LAB 1. Program to demonstrate the Boolean Retrieval Model and Vector Space Model.

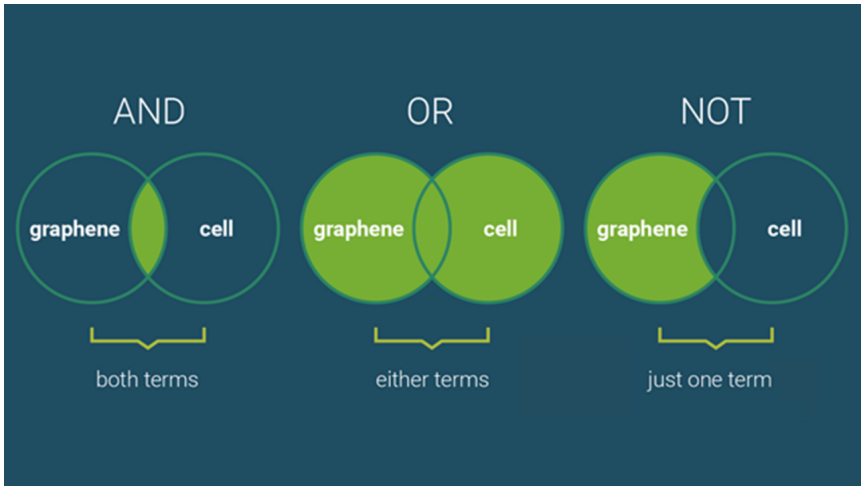
Boolean Retrieval Model and Vector Space Model

 # Introduction

Boolean Retrieval Model and Vector Space Model are fundamental concepts in Information Retrieval used to represent and retrieve text documents. This demonstration showcases their implementation using Python's scikit-learn library.

 # Definitions

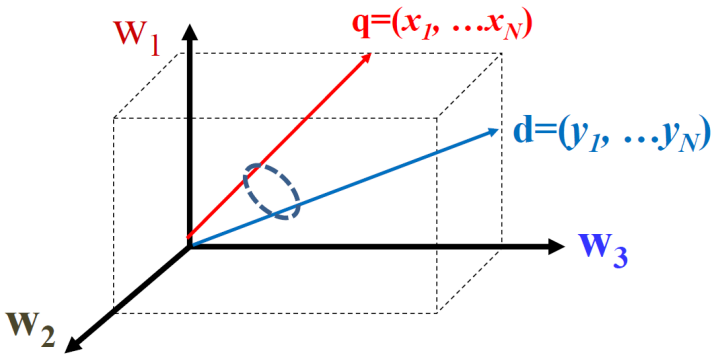
- **Boolean Retrieval Model**: A retrieval model where documents are represented as binary vectors indicating the presence or absence of terms. It operates based on boolean logic, returning documents that match the query exactly.



- **Vector Space Model (VSM)**: A retrieval model that represents documents and queries as vectors in a high-dimensional space, typically using TF-IDF weights. It measures similarity between documents and queries using cosine similarity.

Similarity Instantiation: Dot Product

$$Sim(q,d)=q.d= x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$



 # Algorithm

Boolean Retrieval Model

1. **Input**: Collection of documents.
2. **Preprocessing**: Tokenization, stop-word removal, stemming (optional).
3. **Vectorization**: Convert documents into binary vectors, where each element represents the presence or absence of a term.
    - Example: Document 1: "The cat sat on the mat"
        - Binary Vector: [1, 0, 1, 1, 0, ...] (1 represents the term is present, 0 represents absent)
4. **Query Processing**: Tokenize and vectorize the query.
    - Example: Query: "cat sat"
        - Binary Vector: [1, 1, 0, ...]
5. **Similarity Calculation**: Use boolean operators (AND, OR, NOT) to evaluate the query on document vectors.
    - Example: Query: "cat AND sat"
        - This would only return documents where both "cat" and "sat" are present (e.g., Document 1).
6. **Output**: Ranked list of documents based on whether they satisfy the query.

