Exam Eligibility Assignment

Finish test

102 hr 43 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - **Q8 (0.5 Points)** 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q8 (0.5 Points) 0.5 point Answered

Which of the following is a key advantage of using crowdsourcing for relevance judgments in Web Information Retrieval (WIR)?

- Greater use of advanced IR evaluation metrics
- Higher accuracy than expert judgments
- More consistent labeling across all topics
- Lower cost and faster data collection

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 43 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - **Q7 (0.5 Points)** 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q7 (0.5 Points) 0.5 point Answered

What determines the size of the Term-Document (TD) matrix in an Information Retrieval system?

- Number of documents × number of queries
- Number of users × number of documents
- Number of unique terms × number of documents
- Number of relevant documents × number of terms

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 44 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - **Q6 (0.5 Points)** 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q6 (0.5 Points) 0.5 point Answered

In language models for IR, what does the collection model P(t | C) represent?

- The probability of a term occurring in a specific document
- The likelihood of a document being relevant to a query
- The frequency of a term in a query
- **The probability of a term occurring across the entire collection**

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 44 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - **Q5 (0.5 Points)** 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q5 (0.5 Points) 0.5 point Answered

Which technique allows flexible matching like retrieving both “naïve” and “naive”?

- Boolean retrieval
- Positional index
- Biword indexing
- Wildcard search

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 44 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - **Q4 (0.5 Points)** 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q4 (0.5 Points) 0.5 point Answered

What is the primary role of a web crawler's frontier in web search?

- (●) To store the list of URLs yet to be crawled
- ( ) To filter out spam pages during indexing
- ( ) To process user queries in real-time
- ( ) To rank web pages based on their relevance

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 44 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - **Q3 (0.5 Points)** 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q3 (0.5 Points) 0.5 point Answered

Which modern IR development focuses on generating answers directly from document corpora?

- ○ Document Clustering
- ○ Link Analysis
- ○ Automated Indexing
- ◉ Generative IR

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 45 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q2 (0.5 Points) 0.5 point Answered

Which component of an IR system is responsible for storing term-document relationships?

- Ranking Algorithm
- Inverted Index
- Web Crawler
- Query Processor

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 45 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
   - 1.1 True/False Questions (15 Points) 1
   - 1.2 Multiple Choice Questions (5 Points)
     - **Q1 (0.5 Points)** 1
     - Q2 (0.5 Points) 1
     - Q3 (0.5 Points) 1
     - Q4 (0.5 Points) 1
     - Q5 (0.5 Points) 1
     - Q6 (0.5 Points) 1
     - Q7 (0.5 Points) 1
     - Q8 (0.5 Points) 1
     - Q9 (0.5 Points) 1
     - Q10 (0.5 Points) 1
2. Practical Tasks (50 Points)
   - 2.1 Vector-Based Ranking (22 Points)
     - 2.1.1 TF-IDF (8.8 Points) 1
     - 2.1.2 Cosine Similarity Scores (12 Points) 1
     - 2.1.3 Document Ranking (1.2 Points) 1
   - 2.2 Query Likelihood Scoring (18 Points)
     - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
     - 2.2.2 Laplace Smoothing (6 Points) 1
     - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
   - 2.3 Evaluation (8 Points)
     - 2.3.1 Precision@10 (2 Points) 1
     - 2.3.2 Recall@10 (2 Points) 1
     - 2.3.3 F1-score@10 (2 Points) 1
     - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
   - 2.4 Boolean Logic (2 Points) 1
3. Programming Tasks (30 Points)
   - 3.1 Document-Query Similarity (15 Points)
     - Gap with dropdown 1
   - 3.2 Rocchio Feedback (15 Points)
     - Q1 1
     - Q2 1
     - Q3 1
     - Q4 1
     - Q5 1
     - Q6 1
     - Q7 1
     - Q8 1
     - Q9 1
     - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q1 (0.5 Points)

0.5 point Answered

What is the primary role of a web crawler in an IR system?

- ○ To rank search results based on relevance
- ◉ To collect and index web pages for retrieval
- ○ To process user queries in real-time
- ○ To refine user queries for better accuracy

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 46 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

1.1 True/False Questions (15 Points)

15 points Answered

Each correct answer scores **+0.5 points**.

