Bayesian Phylogenetic Method for Austronesian clustering

Introduction

This report was classified into three parts. Firstly, the sample was selected based on previous hypotheses. Secondly, an improved model for clustering and different linkage methods for clustering were presented. For the sake of clarification, PCA method has been quoted. Finally, theoretical method was elucidated.

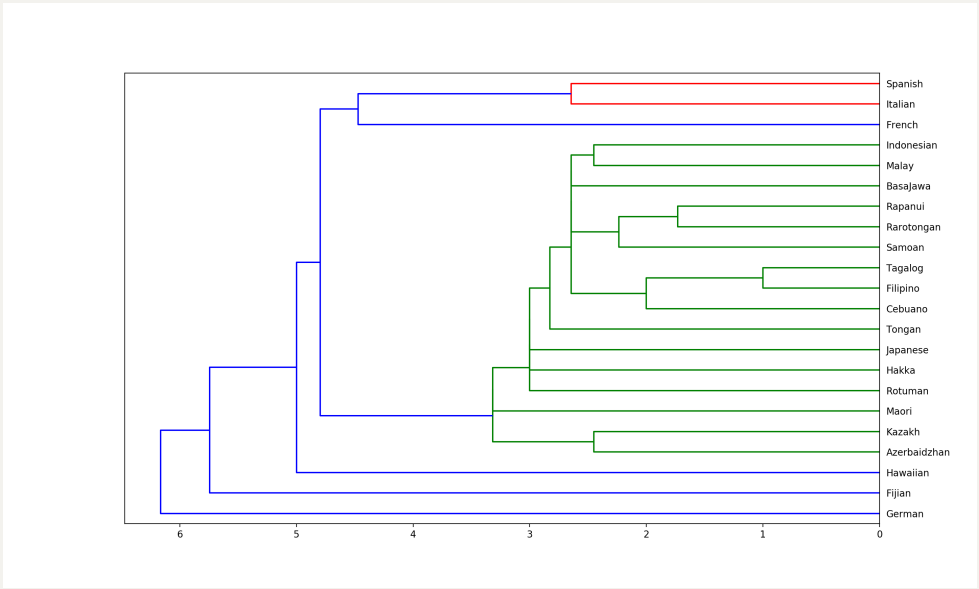
This experiment abandoned the SPSS, replaced by python, mechine learning has been used.

Linkage method for clustering-An Improved Model

1. Single Linkage/Nearest Neighbor Method

$D(A,B) = min(d(y_i, y_j))$

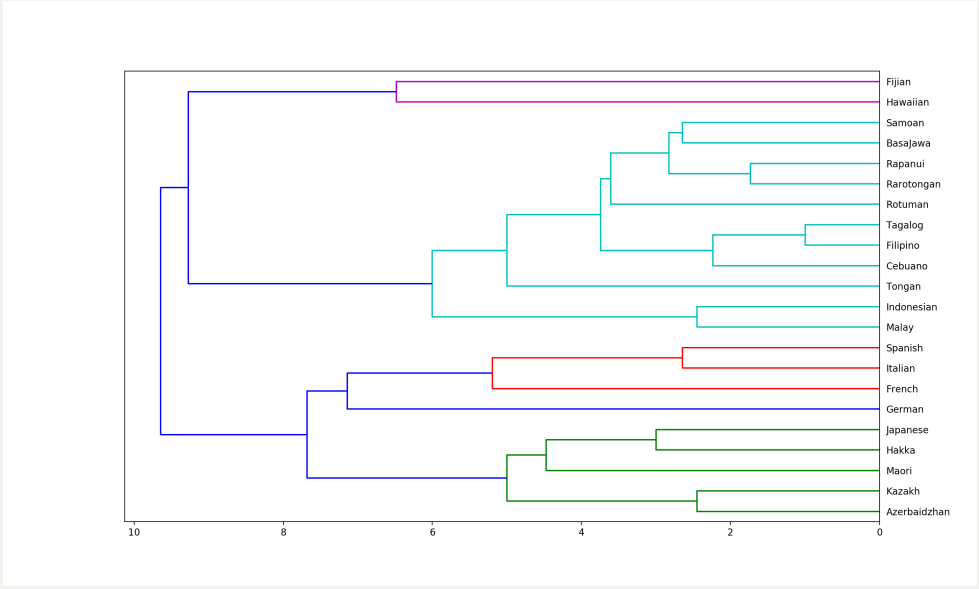
$d(y_i, y_j)$ is Euclid distance



2. Complete Linkage/Farthest Neighbor Method

The distance between groups is defined by the distance between the most distant individuals of the two groups

