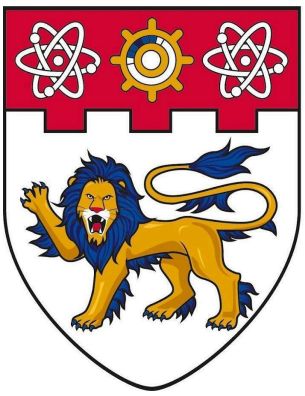


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4020 Data Analytics and Mining

Project 1

Group Members:

Full Name	Matriculation No.
Yap Hong Sheng	U2320913A
Toh Jun Sheng	U2322617B
Leong Wei Zhe	U2223783F
Hasley Hong Zheng Yi	U2220690C

Abstract

This study presents a comprehensive evaluation of three widely used similarity search methods: Cosine Similarity, Euclidean Distance, and Jaccard Similarity, applied to both textual and image datasets. The objective is to assess the effectiveness, accuracy, strengths, and limitations of each method in identifying semantically or visually similar data pairs.

For text similarity, the “STSbenchmark.xlsx” dataset was used, and sentence embeddings were generated using the “all-MiniLM-L6-v2” transformer model. Experimental results show that Cosine Similarity and Euclidean Distance both achieve high accuracy in capturing semantic relationships and contrasting meanings, with Euclidean Distance particularly effective in identifying magnitude differences between vector embeddings. In contrast, Jaccard Similarity proves suitable for fast, surface-level lexical comparisons, such as duplicate or near-duplicate detection.

For image similarity, the “image_pairs.xlsx” dataset was analysed using feature vectors extracted with the ResNet-50 convolutional neural network (CNN) model. Among the three similarity search methods, Cosine Similarity achieved the highest accuracy of 94%, followed by Euclidean Distance at 91.5% and Jaccard Similarity at 81%. These results indicate that Cosine Similarity is most effective in capturing both semantic and structural relationships, Euclidean Distance performs well in identifying intensity-based variations, and Jaccard Similarity is best suited for fast, colour or pixel-level comparisons.

1 Introduction

Data mining is the process of applying various techniques, such as statistical or mathematical methods, to uncover meaningful correlations, patterns, and trends within large amounts of data [1]. In the world of data, there are billions of documents, images, and products, and relying on brute force exact matching is both time-consuming and prone to missing relevant results. To improve efficiency, these items are first converted into a machine-readable representation, such as vectors or embeddings. Once the items are represented as values, a similarity search can be performed to retrieve items that are semantically or visually close to a query. Similarity search is widely used in many practical applications, including content-based image retrieval, recommendation systems, plagiarism detection and semantic document search. A particularly notable example is in Retrieval-Augmented Generation (RAG) chatbots, where a user's query is first embedded and compared against a large knowledge base of documents using similarity search. The most relevant documents are then retrieved to augment the chatbot's response generation. This will result in more accurate and contextually informed answers.

This study focuses on evaluating and comparing the performance of three widely used similarity search techniques, Cosine similarity, Euclidean distance and Jaccard similarity across textual datasets and image datasets.

2 Methods

2.1 Cosine Similarity

Cosine similarity is a widely used method in data mining and information retrieval that measures the similarity between two vectors by calculating the cosine of the angle between them. It pivots on the orientation of vectors while ignoring their magnitude, and this makes it particularly useful for high-dimensional datasets, such as text documents or feature embeddings extracted from images [2]. Mathematically, cosine similarity is defined as

$$Cosine\ Similarity\ (A, B) = \frac{A \cdot B}{||A|| \times ||B||}$$

where $A \cdot B$ is the dot product of vectors A and B, and $||A||$ and $||B||$ denote their magnitudes. Haskova et al. [3] stated that the angle between the vectors determines whether they are pointing in the same or opposite directions. Vectors that are positioned closer together along the axis are more similar, whereas those that are farther apart are less similar.

2.2 Jaccard Similarity

Jaccard similarity is a method for measuring the similarity between two sets. It is commonly used in various data science applications such as text mining, e-commerce, and recommendation systems. Specifically, Jaccard similarity is a measure of similarity between two asymmetric binary vectors. Suppose a binary variable has only two states, 0 and 1, where 0 denotes the absence of the attribute, while 1 denotes its presence. An asymmetric binary model only focuses on the co-occurring positive (1-1 matches). Mathematically, the Jaccard similarity can be represented by the Jaccard index J(A, B) and computed as the size of the intersection divided by the size of the union of two sets A and B, which can be represented in set notation using intersection (A n B) and unions (A U B) of two sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

If two data sets contain exactly the same elements, $|A \cap B| = |A \cup B|$, resulting in a Jaccard index of 1. Sometimes, the Jaccard distance, D_J(A, B) is further calculated to illustrate the dissimilarity between the two sets, which can be computed through

$$D_J(A,B) = 1 - J(A,B)$$

2.3 Euclidean Distance

One of the most fundamental and widely used approaches for measuring similarity is the Euclidean distance method. It quantifies the straight line distance between two points in a multidimensional space and makes it an intuitive and simple metric. Its effectiveness lies in its ability to capture geometric relationships between the data points, which is useful when dealing with continuous and numerical features. In the similarity search context, vectors that are closer in the space are assumed to be more similar. Owing to its simplicity and strong interpretability, Euclidean distance serves as a baseline for evaluating more complex similarity measures and continues to be a cornerstone in both exact and approximate search methods. The Euclidean distance can be calculated using the formula below.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3 Experiments

3.1 Text Dataset

The dataset named "`STSbenchmark.xlsx`" is used in this experiment. A sample of the first 10 rows from the dataset is shown below in Table 1.

Table 1 Sample of the Dataset "`STSbenchmark.xlsx`"

score	sentence1	sentence2	similarity
5	A man with a hard hat is dancing.	A man wearing a hard hat is dancing.	1
4.75	A young child is riding a horse.	A child is riding a horse.	1
5	A man is feeding a mouse to a snake.	The man is feeding a mouse to the snake.	1
2.4	A woman is playing the guitar.	A man is playing guitar.	0
2.75	A woman is playing the flute.	A man is playing a flute.	0
2.615	A woman is cutting an onion.	A man is cutting onions.	0
5	A man is erasing a chalk board.	The man is erasing the chalk board.	1
2.333	A woman is carrying a boy.	A woman is carrying her baby.	0
3.75	Three men are playing guitars.	Three men are on stage playing guitars.	1
5	A woman peels a potato.	A woman is peeling a potato.	1

The "score" column represents the similarity score between sentence 1 and sentence 2, which is based on human-annotated evaluation of how similar the two sentences are. This similarity score serves as a key reference for evaluating the accuracy of the implemented similarity search methods (cosine similarity, Euclidean distance and Jaccard similarity). By comparing the search methods' similarity classification against these reference scores, the performance of the search methods can be assessed more objectively. The "similarity" column lists the binary scores for all the sentence pairs in the dataset. A value of 1 indicates that the sentences are similar, while 0 indicates they are not.

3.1.1 Procedure to Generate Similarity Score

3.1.1.1 Pre-Processing of Data

For the cosine similarity and Euclidean distance method, a pre-trained sentence transformer `'all-MiniLM-L6-v2'` was loaded to convert the sentences into vector embeddings that capture their semantic meaning. The data is further pre-processed to ensure all entries in “sentence 1” and “sentence 2” are strings. Each sentence is converted into a vector embedding, and the cosine similarity and Euclidean distance between the pair of vectors, i.e., the similarity of the sentence pairs, is computed and populated into another column. Unlike the cosine similarity and Euclidean distance method, the sentences are not converted into vector embeddings when using Jaccard similarity; they are simply converted into strings. Each sentence is then tokenised into individual words and stored as a set. The Jaccard similarity is subsequently computed between the two sets using the formula presented in Section 2.2.

