COP509

Natural Language Processing

Document Similarity and Distance Measures

Prof. Georgina Cosma

Department of Computer Science
Loughborough University

g.cosma@lboro.ac.uk

Learning Outcomes

By the end of this session, you will be able to:

1. Explain the difference between similarity and distance
2. Apply five different measures to the same example
3. Explain why cosine similarity is the standard for text
4. Identify when Euclidean, Manhattan, Hamming, and Jaccard are appropriate
5. Choose the right measure for a given task

Prerequisites

This lecture builds on the Vector Space Model (Parts 1–3). You should be comfortable with TF-IDF vectors and cosine similarity before proceeding.

Outline

Part 1

Similarity vs Distance

Similarity and Distance Are Opposites

Similarity:

- **Higher** = more alike
- Usually 0 to 1
- 1 = identical
- 0 = nothing in common

Examples: Cosine, Jaccard

Distance:

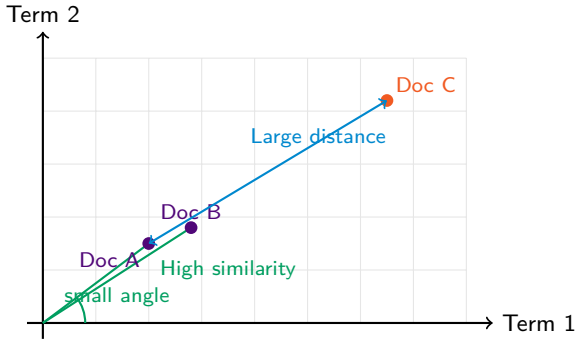
- **Lower** = more alike
- 0 to ∞
- 0 = identical
- Large = very different

Examples: Euclidean, Manhattan, Hamming

The Key Rule

High similarity = low distance, and vice versa. They measure the same thing from opposite ends.

Visualising the Difference



- **Cosine similarity:** measures the angle between vectors. Small angle = high similarity.
- **Euclidean distance:** measures the straight-line gap between points. Large gap = low similarity.

Part 2

Our Running Example

We have three documents. We will compare them using five different measures to see which measures give sensible results for text.

- **Doc A:** A *short* sports article (mentions team, goal, score)
- **Doc B:** A *long* sports article on the *same topic* (same words, twice as many)
- **Doc C:** A technology article (completely different topic)

Term-frequency vectors: Vocabulary = {team, goal, score, code, data}

	team	**goal**	**score**	**code**	**data**
Doc A (short sports)	3	2	1	0	0
Doc B (long sports)	6	4	2	0	0
Doc C (technology)	0	0	0	4	3

Key observation: Doc B = 2× Doc A. They discuss exactly the same topic. A good measure should recognise A and B as the most similar pair.

What Should the Right Answer Be?

Before we calculate anything, let us think about what a sensible measure should tell us:

Comparison	**Expected result**
Doc A vs Doc B	**Very similar** (same topic, different lengths)
Doc A vs Doc C	**Not similar at all** (no shared vocabulary)
Doc B vs Doc C	**Not similar at all** (no shared vocabulary)

The test: any measure that says A and B are *not* similar is being fooled by document length.

Let us now apply each measure and see which ones pass this test.

Part 3

Measure 1: Cosine Similarity

Covered in detail in the VSM Part 3 lectures. Here is the key formula:

$$\cos(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| \times |\vec{d}_2|}$$

$\vec{d}_1 \cdot \vec{d}_2$	Dot product: multiply matching elements and sum
$\lvert\vec{d}_1\rvert, \lvert\vec{d}_2\rvert$	Vector lengths: $\sqrt{d_1^2 + d_2^2 + \cdots + d_n^2}$
Result range	0 (nothing in common) to 1 (identical direction)

Key idea: measures the **angle** between two vectors. Ignores how long the vectors are, focuses only on the direction they point in.

Cosine Similarity: Applied to Our Example

A vs B (short sports vs long sports):

$$\text{Dot product} = (3 \times 6) + (2 \times 4) + (1 \times 2) + 0 + 0 = 18 + 8 + 2 = 28$$

$$|\vec{A}| = \sqrt{9 + 4 + 1 + 0 + 0} = \sqrt{14} \approx 3.74 \quad |\vec{B}| = \sqrt{36 + 16 + 4 + 0 + 0} = \sqrt{56} \approx 7.48$$

$$\cos = \frac{28}{3.74 \times 7.48} = \frac{28}{28.0} = \mathbf{1.00}$$

A vs C (sports vs technology):

$$\text{Dot product} = 0 + 0 + 0 + 0 + 0 = 0 \qquad \cos = \frac{0}{3.74 \times 5.0} = \mathbf{0.00}$$

B vs C: dot product $= 0$, so $\cos = \mathbf{0.00}$

Verdict: Pass

A and B have identical direction (cosine $= 1$). C shares no terms with either (cosine $= 0$). Document length has no effect.

