INFORMATION RETRIEVAL

By

Rohini Naik

OUTLINE

- Components of an IR model
- Text categorization
- IR processes and fields
- Vector Model
- Probabilistic Model
- Latent Semantic Indexing Model

Components of an IR model

- Indexing: Creating Document Representations
- Query Formulation: Creating Query Representations
- Matching the query representation with entity representations
- Selection
- Relevance Feedback and Interactive Retrieval

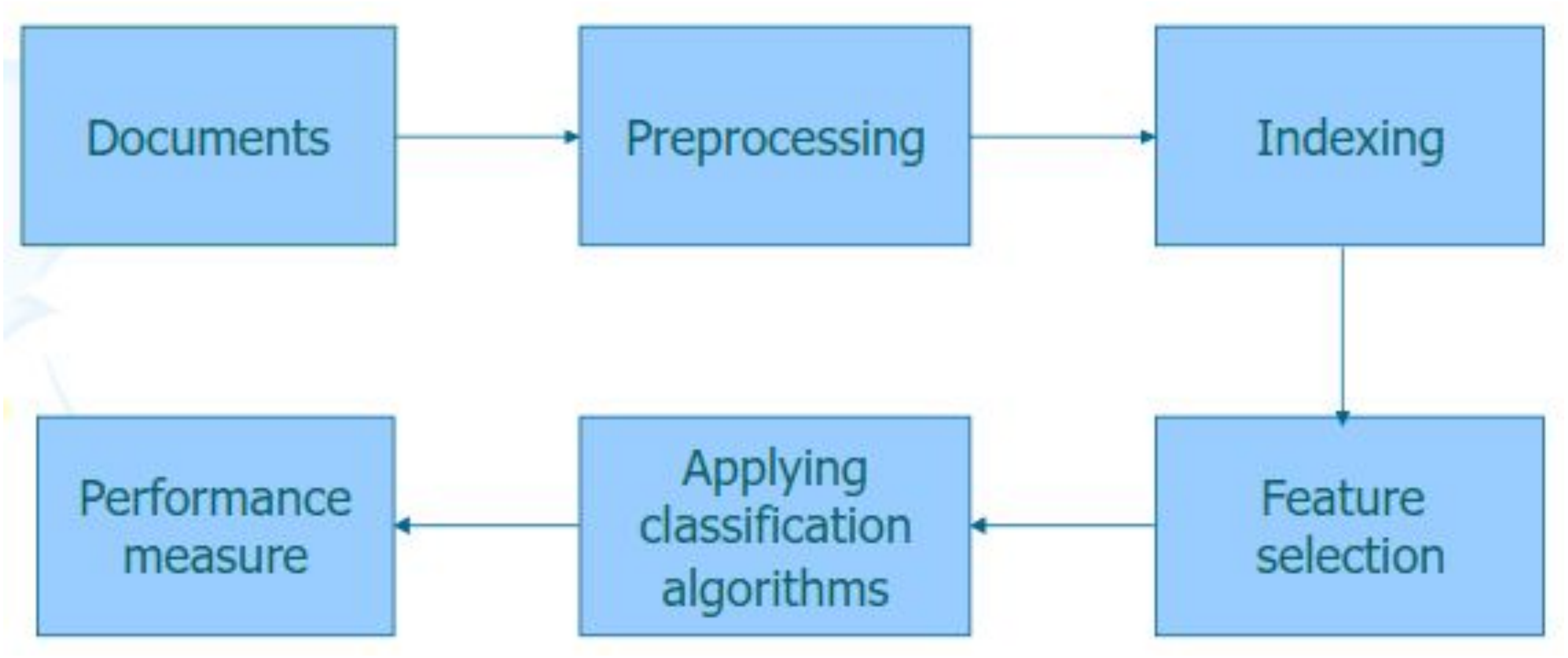
Text categorization

- The process of assigning documents to (mutually exclusive) classes of a classification is also known as text categorization.
- Analyzing the documents having similar features and clustering them in one group lead to identification of unique classes in which the documents belong. These are some initial steps of document classification

Applications of Text Categorization

- Organize web pages into hierarchies
- Domain specific information extraction
- Sort emails into different folders
- Find interests of users ..

