Probabilistic and Language Models

24 Dec, 2024

Outline

Probabilistic and Language Models [4 Hours]

5.1. Probability ranking principle

5.2. Binary independence model

5.3. Language models for IR

Vector space model

- Simple, practical and mathematically based approach
- Provides partial matching and ranked results.
- Problems
  - No notion of relevance
  - Missing syntactic information (e.g. phrase structure, word order, proximity information).
  - Missing semantic information –
    - word sense
    - Assumption of term independence. ignores synonym.
  - Lacks the control of a Boolean model (e.g., requiring a term to appear in a document).
    - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

Missing syntactic information (e.g. phrase structure, word order, proximity information).

Phrase Structure:

Models treat queries and documents as bags of words, ignoring relationships between terms.

Example:

Query: “machine learning”.

A document with “learning about machines” may score equally as one with the exact phrase “machine learning”.

Missing syntactic information (e.g. phrase structure, word order, proximity information).

Word Order:

The order in which words appear in the query or document is not considered.

Example:

Query: “Apple pie recipe”.

A document containing “recipe for pie made with apples” might be treated similarly to “Apple recipe for making pie”, even though contextually they are different.

Missing syntactic information (e.g. phrase structure, word order, proximity information).

Proximity Information:

The closeness of query terms within a document is ignored.

Example:

Query: “solar energy”.

A document where “solar” appears in the first paragraph and “energy” in the last may be scored the same as one where they are adjacent.

Vector space model

- Simple, practical and mathematically based approach
- Provides partial matching and ranked results.
- Problems
  - Missing syntactic information (e.g. phrase structure, word order, proximity information).
  - Missing semantic information –
    - word sense
    - Assumption of term independence. ignores synonym.
  - Lacks the control of a Boolean model (e.g., requiring a term to appear in a document).
    - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

Missing semantic information

Word Sense Ambiguity:

Words with multiple meanings are treated as a single token, leading to irrelevant results.

Example:

Query: "bank".

A user searching for "riverbank" might retrieve documents about financial institutions.

Synonymy:

Different words with similar meanings are not linked, causing relevant documents to be missed.

Example:

Query: "car".

A document about "automobiles" might not rank high because the exact word "car" doesn't appear.

Polysemy:

A single word with multiple related meanings is misinterpreted without proper context.

Example:

Query: "apple".

Results might include both "the fruit" and "the tech company".

...

Context Ignorance:

The context in which terms are used is ignored, leading to shallow matches based on exact keywords.

Example:

Query: "growth of plants".

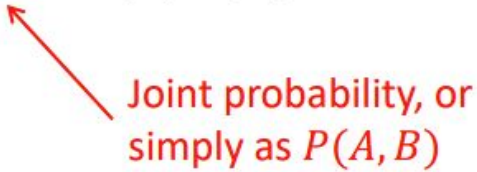
A document about "economic growth" may rank higher due to shared term "growth".

Basic concepts in probability

- Random experiment
  - An experiment with uncertain outcome (e.g., tossing a coin, picking a word from text)
- Sample space (S)
  - All possible outcomes of an experiment, e.g., tossing 2 fair coins, S={HH, HT, TH, TT}
- Event (E)
  - $E \subseteq S$, E happens iff outcome is in S, e.g., E={HH} (all heads), E={HH,TT} (same face)
  - Impossible event ({}), certain event (S)
- Probability of event
  - $0 \leq P(E) \leq 1$

Essential probability concepts

- Probability of events
  - Mutually exclusive events
    - $P(A \cup B) = P(A) + P(B)$
  - General events
    - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
  - Independent events
    - $P(A \cap B) = P(A)P(B)$



Essential probability concepts

- Conditional probability
  - $P(B|A) = P(A,B)/P(A)$
  - Bayes' Rule: $P(B|A) = P(A|B)P(B)/P(A)$
  - For independent events, $P(B|A) = P(B)$
- Total probability
  - If $A_1, \ldots, A_n$ form a non-overlapping partition of S
    - $P(B \cap S) = P(B \cap A_1) + \cdots + P(B \cap A_n)$
    - $P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B|A_1)P(A_1)+\cdots+P(B|A_n)P(A_n)} \propto P(B|A_i)P(A_i)$
    - This allows us to compute $P(A_i|B)$ based on $P(B|A_i)$

