



Search Medium

Write



# Understanding Cosine Similarity: A key concept in data science



Arjun Prakash · Following

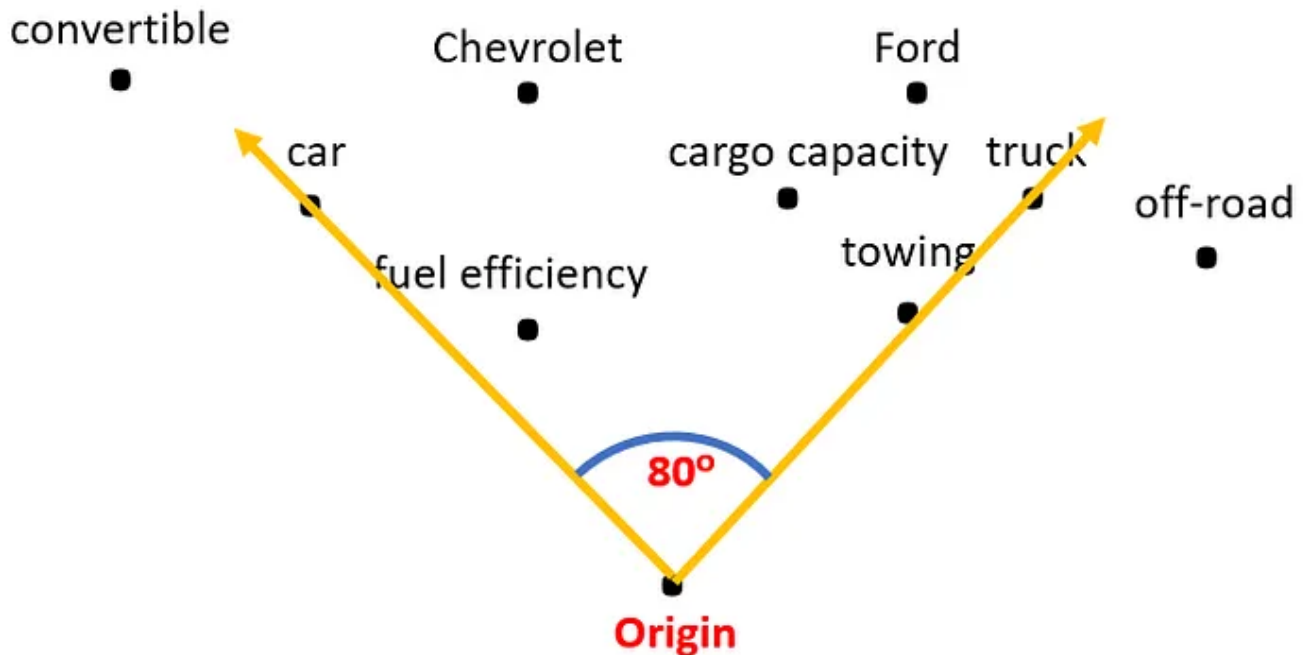
5 min read · Sep 21



6



1

photo from [wikipedia](#)

## Measuring similarity between 2 vectors

### Normal way

The traditional method for quantifying the distance between two points involves a direct measurement of the separation between them, a concept known as Euclidean distance measurement. This approach, named after the ancient Greek mathematician Euclid, who pioneered geometry, employs the formula  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$  to compute the distance between two points in a two-dimensional space.

Euclidean distance isn't limited to two dimensions; it extends naturally to three-dimensional and higher-dimensional spaces by adding more terms to the formula. In higher dimensions, it becomes  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + \dots + (z_1 - z_2)^2}$ .

### **The sexy and accurate way**

The contemporary and intriguing approach to measuring the proximity between two points involves the application of a concept known as cosine similarity. Cosine similarity operates by mapping vectors within a graphical context and then calculating the angle between these vectors, ultimately delivering the cosine of that angle as a measure of their similarity. When the angle between two vectors is precisely 45 degrees, their similarity is represented as  $\cos(45)$ .

Cosine similarity boasts an intuitive interpretation. When two vectors exhibit substantial similarity, their angle is close to 0 degrees, leading to a cosine similarity value of 1. Conversely, as the vectors become dissimilar or distant, the angle between them approaches 90 degrees, resulting in a cosine similarity value of 0. This intuitive interpretation simplifies the understanding of similarity metrics.

Furthermore, cosine similarity is remarkably versatile. It can be applied not only to pairs of points but also to a diverse array of data types, including text

documents, images, and numerical data. This versatility renders cosine similarity a powerful tool in various domains, such as natural language processing, recommendation systems, and data clustering.