Text Categorization-Framework



Preprocessing

- Transform documents into suitable representation
  - Remove HTML tags
  - Remove stop words
  - Perform word stemming…

Indexing

- Indexing by using different weighing schemes-
  - Boolean weighing
  - Word frequency weighing
  - TFIDF weighing
  - Entropy weighing

Feature Selection

- Remove non-informative terms from documents
  - Improve classification effectiveness
  - Reduce computational complexity

Classification Algorithms

- KNN
- Decision Tree
- Naïve Bayes
- ANN
- SVM
- Voting algorithms
- Rocchio’s Algorithm

Vector Model

Disadvantages of Boolean Model:

- Similarity function is Boolean

  Exact match only, no partial match

  Retrieved documents are not ranked

- All terms are equally important

  Boolean operation has much more influence than a critical word

- Query language is expressive but complicated

Vector Model

Documents and queries are assumed to be a part of a t-dimensional vector space, t is the no. if index terms (words, stems, phrases..etc)

Document Di I represented by a vector of index terms: Di=(di1, di2,..,dit)

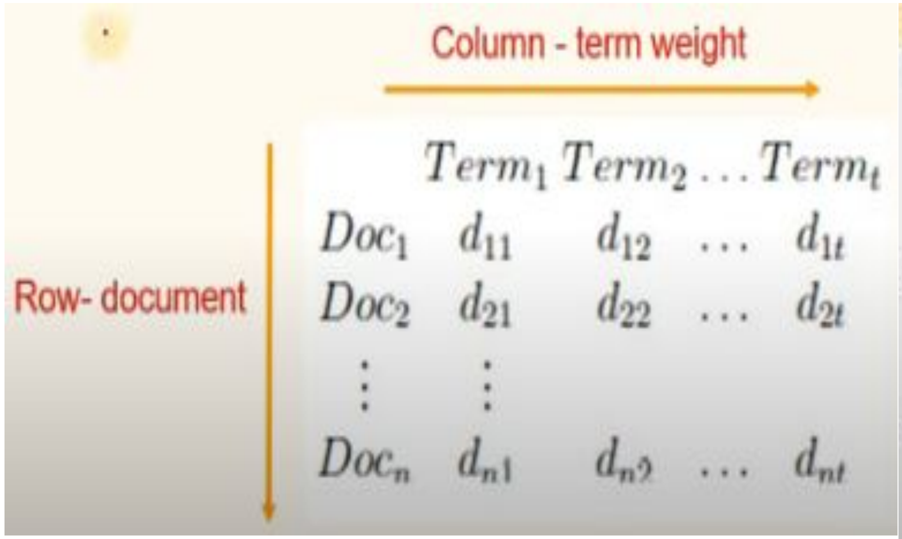
dij represents the weight of the jth term

Query Q is represented by a vector of t weights: Q=(q1,q2,…,qt)

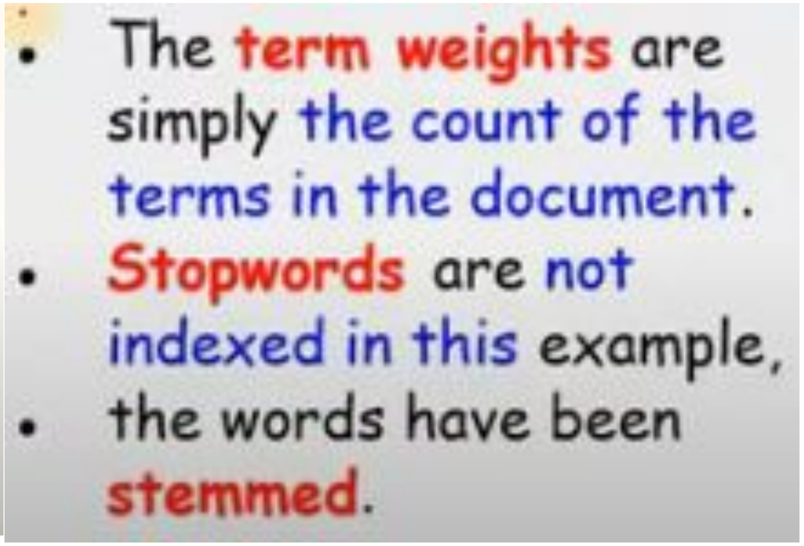
qj is the weight of jth term in the query

Vector Model

A document collection containing n documents can be represented as a matrix of term weights,

