Experiments

1. Input graph:
Nodes: 14, Edges: 29, Communities: 2, Diameter: 6
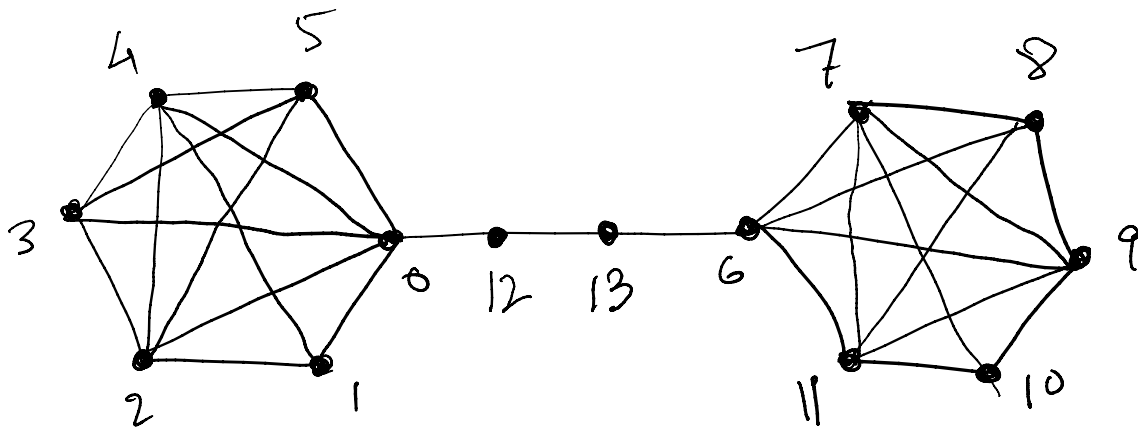
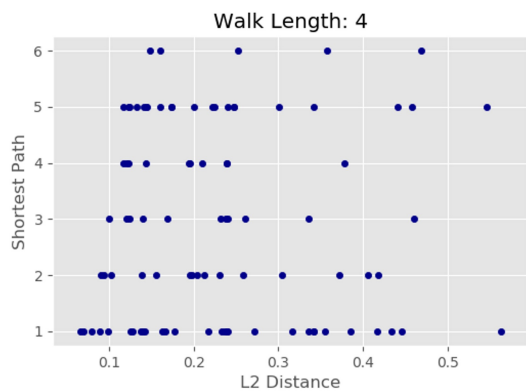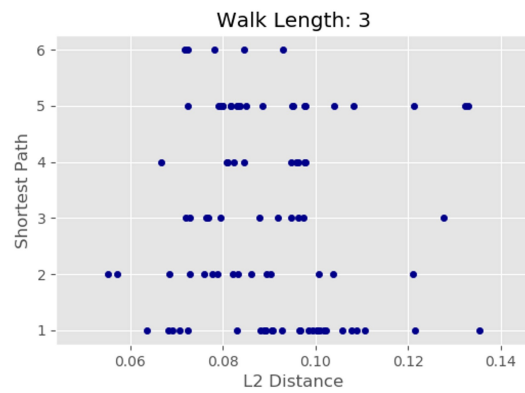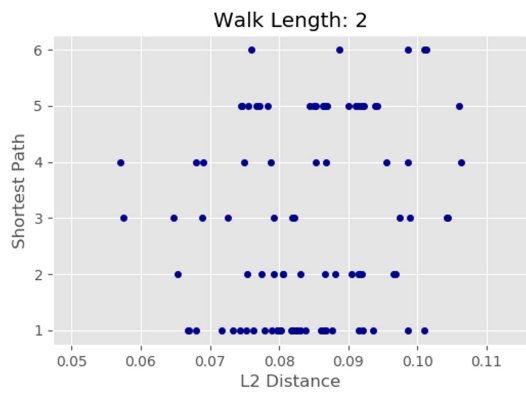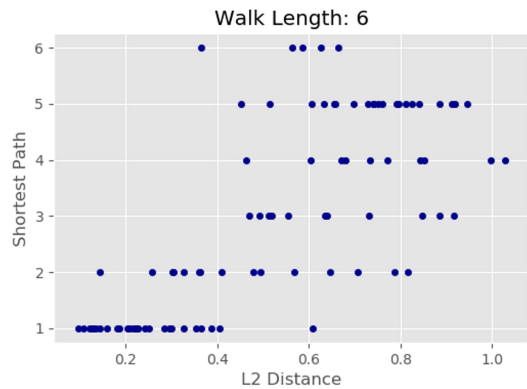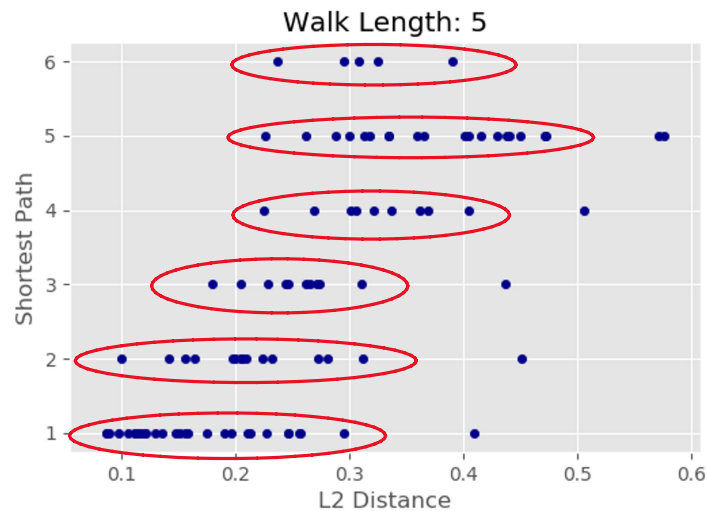