Each incorrect answer scores **–0.5 points**.

Unanswered questions score **0 points**.

Maximum total score: 15 points

Minimum total score: 0 points

Unanswered	Right	Wrong	
☐	☑	☐	People can browse, navigate, or query to find information in web information retrieval (WIR) systems.
☐	☑	☐	The aim of an Information Retrieval (IR) system is to retrieve items that are relevant to the user's query and present them in a ranked list.
☐	☐	☑	Database Management Systems (DBMS) primarily focus on freeform text rather than structured data.
☐	☑	☐	System-centered IR approaches focus on developing algorithms, while user-centered approaches focus on understanding user interactions.
☐	☐	☑	R-Precision uses a fixed cutoff value k regardless of the number of relevant documents.
☐	☑	☐	Users rarely go beyond the first page of search results, influencing the use of Precision at k (p@k).
☐	☐	☑	The ranking order of documents affects the precision and recall values in a Boolean retrieval system.
☐	☑	☐	Fallout measures the fraction of non-relevant documents retrieved by the system.
☐	☐	☑	Fallout is a very widely used metric for evaluating IR systems.
☐	☑	☐	Term normalisation in document indexing converts all terms to the same case, typically lowercase.
☐	☑	☐	Optimizing Boolean retrieval involves intersecting shorter posting lists first to reduce processing time.
☐	☐	☑	Biword indexes can reliably verify that a document contains a long phrase without false positives.
☐	☑	☐	Positional indexes allow for proximity queries, such as finding terms within a certain number of words of each other.
☐	☑	☐	Terms that occur too frequently in a collection are often removed as stopwords because they are less informative.
☐	☐	☑	The Boolean model provides a ranked list of documents based on their relevance to the query.
☐	☑	☐	Pseudo relevance feedback uses top-k results as positive feedback to adjust the query vector without user input.
☐	☐	☑	The sequence of query terms matters when constructing a query vector in the Vector Space Model.
☐	☑	☐	The cosine similarity measure in the Vector Space Model normalizes for document length.
☐	☐	☑	The Jaccard coefficient considers term frequency when computing query-document match scores.
☐	☐	☑	Ranked retrieval models return a set of documents that strictly satisfy a query expression.
☐	☑	☐	Boolean queries often result in either too few or too many results.
☐	☐	☑	The Boolean model is intuitive for the majority of users to express information needs.
☐	☐	☑	Language models consider only the syntax of queries and not their semantics.
☐	☑	☐	A finite state automaton can be viewed as a deterministic language model.
☐	☑	☐	The sparseness issue makes it difficult to estimate probabilities for long phrases in language models.
☐	☐	☑	Language models for IR do not normalize term frequencies by document length.
☐	☑	☐	The probability of a term in a unigram model is calculated as the number of occurrences of the term divided by the total number of word occurrences in the collection.
☐	☐	☑	Pre-trained large language models like BERT are primarily used for simple unigram-based IR tasks.
☐	☑	☐	Dirichlet smoothing incorporates the collection frequency of terms to adjust probabilities.
☐	☑	☐	The use of a global language model in smoothing is to ensure that all words have a non-zero probability of occurring.

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 40 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q10

1.5 point | Answered

What would be the most likely effect of assigning higher weights to certain relevant documents in the feedback dictionary?

- ○ The gamma value will override their influence
- ○ Cosine similarity will not be computable
- ○ These documents will be excluded from centroid calculation
- ◉ The updated query will shift more strongly toward the vocabulary of those documents

Submit answer

Go to top

Exam Eligibility Assignment

Finish test

102 hr 40 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q9

1.5 point Answered

What theoretical assumption does the Rocchio model make about the geometry of the document space?

- ○ Relevant and non-relevant documents form orthogonal subspaces
- ○ Documents follow a Gaussian distribution in each dimension
- ○ All terms are equally informative
- ◉ The ideal query lies somewhere between the centroids of relevant and non-relevant documents

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q8

1.5 point Answered

How does Rocchio relevance feedback affect **recall** in a typical retrieval task?

