

# Cosine Similarity and Its Applications in AI

David Oniani

Luther College

[oniada01@luther.edu](mailto:oniada01@luther.edu)

April 22, 2020

## Abstract

Choosing the right metric [1] can be crucial to designing performant artificial intelligence models. Thousands of packages and libraries have been built and written just for providing these metrics. Cosine similarity is one of many metrics used extensively in natural language processing and artificial intelligence tasks. The paper will introduce the technique and discuss its advantages and disadvantages as well as compare it to other approaches. Additionally, sample implementations of the above-mentioned approaches will also be provided.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Text Document Modeling	3
1.2	Cosine Similarity	3
1.3	Example	4
<b>2</b>	<b>Applications</b>	<b>4</b>
<b>3</b>	<b>Performance and Comparison to Other Approaches</b>	<b>5</b>
<b>4</b>	<b>Current Work</b>	<b>6</b>
<b>5</b>	<b>Summary</b>	<b>6</b>
	<b>References</b>	<b>6</b>

# 1 Introduction

There are many approaches for determining whether two texts are semantically similar to each other. Cosine similarity is one of those methods. Initially arisen from the law of cosines [2], the technique is now widely used in natural language processing (and other) problems.

In order to understand how cosine similarity works, let us first discuss how a text document can be modeled.

## 1.1 Text Document Modeling

There are several ways in which a text document can be modeled. This includes a *bag of words* modeling, where the frequency of a term in a text document represents its weight and therefore, more frequent words are deemed more “important.” Another example is *tf-idf* vectorization. The primary goal of text document modeling is to transform the textual data into the numeric data (usually, term vectors). Once the numeric data is obtained, we can then apply various text semantic similarity techniques in order to compare if two documents are similar to each other.

## 1.2 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them [3]. If we have two vectors  $\vec{a}$  and  $\vec{b}$ , then the cosine of these two vectors is  $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \theta$ , where  $\theta$  is the angle between the vectors. This is also known as the Euclidean dot product formula. The cosine of two vectors can also be computed using their coordinates: if  $\vec{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$  and  $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ , then  $\vec{a} \cdot \vec{b} = a_1b_1 + a_2b_2$ . Hence, for two vectors  $\vec{a}$  and  $\vec{b}$  with coordinates  $a_1, a_2$  and  $b_1, b_2$  respectively, the following holds:

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \theta = a_1b_1 + a_2b_2$$

If the vector coordinates are known, then one can also compute the magnitude of the vector. For

vector  $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$ , the magnitude is  $\sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$ . Hence, if we have two vectors  $\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix}$

and  $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$ , then the cosine of an angle between them can be computed using the formula:

$$\cos \theta = \frac{a_1 b_1 + a_2 b_2 + \dots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \dots + a_n^2} \sqrt{b_1^2 + b_2^2 + \dots + b_n^2}}$$

In other words, if we have two vectors with all the coordinates, it is possible to compute the cosine similarity between them.

### 1.3 Example

As an example, consider two vectors  $\vec{a} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$  and  $\vec{b} = \begin{bmatrix} 5 \\ 12 \end{bmatrix}$ . Then the magnitude of  $a$  is  $\sqrt{3^2 + 4^2} = 5$  and the magnitude of  $b$  is  $\sqrt{5^2 + 12^2} = 13$ . It follows that  $\vec{a} \cdot \vec{b} = 15 + 48 = 63$  and  $|\vec{a}| \times |\vec{b}| = 5 * 13 = 65$ . Therefore, the cosine of an angle between the vectors is  $\frac{63}{65} \approx 0.969$  which is the similarity measure between these two vectors.

## 2 Applications

Cosine similarity is one of the most commonly used approaches in calculating semantic similarity across the documents. Therefore, it is naturally employed in natural language processing tasks. Many NLP applications need to compute the similarity in meaning between two short texts. Search engines, for instance, need to model the relevance of a document to a query, beyond the overlap in words between the two. Similarly, Q&A websites such as Quora need to determine whether a question has already been asked before. This type of text similarity is often computed by first embedding the two short texts and then calculating the cosine similarity between them.

