Relevance feedback

RELEVANCE FEEDBACK MOTIVATION, ROCCHIO'S ALGORITHM, PROBLEM WITH USING TRUE RELEVANCE

Introduction to Information Retrieval Motivation

What is Information Retrieval?

- The **process of finding relevant information from a large collection of resources** (e.g., documents, web pages, databases).
- Think of it as **finding a needle in a haystack**, but the "needle" isn't always identical to what you're looking for.

Why is "Motivation" Important?

- Understanding the challenges in **connecting users with the information they need.**
- Highlighting the reasons **why direct keyword matching often falls short.**
- This section explores the **fundamental problems that lead to less-than-ideal** search results.

Challenge 1- Keyword Mismatch

- **Searching** depends on matching **keywords between user-query and document**.
- This is the **most basic** form of search: **if your query words appear** in a document, that document is **considered relevant**.
- Searchers and document creators may **use different keywords to denote same 'concept'.**
- This is the **core problem. People describe things differently.**
- **Example (User Query):** "How to mend a broken vase?"
- **Relevant Document Keywords:** "**repairing ceramics**," "adhesive for pottery," "fixing cracked pottery."
- **Issue:** The document might be highly relevant, but because it doesn't use the exact words "mend" or "vase," a simple keyword match might miss it entirely. **The "concept" is the same, but the "vocabulary" is different.**

Consequence of Mismatch - Poor Retrieval Quality

- Vocabulary mismatch ⇒ poor retrieval quality
- When the terms used by the user and the document don't align, even for the same concept, the **search system fails to retrieve relevant results**
- **Poor Retrieval Quality Manifestations:**
- **Low Recall: Many relevant documents are *not* found.** (You missed too many needles).
  - ***Example:*** **Searching for "big cats"** and only **getting** results for "**tigers**" but **not** "**lions**" or "**jaguars**" because the system only matches "**tiger**" as a specific keyword.
- **Low Precision: Many irrelevant documents *are* found**. (You got a lot of hay with your needle).
  - ***Example:*** Searching for "**Apple**" (the company) and getting results for "**apple pie recipes**" or "**apple fruit nutrition**" because the system matches "apple" broadly without understanding context.

Challenge 2 - Users Don't Always Know What They Want

- **Users may not know what they are looking for,** but they'll know when they see it.
- This **often happens during** exploration or **research**. **A user** has a vague idea or is exploring a topic and **might not have the precise terminology.**
- **Example:** A user is doing preliminary research for a school project on "sustainable energy."
- **Initial Query:** "**energy**" (too broad)
- They might then see results for "**solar power**," "**wind turbines,"** "geothermal energy," "**hydroelectric**," and "biofuels."
- Seeing these specific examples helps them **refine their understanding and their subsequent searches**, even though they didn't explicitly know these terms at the outset. The "aha!" moment happens upon seeing the information.

Objective - Boost Recall

- "Boost recall: 'find me similar documents...'"
- Recall is the **proportion of *relevant* documents** that were **successfully retrieved**. **Boosting recall** means **finding *more* of the relevant items.**
- This is particularly important in domains like legal discovery, medical research, or patent searches, where missing even one relevant document could have significant consequences.
- **Example (Initial Low Recall):** A doctor searches for "**novel treatments for autoimmune diseases."**
- A **simple keyword search might return** articles with "**novel treatments**" and "**autoimmune**."
- However, if a new treatment is described using terms like "**cutting-edge therapies for immunological disorders**" or "**breakthrough interventions in immune system dysfunction**," a simple search might miss these highly relevant papers.
- The **goal is to expand the search to encompass these synonyms** and **related concepts** to ensure higher recall.

