CS 589 Lecture 2: Probabilistic Retrieval Models - Complete Study Guide

Course: CS 589 - Information Retrieval

Lecture: 2

Topics: Probability Ranking Principle, RSJ Model, BM25, Language Model-Based Retrieval

---

Table of Contents

---

1. Fundamentals: Random Variables & Probability

1.1 Random Variables: The Biased Coin Example

Scenario: You have a biased coin that comes up heads with some unknown probability.

Observation sequence: H, T, H, H, T, T, H, T, H, T, H, H

- Heads: 7 times
- Tails: 5 times

Goal: Estimate the probability of heads

Model: Bernoulli Distribution

```
P(Heads) = α
P(Tails) = 1 - α
```

Parameter: α (what we want to estimate)

Observation: The sequence we observed

Graphical representation:

```
sequence ─────────────────┐
        │
        ↓
       (α)
    parameter
```

1.2 Maximum Likelihood Estimation (MLE)

Question: Given observations, what's the best estimate for α?

Answer: The value that maximizes the probability of observing our data

Probability of our sequence:

```
P(sequence) = α × (1-α) × α × α × (1-α) × (1-α) × α × (1-α) × α × (1-α) × α × α
            = α^count(heads) × (1-α)^count(tails)
            = α^7 × (1-α)^5
```

To maximize, take derivative and set to zero:

```
d/dα [α^7 × (1-α)^5] = 0
```

Result (by calculus):

```
α̂ = count(heads) / [count(heads) + count(tails)]
 = 7 / (7 + 5)
 = 7/12
 ≈ 0.583
```

Visualization of likelihood:

```
Probability
  ↑
0.0025|     /\
  |    /  \
0.0020|   /    \
  |  /      \
0.0015| /        \
  | /         \__
0.0010|/______________\___
  |____________________|→ α
   0  0.2  0.4  0.6  0.8  1.0
          ↑
        peak at 7/12
```

Key insight: The MLE is just the empirical frequency!

1.3 Bayes' Rule

Chain Rule (Joint Distribution)

```
P(A, B) = P(A ∩ B) = P(A|B)P(B) = P(B|A)P(A)
```

Bayes' Rule

From the chain rule, we can derive:

```
P(A|B) = P(B|A)P(A) / P(B)
```

Or equivalently:

```
          P(B|A) × P(A)
P(A|B) = ─────────────────────────────
        ∑_{X∈{A,Ā}} P(B|X)P(X)
```

Terminology

- **P(A|B)**: Posterior (what we want to know)
- **P(B|A)**: Likelihood (probability of data given hypothesis)
- **P(A)**: Prior (what we knew before seeing data)
- **P(B)**: Evidence (normalizing constant)

Proportionality

Since P(B) doesn't depend on A:

```
P(A|B) ∝ P(B|A)P(A)
```

Example: Medical diagnosis

```
P(disease|symptoms) = P(symptoms|disease) × P(disease) / P(symptoms)
```

1.4 Random Variables in Information Retrieval

Mapping to IR

Coin Toss	Information Retrieval
Outcome: H or T	Relevance: 0 or 1
α = P(Heads)	P(rel=1|q,d) = probability document is relevant
Sequence of tosses	Collection of documents

Key notation

- **q**: query
- **d**: document
- **rel**: relevance (0 = not relevant, 1 = relevant)

Example

Query: "artificial intelligence"

Documents in collection:

```
d1 = [artificial, intelligence, machine, intelligence, information, retrieval]
d2 = [covid, patient, virus, ...]
d3 = [machine, learning, deep, neural, ...]
...
```

Observations:

```
rel = [0, 1, 0, 0, 0, 0, 0, 0]
       |  |
       |  └─ d2 is relevant
       └─ d1 is not relevant (for this query)
```

Our goal: Estimate P(rel=1|q, d) for each document

---

2. Probability Ranking Principle (PRP)

2.1 The Principle

Statement:

> Documents should be ranked by their probability of relevance P(rel=1|q,d) in decreasing order.

Theorem (Ripley 1996):

> The PRP is optimal in the sense that it minimizes expected loss.

Why is this important?

- Provides theoretical foundation for retrieval
- Tells us what to estimate (relevance probability)
- Doesn't tell us HOW to estimate it (that's where models come in)

2.2 From Counting to Probability

Naive approach

```
P(rel=1|q,d) = count(rel=1, q, d) / count(q, d)
```

Problems

1. **Not enough data:** Most (q,d) pairs never observed
2. **Cannot adapt to new queries:** Need relevance judgments for every new query

Example: If we have 1 million documents and 1000 queries:

- Possible (q,d) pairs: 1 billion
- Actually judged: maybe 50,000
- 99.995% of pairs have no data!

2.3 Solution: Use Bayes' Rule

Transform the problem:

```
P(rel=1|q,d) = P(d|rel=1,q) × P(rel=1) / P(d)
```

Using odds ratio (helps cancel terms):

```
O(rel=1|q,d) = P(rel=1|q,d) / P(rel=0|q,d)
```

Property of odds:

$$a/(1-a) = 1/(1-a) - 1$$

So if we can rank by odds, we can rank by probability!