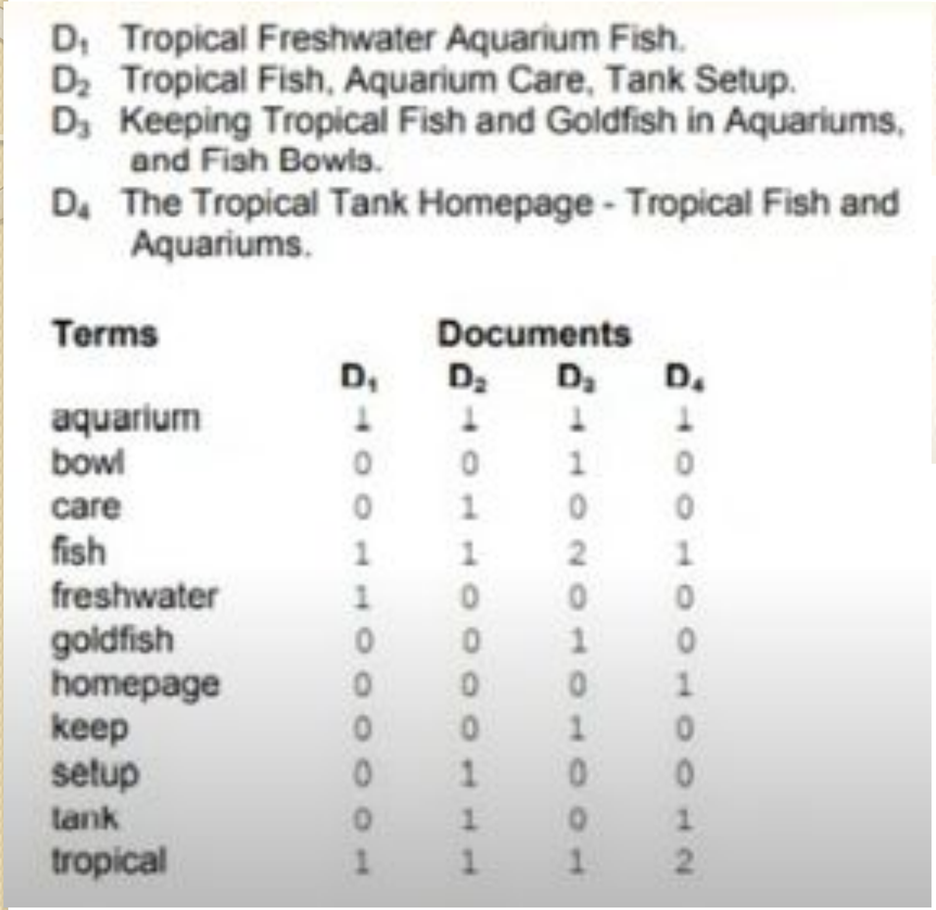


$$\begin{array}{cccccc} & Term_1 & Term_2 & \ldots & Term_t \\ Doc_1 & d_{11} & d_{12} & \ldots & d_{1t} \\ Doc_2 & d_{21} & d_{22} & \ldots & d_{2t} \\ \vdots & \vdots & & & \\ Doc_n & d_{n1} & d_{n2} & \ldots & d_{nt} \end{array}$$



- The term weights are simply the count of the terms in the document.
- Stopwords are not indexed in this example,
- the words have been stemmed.

Example Vector Model



D_1 Tropical Freshwater Aquarium Fish.
D_2 Tropical Fish, Aquarium Care, Tank Setup.
D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	Documents			
	D_1	D_2	D_3	D_4
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

Document D3, for example, is represented by the , vector (1, 1, 0, 2, 0, 1, 0, 1, 0, 0, 1).

