Cosine Similarity and Its Applications in AI

David Oniani

Luther College

oniada01@luther.edu

April 19, 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

1 Introduction

There are a number of approaches for comparing whether two texts are semantically similar to each other. Cosine similarity is one of those methods. In order to understand how cosine similarity works, let us first discuss the modeling of a text document and cosine similarity in general and then proceed by its applications in text document similarity tasks.

1.1 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." The whole idea behind a text document modeling is to quantify the textual data into the numeric data (usually, vectors). Once the numeric data is obtained, we can then apply various text semantic similarity techniques in order to compare whether 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 [2]. If we have two vectors $\vec{a}$ and $\vec{b}$, then the cosine of these two vectors is $\vec{a} \cdot \vec{b}$ which is equal to $\|\vec{a}\| \left\|\vec{b}\right\| \cos\theta$ where θ is the angle between these vectors (Euclidean dot product formula). It can also be represented as the product of two vectors. Therefore if $\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$, then $\vec{a} \cdot \vec{b}$ is also equal to $a_1b_1 + a_2b_2 + a_3b_3$.

But why do we care about cosine of theta? Where does trig fit in with the dot product? Who thought of that? The answer comes from the law of cosines. A great walkthrough is available on the proof wiki [3].

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 size of a is $\sqrt{1^2 + 2^4} = 5$ and the size of b is $\sqrt{3^2 + 4^2} = 13$. Then $\vec{a} \cdot \vec{b} = 15 + 48 = 63$ and $|\vec{a}| \times |\vec{b}| = 5 * 13 = 65$. Therefore, the cosine of an angle between these vectors is $\frac{63}{65}$ which is the similarity measure between these two vectors.

2 Applications

Cosine similarity is one of the most commonly used approaches in text document similarity. Its flexibility allows one to apply it in virtually any setting, as long as the documents can be represented as vectors (also known as term vectors).

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 formula would be

$$SIM(D_1, D_2) = \frac{\vec{t_1} \cdot \vec{t_2}}{|\vec{t_a}| \times |\vec{t_b}|}.$$

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

```python
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)
    size_a: float = np.linalg.norm(a)
    size_b: float = np.linalg.norm(b)
    similarity: float = dot_product / (size_a * size_b)

    return similarity



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

Figure 1: Sample Python Implementation of Cosine Similarity.

3 Comparison to Other Approaches

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 [4], BioBERT [5], and Universal Sentence Encoder (USE) [6] can be used for high-accuracy text semantic similarity score computations. These approaches are thought to, on average, outperform the cosine similarity, yet there are a number of efforts in determining this.

4 Current Work

5 Summary

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 similarity.* URL: https://en.wikipedia.org/wiki/Cosine_similarity.

[3] Wikipedia The Free Encyclopedia. *Cosine Formula for Dot Product.* URL: https://proofwiki.org/wiki/Cosine_Formula_for_Dot_Product.

[4] Iulia Turc et al. "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models". In: *arXiv preprint arXiv:1908.08962v2* (2019).

[5] 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. URL: https://doi.org/10.1093/bioinformatics/btz682.

[6] Daniel Cer et al. "Universal Sentence Encoder". In: (2018). eprint: arXiv:1803.11175.