4) + 2/log₂(5)
= 2 + 1.26 + 1 + 0.86
= 5.1232
```

NDCG@4 = 3.3613 / 5.1232 = 0.656

Important: If # relevant items < k, IDCG calculation stops at the number of relevant items!

🔬 PART 3: EVALUATION METHODOLOGIES

---

3. Evaluation Methodologies

> **Focus:** Understanding how IR systems are evaluated in practice - from lab experiments to real-world A/B testing.

3.1 The Cranfield Experiment (1958)

Basic Ingredients:

1. A corpus of documents (~1.4k paper abstracts)
2. A set of queries (225 queries)
3. Binary relevance judgments for each (query, document) pair
4. **Reusable** relevance judgments

Key Innovation: Created a standardized test collection for IR evaluation

---

3.2 Pooling Strategy

Problem: Too many (query, document) pairs to annotate

- 225 queries × 1,400 docs = 315,000 pairs!

Solution:

1. Run K different systems
2. For each system, take top 100 results
3. **Pool (union)** all these documents
4. Only annotate documents in this pool

Why it works: Relevant documents likely appear in top results of at least one system

🎯 **Exam Tip:** Understand that pooling reduces annotation cost while maintaining quality

---

3.3 Text REtrieval Conference (TREC)

Key Facts:

- Since 1992, hosted by NIST
- Uses Cranfield methodology with pooling
- Relevance judgments based on human annotations
- Goes beyond keyword matching
- Different tracks: Web, Question Answering, Microblog, etc.

Sample TREC Query Format:

```xml
<top>
<num> Number: 794
<title> pet therapy
<desc> Description:
How are pets or animals used in therapy for humans and what are the benefits?
<narr> Narrative:
Relevant documents must include details of how pet- or animal-assisted
therapy is or has been used...
</top>
```

3.4 Online Evaluation & A/B Testing

Why Online Evaluation?

- TREC-style: Explicit, difficult to scale, gets outdated
- Click logs: Implicit, large-scale, up-to-date

Key Concepts:

Position Bias

- Higher positions get more attention
- Same item gets fewer clicks in lower positions
- **Models:**
  - Baseline: No position bias
  - Mixture: Relevance + constant bias
  - **Cascade**: Linear traversal; documents below clicked result not examined (best for lower ranks)

Cascade Model Formula:

```
c_di = r_d ∏(1 - r_docinrank:j) for j=1 to i-1
```

User Click Logs

System logs track:

- Timestamp
- Session ID
- Query ID and content

- Items viewed (sequential order)
- Click/no-click for each item
- User demographics, history, location, device
- Dwell time, browsing time
- Eye tracking information

Click Logs Storage:

- Stored in large tables
- Extract subsets using SQL queries

Decoy Effects

User preferences change based on context:

Example:

- Option A ($400, 20G) vs Option B ($500, 30G): 50-50 split
- Add decoy ($550, 20G): Now Option B gets 60% preference

Implication: Can't assume relevance is fixed; it's context-dependent

---

A/B Testing

Process:

1. Split traffic: 50% System A, 50% System B
2. Compare metrics (clicks, retention, revenue)
3. Use statistical significance tests

Statistical Tests:

Sign Test:

- Counts how many queries System A beats System B
- Simple but ignores magnitude
- Python: `statsmodels.stats.descriptivestats.sign_test`

Wilcoxon Test:

- Considers both direction AND magnitude
- Formula: $W = \Sigma[\text{sgn}(x_{2,i} - x_{1,i}) \cdot R_i]$
- More powerful than sign test

- Python: `scipy.stats.wilcoxon`

p-value Interpretation:

- $p < 0.05$: Statistically significant (reject null hypothesis)
- $p \geq 0.05$: Not significant (can't conclude difference)

Multi-Armed Bandit

Problem with A/B Testing:

- Fixed 50/50 split means poor system hurts user experience

Solution:

- Dynamically adjust traffic based on performance
- Explore vs Exploit tradeoff

Upper Confidence Bound Formula:

```
μ̂_i(t-1) + √(2 log t / T_i(t-1))
```

Where:

- μ̂_i(t-1): Estimated mean reward for arm i
- T_i(t-1): Number of times arm i has been pulled
- t: Current time step

Result: Better system gets more traffic over time while still exploring alternatives

---

Interleaving

Method:

1. Merge results from System A and System B
2. Remove duplicates
3. Show combined ranking to user
4. Track which system's documents get clicked

Example:

```
System A: [Kernel machines, Kernel machines, SVMs, SVM-light, ...]
System B: [Kernel machines, SVMs, Intro to SVMs, Lucent SVM demo, ...]

Interleaved (duplicates removed):
[Kernel machines (A), Kernel machines (B), SVMs (A), SVM-light (A),
 Intro to SVMs (B), Lucent SVM demo (B), ...]

If clicks: Kernel machines (A), SVMs (A), SVM-light (B), Intro to SVMs (B)
→ A clicks = 3, B clicks = 1
```

Advantages:

- More sensitive than A/B testing
- Same user sees both systems
- Controls for user variability

🔄 PART 4: RELEVANCE FEEDBACK

---

4. Relevance Feedback

> **Key Idea:** Using user interactions to improve search results through query refinement.

4.1 Motivation

Problem: Users don't always know how to express their information need

Example: Query "best phone"

- Does user prefer lower-priced or high-end?
- Larger storage or better camera?

Solution: Learn from user interactions (clicks, dwell time) to refine the query

---

4.2 Rocchio Feedback (Vector Space Model)

Formula:

```
q_F = αq + (β/|D_r|)Σd_r - (γ/|D_n|)Σd_n
```

Where:

- q: Original query vector

- q_F: Feedback-adjusted query
- D_r: Set of relevant documents
- D_n: Set of non-relevant documents
- α: Weight for original query
- β: Weight for relevant docs (typically $\beta >> \gamma$)
- γ: Weight for non-relevant docs

Intuition:

- Moves query vector toward relevant documents
- Moves query vector away from non-relevant documents
- Positive evidence weighted more than negative

Visual Representation:

```
Before feedback: q (original query)
After feedback: q_F (moved toward relevant docs cluster)
```

Practical Issues:

1. **Large vocabularies** - Only consider important words
2. **Requires explicit feedback** - User must label docs
3. **Robust and effective** when feedback available

🔗 **Interactive Demo:** https://tinyurl.com/4bkw7cj2

4.3 Pseudo-Relevance Feedback

Problem: What if we don't have relevance judgments?

Solution: Assume top-k retrieved documents are relevant

Process:

1. Run initial query
2. Retrieve top k documents (e.g., k=10)
3. Treat these as "pseudo-relevant"
4. Apply feedback algorithm
5. Re-rank with new query

Why It Works:

Query: "fish tank"

Top results contain: "aquarium", "filter", "water", "gravel", "fish"

Expand query: "fish tank aquarium filter water"

Result: Better recall, finds documents using different terminology

For Vector Space Model:

- Use Rocchio with top-k docs as D_r
- D_n can be empty or lower-ranked docs

For Language Model:

$$\theta_q^{FB} = \lambda\theta_q + (1-\lambda)\theta_d$$

Where θ_d is estimated from feedback documents

4.4 Feedback Language Model

Formula:

$$\theta^{FB} = \lambda\theta_q + (1-\lambda)\theta_d$$

Components:

- θ_q: Query language model
- θ_d: Document language model (from relevant docs)
- λ: Interpolation parameter (controls mixing)

Process:

1. Get initial query model θ_q
2. Retrieve documents
3. Estimate θ_d from top documents
4. Combine using mixture model
5. Re-rank using θ^{FB}

Estimating θ_d:

- Can use EM algorithm
- Accounts for query-specific vs general terms

Quiz Question from Lecture 3:

Given feedback documents, calculate θ^FB:

```
θ^FB = λθ_q + (1-λ)θ_d
```

If $\lambda = 0.7$, this means:

- 70% weight to original query
- 30% weight to feedback documents

Result (from quiz): Probability of "airport security" = **0.38**

4.5 Query Expansion/Reformulation

Techniques:

1. Manual Thesaurus:

- Use WordNet or domain-specific thesaurus
- Example: "skin itch" → add "pruritus", "integumentary system"

2. Automatic Thesaurus:

- Learn word similarities from corpus
- Example nearest neighbors:
  - "absolutely" → "absurd", "whatsoever", "totally"
  - "captivating" → "shimmer", "stunningly", "superbly"

3. Query Log Mining:

- Analyze what users searched and clicked
- Example: Users who search "movie tickets" also search "showtimes"

Google's Query Expansion:

- "what is the most..." → autocomplete suggestions
- "yoga mat" → filter by price, brand

📝 PART 5: PRACTICE PROBLEMS

---

5. Practice Problems

> **Instructions:** Try solving these problems by hand before revealing the solutions. These mirror the quiz and exam format.

Problem 1: Calculate MAP

Given two systems with rankings for 2 queries (5 relevant docs total each):

Query 1:

- System A: `+ + - + - + - - - +` (+ = relevant)
- System B: `+ - + - + - - + - +`

Calculate MAP for each system.

<details> <summary>Click to see solution</summary>

System A (Query 1):

- Relevant at positions: 1, 2, 4, 6, 10
- P@1 = 1/1 = 1.0
- P@2 = 2/2 = 1.0
- P@4 = 3/4 = 0.75
- P@6 = 4/6 = 0.67
- P@10 = 5/10 = 0.5
- **AP = (1.0 + 1.0 + 0.75 + 0.67 + 0.5) / 5 = 0.784**

System B (Query 1):

- Relevant at positions: 1, 3, 5, 8, 10
- P@1 = 1/1 = 1.0
- P@3 = 2/3 = 0.67
- P@5 = 3/5 = 0.6
- P@8 = 4/8 = 0.5
- P@10 = 5/10 = 0.5
- **AP = (1.0 + 0.67 + 0.6 + 0.5 + 0.5) / 5 = 0.654**

For full MAP, average across both queries.

</details>

Problem 2: Calculate NDCG@5

Ranking: [3, 2, 0, 1, 2]

Relevance scale: 0-3

<details> <summary>Click to see solution</summary>

Using Formula 1:

DCG@5:

```
= 3/log₂(2) + 2/log₂(3) + 0/log₂(4) + 1/log₂(5) + 2/log₂(6)
= 3/1 + 2/1.585 + 0/2 + 1/2.322 + 2/2.585
= 3 + 1.262 + 0 + 0.431 + 0.774
= 5.467
```

IDCG@5: Ideal = [3, 2, 2, 1, 0]