- ○ It always decreases recall
- ○ It is designed to optimize only precision
- ◉ It may improve recall by moving the query toward the relevant document space
- ○ It narrows the query to fewer terms, decreasing recall

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q7

1.5 point | Answered

If document d5 has high TF-IDF for term4 and the query has zero weight for term4, what effect does that have before Rocchio is applied?

- ○ d5 will be dropped from the matrix
- ○ Rocchio will automatically increase the query's weight for term4
- ◉ d5 will have low similarity due to orthogonality with the query
- ○ d5 will be ranked highest

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q6

1.5 point Answered

Why is cosine similarity preferred over Euclidean distance in this implementation?

- ○ It penalizes longer documents
- ◉ It is independent of document length and focuses on vector direction
- ○ It avoids the curse of dimensionality
- ○ It works better with binary term frequencies

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q5

1.5 point Answered

Which condition would most likely lead to a **negative term weight** in the updated query vector?

- (●) A term appears only in non-relevant documents with high TF-IDF
- ( ) The alpha parameter is set to zero
- ( ) The original query is zero
- ( ) A term is present in all documents equally

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q4

1.5 point | Answered

How does using np.average(..., weights=...) in the Rocchio function improve flexibility?

- ○ It removes the need for the alpha parameter
- ○ It converts TF-IDF to binary
- ◉ It allows for variable strength of user feedback on individual documents
- ○ It normalizes the entire document matrix

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q3

1.5 point Answered

What could happen if the gamma parameter is set too high?

- ○ Cosine similarity will no longer work
- ◉ The updated query may move too far from the relevant subspace
- ○ The TF-IDF values will be distorted
- ○ The influence of relevant documents will be amplified

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q2

1.5 point Answered

In the Rocchio update, which component is subtracted from the updated query?

- ○ The average of relevant document vectors
- ○ The centroid of the non-relevant documents, scaled by gamma and beta
- ○ The original query vector
- ◉ The centroid of the non-relevant documents, scaled by gamma

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.2 Rocchio Feedback (15 Points)

Read the Python code provided below, which implements a Rocchio-based query update using TF-IDF vectors and cosine similarity. Then, answer a set of multiple choice questions that test your understanding of the algorithm, its mathematical structure, and its behavior.

Tip: Rather than averaging all relevant documents equally, this version of Rocchio Algorithm uses pre-assigned weights to control how much each document influences the query.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from IPython.display import display

tfidf_df = pd.DataFrame([
    [0.3, 0.0, 0.2, 0.0],  # d1 (relevant)
    [0.1, 0.4, 0.3, 0.0],  # d2 (non-relevant)
    [0.0, 0.1, 0.2, 0.5],  # d3
    [0.4, 0.4, 0.0, 0.0],  # d4 (relevant)
    [0.0, 0.0, 0.4, 0.6]   # d5 (non-relevant)
], index=["d1", "d2", "d3", "d4", "d5"], columns=["term1", "term2", "term3", "term4"])

query_vector = pd.Series([0.6, 0.6, 0.3, 0.0], index=tfidf_df.columns)
query_vector = pd.Series(normalize(query_vector.values.reshape(1, -1))[0], index=tfidf_df.columns)

relevance_feedback = {
    'relevant': {'d1': 1.0, 'd4': 0.8},
    'non_relevant': {'d2': 1.0, 'd5': 0.6}
}

def rocchio_weighted(query_vector, tfidf_df, feedback_dict, alpha=1.0, beta=1.0, gamma=4.0, normalize_output=True):
    relevant_docs = list(feedback_dict['relevant'].keys())
    non_relevant_docs = list(feedback_dict['non_relevant'].keys())

    rel_weights = np.array([feedback_dict['relevant'][d] for d in relevant_docs])
    nonrel_weights = np.array([feedback_dict['non_relevant'][d] for d in non_relevant_docs])

    rel_matrix = tfidf_df.loc[relevant_docs].to_numpy()
    nonrel_matrix = tfidf_df.loc[non_relevant_docs].to_numpy()

    rel_centroid = np.average(rel_matrix, axis=0, weights=rel_weights)
    nonrel_centroid = np.average(nonrel_matrix, axis=0, weights=nonrel_weights)

    updated_query = alpha * query_vector.to_numpy() + beta * rel_centroid - gamma * nonrel_centroid

