

# INFORMATION RETRIEVAL

## ASSIGNMENT – 8

---

### ASSIGNMENT – 8.1

Describe and compare the Boolean model, the Vector Space Model and the Binary independence model. Especially focus on the general idea, ranking and the type of term weights?

#### **ANS]- Boolean Model:**

##### **General Idea:**

The **Boolean model** uses Boolean logic (AND, OR, NOT) to match documents to a query. It treats each document as a set of terms and evaluates whether a document satisfies the conditions of the query exactly.

##### **Ranking:**

There is no ranking of results in the Boolean model. A document is either **relevant** (matches the query) or **not relevant** (does not match). This binary decision-making leads to a lack of granularity in determining the degree of relevance.

##### **Term Weights:**

In the Boolean model, term weights are binary: a term is either **present** (1) or **absent** (0) in a document. No additional information, like frequency or importance, is considered.

---

#### **Vector Space Model (VSM):**

##### **General Idea:**

The **Vector Space Model** represents documents and queries as vectors in a multi-dimensional space, where each dimension corresponds to a unique term. Similarity between a document and a

query is computed using measures like **cosine similarity**, which reflects how close their vectors are.

### 3. Cosine Similarity

$$\text{Cosine Similarity} = \frac{\text{Dot Product}}{\|\text{Doc}_1\| \cdot \|\text{Doc}_2\|}$$

#### **Ranking:**

The model ranks documents based on their similarity to the query. Higher similarity scores indicate higher relevance, allowing documents to be ranked in a continuum of relevance.

#### **Term Weights:**

Term weights in VSM are typically based on **TF-IDF (Term Frequency-Inverse Document Frequency)**:

- **Term Frequency (TF)**: Reflects how often a term appears in a document.
- **Inverse Document Frequency (IDF)**: Discounts terms that are frequent across many documents, emphasizing rare and more informative terms.

The product of TF and IDF provides the term weight.

---

### **Binary Independence Model (BIM):**

#### **General Idea:**

The **Binary Independence Model** assumes that terms are independent of one another and that relevance of a document depends only on the presence or absence of terms (binary representation). It is commonly used in probabilistic information retrieval systems.

#### **Ranking:**

The model computes the probability of a document being relevant given a query, using **Bayes' theorem**. Documents are ranked based on their **probability of relevance**.

## Term Weights:

Term weights in BIM reflect the contribution of each term to the relevance probability. These weights are typically learned from training data and can be influenced by the frequency of terms in relevant and non-relevant documents. The binary representation simplifies computation.

### Comparison

Feature	Boolean Model	Vector Space Model (VSM)	Binary Independence Model (BIM)
General Idea	Exact matches using logic	Geometric similarity in vector space	Probabilistic relevance based on binary term presence
Ranking	None (binary output)	Ranked by similarity scores	Ranked by probability of relevance
Term Weights	Binary (0 or 1)	TF-IDF or similar	Probabilistic weights (learned)
Strengths	Simple and interpretable	Supports ranking and partial matches	Effective for binary features and learning relevance
Weaknesses	No ranking, rigid queries	Assumes independence of terms, can be computationally intensive	Assumes term independence, limited to binary representation

Each model has its advantages and is suited for different contexts. The Boolean model is useful for precise queries, the Vector Space Model is effective for ranked retrieval with rich term weight considerations, and the Binary Independence Model excels in probabilistic and binary environments.

-----

-----

## ASSIGNMENT – 8.2

Learn about Language models! Explain the idea! Consider the following document collection:

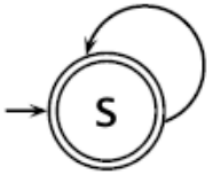
D1: Today is sunny. Sunny Berlin! To be or not to be.

D2: She is in Berlin today. She is a sunny girl. Berlin is always exciting!

Calculate the corresponding Unigram Language Model for each document! Omit the stop (and continue) probability in your calculations! Use these models to rank the documents given the query “sunny Berlin”!

ANS]-

## A Unigram Language Model



$w$	$P(w s)$	$w$	$P(w s)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

- A one-state probabilistic finite-state automaton
- Unigram language model = independence assumption
  - $P_{\text{uni}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$
- STOP is not a word, but a special symbol indicating that the automaton stops
- E.g.: string = frog said that toad likes frog STOP
  - $P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$   
 $= 0.00000000000048$

## Different Language Models



- Different models for each document (same type but different probability estimates)

Language model of $d_1$		Language model of $d_2$	
$w$	$P(w .)$	$w$	$P(w .)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
...	...	...	...