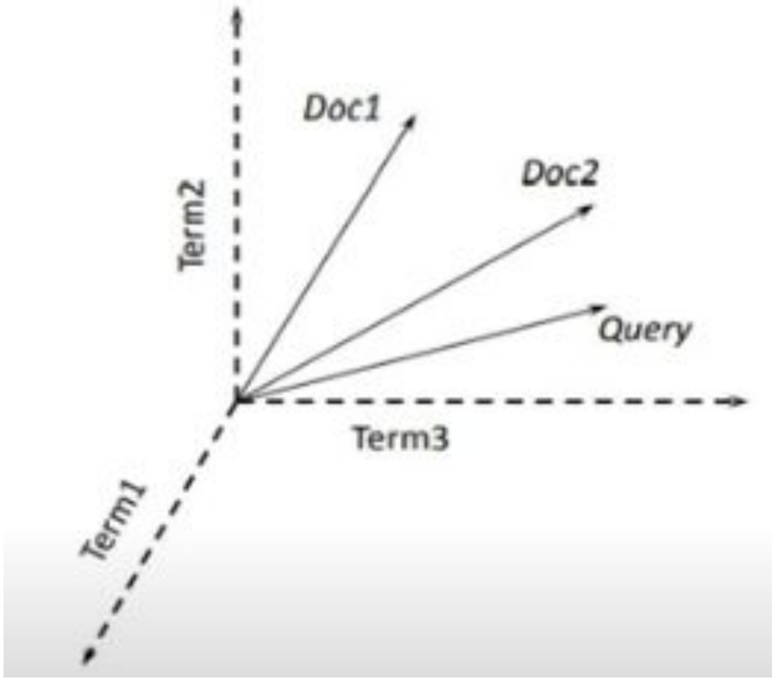
Query : "tropical fish", vector representation of query be (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1).

Example Vector Model

Documents could be ranked by computing the distance between the points representing the documents and the query

A similarity measure is used

Documents with high scores are the most similar to the query



Example Vector Model

The cosine correlation measures the cosine of the angle between the query and the document vectors.

When the vectors are normalized so that all documents and queries are represented by vectors of equal length, the cosine of the angle between two identical vectors will be 1 (the angle is zero), and for two vectors that do not share any non-zero terms, the cosine will be 0. The cosine measure is defined as,

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$

Example Vector Model

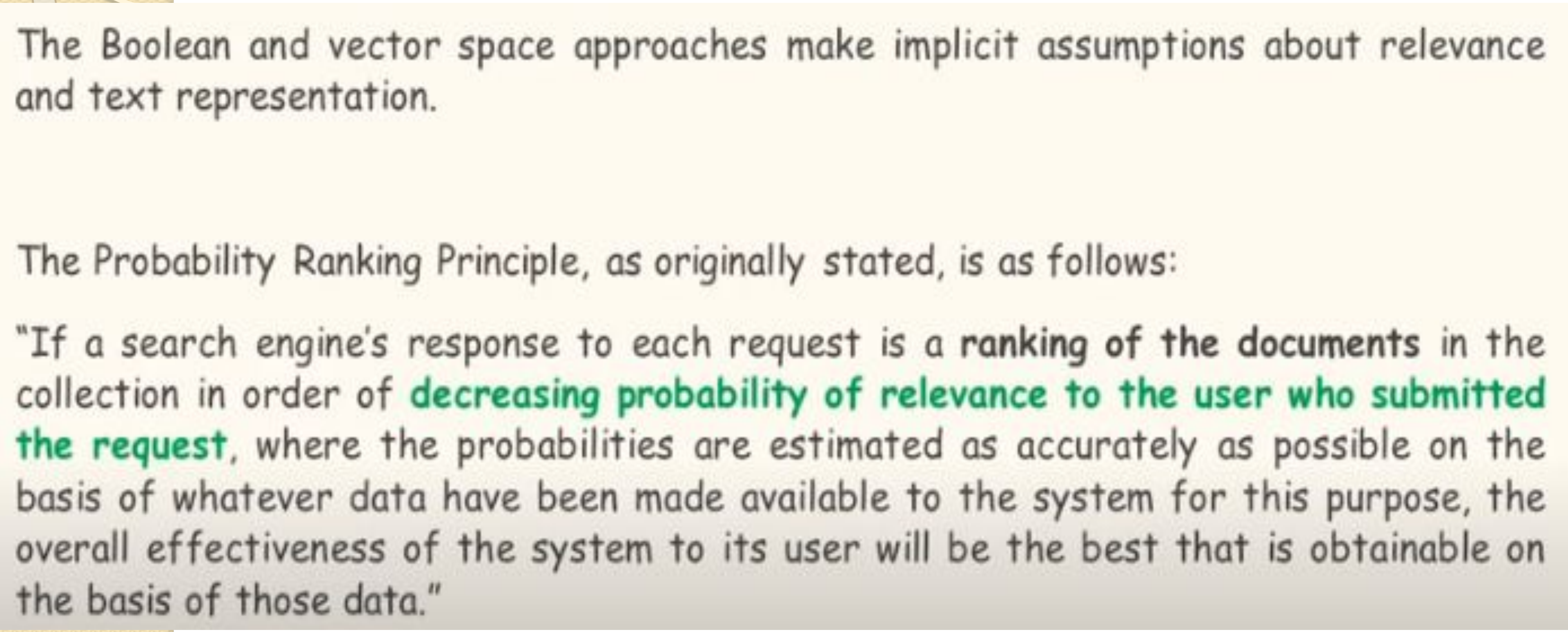
As an example, consider two documents
D1 = (0.5, 0.8, 0.3) and
D2 = (0.9, 0.4, 0.2) indexed by three terms, where the numbers represent term **weights**.