$$D(A,B) = max(d(y_i, y_j))$$



3. Average Linkage

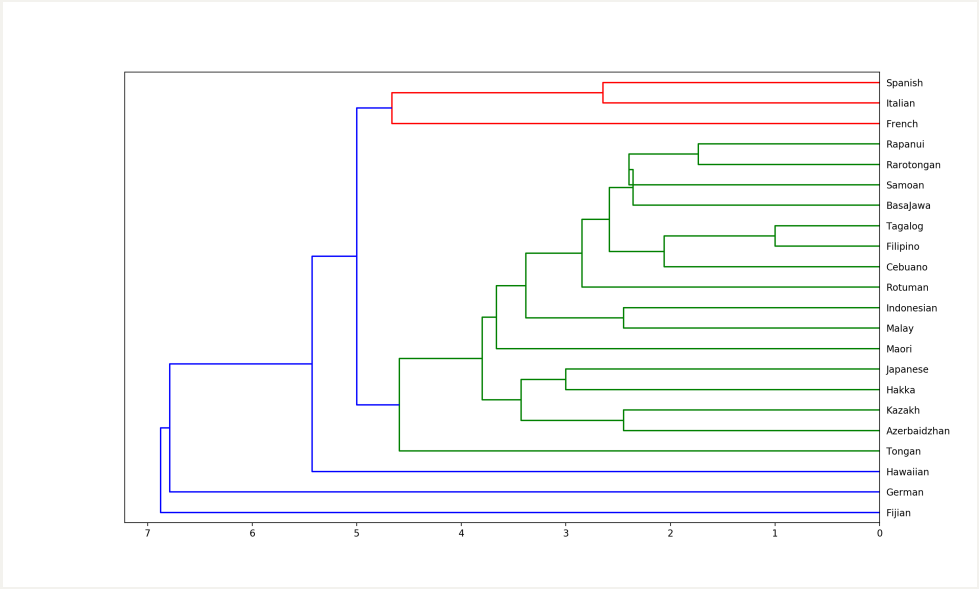
$$D(A,B) = \frac{1}{n_A n_B} \sum \sum d(y_i, y_j)$$

4. Centroid Method

The distance between the two groups is defined as the centroid of mass of each group

$$D(A,B) = d(\hat{y_A}, \hat{y_B}), \hat{y_A} = \frac{1}{n_A} \sum y_i, \hat{y_B} = \frac{1}{n_B} \sum y_j$$