Experiment 1.1: Check effect of Node2vec random walker's length of walk on quality of embedding:
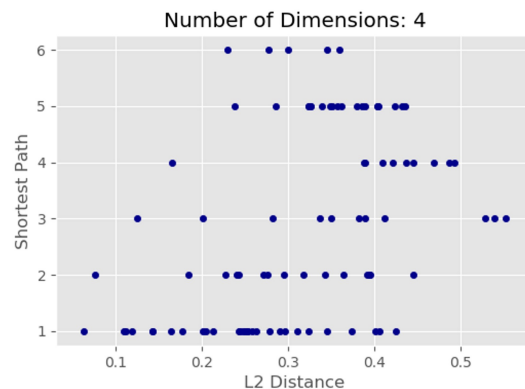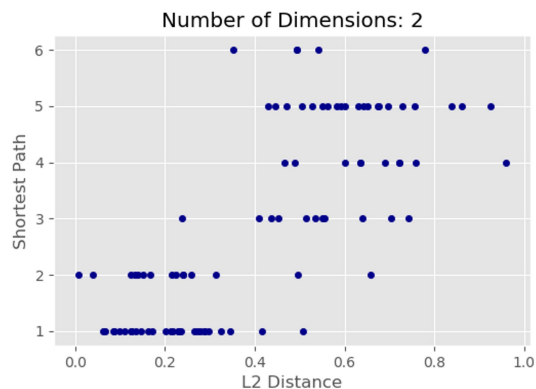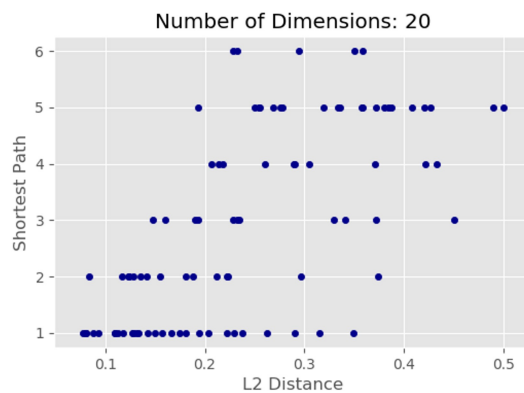
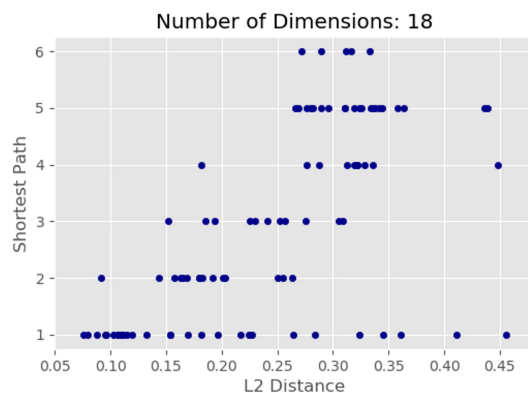Result: Effect of length of walk (dimensions = 24):
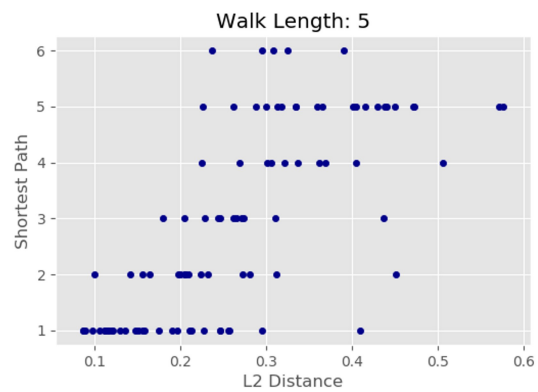
Walk Length: 6