3.1.1.2 Determination of the Threshold Value of Similarity

To determine whether the sentence pairs are similar or not based on the similarity search techniques, threshold values ranging from 0.00 to 1.00 in increments of 0.05 were tested. For example, when the threshold is set to 0.75, any sentence pair with a similarity score greater than 0.75 is assigned a binary similarity value of “1”. These binary values are then compared with the “similarity” column of the original dataset “STSbenchmark.xlsx”. Based on this comparison, the classification accuracy is computed as the percentage of correctly matched values using a confusion matrix. The accuracy values are subsequently plotted against the tested thresholds to identify the optimal threshold. This optimal threshold, which yields the highest accuracy for each similarity search method, is then used for evaluation and comparison. From Figures 1, 2 and 3, the optimal threshold values for each similarity method are identified and marked in red.

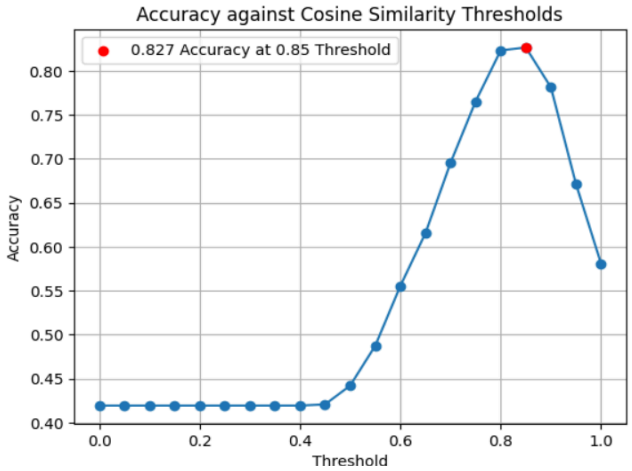


Fig 1. Accuracy against Cosine Similarity Thresholds

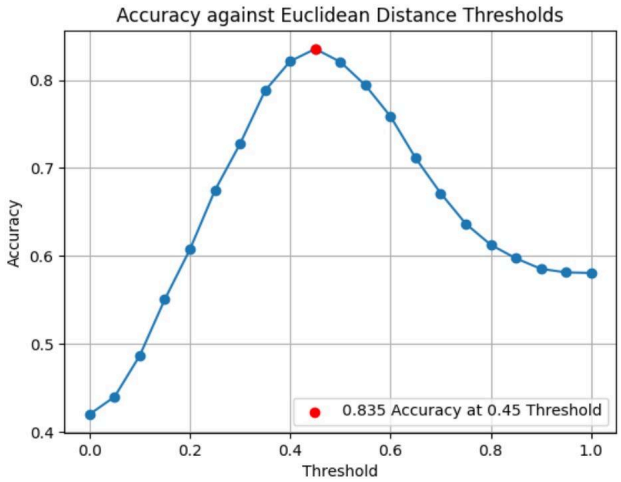


Fig 2. Accuracy against Euclidean Distance Thresholds

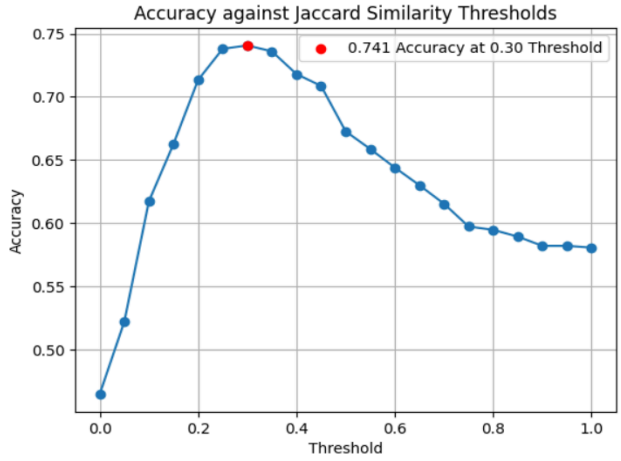


Fig 3. Accuracy against Jaccard Similarity Thresholds

Since Cosine Similarity score returns a value within the range [-1, 1] while Euclidean Distance returns the L2 distance between 2 vectors, we have opted for a few ways to normalise these values into a [0,1] scale.

For Cosine Similarity, the following equation was used.:

$$Cosine\ Similarity\ Scaled\ =\ \frac{(Cosine\ Similarity\ Score + 1)}{2}$$

For Euclidean distance, Min Max normalisation is used:

$$Euclidean\ Score\ =\ 1\ -\ \frac{Euclidean\ Distance - Minimum\ Euclidean\ Distance}{Maximum\ Euclidean\ Distance - Minimum\ Euclidean\ Distance}$$

Since Jaccard Similarity returns a ratio, its range naturally falls within [0,1] and hence no normalisation is required. These three similarity scores are subsequently scaled to match the gold standard similarity score range (0 to 5) and stored in the dataframe.

Two commonly used correlation metrics are reported: Pearson correlation, which measures the linear relationship between the methods' similarity scores and the actual similarity values, and Spearman correlation, which assesses the monotonic relationship by ranking the values. The correlation analysis evaluates how well the similarity scores produced by each method align with the ground truth similarity labels.

Based on the results, cosine similarity achieves a Pearson correlation of 0.8697 and a Spearman correlation of 0.8673. This indicates that cosine similarity has a strong linear and rank-based agreement with ground truth similarities. Euclidean distance demonstrates similar performance with cosine similarity, with a Pearson correlation of 0.8679 and a Spearman correlation of 0.8673. This implies that Euclidean distance is also highly effective at capturing semantic similarity in the dataset. Jaccard similarity has lower correlation values with a Pearson correlation of 0.596 and a Spearman correlation of 0.6017. This reflects that Jaccard similarity is limited to capturing semantic relationships due to its reliance on exact token overlap rather than contextual meaning.

3.1.2 Detailed Analysis of the Similarity Search Methods

3.1.2.1 Cosine Similarity

Table 2 shows the false positive and false negative cases classified by the cosine similarity method. False positives occur when the search method incorrectly classifies the similarity. In the examples below, the surface-level word overlap misled the search method, even though the underlying events or details are different. On the other hand, false negatives occur when cosine similarity fails to identify truly similar sentences. In both sentences of false negatives, the sentences are semantically related, but they are different in phrasing, verb choice, or focus, causing the method to miss the similarity. Overall, cosine similarity demonstrates strong performance as it is able to achieve the highest precision among the search methods. This indicates that it is effective at identifying truly similar sentence pairs while minimising false positives. However, it may still produce false classifications, particularly when the sentences are semantically similar but phrased differently. Nonetheless, it is a reliable and selective approach with balanced accuracy and robustness for measuring text similarity.

Table 2: False Positives and False Negatives of Cosine Similarity

False positive	
sentence1	**sentence2**
A monkey is karate kicking a person.	A monkey practices martial arts.
Men are fighting after a basketball game.	The men are playing a game of basketball.
False negative	
sentence1	**sentence2**
Sly Stallone's Son Sage Found Dead in LA	Sylvester Stallone's Son Found Dead At Home
19 hurt in New Orleans shooting	Police: 19 hurt in NOLA Mother's Day shooting

“A monkey is karate kicking a person” and “A monkey practices martial arts” share the same words like “monkey” and refer to the same context of martial arts, which may cause the embeddings to be moderately aligned. However, the specific actions are different, as “karate kicking a person” describes a precise action when learning martial arts, while “practices martial arts” is a more generic action. Cosine similarity, which relies on vector alignment in the embedding space, is sensitive to word overlap and semantic proximity. Therefore, the surface-level similarity and overlapping concepts likely caused the method to classify the sentences as similar, even though they describe different scenarios.