$$y_{AB} = \frac{n_A \hat{y_A} + n_B \hat{y_B}}{nA + nB}$$

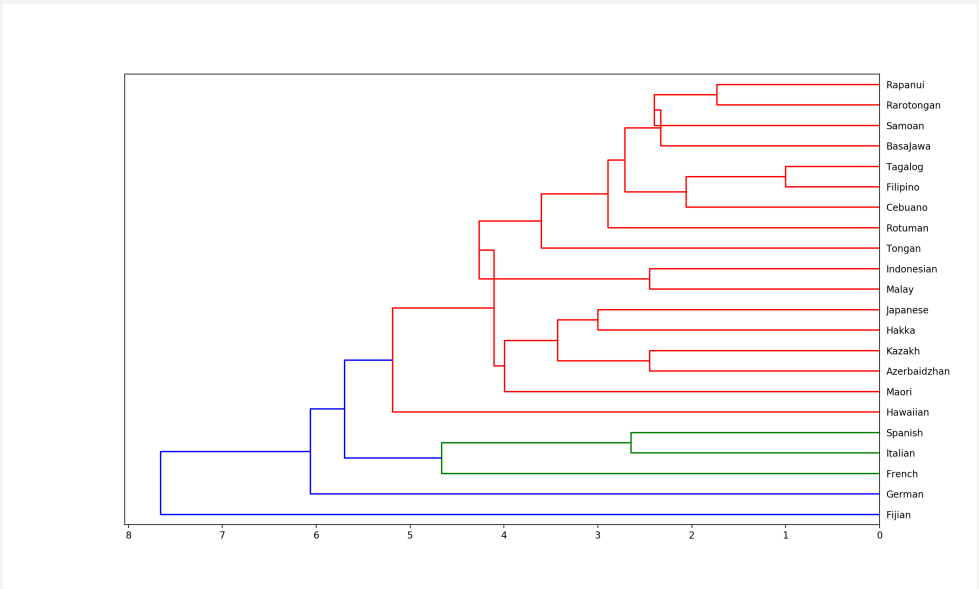


If the combination of the two groups results in a Reversal of the minimum distance (its center of mass changes), it is called Inversion, which is represented in the system tree pattern as Crossover

5. The Centroid Method Based on the Midpoint

In order to reduce the error casued by large sample based on centroid, midpoint could be used as modification.

$m_{AB} = \frac{1}{2}(\hat{y_A} + \hat{y_B})$



6. Ward Method/Incremental sum of squares

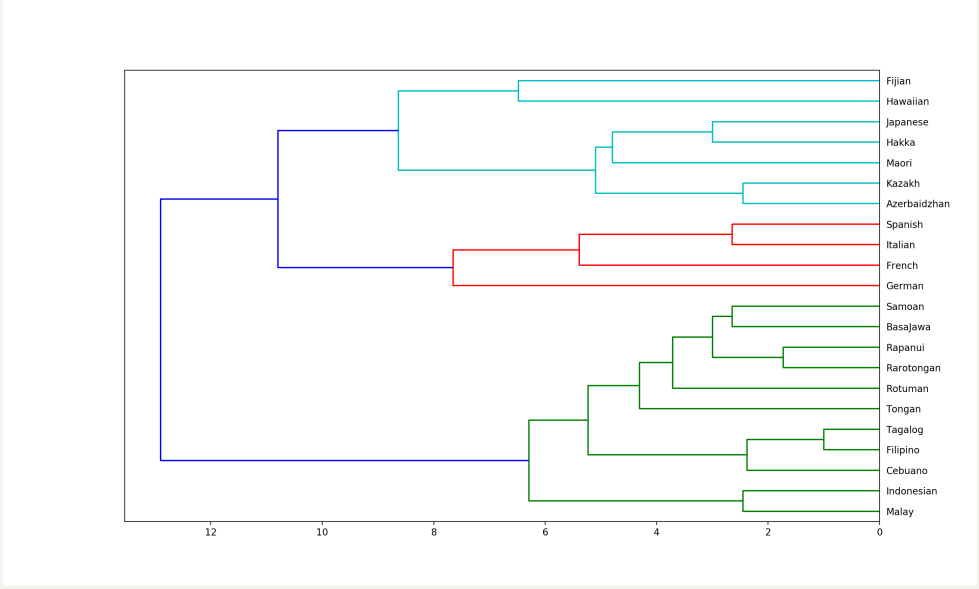
The definition of the distance:

$I_{AB} = SSE_{AB} - (SSE_A + SSE_B)$

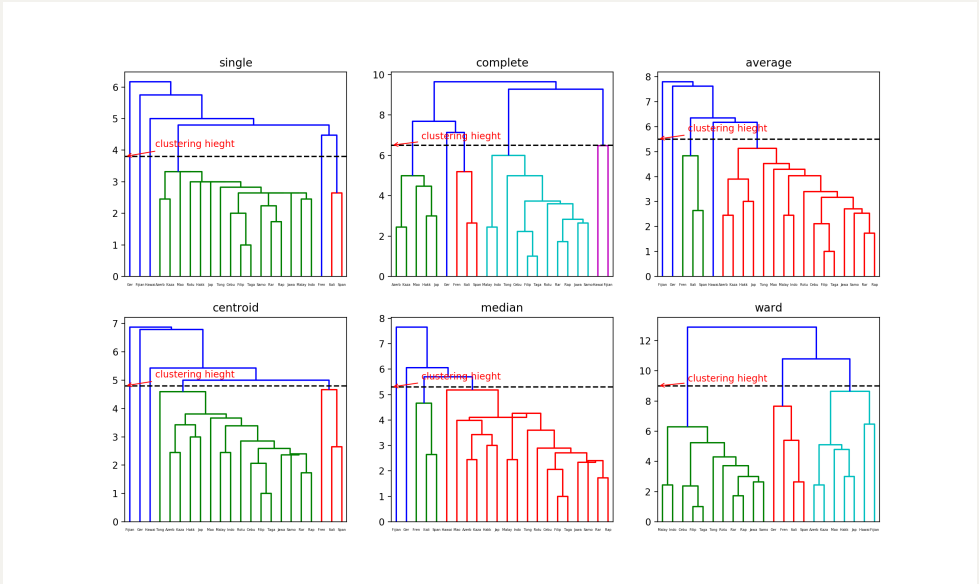
$$SSE_A = \sum_{i=1}^{n_A} (y_i - \bar{y}_A)' (y_i - \bar{y}_A) \text{ for } y_i \in A$$
$$SSE_B = \sum_{i=1}^{n_B} (y_i - \bar{y}_B)' (y_i - \bar{y}_B) \text{ for } y_i \in B$$
$$SSE_{AB} = \sum_{i=1}^{n_{AB}} (y_i - \bar{y}_{AB})' (y_i - \bar{y}_{AB}) \text{ for } y_i \in AB$$