Suppose that we have two documents  $D_1$  and  $D_2$  modeled as term vectors  $\vec{t}_1$  and  $\vec{t}_2$  respectively. Then the similarity of two documents corresponds to the correlation between the vectors and can be quantified as a cosine of the angle between the vectors. The cosine similarity formula [4] would be:

$$SIM_C(\vec{t}_1, \vec{t}_2) = \frac{\vec{t}_1 \cdot \vec{t}_2}{|\vec{t}_1| \times |\vec{t}_2|}.$$

As a result, the similarity value is non-negative, bounded by the closed interval  $[0, 1]$ .

It is important to note that the metric is a measurement of orientation and not the magnitude. It is really a comparison of documents on a normalized space since it does not only consider the magnitude of word counts of each document, but also the angle between the documents.

Below find a Python implementation of cosine similarity.

```
import numpy as np

def cosine_similarity(a: np.array, b: np.array) -> float:
    """Returns the cosine similarity value of two vectors."""

    dot_product: float = np.dot(a, b)
    magnitude_a: float = np.linalg.norm(a)
    magnitude_b: float = np.linalg.norm(b)
    similarity: float = dot_product / (magnitude_a * magnitude_b)

    return similarity

if __name__ == "__main__":
    a: np.array = np.array([1, 0, 1, 0, 1, 1, 1, 1, 1, 1])
    b: np.array = np.array([0, 2, 1, 0, 1, 3, 0, 0, 0, 1])
    print(cosine_similarity(a, b)) # Prints out 0.5
```

Figure 1: Sample Python Implementation of Cosine Similarity.

The program above is fairly simplistic (yet performant) and as shown, it takes two vectors and prints out the cosine similarity.

### 3 Performance and Comparison to Other Approaches

Performance of cosine similarity can be altered by the approach used for vectorization [5] (e.g., tf-idf vectorization might yield better results than Word2Vec vectorization). This means that the right vectorization strategy could drastically increase the performance, while the poor vectorization will, unsurprisingly, result in the poor performance.

There are many other approaches for finding semantic similarity between the texts. The recent advancements in the domains of artificial intelligence and natural language processing allowed for development of models such as BERT [6], BioBERT [7], and Universal Sentence Encoder (USE) [8], all of which can be used for high-accuracy text semantic similarity score computations. The advantage of the cosine similarity, however, is its simplicity and the fact that one could run it over massive datasets. Doing the same with models like BioBERT could possibly take a lot time and hence, be highly inefficient.

## 4 Current Work

Cosine similarity is very much used in the modern, state-of-the-art papers, such as ones cited in this paper. Its flexibility allows one to apply it virtually under any setting, as long as documents can be represented as vectors. Besides, it is widely used for benchmarking purposes and is usually a standard against which the new approaches are usually.

## 5 Summary

We have introduced cosine similarity and walked through examples to strengthen the understanding of the concept. We have discussed the importance of cosine similarity in the domains of artificial intelligence and natural language processing and its prevalence in the state-of-the-art work. Sample Python implementation of the algorithm was also provided.

## References

- [1] Rachel Thomas and David Uminsky. “The Problem with Metrics is a Fundamental Problem for AI”. In: (2020). URL: <https://arxiv.org/abs/2002.08512>.
- [2] Wikipedia The Free Encyclopedia. *Cosine Formula for Dot Product*. URL: [https://proofwiki.org/wiki/Cosine\\_Formula\\_for\\_Dot\\_Product](https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product).
- [3] Wikipedia The Free Encyclopedia. *Cosine similarity*. URL: [https://en.wikipedia.org/wiki/Cosine\\_similarity](https://en.wikipedia.org/wiki/Cosine_similarity).

- [4] Anna Huang. “Similarity measures for text document clustering”. In: *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, Christchurch, New Zealand. 2008.
- [5] Pinky Sitikhu et al. “A Comparison of Semantic Similarity Methods for Maximum Human Interpretability”. In: (2019).
- [6] Iulia Turc et al. “Well-Read Students Learn Better: On the Importance of Pre-training Compact Models”. In: *arXiv preprint arXiv:1908.08962v2* (2019).
- [7] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* (Sept. 2019). ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btz682](https://doi.org/10.1093/bioinformatics/btz682). URL: <https://doi.org/10.1093/bioinformatics/btz682>.
- [8] Daniel Cer et al. “Universal Sentence Encoder”. In: (2018). eprint: [arXiv:1803.11175](https://arxiv.org/abs/1803.11175).