“19 hurt in New Orleans shooting” and “Police: 19 hurt in NOLA Mother's Day shooting” are false negatives. These sentences are semantically similar as both describe the same event: 19 people were injured in a shooting in New Orleans. However, the phrasing and additional context are different, resulting in a false negative value. The second sentence includes extra details, “Police” and “Mother’s Day,” which introduce additional details compared to the first sentence. Furthermore, “New Orleans” and “NOLA” may not be perfectly recognised as equivalent by the embedding model because abbreviations are used. Due to the additional details and abbreviations in the second sentence, the embeddings of the two sentences may not be close enough to surpass the similarity threshold, even though the underlying meaning is closely identical, hence the cosine similarity method returns a false negative result.

3.1.2.2 Euclidean Distance

Even when semantic meaning is encoded into vector embeddings, using Euclidean distance for similarity search can still lead to false positives and false negatives due to the way its distance is calculated in high-dimensional spaces. Unlike cosine similarity, Euclidean distance is sensitive to the vector magnitude. False positives can occur when unrelated items have similar Euclidean distances. Other than that, two embeddings with similar magnitude but different directions might appear close, resulting in false positives. Conversely, false negatives can occur when genuinely related items are separated by slightly larger but still semantically meaningful distances. Two embeddings that are pointing in the same direction but with different lengths can be considered far apart in the Euclidean space, causing false negatives.

Table 3: False Positives and False Negatives of Cosine Similarity

False positive	
sentence1	**sentence2**
A woman is cutting an onion.	A man is cutting onions.
Black domestic cat lying on brown table.	A grey cat laying on a dining table.
False negative	
sentence1	**sentence2**
A woman puts cosmetics on her eyelid.	The woman is pencilling on eye shadow.
The lady cut the tail and body of a shrimp.	A woman is cleaning a shrimp.

“Black domestic cat lying on brown table” and “A grey cat laying on a dining table” contain core semantic components. The subject is cat, action is lying or laying, which shares the same meaning, and the location is on a certain type of table. The sentence transformer maps semantically similar phrases, even with differing adjectives like black vs grey, or brown table vs dining table, to nearby points in the vector space. This will lead to a higher similarity score as the Euclidean distance is shorter. However, Euclidean distance may produce false positives in tasks that require fine-grained distinctions, such as distinguishing the object colour or type. Nevertheless, this example demonstrates that the Euclidean distance method effectively captures the shared general meaning of “cat lying on table” in two different sentences.

“A woman puts cosmetics on her eyelid” and “The woman is pencilling on eye shadow” describe the same action, but the Euclidean distance method may place their embeddings far apart, resulting in them being classified as dissimilar. One of the possible reasons is that many embedding models rely heavily on lexical or distributional similarity. The model might not map these important terms to vectors with short Euclidean distances because they are not exact synonyms. Additionally, because eyeliner and other cosmetics can be applied to the eyelids, the embedding of the first sentence is said to be more generic. Because embedding models often map general phrases to broader conceptual regions in the vector space, whereas more specialised actions may lie in a more specific subregion, their vector magnitudes and directions

may differ. Even though these embeddings are conceptually similar, they may fall outside of a predetermined similarity threshold because Euclidean distance reflects geometric distance.

3.1.2.3 Jaccard Similarity

Jaccard similarity can produce both false positives and false negatives because it measures only token overlap without considering semantic meaning. False positives occur when sentences share many common words, but different meanings and ideas are expressed, resulting in high Jaccard similarity scores despite semantic differences. Conversely, false negatives arise when semantically similar sentences use different vocabulary, synonyms or phrasing, resulting in low Jaccard similarity scores despite conveying the same meaning. Therefore, it is limited by semantic similarity search and is more suitable for tasks requiring exact or near-exact lexical matching.

Table 4: False Positives and False Negatives of Jaccard Similarity

False positive	
sentence1	**sentence2**
A man is playing the drums.	A man is playing guitar.
A man is finding something.	A woman is slicing something.
False negative	
sentence1	**sentence2**
Someone is shredding cabbage leaves with a knife.	Someone is chopping some cabbage leaves.
The man used a sword to slice a plastic bottle.	A man sliced a plastic bottle with a sword.

“A man is finding something” and “A woman is slicing something” are false positives for Jaccard similarity because they share structural and lexical similarities despite having different

meanings. Both sentences follow a similar grammatical pattern and contain overlapping words such as “is” and “something”, which artificially inflate the Jaccard similarity score.

“The man used a sword to slice a plastic bottle” and “A man sliced a plastic bottle with a sword” are false negatives under Jaccard similarity. Although both sentences describe the same event, slight lexical variations such as “used” vs. “sliced”, and accompanying reordering of words lead to differences in the token sets. Because Jaccard only measures exact word overlap and ignores context or semantics, these small variations reduce the intersection of tokens and lower the similarity score.

3.1.3 Comparison between the Similarity Search Methods

3.1.3.1 Confusion Matrix

The confusion matrix for each similarity search method is generated and presented in Figures 4, 5 and 6.