$$\bar{y}_{AB} = \frac{\sum_{i=1}^{n_A} y_i + \sum_{j=1}^{n_B} y_j}{n_A + n_B} = \frac{n_A \bar{y}_A + n_B \bar{y}_B}{n_A + n_B}$$

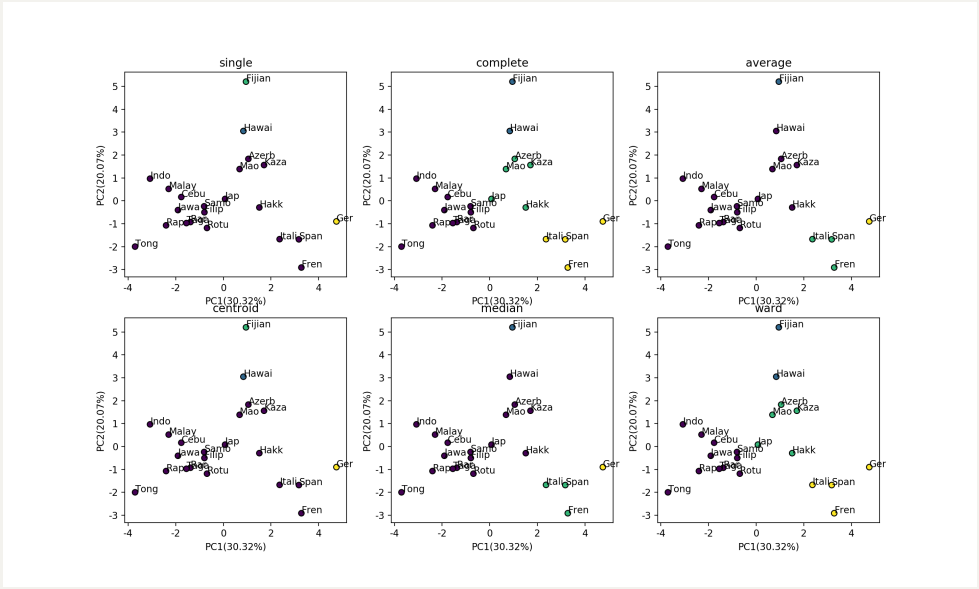
Ward method tends to combine groups with small sample size



Totally, clustering height was texted on the graph so it was clear to see the classification through different color.



In order to visualize the clustering results, we need to reduce the dimension, in mechine learning PCA method was a nice way to do this.



Based on the six method, the result was very clear. Since the result of this experiment was affected by so many factors, such as sampling, cognates, geographical influence and so on, the algorithm could be modified continually.

