Final Report

Team Name：Three's Company Team Leader: Jia Xiaosong (516030910548)

Date:2016.12.30 Team Member: Chai Zhenghao(516030910543)
Li Mingze(516030910550)

1. Prototype System Introduction

1.1 Functions
This program's function is for Chinese word segmentation (Project B).
1.2 Running Environment
Windows 10
1.3 Developing Environment
Python 3.5
Pycharm

2 Task Allocation

Jia Xiaosong: User Interface Component: Design and implement a user interface component for prototype system using TkInter module. Integrate the word segmentation and lexicon component developed by other team members with the user interface. Pack the project and write the readme.

Chai Zhenghao: Lexicon Component: Design and implement a lexicon component. The lexicon includes word entry, index, and other necessary information. The words in the lexicon can be modified or deleted. In addition, new words can be added to the lexicon.

Li Mingze: Word Segmentation Component: Design a word segmentation algorithm. It can segment sentences and words. During the word segmentation process, you can utilize lexicons,

3.Description

3.1 Algorithm Description:

Our algorithm has 3 parts.One part is the sentence segmentation part,one is forward segmentation part,and the last is the inverse segmentation part.
1.Sentence segmentation part:

```
#-----------------seg_sentence----------------------------------------------------
def find_sentence(section,punctuation,current_sentence):
    section=section.replace(" ","")
    section=section.replace("\n","")
    section=section.replace("“","")
    section=section.replace("”","")
    list_sec=list(section)
    if list_sec[-1] not in punctuation:
        section+='。'
        return find_sentence(section,punctuation,current_sentence)
    else:
        for i in range(len(list_sec)):
            if list_sec[i] in punctuation:
                for num in range(i+1):
                    current_sentence+=list_sec[num]
                section = ''
                for num in range(i+1,len(list_sec)):
                    section+=list_sec[num]
                break
        return[section,current_sentence]
def seg_sentence(section):
    try:
        punctuation=[', ','。','? ','?','!','! ',',','……','、']
        well_seg=''
        while section!='':
            well_seg+=find_sentence(section,punctuation,'')[1]+'\n'
            section=find_sentence(section,punctuation,'')[0]
        return well_seg
    except:
        return ''
```

The sentence is segmented by punctuation “,” 、“。” 、 “？” 、 “……” 、 “!”.Since sometimes Chinese “”、 ““’will influence the segmentation,we first delete them from the section.And ‘\n’ and ‘ ’ will make influence ,too,we delete them all.Then if the final character is not a punctuation,we add a ‘。’.

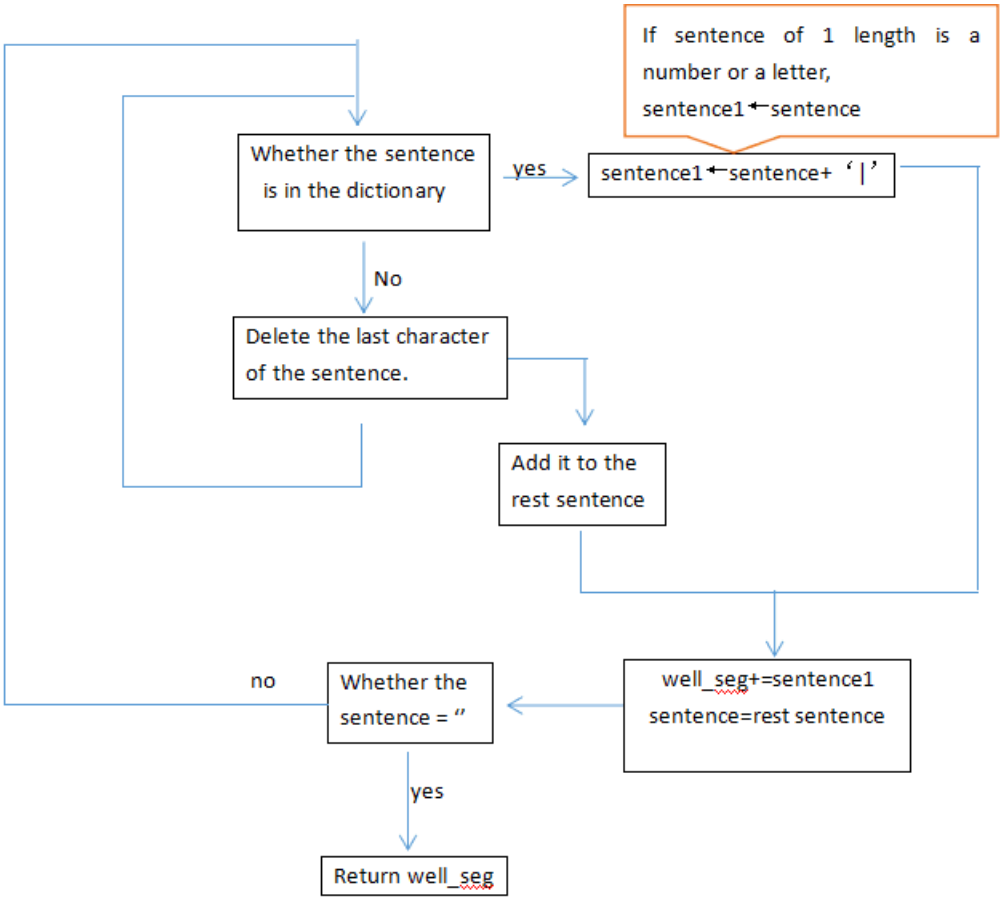
We use find_sentence to segregate the section by the punctuation above.Then we use seg_sentence to store the segmented sentence and delete the segmented sentence from the section.When the section has none sentence in it, the function is over and return the well segmented section.