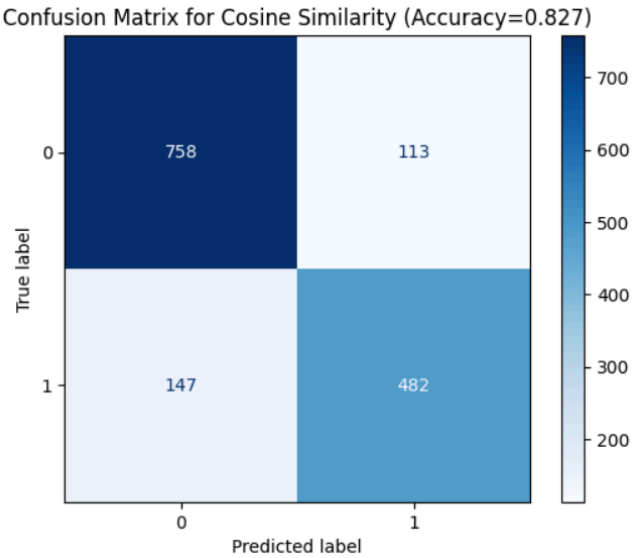


Fig 4. Confusion Matrix for Cosine Similarity

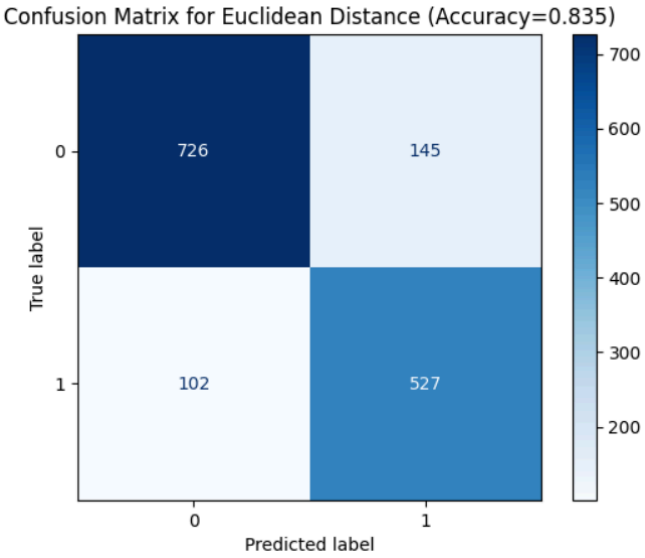


Fig 5. Confusion Matrix for Euclidean Distance

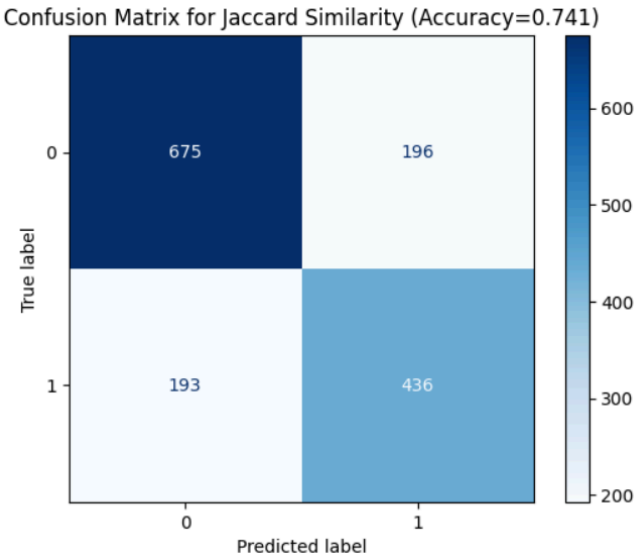


Fig 6. Confusion Matrix for Jaccard Similarity

The accuracy of the three similarity search methods reflects the percentage of the sentence pairs correctly classified by the method. As shown in Figures 4, 5 and 6, the accuracy of cosine similarity is 0.827, the Euclidean distance method is 0.835, and the Jaccard similarity is 0.741. While cosine similarity and Euclidean distance show comparable accuracy, precision and recall are introduced as complementary metrics as accuracy alone does not fully capture a method's ability to correctly identify similar text, especially in cases where the dataset contains subtle semantic variations. Precision measures the proportion of classified similar pairs that are actually correct, while recall measures the proportion of all truly similar pairs that are correctly

identified. In this evaluation, cosine similarity achieves a higher precision of 0.81 and a recall of 0.766, whereas Euclidean distance follows with a precision of 0.78 and a recall of 0.84. Jaccard similarity underperforms in both metrics due to its reliance on exact token overlap, which fails to account for synonyms, paraphrasing or differences in word order. Overall, these results suggest that cosine similarity provides the best balance between accuracy, precision and recall, making it the most effective method for similarity search in text datasets.

3.1.3.2 Strengths and Limitations of the Similarity Search Methods

Table 5: Classic Cases of Lexical Overlap Without Semantic Equivalence

Sentence pair	**similarity**	**Cosine Similarity**	**Euclidean Distance**	**Jaccard Similarity**
A woman is playing the flute. A man is playing a flute.	0	TN	TN	FP

Table 5 presents classic examples of lexically similar but semantically different sentence pairs. In these examples, many words are shared between the two sentences, such as “A”, “is”, “playing” and “flute”. The differences lie only in a few semantically critical words, for instance, “woman” vs. “man”. Since Jaccard similarity considers only token overlap and ignores semantic meaning, the similarity score remains high due to the lexical overlap. It is only well-suited for surface-level similarity detection, such as identifying duplicate or near-duplicate sentences, flagging typographical variations or performing fast lexical filtering before semantic models are applied. In contrast, tasks requiring deeper semantic understanding are better handled by embedding-based methods like cosine similarity or Euclidean distance.

Table 6: Effectiveness of Similarity Metrics in Detecting Semantic Opposition

Sentence pair	similarity	Cosine Similarity	Euclidean Distance
The people are leaving the airplane. The people are entering the plane.	0	TN	FP

Table 6 shows a sentence pair with antonymic or opposite meaning where lexical overlap is high, but the semantic meaning is reversed. Jaccard similarity is not compared here as the sentence pair has high token overlap. Cosine similarity tends to perform better than Euclidean distance for sentence pairs with opposing meaning because it focuses on direction rather than the magnitude of the vectors. Even when two sentences share most of their words, replacing a single term with its antonym causes the overall semantic direction of the sentence embedding to shift. Cosine similarity managed to capture this shift, resulting in a lower similarity score and correctly classified a true negative. Conversely, Euclidean distance measures magnitudes and only a slight directional difference; the Euclidean distance may remain relatively small and result in a high similarity score. This shows that cosine similarity is generally more effective at detecting semantic opposition, while Euclidean distance can be less sensitive to subtle directional changes in meaning.

Table 7: Comparison of Similarity Metrics in Handling Paraphrased Sentence

Sentence pair	similarity	Cosine Similarity	Euclidean Distance
One man is breaking cement on another man's chest. A man breaks cinder blocks on another man.	0	FN	TP

Euclidean distance measures the absolute closeness of the two embedding vectors. Since the sentences are describing the same event (a man is breaking something on another man), their embeddings lie very close to each other in the Euclidean space. The slight difference in wording ("cement" vs "cinder blocks") does not shift the embeddings far apart in magnitude, so the Euclidean score passes the threshold to be a true positive. Cosine similarity focuses on direction rather than the absolute magnitude. Minor wording differences can change the direction of the vector embeddings. Since the optimal threshold obtained at 0.85 previously is

strict, this may lower the cosine similarity just enough to incorrectly classify the pair as false negative even though they are semantically similar. In a nutshell, the Euclidean distance method is advantageous in detecting semantically similar but lexically varied sentences because it measures the closeness in absolute space. Cosine similarity may miss such cases if the directions of the embeddings differ slightly, and Jaccard is the weakest because it cannot capture semantic equivalence at all.

3.2 Image Dataset

The dataset named "`image_pairs.xlsx`" is used in this experiment. A sample of the first 10 rows from the dataset is shown below in Table 8.

Table 8: Sample of Dataset "image_pairs.xlsx"

Image 1	Image 2	Similarity
3730011219_588cdc7972.jpg	3730011701_5352e02286.jpg	1
3710520638_866d542a80.jpg	3711611500_ea47b58b6f.jpg	1
3504940491_94c43792ed.jpg	3504275465_604ce2ef34.jpg	0
3517362674_0f5296de19.jpg	3518126579_e70e0cbb2b.jpg	0
3717809376_f97611ab84.jpg	3670918456_68631d362a.jpg	1

The "Image 1" and "Image 2" columns list the image pairs to compare, while the "Similarity" represents the similarity score between image 1 and image 2, which is based on human-annotated evaluation of whether the image pairs are similar. This binary similarity score serves as a key reference for evaluating the effectiveness of the implemented similarity search models, namely the Cosine similarity, Jaccard similarity, and Euclidean distance. By comparing the method's binary score classification against the binary similarity scores listed in the "Similarity" column in the dataset, the performance of the approaches can be assessed more objectively.

3.2.1 Cosine Similarity

For cosine similarity, the image is first resized to 224 × 224 pixels. Each pixel is represented by its RGB (Red, Green, Blue) components, where each channel has a value ranging from 0 to 255. A higher value in a channel indicates a stronger contribution of that colour to the final pixel. For example, a pixel with an RGB value of [200, 150, 100] appears orange in tone.

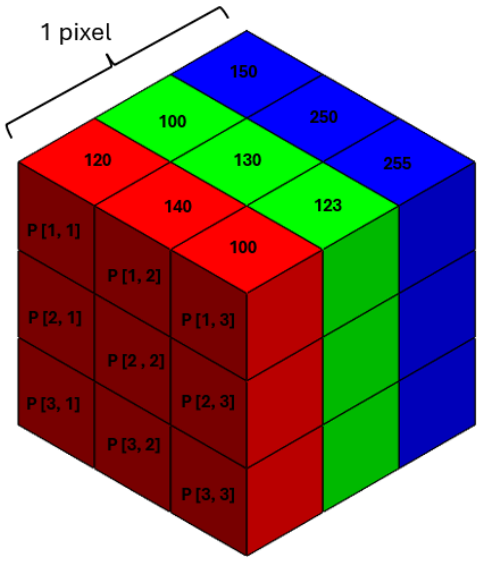


Figure 7: Visualisation of Pixels and RGB Value Tensor

Each pixel's RGB values are then scaled by dividing them by 255 to convert the range from [0, 255] to [0, 1]. Subsequently, the mean and standard deviation of each color channel across all pixels are computed. The image is then normalised using these statistics according to

$$Normalization = \frac{Channel\ Pixel\ Value - Mean\ of\ the\ Channel}{Standard\ Deviation\ of\ the\ Channel}$$

to ensure that all channels are balanced, preventing any color from being disproportionately bright or dark on average. This results in a 224 x 224 pixel image, where each pixel contains a normalised RGB vector.