Interpretation of Bayes' rule

Hypothesis space: $H = \{H_1, \ldots, H_n\}$, Evidence: E

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{P(E)}$$

If we want to pick the most likely hypothesis H*, we can drop $P(E)$

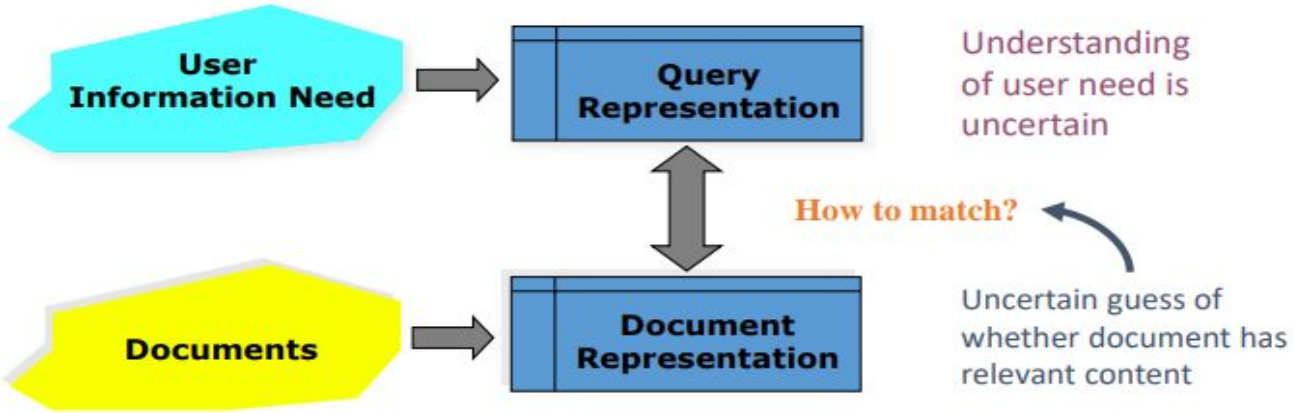
Posterior probability of H_i — **Prior probability of H_i**

$$P(H_i \mid E) \propto P(E \mid H_i)P(H_i)$$

Likelihood of data/evidence given H_i

Why probabilities in IR?

- In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.



Probabilities provide a principled foundation for uncertain reasoning.

Can we use probabilities to quantify our uncertainties?

The 1/0 loss case

- The 1/0 loss case refers to a specific scenario in decision theory or information retrieval where the loss (or cost) function is binary.
- It assigns a loss of 1 for an error and 0 for a correct decision.

In the context of information retrieval (IR):

Loss = 1:

- This occurs when an incorrect decision is made, such as:
  - False Positive: Retrieving a non-relevant document.
  - False Negative: Failing to retrieve a relevant document.

Loss = 0:

- This occurs when the decision is correct, such as:
  - A relevant document is retrieved.
  - A non-relevant document is not retrieved.

…

- This simplifies the evaluation process because all errors are treated equally, and no additional weights are given to different types of mistakes.
- If we are ranking documents for a query, in the 1/0 loss case:
  - Returning a non-relevant document counts as 1 (error).
  - Missing a relevant document also counts as 1 (error).
  - Correctly identifying relevance or non-relevance counts as 0 (no loss).

- This loss function is simple and does not differentiate between the types of errors.
- A mistake (false positive or false negative) costs the same (1), and a correct decision costs nothing (0).

Limitations:

While the 1/0 loss case simplifies evaluation, it may not always reflect real-world scenarios. For instance:

- Missing a relevant document (false negative) might be more costly than retrieving a non-relevant one (false positive), such as in **medical document retrieval**.

- In such cases, we need a more general loss function that assigns different costs to false positives and false negatives, as discussed with retrieval costs.

Probability ranking Principle (PRP) as stated originally - Roberton, 1997:

“ If a reference retrieval system’s response/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.”

- Rank by probability of relevance leads to the optimal retrieval effectiveness

Let's dissect the PRP

- rank documents … by probability of relevance
  - P (relevant | document) ... P (R | D)
- estimated as accurately as possible
  - P_{est} (relevant | document) → P_{true} (rel | doc) in some way
- based on whatever data is available to system
  - P_{est} (relevant | document, session, context, user profile, …)
- best possible accuracy one can achieve with that data
  - recipe for a perfect IR system: just need P_{est} (relevant | …)
  - strong stuff, can this really be true?

PRP Under 1/0 Loss

In the simplest case of PRP (with 1/0 loss), the system **minimizes the total loss** by returning documents in the order of their probability of relevance $P(R=1|d,q)$, where:

- R=1 indicates that the document d is relevant to the query q.
- R=0 indicates that the document d is non-relevant.

The PRP says to rank all documents in **decreasing order of the probability of relevance** $P(R=1|d,q)$ where:

- R=1: The document d is relevant to the query q.
- $P(R=1|d,q)$: Probability that document d is relevant given the query q.

...

The **PRP rule**: Rank all documents d in **decreasing order** of P(R=1|d,q).

If we retrieve the top k documents (as the user examines k results), the PRP ensures that these k documents are the most likely to be relevant,

- thereby minimizing the risk of loss under the 1/0 loss framework.

Bayes Optimal Decision Rule

The Bayes Optimal Decision Rule further refines this principle:

d is relevant iff $P(R=1|d,q)>P(R=0|d,q)$

This rule states that a document d is classified as **relevant** if its probability of relevance exceeds its probability of being non-relevant.

Optimality of the PRP

- **Theorem**: The PRP is **optimal** under 1/0 loss because it **minimizes the expected loss** (also known as **Bayes Risk**).
- **Bayes Risk**:
  - The expected loss of a decision system when it is based on probabilistic reasoning.
- **Condition for Optimality**:
  - The PRP assumes that the probabilities $P(R=1 \mid d,q)$ are **known correctly**.
  - In practice, these probabilities are **never perfectly known**, but the PRP still serves as a solid foundation for IR models.