2.Forward segmentation part：

```
#---------------forward_seg---------------------------------------------------------
def find_word_forward(sentence,dictionary,rest_sentence):
    list_sen=list(sentence)
    len1=len(list_sen)
    if len1>10:
        len1=10
    if len1==1:
        if sentence.isdigit()==True:
            return[sentence,rest_sentence]
        elif ord(sentence)>64 and ord(sentence)<91:
            return[sentence,rest_sentence]
        elif ord(sentence)>96 and ord(sentence)<123:
            return[sentence,rest_sentence]
        else:
            return['|'+sentence+'|',rest_sentence]
    else:
        if sentence in dictionary:
            return[sentence+'|',rest_sentence]
        else:
            list_sen1=list_sen[0:len1-1]
            rest_sentence=''.join(list_sen[len1-1:])+rest_sentence
            sentence1=''
            for i in range(len1-1):
                sentence1+=list_sen1[i]
            return find_word_forward(sentence1,dictionary,rest_sentence)
def seg_forward(sentence,dictionary):
    well_seg=''
    while sentence!='':
        well_seg+=find_word_forward(sentence,dictionary,'')[0]
        sentence=find_word_forward(sentence,dictionary,'')[1]
    well_seg=re.sub('\|+','|',well_seg)
    return well_seg
```

The program is based on dictionary.The first function is searching the word in our dictionary.If the segmented word is not in the dictionary,delete the last character and add it into the rest sentence.Then we see whether the new word is in the dictionary,if not,repeat this operation until the new word is in the dictionary.If we can find the word in the dictionary,then we return the ‘word+ ‘|”,and delete it from the sentence .If the number of segmented words is equal to the number of the word in the original sentence,we succeed.We make the max number of characters is 10,if the length of the sentence is more than 10,cut it and take out 10 characters to segment.This will reduce the time effectively.Sometimes we will meet numbers and English letters,so we design a discussion not to segment them to make them an entirety.

Here is the flow chart of the forward segmentation part:



3.Inverse segmentation part

```
#--------------inverse_seg-----------------------------------------------------
def find_word_inverse(sentence,dict_inverse,rest_sentence):
    list_sen=list(sentence)
    len1=len(list_sen)
    if len1>10:
        len1=10
    if len1==1:
        if sentence.isdigit()==True:
            return[sentence,rest_sentence]
        elif ord(sentence)>64 and ord(sentence)<91:
            return[sentence,rest_sentence]
        elif ord(sentence)>96 and ord(sentence)<123:
            return[sentence,rest_sentence]
        else:
            return['|'+sentence+'|',rest_sentence]
    else:
        if sentence in dict_inverse:
            return['|'+sentence+'|',rest_sentence]
        else:
            list_sen1=list_sen[0:len1-1]
            rest_sentence=''.join(list_sen[len1-1:])+rest_sentence
            sentence1=''
            for i in range(len1-1):
                sentence1+=list_sen1[i]
            return find_word_inverse(sentence1,dict_inverse,rest_sentence)
def seg_inverse(sentence,dict_inverse):
    sentence = sentence[::-1]
    well_seg=''
    while sentence!='':
        well_seg+=find_word_inverse(sentence,dict_inverse,'')[0]
        sentence=find_word_inverse(sentence,dict_inverse,'')[1]
    well_seg=re.sub('\|+','|',well_seg)
    return well_seg[::-1]
```

The idea is to take the inversion of the sentence and use the forward segmentation

part.Finally,take the inversion of the segmented sentence to get the final result.This way is more accurate then forward segmentation owing to the Chinese grammar.

3.2 lexicon description

When dealing with the lexicon component, we have two parts to consider: the process of the lexicon and the collection of words for the establishment of the lexicon.

1.the process of the lexicon

The process of the lexicon includes four parts: loading lexicon, adding words into lexicon, deleting words from lexicon and modifying words in lexicon.

When load the lexicon from the lexicon txt file, I use dictionary data type in Python to store the data with the words as keys and their frequencies in corpus as values. When loading 157212 words from the file, the average time it takes is about 0.62 sec. We load the lexicon when the program initiates, so it will take little time to search words in the lexicon every time.

In the add/delete/modify components, every time after the program finishes processing the words users input, it will print confirmation messages to ensure the results are in accordance with users' expectation or warn the users about wrong inputs. For example, if the words users want to add have been in the lexicon, it will output 'The word has been in the dict.'

2.the collection of the words

The number of the Chinese words are so large that we can't collect them by manual labor. To deal with those problem, I search on the Internet and take the lexicon of Sougou as a basic part of lexicon which contains over 156000 words, enough to satisfy general usage of the word segmentation algorithm.

