Shanghai Jiao Tong University

2013

Final Project Report

Chinese Word Segmentation

陈桢，李健达，袁全宁，刘柏辰

Team Name: Destiny

Index

[**1)** **Overview** 2](#_Toc345071254)

[**a.** **System Function** 2](#_Toc345071255)

[**b.** **Running Environment** 2](#_Toc345071256)

[**c.** **Development Environment** 2](#_Toc345071257)

[**2)** **Task Allocation** 3](#_Toc345071258)

[**3)** **System Architecture** 3](#_Toc345071259)

[**a.** **User Interface Component** 4](#_Toc345071260)

[**b.** **Process Component** 4](#_Toc345071261)

[**4)** **Algorithm Description** 4](#_Toc345071262)

[**a.** **Input Dictionary Files** 4](#_Toc345071263)

[**b.** **Sentence Segmentation** 5](#_Toc345071264)

[**c.** **Choose a Sentence or the Whole Passage** 6](#_Toc345071265)

[**d.** **Cut Sentence into Small Pieces** 7](#_Toc345071266)

[**e.** **Word Segmentation** 10](#_Toc345071267)

[**f.** **Name Segmentation** 11](#_Toc345071268)

[**g.** **Combine** 12](#_Toc345071269)

[**h.** **Rules and Rectify** 12](#_Toc345071270)

[**5)** **Demo and Testing Result** 13](#_Toc345071271)

[**a.** **Testing** 13](#_Toc345071272)

[**b.** **Screenshots** 14](#_Toc345071273)

[**6)** **Conclusion and Comments** 18](#_Toc345071274)

[**a.** **From Chen Zhen** 19](#_Toc345071275)

[**b.** **From Yuan Quanning** 19](#_Toc345071276)

[**c.** **From Li Jianda** 19](#_Toc345071277)

[**d.** **From Liu Baichen** 19](#_Toc345071278)

1. **Overview**

This is the final report for the project Chinese Word Segmentation of Computation Theory course at the Shanghai Jiao Tong University, by Professor Tianfang Yao. This report will explain the task, details in algorithm and the final result of our program in the following chapters.

1. **System Function**

Our task is to make an automatic Chinese Word Segmentation System which can separate a Chinese passage into sentences and sentences into words quickly and accurately especially for the articles on the EXPO website. The program should have the following parts:

* User Interface
* Text Input and Output
* Sentence Segmentation
* Word Segmentation
* Lexicon
* Rules

1. **Running Environment**

The running environment is recommended to be Microsoft Windows 7 with Python 3.2 to support its running.

1. **Development Environment**

The development environment is Microsoft Windows 7 with the development software PyScripter 2.5.3 and language Python 3.2.

1. **Task Allocation**

As the program contains many parts, each team member was required to finish one or more parts and then combined them together to test the whole process. The detail arrangement is shown as follows:

Chen Zhen: Design the algorithm for the sentence and word segmentation, then do the implementation and improve the performance.

Li Jianda: Design the user interface including the menu bar and the command buttons, then makes the text input and output function.

Yuan Quanning: Make the rule functions and debug the whole system, then use some articles on the EXPO website to test the program and improve the lexicon.

Liu Baichen: Find some information to build a lexicon for the system and make it reliable by deleting the unnecessary words in it.

1. **System Architecture**
2. **User Interface Component**

This program has both Chinese and English interface for users to choose. Users can add a word into the lexicon or delete it at any time, so does the rules. We have two text boxes. One is for users to input the passages; the other is for our program to output the results.

Next, we take the process of a passage being transformed into words as an example.

1. **Process Component**

First, we us some sentence punctuation as symbols to cut the passages into sentences. Then for each sentence, we have three procedures to make it into words. The first step, we find the numbers, spaces, English words, website address which can be symbols to help cut the sentence into small pieces. The second step, we use the main segment function to cut each piece into words. The last step, we use a function to find whether there are names in this sentence and combine names into a word. Finally, we use rules to rectify it and the work is done. You can choose the location to save the result into a txt file.