Why odds? Because:

$$O(rel=1|q,d) = [P(d|rel=1,q) \times P(rel=1)] / [P(d|rel=0,q) \times P(rel=0)]$$

The P(d) cancels out! Now we have:

$$O(rel=1|q,d) \propto P(d|rel=1,q) / P(d|rel=0,q)$$

---

3. Model 1: Robertson & Sparck Jones (RSJ)

3.1 Key Assumptions

Independence assumption: Words in documents are independent given relevance

$$P(d|rel,q) = \prod_{i\in V} P(wi|rel,q)$$

Binary occurrence: Each word either appears (wi=1) or doesn't (wi=0)

3.2 Complete Derivation

Step 1: Start with odds

$$O(rel=1|q,d) = P(d|rel=1,q)P(rel=1) / [P(d|rel=0,q)P(rel=0)]$$

Step 2: Apply independence assumption

$$P(d|rel=1,q) = \prod_{i\in V} P(wi|rel=1,q)$$

Step 3: Split into words in document vs. not in document

For each word, either wi=1 (in doc) or wi=0 (not in doc):

$$P(d|rel=1,q) = \prod_{wi=1} P(wi=1|rel=1,q) \times \prod_{wi=0} P(wi=0|rel=1,q)$$

Similarly:

$$P(d|rel=0,q) = \prod_{wi=1} P(wi=1|rel=0,q) \times \prod_{wi=0} P(wi=0|rel=0,q)$$

Step 4: Form the odds ratio

$$
\begin{aligned}
O(rel=1|q,d) = &\left[\prod_{wi=1} P(wi=1|rel=1,q) / P(wi=1|rel=0,q)\right]\\
&\times \left[\prod_{wi=1} P(wi=0|rel=1,q) / P(wi=0|rel=0,q)\right]\\
&\times \left[\prod_{wi=0} P(wi=1|rel=1,q) / P(wi=1|rel=0,q)\right]\\
&\times \left[\prod_{wi=0} P(wi=0|rel=1,q) / P(wi=0|rel=0,q)\right]
\end{aligned}
$$

Step 5: Key simplification

The last two products (over wi=0) don't depend on the document! They're the same for all documents, so for ranking purposes, we can ignore them.

$$
\begin{aligned}
O(rel=1|q,d) \propto &\prod_{wi=1} [P(wi=1|rel=1,q) / P(wi=1|rel=0,q)]\\
&\times [P(wi=0|rel=1,q) / P(wi=0|rel=0,q)]
\end{aligned}
$$

Using $P(wi=0|rel,q) = 1 - P(wi=1|rel,q)$:

$$
\begin{aligned}
O(rel=1|q,d) \propto &\prod_{wi=1} [P(wi=1|rel=1,q) / P(wi=1|rel=0,q)]\\
&\times [(1-P(wi=1|rel=1,q)) / (1-P(wi=1|rel=0,q))]
\end{aligned}
$$

Step 6: Take logarithm (preserves ranking)

$$
\log O(rel=1|q,d) = \sum_{wi=1} \log[\alpha i(1-\beta i) / (\beta i(1-\alpha i))]
$$

where:

- **$\alpha i = P(wi=1|rel=1,q)$**: probability word i appears in relevant documents
- **$\beta i = P(wi=1|rel=0,q)$**: probability word i appears in non-relevant documents

3.3 Final RSJ Formula

Ranking function

$$
score^{RSJ}(q,d) = \sum_{wi=1} \log[\alpha i(1-\beta i) / (\beta i(1-\alpha i))]
$$

Estimation with smoothing (avoid zeros)

$$
\alpha i = [count(wi=1, q, rel=1) + 0.5] / [count(q, rel=1) + 1]
$$

$$
\beta i = [count(wi=1, q, rel=0) + 0.5] / [count(q, rel=0) + 1]
$$

Intuition

- If $\alpha i > \beta i$: word appears more in relevant docs → positive contribution
- If $\alpha i < \beta i$: word appears more in non-relevant docs → negative contribution

- The log ratio amplifies these differences

3.4 Example Calculation

Scenario:

- Query: "machine learning"
- We have 100 relevant documents, 900 non-relevant documents
- "machine": appears in 80 relevant, 200 non-relevant
- "learning": appears in 70 relevant, 150 non-relevant

Calculate αi and βi

For "machine":

```
α_machine = (80 + 0.5) / (100 + 1) = 80.5/101 ≈ 0.797
β_machine = (200 + 0.5) / (900 + 1) = 200.5/901 ≈ 0.222
```

For "learning":

```
α_learning = (70 + 0.5) / (100 + 1) = 70.5/101 ≈ 0.698
β_learning = (150 + 0.5) / (900 + 1) = 150.5/901 ≈ 0.167
```

Score for document containing both words

```
score = log[0.797×(1-0.222) / (0.222×(1-0.797))]
      + log[0.698×(1-0.167) / (0.167×(1-0.698))]

      = log[0.797×0.778 / (0.222×0.203)]
      + log[0.698×0.833 / (0.167×0.302)]

      = log[0.620 / 0.045] + log[0.581 / 0.050]

      = log[13.78] + log[11.62]

      = 2.62 + 2.45

      = 5.07
```

3.5 Advantages and Limitations

Advantages

- Probabilistic foundation
- Can incorporate relevance feedback

- Theoretically justified

Limitations

- Requires relevance judgments
- Binary occurrence only (doesn't use term frequency)
- Independence assumption may not hold

---

4. Model 2: BM25 (Okapi BM25)

4.1 Motivation

Problem with RSJ: Only uses binary occurrence (word present/absent)

Example:

```
d1: "machine learning machine learning machine"
d2: "machine"
```

RSJ treats both the same! But d1 clearly emphasizes "machine" more.

Goal: Incorporate term frequency (TF) into probabilistic model

4.2 The Eliteness Hypothesis

New hidden variable: Eliteness (Ei)

Definition: A document-term pair (d,wi) is "elite" if the document is substantially about the concept denoted by the term.