The proof of this theorem can be found in statistical works like **Ripley (1996)**.

PRP with Retrieval Costs

While the 1/0 loss assumes equal penalties for all errors, in real-world systems, errors may have **different costs**:

- **C1**: Cost of **not retrieving a relevant document** (false negative).
- **C0**: Cost of **retrieving a non-relevant document** (false positive).

To account for these retrieval costs, the PRP can be adjusted as follows:

For a document ddd to be retrieved **next**, it must satisfy:

$$C0 \cdot P(R=0 \mid d) - C1 \cdot P(R=1 \mid d) \leq C0 \cdot P(R=0 \mid d') - C1 \cdot P(R=1 \mid d')$$

Where:

- $P(R=1 \mid d)$: Probability that ddd is relevant.
- $P(R=0 \mid d)$: Probability that ddd is not relevant.
- d': Other documents not yet retrieved.

Intuition:

- If the weighted cost of retrieving d is **lower** than that of any other document d′, then ddd is retrieved next.
- This formalizes the decision-making process where **different costs** are associated with false positives and false negatives.

Why Retrieval Costs Matter ??

- **Real-World Scenarios**:
  - In medical information retrieval:
    - Missing a relevant document (false negative) may have a **higher cost** than retrieving an irrelevant document (false positive).
  - In search engines:
    - Retrieving irrelevant documents may waste user time and resources, leading to a higher emphasis on precision.
- By incorporating retrieval costs into the PRP, we can:
  - **Model different costs** at the retrieval stage.
  - Balance system performance to align with user needs.

Key findings

PRP provides an optimal ranking mechanism under simple assumptions (1/0 loss).

By extending PRP with retrieval costs, the model can account for practical concerns like false positive penalties and false negative costs.

Even though probabilities are rarely known perfectly, the PRP offers a foundation for building practical and effective IR models.

Probabilistic Model: IR as classification

Let's assume relevance is binary, there will be two set of document,

1: relevant

0: non-relevant

Now, the system should classify the documents as relevant or non-relevant

And retrieve it if it is relevant.

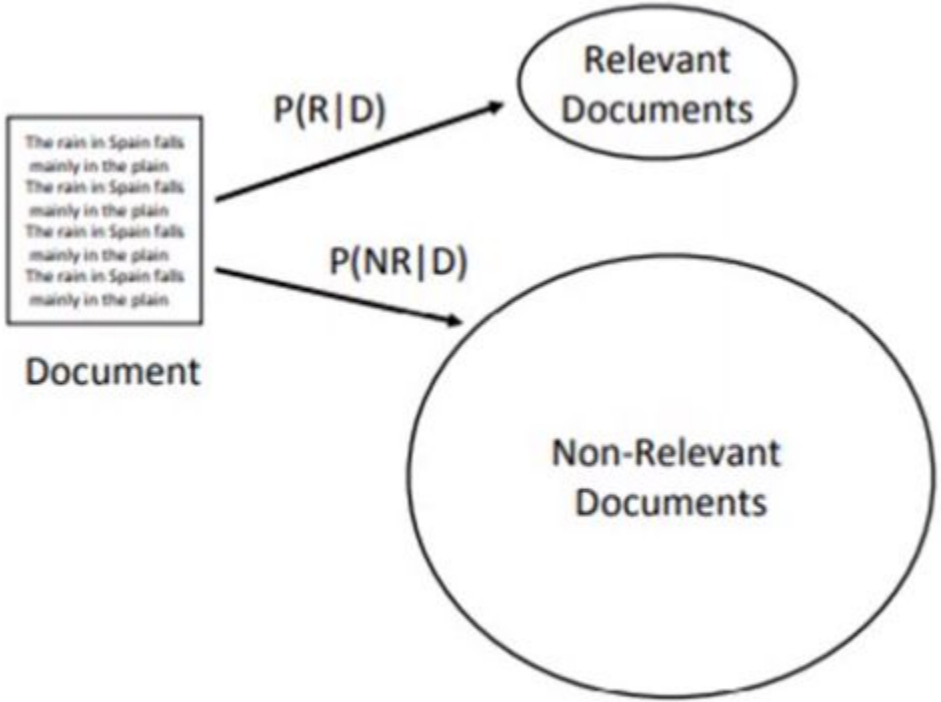


Fig. 7.3. Classifying a document as relevant or non-relevant

…

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

P(NR|D) is a conditional probability of non-relevance…

Document D is relevant if P(R|D) > P(NR|D)

Now,

Let’s focus n P(R|D) -> to calculate this we first calculate P(D|R)

I.e, if we had information about how often specific words occurred in the relevant sets, then, given a new document, it would be relatively straightforward to calculate how likely it would be to see the combinations 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",

- the probability of observing that combination of words in the relevant set is 0.02 x 0.03 = 0.0006,
- assuming that the **two words occur independently**"

How does calculating P(D|R) get us to the probability of relevance?

...