But how can we do this, how can we find the names accurately, please look the following chapter.

1. **Algorithm Description**
2. **Input Dictionary Files**

We have five dictionary files: main dictionary, blacklist, whitelist, single first name and double first name. In each file, we make one word one line so that we can get each word easily. But we should notice that at the end of each line, there is a “\n” which we have to neglect when we input the dictionary.

The related function source code is shown as follows:

def input\_dic(a):

dic=[]

lex=open(a,"r")

for line in lex:

if line[-1]=="\n":

dic.append(line[:-1])

else:

dic.append(line)

lex.close()

return dic

single\_name=input\_dic("lexicon/single.txt")

double\_name=input\_dic("lexicon/double.txt")

dic=input\_dic("lexicon/dict.txt")

blacklist=input\_dic("lexicon/blacklist.txt")

whitelist=input\_dic("lexicon/whitelist.txt")

1. **Sentence Segmentation**

For a passage, we notice that some punctuation like “；。？！\n” can be symbol of a sentence except that these punctuation are followed by “’””. For punctuation like “：，”, sometimes they are symbols of a sentence but sometimes not. For example, when “：” is used as a part of time like “12：50”, then it cannot be separated. For “\n” and “\u3000”, we should neglect it when construct the new sentence. Finally, we should give each sentence a number except the empty sentence which has only “\n” or space so that users can choose which sentence he want to segment into words.

The related function source code is shown as follows:

def sentence\_seg(a):

n="1234567890１２３４５６７８９０"

t=0

new=[]

for i in range(len(a)):

if i==len(a)-1:

new.append(a[t:i+1])

elif i+1<len(a) and a[i] in "；。？！" and a[i+1] not in "”’":

new.append(a[t:i+1])

t=i+1

elif i+2<len(a) and a[i]+a[i+1]+a[i+2]=="。’”":

new.append(a[t:i+3])

t=i+3

elif i+1<len(a) and a[i]+a[i+1] in ["。”","。’","","……"]:

new.append(a[t:i+2])

t=i+2

elif a[i] in "，：" and i+1<len(a):

if a[i+1] not in n or a[i-1] not in n:

new.append(a[t:i+1])

t=i+1

elif a[i]=="\n" or a[i]=="\u3000":

new.append(a[t:i])

t=i+1

s=""

p=1

for i in range(len(new)):

t=""

for j in range(len(new[i])):

if new[i][j]!="\n":

t=t+new[i][j]

new[i]=t

if new[i]!="" and new[i]!=" " and new[i]!=" ":

s=s+"{0}.{1}\n".format(p,new[i])

p=p+1

return new,s,p

(new,s,p)=sentence\_seg(list\_ch)

1. **Choose a Sentence or the Whole Passage**

We give two choices to the users. One is to segment only one sentence; the other is to segment the whole passage. So at this time, we have to accept the information from users and get the right sentence from the list. To be careful with the detail, if there is only one sentence, we will not ask the user to do the choice.

The related function source code is shown as follows:

def one(p,new):

if p==0:

return new

else:

t=0

for i in range(len(new)):

if new[i]!="" and new[i]!=" " and new[i]!=" ":

t=t+1

if t==p:

return [new[i]]

return new

if p>2:

p=int(input("Would you like to select one of the above sentences to look at the result of word segmentation?\nIf yes, please input the indicated sentence No.\nIf no, please input 0."))

new=one(p,new)

1. **Cut Sentence into Small Pieces**

Some sentences can be very long which is hard for a program to cut it quickly, so we cut it into small pieces first and deal with some small problems at the same time.

The first thing we consider is the space. If we detect a space, we will report it to the user and ask whether the program should neglect the space. If the space is not neglected, it will be a separate symbol and the sentence becomes some short sentences.