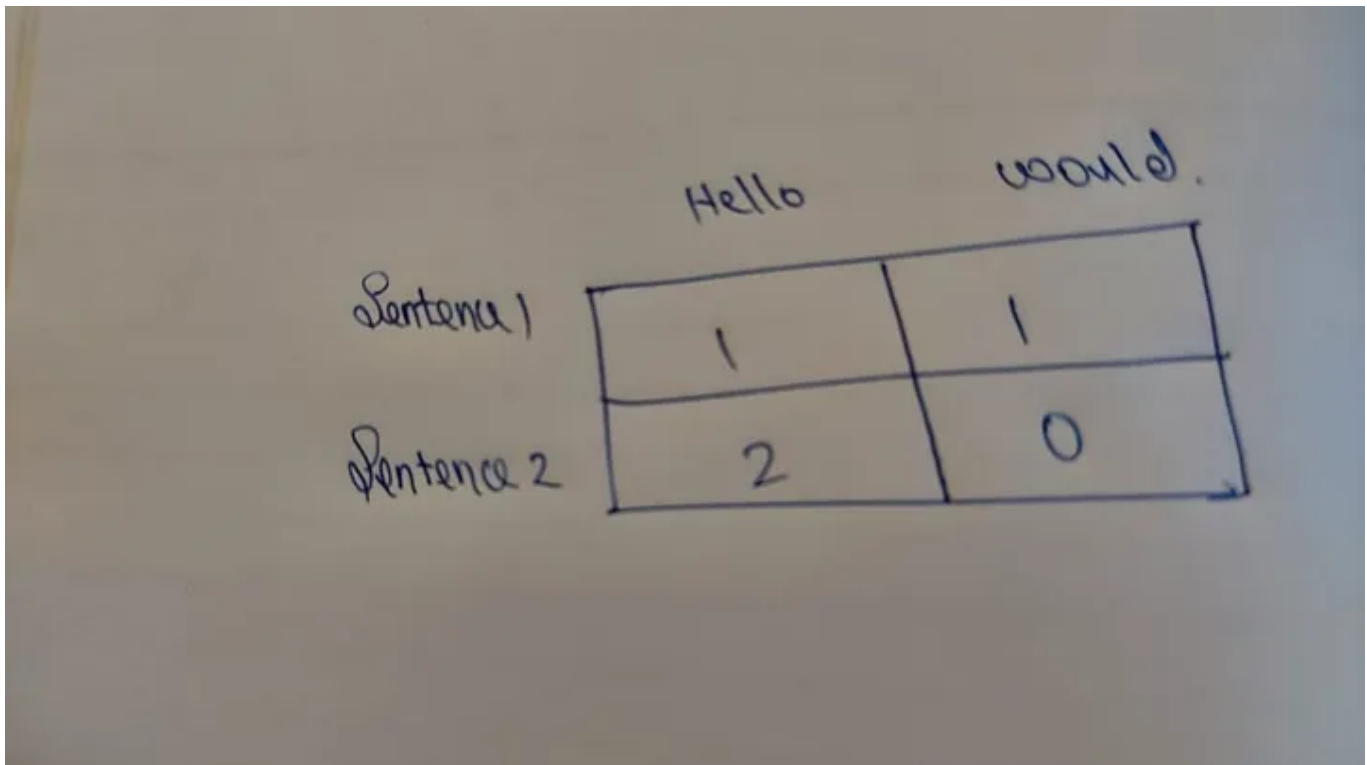
## Example Demonstration

Consider there are 2 sentences.