There is a relationship between P(D|R) and P(R|D) expressed by Bayes' rule:

Assume : P(R=1|D,Q) or P(R=1|D) or P(R|D)

$$P(R=1|D) = \frac{P(D|R=1)P(R=1)}{P(D)}$$

$$P(R=0|D) = \frac{P(D|R=0)P(R=0)}{P(D)}$$

If a user likes document d, how likely would the user enter query q (in order to retrieve d)?

Where, P(R) is the a-priori probability of relevance (how likely any document is to be relevant)

Estimating the probabilities

Binary Independence Model (BIM) - the simplest model

Assumptions:

- Relevance of each document is independent of relevance of other documents
  - Which is in real world not **TRUE.**
- Boolean model of relevance (either relevant or not relevant).
- Order documents by decreasing probability of relevance.

BIM

The BIM makes the following key assumptions:

1. **Binary Term Representation**:
   - Each term in a document is represented as a **binary variable**.
   - A term ti is either **present** (1) or **absent** (0) in a document d.
   - There is **no consideration of term frequency**—only presence or absence matters.

2. **Term Independence**:
   - The presence or absence of a term ti in a document is **independent** of the presence or absence of any other term tj.
     Formally: $P(t_i \mid t_j) = P(t_i)$ for all $i \neq j$
   - The occurrence of terms is independent of one another.

...

3. **Relevance Independence**:

- The probability of a term ti appearing in a document depends **only on the relevance** of the document, not on other terms.
- Therefore:
  - $P(t_i | R=1)$: Probability that term ti occurs in relevant documents.
  - $P(t_i | R=0)$: Probability that term ti occurs in non-relevant documents.

Bayes’ Rule and BIM

To compute the probability $P(R=1 \mid d,q)$P(R = 1 \mid d, q)$P(R=1 \mid d,q)$, BIM uses **Bayes' Rule**:

$P(R=1 \mid d,q)=P(d \mid R=1,q) \cdot P(R=1 \mid q) / P(d \mid q)$.
Or
$P(R \mid d)=P(d \mid R) \cdot P(R) / P(d)$.

However, the exact computation is often impractical because it requires knowing $P(d \mid q)$ or $p(d)$.
Instead, BIM simplifies this using the assumptions of term independence and binary representation.

The Log-Odds of Relevance

The BIM compares the **odds** that a document is relevant versus non-relevant:

$$\text{Odds}(R = 1 \mid d, q) = \frac{P(R = 1 \mid d, q)}{P(R = 0 \mid d, q)}.$$

Taking the **logarithm** of the odds and By substituting and simplifying under the BIM assumptions, the log-odds score becomes:

$$\log \text{Odds}(R = 1 \mid d, q) = \sum_i \log \frac{P(t_i \mid R = 1)}{P(t_i \mid R = 0)} \cdot x_i + C,$$

Where:

- x_i: Binary variable (1 if t_i is present in d, 0 otherwise).
- $\frac{P(t_i|R=1)}{P(t_i|R=0)}$: The **weight** of term t_i, which measures its contribution to relevance.
- C: A constant that does not affect ranking.

...

The log-odds formula can be rewritten as:

$$\text{Score}(d) = \sum_{t_i \in q} w_i \cdot x_i$$

Where:

- $w_i = \log \frac{P(t_i|R=1)}{P(t_i|R=0)}$ is the **weight** of term t_i.
- The score for a document d is the **sum of the weights** of the query terms t_i that appear in d.

Documents are ranked based on their **scores**:

- A higher score means the document is more likely to be relevant.

Documents are ranked based on their scores:

- A higher score means the document is more likely to be relevant.

Why is IR Viewed as Classification?

In classification:

- A document is assigned a class (Relevant or Non-Relevant) based on its probability.
- Features used for classification are the terms in the document.
- Probabilistic models estimate likelihoods for each class based on observed data.

Steps:

- Train the model using relevant and non-relevant documents.
- Score new documents by estimating probabilities.
- Rank documents based on their relevance scores.

Limitations of BIM

1. Binary Term Representation:
   - It ignores term frequency, which can be critical for relevance.

2. Term Independence:
   - Terms often co-occur in meaningful ways (e.g., "information retrieval"), but BIM assumes independence.

3. Equal Importance of Query Terms:
   - BIM treats all query terms equally, whereas some terms may be more significant.

Advantages of BIM

- Theoretical Foundation: Based on probability theory and the PRP.
- Efficiency: Simple to compute using term presence/absence.
- Interpretability: Log-odds weights provide a clear measure of term importance.

Example

1. Define a Query

A query represents the user's information need. For example:

- Query: **"climate change policy"**

This query contains three terms: t1="climate", t2="change", t3="policy"

Document ID	Terms in Document	Relevant (R)
d_1	"climate change policy"	1 (Relevant)
d_2	"climate action goals"	1 (Relevant)
d_3	"policy reform government"	0 (Nonrelevant)
d_4	"economic change policy"	0 (Nonrelevant)
d_5	"climate change impacts"	1 (Relevant)
d_6	"government reform policy"	0 (Nonrelevant)

We estimate $P(t_i \mid R = 1)$ and $P(t_i \mid R = 0)$ for each term in the query:

- $P(t_i \mid R = 1)$: Fraction of **relevant documents** containing term t_i.
- $P(t_i \mid R = 0)$: Fraction of **nonrelevant documents** containing term t_i.