But considering the lack of prop nouns about G20 in the basic lexicon, I use another method to filter words from the G20 corpus in order to find prop nouns.

First, I use Python spider to get 832 pages of texts in the website http://g20chn.org/ as the original corpus.

Second, I use 2 methods to filter words from the corpus.

The first method is using mutual information to judge two or more Chinese characters are more likely to be seen together in the corpus.. That is

$I(A,B) = \log(P(A,B)/(P(A)*P(B)))$

where A B are Chinese characters and P(A) P(B) are their frequencies in the corpus.

The large the value is, the more likely it is to be a word.

The second method is using the entropy of prefixes and suffixes of the words to

judge whether the prefix or suffix of a possible word is fixed or not.. That is

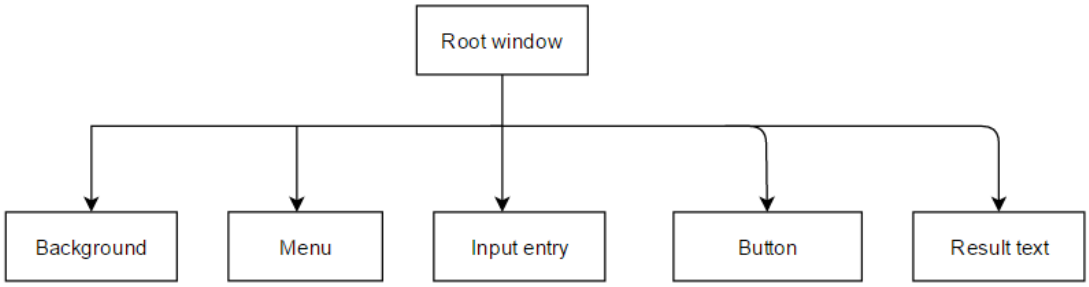
H(x) = -∑P(xi)log(2,P(xi)) where xi is prefixes or suffixes of an possible Chinese words, and P(xi) is its frequency of being the prefixes or suffixes of the words.

If the prefix or suffix is fixed, it is more likely that its prefix or suffix is also a part of the word.

After filtering by these standards, we get 3603 possible words. After deleting words that already in the basic lexicon and human filtering, finally we get 423 new words from the corpus in the website, including two characters words 荻浦、电商、乌镇 etc, three characters words 奥巴马、埃博拉、片儿川 etc and words with more characters 晴好雨奇、影子银行、洛斯卡沃斯 etc.

After adding the new words generating from corpus into the basic lexicon, I get the final edition of the lexicon.

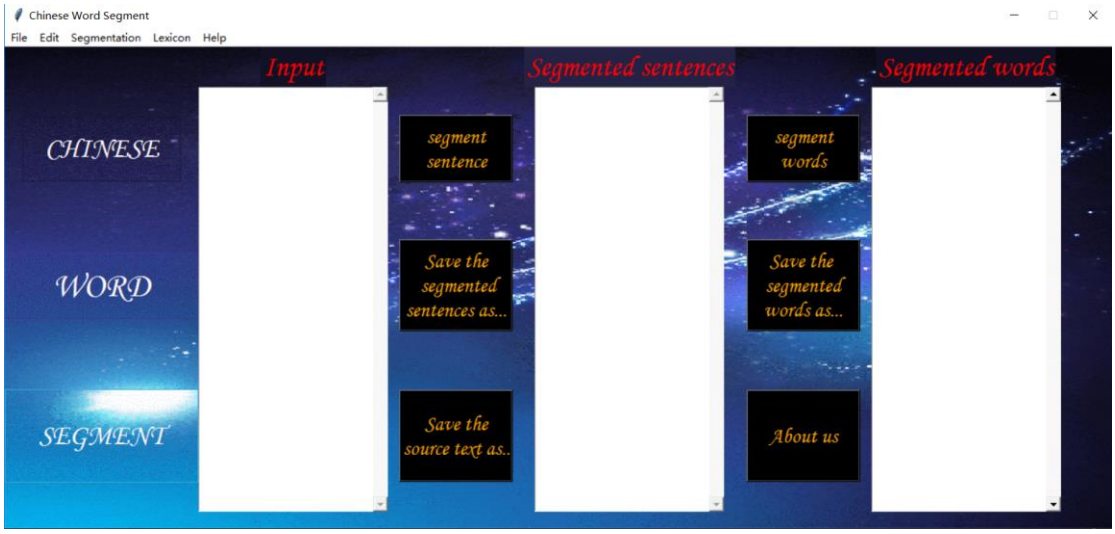
3.3 User Interface Component



The realization of the user interface is divided into five parts: Background, Menu, Input Entry, Button, Result text. Background contains the background pictures and the size of the window. Menu contains file menu, edit menu, segmentation menu, lexicon menu, help menu. The others are about the main function of that software.