Part 4

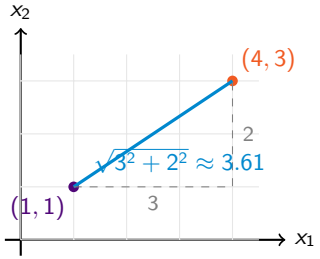
Measure 2: Euclidean Distance

Euclidean Distance: What Is It?

The straight-line distance between two points. The same idea as using a ruler on a map.

$$d_{\text{Euclidean}}(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

In words: for each dimension, find the difference, square it, add them all up, take the square root.



Key Properties

Euclidean Distance: Applied to Our Example

A vs B (short sports vs long sports):

$$d = \sqrt{(3-6)^2 + (2-4)^2 + (1-2)^2 + 0 + 0} = \sqrt{9+4+1} = \sqrt{14} \approx \mathbf{3.74}$$

A vs C (sports vs technology):

$$d = \sqrt{(3-0)^2 + (2-0)^2 + (1-0)^2 + (0-4)^2 + (0-3)^2} = \sqrt{9+4+1+16+9} = \sqrt{39} \approx \mathbf{6.24}$$

B vs C (long sports vs technology):

$$d = \sqrt{(6-0)^2 + (4-0)^2 + (2-0)^2 + (0-4)^2 + (0-3)^2} = \sqrt{36+16+4+16+9} = \sqrt{81} = \mathbf{9.00}$$

Verdict: Fail

A and B are the same topic, yet Euclidean says they are 3.74 apart. Worse, it says B is further from C (9.00) than A is from C (6.24), purely because B is a longer document.

Part 5

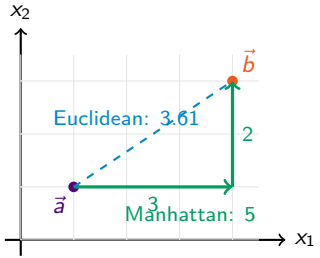
Measure 3: Manhattan Distance

Manhattan Distance: What Is It?

Also called "city block" or "taxicab" distance. Instead of a straight line, you travel along the grid like a taxi.

$$d_{\text{Manhattan}}(\vec{a}, \vec{b}) = \sum_{i=1}^{n} |a_i - b_i|$$

In words: for each dimension, find the absolute difference, then add them all up.



Key Properties

Manhattan Distance: Applied to Our Example

A vs B (short sports vs long sports):

$$d = |3-6| + |2-4| + |1-2| + |0-0| + |0-0| = 3+2+1+0+0 = \mathbf{6}$$

A vs C (sports vs technology):

$$d = |3-0| + |2-0| + |1-0| + |0-4| + |0-3| = 3+2+1+4+3 = \mathbf{13}$$

B vs C (long sports vs technology):

$$d = |6-0| + |4-0| + |2-0| + |0-4| + |0-3| = 6+4+2+4+3 = \mathbf{19}$$

Verdict: Fail

Same problem as Euclidean. A and B should be distance 0 (same topic), but Manhattan says 6. And B vs C is inflated to 19 simply because B has larger numbers.

Part 6

Measure 4: Hamming Distance

Hamming Distance: What Is It?

Count the number of positions where two sequences differ.
Both sequences must be the **same length**.

$$d_{\text{Hamming}}(\vec{a}, \vec{b}) = \text{number of positions where } a_i \neq b_i$$

Example with binary strings:

String A:	1	0	**1**	1	**1**	0
String B:	1	0	**0**	1	**0**	0
Match?	✓	✓	×	✓	×	✓

Hamming distance = 2 (two positions differ)

Also works for words (e.g. spell-checking): "karolin" vs "kathrin" = 3 (positions 3, 4, 5 differ).

Hamming Distance: Applied to Our Example

To use Hamming, we first convert to **binary vectors** (1 = term present, 0 = absent):

	team	**goal**	**score**	**code**	**data**
Doc A	1	1	1	0	0
Doc B	1	1	1	0	0
Doc C	0	0	0	1	1

A vs B: all 5 positions match $\rightarrow$ Hamming $= \mathbf{0}$
A vs C: all 5 positions differ $\rightarrow$ Hamming $= \mathbf{5}$
B vs C: all 5 positions differ $\rightarrow$ Hamming $= \mathbf{5}$

Verdict: Pass (but with a catch)

Correctly identifies A and B as identical. However, Hamming only works with binary data, so it throws away all frequency information. It cannot tell apart a document that mentions "team" 3 times from one that mentions it 100 times.