```
= 3/1 + 2/1.585 + 2/2 + 1/2.322 + 0
= 3 + 1.262 + 1 + 0.431 + 0
= 5.693
```

NDCG@5 = 5.467 / 5.693 = 0.96

</details>

Problem 3: Calculate MRR

3 queries:

- Query 1: First relevant at position 1
- Query 2: First relevant at position 3
- Query 3: First relevant at position 2

<details> <summary>Click to see solution</summary> ``` $RR_1 = 1/1 = 1.0$ $RR_2 = 1/3 = 0.333$ $RR_3 = 1/2 = 0.5$

MRR = (1.0 + 0.333 + 0.5) / 3 = 0.611

</details>

Problem 4: Feedback Language Model (From Quiz)

Scenario:
- Query: "airport security"
- $\lambda = 0.7$
- θ_q: probability of "airport" in query = 0.5
- θ_d: probability of "airport" in feedback docs = 0.2

Calculate θ^{FB} for "airport":

<details>
<summary>Click to see solution</summary>

$\theta^{FB} = \lambda\theta_q + (1-\lambda)\theta_d$

$= 0.7 \times 0.5 + 0.3 \times 0.2$

$= 0.35 + 0.06$

$= 0.41$

Answer from quiz: 0.38 (may use different values or formula variant)

</details>

Problem 5: Understanding IDCG with Fewer Relevant Items

Question: If you have only 3 relevant items but calculate NDCG@5, how do you compute IDCG?

<details>
<summary>Click to see solution</summary>

Scenario: Relevant items have scores [2, 1, 1]

IDCG@5 calculation:
- Ideal ranking: [2, 1, 1, 0, 0]
- Only compute up to position 3 (last relevant item):

$IDCG@5 = 2/\log_2(2) + 1/\log_2(3) + 1/\log_2(4)$

$= 2/1 + 1/1.585 + 1/2$

$= 2 + 0.631 + 0.5$

$= 3.131$

Note: Positions 4 and 5 contribute 0, so can be ignored.

Impact: nDCG value will be lower if you don't retrieve all relevant items in top-k.

</details>

🔗 PART 6: ADDITIONAL RESOURCES

---

6. Additional Resources

> **Enhance Your Learning:** Curated videos, readings, and tools to deepen your understanding.

📖 **Recommended Reading**

1. **Stanford IR Book** (Your professor's reference)
   - Chapter 8: Evaluation in Information Retrieval
   - https://nlp.stanford.edu/IR-book/
   - Sections 8.1-8.4 (TREC, Precision/Recall)
   - Section 8.7 (Results Snippets)

2. **Manning, Raghavan, Schütze - Introduction to Information Retrieval**
   - Chapter 9: Relevance Feedback
   - Free online: https://nlp.stanford.edu/IR-book/pdf/09expand.pdf

3. **Research Papers:**
   - Järvelin & Kekäläinen (2002): "Cumulated gain-based evaluation of IR techniques" (NDCG)
   - Craswell et al. (2008): "An experimental comparison of click position-bias models"

🎥 **Video Tutorials**

Evaluation Metrics

1. **Precision & Recall (StatQuest):**
   - https://www.youtube.com/watch?v=vP06aMoz4v8
   - Duration: 15 minutes
   - Great visual explanations

2. **MAP Explained:**
   - https://www.youtube.com/watch?v=vYVSVhYNX7E
   - Duration: 10 minutes
   - Step-by-step calculation

3. **NDCG Intuition:**
   - https://www.youtube.com/watch?v=7L76BaqOYCI
   - Duration: 12 minutes
   - Why DCG uses logarithmic discount

4. **F1 Score (related to Precision/Recall):**
   - https://www.youtube.com/watch?v=jJ7ff7Gcq34
   - Duration: 8 minutes

Online Evaluation

5. **A/B Testing Basics:**
   - https://www.youtube.com/watch?v=zFMgpxG-chM
   - Duration: 20 minutes
   - Statistical testing fundamentals

6. **Multi-Armed Bandit:**
   - https://www.youtube.com/watch?v=e3L4VocZnnQ
   - Duration: 18 minutes
   - Exploration vs exploitation

Mathematical Background

7. **Logarithms Review (Khan Academy):**
   - https://www.khanacademy.org/math/algebra2/x2ec2f6f830c9fb89:logs
   - Useful for understanding DCG discount factor

8. **Statistical Significance:**
   - https://www.youtube.com/watch?v=5Z9OIYA8He8
   - Duration: 15 minutes
   - p-values and hypothesis testing

🛠️ **Practice Tools & Code**

Python Libraries
```python
# Precision, Recall, F1
from sklearn.metrics import precision_score, recall_score, f1_score

# Statistical tests
```

```
from scipy.stats import wilcoxon
from statsmodels.stats.descriptivestats import sign_test

# Calculate NDCG
from sklearn.metrics import ndcg_score
```

Example Code: Calculate MAP

```python
def average_precision(relevant_positions, total_docs):
    """
    Calculate average precision for a single query.