    if normalize_output:
        updated_query = normalize(updated_query.reshape(1, -1))[0]

    return pd.Series(updated_query, index=query_vector.index)

updated_query_vector = rocchio_weighted(query_vector, tfidf_df, relevance_feedback)

cos_sim_before = cosine_similarity(tfidf_df.values, query_vector.values.reshape(1, -1))
cos_sim_after = cosine_similarity(tfidf_df.values, updated_query_vector.values.reshape(1, -1))

ranked_docs_before = pd.Series(cos_sim_before.flatten(), index=tfidf_df.index).sort_values(ascending=False)
ranked_docs_after = pd.Series(cos_sim_after.flatten(), index=tfidf_df.index).sort_values(ascending=False)
```

Hide

Q1

1.5 point Answered

Suppose the defined query vector emphasizes term1 and term2. How does this affect the cosine similarity before feedback?

- ○ The similarity scores will all be equal
- ○ Documents with higher values in term3 and term4 will rank higher
- ◉ Documents emphasizing term1 and term2 will likely rank higher
- ○ Only relevant documents are matched

Submit answer | Next question >

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

Hide description

3.1 Document-Query Similarity (15 Points)

Complete the following Python program that calculates the **cosine similarity** between a query and a set of documents using the **Vector Space Model (VSM)**.
Fill in the blanks with the appropriate code snippets from the provided choices.

Each blank is worth 1.5 points.

Hide

Gap with dropdown

15 points Answered

```
import pandas as pd
import numpy as np

corpus = [
    "sun sun",
    "cloud wind wind rain rain",
    "sun cloud cloud wind",
    "sun sun sun cloud cloud cloud wind wind wind rain",
    "wind wind snow snow"
]
query = "cloud wind"
vocabulary = ['sun', 'cloud', 'wind', 'rain', 'snow']
def build_tf_matrix(corpus, vocabulary):
    tf_matrix = []
    for doc in corpus:
        words = _________
        tf_vector = [_________ for term in vocabulary]
        tf_matrix.append(tf_vector)
    return pd.DataFrame(_________, columns=vocabulary)

tf_matrix = build_tf_matrix(corpus, vocabulary)

def compute_idf(tf_matrix):
    N = len(_________)
    df = (tf_matrix > _________).sum(axis=0)
    idf = np.log10(_________ / df)
    return idf

idf_vector = compute_idf(tf_matrix)
tfidf_matrix = _________ * idf_vector

def build_query_vector(query, vocabulary, idf_vector):
    words = query.split()
    tf = pd.Series([words.count(term) for term in vocabulary], index=vocabulary)
    return tf * idf_vector

query_vector = build_query_vector(query, vocabulary, idf_vector)

def cosine_similarity(tfidf_matrix, query_vector):
    similarities = {}
    q_vec = _________.to_numpy()
    q_norm = np.linalg.norm(q_vec)
    for i, doc_vector in tfidf_matrix.iterrows():
        d_vec = doc_vector.to_numpy()
        d_norm = np.linalg.norm(_________)
        dot_product = np.dot(d_vec, q_vec)
        sim = dot_product / (d_norm * q_norm) if d_norm > 0 and q_norm > 0 else _________
        similarities[f'd{i+1}'] = sim
    return similarities

similarities = cosine_similarity(tfidf_matrix, query_vector)
print("TF-IDF Cosine Similarities:", similarities)
print(tfidf_matrix)
```

Line 16 doc.split()
Line 17 words.count(term)
Line 19 tf_matrix
Line 24 tf_matrix
Line 25 0
Line 26 N
Line 30 tf_matrix
Line 41 query_vector
Line 45 d_vec
Line 47 0.0

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)

- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)

- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- **2.4 Boolean Logic (2 Points)** 1

3. Programming Tasks (30 Points)

- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

2.4 Boolean Logic (2 Points)

2 points Answered

Translate the natural language query below into a Boolean logic expression using AND, OR, and NOT:
Find articles that mention **"climate change" but exclude "global warming" unless "carbon emissions" is also mentioned**.

- ("climate change") AND (NOT "carbon emissions" OR "global warming")
- ("climate change") AND (NOT "global warming" OR "carbon emissions")
- ("climate change") AND NOT ("carbon emissions") OR (("global warming") AND ("carbon emissions"))
- ("climate change") AND NOT ("global warming") OR (("global warming") AND ("carbon emissions"))
- ("climate change") OR (NOT "global warming" OR "carbon emissions")

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - **2.3.4 Mean Average Precision (MAP) (2 Points)** 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.3 Evaluation (8 Points)

Suppose you're building a **web search engine**. A user submits the query:

```
Mental health interventions using digital platforms
```

Your system returns the following ranked documents (top 10):

Rank	Document ID	Is Relevant?
1	D8	Yes
2	D3	No
3	D5	Yes
4	D6	Yes
5	D1	No
6	D7	Yes
7	D2	No
8	D9	No
9	D10	Yes
10	D4	No

Compute the following evaluation metrics at cut-off k=10:

- Precision@10
- Recall@10 (Assume total relevant documents in the collection = 6)
- F1-score@10
- Mean Average Precision (MAP)

Hide

2.3.4 Mean Average Precision (MAP) (2 Points)

2 points Answered

Note: Give the numerical answers rounded to **3** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0, 0.5→ 0.5

Mean Average Precision (MAP) : 0.606

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - **2.3.3 F1-score@10 (2 Points)** 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.3 Evaluation (8 Points)

Suppose you're building a **web search engine**. A user submits the query:

```
Mental health interventions using digital platforms
```

Your system returns the following ranked documents (top 10):

Rank	Document ID	Is Relevant?
1	D8	Yes
2	D3	No
3	D5	Yes
4	D6	Yes
5	D1	No
6	D7	Yes
7	D2	No
8	D9	No
9	D10	Yes
10	D4	No

Compute the following evaluation metrics at cut-off k=10:

- Precision@10
- Recall@10 (Assume total relevant documents in the collection = 6)
- F1-score@10
- Mean Average Precision (MAP)

Hide

2.3.3 F1-score@10 (2 Points)

2 points Answered

Note: Give the numerical answers rounded to **3** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0, 0.5→ 0.5

F1-score@10 (use the values for precision and recall, you computed in the previous questions): 0.625

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 41 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - **2.3.2 Recall@10 (2 Points)** 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.3 Evaluation (8 Points)

Suppose you're building a **web search engine**. A user submits the query:

```
Mental health interventions using digital platforms
```

Your system returns the following ranked documents (top 10):

Rank	Document ID	Is Relevant?
1	D8	Yes
2	D3	No
3	D5	Yes
4	D6	Yes
5	D1	No
6	D7	Yes
7	D2	No
8	D9	No
9	D10	Yes
10	D4	No

Compute the following evaluation metrics at cut-off k=10:

- Precision@10
- Recall@10 (Assume total relevant documents in the collection = 6)
- F1-score@10
- Mean Average Precision (MAP)

Hide

2.3.2 Recall@10 (2 Points)

2 points Answered

Note: Give the numerical answers rounded to **3** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0, 0.5→ 0.5

Recall@10 (Assume total relevant documents in the collection = 6):

0.833

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - **2.3.1 Precision@10 (2 Points)** 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.3 Evaluation (8 Points)

Suppose you're building a **web search engine**. A user submits the query:

```
Mental health interventions using digital platforms
```

Your system returns the following ranked documents (top 10):

Rank	Document ID	Is Relevant?
1	D8	Yes
2	D3	No
3	D5	Yes
4	D6	Yes
5	D1	No
6	D7	Yes
7	D2	No
8	D9	No
9	D10	Yes
10	D4	No

Compute the following evaluation metrics at cut-off k=10:

- Precision@10
- Recall@10 (Assume total relevant documents in the collection = 6)
- F1-score@10
- Mean Average Precision (MAP)

Hide

2.3.1 Precision@10 (2 Points)

2 points Answered

Note: Give the numerical answers rounded to **3** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0, 0.5→ 0.5

Precision@10: 0.5

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

Hide description

2.2 Query Likelihood Scoring (18 Points)

You are given a document:

d1 = "deep learning improves natural language understanding in many applications"

And a query:

q = "language applications"

If we assume the document itself is the collection, compute the **query likelihood** score:

Hide

2.2.3 Jelinek-Mercer Smoothing (6 Points)

6 points Answered

Under a **unigram** language model using Jelinek-Mercer smoothing (λ = 0.8, use uniform background model) 0.012

Note:

A **uniform background model** assumes that every term in the vocabulary has **equal probability** in the global collection:

$$P(w \mid C) = \frac{1}{|V|}$$

for all w in the vocabulary V.

Also: Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - **2.2.2 Laplace Smoothing (6 Points)** 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.2 Query Likelihood Scoring (18 Points)

You are given a document:

d1 = "deep learning improves natural language understanding in many applications"

And a query:

q = "language applications"

If we assume the document itself is the collection, compute the **query likelihood** score:

Hide

2.2.2 Laplace Smoothing (6 Points) 6 points Answered

Under a **unigram** language model using Laplace smoothing (use '1' in the Laplace formula for *alpha*) 0.012

Note: Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0

Submit answer Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
   - 1.1 True/False Questions (15 Points) 1
   - 1.2 Multiple Choice Questions (5 Points)
     - Q1 (0.5 Points) 1
     - Q2 (0.5 Points) 1
     - Q3 (0.5 Points) 1
     - Q4 (0.5 Points) 1
     - Q5 (0.5 Points) 1
     - Q6 (0.5 Points) 1
     - Q7 (0.5 Points) 1
     - Q8 (0.5 Points) 1
     - Q9 (0.5 Points) 1
     - Q10 (0.5 Points) 1
2. Practical Tasks (50 Points)
   - 2.1 Vector-Based Ranking (22 Points)
     - 2.1.1 TF-IDF (8.8 Points) 1
     - 2.1.2 Cosine Similarity Scores (12 Points) 1
     - 2.1.3 Document Ranking (1.2 Points) 1
   - 2.2 Query Likelihood Scoring (18 Points)
     - **2.2.1 Maximum Likelihood Estimation (6 Points)** 1
     - 2.2.2 Laplace Smoothing (6 Points) 1
     - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
   - 2.3 Evaluation (8 Points)
     - 2.3.1 Precision@10 (2 Points) 1
     - 2.3.2 Recall@10 (2 Points) 1
     - 2.3.3 F1-score@10 (2 Points) 1
     - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
   - 2.4 Boolean Logic (2 Points) 1
3. Programming Tasks (30 Points)
   - 3.1 Document-Query Similarity (15 Points)
     - Gap with dropdown 1
   - 3.2 Rocchio Feedback (15 Points)
     - Q1 1
     - Q2 1
     - Q3 1
     - Q4 1
     - Q5 1
     - Q6 1
     - Q7 1
     - Q8 1
     - Q9 1
     - Q10 1

Hide description

2.2 Query Likelihood Scoring (18 Points)

You are given a document:

d1 = "deep learning improves natural language understanding in many applications"

And a query:

q = "language applications"

If we assume the document itself is the collection, compute the **query likelihood** score:

Hide

2.2.1 Maximum Likelihood Estimation (6 Points)

6 points Answered

Under a **unigram** language model using Maximum Likelihood Estimation (MLE): 0.012

Under a **diagram** language model using Maximum Likelihood Estimation (MLE): 0.0

Note: Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

Hide description

2.1 Vector-Based Ranking (22 Points)

Given the following documents:

```
d0 = "learning vector representation classification"
d1 = "probabilistic inference language model"
d2 = "vector space model for retrieval"
```

and the Query:

```
q = "vector model retrieval"
```

Hide

2.1.3 Document Ranking (1.2 Points)

1.2 point | Answered

Rank the documents by similarity. Which document has the highest rank?

- ○ d0
- ○ d1
- ◉ d2

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

Hide description

2.1 Vector-Based Ranking (22 Points)

Given the following documents:

```
d0 = "learning vector representation classification"
d1 = "probabilistic inference language model"
d2 = "vector space model for retrieval"
```

and the Query:

```
q = "vector model retrieval"
```

Hide

2.1.2 Cosine Similarity Scores (12 Points)

12 points Answered

Compute **cosine similarity** scores between the query and each document. Each right answer scores 4 points. No negative points are deducted for wrong answers.

Note: Give the numerical answers rounded to **two** decimal places, e.g., 0.18. Round the last digit as follows: 0.182 → 0.18, 0.185 → 0.19, 0.186 → 0.19

d0: 0.18

d1: 0.18

d2: 0.65

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - **2.1.1 TF-IDF (8.8 Points)** 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

2.1 Vector-Based Ranking (22 Points)

Given the following documents:

```
d0 = "learning vector representation classification"
d1 = "probabilistic inference language model"
d2 = "vector space model for retrieval"
```

and the Query:

```
q = "vector model retrieval"
```

Hide

2.1.1 TF-IDF (8.8 Points)

8.8 points Answered

Compute the TF-IDF vectors for each document and the query. Each right answer scores 0.2 points.

Note: Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0.0

	do	d1	d2	query
classification	0.542	0.0	0.0	0.0
for	0.0	0.0	0.539	0.0
inference	0.0	0.542	0.0	0.0
language	0.0	0.542	0.0	0.0
learning	0.542	0.0	0.0	0.0
model	0.0	0.346	0.344	0.533
probabilistic	0.0	0.542	0.0	0.0
representation	0.542	0.0	0.0	0.0
retrieval	0.0	0.0	0.425	0.658
space	0.0	0.0	0.539	0.0
vector	0.346	0.0	0.344	0.533

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - Q9 (0.5 Points) 1
  - **Q10 (0.5 Points)** 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q10 (0.5 Points) 0.5 point Answered

In the context of IR evaluation, what does the Average Precision (AP) metric measure?

- ○ The harmonic mean of precision and recall
- ○ The fraction of non-relevant documents retrieved
- ◉ The average of precision values at each relevant document in the ranked list
- ○ The total number of relevant documents retrieved

Submit answer | Next question

Go to top

Exam Eligibility Assignment

Finish test

102 hr 42 min | End at 6:00 PM hour

1. Knowledge Tasks (20 Points)
- 1.1 True/False Questions (15 Points) 1
- 1.2 Multiple Choice Questions (5 Points)
  - Q1 (0.5 Points) 1
  - Q2 (0.5 Points) 1
  - Q3 (0.5 Points) 1
  - Q4 (0.5 Points) 1
  - Q5 (0.5 Points) 1
  - Q6 (0.5 Points) 1
  - Q7 (0.5 Points) 1
  - Q8 (0.5 Points) 1
  - **Q9 (0.5 Points)** 1
  - Q10 (0.5 Points) 1

2. Practical Tasks (50 Points)
- 2.1 Vector-Based Ranking (22 Points)
  - 2.1.1 TF-IDF (8.8 Points) 1
  - 2.1.2 Cosine Similarity Scores (12 Points) 1
  - 2.1.3 Document Ranking (1.2 Points) 1
- 2.2 Query Likelihood Scoring (18 Points)
  - 2.2.1 Maximum Likelihood Estimation (6 Points) 1
  - 2.2.2 Laplace Smoothing (6 Points) 1
  - 2.2.3 Jelinek-Mercer Smoothing (6 Points) 1
- 2.3 Evaluation (8 Points)
  - 2.3.1 Precision@10 (2 Points) 1
  - 2.3.2 Recall@10 (2 Points) 1
  - 2.3.3 F1-score@10 (2 Points) 1
  - 2.3.4 Mean Average Precision (MAP) (2 Points) 2
- 2.4 Boolean Logic (2 Points) 1

3. Programming Tasks (30 Points)
- 3.1 Document-Query Similarity (15 Points)
  - Gap with dropdown 1
- 3.2 Rocchio Feedback (15 Points)
  - Q1 1
  - Q2 1
  - Q3 1
  - Q4 1
  - Q5 1
  - Q6 1
  - Q7 1
  - Q8 1
  - Q9 1
  - Q10 1

Hide description

1.2 Multiple Choice Questions (5 Points)

Select the correct answer. No negative points are deducted for incorrect answers.

Hide

Q9 (0.5 Points) 0.5 point Answered

What is the purpose of a posting list in an inverted index?

- It records term frequencies
- It stores all stopwords
- It lists documents where a term occurs
- It compresses the entire document

Submit answer Next question

Go to top