Screenshots:

2.1 main window:



2.2 source code and explanation

A. root window and module

This part is to form the root window. Besides, it create the lexicon.

```
#Main Window
root = Tk()
root.title('Chinese Word Segment')
root.geometry('1270x550')
root.resizable(False, False)
path = os.getcwd()
lex_name = '{}/doc/word_dict.txt'.format(path)
lex = dict_process.load_dict(lex_name)
lex_inverse = {}
path = os.getcwd()
for key in lex:
    lex_inverse[key[::-1]] = lex[key]
algorithm = 'inverse'
```

B. Background

This part is to set the background picture and the logo.

```
# Background
background = PhotoImage(file='{}/background/background.gif'.format(path))
pic1 = PhotoImage(file='{}/background/logo1.gif'.format(path))
pic2 = PhotoImage(file='{}/background/logo2.gif'.format(path))
pic3 = PhotoImage(file='{}/background/logo3.gif'.format(path))
background_label = Label(root, image=background)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
logo1 = Label(root, text='CHINESE', relief=SUNKEN, borderwidth=0,  bg='#1B1A3B', image=pic1, compound='center',
              fg='Snow', font=('Monotype Corsiva', 25, 'normal')).grid(row=1, column=1)
logo2 = Label(root, text='WORD', relief=SUNKEN, borderwidth=0, bg='#373A95', image=pic2, compound='center', fg='Snow',
              font=('Monotype Corsiva', 30, 'normal')).grid(row=2, column=1)
logo3 = Label(root, text='SEGMENT', relief=SUNKEN, borderwidth=0, bg='#33AACC', image=pic3, compound='center',
              fg='Snow', font=('Monotype Corsiva', 25, 'normal')).grid(row=3, column=1)
```

C. Main window

(1) Input entry

This part is to create the input entry and output the source text.

```
# Input entry
def save_source_as():
    content = source.get(1.0, END)
    filetypes = [('Text Files', '*.txt', 'TEXT')]
    fobj = asksaveasfile(filetype=filetypes, initialfil="未命名.txt", defaultextension=".txt",initialdir='{}/save'.format(path))
    if fobj:
        fobj.write(content)


Label(root, text='Input', bg='#131125', fg='red', font=('Monotype Corsiva', 25, 'normal')).grid(row=0, column=2)
source = ScrolledText(root, width=28, height=37)
source.grid(row=1, column=2, rowspan=3)
Button(root, text='Save the\n source text as...', bg='black', fg='orange', width=10, height=3,
       font=('Monotype Corsiva', 18, 'normal'), command=save_source_as).grid(row=3, column=4, padx=13, sticky=E)
```

(2) Segment sentence part

This part is to create the button and output text. Besides, it defines the functions to segment the sentence.

```
# Segment sentence UI
def seg_sent():
    sent = source.get(0.0, END)
    sent = sent[0:-1]
    well_seg = segmentation.seg_sentence(sent)
    if well_seg == '':
        showinfo(title='Segment sentences', message='The input text is empty!')
    else:
        sen_seged.delete(1.0, END)
        sen_seged.insert(END, well_seg)


def save_sent_as():
    content = sen_seged.get(1.0, END)
    filetypes = [('Text Files', '*.txt', 'TEXT')]
    fobj = asksaveasfile(filetype=filetypes, initialfil="未命名.txt", defaultextension=".txt", initialdir='{}/save'.format(path))
    if fobj:
        fobj.write(content)

Label(root, text='Segmented sentences', bg='#242245', fg='red', font=('Monotype Corsiva', 25, 'normal')).grid(row=0, column=5)
sen_seged = ScrolledText(root, width=28, height=37)
sen_seged.grid(row=1, column=5, rowspan=3)
```

```
Button(root, text='segment\nsentence', bg='black', fg='orange',width=10, height=2,
       font=('Monotype Corsiva', 18, 'normal'), command=seg_sent).grid(row=1, column=4, padx=13,sticky=E)
Button(root, text='Save the\n segmented\nsentences as...', bg='black', fg='orange', width=10, height=3,
       font=('Monotype Corsiva', 18, 'normal'), command=save_sent_as).grid(row=2, column=4, padx=13, sticky=E)
```

（3） Segment words part

This part is to create the segment, output text and about us three buttons. Besides, it defines the functions to segment the words.

```
# Segment words UI
def seg_words():
    global lex
    global lex_inverse
    sent = sen_seged.get(0.0, END)
    sent = sent[0:-1]
    if algorithm == 'inverse':
        well_sent = segmentation.seg_inverse(sent, lex_inverse)
    elif algorithm == 'forward':
        well_sent = segmentation.seg_forward(sent, lex)
    if well_sent == '':
        showinfo(title='Segment words', message='The segmented sentences text is empty!')
    else:
        word_seged.delete(1.0, END)
        word_seged.insert(END, well_sent)

def save_word_as():
    content = word_seged.get(1.0, END)
    filetypes = [('Text Files', '*.txt', 'TEXT')]
    fobj = asksaveasfile(filetype=filetypes, initialfil="未命名.txt", defaultextension=".txt",initialdir='{}/save'.format(path))
    if fobj:
        fobj.write(content)
```

```
def copyright():
    showinfo(title='Copyright', message='''Author: Xiaosong Jia, Mingge Li,Zhenghao Chai.
    \nCopyright © 2016 Three's Company.All rights reserved.''')
    return
Label(root, text='Segmented words', bg='#131125', fg='red',
      font=('Monotype Corsiva', 25, 'normal')).grid(row=0, column=8)
Button(root, text='segment\nwords', bg='black', fg='orange', width=10, height=2,
       font=('Monotype Corsiva', 18, 'normal'),command=seg_words).grid(row=1, column=6, padx=13)
Button(root, text='Save the \nsegmented\nwords as...', bg='black', width=10, height=3, fg='orange',
       font=('Monotype Corsiva', 18, 'normal'),command=save_word_as).grid(row=2, column=6, padx=13)
Button(root, text='About us', bg='black', width=10, height=3, fg='orange',
       font=('Monotype Corsiva', 18, 'normal'),command=copyright).grid(row=3, column=6, padx=13)
word_seged = ScrolledText(root,  width=28, height=37)
word_seged.grid(row=1, column=8, rowspan=3)
```