Observation: The quality of embeddings improve as the length of walk is increased till it is the same as the graph diameter. The walk length of 5 gives embeddings good enough to clearly see the adjacent node pairs having shortest distance of 1 having the lowest L2 distances. This pattern still holds for node pairs 2,3,... Hops away from each other.

Walk Length: 5

Experiment 1.2: Check effect of Node2vec's input parameter 'd' (dimension of embedding) on the quality of embedding.

Result: Effect of #dimension (length of walk = 5):

Number of Dimensions: 2

Number of Dimensions: 4

Number of Dimensions: 6

Number of Dimensions: 8

Number of Dimensions: 10

Number of Dimensions: 12

Number of Dimensions: 14

Number of Dimensions: 16

Number of Dimensions: 18

Number of Dimensions: 20

Number of Dimensions: 22 | Number of Dimensions: 24
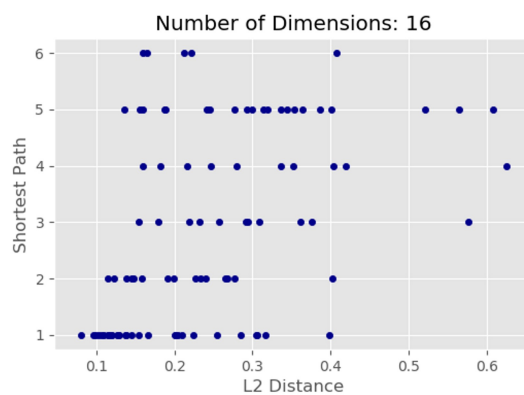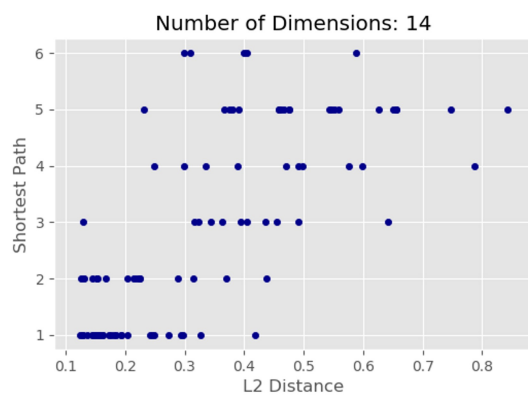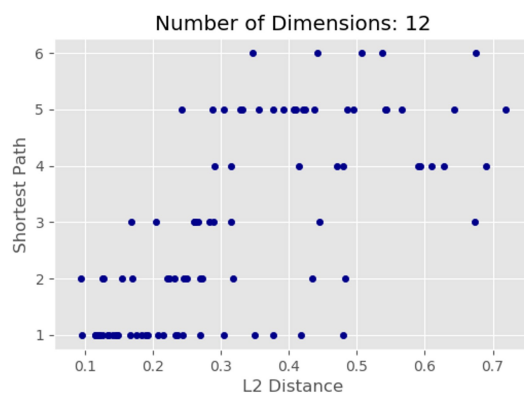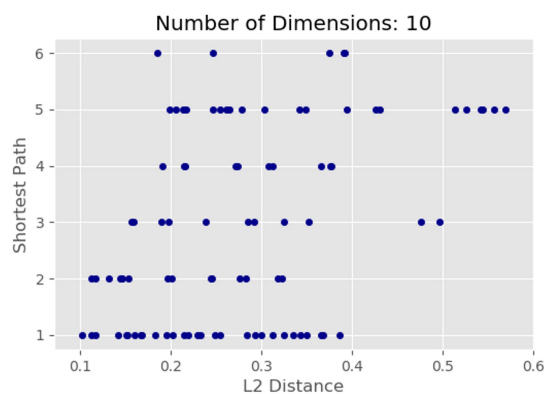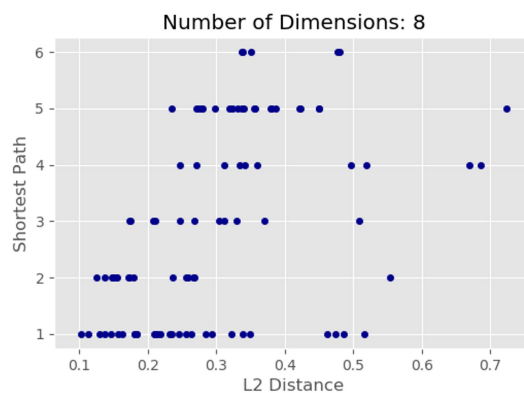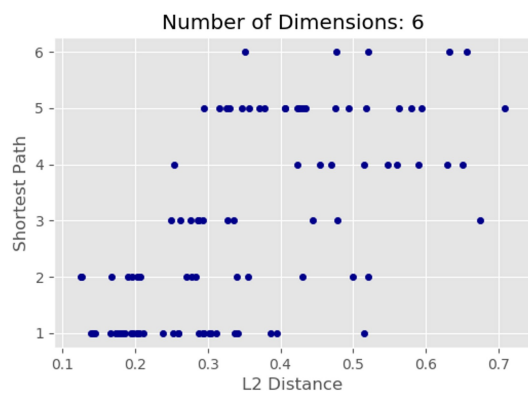
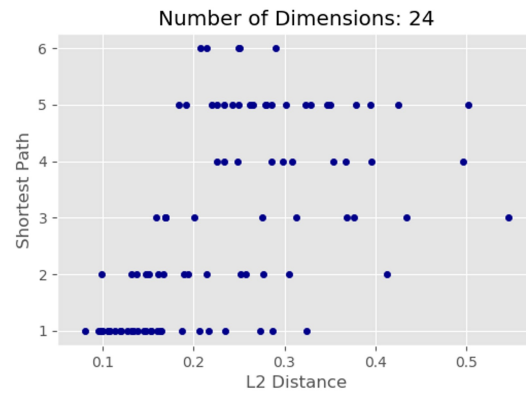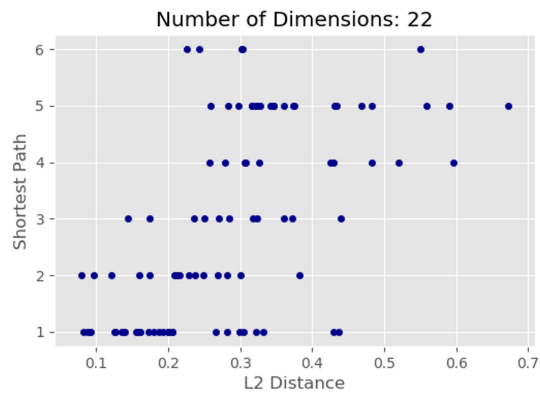Observation: Maybe I am missing something but I couldn't find any pattern which might indicate a negative or a positive effect on the quality of embedding by increasing the dimensionality of the embedding.