- E.g.: query = frog said that toad likes frog STOP
  - $P(\text{query}|M_{d1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$   
 $= 0.0000000000048 = 4.8 \cdot 10^{-12}$
  - $P(\text{query}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2$   
 $= 0.0000000000120 = 12 \cdot 10^{-12}$
  - $P(\text{query}|M_{d1}) < P(\text{query}|M_{d2}) \rightarrow d_2$  is “more relevant” concerning the query than  $d_1$

## Language Models in NLP:

Language models assign probabilities to sequences of words, estimating the likelihood of text. A Unigram Language Model assumes that each word in a document is independent of others, which simplifies the computation of probabilities.

For a given document  $D$ , the probability of generating a query  $Q$  under a unigram model is:

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

where  $P(w|D)$  is the probability of word  $w$  in  $D$ , calculated as:

$$P(w|D) = \frac{\text{Frequency of } w \text{ in } D}{\text{Total words in } D}$$

## Step-by-Step Calculation

### Documents:

- D1: Today is sunny. Sunny Berlin! To be or not to be.
- D2: She is in Berlin today. She is a sunny girl. Berlin is always

exciting!

- **Stop Words:**

We ignore stop words like "is", "to", "be", "or", "not", etc.

**Pre-processed Documents:**

- D1: sunny sunny Berlin today
- D2: she sunny Berlin today sunny girl Berlin always exciting

**Word Frequencies and Total Words:**

# Information Retrieval

## → Assignment 8.2

1. D1:

Words: Sunny (2), Berlin (1), today (1)  
Total Words: 4

2. D2:

Words: Sunny (2), Berlin (2), today (1), she (1),  
girl (1), d always (1), exciting (1)  
Total Words: 9

## → UNIGRAM LANGUAGE MODEL -

1. Doc D1:

$$P(\text{Sunny}, D1) = \frac{2}{4} = 0.5,$$

$$P(\text{Berlin}, D1) = \frac{1}{4} = 0.25,$$

$$P(\text{today}, D1) = \frac{1}{4} = 0.25.$$

2. Doc D2:

$$P(\text{Sunny}, D2) = \frac{2}{9}$$

$$P(\text{Berlin}, D2) = \frac{2}{9}$$

$$P(\text{today}, D2) = \frac{1}{9}$$



$$P(\text{she}, D_2) = \frac{1}{9}$$

$$P(\text{girl}, D_2) = \frac{1}{9}$$

$$P(\text{always}, D_2) = \frac{1}{9}$$

$$P(\text{exciting}, D_2) = \frac{1}{9}$$

→ Ranking Documents based on Query -

⇒ Query = "Sunny Berlin".

Compute  $P(Q|D)$ .

a) Doc D1.

$$\begin{aligned} P(Q|D_1) &= P(\text{Sunny}|D_1) \cdot P(\text{Berlin}|D_1) \\ &= 0.5 \cdot 0.25 \\ &= \underline{\underline{0.125}} \end{aligned}$$

b) Doc D2.

$$\begin{aligned} P(Q|D_2) &= P(\text{Sunny}|D_2) \cdot P(\text{Berlin}|D_2) \\ &= \frac{2}{9} \cdot \frac{2}{9} \\ &= \frac{4}{81} \approx \underline{\underline{0.049}} \end{aligned}$$

→ Document Ranking

$D_1 > D_2$



## ASSIGNMENT – 8.3

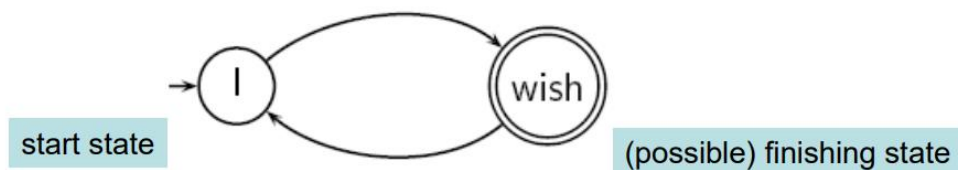
How can a language model be used in a spelling correction system? In particular, consider the case of context-sensitive spelling correction, and correcting incorrect usages of words such as “their” in “Are you their”?

