Q3] **Math behind AI-ML**
A suitable metric to compare distances between any two different paths on a 2D grid would be the Frechet distance. Frechet distance takes into account the location and ordering of the points along the paths, which is why it is particularly well-suited for comparing curves or paths.
Some benefits of using Frechet distance-
1] Unlike simpler metrics like Euclidean distance, the Frechet distance considers the shape of the paths and the order in which points are traversed. This makes it more sensitive to the actual paths taken rather than just the endpoints.
2] Many real-world paths are not straight lines, and the Frechet distance can handle curves and bends more effectively.
3] Small deviations along the paths do not drastically change the distance measurement, making it a robust metric for practical applications.

Other Metrics:

1] Euclidean Distance

2] Manhattan Distance

3] Dynamic Time Warping (DTW)

4] Hausdorff Distance

Common criterion used to define 'likeliness' between any two words mathematically is the Cosine similarity because it-

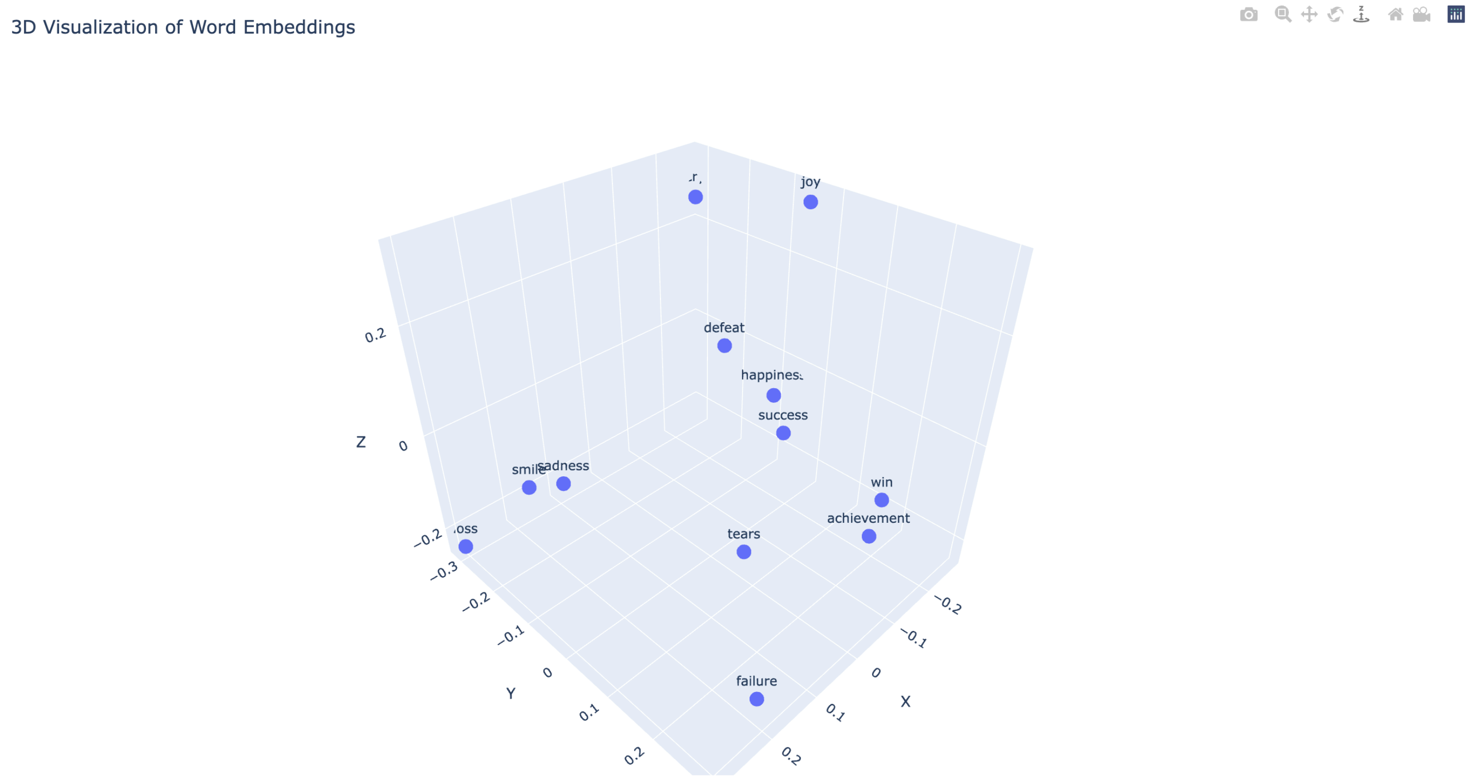
- measures the cosine of the angle between two vectors, making it invariant to the magnitude of the vectors. This is particularly useful for embeddings where the direction of the vector is more important than its length.
- effectively captures the semantic similarity between word vectors in high-dimensional space.
- is computationally efficient and widely supported in machine learning libraries.

Code for converting corpus of words into embeddings:

```
import numpy as np

import plotly.express as px

from gensim.models import Word2Vec


# Sample corpus

data = [["happiness", "joy", "smile"],

        ["sadness", "tears", "cry"],

        ["success", "achievement", "win"],

        ["failure", "loss", "defeat"]]
```

```python
# Train Word2Vec model
model = Word2Vec(sentences=data, vector_size=3, window=2,
min_count=1, sg=0)

# Extract word vectors and labels
words = list(model.wv.index_to_key)
word_vectors = np.array([model.wv[word] for word in words])
print(word_vectors)

# Create a DataFrame for Plotly
import pandas as pd
df = pd.DataFrame(word_vectors, columns=['x', 'y', 'z'])
df['word'] = words

# Create a 3D scatter plot
fig = px.scatter_3d(df, x='x', y='y', z='z', text='word')

# Customize the layout
fig.update_layout(
    title='3D Visualization of Word Embeddings',
    scene=dict(
        xaxis_title='X',
        yaxis_title='Y',
        zaxis_title='Z'
    )
)

# Show the plot
fig.show()
```

3D plot of embeddings of words


3D Visualization of Word Embeddings
joy
defeat
happiness
success
smile
sadness
oss
win
achievement
tears
failure
Z
Y
X