The 224 x 224 x 3 tensor is subsequently passed through multiple convolutional layers to extract hierarchical features from the image. In the initial convolutional layers, the network

identifies low-level features such as vertical and horizontal edges and corners. In the intermediate layers, these simple features are combined to form more complex patterns, including contours and basic shapes. As the network progresses to deeper layers, it captures high-level features such as object parts and textures. Ultimately, in the final layer, these learned representations are aggregated to recognise and classify the object as a whole.

After each convolutional layer, the network produces a tensor of dimensions p x q x r, where p and q represent the spatial dimensions (row and columns of the feature map), and r denotes the number of features detected in that layer. For instance, after the first convolutional layer, a 224 x 224 x 64 tensor is generated, corresponding to 64 low-level feature maps extracted from the input image. Subsequently, a Max Pooling layer is applied to reduce the spatial dimensions of the feature maps. This operation selects the maximum value within a small patch of region of the feature map, effectively compressing the data while retaining the more significant features. As a result, the spatial dimensions are reduced from 224 x 224 to 112 x 112, decreasing computational complexity while preserving essential visual information. After all the convolutional layers, the network produces a 1 x 1 x 2048 tensor, representing the final feature vector of the image. Each of the 2048 values corresponds to the activation strength of a specific high-level feature learned by the model. A higher activation value indicates that the corresponding feature is more significant in the image.

For each image pair in the dataset, cosine similarity is computed between their respective feature vectors based on the formula expressed in Section 2.1. The resulting similarity score ranges from -1 to 1, which is normalised to a range of [0,1], where values closer to 1 indicate higher similarity. A threshold value is then applied to classify the image pair, i.e., if the cosine similarity score is higher than the defined threshold, the two images are considered similar, and a binary score of 1 is assigned. In contrast, if the cosine similarity score is lower than the defined threshold, the two images are considered dissimilar, and a binary score of 0 is assigned.

To determine the appropriate threshold value, a parametric study is performed by testing different threshold values ranging from 0.00 to 1.00 in increments of 0.05. For each threshold, cosine similarity is recomputed across the image pairs, and the resulting binary scores are compared with the ground truth binary score (“Similarity” column) provided in the dataset. The percentage of matching results, representing the accuracy, was computed for each threshold. The threshold that produces the highest accuracy is selected as the optimal value for

subsequent analysis. From Figure 8, the optimal cosine similarity threshold value is 0.80 with an accuracy of 94%.

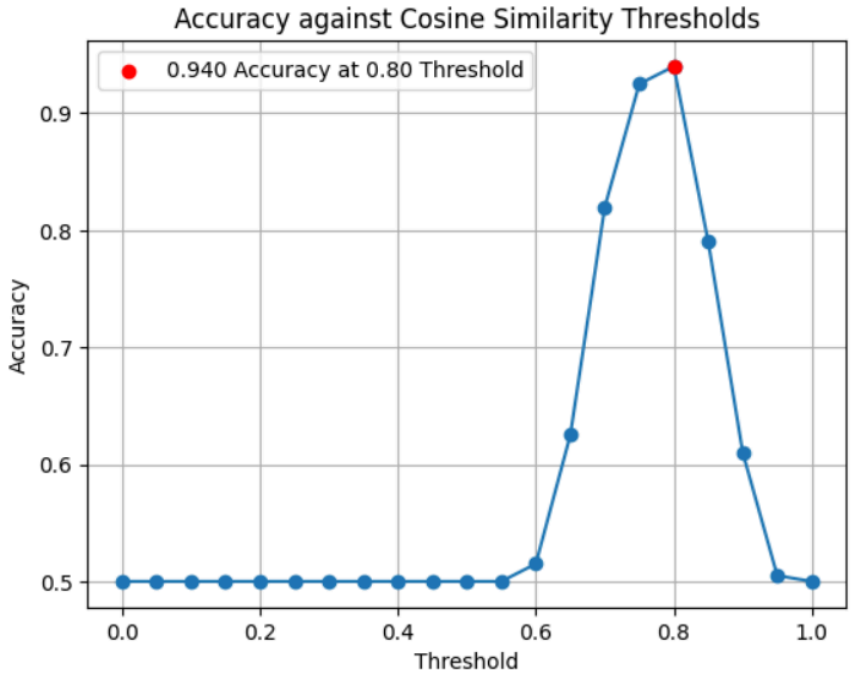


Figure 8: Accuracy for Different Cosine Similarity Threshold

3.2.1 Jaccard Similarity

In Jaccard similarity, the RGB color space is discretised into a fixed grid with 8 bins per channel. Each bin represents a specific range of intensity values within the 0 - 255 interval for each RGB component. The corresponding value range for each bin is shown in Table 9. As each of the three channels (R, G and B) contains 8 bins, the resulting colour space consists of $8^3 = 512$ unique colour tokens.

Table 9: Range of Values for each Bin

Bin	Range of Values
0	0 - 31
1	32 - 63
2	64 - 95
3	96 - 127
4	128 - 159
5	160 - 191
6	192 - 223
7	224 - 255

Each pixel is assigned to one bin within each color channel, i.e. r_bin for the red channel, g_bin for the green channel, and b_bin for the blue channel. The pixel can thus be represented by its corresponding bin vector [r_bin, g_bin, b_bin], which is one of the 512 colour tokens. The bin index for each pixel is then computed using the following expression:

$$\text{Bin Index} = r\_bin \times 8^2 + g\_bin \times 8 + b\_bin$$

Each pixel's corresponding colour bin index is then collected into a set of unique tokens, forming the image's feature vector, which serves as the vector to perform Jaccard Similarity based on the formula expressed in Section 2.2. For instance, a pixel with the RGB value of (10, 240, 70):

RGB value	Bin Number	Bin Index
R = 10 (0 - 31)	0	0*64 + 7*8 + 2 = 58
G = 240 (224 - 255)	7	
B = 70 (64 - 95)	2	

To determine the appropriate threshold value, a parametric study is performed by testing different threshold values ranging from 0.00 to 1.00 in increments of 0.05. For each threshold, Jaccard similarity is recomputed across the image pairs, and the resulting binary scores are compared with the ground truth binary score (“Similarity column) provided in the dataset. The percentage of matching results, representing the accuracy, was computed for each threshold. The threshold that produces the highest accuracy is selected as the optimal value for subsequent analysis. From Figure 9, the optimal cosine similarity threshold value is 0.60 with an accuracy of 81%.