Solution - Better Representation of Information Need

- **Solution:** try to make a better representation of the information need
- **Instead of** just taking the user's literal **query**, **the system tries to understand the *underlying*** *intent* or ***concept***
- This "**information need**" is **richer than just a few keywords**.
- **Method:** "**expand the query by adding related words/phrases.**"
- This is a **common technique to improve retrieval.** It involves taking the original query and **programmatically adding synonyms**, **related terms**, or broader/narrower concepts to it.
- **Example (Initial Query):** "electric cars"
- **Expanded Query Terms:** "EVs," "**battery vehicles,**" "hybrid cars," "**zero-emission vehicles**," "charging stations," "**sustainable transport**."
- By adding these, the search system can now find documents that use any of these terms, even if they don't explicitly say "electric cars." This casts a wider net.

Issues with Query Expansion

- **Select which terms to add to query**
  - How do we know ***which*** **terms are truly related** and helpful, and **which are noise?**
  - ***Example:*** If a user searches for "**bank**" (**financial institution**), adding "**river bank**" or "**blood bank**" would be detrimental, leading to **irrelevant results** (**low precision**). The system needs to disambiguate.

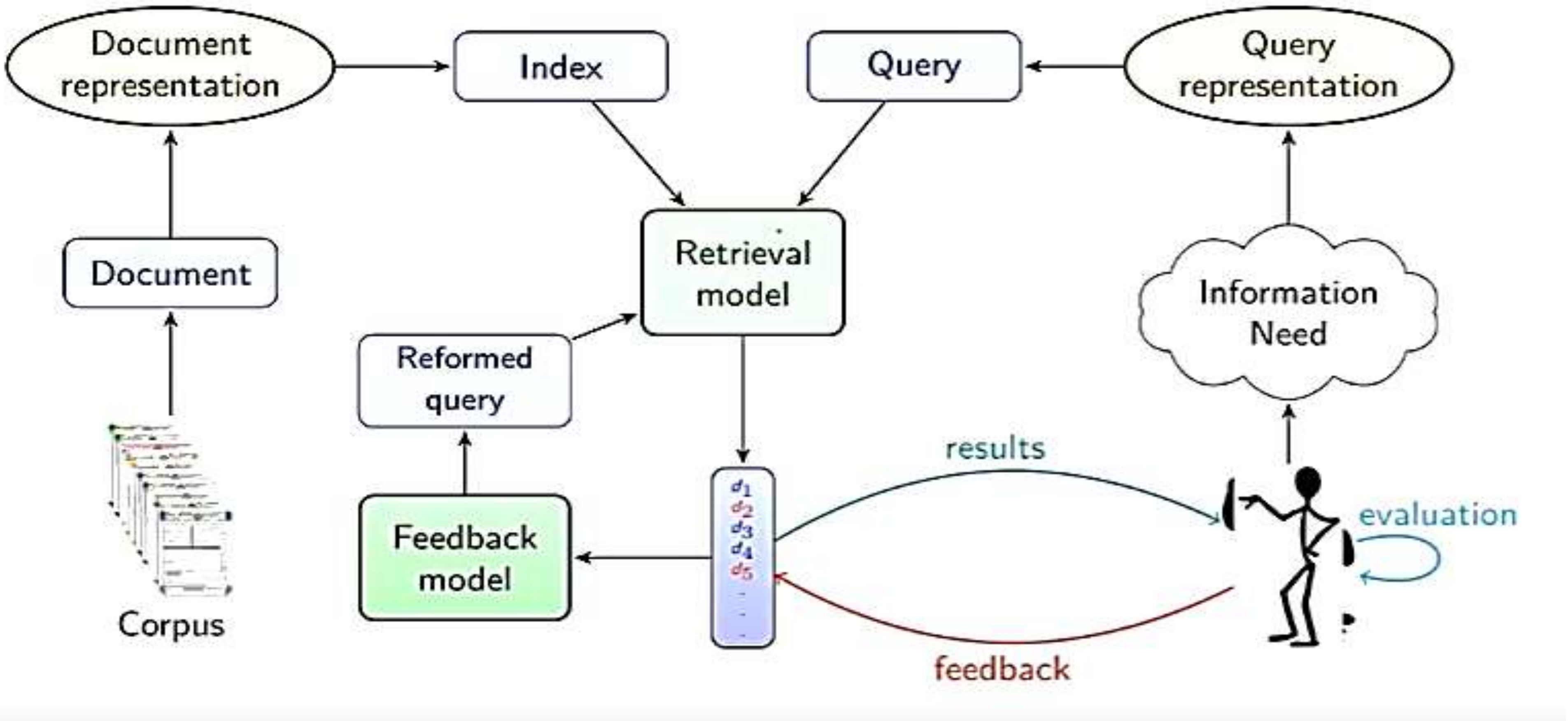
Issues with Query Expansion

- **Calculate weights for added terms**
  - Not **all added terms are equally important**. Some might be very close synonyms, others broader concepts.
  - ***Example:*** For "**electric cars**," "EVs" is **a very strong synonym**, perhaps deserving a high weight. "**Sustainable transport**" is related **but broader**, perhaps **deserving a lower weight**. If all terms are **weighted equally**, the search might **become too generic** or drift from the original intent.

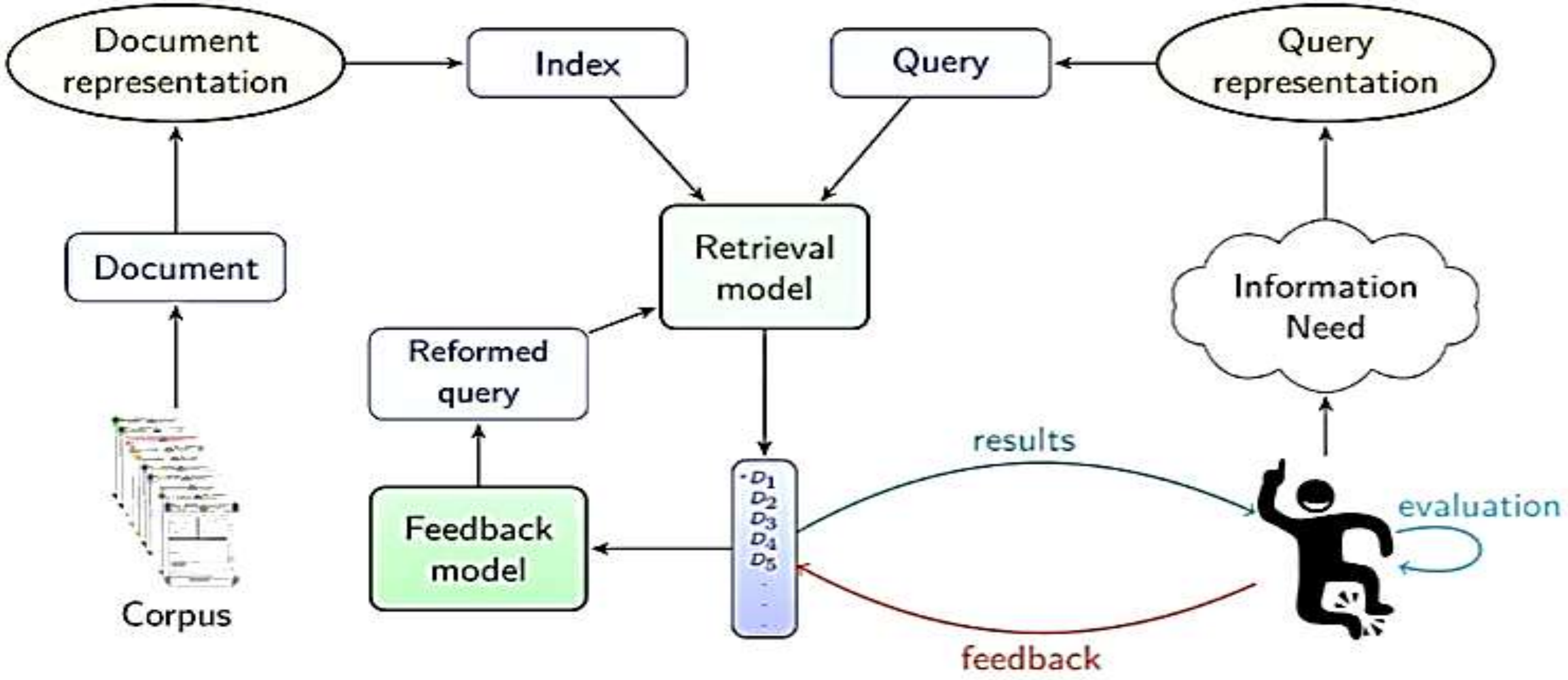
Next Steps in Information Retrieval Research