Term	$P(t_i \mid R = 1)$	$P(t_i \mid R = 0)$
"climate"	$\frac{2}{3} = 0.67$	$\frac{0}{3} = 0.00$
"change"	$\frac{2}{3} = 0.67$	$\frac{1}{3} = 0.33$
"policy"	$\frac{2}{3} = 0.67$	$\frac{2}{3} = 0.67$

For each term, compute the weight w_i using:

$$w_i = \ln\left(\frac{P(t_i \mid R = 1)}{P(t_i \mid R = 0)}\right)$$

Term	$P(t_i \mid R = 1)$	$P(t_i \mid R = 0)$	w_i
"climate"	0.67	0.00	$\ln(\infty) = \text{Very Large}$
"change"	0.67	0.33	$\ln(2) \approx 0.69$
"policy"	0.67	0.67	$\ln(1) = 0.00$

The BIM score for a document is the **sum of weights** of query terms present in the document:

$$\text{Score}(d) = \sum_{t_i \in \text{query}} w_i \cdot \mathbf{1}[t_i \in d]$$

Document Scoring Table:

Document ID	Terms in Document	Contains "climate"?	Contains "change"?	Contains "policy"?	Total Score
d_1	"climate change policy"	Yes	Yes	Yes	Very Large + 0.69 + 0.00 = **Very Large**
d_2	"climate action goals"	Yes	No	No	Very Large + 0.00 + 0.00 = **Very Large**
d_3	"policy reform government"	No	No	Yes	0.00 = **0.00**
d_4	"economic change policy"	No	Yes	Yes	0.69 + 0.00 = **0.69**
d_5	"climate change impacts"	Yes	Yes	No	Very Large + 0.69 = **Very Large**
d_6	"government reform policy"	No	No	Yes	0.00 = **0.00**

...

Rank the Documents

Documents are ranked by their scores:

1. **Top-ranked documents**: d1,d2,d5
   These contain "climate," the most discriminative term (high weight).
2. **Moderately ranked document**: d4
   Contains "change" but lacks "climate."
3. **Lowest-ranked documents**: d3,d6
   These lack both "climate" and "change," contributing very little to relevance.

...

- BIM effectively ranks documents based on the probability of relevance, leveraging term weights derived from training data.

Limitations:

- Term Independence:
  - Assumes terms like "climate" and "change" are independent, ignoring their strong semantic connection.
- Equal Contributions:
  - Assigns equal importance to all query terms unless corrected by probabilities.

Modern models (e.g., BM25) address these limitations by incorporating term frequency, document length normalization, and more nuanced weighting schemes.

BM25

Assignments

1. Implement BM25 model with example.

Language model for IR

To coming up with good queries is to think of words that would likely appear in a relevant document, and to use those words as the query.

The language modeling approach to IR directly models that idea: a document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words often.

This approach thus provides a different realization of some of the basic ideas for document ranking.

- Instead of overtly modeling the probability $P(R=1|q,d)$ of relevance of a document d to a query q, as in the traditional probabilistic approach to IR ,
- the basic language modeling approach instead builds a probabilistic language model M_d from each document d, and ranks documents based on the probability of the model generating the query: $P(q|m_d)$.

Language model

- A model for how humans generate language
- Used in many language oriented-tasks (MT, word prediction, IR)
- Usually probabilistic in nature (e.g. multinomial, neural)

I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish I wish I wish
…

CANNOT GENERATE: wish I wish

► **Figure 12.1** A simple finite automaton and some of the strings in the language it generates. → shows the start state of the automaton and a double circle indicates a (possible) finishing state.

What do we mean by a document model generating a query?

A traditional generative model of a language, of the kind familiar from formal language theory, can be used either to recognize or to generate strings.

For example, the figure above can generate strings that include the examples shown.

- The full set of strings that can be generated is called the ***language*** of the automaton

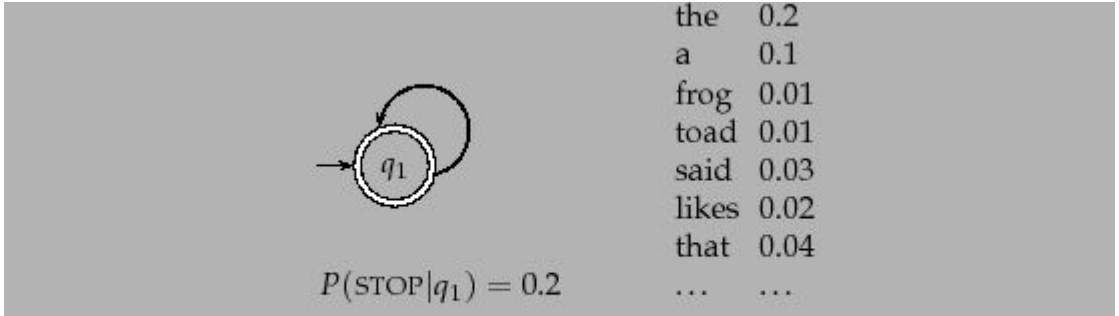
If instead each node has a probability distribution over generating different terms, we have a language model.