Bayesian Phylogenetic Method and introduction to MCMC

Introduction to MCMC and Bayesian method:

- Inference
    - Accurate Inference
    - Approximate Inference
        - Determinist (Varational Inference)
        - Stochastic (Markov chain & Monte Carlo)

Bayes formula:

$$f\left(\tau_i \mid \boldsymbol{X}\right)=\frac{f(\boldsymbol{X}|\tau_i)f(\tau_i)}{\sum_{j=1}^{B(s)} f(\boldsymbol{X}|\tau_j)f(\tau_j)}$$

$$E_{z|x}(f(z))=\int p(z|x)f(x)dx=\frac{1}{N}\sum f(z_i)$$

which could generate sample as, $z_1, z_2 \dots z_n$

There was one way which was suitable for the study, called importance sampling

$$E_{p(z)}(f(z))=\int p(z)f(z)dz=\int \frac{p(z)}{q(z)}q(z)f(z)dz=\int f(z)\frac{p(z)}{q(z)}q(z)dz=\frac{1}{N}\sum f(z_i)\frac{p(z_i)}{q(z_i)}$$

Markov chain: time and the situation were discrete

$p(x_{t+1=x}|x_1, x_2..x_t) = p(x_t + 1|x_t)$

Detail Balance:

$\pi(x)p(X|x) = \pi(X)p(x|X)$

However, this is not always the case, we need another varible to balance this equation.

$\alpha(xX) = min(1, \frac{p(X)Q(x|X)}{p(x)Q(X|x)})$

this is called acceptance rate, which is used to determinate wheter to acceptance the sample or not.

Theorotically, if we put data into BEAST or MEGA7, we could get a graph very similar to the clustering by python but with the posterior probability.

Appendix

The dataset was simplified as follows:

Instead of plugging these values into function, I kept the origin number as dataset, which represent the least steps to do the string-changing.

Malay	4	3	4	5	3	4	5	5	6	6	5	4	4	5	4	4	4
Filip	3	5	4	4	3	3	5	5	4	5	4	5	4	4	4	4	4
Hawai	5	4	5	4	4	4	5	4	3	3	8	3	4	6	3	4	4
Fijian	4	3	4	4	3	3	5	5	3	2	7	3	8	5	7	4	5
Jawa	5	3	4	5	3	4	5	5	4	6	4	4	4	5	3	4	4
Samo	4	3	4	3	3	3	5	5	4	5	4	4	4	5	3	4	4
Indo	4	3	4	5	3	4	5	7	7	6	5	4	4	5	4	4	5
Hakk	3	3	5	4	4	3	5	4	3	3	4	5	3	4	3	4	3
Jap	4	3	5	4	4	4	5	4	4	4	4	5	3	6	3	4	4
Azerb	3	3	5	3	4	4	4	6	6	2	4	5	4	5	4	4	3
Mao	4	3	3	4	3	3	5	5	4	3	5	3	4	4	4	4	5
Rar	3	3	3	4	3	3	5	5	4	6	4	4	4	5	4	4	4
Rap	4	3	3	4	3	3	5	5	4	7	4	4	4	5	4	4	5
Tong	4	3	4	3	3	3	5	5	3	9	4	4	4	5	4	4	4
Rotu	3	3	5	4	3	3	5	5	3	6	4	4	4	5	4	2	4
Cebu	3	4	4	4	3	4	5	5	4	6	5	5	5	4	4	4	4
Taga	3	5	4	4	3	3	5	5	4	6	4	5	4	4	4	4	4
Itali	2	3	2	6	4	2	3	5	2	4	5	5	4	6	4	2	4
Fren	2	4	4	5	3	0	3	4	3	3	3	4	3	5	2	3	3
Ger	3	3	3	3	3	4	2	3	3	2	4	2	3	3	3	0	4
Span	2	3	2	5	4	2	3	4	3	3	4	5	4	6	3	2	5
Kaza	3	3	5	3	4	4	4	4	5	2	4	5	4	5	4	4	4

```
# rewrite them as array
[[4 3 4 5 3 4 5 5 6 6 5 4 4 5 4 4 4]
 [3 5 4 4 3 3 5 5 4 5 4 5 4 4 4 4 4]
 [5 4 5 4 4 4 5 4 3 3 8 3 4 6 3 4 4]
 [4 3 4 4 3 3 5 5 3 2 7 3 8 5 7 4 5]
 [5 3 4 5 3 4 5 5 4 6 4 4 4 5 3 4 4]
 [4 3 4 3 3 3 5 5 4 5 4 4 4 5 3 4 4]
 [4 3 4 5 3 4 5 7 7 6 5 4 4 5 4 4 5]
 [3 3 5 4 4 3 5 4 3 3 4 5 3 4 3 4 3]
```

```
 [4 3 5 4 4 4 5 4 4 4 4 5 3 6 3 4 4]
 [3 3 5 3 4 4 4 6 6 2 4 5 4 5 4 4 3]
 [4 3 3 4 3 3 5 5 4 3 5 3 4 4 4 4 5]
 [3 3 3 4 3 3 5 5 4 6 4 4 4 5 4 4 4]
 [4 3 3 4 3 3 5 5 4 7 4 4 4 5 4 4 5]
 [4 3 4 3 3 3 5 5 3 9 4 4 4 5 4 4 4]
 [3 3 5 4 3 3 5 5 3 6 4 4 4 5 4 2 4]
 [3 4 4 4 3 4 5 5 4 6 5 5 5 4 4 4 4]
 [3 5 4 4 3 3 5 5 4 6 4 5 4 4 4 4 4]
 [2 3 2 6 4 2 3 5 2 4 5 5 4 6 4 2 4]
 [2 4 4 5 3 0 3 4 3 3 3 4 3 5 2 3 3]
 [3 3 3 3 3 4 2 3 3 2 4 2 3 3 3 0 4]
 [2 3 2 5 4 2 3 4 3 3 4 5 4 6 3 2 5]
 [3 3 5 3 4 4 4 4 5 2 4 5 4 5 4 4 4]]
```