Vector Space Model

1. **Input**: Collection of documents.
2. **Preprocessing**: Tokenization, stop-word removal, stemming (optional).
3. **Vectorization**: Convert documents into TF-IDF weighted vectors.
    - TF-IDF (Term Frequency-Inverse Document Frequency):
        - TF (Term Frequency): Measures how often a term appears in a document.
        - IDF (Inverse Document Frequency): Measures how unique a term is across the document collection.
    - Example: Document 1: "The cat sat on the mat" (assuming "the" is a stop word)
        - TF-IDF Vector: [weight_1, weight_2, weight_3, ...] (weights are calculated based on TF-IDF)
4. **Query Processing**: Tokenize and vectorize the query.
5. **Similarity Calculation**: Use cosine similarity to calculate similarity between document vectors and query vector.
    - Cosine Similarity:
        - Formula: `cos(θ) = (A · B) / ||A|| ||B||`
        - A and B are the document and query vectors, respectively.
        - · represents the dot product.
        - ||A|| and ||B|| represent the magnitudes of vectors A and B.
6. **Output**: Ranked list of documents based on cosine similarity to the query.

 # Theory

- **Cosine Similarity**: A measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It's a common metric used in Information Retrieval to determine the similarity between documents and queries.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: A numerical statistic that reflects the importance of a word in a document relative to a collection of documents. It is often used as a weighting factor in information retrieval.

CODE

In [36]:

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Sample documents
documents = [
    "The sky is blue",
    "The sun is bright",
    "The sun in the sky is bright",
    "We can see the shining sun, the bright sun"
]

# Function to perform boolean retrieval based on the query and documents
def boolean_retrieval(query, documents, vocabulary):
    matching_documents = []
    for i, doc in enumerate(documents):
        if all(word in doc.lower() for word in query.split()):
            matching_documents.append(doc)
    return matching_documents

# Take input query from user
query = input("Enter your boolean query: ")

# Perform boolean retrieval
matching_documents = boolean_retrieval(query, documents, None)

# Print matching documents
print("-----------------------------------------------------------------")
print("\n\nBoolean Retrieval Model:")
print("-----------------------------------------------------------------")
boolean_similarity = [int(all(word in doc.lower() for word in query.split())
boolean_df = pd.DataFrame({"Document": documents, "Similarity": boolean_simi
boolean_df = boolean_df.sort_values(by='Similarity', ascending=False)
print(boolean_df)

# Vector Space Model
vectorizer = CountVectorizer()
doc_vectors = vectorizer.fit_transform(documents)

# Perform Vector Space retrieval
query_vector = vectorizer.transform([query])
similarity = cosine_similarity(doc_vectors, query_vector).flatten()

# Sort documents based on similarity
ranked_indices = np.argsort(similarity)[::-1]
ranked_documents = [documents[i] for i in ranked_indices]

# Print ranked documents
print("-----------------------------------------------------------------")
print("\n\nVector Space Model:")
print("-----------------------------------------------------------------")
vector_similarity = similarity
vector_df = pd.DataFrame({"Document": documents, "Similarity": vector_simila
vector_df = vector_df.sort_values(by='Similarity', ascending=False)
print(vector_df)
print("-----------------------------------------------------------------")
```

```
Enter your boolean query: sun bright
------------------------------------------------------------


Boolean Retrieval Model:
------------------------------------------------------------
                                     Document  Similarity
1                            The sun is bright           1
2                The sun in the sky is bright           1
3  We can see the shining sun, the bright sun           1
0                              The sky is blue           0
------------------------------------------------------------


Vector Space Model:
------------------------------------------------------------
                                     Document  Similarity
1                            The sun is bright    0.707107
3  We can see the shining sun, the bright sun    0.588348
2                The sun in the sky is bright    0.471405
0                              The sky is blue    0.000000
------------------------------------------------------------
```