Example:

- Query: "artificial intelligence"
- d1: "...artificial intelligence systems...learning...neural..."
  - "artificial": ELITE (document is about AI)
  - "intelligence": ELITE
  - "learning": ELITE
  - "the": NOT ELITE
- d2: "...studying the artificial flowers..."
  - "artificial": NOT ELITE (different meaning)

Graphical model:

```
document d
   ↓
Eliteness Ei ────→ Term frequency tfi
   ↑
parameters θ
```

Key insight: Term occurrence depends on eliteness

4.3 Two-Poisson Mixture Model

Model term frequency as mixture of two Poisson distributions:

```
P(wi=tfi|q,rel=1) = π × [λ^tfi × e^(-λ) / tfi!] + (1-π) × [μ^tfi × e^(-μ) / tfi!]
                    \_________________/         \___________________/
                         ELITE              NON-ELITE
                    (high frequency)        (low frequency)
```

Parameters:

- **π**: probability document is elite for this term
- **λ**: rate parameter for elite distribution (large, e.g., 3-5)
- **μ**: rate parameter for non-elite distribution (small, e.g., 0.5)

Why two Poisson?

One Poisson can't capture "burstiness":

```
Poisson λ=3        Reality
   ↑                  ↑
0.25 |  ██           |  ██   ██
    | █ █           | █ █  █ █
0.15 | █  █          |   ███
    |  ██           |
0.05 |______███      |________________→
     0 1 2 3 4 5 6    0 1 2 3 4 5 6

     Single peak      Bimodal
```

Intuition: Words either appear rarely (background) or appear frequently (topical)

4.4 Approximation: Saturation Function

Problem: Three unknown parameters (π, λ, μ) are hard to estimate

Solution: Design a parameter-free approximation!

The 2-Poisson mixture behaves like a saturation function:

```
c_i^elite(tfi) ≈ tfi / (k1 + tfi)
```

Why this works

As tfi → 0:

```
tfi / (k1 + tfi) → 0
```

As tfi → ∞:

```
tfi / (k1 + tfi) → 1
```

Visualization:

```
contribution
   1.0 |_________________ (saturates)
       |      /
   0.8 |      /
       |     /
   0.6 |     /
       |   /
   0.4 |   /
       | /
   0.2 | /
       |/
   0.0 |________________→ tfi
       0  2  4  6  8  10

Different k1 values:
x/(0.2+x) ──── (fast saturation)
x/(1+x)   ──── (medium)
x/(3+x)   ──── (slow)
x/(10+x)  ──── (very slow)
```

Different curves for different k1:

```
k1 = 0.5:  Very fast saturation (aggressive)
k1 = 1.2:  Fast saturation (typical lower bound)
k1 = 2.0:  Medium saturation (typical upper bound)
k1 = 3.0:  Slow saturation
```

4.5 Complete BM25 Formula

Final scoring function:

```
score^BM25(q,d) = ∑_{wi∈q} log(N/dfi) × [tfi(k1+1)] / [k1((1-b)+b|d|/avgdl) + tfi]
```

Component analysis

1. IDF term:

```
log(N/dfi)
```

- N = total number of documents
- dfi = number of documents containing term i
- Exactly like TF-IDF!

2. Normalized TF term:

```
[tfi(k1+1)] / [k1×normalization + tfi]
```

3. Document length normalization (pivoting):

```
normalization = (1-b) + b×(|d|/avgdl)
```

- |d| = document length
- avgdl = average document length in collection
- b ∈ [0,1] controls strength (typically b=0.75)

4.6 Breaking Down the Formula

Let's see each component's effect

Scenario:

- N = 1,000,000 documents
- avgdl = 1,000 words
- k1 = 1.5, b = 0.75

Term: "neural"

- df = 10,000 documents contain it
- IDF = log(1,000,000/10,000) = log(100) ≈ 4.6

Document d1:

- Length: 500 words (short)
- "neural" appears 3 times

```
normalization = (1-0.75) + 0.75×(500/1000) = 0.25 + 0.375 = 0.625

TF component = [3×(1.5+1)] / [1.5×0.625 + 3]
            = [3×2.5] / [0.9375 + 3]
            = 7.5 / 3.9375
            = 1.905

score contribution = 4.6 × 1.905 = 8.76
```

Document d2:

- Length: 3000 words (long)
- "neural" appears 5 times

```
normalization = (1-0.75) + 0.75×(3000/1000) = 0.25 + 2.25 = 2.5

TF component = [5×(1.5+1)] / [1.5×2.5 + 5]
            = [5×2.5] / [3.75 + 5]
            = 12.5 / 8.75
            = 1.429

score contribution = 4.6 × 1.429 = 6.57
```

Notice: Despite d2 having higher TF (5 vs 3), d1 scores higher because it's shorter!

4.7 Parameter Effects

Effect of k1

```
k1 = 1.2:  More aggressive saturation
  tf=2 → 2(2.2)/(1.2+2) = 1.375
  tf=5 → 5(2.2)/(1.2+5) = 1.774

k1 = 2.0:  Less aggressive saturation
  tf=2 → 2(3)/(2+2) = 1.5
  tf=5 → 5(3)/(2+5) = 2.143
```

Effect of b

```
b = 0:  No length normalization
   Short doc (500 words): norm = 1.0
   Long doc (3000 words): norm = 1.0

b = 1:  Full length normalization
   Short doc: norm = 500/1000 = 0.5
   Long doc: norm = 3000/1000 = 3.0

b = 0.75:  Typical setting (partial normalization)
   Short doc: norm = 0.25 + 0.75×0.5 = 0.625
   Long doc: norm = 0.25 + 0.75×3.0 = 2.5
```

4.8 BM25F: Multi-Field Extension

Motivation: Documents have structure (title, body, metadata)

Example: StackOverflow question

```
Title: "How to cite presentation slides?"
Body: "A friend has made some nice slides..."
Tags: ["citations", "academic"]
```

Different fields have different importance!