Code (python) for clustering:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, ward
from sklearn.decomposition import PCA

dataset = pd.read_excel('./语言研究.xlsx',encoding="utf-
8")
A = dataset.iloc[:,1:18].values
distA = sch.distance.pdist(A,'euclidean')
distB =
pd.DataFrame(sch.distance.squareform(distA.round(2)),col
umns=[i for i in range(22)],index=[i for i in
range(22)])
Z = sch.linkage(A, method ='ward',metric='euclidean')
#euclidean代表欧式距离，#single代表简单连接



fig=plt.figure(figsize=(6,10))
dendrogram = sch.dendrogram(Z,labels =
dataset['language'].values
                ,orientation='left'
```

```
                ,leaf_rotation=0
                ,leaf_font_size=10
               )


plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

fig=plt.figure(figsize=(13.5,20))

method_type=
['single','complete','average','centroid','median','ward
']
hlines_cut=[3.8,6.5,5.5,4.8,5.3,9]

for i in range(len(method_type)):
    ax=fig.add_subplot(2,3,i+1)
    Z = sch.linkage(A, method
=method_type[i],metric='euclidean')

    sch.dendrogram(Z,labels =
dataset['language'].values,leaf_rotation=0
,leaf_font_size=3.5)

 plt.hlines(y=hlines_cut[i],xmin=0,xmax=1000,linestyles=
'dashed')
    plt.annotate('clustering hieght',xy=
(0,hlines_cut[i]),xytext=(30,hlines_cut[i]+0.3)
,color='r',arrowprops=dict(arrowstyle="-
>",color='red',connectionstyle="arc3"))
    plt.title(method_type[i])
plt.show()
```

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
from scipy.cluster.hierarchy import dendrogram, ward
from sklearn.decomposition import PCA
```

```
dataset = pd.read_excel('./语言研究.xlsx',encoding="utf-8")
A = dataset.iloc[:,1:18].values

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False


pca = PCA(n_components = 0.95)
result =pca.fit_transform(A)

fig=plt.figure(figsize=(13.5,16))

method_type=['single','complete','average','centroid','median','ward']

for i in range(len(method_type)):
    ax=fig.add_subplot(2,3,i+1)
    Z = sch.linkage(A, method=method_type[i],metric='euclidean')
    label = sch.cut_tree(Z,n_clusters=4)
    dataset['cluster_%s'%(method_type[i])]=label


    plt.scatter(result[:, 0], result[:, 1], c=dataset['cluster_%s'%(method_type[i])], edgecolor='k')
    for j in range(result[:,0].size):

 plt.text(result[j,0],result[j,1],dataset['language'].values[j])
        x_label = 'PC1(%s%%)' % round((pca.explained_variance_ratio_[0]*100.0),2)
        y_label = 'PC2(%s%%)' % round((pca.explained_variance_ratio_[1]*100.0),2)
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.title(method_type[i])

plt.show()
```