Final Report

Team Name : Rosseta （Project B）

Team Leader: Yuchen Lin (林禹臣 5140309507)

Team Member: Zhaorun Han (撖朝润 5140309504)

Date: Jan. 9th 2015

1 Prototype System Introduction

1.1 Functions

Our project named _Rosseta Seg_ can segment Chinese passages into sentences and words mostly correctly even when some other language characters or symbols in the sentence.

We also provide the capability of manipulating lexicon and rules library. Users can DIY the dictionary and rules with their own wills.

1.2 Running Environment

Windows 7 8 8.1
Mac OS X 10.10
Other editions as long as installed python

1.3 Developing Environment

PyCharm 4.0 community edtion.

2 Task Allocation

Yuchen Lin → All background program.
(Segmentation Algorithm & Lexicon Modify & Rules Modify & Connection to UI)

Zhaorun Han → UI component. Every windows and every buttons.

3 Features

3.1 Friendly UI

Our UI is very easy to use and understand the functions of every button.

3.2 Accuracy

We can recognize words which are not in the lexicon well such as name of people and name of place.

3.3 Extensible

You can directly modify the lexicon and rules (such as " DELETE 的" or "CHANGE 我 WITH 笔者") .

4 System Architecture

4.1 UI

4.1.1 Main Window

(1) There is a menu on the top, which can lead us to all functions of this program. File→Open can let you open a text file, which contains the paragraph you want to segment into words.
(2) The two big text boxes are very convenient for users to input the sentence he want to segment and copy the result.
(3) The right two buttons can make you segment as your will. If you just want to get sentences, please press the first button. Otherwise, press the other one to get segmented words.
(4) Press the save button, you can save your result and create a new file.

4.1.2 Lexicon Modify Window

When you open this window, you will find the left part is a text box, which contains many words and numbers. The numbers is the frequency of the word, and you can directly modify both of them.

The right part can enable you to add a new word into the dictionary.

4.1.3 Rules Modify Window

It's almost the same as Lexicon part.
Now, we just provide two rules you can modify.

DELTE
CHANGE…WITH

4.1.4 Help

4.2 Background

In order to obtain a better result, I constructed two level (or we may say, two layers) of segmentation algorithm.

Layer 1 is based on dictionary and frequency of each word. I consider every character as a vertex (point) and every word as an edge connecting the two vertexes, and the possibility (frequency) is actually the weight of the edge.

Thus, we get a DAG, which means *Directed Acyclic Graph (有向无环图).* And therefore, we could use the knowledge in Graph

Theory to find the largest possibility route of this graph.

Because of being based on dictionary, layer1 can not process un-login words. (未登录词) Therefore, I made layer 2.

Layer 2 , however , is based on ***HMM (Hidden Markov Model,隐马尔科夫模型)*** and ***Viterbi Algorithm***(维特比算法). I consider the position (B,E,M,S) of every character is a hidden state and the sentence to segment is a visible output of the chain of those hidden states.
In addition, we have to consider that the position of every character is only the result of the former one so that the sentence can be regard as a ***Markov Chain***(马尔科夫链).

5 Algorithm Description

5.1 UI

5.1.1 TKinter

We choose TKinter as our platform to establish our UI, because it’s based on TCL language, so that it can be run on many OS like windows and MAC.
All buttons and textbox… are the build-in parts of tkinter.

5.1.2 Commands

When we press each button, we want to call some function to get some kinds of results. This is based on command. Wherever the button is, the command is there.

5.1.3 Advantages

5.1.3.1 Easy is beautiful.

Our UI is very easy to know the function of every button, which is friendly and charming.