2. Input Graph: Twitter dataset (taken from Snap)

Dataset statistics	
Nodes	81306
Edges	1768149
Nodes in largest WCC	81306 (1.000)
Edges in largest WCC	1768149 (1.000)
Nodes in largest SCC	68413 (0.841)
Edges in largest SCC	1685163 (0.953)
Average clustering coefficient	0.5653
Number of triangles	13082506
Fraction of closed triangles	0.06415
Diameter (longest shortest path)	7
90-percentile effective diameter	4.5

Experiment: Check Node2vec's scalability by using it on graphs of different size (obtained by snowballing the Twitter graph)

Scaling of Node2vec for different sized input graph
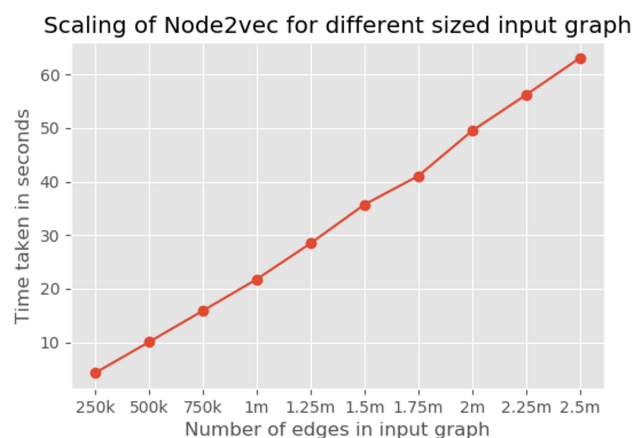
Observation: Node2vec scales linearly with the increasing number of edges.