BM25F formula

```
score^BM25F(q,d) = ∑_{wi∈q} log(N/dfi) × [tfi^F(k1+1)] / [k1((1-b)+b|d|^F/avgdl^F) + tfi^F]
```

Weighted combination:

```
tfi^F = ∑_f αf × tfi,f
|d|^F = ∑_f αf × |d|f
avgdl^F = ∑_f αf × avgdlf
```

Example weights:

```
α_title = 3.0    (title is very important)
α_body = 1.0     (body is baseline)
α_tags = 2.0     (tags are important)
```

Calculation

Document:

- Title (5 words): "cite presentation slides"
- Body (100 words): "...slides...cite..."

- Tags: ["citations"]

Query: "cite presentation"

```
tf_cite^F = 3.0×1 (title) + 1.0×1 (body) + 2.0×0 (tags) = 4.0
tf_presentation^F = 3.0×1 (title) + 1.0×0 (body) + 2.0×0 (tags) = 3.0

|d|^F = 3.0×5 + 1.0×100 + 2.0×2 = 119
```

4.9 Advantages and Limitations

Advantages

- State-of-the-art non-learning model
- Still widely used (Elasticsearch, Solr)
- Incorporates TF with principled normalization
- Few parameters to tune

Limitations

- Parameters (k1, b) hard to estimate theoretically
- No explicit relevance feedback mechanism (unlike RSJ)
- Still bag-of-words (no phrase information)

---

5. Model 3: Language Model-Based Retrieval

5.1 Key Paradigm Shift

RSJ/BM25 approach:

```
P(rel=1|q,d)  "Is document relevant given query?"
```

Language model approach:

```
P(q|d)  "Could this document generate this query?"
```

Graphical models:

```
RSJ:              LM:
 rel              q
 / \              |
/   \             |
q    d            d
```

Intuition:

- Imagine each document is a "language model" (probability distribution over words)
- Rank documents by likelihood they would generate the query

5.2 Statistical Language Model

Definition: A probability distribution over word sequences

Examples:

```
P("Today is Wednesday") ≈ 0.001        (reasonable sentence)
P("Today Wednesday is") ≈ 0.000000001   (ungrammatical)
P("Eigenvalue positive is") ≈ 0.0000001 (weird)
```

Unigram language model:

- Generate each word independently
- P(w1 w2 ... wn) = P(w1) × P(w2) × ... × P(wn)

Parameters: {P(w1), P(w2), ..., P(wN)} where ∑P(wi) = 1

Graphical model:

```
θ (parameters)
 |
 ├───→ w1
 ├───→ w2
 ├───→ w3
...
 └───→ wn
```

Example

Language model:

```
P("the") = 0.05
P("is") = 0.02
P("cat") = 0.001
P("eigenvalue") = 0.00001
...
```

Generate sentence:

```
P("the cat is") = P("the") × P("cat") × P("is")
              = 0.05 × 0.001 × 0.02
              = 0.000001
```

5.3 Query Likelihood Model

Ranking function:

```
score(q,d) = P(q|d) = ∏_{wi∈q} P(wi|d)
```

Taking log (preserves ranking, easier to compute):

```
score(q,d) = log P(q|d) = ∑_{wi∈q} log P(wi|d)
```

Example

Query: "presidential campaign news"

Document d: "...news of presidential campaign..... ...presidential candidate..."

Estimate P(wi|d) by MLE:

```
P_MLE(wi|d) = count(wi, d) / |d|
```

```
P("presidential"|d) = 2/100 = 0.02
P("campaign"|d) = 1/100 = 0.01
P("news"|d) = 1/100 = 0.01
```

```
score(q,d) = log(0.02) + log(0.01) + log(0.01)
           = -3.91 + (-4.61) + (-4.61)
           = -13.13
```

Problem: What if query word doesn't appear in document?

```
P("election"|d) = 0/100 = 0
log(0) = -∞

Entire score = -∞
```

This is catastrophic! We need smoothing.

5.4 Smoothing Methods

Core idea: Mix document language model with collection language model

Collection language model:

```
P(wi|C) = ∑_{d∈C} count(wi,d) / ∑_{d∈C} |d|
```

Method 1: Dirichlet Smoothing

Formula:

```
P_s(wi|d) = [count(wi,d) + μ×P(wi|C)] / [|d| + μ]
```

Rewrite as weighted combination:

```
P_s(wi|d) = [|d|/(|d|+μ)] × P_MLE(wi|d) + [μ/(|d|+μ)] × P(wi|C)
           \___________/                    \_________/
           weight on document           weight on collection
```

Parameter μ: Controls smoothing strength

- μ small (e.g., 100): Trust document more
- μ large (e.g., 5000): Trust collection more
- Typical value: μ ≈ 2000

Intuition:

```
              Document only
                   ↓
Long doc (|d|=5000):  5000/7000 = 71% document, 29% collection
Short doc (|d|=100):  100/2100 = 5% document, 95% collection
                          ↑
                   More smoothing needed
```

Example calculation:

Document d (100 words):

- "neural" appears 3 times
- P("neural"|C) = 0.001 (collection)
- μ = 2000

```
P_s("neural"|d) = [3 + 2000×0.001] / [100 + 2000]
              = [3 + 2] / 2100
              = 5/2100
              = 0.00238
```

For unseen word "quantum":

```
P_s("quantum"|d) = [0 + 2000×0.0001] / [100 + 2000]
               = 0.2 / 2100
               = 0.000095
```

Still non-zero!