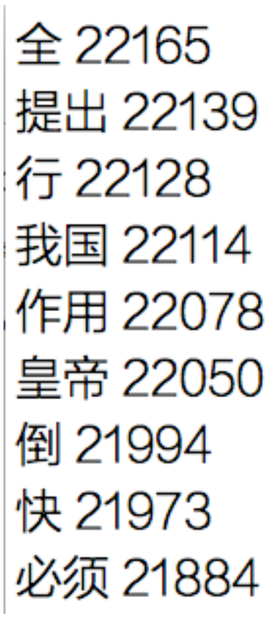
5.1.3.2 Considerate.

When you segment the sentences or add a word into Lexicon, we will take the curse and scroll bar to the end, which is very convenient for users.

5.2 Background

5.2.1 Lexicon

a. Data Structure :
dict.txt



全	22165
提出	22139
行	22128
我国	22114
作用	22078
皇帝	22050
倒	21994
快	21973
必须	21884

b. Load Method

```
#cut every word in Dict into several pieces to be prefix
def generatePrefixDict():
global Possibility,PrefixDict,total,Dictionary_URL
fo = Lexicon.loadDict(Dictionary_URL)
PrefixDict = set()
FREQ = {}
for line in fo.read().rstrip().split("\n"):
word,freq = line.split(' ')[:2]
FREQ[word] = float(freq)
total += float(freq) #calculate the total number of words
for idx in range(len(word)): #generate the prefix
dictionary
prefix = word[0:idx+1]
PrefixDict.add(prefix)
fo.close()
#Transform the freq into possibilities
Possibility = dict((key,log(value/total)) for key,value in
FREQ.items())
```

c. Prefix Dictionary
   This is a data structure called Trie Tree(Trie 树 前缀树)

5.2.2 DAG & Route

```
#get the DAG of a sentence
def getDAG(sentence:str):
    global PrefixDict,Possibility
    DAG = {}
    N = len(sentence)
    for i in range(N):
        lst_pos = []
        #this list is to save the indexes of those chars who can make up a word
with i-th char.
        j = i
        word = sentence[i]
        #if this char is not in dict, we must let it be a single word
        lst_pos.append(i)
        while(j<N and word in PrefixDict):
            if(word in Possibility):
                if(lst_pos[0]!=j):
                    lst_pos.append(j)
            j+=1
            word = sentence[i:j+1]
        #put this list of i-th into DAG[i]
        DAG[i]=lst_pos
    return  DAG
```

After we got a DAG, we should calculate the route for this graph, which has the max possibility.

```
#Dynamic Planning.
#calculate the most possible route of every beginning-char.
def calculateRoute(DAG:dict,route:dict,sentence:str):
    #from right to left
    global Possibility , min_Possibility

    N = len(sentence)
    route[N]=(0.0,'')
    #In fact, it is a recurse procedure.
    for i in range(N-1,-1,-1):
        max_psblty = None
        max_cur = i
        for j in DAG[i]:
            #to consider every possible word
            word = sentence[i:j+1]
            posb1 = Possibility.get(word,min_Possibility)
            posb2 = route[j+1][0]
            posb = posb1 + posb2
            if max_psblty is None:
                max_psblty = posb
            if(posb>=max_psblty):
                max_psblty = posb
                max_cur = j
        route[i] = (max_psblty,max_cur)
```

5.2.3 HMM

HMM can be considered as two parts. One is the model to POS, the other one is to calculate the best way. (Viterbi Algorithm)

```
def viterbi(obs):
     V = [{}]
     path={}
     for s in States:
          V[0][s] = start_P[s] + emit_P[s].get(obs[0],MIN_FLOAT)
          path[s] = [s]
     for i in range(1,len(obs)):
          char = obs[i]
          V.append({})
          newPath = {}
          for s in States:
               emit =   emit_P[s].get(char, MIN_FLOAT)
               prob_max=MIN_FLOAT
               state_max = PrevStatus[s][0]
               for preState in PrevStatus[s]:
                    prob = V[i-1][preState] + trans_P[preState][s] + emit
                    if(prob>prob_max):
                         prob_max = prob
                         state_max = preState
               V[i][s] = prob_max
               newPath[s] = path[state_max] + [s]
          path = newPath

     finalProb = MIN_FLOAT
     finalState = ""
     for fs in ('E','S'):
          p = V[len(obs)-1][fs]
          if(p>finalProb):
               finalProb = p
               finalState = fs
     return (finalProb,path[finalState])
```

5.2.4 Segment & Output

After we have the route and how to cut the un-login word, the rest part is very easy, which just needs us to control the process. We especially paid attention to the English words, digits and some other symbols. The code of this part is huge, and therefore I decided not to present them here.