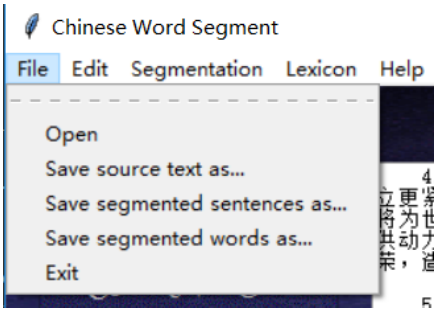
D. Menu

(1) main menu

This is the root of all the submenu.

```
#Menu
main_menu = Menu(root)
root['menu'] = main_menu
```

(2) file menu

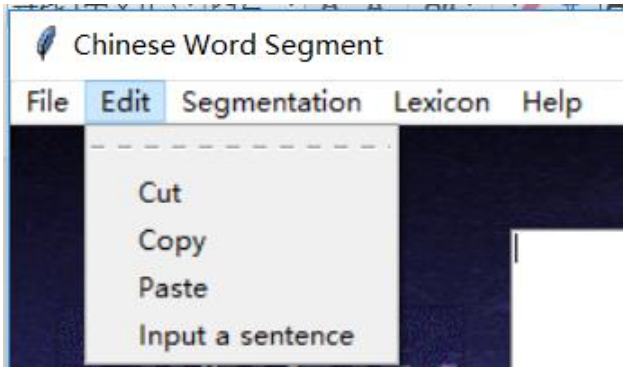


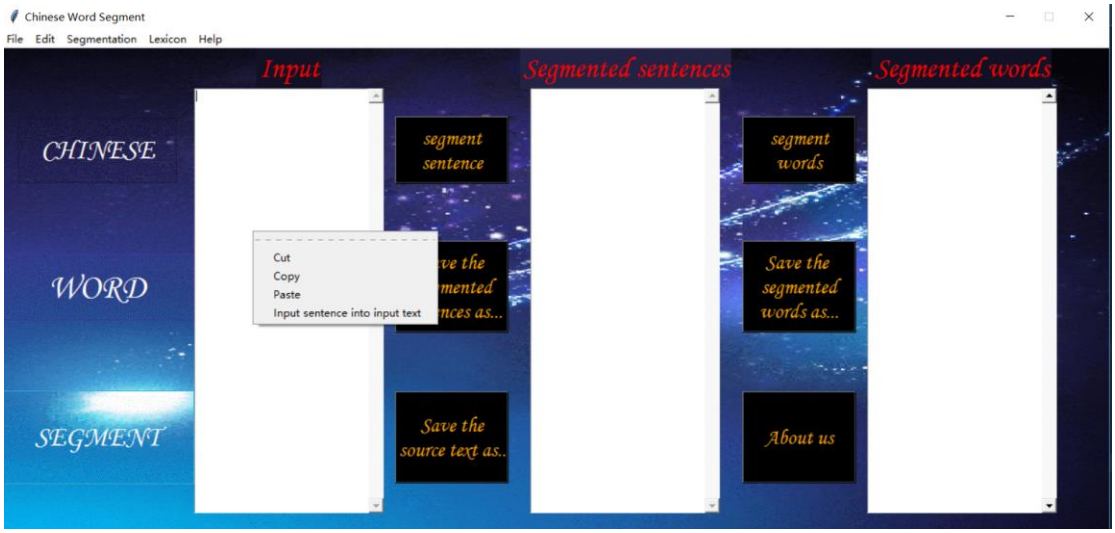
This menu includes the functions: open, save and exit.

```
#File menu
File_menu = Menu(main_menu)
main_menu.add_cascade(label='File', menu=File_menu)

def open_a_file():
    filetypes = [('Text Files', '*.txt', 'TEXT')]
    filename = askopenfilename(filetype=filetypes,initialdir='{}/doc'.format(path))
    if filename == '':
        filename = None
    else:
        try:
            with open(filename, 'r') as f:
                text = f.read()
                source.delete(1.0, END)
                source.insert(1.0, text)
        except:
            showinfo(title='Open file', message='Fail to open the file(Tips:encoding should be utf8)')
```

```
def Exit():
    root.destroy()

File_menu.add_command(label='Open', command=open_a_file)
File_menu.add_command(label='Save source text as...', command=save_source_as)
File_menu.add_command(label='Save segmented sentences as...', command=save_sent_as)
File_menu.add_command(label='Save segmented words as...', command=save_word_as)
File_menu.add_command(label='Exit', command=Exit)
```