- ❖ Semantic search: understanding meaning beyond keywords.
- ❖ Relevance feedback: using user interaction to refine results.
- ❖ Personalized search: tailoring results to individual users.

Graphical Rep.



Graphical Rep.



Relevance Feedback: Basic Idea

- ✓ User issues a query.
- ✓ Search engine returns a set of documents.
- ✓ User marks some documents as relevant, some as non-relevant.
- ✓ Search engine uses this feedback information, together with collection statistics to formulate a better representation of the information need.
- ✓ Search engine performs retrieval with the reformulated query.
- ✓ The new set of documents returned by the engine, having (hopefully) better recall.
- ✓ Can iterate this: several rounds of relevance feedback.

Example of RF

TREC 2 Topic - 113 : *New Space Satellite Applications*

Example of RF

TREC 2 Topic - 113 : *New Space Satellite Applications*

Rank	Score	Document title
1	0.539	NASA Hasn't Scrapped Imaging Spectrometer
2	0.533	NASA Scratches Environment Gear From Satellite Plan
3	0.528	Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
4	0.526	A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
5	0.525	Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
6	0.524	Report Provides Support for the Critics of Using Big Satellites to Study Climate
7	0.516	Arianespace Receives Satellite Launch Pact From Telesat Canada
8	0.509	Telecommunications Tale of Two Companies

Example of RF

Initial query: *new space satellite applications*

Expanded query after relevance feedback

2.074	new	15.106	space
30.816	satellite	5.660	application
5.991	nasa	5.196	eos
4.196	launch	3.972	aster
3.516	instrument	3.446	arianespace
3.004	bundespost	2.806	ss
2.790	rocket	2.053	scientist
2.003	broadcast	1.172	earth
0.836	oil	0.646	measure

Example of RF

TREC 2 Topic - 113 : *New Space Satellite Applications*

Rank	Score	Document title
1	0.513	NASA Scratches Environment Gear From Satellite Plan
2	0.500	NASA Hasn't Scrapped Imaging Spectrometer
3	0.493	When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
4	0.493	NASA Uses 'Warm' Superconductors For Fast Circuit
5	0.492	Telecommunications Tale of Two Companies
6	0.491	Soviets May Adapt Parts of SS-20 Missile For Commercial Use
7	0.490	Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
8	0.490	Rescue of Satellite By Space Agency To Cost $90 Million

Rocchio's Algorithm

- A classic method for **relevance feedback** in Information Retrieval (IR) systems.
- **Goal:** To **improve the initial user query based on user judgments** of retrieved documents.
- **How: Modifies the original query vector to move it closer to relevant** documents and further from non-relevant documents in a vector space.

Rocchio's Algorithm

- **Underlying Model:** Assumes a **Vector Space Model (VSM)**.
  - Queries and documents are represented as vectors.
  - Each dimension of the vector corresponds to a term (word).
  - Values in the vector represent term weights (e.g., TF-IDF scores).

The Core Idea

- **Ideal Query Concept:** An ideal query should be maximally similar to all relevant documents and maximally dissimilar to all non-relevant documents.
- **Approximation:** Since the ideal query is unknown, Rocchio approximates it.
- **Mechanism:**
  - Starts with the original query.
  - **Adds a weighted** sum of vectors of documents the user marked as **relevant**.
  - **Subtracts a weighted** sum of vectors of documents the user marked as **non-relevant**.
- **Outcome:** A new, **reformulated query** used for a subsequent retrieval round.