Part 7

Measure 5: Jaccard Similarity

Jaccard Similarity: What Is It?

Measures the overlap between two sets. Of all the items in either set, how many appear in both?

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{items in both}}{\text{items in either}}$$

Important: Jaccard treats documents as **sets of words**. It only cares whether a word is present or absent. It ignores how many times a word appears.

- $J = 1$: the two sets are identical
- $J = 0$: the two sets share nothing

Jaccard distance is simply: $d_{\text{Jaccard}} = 1 - J(A, B)$

Jaccard Similarity: Applied to Our Example

First, convert each document to a **set of words** (ignoring counts):

- Doc A: {team, goal, score}
- Doc B: {team, goal, score}
- Doc C: {code, data}

A vs B:

$$\text{Intersection} = \{\text{team, goal, score}\}, \quad \text{Union} = \{\text{team, goal, score}\}$$

$$J = \frac{3}{3} = \mathbf{1.0}$$

A vs C: Intersection $= \emptyset$, Union $= \{\text{team, goal, score, code, data}\} \rightarrow J = \frac{0}{5} = \mathbf{0.0}$
B vs C: same as A vs C $\rightarrow J = \mathbf{0.0}$

Verdict: Pass (but with a catch)

Correctly identifies A and B as identical. But like Hamming, Jaccard ignores frequency. It cannot tell apart "cat cat cat dog" from "cat dog dog dog" since both have the set {cat, dog}.

Part 8

The Full Picture

All Five Measures Compared

	A vs B	A vs C	B vs C	A=B same?	C different?
Cosine sim	**1.00**	0.00	0.00	✓	✓
Euclidean dist	3.74	6.24	9.00	×	✓
Manhattan dist	6	13	19	×	✓
Hamming dist	0	5	5	✓	✓
Jaccard sim	1.00	0.00	0.00	✓	✓

What this tells us:

- **Euclidean and Manhattan fail** because document length inflates the distance between documents on the same topic.
- **Cosine, Jaccard, and Hamming** all correctly identify A and B as the same topic.
- But Jaccard and Hamming throw away frequency information.
- **Cosine similarity** is the only measure that handles both document length *and* word frequency correctly.

When to Use Each Measure

Measure	What it measures	Range	Type	Best for
Cosine	Angle between vectors	[0, 1]	Similarity	**Text and documents**
Euclidean	Straight-line gap	$[0, \infty)$	Distance	Continuous numerical data
Manhattan	Grid-based gap	$[0, \infty)$	Distance	Grid data, reducing outlier effects
Hamming	Positions that differ	$[0, n]$	Distance	Binary strings, categorical data
Jaccard	Set overlap ratio	[0, 1]	Similarity	Tags, categories, presence/absence

The Rule for This Course

- **Comparing documents or queries?** Use **cosine similarity**. Always.
- **Comparing sets of tags or categories?** Use Jaccard.
- **Comparing binary or categorical sequences?** Use Hamming.
- **Working with spatial or numerical data?** Use Euclidean or Manhattan.

Part 9

Practical Applications in NLP

1. User types a query
2. Convert the query to a TF-IDF vector
3. Compute **cosine similarity** between the query and every document
4. Rank documents by similarity (highest first)
5. Return the top results

Example: Query = "machine learning algorithms"

Document	Cosine Similarity	Rank
Article on ML techniques	0.91	**1**
Brief note on algorithms	0.58	2
Article on quantum physics	0.00	3

The ML article ranks first because it shares the most vocabulary with the query.

Application 2: Document Clustering

Goal: automatically group similar documents together.

How it works (K-means with cosine similarity):

1. Represent each document as a TF-IDF vector
2. Pick k starting cluster centres
3. Assign each document to the cluster with the highest cosine similarity
4. Update cluster centres (average of member vectors)
5. Repeat steps 3–4 until the clusters stop changing

Result: documents naturally group by topic:

- Cluster 1: Sports articles
- Cluster 2: Politics articles
- Cluster 3: Technology articles

Use cases: organising news feeds, grouping customer support tickets, discovering topics in large collections.

Application 3: Plagiarism Detection

Goal: find suspiciously similar student essays.