The notion of a language model is inherently probabilistic.

After generating each word, we decide whether to stop or to loop around and then produce another word, and so the model also requires a probability of stopping in the finishing state.

Such a model places a probability distribution over any sequence of words.

By construction, it also provides a model for generating text according to its distribution.



► **Figure 12.2** A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

Steps for Text Generation

1. The model starts in the initial state.
2. It generates a word according to the assigned probabilities (e.g., P(frog)=0.01).
3. After generating a word, the model decides whether to:
    - **Stop** (end the sequence) or,
    - **Continue** (generate another word).

This requires an additional **stop probability** to decide when to terminate text generation.

Calculating the Probability of a Word Sequence

To compute the probability of a sequence s="frog said that toad likes frog"

1. **Multiply the term probabilities** (emission probabilities) for each word in the sequence:
   P("frog said that toad likes frog")=P(frog)×P(said)×P(that)×P(toad)×P(likes)×P(frog)
2. Example with M1:
   $P(s|M_1)=0.01\times0.03\times0.04\times0.01\times0.02\times0.01$
3. **Multiply by stop probabilities**: After generating each word, you consider the probability of **not stopping** (continuing) or stopping:
   P(continuing after "frog")=0.8(assuming a 0.8 probability to continue)
   P(stopping after "frog")=0.2(probability to stop after the last word)
   For this example:
   $P(s|M_1)\times P(\text{continuing})\times P(\text{stopping})=0.00000000001573$

Likelihood Ratio

To compare two language models M_1 and M_2 on a given sequence, compute their **likelihood ratio**:

$$\text{Likelihood Ratio} = \frac{P(s|M_1)}{P(s|M_2)}$$

Example:

For $s =$ "frog said that toad likes that dog":

- Calculate $P(s|M_1)$:

$$P(s|M_1) = 0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.005 = 0.000000000048$$

- Calculate $P(s|M_2)$:

$$P(s|M_2) = 0.0002 \times 0.03 \times 0.04 \times 0.0001 \times 0.04 \times 0.01 = 0.00000000000384$$

- Likelihood ratio:

$$\frac{P(s|M_1)}{P(s|M_2)} = \frac{0.000000000048}{0.00000000000384} = 12.5$$

Thus, M_1 is 12.5 times more likely to have generated this sequence than M_2.

What is a language model?

- Probability distribution over strings of text
  - How likely is the occurrence of a given string (observation) in a given language
    - P1 = P(“a quick brown dog”)
    - P2 = P(“dog quick a brown”)
    - P3 = P(“un chien brun rapide”)
    - P4 = P(“un brown dog rapide”)
    - For English Language: p1>p2>p3>p4
    - In most of IT,
      - Assume that p1==p2
- Language model assigns a probability to a sequence of words by means of a probability distribution.

Types of language models

How do we build probabilities over sequences of terms?

- We can always use the chain rule to decompose the probability of a sequence of events into the probability of each successive event conditioned on earlier events:

$P(t1t2t3t4) = P(t1)P(t2|\ t1)P(t3|t1t2)P(t4|\ t1t2t3)$

The simplest form of language model simply throws away all conditioning context, and estimates each term independently.

Such a model is called a unigram language model :

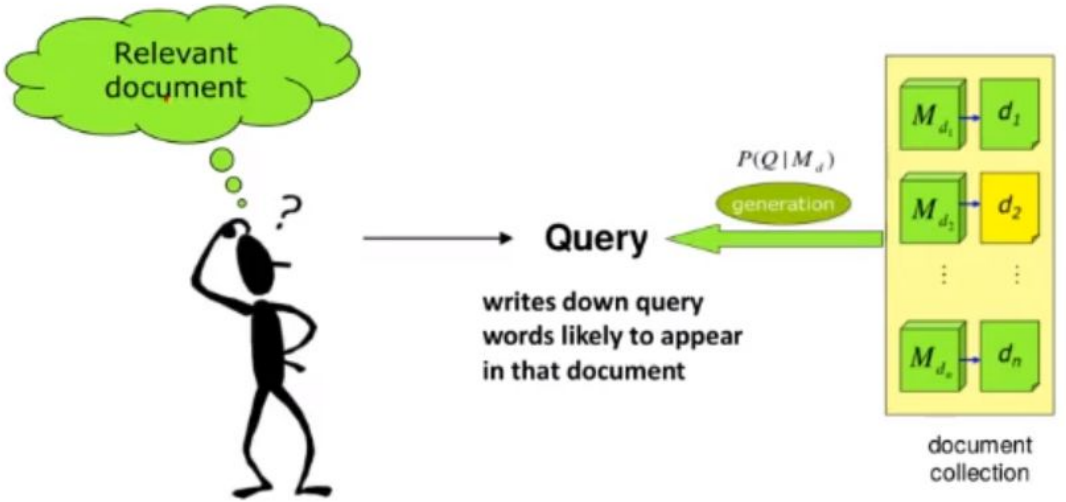
$P_{uni}(t1t2t3t4) = P(t1)P(t2)P(t3)P(t4)$

There are many more complex kinds of language models, such as bigram language models , which condition on the previous term,

$P_{bi}(t1t2t3t4) = P(t1)P(t2|\ t1)P(t3|\ t2)P(t4|\ t3)$

Retrieval based on language model

- A document is a good match if the document model is likely to generate the query.