The second thing we consider is the number. If we find a number in the sentence, the program will judge whether it is a day, month or year. Then the program can separate it. Having notice that some numbers are represented in the form of “123,456,789”, so this kind of number should be separated into one part. The punctuation “，.” should be neglected when they are between numbers.

The third thing we consider is the English words and website address. We think such kind of things can be symbol for us to cut sentence into small pieces. Also, the punctuation “./” should be neglected when they are between English words.

The fourth thing we consider is the punctuation “、” which is commonly used in the name list. If we can use it to separate the sentence, it will be quite convenient for the following steps.

After we finish all of the above things, each sentence will become some short pieces and the following process can be pertinence.

The related function source code is shown as follows:

def first(new,sk):

sp=0

for i in range(len(new)):

if " " in new[i]:# spaces

j=0

while j<len(new[i]):

if sk==0:

if new[i][j]==" ":

new[i]=new[i][:j]+"|"+new[i][j+1:]

sp=1

j=j+1

while j<len(new[i]) and new[i][j]==" ":

new[i]=new[i][:j]+new[i][j+1:]

else:

while j<len(new[i]) and new[i][j]==" ":

new[i]=new[i][:j]+new[i][j+1:]

j=j+1

j=0# number

n="1234567890１２３４５６７８９０"

m="年月日"

p=",.:，：．"

while j<len(new[i]):

if new[i][j] in n:

t=j

mark=0

while new[i][j] in n:

j=j+1

if j==len(new[i]):

new[i]=new[i][:t]+"|"+new[i][t:j]+"|"

mark=1

break

if new[i][j] in m:

new[i]=new[i][:t]+"|"+new[i][t:j]+"|"+new[i][j]+"|"+new[i][j+1:]

mark=1

break

if new[i][j] in p:

j=j+1

if mark==0:

new[i]=new[i][:t]+"|"+new[i][t:j]+"|"+new[i][j:]

j=j+1

j=0# website

n=string.ascii\_lowercase+"./"

while j<len(new[i]):

if new[i][j] in n:

t=j

mark=0

while new[i][j] in n:

j=j+1

if j==len(new[i]):

new[i]=new[i][:t]+"|"+new[i][t:j]+"|"

mark=1

break

if mark==0:

new[i]=new[i][:t]+"|"+new[i][t:j]+"|"+new[i][j:]

j=j+1

j=0# punctuation

while j<len(new[i]):

if new[i][j]=="、":

new[i]=new[i][:j]+"|"+new[i][j]+"|"+new[i][j+1:]

j=j+2

j=j+1

j=0# devide

t=0

k=[]

if len(new[i])!=0:

if new[i][-1]!="|":

new[i]=new[i]+"|"

while j<len(new[i]):

if new[i][j]=="|":

k.append(new[i][t:j])

t=j+1

j=j+1

new[i]=k

j=0# remove

while j<len(new[i]):

if new[i][j]=="":

new[i][j:j+1]=[]

j=j+1

return(new,sp)

backup=new[:]

(new,sp)=first(new,0)

if sp==1:

if input("Space has been detected.\nDo you want to neglect the space?\n(1=Yes,0=No)")!="0":

(new,sp)=first(backup,1)

1. **Word Segmentation**

For each small piece, if it is not started with the number, English words and punctuation, we use a main cut function to cut it into words. In this function, we use RMM method which means we take maximum match from right to left. If the word is in the dictionary, the program will cut it. Otherwise, the program will make the length smaller and match it again. During this procedure, we use recursion to write this function so that the function will be neat and easy to read.