Given the query Q = (1.5, 1.0, 0) indexed by the same terms, the cosine measures for the two documents are:

$$\begin{aligned} Cosine(D_1, Q) &= \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}} \\ &= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87 \end{aligned}$$

$$\begin{aligned} Cosine(D_2, Q) &= \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}} \\ &= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97 \end{aligned}$$

Probabilistic Models



The Boolean and vector space approaches make implicit assumptions about relevance and text representation.

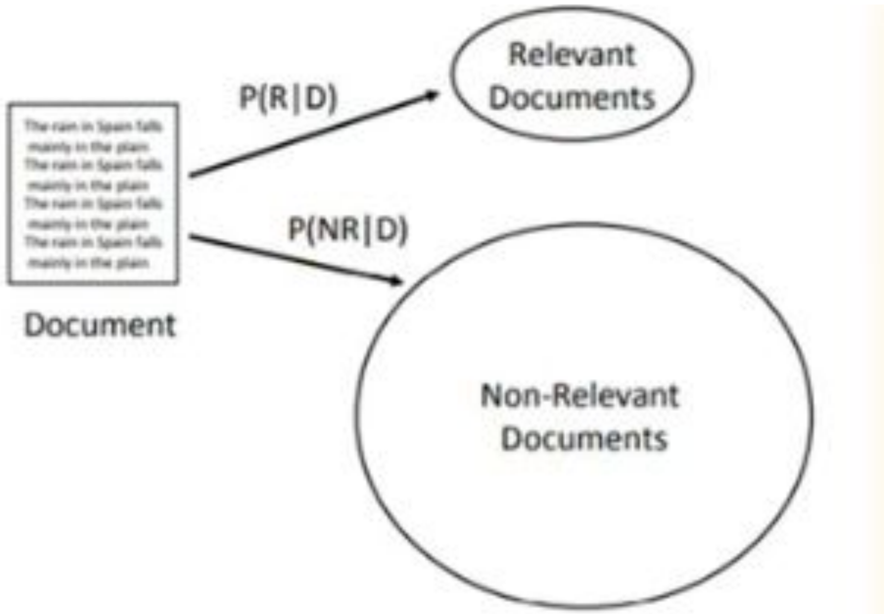
The Probability Ranking Principle, as originally stated, is as follows:

"If a search engine's response to each request is a **ranking of the documents** in the collection in order of **decreasing probability of relevance to the user who submitted the request**, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