    Args:
        relevant_positions: List of positions (1-indexed) where relevant docs appear
        total_docs: Total number of relevant documents

    Returns:
        Average precision score
    """
    if not relevant_positions or total_docs == 0:
        return 0.0

    precisions = []
    for i, pos in enumerate(sorted(relevant_positions)):
        precision_at_k = (i + 1) / pos
        precisions.append(precision_at_k)

    return sum(precisions) / total_docs

# Example
relevant_pos = [1, 2, 5, 8]  # relevant docs at positions 1, 2, 5, 8
total_relevant = 5  # 5 relevant docs total
ap = average_precision(relevant_pos, total_relevant)
print(f"Average Precision: {ap:.3f}")
```

Example Code: Calculate NDCG

```python
```

```python
import numpy as np

def dcg_at_k(relevances, k):
    """Calculate DCG@k"""
    relevances = np.asarray(relevances)[:k]
    if relevances.size:
        # Formula: rel_i / log2(i+2) for i starting at 0
        return np.sum(relevances / np.log2(np.arange(2, relevances.size + 2)))
    return 0.0

def ndcg_at_k(relevances, k):
    """Calculate NDCG@k"""
    dcg = dcg_at_k(relevances, k)
    idcg = dcg_at_k(sorted(relevances, reverse=True), k)
    if idcg == 0:
        return 0.0
    return dcg / idcg

# Example
ranking = [3, 2, 0, 1, 2]
k = 5
ndcg = ndcg_at_k(ranking, k)
print(f"NDCG@{k}: {ndcg:.3f}")
```

Interactive Tools

1. **Rocchio Feedback Demo:**
    - https://tinyurl.com/4bkw7cj2
    - Visualize query refinement
2. **IR Metrics Calculator:**
    - Create your own spreadsheet to practice
    - Or use online calculators (search "MAP calculator IR")
3. **Python Notebook Examples:**
    - Google Colab notebook for IR metrics
    - Practice with your own data

📊 Comparison Tables

When to Use Which Metric?

Metric	Use When	Pros	Cons
Precision@k	Fixed cutoff important	Simple, interpretable	Ignores rank within top-k
Recall@k	Need to find all relevant	Measures completeness	Doesn't penalize bad ranking
MAP	Multiple relevant docs, order matters	Emphasizes top ranks	Binary relevance only
MRR	Only need 1 relevant doc	Good for navigational queries	Ignores other relevant docs
NDCG	Graded relevance, order matters	Handles multiple relevance levels	More complex to calculate

Evaluation Methodology Comparison

Method	Pros	Cons	Best For
TREC-style (Pooling)	Reusable, controlled	Expensive, gets outdated	Research, benchmarking
Click logs	Large-scale, up-to-date	Biased, noisy	Industry, real-world
A/B testing	Real user feedback	Needs traffic, statistical power	Production systems
Interleaving	More sensitive than A/B	More complex	Fine-tuning systems

---

📝 Quick Reference Sheet

> 💡 **Print This Page:** This section is designed to be printed and used as a quick reference during your final review.

Formulas Summary

```
Precision@k = (# relevant in top k) / k

Recall@k = (# relevant in top k) / (total # relevant)

F1@k = 2 × (Precision@k × Recall@k) / (Precision@k + Recall@k)

Average Precision = Σ(P(k) × rel(k)) / (# relevant docs)
                    where k ranges over all positions

MAP = (1/|Q|) Σ AveP(q) for all queries Q

MRR = (1/|Q|) Σ (1 / rank of first relevant doc)

DCG_p = Σ (rel_i / log_2(i+1))  [i from 1 to p]

Alternative DCG_p = Σ ((2^rel_i - 1) / log_2(i+1))

