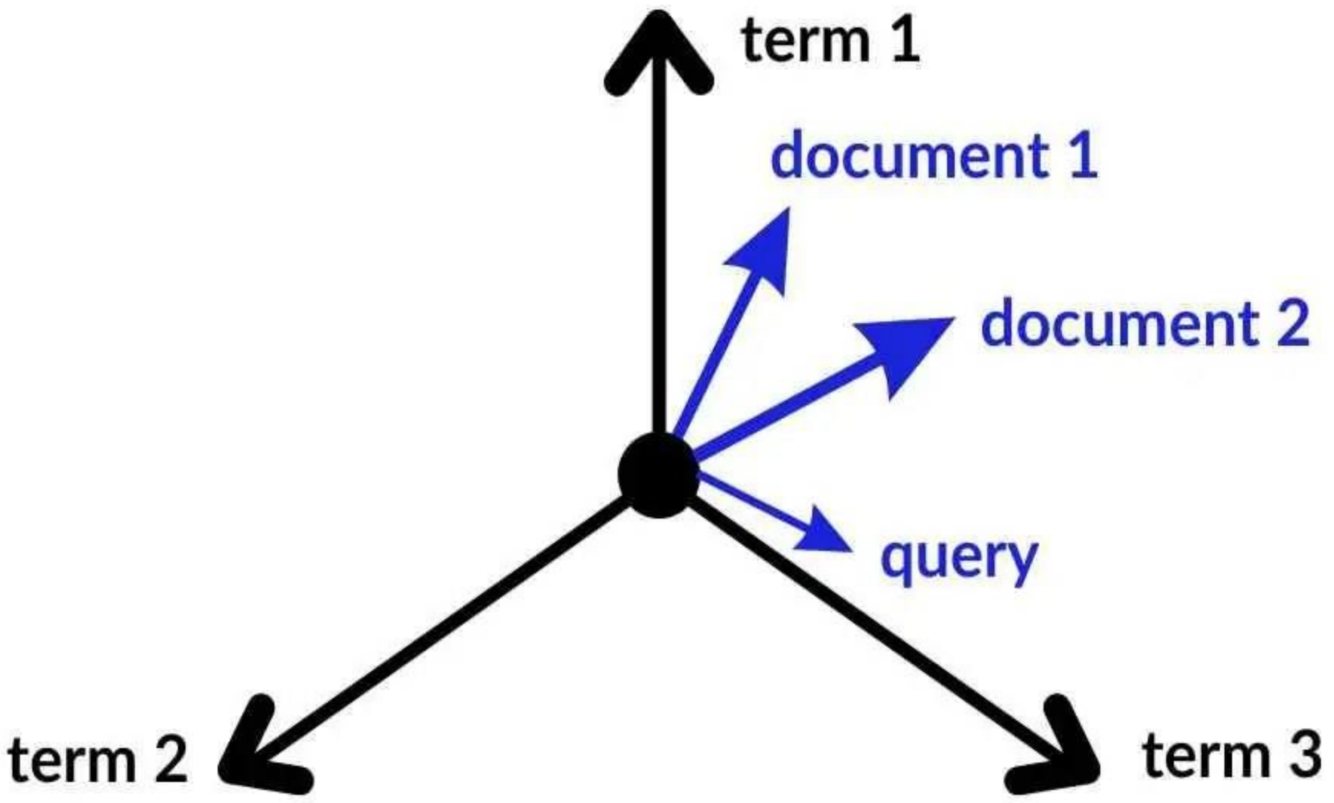
## Vector Space Model

The Vector Space Model represents documents and terms as vectors in a multi-dimensional space. Each dimension corresponds to a unique term in the entire corpus of documents.



Each dimension corresponds to a unique term, while the documents and queries can be represented as a vector within that space.

Here’s a basic overview of how the VSM works:

1. **Document-Term Matrix**: To create the vector representation of a collection of documents, you first construct a Document-Term Matrix (DTM) or Term-Document Matrix (TDM). Rows in this matrix represent documents, and columns represent terms (words or phrases). Each cell contains a numerical value representing a term's frequency or importance within a document.
2. **Term Frequency-Inverse Document Frequency (TF-IDF)**: Once you have the DTM, you often apply a TF-IDF transformation to the raw term frequencies. TF-IDF stands for Term Frequency-Inverse Document Frequency and is a measure that reflects the importance of a term within a document relative to its importance across all documents in the corpus. It helps in highlighting important terms while downplaying common terms.
3. **Vectorization**: After TF-IDF or a similar transformation, each document is represented as a vector in the high-dimensional space.
4. **Cosine Similarity**: To compare documents or perform text retrieval, you can use cosine similarity as a metric to measure the similarity between two document vectors. Cosine similarity calculates the cosine of the angle between two vectors and ranges from -1 (entirely dissimilar) to 1 (completely similar). A higher cosine similarity indicates a more significant similarity between documents.

