Part 1: Implementation

**FastMap**

Data structure

Instead of being provided the distance function, we had only the result from the “fastmap-data.txt”. Therefore, we constructed a dictionary that stored the distances between each object to be the distance function in this program. In addition, the array (P) whose dimension is (number of object, dimension of space created) is created to store the outputs after each iteration.

Code-level optimizations

We first used library pandas to read the file. However, we later found that it is unnecessary to use the dataframe structure as the column names are not important to us and the way to get the data is not instinct, for example, if we want to get the value at row1, column2, it is not df[0][1] but df[1][0]. We then turned to loadtxt function from numpy, which gave us a matrix so that it is more convenient and instinct when retrieving data.

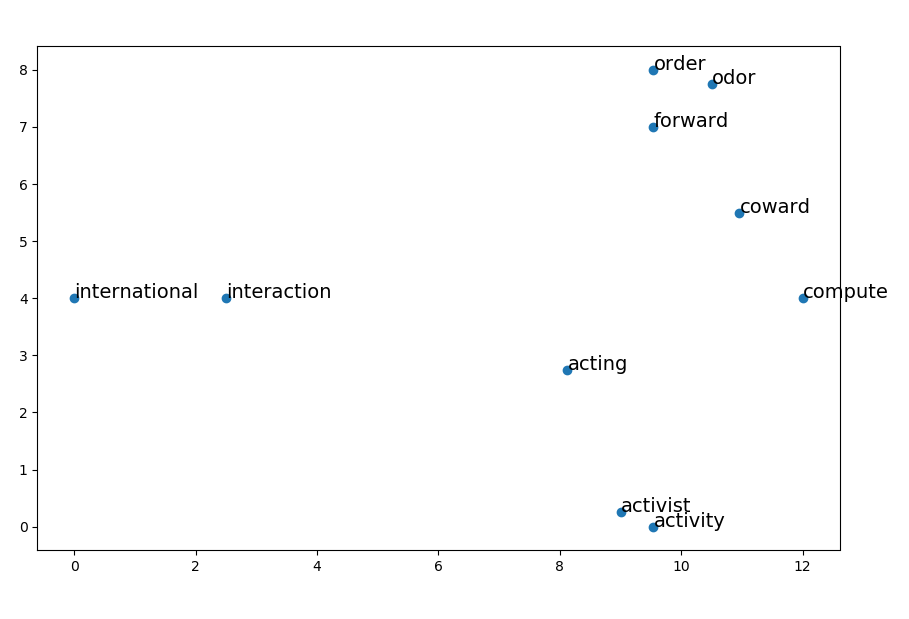
In addition, the functions were built independently at first. It was noticed that each function took lots of parameters and not organized. Hence, to increase the readability of the program, the program was reorganized into object-oriented style, building all the functions needed in fastmap class.

Challenge

The biggest challenge was we were not familiar with the fastmap concepts, so when the first result came out, we were not sure if that makes sense. All we understood was that the similar words should be closer than other words. Although part of the result seemed not to follow such rule, we could not find error after reviewing the codes at that time. After being stuck for a while, we came up with the way to check if the max pair in each dimension shows the longest distance in corresponding axis, then we found that there was an error that when calculating the distance.

2D Plane Output

We got the output as below, showing that the similar words are closer to each other and the wordlists provided could be therefore clustered into several groups if we need further analysis on those words.



Part 3: Implementation

**FastMap**

As FastMap transform abstract distance concepts into Euclidean distance, it makes it possible to plot data points in 2D space or even higher dimensions while remaining their distance features. Therefore, Fastmap is found being applied on clustering and visualization a lot. Take WINE dataset for example, there are 13 variables of each data point (row) and the wine is derived from three different cultivars. With the help of Fastmap, it is feasible to separate one of the clusters on a 2D plane. Moreover, the clusters are distinguished nearly accurately on the 3D plot.