Method 2: Jelinek-Mercer Smoothing

Formula:

```
P_s(wi|d) = λ×P_MLE(wi|d) + (1-λ)×P(wi|C)
```

Fixed interpolation weight:

- $\lambda \in [0, 1]$
- Typical: $\lambda \approx 0.7$

Difference from Dirichlet:

- Dirichlet: Adapts to document length
- JM: Same λ for all documents

Example:

$\lambda = 0.7$, same document as before:

```
P_s("neural"|d) = 0.7×(3/100) + 0.3×0.001
                = 0.7×0.03 + 0.0003
                = 0.021 + 0.0003
                = 0.0213
```

For unseen "quantum":

```
P_s("quantum"|d) = 0.7×0 + 0.3×0.0001
                 = 0.00003
```

5.5 Ranking Formula (Dirichlet)

Starting from:

```
log P(q|d) = ∑_{wi∈q} log P_s(wi|d)
```

Substitute Dirichlet smoothing:

```
= ∑_{wi∈q} log([count(wi,d) + μ×P(wi|C)] / [|d| + μ])
```

Simplify (see derivation link: https://www.overleaf.com/read/jbztxtnbzwcx):

```
= ∑_{wi∈q,wi∈d} log(count(wi,d) / [μ×P(wi|C)]) + |q|×log[μ/(μ+|d|)]
  \__________________________________/              \________________/
     depends on matching                          document length penalty
```

For ranking (ignore constant |q|):

```
score^Dir(q,d) = ∑_{wi∈q,wi∈d} log(1 + count(wi,d)/(μ×P(wi|C))) + log(μ/(μ+|d|))
```

Intuition:

1. For each query term in document, get score based on its frequency
2. Longer documents get penalized by log term

5.6 Feedback Language Model

Problem: Query is often too short and ambiguous

Example: Query: "apple"

- Could mean: fruit, company, music, records, pie, tree...

Solution: Use pseudo-relevance feedback

Pseudo-relevance feedback

1. Retrieve top-k documents with initial query
2. Assume these are relevant (even without judgments)
3. Expand query using these documents

Model: Feedback creates improved query model

```
θq^FB = λ×θq + (1-λ)×θq^F
```

where:

- θq = original query model
- θq^F = model learned from feedback documents
- λ ∈ [0,1] = interpolation weight

How to estimate θq^F?

EM Algorithm:

Assume feedback documents are mixture:

```
P(d|θq^F) = ∏_{w∈d} [θq^F(w)]^count(w,d)
```

E-step: Estimate which words came from topic vs. background

```
P(z=topic|w,d) = [θq^F(w)×λ] / [θq^F(w)×λ + P(w|C)×(1-λ)]
```

M-step: Update θq^F

```
θq^F(w) ∝ ∑_{d∈feedback} count(w,d) × P(z=topic|w,d)
```

Iterate until convergence

5.7 Complete Example

Query: "airport security"

Step 1: Initial retrieval using query likelihood

Retrieve top 5 documents

Step 2: Extract top terms from feedback documents

```
Document 1: "...airport security measures... passengers... screening..."
Document 2: "...TSA procedures...airport... safety..."
Document 3: "...security checkpoints... baggage... inspection..."
```

Step 3: Learn θq^F using EM

Discovered related terms:

```
P("passengers"|θq^F) = 0.08
P("screening"|θq^F) = 0.06
P("TSA"|θq^F) = 0.05
P("checkpoints"|θq^F) = 0.04
...
```

Step 4: Combine with original query

```
θq^FB("airport") = 0.7×0.5 + 0.3×0.03 = 0.359
θq^FB("security") = 0.7×0.5 + 0.3×0.02 = 0.356
θq^FB("passengers") = 0.7×0 + 0.3×0.08 = 0.024  ← NEW
θq^FB("screening") = 0.7×0 + 0.3×0.06 = 0.018   ← NEW
```

Step 5: Rerank using expanded query

5.8 Advantages and Limitations

Advantages

- Flexible framework

- Can easily incorporate feedback
- Probabilistic foundation
- State-of-the-art performance with good smoothing

Limitations

- Query-document equivalence assumption unrealistic
- Unigram model (no phrases)
- Smoothing parameters must be tuned

5.9 Comparison: BM25 vs Language Models

Empirical results (Bennett et al. 2008):

On some datasets:

- BM25 outperforms LM (e.g., TREC TD)
- LM with feedback outperforms baseline LM

Typical performance ranking:

```
BM25 ≈ LM-Dirichlet > LM-JM > RSJ
```

When to use which:

- **BM25:** Simple, robust, well-tested
- **LM:** Want feedback, need theoretical framework
- **BM25F:** Multi-field documents
- **Feedback LM:** Query expansion is critical

---

6. Practice Problems with Solutions

Problem 1: MLE for Biased Coin

Question: You flip a coin 20 times:

- Heads: 13 times
- Tails: 7 times

What's the MLE for P(Heads)?

Solution:

```
α̂ = count(heads) / [count(heads) + count(tails)]
  = 13 / (13 + 7)
  = 13/20
  = 0.65
```

Problem 2: Bayes Rule Application

Question:

- P(disease) = 0.01 (1% of population has disease)
- P(positive test|disease) = 0.95 (95% sensitivity)
- P(positive test|no disease) = 0.05 (5% false positive)

If someone tests positive, what's P(disease|positive test)?

Solution:

Use Bayes:

```
P(disease|pos) = P(pos|disease)×P(disease) / P(pos)
```

Calculate P(pos):

```
P(pos) = P(pos|disease)×P(disease) + P(pos|no disease)×P(no disease)
       = 0.95×0.01 + 0.05×0.99
       = 0.0095 + 0.0495
       = 0.059
```

Therefore:

```
P(disease|pos) = (0.95×0.01) / 0.059
               = 0.0095 / 0.059
               ≈ 0.161 or 16.1%
```

Surprising! Despite positive test, only 16% chance of disease (because disease is rare).