5.2.5 Rules

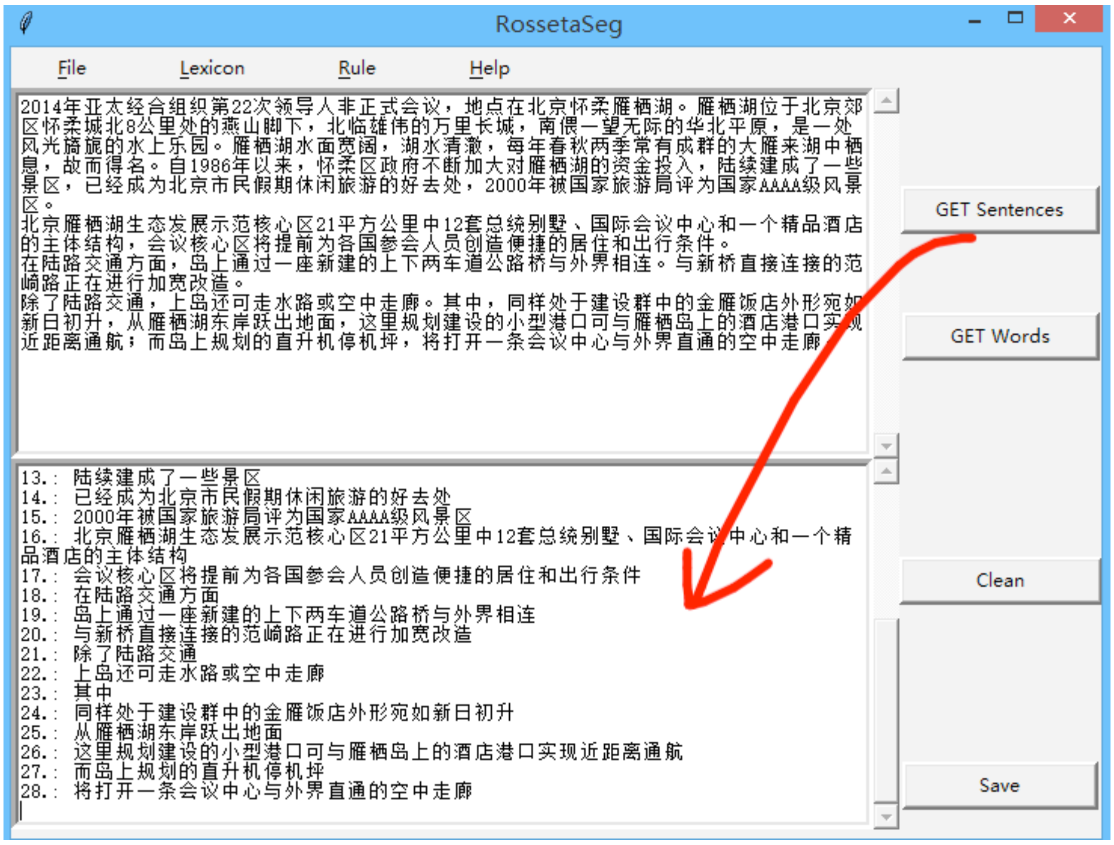
I used dictionary and list to save the rules and applied the rules in the pre-process section.

```
Rules = {"CHANGE":{},"DELETE":[]}
RuleUrl = "rule.config"

#load rules into memory
def loadRules(url=RuleUrl):
	with open(url,"r",encoding="gbk") as fo:
		lst = fo.read().rstrip().split("\n")
		while '' in lst:
			lst.remove('')
		for line in lst:
			ins = line.split(" ")
			typ = ins[0]
			if(typ!="CHANGE"):
				Rules[typ].append(ins[1])
			elif(typ=="CHANGE"):
				Rules["CHANGE"][ins[1]]=ins[3]
```