The related function source code is shown as follows:

def cut(new,dic):

n="1234567890１２３４５６７８９０"+string.ascii\_lowercase

for i in range(len(new)):

j=0

while j <len(new[i]):

if new[i][j][0] not in n and len(new[i][j])!=1:

(s,t)=rmm(new[i][j],new[i][j][-7:],dic,[])

t=t[::-1]

new[i][j:j+1]=[]

for k in range(len(t)):

new[i][j:j]=[t[k]]

j=j+1

else:

j=j+1

return new

def rmm(s,s1,dic,t):

if len(s)==0:

return s,t

else:

if len(s1)==1 or s1=="……" or s1=="——" or s1 in dic:

t.append(s1)

s=s[:len(s)-len(s1)]

rmm(s,s[-7:],dic,t)

return s,t

else:

return rmm(s,s1[1:],dic,t)

new=cut(new,dic)

1. **Name Segmentation**

After the word segmentation, the names in sentence will be cut into one letter by one letter. So we use a first name dictionary to find whether there is a first name. If the program finds a first name, it will judge whether the following two parts of this letter is also a letter and not in the blacklist. If yes or the following part is in the whitelist, the program will consider it as a name and make these separated letters together.

The related function source code is shown as follows:

def final(new,single\_name,double\_name,blacklist,whitelist):

for i in range(len(new)):

j=0

while j <len(new[i]):

if new[i][j] in double\_name or new[i][j] in single\_name:

t=j

j=j+1

m=0

p=0

while m<2 and j<len(new[i]):

if len(new[i][j])==1 and new[i][j] not in blacklist:

j=j+1

elif new[i][j] in whitelist:

p=1

m=m+1

new[i][t:j+1]=[new[i][t]+new[i][t+1]]

else:

break

m=m+1

j=j-1

if j-t==1 and p==0:

new[i][t:j+1]=new[i][t]+new[i][t+1]

if j-t==2 and p==0:

new[i][t:j+1]=new[i][t]+new[i][t+1]+new[i][t+2]

j=j-2

j=j+1

return new

new=final(new,single\_name,double\_name,blacklist,whitelist)

1. **Combine**

Since all the sentences have been cut into words, we have to combine them into a string preparing for the output and rectify. This is really easy.

The related function source code is shown as follows:

def combine(new):

s=""

for i in range(len(new)):

for j in range(len(new[i])):

s=s+new[i][j]+"|"

s=s+"\n"

return s

1. **Rules and Rectify**

We have prepared three kinds of rules for users to create to rectify the segmentation result. These rules can make one word into two parts, two parts into one word or move the segmentation symbol.

The related function source code is shown as follows:

def concat(old\_word,s):

if old\_word in s:

new\_word=""

for word in old\_word:

if word!="|":

new\_word+=word

s=s.replace(old\_word,new\_word)

return s

def divide(old\_word,position,s):

if old\_word in s:

new\_word=old\_word[:position]+"|"+old\_word[position:]

s=s.replace(old\_word,new\_word)

return s

def slide(old\_word,direction,length,s):

if old\_word in s:

if direction=="left":

length=-length

new\_word=""

for word in old\_word:

if word!="|":

new\_word+=word

n=old\_word.find("|")

if 0<n+length<len(new\_word):

new\_word=new\_word[:n+length]+"|"+new\_word[n+length:]

s=s.replace(old\_word,new\_word)

else:

return s

return s

1. **Demo and Testing Result**
2. **Testing**

We use an article from Expo website to test our program. The result is shown as follows:

世博网|10|月|31|日|消息|：|

展现|城市|生活|美好|前景|，|

开启|人类|文明|崭新|篇章|。|

中国|2010|年|上海|世界|博览会|闭幕式|31|日|晚|在|上海|世博|文化|中心|隆重|举行|。|

国务院|总理|温家宝|出席|闭幕式|并|宣布|上海|世博会|闭幕|。|

国际|展览局|主席|蓝峰|，|

来自|世界|各地|的|领导|人和|贵宾|出席|闭幕式|，|

共同|庆祝|上海|世博会|取得|圆满|成功|。|

上海|世博会|是|继|北京|奥运会|后|我国|举办|的|又|一|国际|盛会|，|

也是|第|一|次|在|发展中国家|举办|的|注册|类|世界|博览会|。|

本届|世博会|的|主题|是|“|城市|，|

让|生活|更|美好|”|。|

自|5|月|1|日|开幕|以来|，|

来自|246|个|国家|、|国际|组织|的|参展|方|，|

通过|展示|、|论坛|、|表演|等|形式|，|

一起|探讨|城市|未来|发展|前景|，|

共同|谱写|了|一|曲|人类|文明|和谐|共生|的|激情|乐章|，|

生动|诠释|了|“|理解|、|沟通|、|欢聚|、|合作|”|的|世博|理念|。|

浦江|两岸|灯火|璀璨|，|

申城|之|夜|流光溢彩|。|

造型|宛若|飞碟|的|上海|世博|文化|中心|内|，|

8000|多|名|观众|欢聚一堂|。|

当|时针|指向|20|时|，|

上|百|名|天真烂漫|的|少年|儿童|手|持|绚丽|的|花束|和|花环|走|上|舞台|中央|，|

组成|喜庆|的|花坛|。|

两侧|台阶|上|，|

200|多|位|少女|手|执|世博会|参展|国家|和|国际|组织|的|旗帜|，|

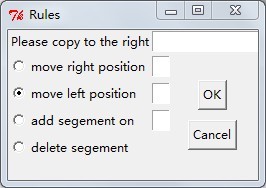
整齐|站立|。|

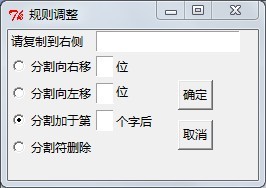
舞台|旁|，|

中华人民共和国|国旗|和|国际|展览局|旗|、|上海|世博会|旗|高高|飘扬|。|

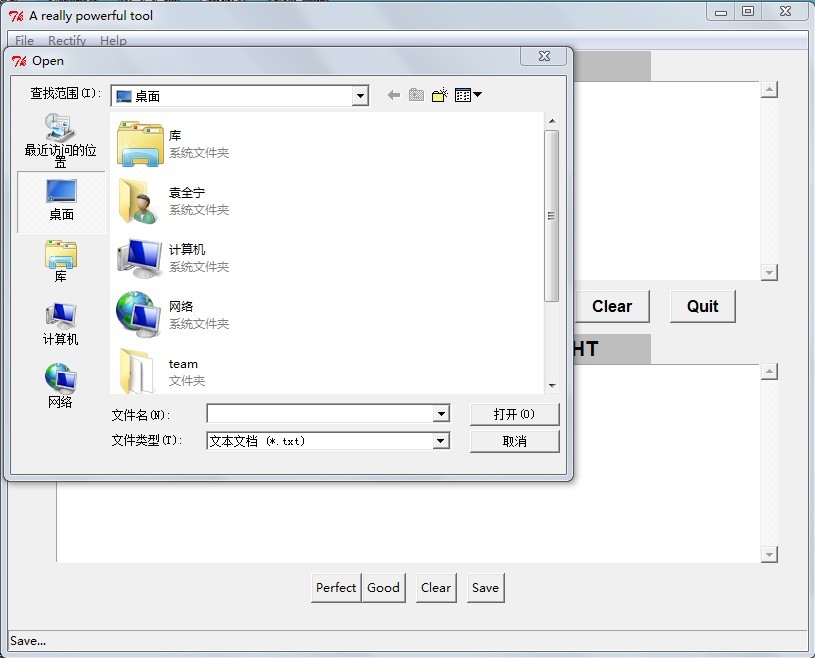
From this result, we can see that the names are be separated correctly such as “蓝峰” which doesn’t exist in the dictionary. So the name function does work and is really wonderful. Because of the incomplete lexicon, the “会旗” cannot be separated which leads a mistake. So we will keep refining the lexicon and make it complete.

