

Distance and Similarity

Imagine three sentences (documents):

1. I love you.
 2. I love you, you.
 3. Stocks dropped.
- } intuitively, we can tell that sentence 1 and 2 are much more "similar" than 3.

Step 1: Vectorize the documents

So far, we've only learned about count vectorization (word count) but we'll discuss TF-IDF and embeddings soon.

	I	love	you	stocks	dropped
1. I love you.	1	1	1	0	0
2. I love you, you	1	1	2	0	0
3. Stocks dropped.	0	0	0	1	1

How do we quantify the degree of similarity between these documents?

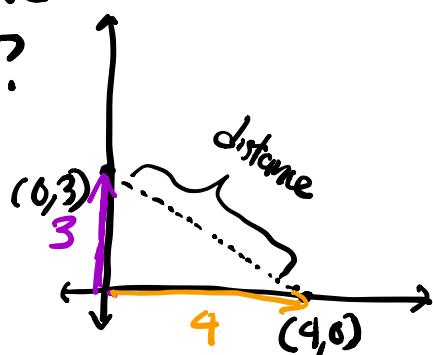
Option #1: Euclidean Distance

Remember Pythagorean Theorem?

$$a^2 + b^2 = c^2$$

$$3^2 + 4^2 = c^2$$

$$c = \sqrt{3^2 + 4^2} \Rightarrow c = \sqrt{25} = 5$$



Pythagorean theorem works in 2 dimensions (x, y) , but we can extend it to any # of dimensions:

$$\text{distance}(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

$$= \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

Going back to our three sentences:

- I love you stocks dropped
- A. I love you. $\rightarrow [1 \ 1 \ 1 \ 0 \ 0]$
 - B. I love you, you $\rightarrow [1 \ 1 \ 2 \ 0 \ 0]$
 - C. Stocks dropped. $\rightarrow [0 \ 0 \ 0 \ 1 \ 1]$
- d +s/ llo de s s l sty

$$\text{distance}(A, B) = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + (A_3 - B_3)^2 + (A_4 - B_4)^2 + (A_5 - B_5)^2}$$

$$= \sqrt{(1-1)^2 + (1-1)^2 + (1-2)^2 + (0-0)^2 + (0-0)^2}$$

$$= \sqrt{0+0+(-1)^2+0+0}$$

$$= \boxed{1}$$

$$\text{distance}(A, C) = \sqrt{(1-0)^2 + (1-0)^2 + (1-0)^2 + (0-1)^2 + (0-1)^2}$$

$$= \sqrt{1+1+1+(-1)^2+(-1)^2}$$

$$= \sqrt{5}$$

$$\approx \boxed{2.24} \quad \underline{\text{A is closer to B than C}}$$

But what is the issue with Euclidean distance?

1. It is sensitive to the norms (magnitudes) of the sentence vectors. For instance:

A. I love I love $\Rightarrow [2, 2]$ } These are basically the same! One is just a longer sentence
B. I love $\Rightarrow [1, 1]$

$$\text{distance}(A, B) = \sqrt{(2-1)^2 + (2-1)^2} = \sqrt{2} \leftarrow \begin{array}{l} \text{conceptually this should be} \\ \approx 0, \text{ since } A \text{ and } B \text{ are extremely similar.} \end{array}$$

Norm: the norm of a vector x is

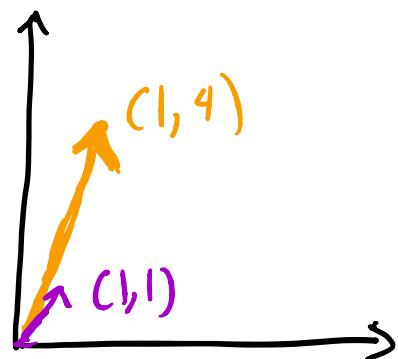
$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}$$

It represents the "strength" of a vector. In natural language processing, when you want to vectorize a document, the longer documents will have higher norms.

A. I love love love love $\Rightarrow [1, 4]$

B. I love $\Rightarrow [1, 1]$

A has a higher norm than B



2. Euclidean Distance performs poorly in high-dimensional feature spaces: in high dimensions, everything will seem far apart; the ratio between a point's farthest and closest neighbors approaches 1, which essentially means all points are "relatively" uniformly distant from one another.

An alternative? Cosine Similarity

$$\cos(a, b) = \frac{a \cdot b}{\|a\|_2 \|b\|_2}$$

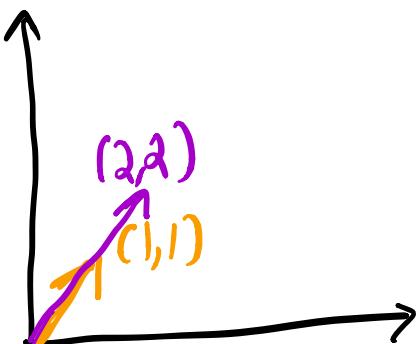
the numerator is a dot product, and indicates the extent to which the a and b vectors agree in their direction.

the denominator is a "normalizing" factor that helps to account for the different magnitudes of a and b.

A. I love $\Rightarrow [1, 1]$

B. I I love love $\Rightarrow [2, 2]$

basically the same document!



Cosine similarity measures the "cosine" of the angle created by the two vectors. In this case, it is

0° : $\cos(0) = 1$, which means maximum similarity.

$$\cos(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{1 \times 2 + 1 \times 2}{\sqrt{1^2 + 1^2} \times \sqrt{2^2 + 2^2}} = \frac{4}{\sqrt{2} \times \sqrt{8}} = \frac{4}{\sqrt{16}} = 1$$

Remember how Euclidean distance was sensitive to the magnitudes of the vectors?

A. I love $\Rightarrow [1, 1]$ $\cos(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$

B. III hellocat $\Rightarrow [3, 3]$ $= \frac{1 \times 3 + 1 \times 3}{\sqrt{1^2 + 1^2} \times \sqrt{3^2 + 3^2}} =$

$\cos(A, B)$ is still 1! $= \frac{6}{\sqrt{2} \times \sqrt{18}} = \frac{6}{\sqrt{36}} = 1$

Cosine similarity is more frequently used in NLP because all of our text data is typically sparse matrices. In my examples, I used only a dimension of 2, but in practice, we'll often have dimensions $> 20,000$ (typically one dimension for each unique word in the English language). This means when you count vectorize your document, most of your elements will be 0s: $\sim 20,000$ elements

A. I love cats \Rightarrow [I 00000010000 ... 000100000000 dogs 000000000000 cats 1000000000 late]

B. I hate dogs $\Rightarrow [0000000100000000\ldots000000100000000000000000000000]$

Cosine similarity's dot product takes into account only the elements where the elements are non-zero. Thus, it typically performs better on high-dimensional data than Euclidean distance.

* Make sure ... *

given documents, you can

1. convert to count vectors
 2. find Euclidean distance
 3. find cosine similarity
 4. understand the relationship between cosine distance and similarity:

$$\cos \text{distance} = 1 - \text{similarity}$$

$\cos \text{similarity} = 1 - \text{distance}$.