(3) edit menu





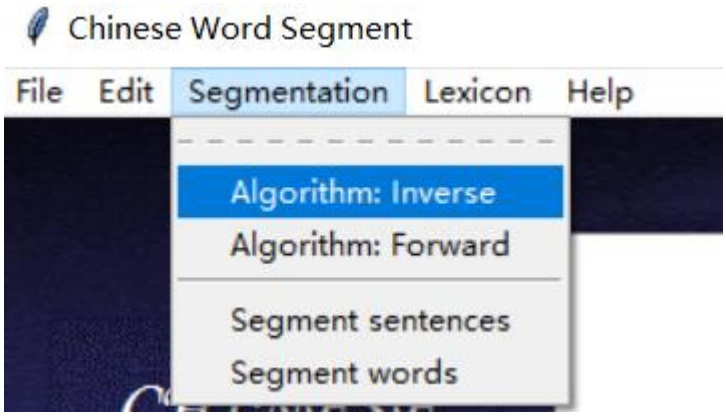
This menu includes the functions: cut, copy, paste and input sentence. Besides, it also creates right-hand button pop menu.

```
#Edit menu
Edit_menu = Menu(main_menu)
main_menu.add_cascade(label='Edit', menu=Edit_menu)


# Let edit menu can be used to all text
def seteditor(event=None):
    global editor
    editor = event.widget
root.bind_class("Text", "<Any-KeyPress>", seteditor, add=True)
root.bind_class("Text", "<Button-1>", seteditor, add=True)


def cut():
    try:
        editor.event_generate('<<Cut>>')
        return
    except:
        pass
```

```
def copy():
    try:
        editor.event_generate('<<Copy>>')
        return
    except:
        pass
def paste():
    try:
        editor.event_generate('<<Paste>>')
        return
    except:
        pass


def input_sentence():
    sentence=simpledialog.askstring('input sentence', 'Please input the sentence')
    if isinstance(sentence, str):
        source.insert(END, sentence)
    return

Edit_menu.add_command(label='Cut', command=cut)
Edit_menu.add_command(label='Copy', command=copy)
Edit_menu.add_command(label='Paste', command=paste)
Edit_menu.add_command(label='Input a sentence', command=input_sentence)
```

```
#Popupmenu
def popmenu(event):
        Edit_menu.post(event.x_root, event.y_root)
root.bind('<Button-3>', popmenu)
```

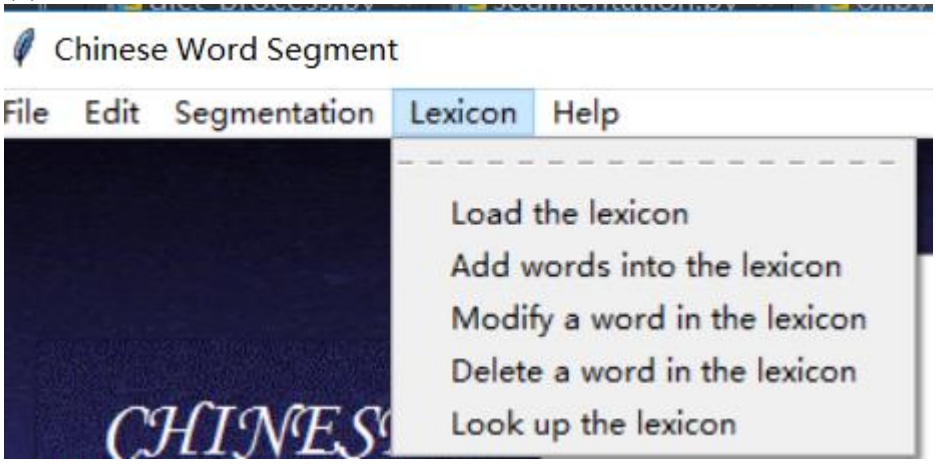
(4) segment menu



This menu contains two part: choose the algorithm and choose the function.

```
#Segmentation menu
Seg_menu=Menu(main_menu)
main_menu.add_cascade(label='Segmentation', menu=Seg_menu)


def inverse():
    global algorithm
    algorithm = 'inverse'
def forward():
    global algorithm
    algorithm = 'forward'
Seg_menu.add_radiobutton(label='Algorithm: Inverse', command=inverse)
Seg_menu.add_radiobutton(label='Algorithm: Forward ', command=forward)
Seg_menu.add_separator()
Seg_menu.add_command(label='Segment sentences', command=seg_sent)
Seg_menu.add_command(label='Segment words', command=seg_words)
```

(5) Lexicon menu



This menu contains the function about lexicon.

```
#Lexicon menu
Lex_menu = Menu(main_menu)
main_menu.add_cascade(label='Lexicon', menu=Lex_menu)


def load_lex():
    global lex
    global lex_name
    global lex_inverse
    try:
        lex_new_name = askopenfilename()
        lex = dict_process.load_dict(lex_new_name)
        lex_inverse = {}
        for key in lex:
            lex_inverse[key[::-1]] = lex[key]
        lex_name = lex_new_name
        showinfo(title='Load the lexicon', message='Load the lexicon successfully!')
    except:
        showinfo(title='Load the lexicon', message='Fail to load the lexicon!')
    return
```