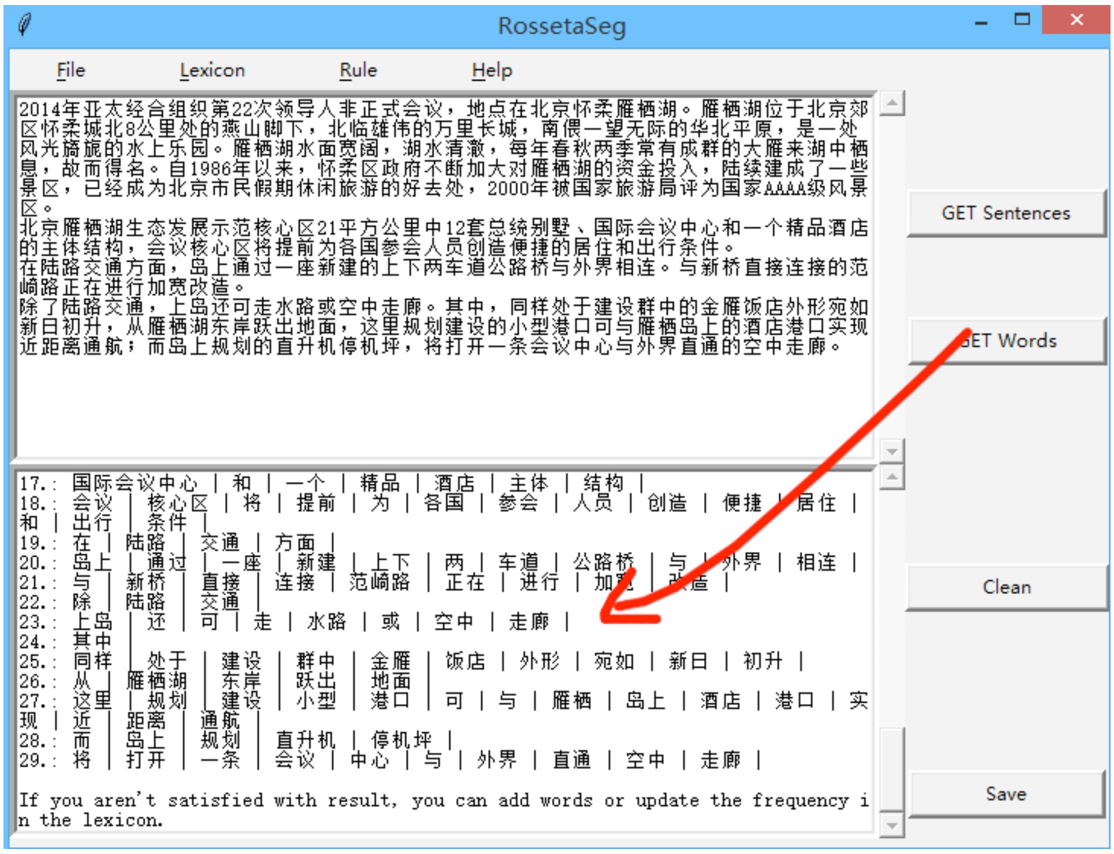
6 Demo

6.1 Open & Segment



```
2014年亚太经合组织第22次领导人非正式会议，地点在北京怀柔雁栖湖。雁栖湖位于北京郊
区怀柔城北8公里处的燕山脚下，北临雄伟的万里长城，南偎一望无际的华北平原，是一处
风光旖旎的水上乐园。雁栖湖水面宽阔，湖水清澈，每年春秋两季常有成群的大雁来湖中栖
息，故而得名。自1986年以来，怀柔区政府不断加大对雁栖湖的资金投入，陆续建成了一些
景区，已经成为北京市民假期休闲旅游的好去处，2000年被国家旅游局评为国家AAAA级风景
区。
北京雁栖湖生态发展示范核心区21平方公里中12套总统别墅、国际会议中心和一个精品酒店
的主体结构，会议核心区将提前为各国参会人员创造便捷的居住和出行条件。
在陆路交通方面，岛上通过一座新建的上下两车道公路桥与外界相连。与新桥直接连接的范
崎路正在进行加宽改造。
除了陆路交通，上岛还可走水路或空中走廊。其中，同样处于建设群中的金雁饭店外形宛如
新日初升，从雁栖湖东岸跃出地面，这里规划建设的小型港口可与雁栖岛上的酒店港口实现
近距离通航；而岛上规划的直升机停机坪，将打开一条会议中心与外界直通的空中走廊
```

```
13.:  陆续建成了一些景区
14.:  已经成为北京市民假期休闲旅游的好去处
15.:  2000年被国家旅游局评为国家AAAA级风景区
16.:  北京雁栖湖生态发展示范核心区21平方公里中12套总统别墅、国际会议中心和一个精
品酒店的主体结构
17.:  会议核心区将提前为各国参会人员创造便捷的居住和出行条件
18.:  在陆路交通方面
19.:  岛上通过一座新建的上下两车道公路桥与外界相连
20.:  与新桥直接连接的范崎路正在进行加宽改造
21.:  除了陆路交通
22.:  上岛还可走水路或空中走廊
23.:  其中
24.:  同样处于建设群中的金雁饭店外形宛如新日初升
25.:  从雁栖湖东岸跃出地面
26.:  这里规划建设的小型港口可与雁栖岛上的酒店港口实现近距离通航
27.:  而岛上规划的直升机停机坪
28.:  将打开一条会议中心与外界直通的空中走廊
```