sentence 1: *hello world*

sentence 2: *hello hello*

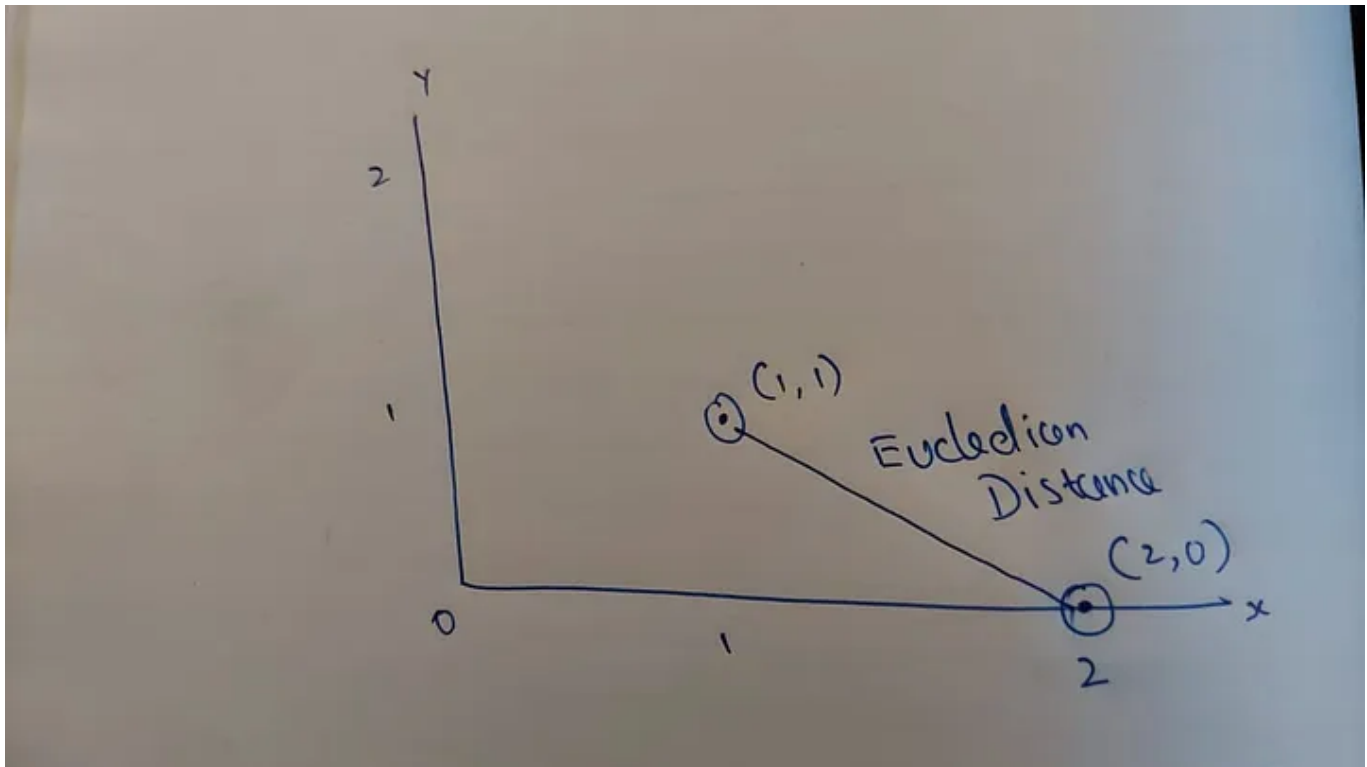
mapping them to vector, it will be something like this



A hand-drawn table illustrating the mapping of two sentences to a vector space defined by the words 'Hello' and 'world'. The table has two columns labeled 'Hello' and 'world' at the top. The rows are labeled 'Sentence 1' and 'Sentence 2' on the left. The values in the cells represent the count of each word in the sentence.