The Rocchio Formula

- The reformulated query vector q' is calculated as:

$$q' = \alpha q + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_k \in D_{nr}} d_k$$

  - q: The original query vector.
  - Dr: Set of user-identified **relevant** documents.
  - |Dr|: Number of relevant documents.
  - Dnr: Set of user-identified **non-relevant** documents.
  - |Dnr|: Number of non-relevant documents.
  - dj: Vector for a relevant document j.
  - dk: Vector for a non-relevant document k.

Understanding the Weighting Parameters (α,β,γ)

- These parameters (typically between 0 and 1) control the influence of different components.
- **α (Original Query Weight):**
  - Controls how much the original query influences the new query.
  - High α: Reformulated query stays close to the initial query.
- **β (Relevant Documents Weight):**
  - Controls the influence of relevant documents.
  - High β: Relevant terms are strongly boosted, pulling the query towards relevant documents.
- **γ (Non-Relevant Documents Weight):**
  - Controls the influence of non-relevant documents.
  - High γ: Non-relevant terms are heavily penalized, pushing the query away from non-relevant documents.
- **Common Default:** $\alpha=1, \beta=0.75, \gamma=0.25$.
  - This prioritizes the original query and gives more weight to positive feedback (relevant documents) than negative feedback (non-relevant documents).
  - If no non-relevant documents are identified ($|D_{nr}|=0$), the γ term is ignored.

Numerical Example - Setup

- **Vocabulary:** { "car", "engine", "wheel", "road", "fast" }
  - Vector order: [car, engine, wheel, road, fast]
- **Document Vectors (Term Frequencies for simplicity):**
  - **Document 1 (D1):** "The car has a powerful engine and four wheels."
    $d_1 = [1, 1, 1, 0, 0]$
  - **Document 2 (D2):** "A fast car drives on the road."
    $d_2 = [1, 0, 0, 1, 1]$
  - **Document 3 (D3):** "The engine makes the car go fast."
    $d_3 = [1, 1, 0, 0, 1]$
- **Initial Query:** "fast car"
  $q = [1, 0, 0, 0, 1]$
- **User Feedback:**
  - D2 marked as **relevant** ($D_r = D2$)
  - D1 marked as **non-relevant** ($D_{nr} = D1$)
- **Parameters:** $\alpha = 1, \beta = 0.75, \gamma = 0.25$

Numerical Example - Solution

$$q' = \alpha q + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_k \in D_{nr}} d_k$$

1. **Original Query Term:**
   $q = [1, 0, 0, 0, 1]$
2. **Relevant Documents Term (D_r):**
   - Only D2 is relevant: $d_2 = [1, 0, 0, 1, 1]$
   - Average of relevant documents: $\frac{1}{|D_r|} \sum d_j = \frac{1}{1} d_2 = [1, 0, 0, 1, 1]$
3. **Non-Relevant Documents Term (D_{nr}):**
   - Only D1 is non-relevant: $d_1 = [1, 1, 1, 0, 0]$
   - Average of non-relevant documents: $\frac{1}{|D_{nr}|} \sum d_k = \frac{1}{1} d_1 = [1, 1, 1, 0, 0]$

Numerical Example - Solution

4. Apply Rocchio Formula:

$$q' = \alpha q + \beta \left(\frac{1}{|D_r|} \sum d_j \right) - \gamma \left(\frac{1}{|D_{nr}|} \sum d_k \right)$$

Substitute the values:

$$q' = 1 \cdot [1, 0, 0, 0, 1] + 0.75 \cdot [1, 0, 0, 1, 1] - 0.25 \cdot [1, 1, 1, 0, 0]$$

Perform multiplications:

$$q' = [1, 0, 0, 0, 1] + [0.75, 0, 0, 0.75, 0.75] - [0.25, 0.25, 0.25, 0, 0]$$

Perform component-wise addition and subtraction:

- Car: $1 + 0.75 - 0.25 = 1.5$
- Engine: $0 + 0 - 0.25 = -0.25$
- Wheel: $0 + 0 - 0.25 = -0.25$
- Road: $0 + 0.75 - 0 = 0.75$
- Fast: $1 + 0.75 - 0 = 1.75$