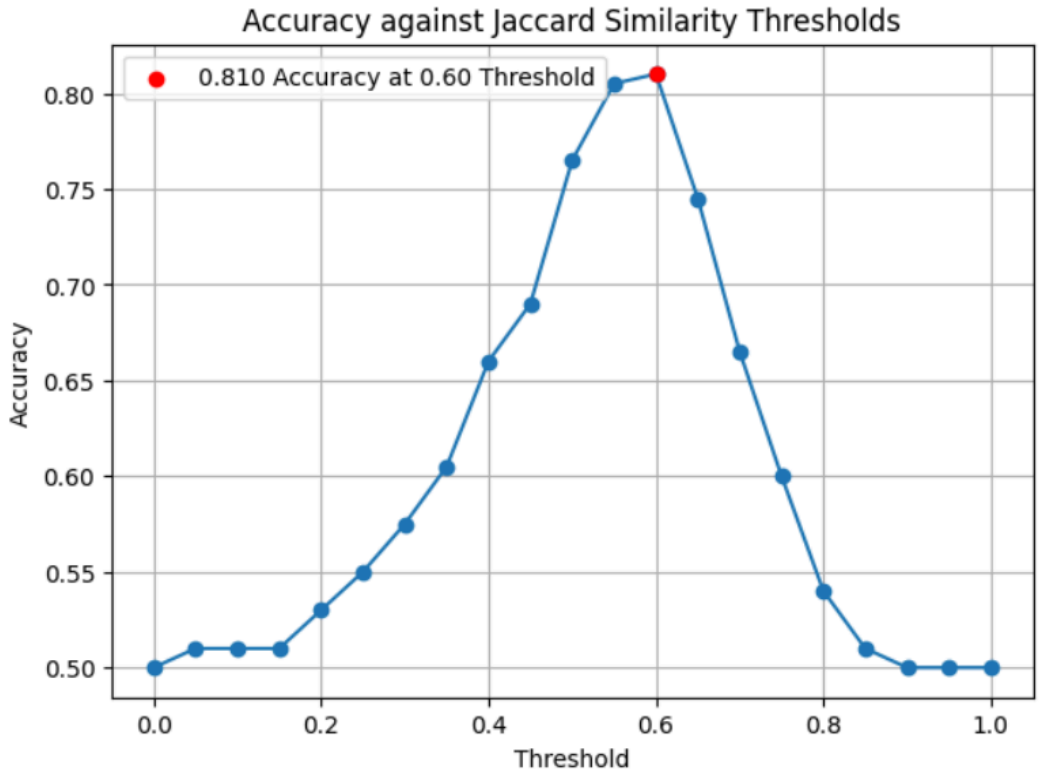


Figure 9: Accuracy of Different Jaccard Similarity Threshold

3.2.1 Euclidean Distance

For the Euclidean distance method, the process of feature extraction and tokenisation follows the same approach as used for cosine similarity, resulting in a 1 × 1 × 2048 feature vector for each image. The Euclidean distance between two images is computed using the mathematical expression described in Section 2.3, which measures the absolute difference between their corresponding feature vectors. A smaller Euclidean distance indicates a higher degree of similarity between the two images.

To provide a more intuitive interpretation, the Euclidean distance values are further transformed into Euclidean similarity scores through normalisation. This conversion ensures that higher similarity values correspond to more similar image pairs. Moreover, expressing the results as Euclidean similarities facilitates parameter tuning and threshold selection during subsequent parametric analyses.

The determination of the optimal threshold for achieving the highest accuracy was conducted using the same analytical approach as applied for cosine similarity and Jaccard similarity. Based on this analysis, the optimal threshold value was identified as 0.50, corresponding to an overall accuracy of 91.5%.

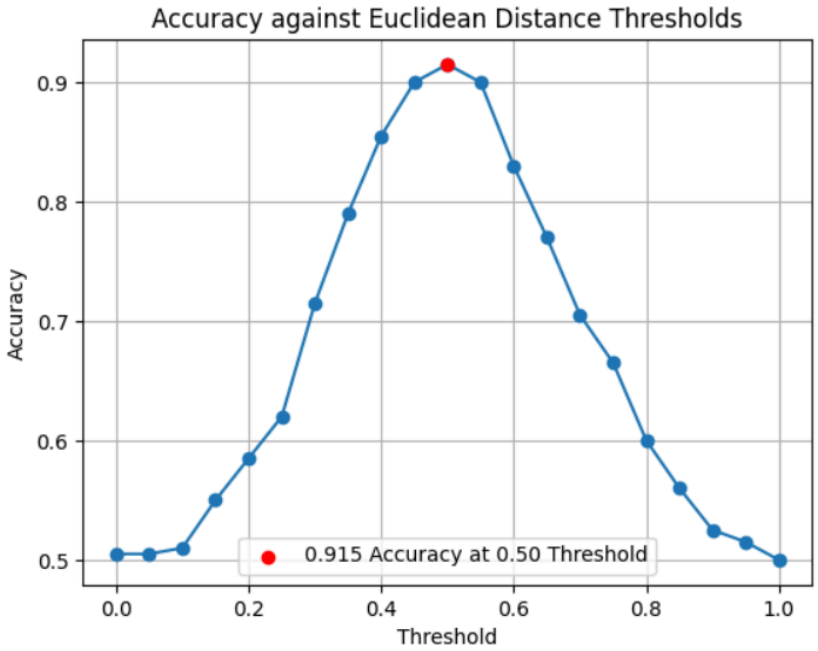


Figure 10: Accuracy of Different Euclidean Distance Thresholds

3.2.2 Detailed Analysis of the Similarity Search Methods

3.1.3.1 Confusion Matrix

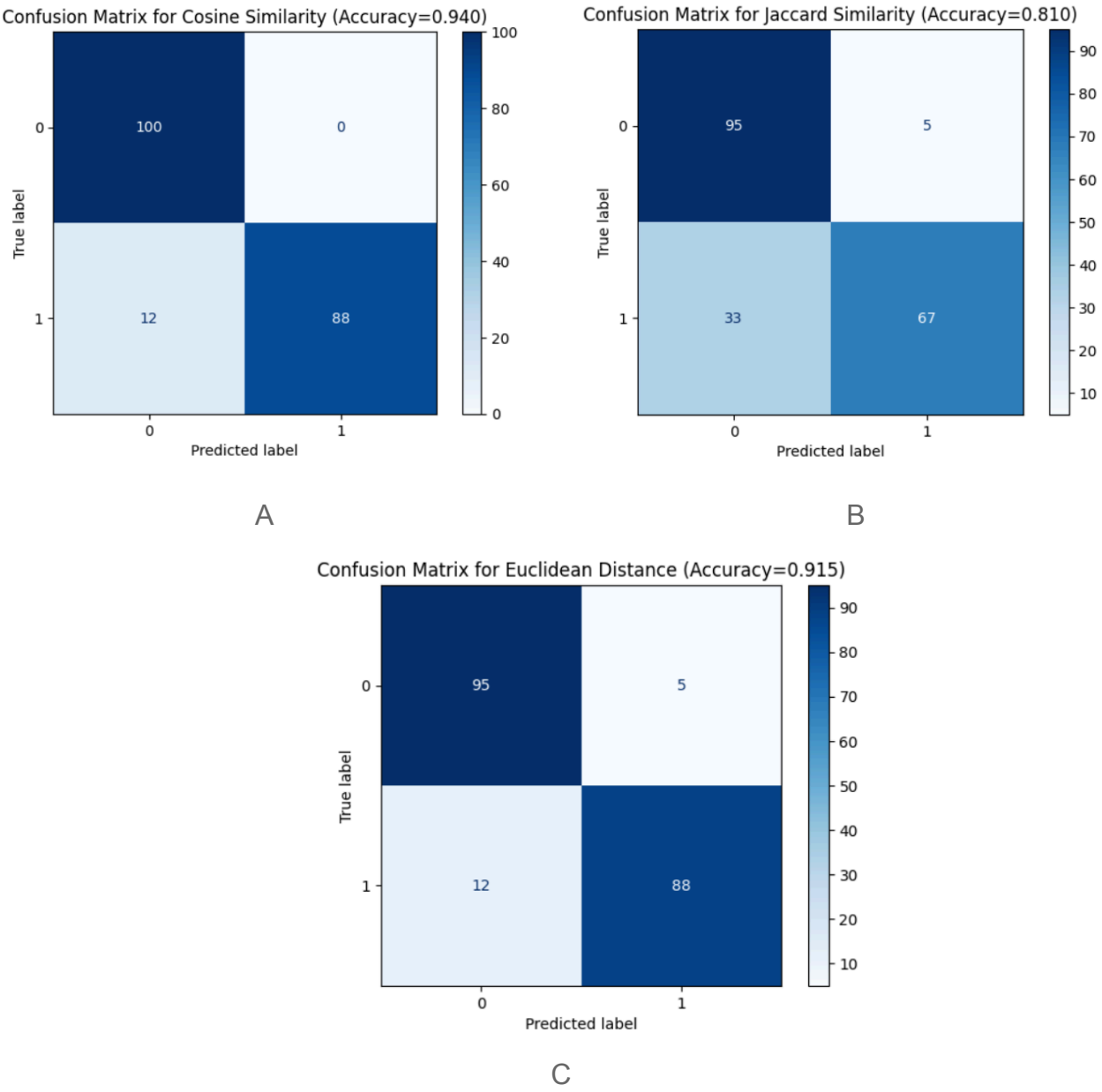


Figure 11: Confusion Matrix for (A) Cosine Similarity (B) Jaccard Similarity (C) Euclidean Distance