...

- Coming with a good query
  - Use words that would likely to appear in a relevant documents as the query terms

- The language modelling approach to IR directly models this idea:
  - A document is a good match to a query if the document model is likely to generate the query
  - This happens if the document contains the query terms often.

- Build a probabilistic language model $\boldsymbol{M_d}$ from each document ***d.***
- Build a probabilistic language model $\boldsymbol{M_Q}$ form the query ***Q.***
- Rank documents based on the probability of document model generating the query. $\mathbf{P(M_q|M_d)}$
  - For all documents belonging to the Collection : score(d,Q) = $p(Q|M_d)$ for all documents containing Q terms.

Steps:

1. Construct language models associated with each of the documents of the collection.
2. Given query Q, compute the generation probability of Q form each of **documents**
   - **Then compute, $P(Q|m_d)$**
3. Rank the documents based on the probability in decreasing order.

Constructing the Language model of a document

In some modern IR systems, language models can be constructed during the **indexing phase**. This is particularly relevant in the context of the **language modeling approach to IR**.

- **Document Language Model**: During indexing, each document can be used to construct a language model that represents the distribution of words within the document.
    - This can be done using techniques like **Maximum Likelihood Estimation (MLE)**, where the probability of a word occurring in the document is estimated.

Constructing the Language model of a document

Unigram

In a unigram model, each word in a document is assumed to be generated independently of others.

- This means that word order is irrelevant, and only the frequency of each word matters.

This is often referred to as a "bag of words" model because the structure of the sentence is ignored.

- unigram language models are constructed based on the multinomial distribution of words in documents.
- Will be constructed during the indexing.

Maximum Likelihood Estimation (MLE) in IR:

MLE is a method used to estimate the parameters of a statistical model.

In the case of a language model, MLE is used to estimate the probabilities of words in a document.

When constructing a document's language model, we treat the occurrence of each word as a random variable, and we aim to find the probability distribution of these words.

The basic idea behind MLE is to maximize the likelihood of observing the actual data (i.e., the words in a document) given the model's parameters.

...

For example, if we have a document D with a collection of words $w_1, w_2, \ldots, w_n$, the **MLE** for the probability of word w_i in the document is:

$$P(w_i|D) = \frac{C(w_i, D)}{|D|}$$

Where:

- $C(w_i, D)$ is the count of occurrences of word w_i in document D
- $|D|$ is the total number of words in the document

This equation represents a **Multinomial Distribution** over the words in the document, where the parameters of the distribution are the word counts.

…

Build Document Language Models (LM)

For each document, you will construct a Language Model.

This is done by calculating the probability of each word in the document.

Let’s assume the documents only contain the words in their respective sets,

- and we use Maximum Likelihood Estimation (MLE) for word probabilities.

General estimation approach

- tokenize/split the text into terms.
- Count total number of occurrences of each term ***(|D|).***
- Count the number of occurrences of each term ***(tf(t,d))***.
- Assign term ***t*** a probability p_t using MLE.

$$p_t = \begin{cases} \frac{tf(t,D)}{|D|} & \text{when span of text is document} \\ \frac{cf(t)}{|V|} & \text{when span of text is collection / corpus} \\ \frac{tf(t,Q)}{|Q|} & \text{when span of text is query} \end{cases}$$

Document Language model

- The language model estimated from document D is sometimes denoted as:

$$\theta_D \quad \text{OR} \quad M_D$$

- The probability given to term t by the language model estimated from document D is sometimes denoted as:

$$p_t = P(t|D) = P(t|M_D) = \frac{tf(t, D)}{|D|}$$

We can use this model to determine the probability of a particular sequence of text.

Example

Let’s calculate the word probabilities for each document:

Document D1: "apple orange banana"

- Total words in D1 = 3
- Word counts:
  - P(apple | D1) = 1/3
  - P(orange | D1) = 1/3
  - P(banana | D1) = 1/3

Document D2: "apple orange"

- Total words in D2 = 2
- Word counts:
  - P(apple | D2) = 1/2
  - P(orange | D2) = 1/2

Document D3: "banana orange apple"

- Total words in D3 = 3
- Word counts:
  - P(apple | D3) = 1/3
  - P(orange | D3) = 1/3
  - P(banana | D3) = 1/3

Steps:

1. Construct language models associated with each of the documents of the collection.
2. Given query Q, compute the generation probability of Q form each of **documents**
   - **Then compute, $P(Q|m_d)$**
3. Rank the documents based on the probability in decreasing order.

Before this…

- Task: given a query, retrieve relevant documents
- Objective: rank documents based on the probability that they are on the same topic as the query
- Solution:
  - Every document in the collection is associated with a language model
  - Score each document (D) according to probability given by its language model (M_D) to the query (Q)
  - Rank documents based on $P(D|Q)$

$$P(D|Q) = \frac{P(Q|D)P(D)}{P(Q)} \quad .$$

- $P(Q)$: same for all document $\rightarrow$ ignore
- $P(D)$: prior probability of selection of D - often treated as the same for all D
  - But we can give a prior to *high-quality* documents, e.g., those with high *PageRank*.
- $P(Q|D)$: probability of Q given D.