## Cosine Similarity in a Vector Space Model

Now that we understand how the Vector Space Model (VSM) represents text as vectors, it's time to explore one of the key concepts that make VSM so powerful in Natural Language Processing: **cosine similarity**.

### What is Cosine Similarity?

Cosine similarity is a metric that measures the similarity between two vectors in a multi-dimensional space, such as the vectors representing documents in the VSM. In the context of VSM, it quantifies how alike two documents are based on their vector representations.

The key idea behind cosine similarity is to calculate the cosine of the angle between two vectors. If the vectors are very similar, their angle will be small, and the cosine value will be close to 1. Conversely, if the vectors are dissimilar, the angle will be large, and the cosine value will approach 0.

## How is Cosine Similarity Calculated?

The formula for calculating cosine similarity between two vectors A and B is as follows:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Where:

- $A \cdot B$ represents the dot product of vectors A and B.
- $\|A\|$ and $\|B\|$ represent the Euclidean norms (magnitudes) of vectors A and B, respectively.

The cosine similarity value ranges from -1 (completely dissimilar) to 1 (completely similar). A higher cosine similarity score indicates greater similarity between the two vectors.

## Cosine Similarity in a Vector Space Model

In a VSM, cosine similarity is crucial for information retrieval and document ranking. Here’s how it works in practice:

1. **Vector Representation**: We represent documents and queries as vectors using techniques like TF-IDF. Each document in the corpus and the query are converted into vectors in the same high-dimensional space.
2. **Cosine Similarity Calculation**: To determine the relevance of a document to a query, we calculate the cosine similarity between the query vector and the vectors representing each document in the corpus.
3. **Ranking**: Documents with higher cosine similarity scores to the query are considered more relevant and are ranked higher. Those with lower scores are ranked lower.

## Why Cosine Similarity?

Cosine similarity has several advantages when applied to text data:

1. **Scale Invariance**: Cosine similarity is scale-invariant, meaning it’s not affected by the magnitude of the vectors. This makes it suitable for documents of different lengths.
2. **Angle Measure**: It focuses on the direction of vectors rather than their absolute values, which is crucial for text similarity, where document length can vary.
3. **Efficiency**: Calculating cosine similarity is computationally efficient, making it suitable for large-scale text datasets.

Here are two very short texts to compare:

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

We want to know how similar these texts are, purely in terms of word counts (and ignoring word order). We begin by making a list of the words from both texts:

me Julie loves Linda than more likes Jane

Now we count the number of times each of these words appears in each text:

| | | |
|---|---|---|
| me | 2 | 2 |
| Jane | 0 | 1 |

| | | |
|---|---|---|
| Julie | 1 | 1 |
| Linda | 1 | 0 |
| likes | 0 | 1 |
| loves | 2 | 1 |
| more | 1 | 1 |
| than | 1 | 1 |

We are not interested in the words themselves though. We are interested only in those two vertical vectors of counts. For instance, there are two instances of 'me' in each text. We are going to decide how close these two texts are to each other by calculating one function of those two vectors, namely the cosine of the angle between them.

The two vectors are, again:

a: [2, 0, 1, 1, 0, 2, 1, 1]

b: [2, 1, 1, 0, 1, 1, 1, 1]

$$\text{cosine sim} = \frac{4+1+2+1+1}{\text{Sqrt}(12) \times \text{sqrt}(10)}$$

The cosine of the angle between them is about 0.822.