NDCG_p = DCG_p / IDCG_p

Rocchio: q_F = αq + (β/|D_r|)Σd_r - (γ/|D_n|)Σd_n

Feedback LM: θ^FB = λθ_q + (1-λ)θ_d

Upper Confidence Bound: μ̂_i(t-1) + √(2 log t / T_i(t-1))
```

Common Parameter Values

```
Rocchio: α = 1.0, β = 0.75, γ = 0.15
BM25: b = 0.75, k_1 ∈ [1.2, 2.0]
Feedback LM: λ ∈ [0.5, 0.9]
Pseudo-relevance: k ∈ [5, 20] documents
```

✅ PRE-EXAM CHECKLIST & FINAL REVIEW

✅ Pre-Exam Checklist

> **Use This:** Go through this checklist 24 hours before your exam to identify any gaps in your knowledge.

Calculation Skills

- [ ] Can calculate Precision@k and Recall@k by hand
- [ ] Can calculate Average Precision for a single query (remember: divide by total # relevant!)

- [ ] Can calculate MAP across multiple queries
- [ ] Can calculate MRR for navigational queries
- [ ] Can calculate DCG@k using the logarithmic discount formula
- [ ] Can determine IDCG@k (remember: sort by relevance descending!)
- [ ] Can calculate NDCG@k from DCG and IDCG
- [ ] Can apply Rocchio feedback formula

Conceptual Understanding

- [ ] Understand when to use each evaluation metric
- [ ] Know the difference between Precision and Recall
- [ ] Can explain why MAP divides by # relevant, not # retrieved
- [ ] Understand why NDCG uses logarithmic discounting
- [ ] Can explain the Cranfield experiment methodology
- [ ] Understand how pooling reduces annotation cost
- [ ] Know the three types of position bias models (Baseline, Mixture, Cascade)
- [ ] Can explain the difference between A/B testing and interleaving
- [ ] Understand how Rocchio feedback works (move toward relevant, away from non-relevant)
- [ ] Can explain pseudo-relevance feedback and its risks

Common Pitfalls to Avoid

- [ ] **MAP:** Don't divide by # retrieved documents - always use total # relevant
- [ ] **NDCG:** Don't forget the logarithmic discount - it's $\log_2(i+1)$, not just (i+1)
- [ ] **IDCG:** When # relevant < k, IDCG only sums up to # relevant items
- [ ] **MRR:** Only considers the FIRST relevant document
- [ ] **Indexing:** Check if positions start at 0 or 1 (be consistent!)
- [ ] **Rocchio:** β should be >> γ (positive evidence weighted more)
- [ ] **Pseudo-relevance:** Can make things worse if initial results are bad

Exam Strategy

1. **Read problems carefully** - Check if positions are 0-indexed or 1-indexed
2. **Show your work** - Partial credit for correct process
3. **Double-check denominators** - MAP uses # relevant, not # retrieved
4. **Verify your IDCG** - Should be ≥ DCG always
5. **Watch for special cases** - What if no relevant docs? What if k > # docs?

🎯 Final Tips for Success

Study Strategy

1. **Practice calculations by hand** first

- Don't rely only on code/calculators
- Work through examples step-by-step
- The quiz solutions are excellent practice

2. **Understand the "why"**
   - Why does NDCG use log discount? (Users scan top results more carefully)
   - Why $\beta > \gamma$ in Rocchio? (Positive evidence more reliable)
   - Why does pooling work? (Good systems will find relevant docs)
3. **Review quiz solutions thoroughly**
   - Quiz closely reflects exam format
   - Pay attention to edge cases
   - Note the exact formulas used
4. **Create your own examples**
   - Make up small ranking examples
   - Calculate metrics by hand
   - Verify with Python code

Time Management

- **Calculation problems:** Budget ~10 min each
- **Conceptual questions:** ~5 min each
- **Leave 10-15 min** for review

During the Exam

✅ **Do:**

- Write down formulas first
- Show all calculation steps
- Box or highlight final answers
- Check units and ranges (e.g., NDCG ∈ [0,1])

❌ **Don't:**

- Rush through calculations
- Forget to label your answers
- Skip showing work (even if answer is wrong, process matters!)

🔗 External Links Summary

Recommended Videos:

- Precision/Recall: https://www.youtube.com/watch?v=vP06aMoz4v8
- MAP: https://www.youtube.com/watch?v=vYVSVhYNX7E
- NDCG: https://www.youtube.com/watch?v=7L76BaqOYCI
- A/B Testing: https://www.youtube.com/watch?v=zFMgpxG-chM
- Multi-Armed Bandit: https://www.youtube.com/watch?v=e3L4VocZnnQ

Reading:

- Stanford IR Book: https://nlp.stanford.edu/IR-book/
- Interactive Rocchio Demo: https://tinyurl.com/4bkw7cj2

Tools:

- Python sklearn: `from sklearn.metrics import ndcg_score, precision_score`
- SciPy: `from scipy.stats import wilcoxon`

⏰ Last-Minute Review (30 Minutes Before Exam)

> 🚨 **Emergency Prep:** If you only have 30 minutes, focus on this section!

Quick Formulas Review (5 min)