ANS]-

### What are Language Models?



- Generative model of a language
  - Recognizing strings
  - Generating strings
- E.g. the following finite automaton



- Language it generates: I wish I wish I wish ...
- Language it cannot generate: “I wish I”, “wish I wish”
- Idea: documents are generated by different automaton similar to this

[

### Forms of spelling correction:

We focus on two specific forms of spelling correction that we refer to as **isolated-term correction** and **context-sensitive correction**.

]

A **language model** can be an integral component of a spelling correction system, especially in handling **context-sensitive spelling errors** like homophones or common misuse of words (e.g., “their” vs. “there” vs. “they’re”). Here’s how a language model facilitates this:

## 1. General Framework for Spelling Correction

- **Candidate Generation:** Generate a set of possible corrections for the misspelled or incorrect word (e.g., {“there,” “their,” “they’re”} for “their”).
  - **Candidate Scoring:** Use the language model to evaluate each candidate based on the context of the sentence.
  - **Best Match Selection:** Select the candidate with the highest score as the correction.
- 

## 2. Role of a Language Model in Context-Sensitive Spelling Correction

A language model leverages its ability to understand **context** and predict the most likely sequence of words. Here’s how it works step-by-step:

### Step 1: Identify Incorrect Words

Detect potential spelling or usage errors using:

- A predefined list of commonly confused words (e.g., "their," "there," "they're").
- Grammar-checking heuristics or rules.
- A separate spell-checking component to flag unlikely words.

### Step 2: Generate Candidates

For flagged words, generate possible correct alternatives:

- Use predefined confusion sets (e.g., "their," "there," "they're").

- Employ edit distance metrics (e.g., Levenshtein distance) for other types of misspellings.

### Step 3: Evaluate Candidates with the Language Model

For each candidate, substitute it into the original sentence and compute its likelihood or score using the language model:

- **n-gram models:** Compute the probability of the sentence  $P(\text{sentence})$  as the product of probabilities of each word conditioned on its history. For example:  

$$P(\text{Are you their}) = P(\text{Are}) \cdot P(\text{you} \mid \text{Are}) \cdot P(\text{their} \mid \text{Are you})$$

$$P(\text{Are you their}) = P(\text{Are}) \cdot P(\text{you} \mid \text{Are}) \cdot P(\text{their} \mid \text{Are you})$$
- **Neural language models (e.g., GPT):** Score the entire sentence in context, utilizing deep contextual embeddings to predict the most plausible sequence.

### Step 4: Select the Best Candidate

Rank the candidates by their scores and choose the one with the highest likelihood. For example:

- “Are you there?” may have a higher probability than “Are you their?” in most contexts, leading to the correct correction.

---

## 3. Advantages of Using a Language Model

- **Context Awareness:** Unlike traditional spell-checkers, a language model considers the broader context, making it effective for homophones and grammar-related errors.
- **Robust to Noise:** Neural language models, in particular, handle complex or noisy inputs better than rule-based systems.

- **Dynamic Corrections:** Adapt to varied sentence structures and styles without requiring explicit rules for every possible confusion set.
- 

## 4. Example

### Input Sentence:

“Are you their?”

### Steps:

1. Detect possible misuse of "their."
  2. Generate candidates: {"their," "there," "they're"}.
  3. Score candidates using the language model:
    - “Are you their?": Moderate probability (suits possessive contexts).
    - “Are you there?": High probability (common phrasing for location queries).
    - “Are you they're?": Low probability (incorrect grammar).
  4. Output: “Are you there?” as the corrected sentence.
- 

## 5. Limitations

- **Ambiguity in Context:** If the context is vague, even a strong language model might choose an incorrect correction.
  - **Training Data Bias:** Errors in the training data or lack of coverage for specific contexts can lead to poor predictions.
  - **Computational Cost:** High-quality neural language models can be computationally expensive to deploy at scale.
- 

## Conclusion

Language models excel in **context-sensitive spelling correction** by leveraging their ability to understand sentence structure and predict word usage. They are particularly effective for correcting homophones and subtle grammatical errors, making them a cornerstone of modern, intelligent spelling correction systems.

---

---

## **ASSIGNMENT – 8.4**