Figure 11 presents the confusion matrices for image-based similarity search using three methods: cosine similarity, Jaccard similarity, and Euclidean distance. From the results, the accuracy achieved by cosine similarity, Jaccard similarity, and Euclidean distance are 94%, 81%, and 91.5%, respectively. While cosine similarity and Euclidean distance exhibit comparable accuracy, precision and recall can provide further insights. Cosine similarity attains 100% precision, indicating that all predicted similar image pairs are correctly identified, whereas the Euclidean distance method achieves 94.6% precision. Overall, cosine similarity demonstrates superior performance in terms of accuracy, precision, and recall, making it the most effective approach for image-based similarity search.

3.2.2.1 Cosine Similarity

Examples of false positive and false negative cases identified by the cosine similarity method are presented in Table 10. In this method, the RGB pixel values of each image are processed through a series of convolutional neural network layers, where features such as colour, edges, and shapes are detected and subsequently compressed into a 1 × 1 × 2048 feature vector, which corresponds to 2048 features being compared. Each component of the feature vector represents how strongly that visual feature appears in the image. In a vector space, the direction is determined by the ratio of its components. Hence, if the activation strength of a component in the feature vector is greater than others, the vector tilts more toward that particular feature. Therefore, the direction of the vector encodes the pattern of which feature dominates the image. A false negative occurs when the same objects appear in different colours, angles, or zoom levels, causing variations in the extracted feature representations. Conversely, a false positive arises when two distinct images share similar shapes and colour distributions, leading to misclassifications.

Table 10: False Negative Cases Predicted by Cosine Similarity Method

False Negative	
Image 1	**Image 2**
	
	

As illustrated in Table 10 (a), both images depict two dogs of the same object and quantity; however, differences in coat colour result in distinct feature vectors. Additionally, the images were captured from different perspectives. Image 1 from a side view and Image 2 from a near-frontal view, where the dog's facial features are not visible. For the second image pair,

Image 1 presents a near-frontal view, while Image 2 is taken from the side. In Image 1, the girl's legs are covered with long pants, whereas in Image 2, her legs are exposed. Conversely, the boy's hands are uncovered, while the girl's hands are concealed. Furthermore, the background shapes and colours of the two images differ significantly. Since the cosine similarity method employs global feature averaging to produce a 1 × 1 × 2048 feature vector for similarity computation, variations in background composition, viewing angle, and object orientation can cause the averaged feature vector to diverge in direction. Consequently, these differences lead to inconsistencies in similarity prediction despite apparent visual resemblance in object form.

3.2.2.2 Jaccard Similarity

The Jaccard similarity method is susceptible to both false positive and false negative outcomes, as it considers only the colour composition of the image without utilising deep neural network feature extraction. Additionally, the Jaccard similarity measure is sensitive to the proportion of colour present in an image. When the amount of colour occupying the image is small, the denominator in the Jaccard similarity calculation becomes large, resulting in a higher similarity value, and vice versa. A false positive occurs when two images share similar colour distributions despite depicting entirely different objects. Conversely, a false negative arises when minor variations in colour tone or lighting conditions cause two visually identical objects to be represented as dissimilar. Hence, the Jaccard similarity approach is more appropriate for identifying images with identical colour patterns or pixel-level similarities rather than those with the same object under varying visual appearances.

Table 11: False Negative Case Predicted by Jaccard Similarity

False Negative	
Image 1	**Image 2**
	

Table 11 presents two images depicting a child playing on a seesaw. Although both images feature the same object captured at a similar orientation, approximately at the peak of the seesaw's swing, the background colours differ significantly. In Image 1, the playground ground exhibits a yellowish-brown tone, whereas Image 2 features a sky-blue background. Due to this substantial variation in colour composition, the Jaccard similarity model classifies the two images as dissimilar.

Table 12: False Positive Case Predicted by Jaccard Similarity

False Positive	
Image 1	**Image 2**
	

In Table 12, Image 1 primarily depicts a fountain as its main subject, whereas Image 2 focuses on a dog jumping over a fence. Both images, however, contain large background areas dominated by the same green colour tone. Moreover, each image exhibits only two predominant colours: green, white and black in Image 1, and green and black in Image 2. As a result, the images utilise a limited number of colour bins, leading to significant overlap in their colour sets. Consequently, the Jaccard similarity measure becomes less reliable when the feature set is small, as it lacks the sensitivity to differentiate subtle content variations.

3.2.2.2 Euclidean Distance

Although the feature vector generation process is identical to that used in cosine similarity, where RGB values are compressed into a 1 x 1 x 2048 feature vector, the Euclidean distance evaluates similarity based on the magnitude difference between the vectors. When an image's overall intensity is higher, all activation values scale proportionally, increasing the length of the vector while maintaining its direction. From a visual interpretation standpoint, the vector's magnitude reflects the overall brightness or tonal intensity of the image colours. As a result, false negatives may occur when two identical images differ only slightly in brightness or

contrast, producing a noticeable difference in vector magnitude. Conversely, false positives can happen when two unrelated images share similar overall brightness levels, such as both being predominantly bright or dark, resulting in a small Euclidean distance due to comparable pixel intensities.

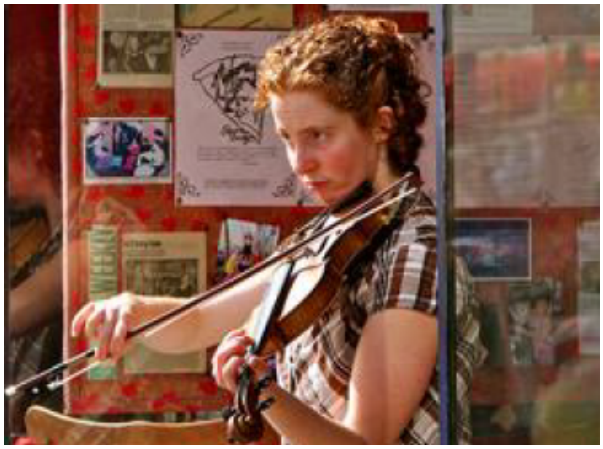
Table 13: False Negative Cases Predicted by Euclidean Distance

False Negative	
Image 1	Image 2
	
	

As shown in Table 13, Image 1 depicts a dog leaping over water, while Image 2 shows two men wearing dull colour shirts seated on a mat and a chair, respectively. Both image pairs share similar characteristics in terms of shape, viewing angle, and overall colour composition. However, in both cases, the second image was captured under a darker tone, whereas the first

image appears lighter. This difference in brightness intensity contributes to a greater Euclidean distance between the feature vectors.

Table 14: False Positive Cases for Euclidean Distance Method

False Positive	
Image 1	Image 2
	

According to Table 14, Image 1 depicts a child jumping from the water slides, whereas Image 2 shows a woman playing the violin. The two images are entirely unrelated. Notably, both images were captured under similar lighting conditions, exhibiting comparable brightness and tonal balance. As a result, the Euclidean distance between their feature vectors is relatively small, leading the Euclidean distance method to incorrectly classify the two images as similar despite them being totally unrelated.