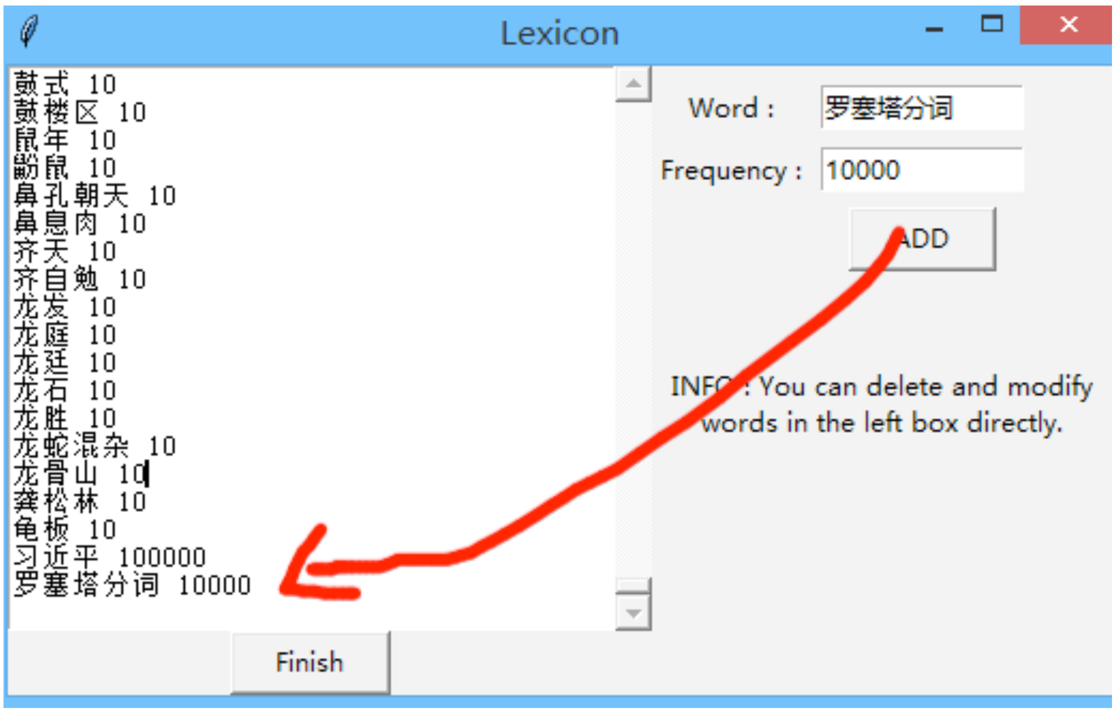


```
2014年亚太经合组织第22次领导人非正式会议，地点在北京怀柔雁栖湖。雁栖湖位于北京郊
区怀柔城北8公里处的燕山脚下，北临雄伟的万里长城，南偎一望无际的华北平原，是一处
风光旖旎的水上乐园。雁栖湖水面宽阔，湖水清澈，每年春秋两季常有成群的大雁来湖中栖
息，故而得名。自1986年以来，怀柔区政府不断加大对雁栖湖的资金投入，陆续建成了一些
景区，已经成为北京市民假期休闲旅游的好去处，2000年被国家旅游局评为国家AAAA级风景
区。
北京雁栖湖生态发展示范核心区21平方公里中12套总统别墅、国际会议中心和一个精品酒店
的主体结构，会议核心区将提前为各国参会人员创造便捷的居住和出行条件。
在陆路交通方面，岛上通过一座新建的上下两车道公路桥与外界相连。与新桥直接连接的范
崎路正在进行加宽改造。
除了陆路交通，上岛还可走水路或空中走廊。其中，同样处于建设群中的金雁饭店外形宛如
新日初升，从雁栖湖东岸跃出地面，这里规划建设的小型港口可与雁栖岛上的酒店港口实现
近距离通航；而岛上规划的直升机停机坪，将打开一条会议中心与外界直通的空中走廊。
```

```
17.:  国际会议中心 | 和 | 一个 | 精品 | 酒店 | 主体 | 结构 |
18.:  会议 | 核心区 | 将 | 提前 | 为 | 各国 | 参会 | 人员 | 创造 | 便捷 | 居住 |
和 | 出行 | 条件 |
19.:  在 | 陆路 | 交通 | 方面 |
20.:  岛上 | 通过 | 一座 | 新建 | 上下 | 两 | 车道 | 公路桥 | 与 | 外界 | 相连 |
21.:  与 | 新桥 | 直接 | 连接 | 范崎路 | 正在 | 进行 | 加宽 | 改造 |
22.:  除 | 陆路 | 交通 |
23.:  上岛 | 还 | 可 | 走 | 水路 | 或 | 空中 | 走廊 |
24.:  其中
25.:  同样 | 处于 | 建设 | 群中 | 金雁 | 饭店 | 外形 | 宛如 | 新日 | 初升 |
26.:  从 | 雁栖湖 | 东岸 | 跃出 | 地面 |
27.:  这里 | 规划 | 建设 | 小型 | 港口 | 可 | 与 | 雁栖 | 岛上 | 酒店 | 港口 | 实
现 | 近 | 距离 | 通航 |
28.:  而 | 岛上 | 规划 | 直升机 | 停机坪 |
29.:  将 | 打开 | 一条 | 会议 | 中心 | 与 | 外界 | 直通 | 空中 | 走廊 |

If you aren't satisfied with result, you can add words or update the frequency i
n the lexicon.
```