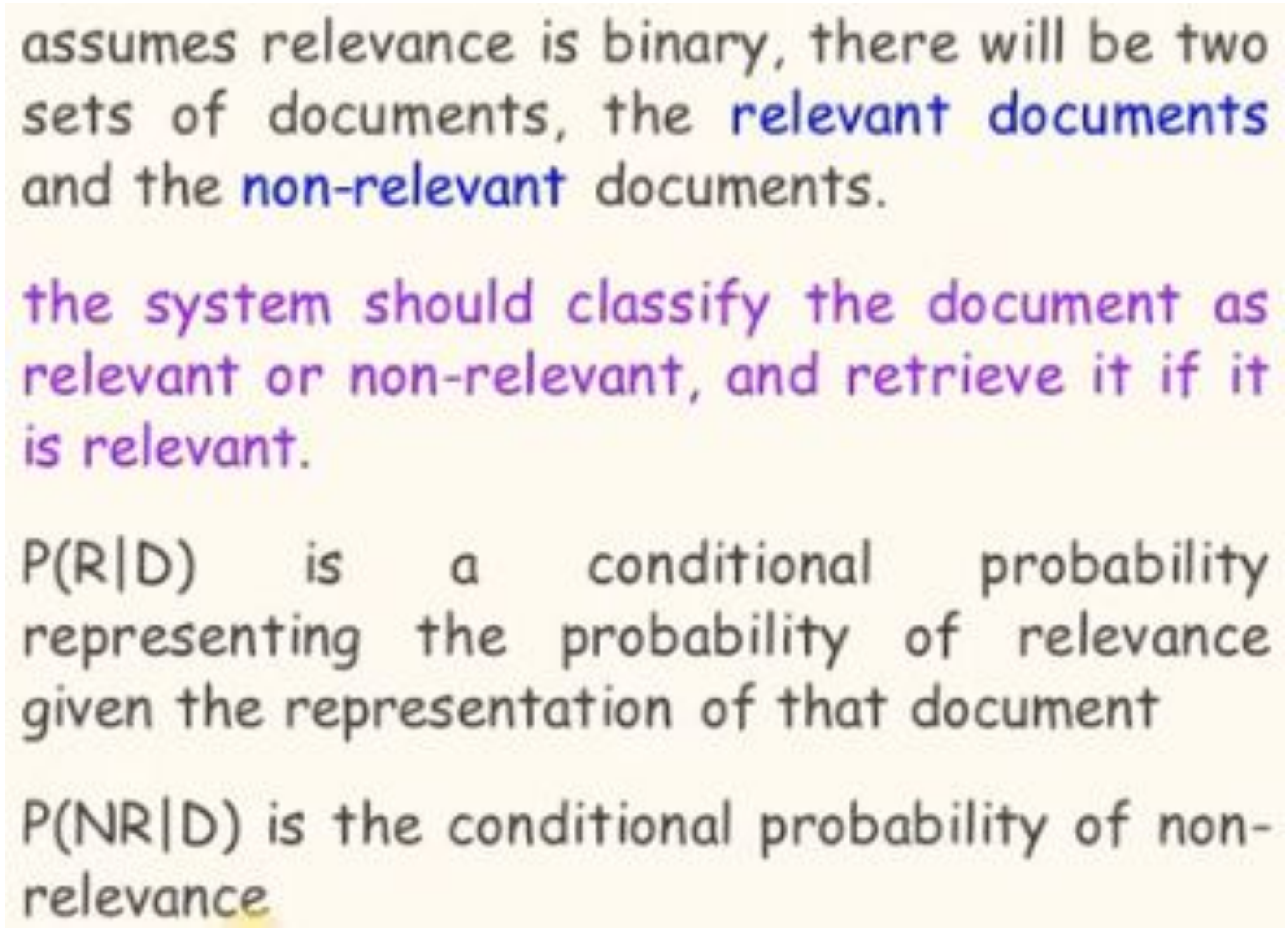
Probabilistic Models : IR as Classification

Probability Ranking Principle doesn't tell us how to calculate or estimate the probability of relevance.

simple probabilistic model based on treating information retrieval as a classification problem.



The rain in Spain falls mainly in the plain
The rain in Spain falls mainly in the plain
The rain in Spain falls mainly in the plain
The rain in Spain falls mainly in the plain
Document
P(R|D)
P(NR|D)
Relevant Documents
Non-Relevant Documents



assumes relevance is binary, there will be two sets of documents, the relevant documents and the non-relevant documents.