3.2.3 Comparison between the Similarity Search Methods

Table 15: Cases of Semantic Overlap

Image Pair	
 	
Simialrity	1
Cosine Similarity	TP
Euclidean Similarity	TP
Jaccard Similarity	FN

Table 15 presents an example of a semantic overlap with a low colour equivalence case. Both images depict a man engaged in river rafting, wearing a safety vest and helmet, and holding a paddle at a similar angle while navigating through fast-flowing water. The colour differences between corresponding objects in both images are minimal, enabling the cosine similarity method to effectively capture edges and shapes. Consequently, the cosine similarity yields a true positive result.

In addition, both images share a similar overall colour tone, being captured under bright sunlight conditions. This results in small differences in pixel intensity, thereby producing a high similarity score under the Euclidean distance method. However, the kayak in Image 1 is blue, whereas in Image 2 it is red. Since the Jaccard similarity method measures the overlap of colour bins present in the images, this colour variation reduces the intersection between the two, resulting in a low Jaccard similarity score.

Therefore, this case demonstrates that cosine similarity and Euclidean distance are more effective than Jaccard similarity in identifying visual similarity when object shape and structural features are involved. This is because the feature vectors used in cosine and Euclidean

methods are derived through deep convolutional neural networks, which capture high-level semantic features beyond mere colour presence.

Table 16: Comparison between Cosine Similarity & Euclidean Similarity

Image Pair	
	
Similarity	1
Cosine Similarity	TP
Euclidean Similarity	FN

The preprocessing steps for both the cosine similarity and Euclidean distance methods are identical, where each image is converted into a 1 × 1 × 2048 feature vector. In the cosine similarity method, the angular distance between the two normalised vectors is computed, whereas in the Euclidean distance method, the L2 norm between the two feature vectors is calculated.

As illustrated in the image pair showing a dog jumping over water, both images capture the same object from an identical angle. The cosine similarity method correctly identifies the two images as representing the same object, while the Euclidean distance method produces an incorrect classification. This occurs because cosine similarity is sensitive to the directional alignment of feature vectors, which reflects the spatial distribution of features such as edges, shapes, and colours, while being less affected by vector magnitude due to normalisation. In contrast, the Euclidean distance method is more sensitive to vector magnitude, which corresponds to colour intensity and tonal variations, but less responsive to the spatial arrangement of features within the image.

From the comparison, it can be concluded that the cosine similarity method is effective in identifying images with similar edges, corners, shapes, and overall object structures. In contrast, the Euclidean distance method is more suitable for detecting variations in tone or colour intensity among images that have already been recognised as similar in structure by the cosine similarity method. The Jaccard similarity method, on the other hand, primarily captures differences in colour composition between images and therefore demonstrates the weakest performance among the three approaches in image similarity detection.

4 Conclusion

The evaluation of the three similarity search methods, cosine similarity, Euclidean distance and Jaccard similarity, demonstrates clear differences in their strengths and limitations. In the context of text similarity, cosine similarity achieves the best balance between precision and recall, followed by Euclidean distance. Jaccard similarity underperforms across all metrics due to its reliance on exact token overlap in the “STSbenchmark.xlsx” dataset, which prevents it from capturing semantic meaning, synonyms or paraphrases.

The analysis of sentence pairs reinforces these findings. Cosine similarity is more effective at detecting semantic opposition and nuanced meaning differences. It is suitable for tasks requiring precise semantic understanding. Euclidean distance is excellent at identifying paraphrased or lexically varied but semantically similar sentences. Jaccard similarity remains useful for fast, surface-level lexical comparisons such as detecting duplicates or near-duplicates. The choice of similarity search method should therefore depend on the characteristics of the dataset and the type of text relationships being evaluated. A summary of the strengths and weaknesses of each method is presented in Table 17.

Table 17. Strengths and Weaknesses of Cosine Similarity, Euclidean Distance and Jaccard Similarity in Text-based Similarity Search

Similarity Search Method	Strengths	Weaknesses	Suitable Application
Cosine Similarity	Distinguishes semantically opposing or related meanings when embeddings capture meaningful semantic directions. Effective for measuring semantic similarity regardless of sentence length.	Ignores absolute magnitude; sentences with different word counts may yield similar scores. Highly dependent on the quality of vector representations.	Sentences that possess opposite meaning (high lexical overlap but semantic contrast)
Euclidean Distance	Captures magnitude differences and performs well with	Sensitive to scale and magnitude, not great for sparse text.	Sentences that are paraphrased (different wording but similar

	dense embeddings. Useful for measuring absolute differences between feature vectors.	Requires normalisation to be meaningful in many NLP tasks.	meaning)
Jaccard Similarity	Simple and fast for detecting exact lexical overlap or identical words. Efficient for surface-level comparisons and duplicate detection	Ignores frequency and meaning, sensitive to wording differences. Performs poorly on paraphrased or semantically equivalent text.	Lexical overlap or duplicate detection tasks, such as identifying near-duplicates or exact matches.

Similar to the findings in text similarity search, cosine similarity achieves the highest accuracy in identifying similar images. This is because cosine similarity measures the direction of the feature vectors extracted from images, which effectively captures the spatial distribution of visual features such as edges, shapes, and contours. Since it does not account for vector magnitude, cosine similarity is less sensitive to variations in brightness or colour intensity, allowing it to focus primarily on structural and object-level similarity.

In contrast, the Euclidean distance method computes the magnitude difference between feature vectors, making it more responsive to changes in tone, brightness, and colour intensity. It performs well when comparing images that share similar object structures but differ slightly in illumination or exposure levels.

The Jaccard similarity method exhibits the weakest performance in image similarity search as it relies solely on colour composition and does not utilise deep convolutional neural network feature extraction. While it can detect colour variations, it fails to recognise identical objects under different lighting conditions or colour tones, limiting its applicability to low-level, pixel-based comparisons.

Table 18. Strengths and Weaknesses of Cosine Similarity, Euclidean Distance and Jaccard Similarity in Image-based Similarity Search

Similarity Search Method	Nature of Search Method	Strengths	Weaknesses	Suitable Application
Cosine Similarity	Computes the direction of feature vectors encoded from images.	Captures the spatial distribution of visual features such as edges, shapes, and contours.	Insensitive to brightness and colour intensity variations due to normalisation.	Identifying object-level and structural similarity in images.
Euclidean Distance	Measures the magnitude difference between feature vectors.	Sensitive in detecting differences in tone, brightness and colour intensity.	Less effective for distinguishing structural similarity.	Comparing images with similar content but different lighting or exposure conditions.
Jaccard Similarity	Determines the colour composition overlap between images.	Effective in detecting colour variations and simple pixel-level similarities.	Fails to identify structural or semantic similarity	Performing basic pixel-level similarity checks or colour-based image comparisons.

References

[1] Mining, W. I. D. (2006). Introduction to data mining. *Mining Multimedia Databases, Mining Time Series and*.

[2] Januzaj, Y., & Luma, A. (2022). Cosine similarity–a computing approach to match similarity between higher education programs and job market demands based on maximum number of common words. *International Journal of Emerging Technologies in Learning (iJET)*, *17*(12), 258-268.

[3] Genc, Z., Babieva, N. S., Zarembo, G. V., Lobanova, E. V., & Malakhova, V. Y. (2021). The views of special education department students on the use of assistive technologies in special education. *International Journal of Emerging Technologies in Learning*, *16*(19).

Appendix

The complete source code, datasets, classification results, and experimental notebooks supporting this study are available in the GitHub repository below:

GitHub Repository: https://github.com/YapHS0514/SC4020-Project-1/tree/main

This repository includes:

├── data/

├── results/

├── .gitignore

├── image_similarity.ipynb

├── requirements.txt

├── text_similarity.ipynb