HW3 REPORT

Author: Jiayi Liu (9749111299)

This is an individual work. The project structure is as follows.

```
# FastMap Implementation
fastmap.py
# PCA Implementation
pca.py
```

IMPLEMENTATION DETAIL

1. PCA

To rerun the code, just `python main.py`. The matrix operations in this code are implemented with the help of NumPy library. The first two principle components are as follows.

```
[[ 0.86667137 -0.4962773 ]
 [-0.23276482 -0.4924792 ]
 [ 0.44124968  0.71496368]]
```

2. FastMap

To rerun the code, just `python fastmap.py`. If you want to see the visualization, please uncomment the code at `line 2, 14, 66-73`, and make sure `matplotlib` installed.

2.1 Initailization

All the implementation is encapsulated in the class `FastMap`.

```
class FastMap:
    def __init__(self, info, k):
        self.wordlist = info["wordlist"]
        self.N = len(self.wordlist)
        self.dis = info["distance"]
        self.k = k
        self.coord = np.zeros((self.N, self.k))
```

We take `info` read from the .txt file and `k` as the input from user.

- `self.dis` : the distance matrix which is a 2D symmetric N x N matrix storing the distance between objects. This is a numpy array.
- `self.coord` : the result matrix which is a 2D N x k matrix storing the k-dimention coordinates of each objects. This is also a numpy array.

2.2 Core Algorithm

The FastMap algorithm is a recursive process. The function `fastmap` is the recursive function itself. The number of iteration is determined by `k`. So we need to track of both the new distance matrix and k.

```
def fastmap(self, k, dis):
        # Termination
        if k <= 0:
            return

        # The column of coordinate matrix to be updated
        col = self.k - k

        # New distance matrix
        new_dis = np.zeros_like(dis)

        # Find the farthest pair of objects
        a, b = self.choose_distant_objects(dis)
        if dis[a, b] == 0:
            self.coord[:, col] = 0
            return

        # Update coordinates of a and b
        self.coord[a, col] = 0
        self.coord[b, col] = dis[a, b]

        # Update coordinates of other objects
        for i in range(self.N):
            if i != a and i != b:
                self.coord[i, col] = self.compute_xi(dis[a, b], dis[a, i], dis[b,
i])
```

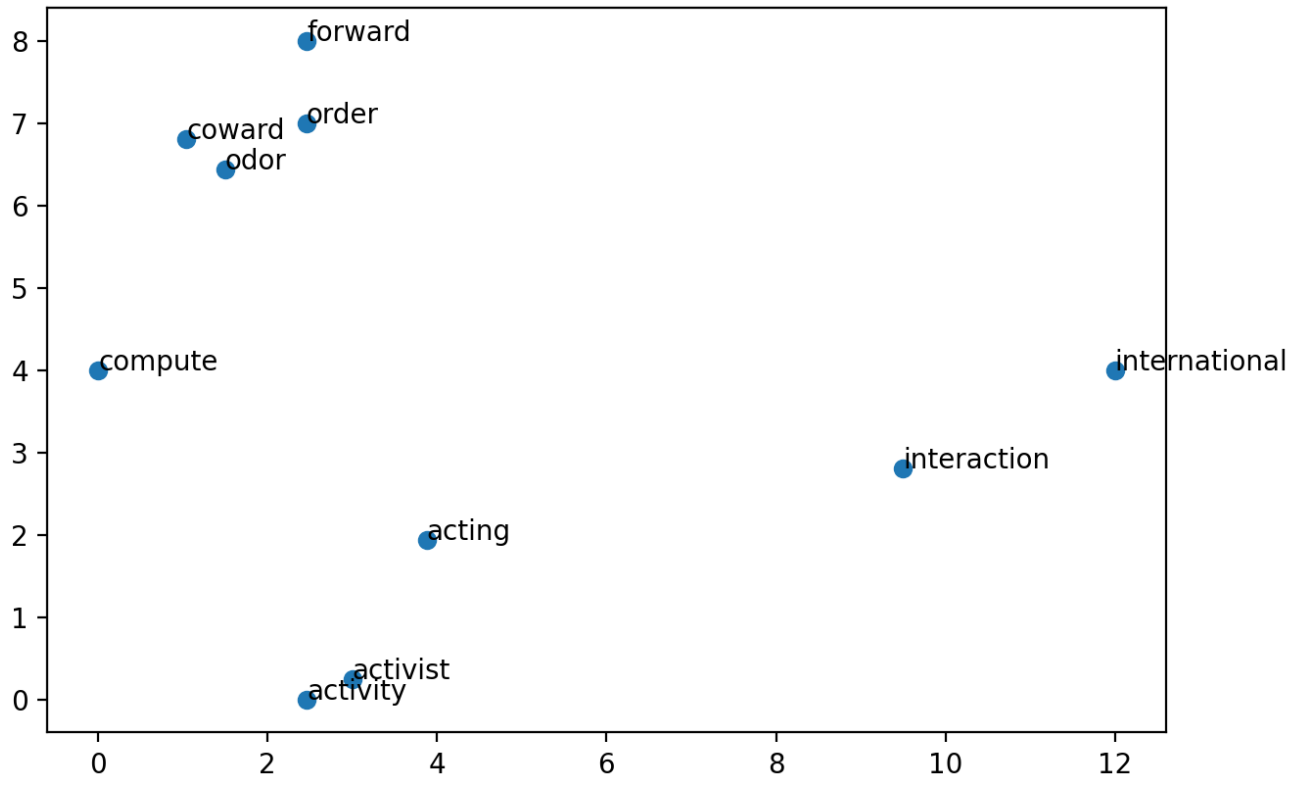
```
        # Update distance matrix
        for i in range(self.N):
            for j in range(i + 1, self.N):
                new_dis[i, j] = np.sqrt(dis[i, j] ** 2 - (self.coord[i, col] -
self.coord[j, col]) ** 2)
                new_dis[j, i] = new_dis[i, j]

        # Recursion
        self.fastmap(k - 1, new_dis)
```

2.3 Result

At the end of the recursion, we can see the coordinates of objects printed out as follow. The first and second columns are the x-coordinate and y-coordinate, respectively.

```
--- The coordinates matrix of words ---
[[ 3.875       1.9375    ]
 [ 3.          0.25      ]
 [ 0.          4.        ]
 [ 1.04166667  6.8125    ]
 [ 2.45833333  8.        ]
 [ 9.5         2.8125    ]
 [ 2.45833333  0.        ]
 [ 1.5         6.4375    ]
 [ 2.45833333  7.        ]
 [12.          4.        ]]
```

In addition, I plotted the mapping result with the help of `matplotlib`.


forward
order
coward
odor
compute
international
interaction
acting
activist
activity
0
2
4
6
8
10
12
0
1
2
3
4
5
6
7
8