1. I've iterate over number of tri-graph and found 36 entry has the most accuracy. 36 entry mean 9 words then 9 words is enough.
2. No because more people mean more words needed for identify a person. Hence, thousand people require very long sentence to classify.
3. No because it's not practical and I need my system to be scalable

I use two code 1. Data collect 2. Classification

```
from tkinter import *
import time
from collections import defaultdict

class GUI:
    def __init__(self, master):

        # self.targetText = "it was at that moment that he learned there are c
ertain parts of the body that you should never nair"
        self.targetText = "the quick brown fox jumps over the lazy dog"
        self.name = None


        self.master = master
        master.title("A simple GUI")

        self.data = defaultdict(list)

        self.nameLabelText = StringVar()
        self.nameLabelText.set('Enter username')


        self.nameLabel = Label(master, textvariable=self.nameLabelText)
        self.nameLabel.pack()

        self.nameEntry = Entry(master)
        self.nameEntry.pack()

        self.targetLabel = Label(master, text=self.targetText)
        self.targetLabel.pack()

        self.currentLabelText = StringVar()
        self.currentLabelText.set('Please enter username')

        self.currentLabel = Label(master, textvariable=self.currentLabelText,
fg='green')
        self.currentLabel.pack(pady=15)

        self.startCollectButton = Button(master, text="Greet", command=self.st
art)
        self.startCollectButton.pack()
```

```
        self.close_button = Button(master, text="Close", command=master.quit)
        self.close_button.pack(pady=15, padx=20)

        self.currentLabel.bind("<Key>", self.onKey)
        self.reset()

    def start(self):
        self.name = self.nameEntry.get()
        self.nameEntry.config(state='disabled')
        self.nameLabelText.set('User: {}'.format(self.name))
        self.currentLabel.focus();
        self.reset()

        self.isStartType = True

    def reset(self):
        self.isStartType = False
        self.currentInput = ''
        self.nextIndex = 0
        self.currentLabelText.set("Please type")
        self.triGraph = list()
        self.times = []

    def finish(self):
        self.nameEntry.config(state='normal')
        self.data[self.name].append(self.triGraph)
        print(self.data)

    def onKey(self, event):
        if not self.isStartType:
            return
        key = str(event.char)
        index = self.nextIndex
        self.nextIndex += 1

        if self.targetText[index] == key:
            self.currentInput += key

            self.times.append(time.time_ns())

            # finish
            if self.nextIndex == len(self.targetText):
                print(self.triGraph)
                self.finish()
                self.reset()
                self.currentInput = "Finsih"

            if index >= 2:
```

```
                self.triGraph.append(self.times[index] - self.times[index-2])

        else:
            self.reset()
        self.currentLabelText.set(self.currentInput)


root=Tk()
gui = GUI(root)
root.mainloop()
```

[4]:
```
from sklearn.neighbors import NearestNeighbors
```

[5]:
```
data = {'flap': [[516637100, 603540400, 423332000,
```

[13]:
```
train = data['flap'][0:4] + data['yok'][0:4]
```

[12]:
```
test_flap = data['flap'][4]
test_yok = data['yok'][4]
```

[30]:
```
neigh = NearestNeighbors(n_neighbors=1)
```

[31]:
```
neigh.fit(train)
```

[31]:
```
NearestNeighbors(n_neighbors=1)
```

[35]:
```
neigh.kneighbors([test_yok])[1][0][0] > 3
```

[35]:
```
True
```

[37]:
```
neigh.kneighbors([test_flap])[1][0][0] <= 3
```

[37]:
```
True
```

[ ]:
```
# if class < 3 -> flap, class > 3 -> yok
```