	Hello	world
Sentence 1	1	1
Sentence 2	2	0

this would look like this on graph

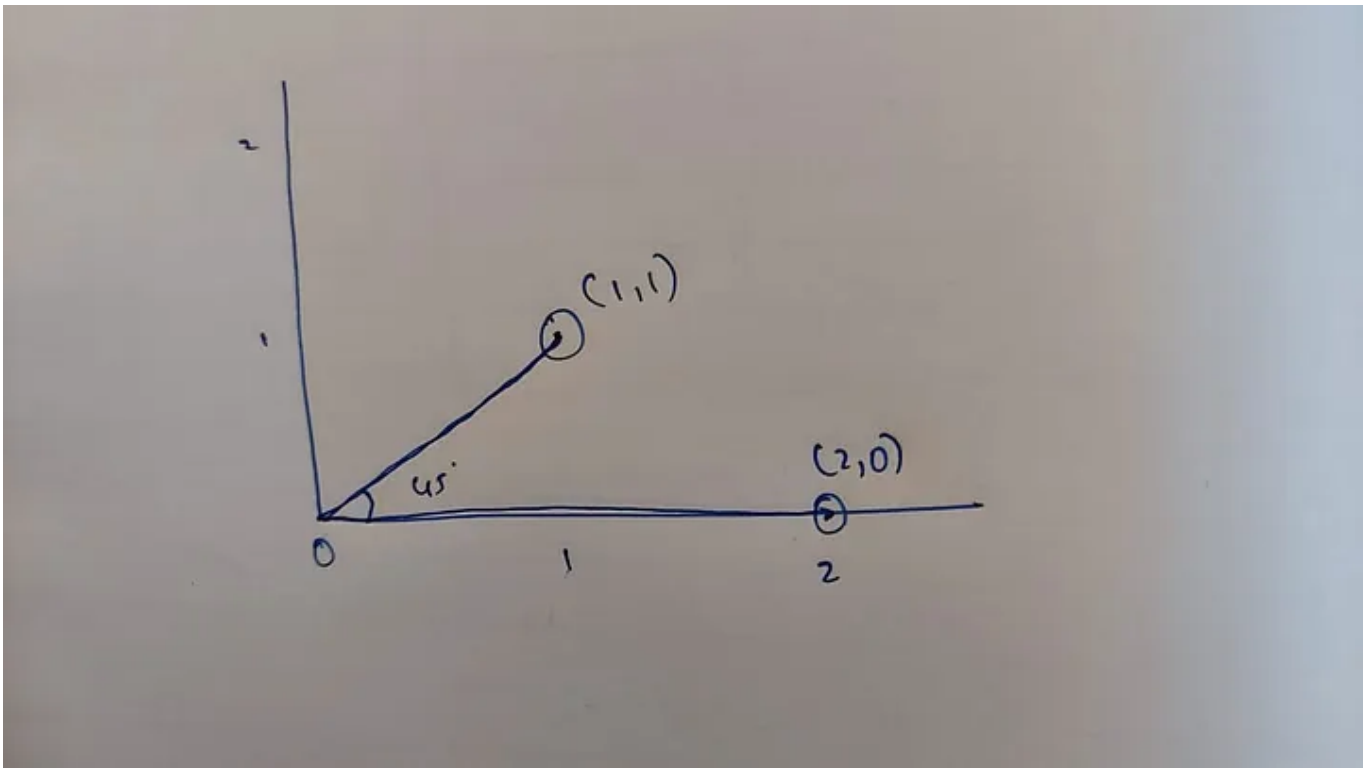


Now applying Euclidian distance formula to this,

$$\begin{aligned} ED &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ &\Rightarrow \sqrt{(2 - 1)^2 + (1 - 0)^2} \\ &\Rightarrow \sqrt{1 + 1} \\ &\Rightarrow \sqrt{2} \end{aligned}$$

root of 2 is around 1.4142

## Now applying cosine similarity to this

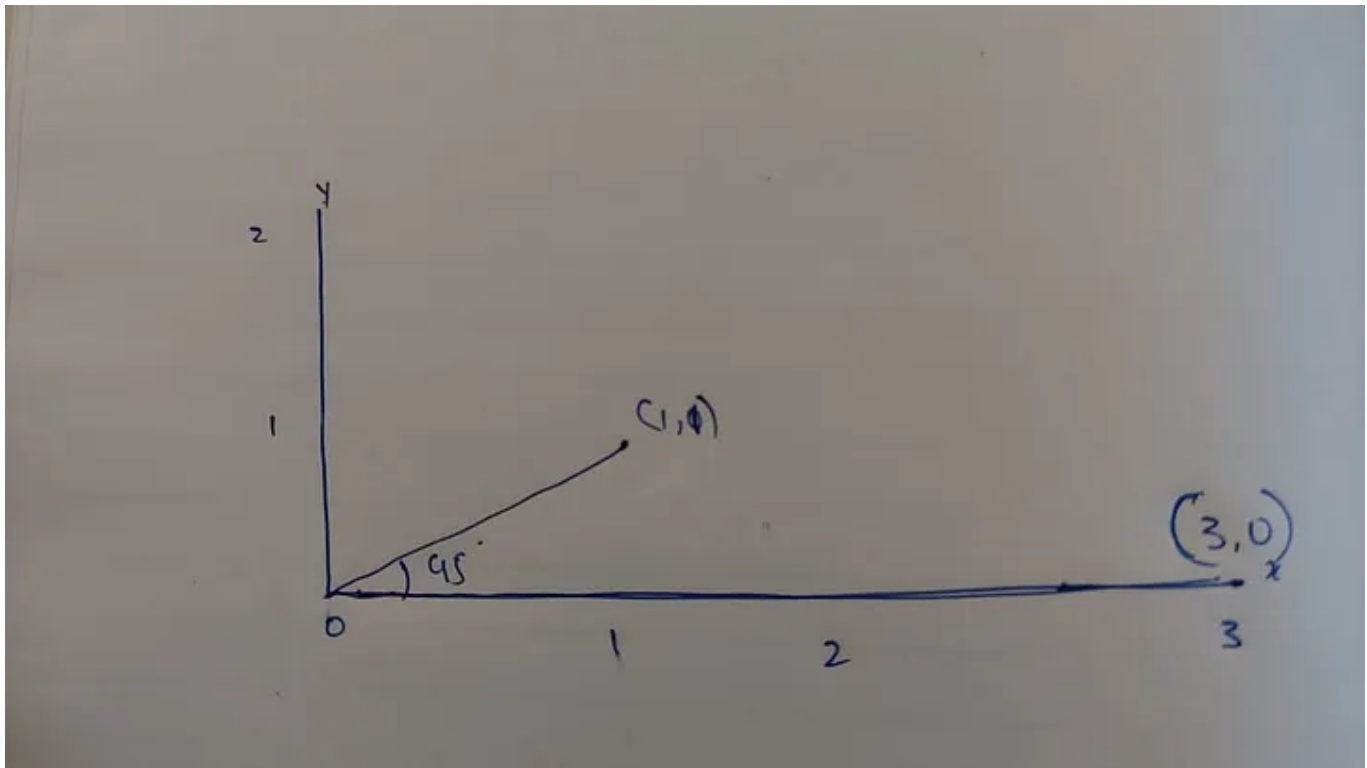


applying formula

Cosine Similarity

$$\cos(45^\circ) = 0.7071$$

this is the same even if sentence 2 is hello hello hello



while Euclidean distance will give you the value 2.2361 but cosine similarity will give you the same 0.7071

## Formula to calculate similarity for multi-dimensional arrays

The above way is an basic demonstration of cosine similarity, but in real world with text and images and other data there are no 2d vectors, they will be multi-dimensional vectors with 5 or more dimensions.