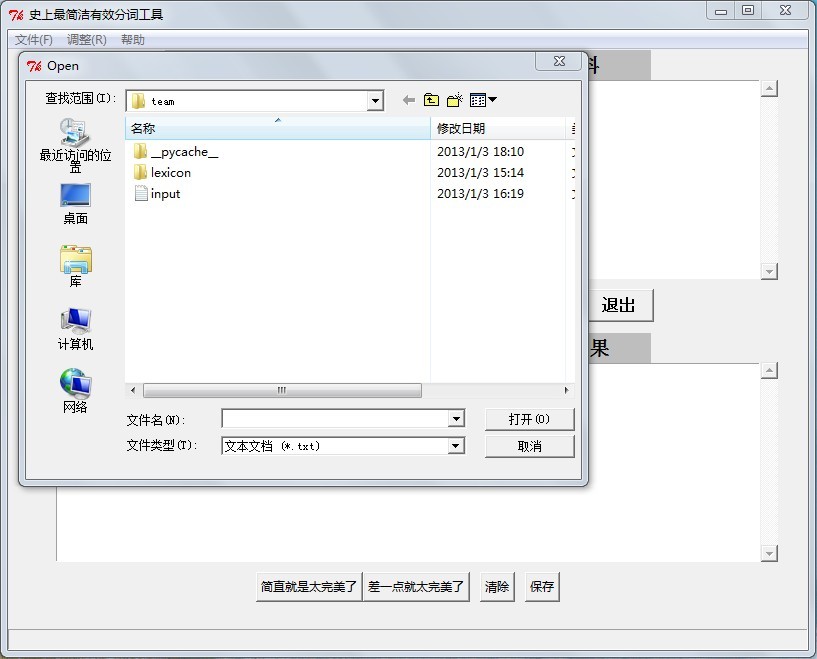
1. **Screenshots**





Add a rule



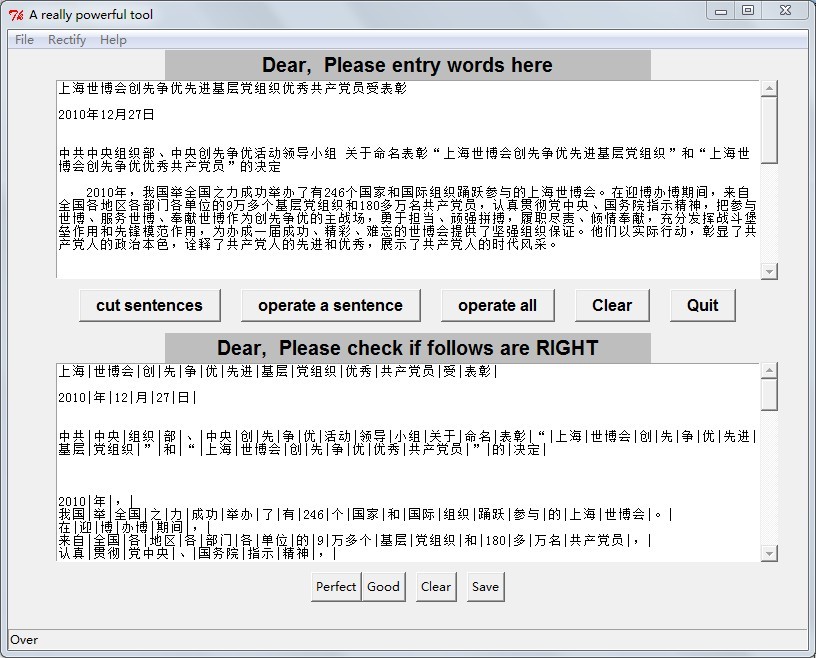


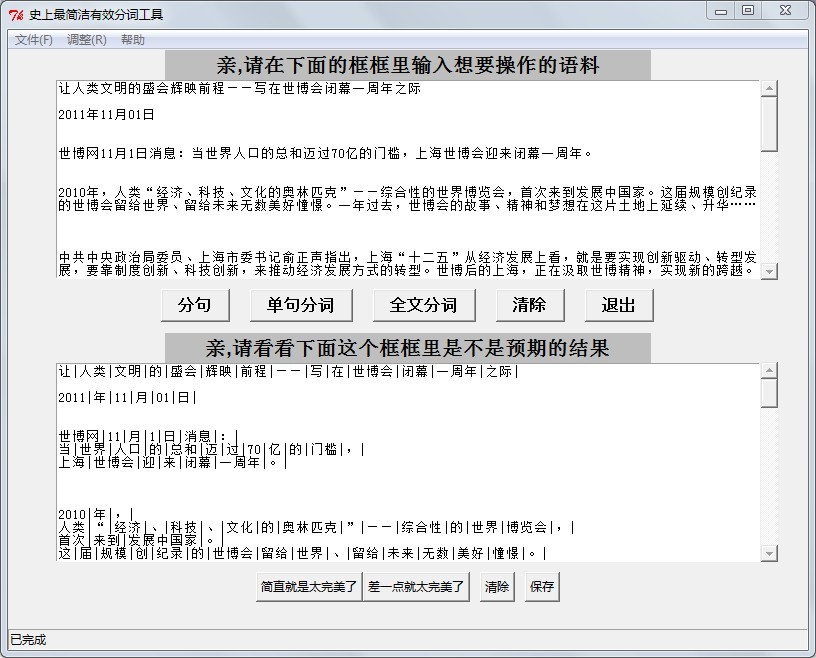
Load a file





The initial





The result

1. **Conclusion and Comments**

By three word segmentation procedures, we can get a good result and separate most of names right. This is what we most satisfied with. But how to improve the program to make it better is a little bit difficult. On one hand, a good lexicon is really important, so we have to keep refining it. On the other hand, we should consider the performance and try to make it run faster.

During making this program, we succeeded in almost everything except two things. One is multiprocessing; the other is packaging the program into exe file. Although we have learnt many knowledge from the Internet, that is not enough. When we tried to use multiprocessing to make our program faster, at the first time, we failed because we cannot terminate the new process which leaded the computer down; at the second time, we failed because we found that using multiprocessing is slower. So it shows that we still have much to learn.

1. **From Chen Zhen**

I’m the designer and the code writer of this program. This project really challenges my logical and algorithm designing ability which also improves my thinking method. At first, I thought name segmentation may be difficult. But at last, I just make it only with my brain. And as a team leader, l learn that distributing tasks to each team member and setting an appropriate deadline so that they will spend time working on the project is quite important. This team project is really meaningful.

1. **From Yuan Quanning**

In this project, I learn something important. In our team, we rely on each other and help each other. Though each of us has our independent task, we also solve some problems together. What's more, I practiced the knowledge that I learned in the class. We used many functions in the program, and I practiced using different of types of data. The most important is that I learned that we write a program is not only to solve a problem but also to let people use it conveniently. To think as a user is what we should notice and to be a good programmer is not to solve the problem but to serve the people.

1. **From Li Jianda**

In this team, I take charge of the interface part. By doing this, I learned not only the knowledge about Python Tkinter, but how to complete a program with a team as well. Solving a problem in team is nothing like doing it by oneself. In a team, communication is the most important thing. Without communication, we can’t combine what we have done respectively to a whole thing. I really enjoy the time working with my teammates. Thank you all for giving me such a wonderful experience.

1. **From Liu Baichen**

In the team project, I do the work about lexicon. The lexicon is easy to get but hard to optimize. I spend most time in optimizing the lexicon. I write some program to delete space blank, English words, numbers and punctuations. And then I check the words one after another to see whether it is appropriate. For instance, “踢球” should be separate, and “诚实守信” should be separate as “诚实” and “守信” . Maybe it is a physical work. But it is really tiring. Everyone has made a contribution to our team project. We are all really nice and we are confident in our team project!