Reformulated Query Vector:

$$q' = [1.5, -0.25, -0.25, 0.75, 1.75]$$

Numerical Example – Interpretation and Usage

Reformulated Query Vector:

$$q' = [1.5, -0.25, -0.25, 0.75, 1.75]$$

Initial Query: "fast car"

$$q = [1, 0, 0, 0, 1]$$

Interpretation of Reformulated Query (q′):

- "Car" (1.5) and "Fast" (1.75) retain high positive weights (original query & relevant doc).
- "Road" (0.75) gains positive weight (appeared in relevant D2, not non-relevant D1) → indicates a good term.
- "Engine" (-0.25) and "Wheel" (-0.25) gain negative weights (appeared in non-relevant D1, not relevant D2) → pushed away from these terms.

How it's Used:

- This new query vector **q′ is used for a second round** of retrieval.
- Documents are **re-ranked** based on their similarity to q′.
- Documents with **terms like "road"** will now **be ranked higher.**
- Documents with **"engine" or "wheel"** will **be ranked lower** (unless balanced by other strong positive terms).

Iterative Process: This process can be repeated for several rounds of relevance feedback, continuously refining the query and improving retrieval performance.

Potential Problems and Limitations

1. **User Effort & Burden:**
   - Requires explicit relevance judgments from the user (marking documents as relevant/non-relevant).
   - This can be **tedious and time-consuming**, especially for large result sets. **Users** might be **unwilling** to **provide** extensive **feedback.**
2. **"Too Few" or "No" Relevant Documents in Initial Results:**
   - If **the initial search results contain very few or no truly relevant documents,** Rocchio's algorithm has little to no positive examples to learn from.
   - The reformulated query might not improve significantly, or could even **drift further from the user's actual information need**. This is known as the "cold start" problem for relevance feedback.

Potential Problems and Limitations

3. Negative Feedback Issues (Non-Relevant Documents):

- **Assumption of Non-Relevance: Users might mark a document as "non-relevant"** because it's *not exactly what they want*, rather than being truly irrelevant to the broader topic. This can lead to the algorithm incorrectly penalizing potentially useful terms.
- **Sparse Non-Relevant Feedback:** Sometimes users **only provide positive feedback**, or **very limited negative feedback.** The γ term might not be effective or used at all.

4. Assumes Global Relevance:

- Rocchio's algorithm **assumes that all relevant** documents share common characteristics (terms) that can be learned and amplified, and all **non-relevant documents share characteristics** that can be suppressed.
- This might not hold if the relevant documents are diverse or if the user's information need is ambiguous and can be satisfied by different "types" of relevant documents.
- ***Example:*** Query "**jaguar**." Relevant **documents could be about the car**, the **animal**, or the **operating system**. If the user marks a **car document as relevant** and an **animal** document as **non-relevant,** the algorithm might **over-emphasize car-related terms** and **completely miss other valid interpretations** if the user later changes their mind or has a mixed intent.

Potential Problems and Limitations

5. Optimal Parameter Selection (α,β,γ):

- The performance of the algorithm is **sensitive to the choice of the weighting** parameters.
- **Optimal values can vary depending on the dataset**, the type of **queries**, and even the **user's interaction patterns**. Choosing good parameters often requires empirical tuning.

6. Short Query Bias:

- Rocchio's algorithm can sometimes **over-expand a short query** by adding too many terms from relevant documents, potentially making the query too broad or losing its original focus (**leading to lower precision**).

Potential Problems and Limitations

7. Ignores Term Dependencies:

- Like the underlying Vector Space Model, Rocchio **treats terms as independent entities**. It **doesn't** inherently **understand** phrases or **semantic relationships** between words (e.g**.**, **"New York"** as a single concept, not just "New" and "York").

8. Drift Problem:

- In iterative feedback**, if a user mistakenly marks a non-relevant document as relevant**, or if the algorithm **consistently introduces slightly off-topic terms**, the **query can "drift" away from the original information** need over multiple rounds of feedback.