There also a formula to calculate this using cosine similarity.

$$\text{Cosine Similarity } (\cos \theta) = (A \cdot B) / (||A|| * ||B||)$$

*A · B: The dot product of vectors A and B.*

$\|A\|$  and  $\|B\|$ : The magnitude (Euclidean norm) of vectors  $A$  and  $B$ .

Consider these sentence.

sentence 1: I am Iron Man

sentence 2: Iron can rust.

we will get vector matrix like this,

A hand-drawn vector matrix on a piece of paper. The words 'I', 'am', 'Iron', 'man', 'can', and 'Rust' are written at the top, serving as column headers. The rows are labeled 'Sentence 1' and 'Sentence 2' on the left. The matrix contains binary values (0 or 1) indicating the presence of each word in each sentence.

	I	am	Iron	man	can	Rust
Sentence 1	1	1	1	1	0	0
Sentence 2	0	0	1	0	1	1

Cosine similarity formula,

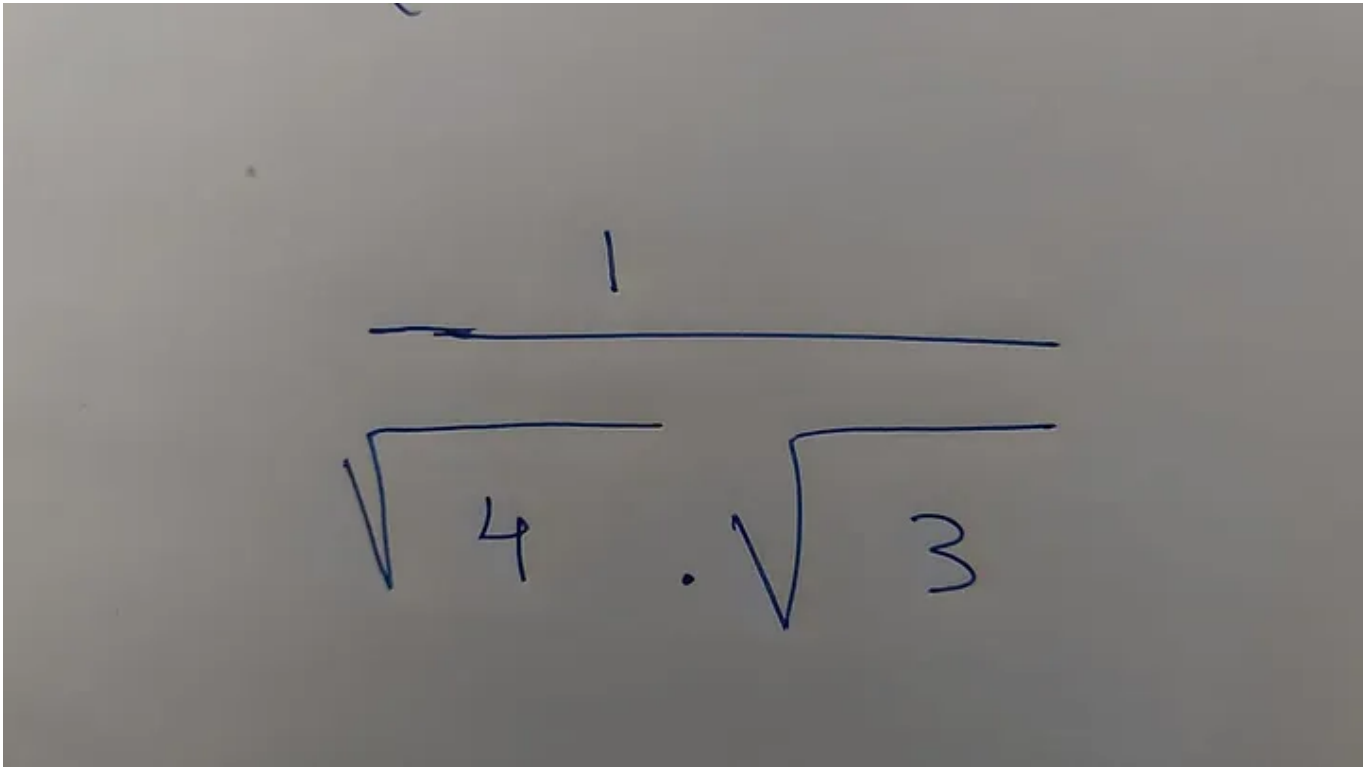
$$\text{Cos Similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2 \cdot \sum_{i=1}^n B_i^2}}$$