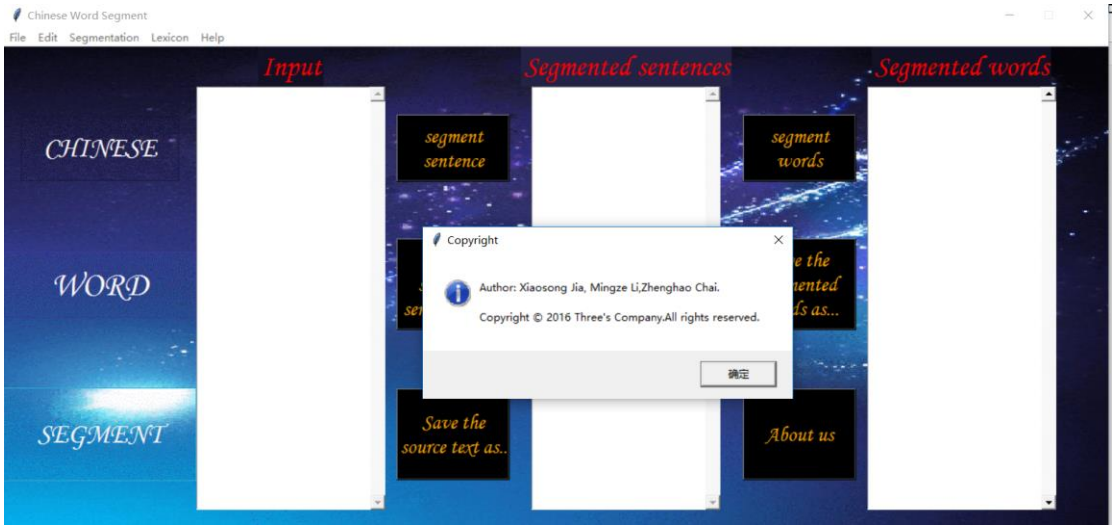
```
def add_word():
    global lex
    global lex_name
    global lex_inverse
    tip='''Input the words you want to add. Format:(words only) or (words with frequency '一个 818357166') '''
    word = simpledialog.askstring('Add words', tip)
    if isinstance(word, str):
        result, lex = dict_process.add_words(word, lex, lex_name)
        lex_inverse = {}
        for key in lex:
            lex_inverse[key[::-1]] = lex[key]
        showinfo(message=result, title='Add words')
        return
    else:
        return
```

```
def modify_word():
    global lex
    global lex_name
    global lex_inverse
    old = simpledialog.askstring('Old word', 'Please input the word you want to modify')
    if isinstance(old, str):
        new = simpledialog.askstring('New word', 'Please input the word you want to add')
        if isinstance(old, str) and isinstance(new, str):
            result, lex = dict_process.revise_words(old, new, lex, lex_name)
            showinfo(message=result, title='Modify words')
            lex_inverse = {}
            for key in lex:
                lex_inverse[key[::-1]] = lex[key]
    else:
        return
```

```
def del_word():
    global lex
    global lex_name
    global lex_inverse
    tip = '''format of the words you want to delete: '大家 你好' (words separated by space) '''
    word = simpledialog.askstring('Delete words', tip)
    if isinstance(word, str):
        result, lex = dict_process.delete_words(word, lex, lex_name)
        showinfo(message=result, title='Add words')
        lex_inverse = {}
        for key in lex:
            lex_inverse[key[::-1]] = lex[key]
    else:
        return


def look_up():
    global lex_name
    os.system(lex_name)
    return
```

```
def look_up():
    global lex_name
    os.system(lex_name)
    return

Lex_menu.add_command(label='Load the lexicon', command=load_lex)
Lex_menu.add_command(label='Add words into the lexicon', command=add_word)
Lex_menu.add_command(label='Modify a word in the lexicon', command=modify_word)
Lex_menu.add_command(label='Delete a word in the lexicon', command=del_word)
Lex_menu.add_command(label='Look up the lexicon', command=look_up)


#Help menu
Help_menu = Menu(main_menu)
main_menu.add_cascade(label='Help', menu=Help_menu)
```