6.2 Lexicon



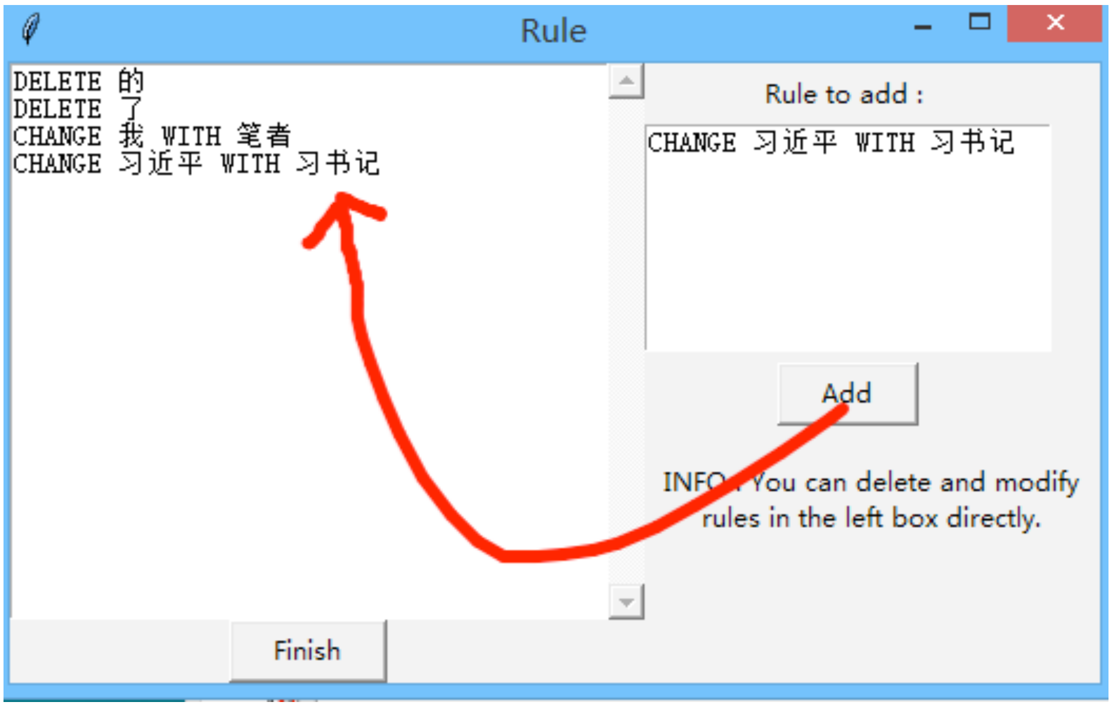
```
鼓式 10
鼓楼区 10
鼠年 10
鼢鼠 10
鼻孔朝天 10
鼻息肉 10
齐天 10
齐自勉 10
龙发 10
龙庭 10
龙廷 10
龙石 10
龙胜 10
龙蛇混杂 10
龙骨山 10
龚松林 10
龟板 10
习近平 100000
罗塞塔分词 10000
```

Word : 罗塞塔分词

Frequency : 10000

INFO : You can delete and modify words in the left box directly.

6.3 Rules



```
DELETE 的
DELETE 了
CHANGE 我 WITH 笔者
CHANGE 习近平 WITH 习书记
```

Rule to add :

CHANGE 习近平 WITH 习书记

INFO : You can delete and modify rules in the left box directly.

7 Conclusion

Through this experience of team project, we know that these qualities are essential for us students, which are:

Learn by ourselves

All knowledge of HMM, graph theory and Possibility theory are new and difficult to us, but only can we search on the internet and learn the basic mechanism it can work. I usually work all the night to make it work.

Especially, UI section is the totally new to Zhaorun Han, so he also often stayed up to study it.

Cooperation

A very essential part of our work is to combine the UI and background program, which needs our understanding with each other.

Improve Skills By Debugging

Sometimes the way we want to get new knowledge is to debug the program when it’s wrong. While we were doing this, we got a lot.

8 References

HMM:

http://blog.csdn.net/likelet/article/details/7056068

Viterbi:

http://en.wikipedia.org/wiki/Viterbi_algorithm