Problem 3: RSJ Scoring

Given:

- Query: "machine learning"
- 50 relevant documents, 950 non-relevant
- "machine": 40 relevant, 100 non-relevant

- "learning": 35 relevant, 80 non-relevant

Calculate RSJ score for document containing both terms.

Solution:

Calculate α and β with smoothing:

```
α_machine = (40 + 0.5) / (50 + 1) = 40.5/51 ≈ 0.794
β_machine = (100 + 0.5) / (950 + 1) = 100.5/951 ≈ 0.106

α_learning = (35 + 0.5) / (50 + 1) = 35.5/51 ≈ 0.696
β_learning = (80 + 0.5) / (950 + 1) = 80.5/951 ≈ 0.085
```

Calculate score:

```
score = log[0.794×(1-0.106) / (0.106×(1-0.794))]
    + log[0.696×(1-0.085) / (0.085×(1-0.696))]

    = log[0.794×0.894 / (0.106×0.206)]
    + log[0.696×0.915 / (0.085×0.304)]

    = log[0.710 / 0.022] + log[0.637 / 0.026]

    = log[32.27] + log[24.50]

    = 3.47 + 3.20

    = 6.67
```

Problem 4: BM25 Calculation

Given:

- N = 10,000 documents
- avgdl = 500 words
- k1 = 1.5, b = 0.75
- Term "neural": df = 100
- Document: length = 300, "neural" appears 4 times

Calculate BM25 score contribution.

Solution:

IDF:

```
IDF = log(N/df) = log(10,000/100) = log(100) ≈ 4.605
```

Document length normalization:

```
norm = (1-b) + b×(|d|/avgdl)
    = (1-0.75) + 0.75×(300/500)
    = 0.25 + 0.75×0.6
    = 0.25 + 0.45
    = 0.70
```

TF component:

```
TF_score = [4×(1.5+1)] / [1.5×0.70 + 4]
        = [4×2.5] / [1.05 + 4]
        = 10 / 5.05
        = 1.980
```

Total:

```
score = IDF × TF_score
     = 4.605 × 1.980
     = 9.118
```

Problem 5: Language Model Smoothing

Given:

- Document: "neural networks deep learning neural" (5 words)
- Query: "neural quantum"
- P("neural"|C) = 0.002
- P("quantum"|C) = 0.0001
- $\mu = 1000$

Calculate score using Dirichlet smoothing.

Solution:

For "neural" (appears 2 times):

```
P_s("neural"|d) = [2 + 1000×0.002] / [5 + 1000]
               = [2 + 2] / 1005
               = 4/1005
               = 0.00398
```

For "quantum" (appears 0 times):

```
P_s("quantum"|d) = [0 + 1000×0.0001] / [5 + 1000]
                = 0.1 / 1005
                = 0.0000995
```

Score:

```
score = log(0.00398) + log(0.0000995)
     = -5.53 + (-9.22)
     = -14.75
```

Problem 6: Feedback Language Model

Given:

- Original query model: P("apple"|θq) = 0.5, P("pie"|θq) = 0.5
- Feedback documents yield: P("apple"|θq^F) = 0.3, P("pie"|θq^F) = 0.2, P("recipe"|θq^F) = 0.15
- λ = 0.7

Calculate θq^FB.

Solution:

```
P("apple"|θq^FB) = 0.7×0.5 + 0.3×0.3 = 0.35 + 0.09 = 0.44

P("pie"|θq^FB) = 0.7×0.5 + 0.3×0.2 = 0.35 + 0.06 = 0.41

P("recipe"|θq^FB) = 0.7×0 + 0.3×0.15 = 0 + 0.045 = 0.045  ← NEW TERM!
```

Notice: "recipe" wasn't in original query but now has probability 0.045

---

7. Quiz Questions & Detailed Solutions

Quiz Question 1 (From Lecture 2 Quiz)

Question: Suppose we have one query and two documents:

- q = "covid 19"

- d1 = "covid patient"
- d2 = "19 99 car wash"
- d3 = "19 street covid testing facility is reopened next week"

If we use the vector space model **without IDF**, which document has the highest and the lowest score?

Options:

- d1, d2
- d1, d3
- d3, d1
- d3, d2

Solution:

Without IDF, we're just counting term matches (TF only).

Query terms: {"covid", "19"}

d1: "covid patient"

- Contains: "covid" ✓
- Contains: "19" ✗
- **Matches: 1/2 query terms**

d2: "19 99 car wash"

- Contains: "covid" ✗
- Contains: "19" ✓
- **Matches: 1/2 query terms**

d3: "19 street covid testing facility is reopened next week"

- Contains: "covid" ✓
- Contains: "19" ✓
- **Matches: 2/2 query terms**

Scores (proportional to matches):

```
score(q, d1) ∝ 1
score(q, d2) ∝ 1
score(q, d3) ∝ 2
```

Ranking:

```
d3 (highest) > d1 = d2 (tied for lowest)
```

Answer: d3 has highest, d1 and d2 tied for lowest

From quiz solutions: d1, d2

Quiz Question 2 (From Lecture 2 Quiz)

Question: What about using vector space model with IDF?