(6) Help menu

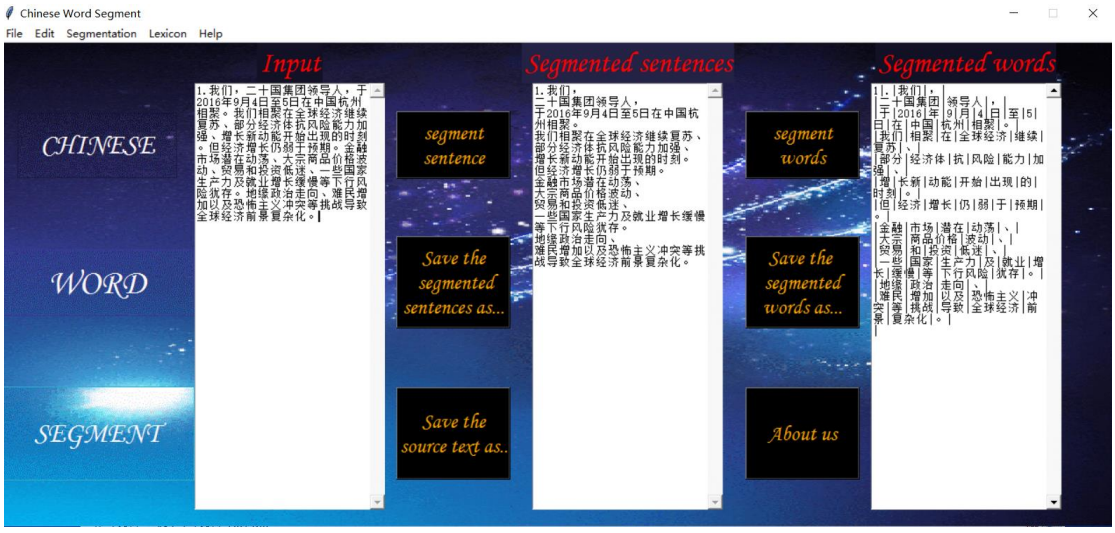


This part is about copyright and instruction.

```
#Help menu
Help_menu = Menu(main_menu)
main_menu.add_cascade(label='Help', menu=Help_menu)


def instruction():
    os.system('readme.txt')
    return


Help_menu.add_command(label='Instruction', command=instruction)
Help_menu.add_command(label='Copyright', command=copyright)

root.mainloop()
```

4. Testing Procedure, Data and Result

There's an example:

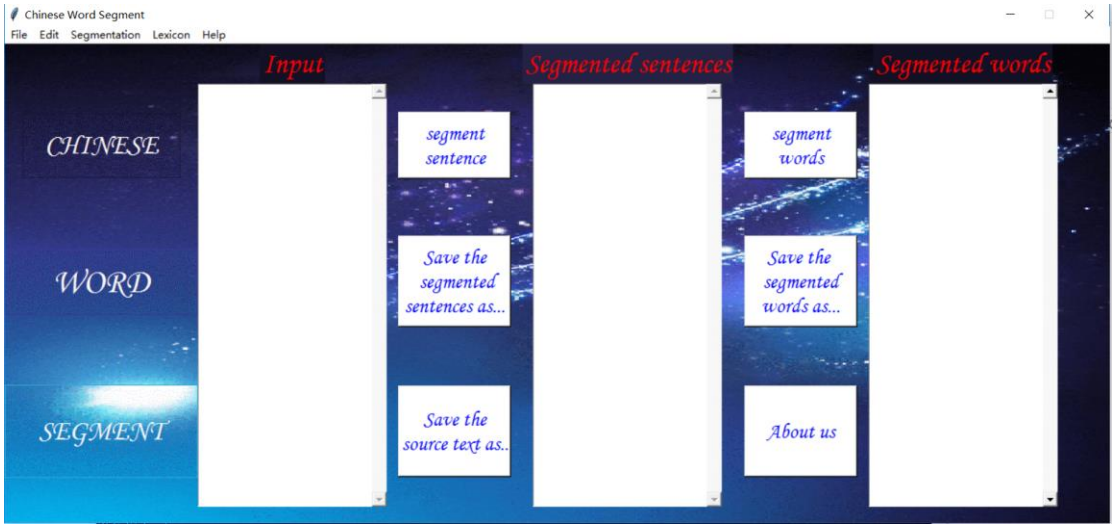


That's the result of the segmented sentences.

That’s the result of the segmented words

We can see, most of them are proper, but there are still some mistakes.
Actually, when we segment a text contains about 13000 words, it uses 8 seconds.

With the advice of the teacher, I change the color of the button to be white instead of black. It’s more clear than before.



5.conclusion

Li Mingze:In this project my duty is to build the main algorithm.At first I think my mission is too easy.But as soon as I took over it,I'm wrong.But after I looked through lots of blogs and thought the idea over and over again,I finally solved it in the

main.Though it has lots of problems,my partners are very generous and whenever I come across difficulty, they are there to help me.I think I can do better and optimize my algorithm.I am very glad to have this opportunity to cooperate with my classmates.Thank you, teachers and teaching assistants.

Chai Zhenghao: In this program, my work is to create a lexicon in accord with the word segmentation algorithm. During the process of working, I have learnt many useful things, including many kinds of word segmentation algorithms, configuration and establishment of network spider in Python and capacity of modular programming. Most importantly, I have understood how to cooperate with other people to work on a project, which will be very useful in the future.

Chai Zhenghao: In this program, my work is to create a lexicon in accord with the word segmentation algorithm. During the process of working, I have learnt many useful things, including many kinds of word segmentation algorithms, configuration and establishment of network spider in Python and capacity of modular programming. Most importantly, I have understood how to cooperate with other people to work on a project, which will be very useful in the future.

Jia Xiaosong: In this project, my work is to design the user interface. It is a new challenge for me since I have never done that. To realize it, I learn from the instruction of the tkinter. Besides, I also integrate other components, write the readme(instruction), pack it into several folders and a steup.exe.

As the leader of the team, I assign tasks for them and hold meetings. I think the coordination work is really important to complete a project efficiently. Luckily, my work is UI. So I can know everyone’s work easily and tell them what to do next.

From this project, I learn a lot about software project and how to work as a team to complete a project. I will strive for excellence in my study process in the future.