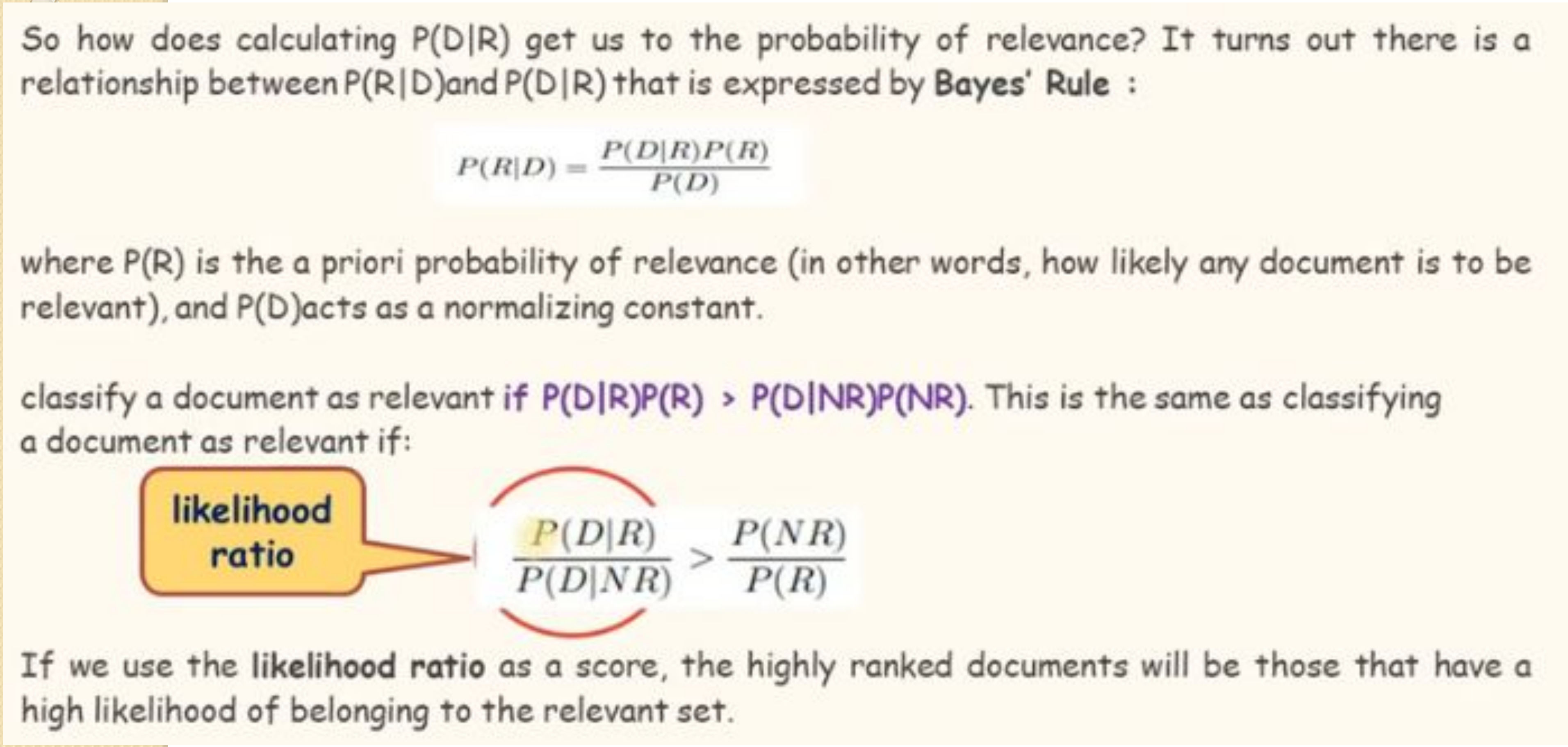
the system should classify the document as relevant or non-relevant, and retrieve it if it is relevant.

P(R|D) is a conditional probability representing the probability of relevance given the representation of that document

P(NR|D) is the conditional probability of non-relevance let's focus on $P(R|D)$..... $P(R|D) \longrightarrow P(D|R)$

if we had information about how often specific words occurred in the relevant set, then, given a new document, it would be relatively straightforward to calculate how likely it would be to see the combination of words in the document occurring in the relevant set.

Let's assume that the probability of the word "president" in the relevant set is 0.02, and the probability of "lincoln" is 0.03. If a new document contains the words "president" and "lincoln", we could say that the probability of observing that combination of words in the relevant set is $0.02 \times 0.03 = 0.0006$, assuming that the two words occur independently



So how does calculating P(D|R) get us to the probability of relevance? It turns out there is a relationship between P(R|D)and P(D|R) that is expressed by **Bayes' Rule** :

$$P(R|D) = \frac{P(D|R)P(R)}{P(D)}$$

where P(R) is the a priori probability of relevance (in other words, how likely any document is to be relevant), and P(D)acts as a normalizing constant.

classify a document as relevant if P(D|R)P(R) > P(D|NR)P(NR). This is the same as classifying a document as relevant if:

likelihood ratio

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

If we use the **likelihood ratio** as a score, the highly ranked documents will be those that have a high likelihood of belonging to the relevant set.

Thank You!!