## Vector Model:

- document1 = ‘one two ‘
- document2 = ‘three two four ‘
- document3 =’one two three ‘

- document4 ='one two ‘

QUERY: Document containing ‘ one three three ‘

Representation in boolean model

| | one | two | three | four |
|---|---|---|---|---|
| **document1** | 1 | 1 | 0 | 0 |
| **document2** | 0 | 1 | 1 | 1 |
| **document3** | 1 | 1 | 1 | 0 |
| **document4** | 1 | 1 | 0 | 0 |

Calculation of term frequency

```
one --> 3/4 = 0.75

two --> 4/4 = 1

three --> 2/4 = 0.5

four --> 1/4 = 0.25
```

the IDF values of the terms are:

```
one --> log₂(4/3) = 0.4147
two --> log₂(4/4) = 0
three --> log₂(4/2) = 1
four -->log₂(4/1) = 2
```

Calculation of **weights ( tf * idf )**

```
weight(one) --> 0.75 * 0.4147 = 0.3110

weight(two) --> 1 * 0 = 0

weight(three) --> 0.5 * 1 = 0.5

weight(four) --> 0.25 * 2 = 0.5
```

Representation of vector model in terms of weights

| | **one** | **two** | **three** | **four** |
|---|---|---|---|---|
| **document1** | 0.3110 | 0 | 0 | 0 |
| **document2** | 0 | 0 | 0.5 | 0.5 |

| | | | | |
|---|---|---|---|---|
| **document3** | 0.3110 | 0 | 0.5 | 0 |
| **document4** | 0.3110 | 0 | 0 | 0 |

QUERY: Document containing ‘ one three three ‘

Calculation of weights for query terms(term frequency)

- weight(one) –> 1/3 = 0.333
- weight(three) –> 2/3 = 0.667

Vector representation

- Document
  $\vec{d_j} = \{0.3110, 0, 0.5, 0.5\}$

- Query
  $\vec{q} = \{0.333, 0, 0.667, 0\}$

Cosine Similarity calculation:

$$sim(d1,q) = \frac{0.3110*0.333+0*0+0*0.667+0*0}{\sqrt{(0.3110^2+0^2+0^2+0^2)}*\sqrt{(0.333^2+0^2+0.667^2+0^2)}} = 0.4466$$

$$sim(d2,q) = \frac{0*0.333+0*0+0.5*0.667+0.5*0}{\sqrt{(0^2+0^2+0.5^2+0.5^2)}*\sqrt{(0.333^2+0^2+0.667^2+0^2)}} = 0.4001$$

$$sim(d3,q) = \frac{0.3110*0.333+0*0+0.5*0.667+0*0}{\sqrt{(0.3110^2+0^2+0.5^2+0^2)}*\sqrt{(0.333^2+0^2+0.667^2+0^2)}} = 0.9086$$

$$sim(d4,q) = \frac{0.3110*0.333+0*0+0*0.667+0*0}{\sqrt{(0.3110^2+0^2+0^2+0^2)}*\sqrt{(0.333^2+0^2+0.667^2+0^2)}} = 0.4466$$

## Ranking

Document 3

Document 1 & Document 4

Document 2

# Vector Space Model example

Let’s walk through a simple example of the Vector Space Model (VSM) using a small corpus of documents and a query. In this example, we’ll represent documents and a query as vectors and calculate cosine similarity to retrieve relevant documents based on the query.

- **Step 1: Corpus and Query**

  Let’s start with a small corpus of three documents and a query:

  **Document 1**: “The quick brown fox jumps over the lazy dog.”
  **Document 2**: “A brown dog chased the fox.”
  **Document 3**: “The dog is lazy.”
  **Query**: “brown dog”

- **Step 2: Create the Document-Term Matrix (DTM)**

  We create a DTM where rows represent documents and columns represent terms. We’ll use TF-IDF values for each term in the matrix:

| | A | Brown | Chased | Dog | Fox | Is | Jumps | Lazy | Over | Quick | the |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DOC1 | | | | | | | | | | | |
| DOC2 | | | | | | | | | | | |
| DOC3 | | | | | | | | | | | |

Here, we’ve calculated TF-IDF values for each term in the documents and the query. You can use different formulas for TF-IDF, but this is common.

- **Step 3: Vectorize the Query**
-
-
- **Step 4: Calculate Cosine Similarity**

  Now, we calculate the cosine similarity between the query vector and each document vector. The formula for cosine similarity is:

$$\text{Cosine Similarity} = \frac{\text{Query Vector} \cdot \text{Document Vector}}{\|\text{Query Vector}\| \cdot \|\text{Document Vector}\|}$$

  Using this formula, we calculate the cosine similarity between the query and each document:

- Cosine Similarity(Query, Doc 1)
- Cosine Similarity(Query, Doc 2)
- Cosine Similarity(Query, Doc 3)
- **Step 5: Rank Documents by Similarity**