How it works:

1. Convert each essay to a TF-IDF vector
2. Compute cosine similarity for every pair of essays
3. Flag pairs above a threshold (e.g. 0.85)
4. Manually review the flagged pairs

Why it works: copied essays have very similar word frequencies, even if some words are changed.

Limitations:

- Does not catch sophisticated paraphrasing (different words, same ideas)
- Good as a first-pass screening tool, not a final verdict
- More advanced tools use sentence embeddings for deeper comparison

"If you liked this article, you might also like..."

Content-based filtering:

1. User reads Article A
2. Compute cosine similarity between Article A and all other articles
3. Recommend the top k most similar articles

Example: User reads "Introduction to Neural Networks"

Article	Cosine Similarity
"Deep Learning Basics"	0.89
"Backpropagation Explained"	0.82
"Machine Learning Overview"	0.78

Advantage: works even for brand new articles with no user ratings.

Application 5: Document Classification (k-NN)

Goal: classify a new document into a category.

k-Nearest Neighbours algorithm:

1. Have a training set of labelled documents
2. New (unlabelled) document arrives
3. Compute cosine similarity between the new document and all training documents
4. Find the k most similar training documents
5. The new document gets the most common label among those k neighbours

Example ($k = 3$): New document = "The football team won the championship"

Neighbour	Similarity	Label
Training doc 1	0.91	Sports
Training doc 2	0.87	Sports
Training doc 3	0.65	Politics

Classification: Sports (2 out of 3 neighbours).

Part 10

Self-Test Questions

Test Your Understanding

Try these before looking at the answers on the next slides:

1. Why is cosine similarity better than Euclidean distance for comparing documents?
2. Given $\vec{a} = (4, 0, 2)$ and $\vec{b} = (2, 0, 1)$, calculate:
   (a) Cosine similarity
   (b) Euclidean distance
   (c) Manhattan distance
3. If two document vectors have a cosine similarity of 0, what does that mean?
4. Doc A = {python, code, debug} and Doc B = {python, code, test, deploy}. What is the Jaccard similarity?
5. When would you use Jaccard similarity instead of cosine similarity?

1. Why is cosine similarity better than Euclidean distance for comparing documents?
Cosine measures the *angle* (direction/topic), not the magnitude (length). A short article and a long book on the same topic get high cosine similarity, but Euclidean distance would incorrectly say they are far apart because of the difference in word counts.

2. Given $\vec{a} = (4, 0, 2)$ and $\vec{b} = (2, 0, 1)$:
(a) Cosine: Dot product $= 8 + 0 + 2 = 10$. Lengths: $\sqrt{20}$ and $\sqrt{5}$. Product of lengths: $\sqrt{20} \times \sqrt{5} = \sqrt{100} = 10$. Cosine $= \frac{10}{10} = \mathbf{1.0}$
Note: $\vec{a} = 2 \times \vec{b}$, so they point in exactly the same direction.
(b) Euclidean: $\sqrt{(4-2)^2 + (0-0)^2 + (2-1)^2} = \sqrt{4+0+1} = \sqrt{5} \approx \mathbf{2.24}$
(c) Manhattan: $|4-2| + |0-0| + |2-1| = 2 + 0 + 1 = \mathbf{3}$

3. Cosine similarity of 0?
The vectors are orthogonal (90° angle). They share no common terms at all.

4. Jaccard similarity of Doc A = {python, code, debug} and Doc B = {python, code, test, deploy}?

- Intersection: {python, code} $\rightarrow$ size $= 2$
- Union: {python, code, debug, test, deploy} $\rightarrow$ size $= 5$
- Jaccard similarity $= \frac{2}{5} =$ **0.4**

5. When would you use Jaccard instead of cosine?

Use Jaccard when:

- You only care about *presence or absence* of items, not how often they appear
- Comparing sets of tags, hashtags, or categories
- Working with very short texts where word frequency is not meaningful
- Example: comparing product tags, user interests, or hashtags in tweets

Use cosine when word frequency matters (which is most document comparison tasks).

Key Takeaways

1. **Similarity** (higher = more alike) and **distance** (lower = more alike) are opposite ways of measuring the same thing.
2. **Cosine similarity** is the standard for text. It ignores document length and focuses on content.
3. **Euclidean** and **Manhattan** distances are useful for numerical data but not for text, because they are sensitive to magnitude.
4. **Hamming distance** counts position differences in equal-length sequences.
5. **Jaccard similarity** measures set overlap, ignoring word frequency.
6. These measures power real applications: search engines, clustering, plagiarism detection, recommendations, and classification.

Questions?

Prof. Georgina Cosma

g.cosma@lboro.ac.uk