**QUESTION-3**

**Comparing Distances Between Paths in a 2D Grid**

To compare distances between any two different paths on a 2D grid, a suitable metric would be the **Hausdorff distance**. This metric measures the greatest of all the distances from a point in one path to the closest point in the other path. The Hausdorff distance is particularly effective for comparing paths because it accounts for the overall shape and distribution of points along the paths, rather than just comparing endpoints or aggregate distances.

**Why Hausdorff Distance is Better**

1. **Shape Sensitivity**: Hausdorff distance considers the overall shape of the paths, making it more sensitive to differences in the paths' trajectories.
2. **Maximal Deviation**: It measures the maximal deviation between two paths, providing a robust measure of dissimilarity.
3. **Path Distribution**: Unlike simple distance measures like Euclidean or Manhattan distance that might focus on endpoints or average distances, Hausdorff considers the distribution of points along the paths.

**Other Metrics**

1. **Euclidean Distance**: Measures the straight-line distance between corresponding points but doesn't capture the shape of the paths.
2. **Manhattan Distance**: Sum of the absolute differences of their coordinates. Useful for grid-based paths but still lacks shape sensitivity.
3. **Frechet Distance**: Considers the locations and ordering of points along the paths, often used in comparing curves. More computationally intensive than Hausdorff.
4. **Dynamic Time Warping (DTW)**: Aligns paths that may vary in speed. Useful in time series but less common for static path comparisons.

**Word Embeddings and Likeliness**

When transforming words into numerical vectors in high-dimensional space, the common criterion used to define the ‘likeliness’ between any two words is **cosine similarity**.

**Why Cosine Similarity?**

1. **Direction Over Magnitude**: Cosine similarity measures the cosine of the angle between two vectors, focusing on their direction rather than magnitude. This is crucial in high-dimensional spaces where Euclidean distances can be misleading.
2. **Normalized Measure**: It normalizes the dot product of the vectors by the product of their magnitudes, providing a measure that is independent of the vector length.

**Python Implementation: Embeddings and Visualization**

I have generated a corpus of words, convert them into embeddings using a pre-trained model, compute cosine similarities, and plot the embeddings in 3D space using Plotly.