Applying this formula on top matrix,

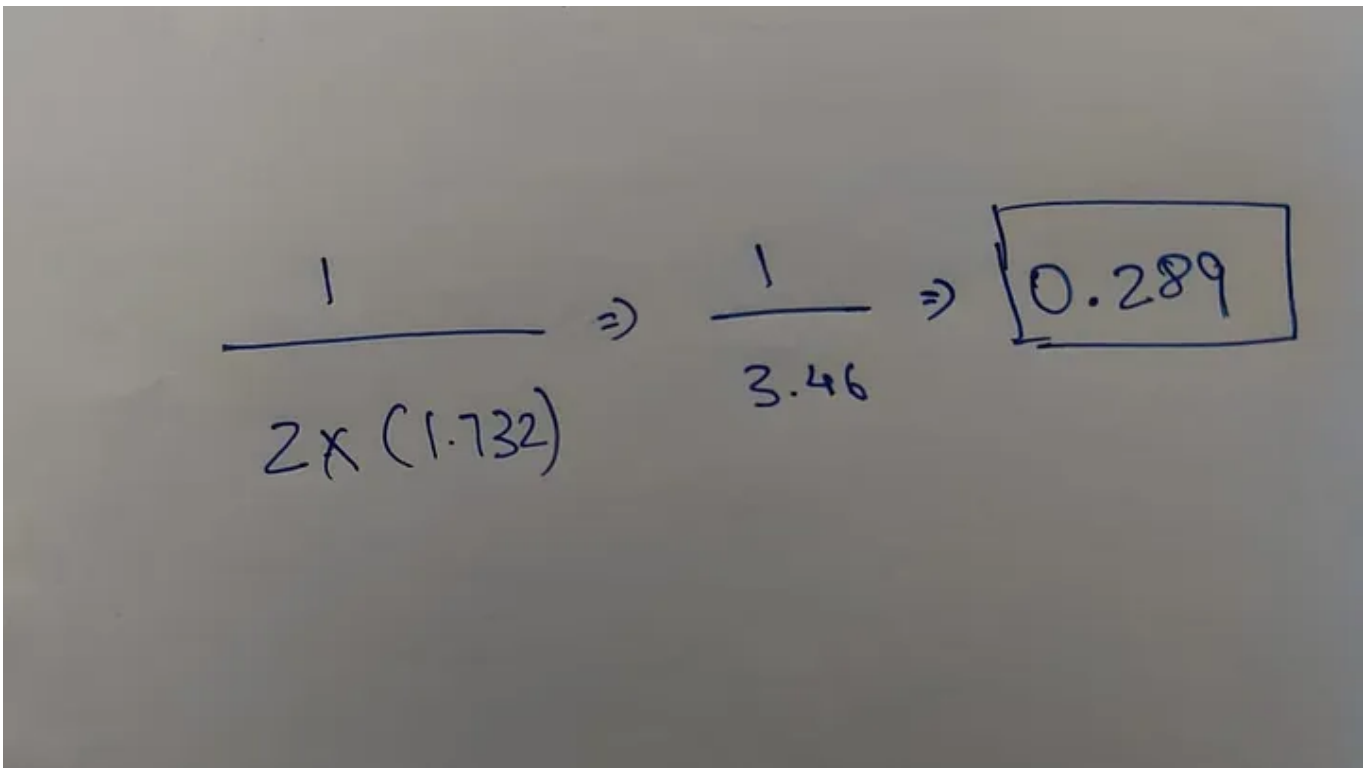
$$\frac{(1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 1)}{\sqrt{((1)^2 + (1)^2 + (1)^2 + (1)^2 + 0^2 + 0^2) \cdot (0^2 + 0^2 + 1^2 + 0^2 + 1^2 + 1^2)}}$$

Giving us this on further calculation,




$$\frac{1}{\sqrt{4} \cdot \sqrt{3}}$$

The final similarity is


$$\frac{1}{2 \times (1.732)} \Rightarrow \frac{1}{3.46} \Rightarrow \boxed{0.289}$$

The similarity is 0.289, which seems accurate given the sentences.

# Use Cases and disadvantages

## Use Cases:

1. **Document Similarity:** Cosine similarity is widely used in natural language processing to measure the similarity between documents. It's applied in plagiarism detection, document clustering, and content recommendation systems.
2. **Recommendation Systems:** In collaborative filtering-based recommendation systems, cosine similarity helps identify users or items with similar preferences. It's used in movie, music, and product recommendations.
3. **Text Classification:** In text classification tasks like spam detection, sentiment analysis, or topic modeling, cosine similarity can be used to compare the similarity between a document and predefined categories.
4. **Information Retrieval:** Search engines use cosine similarity to match user queries with relevant documents by comparing the query vector to document vectors.
5. **Clustering:** Cosine similarity aids in clustering similar data points together. For example, it's used in grouping similar news articles or social media posts.

## Disadvantages:

1. **Sensitivity to Document Length:** Cosine similarity doesn't consider the length of documents. Longer documents may have lower cosine similarity scores even if they share substantial content.
2. **Lack of Semantic Understanding:** Cosine similarity treats words or features as independent entities. It doesn't capture the semantic meaning of words, making it less effective in understanding context.