Solution:

Now we include IDF: IDF(w) = log(N/df(w))

Document frequencies:

- "covid": appears in d1, d3 → df = 2
- "19": appears in d2, d3 → df = 2
- N = 3 (total documents)

IDF values:

```
IDF("covid") = log(3/2) = log(1.5) ≈ 0.405
IDF("19") = log(3/2) = log(1.5) ≈ 0.405
```

Scores:

d1: "covid patient"

```
score = TF("covid") × IDF("covid")
    = 1 × 0.405
    = 0.405
```

d2: "19 99 car wash"

```
score = TF("19") × IDF("19")
    = 1 × 0.405
    = 0.405
```

d3: "19 street covid testing facility is reopened next week"

```
score = TF("covid")×IDF("covid") + TF("19")×IDF("19")
    = 1×0.405 + 1×0.405
    = 0.810
```

Results:

- d3: 0.810 (highest)
- d1: 0.405 (tied lowest)
- d2: 0.405 (tied lowest)

Ranking:

```
d3 > d1 = d2
```

From quiz solutions: d1, d2

Quiz Question 3 (Feedback Language Model)

Question: Go to this notebook:
https://drive.google.com/file/d/1APsRt8T41kLxNchQv2uJ3Ws5aZN0aI4Hk/view?usp=sharing

Suppose query is "airport security" and lambda = 0.7.

Use this notebook to answer the following question: what is the probability of "airport" using the language model-based retrieval model? (4 decimal places)

Given:

```
θ^FB = λθq + (1-λ)θd
```

Solution:

Using the feedback documents d1, d2, ..., calculate:

```
P("airport"|θ^FB) = 0.7×P("airport"|query) + 0.3×P("airport"|feedback_docs)
```

Answer from quiz solutions: 0.38

Quiz Question 4 (Lambda Effect)

Question: How does increasing the value of lambda change the probability of "all" in theta^FB?

How does adding stop words to the query, e.g., "airport security" → "the airport security" change the probability of "all" in theta^FB?

Solution:

Part 1: Increasing λ

Formula:

```
θ^FB = λθq + (1-λ)θd
```

- Increasing λ → more weight on original query
- "all" is likely a stop word (appears in corpus but not query)
- Higher λ → **decreases** contribution from documents
- Higher λ → **decreases** "all" probability

Part 2: Adding stop words to query

If we add "the" to query:

- "the" will have higher probability in θ_q
- But "all" is not in the query
- The probability of "all" remains **the same**

Answer from quiz solutions: decrease, same

---

8. Video Resources

Probability & Statistics Foundations

1. **Bayes' Theorem (3Blue1Brown)**
   - Title: "Bayes theorem, the geometry of changing beliefs"
   - URL: https://www.youtube.com/watch?v=HZGCoVF3YvM
   - **Why:** Best visual intuition for Bayes' rule
   - **Duration:** 15 minutes
2. **Maximum Likelihood Estimation (StatQuest)**
   - Title: "Maximum Likelihood, clearly explained!!!"
   - URL: https://www.youtube.com/watch?v=XepXtl9YKwc
   - **Why:** Simple explanation with examples
   - **Duration:** 6 minutes
3. **Probability Distributions (Khan Academy)**
   - URL: https://www.khanacademy.org/math/statistics-probability
   - **Why:** Covers Bernoulli, Binomial, Poisson
   - **Duration:** Series of short videos

Information Retrieval Specific

4. **Stanford CS276 Lectures**
   - **Probabilistic IR:** Lecture 11
   - **Language Models:** Lecture 12
   - URL: http://web.stanford.edu/class/cs276/
   - **Why:** Detailed derivations from experts
   - **Duration:** 75 minutes each
5. **BM25 Explanation (Elastic)**
   - Title: "BM25 The Next Generation of Lucene Relevance"
   - URL: https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables
   - **Why:** Practical implementation perspective
   - **Duration:** Reading (15 min)

Mathematical Concepts

6. **Logarithms Review (Khan Academy)**
   - Why: Essential for understanding log-odds, IDF
   - URL: https://www.khanacademy.org/math/algebra2/x2ec2f6f830c9fb89:logs
   - **Duration:** 30 minutes
7. **Poisson Distribution (StatQuest)**
   - Title: "The Poisson Distribution, Clearly Explained!!!"
   - URL: https://www.youtube.com/watch?v=cPOChr_kuQs
   - **Why:** Understand the foundation of 2-Poisson model
   - **Duration:** 11 minutes
8. **EM Algorithm (Normalized Nerd)**
   - Title: "Expectation Maximization: how it works"
   - URL: https://www.youtube.com/watch?v=iQoXFmbXRJA
   - **Why:** Understand feedback language model estimation
   - **Duration:** 15 minutes

Quick Reference Guide

Key Formulas

RSJ:

```
score = ∑_{wi=1} log[αi(1-βi) / (βi(1-αi))]

where:
  αi = [count(wi=1, q, rel=1) + 0.5] / [count(q, rel=1) + 1]
  βi = [count(wi=1, q, rel=0) + 0.5] / [count(q, rel=0) + 1]
```

BM25:

```
score = ∑_{wi∈q} log(N/dfi) × [tfi(k1+1)] / [k1((1-b)+b|d|/avgdl) + tfi]

Typical parameters:
  k1 = 1.5
  b = 0.75
```

Language Model (Dirichlet):

```
score = ∑_{wi∈q,wi∈d} log(1 + count(wi,d)/[μ×P(wi|C)]) + log[μ/(μ+|d|)]

Smoothing:
  P_s(wi|d) = [count(wi,d) + μ×P(wi|C)] / [|d| + μ]

Typical parameter:
  μ = 2000
```

Feedback LM:

```
θq^FB = λθq + (1-λ)θq^F

Typical parameter:
  λ = 0.7
```

Parameter Typical Values

BM25:

```
k1 ∈ [1.2, 2.0]  (typical: 1.5)
b ∈ [0, 1]       (typical: 0.75)
```

Language Models:

```
μ ∈ [1000, 5000]  (typical: 2000) - Dirichlet
λ ∈ [0.6, 0.9]    (typical: 0.7)  - JM & Feedback
```