```
P@k = relevant in top k / k
R@k = relevant in top k / total relevant
AP = Σ(P(k)×rel(k)) / total_relevant
DCG = Σ(rel_i / log₂(i+1))
NDCG = DCG / IDCG
```

Practice One Problem Each (15 min)

1. Calculate MAP for one ranking
2. Calculate NDCG@5 for one ranking
3. Apply Rocchio formula once

Concept Review (10 min)

- Pooling: Take top-k from multiple systems, annotate only pool
- Pseudo-relevance: Assume top-k are relevant, expand query

- Position bias: Higher positions get more attention
- Interleaving: Mix two systems' results, compare clicks

💪 You've Got This!

What You've Mastered:

- ✅ All major evaluation metrics (P/R, MAP, MRR, NDCG)
- ✅ Evaluation methodologies (TREC, pooling, online)
- ✅ Statistical testing (Sign, Wilcoxon)
- ✅ Relevance feedback (Rocchio, pseudo-relevance, LM)
- ✅ Practical considerations (position bias, A/B testing, bandits)

Remember:

- 📚 The quiz solutions match the exam format closely
- ✍️ If you can solve those problems by hand, you're ready!
- 🎯 Focus on understanding WHY, not just memorizing formulas
- ⏰ Manage your time during the exam
- 💡 Show your work for partial credit

Final Confidence Boosters:

1. You have comprehensive notes covering all testable material
2. You have worked examples for every type of problem
3. You have a formula reference sheet ready
4. You know the common mistakes to avoid
5. You have a clear exam strategy

Good luck on your midterm! 🍀 📚

You've prepared well. Trust your preparation and stay calm during the exam!

📄 Document Information

Document Details:

Title: CS 589 Lecture 3 - Complete Study Guide
Course: CS 589 - Text Mining and Information Retrieval
Institution: Stevens Institute of Technology
Topics Covered:

- Evaluation Metrics (Precision, Recall, MAP, MRR, NDCG)
- Evaluation Methodologies (Cranfield, TREC, Pooling)
- Online Evaluation (A/B Testing, Interleaving, Multi-Armed Bandits)
- Relevance Feedback (Rocchio, Pseudo-relevance, Language Models)

Exam Preparation Materials:

- ✅ Worked examples from lecture slides
- ✅ Quiz solutions with detailed explanations
- ✅ Practice problems with step-by-step solutions
- ✅ Formula quick reference sheet
- ✅ Pre-exam checklist

Additional Resources:

- Video tutorials with timestamps
- Python code examples
- Interactive demos
- External reading materials

📧 Questions or Found an Error?

If you spot any errors or have questions about the material, please:

1. Review the lecture slides at the referenced page numbers
2. Check the quiz solutions document
3. Consult the Stanford IR textbook (Chapter 8)
4. Ask your professor or TA during office hours

Good luck with your studies! 🎓