3. **Sparse Data Issues:** In high-dimensional spaces, where data is often sparse, cosine similarity can be less reliable. It might not accurately reflect the true similarity between data points.
4. **Normalization Dependency:** Cosine similarity depends on vector normalization. Different normalization methods can yield different results, making comparisons sensitive to preprocessing choices.
5. **No Negative Values:** Cosine similarity produces values between -1 and 1. It doesn't handle negative associations well, which might be relevant in certain applications.

Consider giving me a follow on medium and connect with me on [LinkedIn](#)

visit my portfolio [here](#).

[Data Science](#)[NLP](#)[Cosine Similarity](#)[Comparison](#)[Image Classification](#)

---

### More from the list: "NLP"

Curated by Himanshu Birla



Jon Gi... in Towards Data ...

## Characteristics of Word Embeddings

★ · 11 min read · Sep 4, 2021



Jon Gi... in Towards Data ...

## The Word2vec Hyperparameters

★ · 6 min read · Sep 3, 2021



Jon Gi... in

## The Word2vec

★ · 15 min rea

[View list](#)

## Written by Arjun Prakash

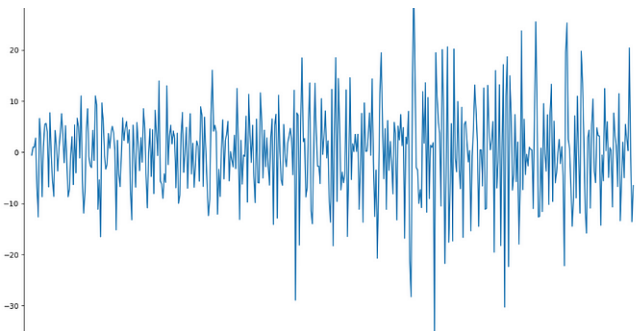
9 Followers

Following



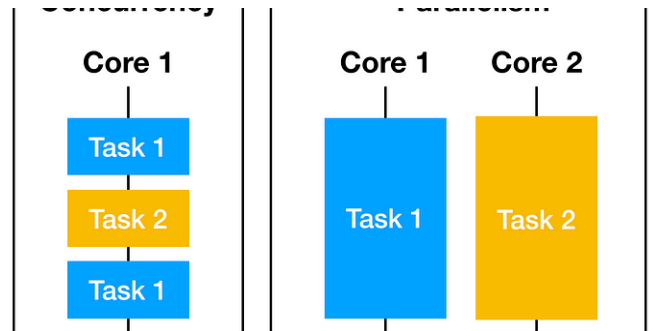
A tech enthusiast and self-learning software engineer passionate about web development, AI, ML, Data Science, and all things tech-related.

### More from Arjun Prakash



Arjun Prakash

## Decomposing a time-series, Why and How.




Arjun Prakash

## Threading vs Multiprocessing in Python: A Comprehensive Guide

Introduction

A small introduction and python code to perform time series decomposition

5 min read · 5 days ago


 12











 Arjun Prakash

Simplifying User Authentication with Django and React: A Guide to...


Discover how to integrate Django, React, and Google Sign-In for a secure and seamless...

19 min read · Jun 7

 2







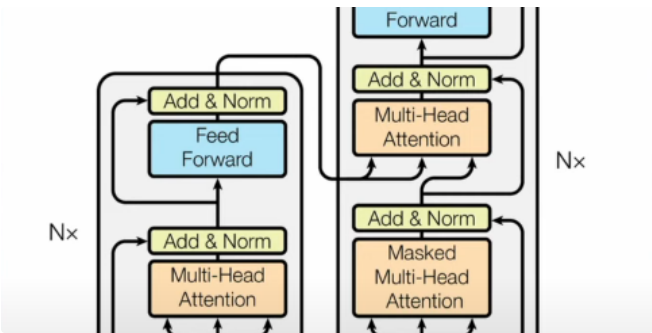
9 min read · Jun 29


 2











 Arjun Prakash

Transforming the Future: Unleashing the Power of Deep...