Model Comparison

Feature	RSJ	BM25	LM
Term Frequency	Binary	Saturated	Linear with smoothing
Doc Length Norm	No	Yes (pivoting)	Yes (implicit in smoothing)
IDF	Implicit in α,β	Explicit log(N/df)	Implicit in P(w|C)
Feedback	Easy	Hard	Easy
Parameters	Hard to estimate	2 parameters	1-2 parameters
Performance	Baseline	SOTA	SOTA with feedback

When to Use Each Model

Use RSJ when:

- You have relevance judgments
- You want theoretical foundation
- Binary occurrence is sufficient

Use BM25 when:

- You need state-of-the-art non-learning model
- You want robust, well-tested performance
- You have multi-field documents (BM25F)

Use Language Models when:

- You need query expansion/feedback
- You want flexible probabilistic framework
- You're willing to tune smoothing parameters

Study Tips for Midterm

What to Memorize

1. **Bayes' rule formula**

```
P(A|B) = P(B|A)P(A) / P(B)
```

2. **MLE formula**

```
α̂ = count(heads) / total
```

3. **Complete BM25 formula**

```
score = ∑ log(N/df) × [tf(k1+1)] / [k1((1-b)+b|d|/avgdl) + tf]
```

4. **Dirichlet smoothing formula**

```
P_s(w|d) = [count(w,d) + μP(w|C)] / [|d| + μ]
```

5. **What α_i and β_i represent in RSJ**
   - $\alpha_i = P(w_i=1|rel=1,q)$
   - $\beta_i = P(w_i=1|rel=0,q)$

What to Understand Conceptually

1. **Why we use odds ratio in RSJ derivation**
   - Cancels out P(d) term
   - Easier to work with products
2. **Why 2-Poisson models term burstiness**
   - One Poisson can't capture bimodal distribution
   - Words appear rarely (background) or frequently (topic)
3. **Why document length normalization is needed**
   - Long documents accumulate higher scores
   - Need to balance between no penalty and full penalty
4. **Why language models need smoothing**
   - Zero probability for unseen words
   - Would make entire query probability zero
5. **How feedback improves retrieval**
   - Expands query with related terms
   - Learns from pseudo-relevant documents

Practice Skills

1. **Calculate RSJ score by hand** (given α, β)
2. **Calculate BM25 score by hand** (given all parameters)
3. **Apply Dirichlet smoothing** (given document and collection)

4. **Explain when to use each model**
5. **Derive formulas from first principles**

Common Mistakes to Avoid

1. ❌ Confusing P(q|d) vs P(d|q)
2. ❌ Forgetting to add smoothing constant (0.5 in RSJ)
3. ❌ Wrong order of operations in BM25
4. ❌ Forgetting log when working with language models
5. ❌ Mixing up λ (feedback weight) and μ (smoothing parameter)

Step-by-Step Problem Solving

For RSJ problems:

1. Calculate α for each term (with +0.5 smoothing)
2. Calculate β for each term (with +0.5 smoothing)
3. For each term in document: log[α(1-β)/(β(1-α))]
4. Sum all terms

For BM25 problems:

1. Calculate IDF: log(N/df)
2. Calculate normalization: (1-b) + b(|d|/avgdl)
3. Calculate TF component: [tf(k1+1)] / [k1×norm + tf]
4. Multiply: IDF × TF component
5. Sum for all query terms

For Language Model problems:

1. For each query term, check if in document
2. If yes: use [count + μP(w|C)] / [|d| + μ]
3. If no: use [μP(w|C)] / [|d| + μ]
4. Take log of each probability
5. Sum all logs

Exam Strategy

1. **Read the entire question first**
    - Identify what model is being used

- Note what parameters are given

2. **Write down relevant formulas**
   - Don't try to remember while calculating
   - Write them at the top of your work
3. **Show your work**
   - Partial credit is real
   - Label intermediate steps
4. **Check units and ranges**
   - Probabilities must be [0,1]
   - Log of probabilities is negative
   - Scores can be any real number
5. **Manage your time**
   - Don't get stuck on one problem
   - Come back to difficult ones

Additional Resources

Official Course Materials

- **Stanford IR Book:** https://nlp.stanford.edu/IR-book/
  - Chapter 11: Probabilistic Information Retrieval
  - Chapter 12: Language Models
- **Overleaf Derivation:** https://www.overleaf.com/read/jbztxtnbzwcx
  - Complete mathematical derivations

Research Papers (Optional but Helpful)

1. **RSJ Model:**
   - Robertson & Sparck Jones (1976)
   - "Relevance weighting of search terms"
2. **BM25:**
   - Robertson et al. (1994)
   - "Okapi at TREC-3"
3. **Language Models:**
   - Ponte & Croft (1998)

- "A language modeling approach to information retrieval"

4. **Feedback Language Models:**
    - Zhai & Lafferty (2001)
    - "Model-based feedback in the language modeling approach to IR"

Practice Problems Online

- Stanford CS276 Homeworks
- TREC evaluation datasets
- Kaggle information retrieval competitions

Summary Checklist

Before the midterm, make sure you can:

- [ ] Explain Bayes' rule and apply it
- [ ] Calculate MLE from observations
- [ ] State the Probability Ranking Principle
- [ ] Derive RSJ model from first principles
- [ ] Calculate RSJ score by hand
- [ ] Explain eliteness hypothesis
- [ ] Understand 2-Poisson mixture model
- [ ] Calculate BM25 score by hand
- [ ] Explain document length normalization
- [ ] Understand saturation function
- [ ] State the paradigm shift in LM approach
- [ ] Explain why smoothing is necessary
- [ ] Calculate Dirichlet smoothing by hand
- [ ] Understand feedback language model
- [ ] Compare RSJ, BM25, and LM models
- [ ] Know when to use each model

Good luck on your midterm!

Last updated: [Current Date]
For questions or corrections, contact: [Your Email]