Query Likelihood in IR

Query Likelihood is an approach in language modeling for IR, where we estimate the probability of a query Q being generated by a document d's language model M_d, denoted as $P(Q|M_d)$.

This is the **generation probability** of the query from the document model.

Treat the query $Q = \{q_1, q_2, \ldots, q_k\}$ as a sequence of words.

Compute the probability $P(Q|M_d)$ as the product of the probabilities of individual query terms under the document model:

$$P(Q|M_d) = \prod_{q \in Q} P(q|M_d)$$

Generation probability of Q

- This is directly related to the likelihood of generating the query Q from the document language model M_d.
- In practical terms, this step evaluates how well each document can "explain" or "generate" the query.

For example:

- If query Q={toad, likes}
- Document d: M1, $P(\text{toad} \mid M1)=0.01$, $P(\text{likes} \mid M1)=0.02$,
- Then: $P(Q \mid M1)=P(\text{toad} \mid M1) \cdot P(\text{likes} \mid M1)=0.01 \cdot 0.02=0.0002$

...

Ranking Documents

- Once P(Q|Md) is computed for all documents in the collection, the documents are ranked in **descending order** of P(Q|Md).
- This ranking prioritizes documents whose language models are more likely to generate the query.

Example:

Document Collection

Suppose we have two documents:

- **Doc1:** "information retrieval is interesting"
- **Doc2:** "retrieval models use statistics"

Step 1: Tokenization

Break down the documents into unigrams:

- **Doc1 Tokens:** ["information", "retrieval", "is", "interesting"]
- **Doc2 Tokens:** ["retrieval", "models", "use", "statistics"]

Step 2: Vocabulary Creation

Combine unique words across all documents:

```
Vocabulary = {"information", "retrieval", "is", "interesting", "models", "use",
"statistics"}
```

Step 3: Compute Word Probabilities Using MLE

For each document, calculate the **MLE probability** of each word in the vocabulary.

The MLE for a word w in a document D is:

$$P(w \mid D) = \frac{\text{count}(w, D)}{\text{total words in } D}$$

For Doc1:

- Total words = 4
- $P(\text{information} \mid \text{Doc1}) = \frac{1}{4} = 0.25$
- $P(\text{retrieval} \mid \text{Doc1}) = \frac{1}{4} = 0.25$
- $P(\text{is} \mid \text{Doc1}) = \frac{1}{4} = 0.25$
- $P(\text{interesting} \mid \text{Doc1}) = \frac{1}{4} = 0.25$
- $P(\text{models} \mid \text{Doc1}) = 0, P(\text{use} \mid \text{Doc1}) = 0, P(\text{statistics} \mid \text{Doc1}) = 0$

For Doc2:

- Total words = 4
- $P(\text{retrieval} \mid \text{Doc2}) = \frac{1}{4} = 0.25$
- $P(\text{models} \mid \text{Doc2}) = \frac{1}{4} = 0.25$
- $P(\text{use} \mid \text{Doc2}) = \frac{1}{4} = 0.25$
- $P(\text{statistics} \mid \text{Doc2}) = \frac{1}{4} = 0.25$
- $P(\text{information} \mid \text{Doc2}) = 0, P(\text{is} \mid \text{Doc2}) = 0, P(\text{interesting} \mid \text{Doc2}) = 0$

Step 4: Query Likelihood Scoring

Suppose we have a query: **"information retrieval"**.

Tokenize the query into unigrams: ["information", "retrieval"].

The query likelihood for each document is computed as:

$$P(Q \mid D) = \prod_{w \in Q} P(w \mid D)$$

For Doc1:

$$P(Q \mid \text{Doc1}) = P(\text{information} \mid \text{Doc1}) \cdot P(\text{retrieval} \mid \text{Doc1})$$

$$P(Q \mid \text{Doc1}) = 0.25 \cdot 0.25 = 0.0625$$

For Doc2:

$$P(Q \mid \text{Doc2}) = P(\text{information} \mid \text{Doc2}) \cdot P(\text{retrieval} \mid \text{Doc2})$$

$$P(Q \mid \text{Doc2}) = 0 \cdot 0.25 = 0$$

...

Step 5: Ranking

- **Doc1:** $P(Q \mid \text{Doc1}) = 0.0625$
- **Doc2:** $P(Q \mid \text{Doc2}) = 0$

Conclusion: Doc1 is ranked higher for the query "information retrieval" because it has a higher likelihood score.

Smoothing

Smoothing: In practice, a smoothing technique like **Laplace smoothing** or **Dirichlet smoothing** is used to avoid zero probabilities for words not present in a document.

Strengths: Simple, interpretable, and effective for short queries.

Weaknesses: Assumes word independence, which may not capture the context effectively.

Assignment:

1. Explain BM25 algorithm.
2. Explain unigram language model with MLE and laplace correction.

Thanks for your time and attention.