Introduction

7 min read · Jun 14



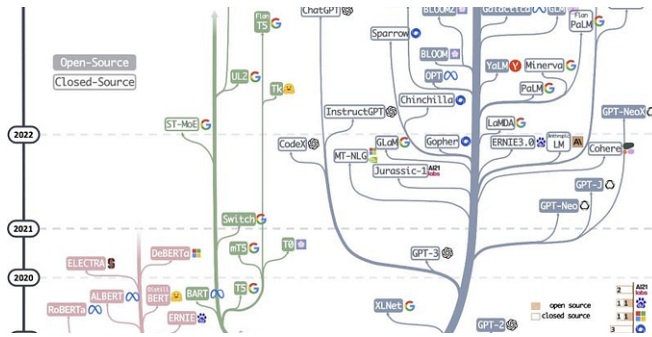






See all from Arjun Prakash

Recommended from Medium



Haifeng Li

## A Tutorial on LLM

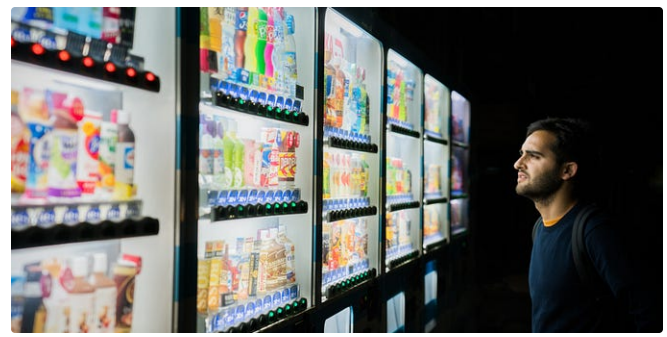
Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14

372



...



Ovbude Ehi

## Recommendation Engines

Practical Techniques: Content-Based Filtering, Collaborative Filtering and...

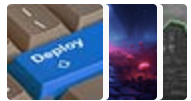
11 min read · May 8

5



...

## Lists



### Predictive Modeling w/ Python

20 stories · 452 saves



### Natural Language Processing

669 stories · 283 saves



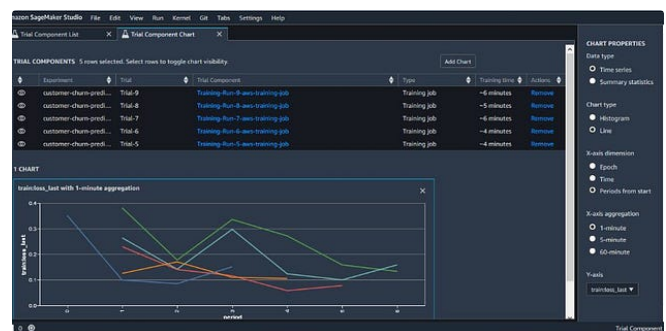
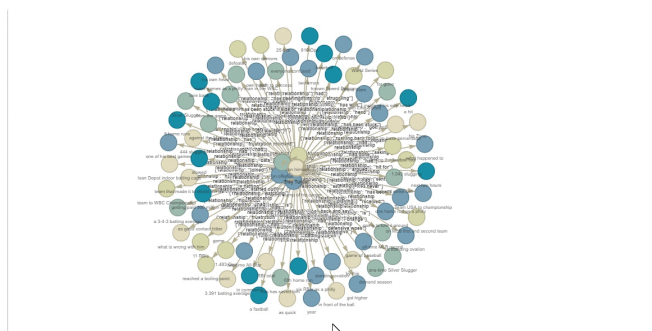
### New\_Reading\_List

174 stories · 133 saves



### Practical Guides to Machine Learning

10 stories · 519 saves



Wenqi Glantz in Better Programming

## 7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

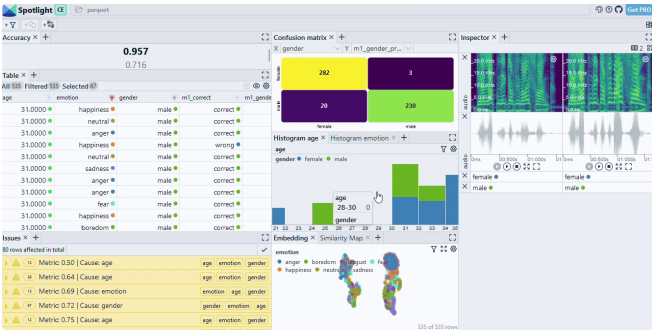
★ · 17 min read · 4 days ago


 501

 4







Stefan Suwelack

## Hands-On Voice Analytics with Transformers

Learn how to use open source models for gender detection and sentiment analysis.

7 min read · Sep 27

 51







Yugank Aman

## Top MLOps Tools to Manage Machine Learning Lifecycle


Businesses continue transforming their operations to increase productivity and...

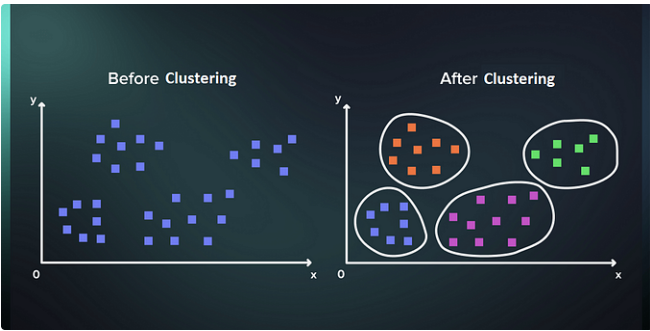
10 min read · May 30


 25










Deniz Gunay

## Clustering

Clustering

20 min read · Sep 18

 26







See more recommendations