Final Report

Team Name:Pyson

Team Leader:ZhouBen

Date: 2015-12-24

1.Prototype System Introduction

1.1 Introduction

In this project,we have written a Chinese segmentation prototype system including a graphic user interface.

The system can:

1.Sentence segmentation:

Segment Chinese articles into sentences

2.Word segmentation:

Segment sentences into words.

3.Editing:

Use the text region like a simple notepad.

4.Sentence selection:

Choose some of sentences to do word segmentation instead of all.

5.One-key segmentation:

Apply word segmentation directly to whole paragraph without clicking sentence segmentation.

6.File export

Choose a file and directly export the word segmentation result of the whole file to the place you like.

7.Intelligent segmentation

Including the identification of quotation,terminology and special situations.Switchable.

8.Data inspection and modification

See the lexicon of the system and some special rule settings.All modifiable.

1.2 Running Environment

Windows 7 or later

Mac OSX or later

（此处应有截图）

1.3 Developing Environment

PyScripter 2.6.0

PyCharm 4.5.4

Github

2.Task Allocation

Yuan Meng:

GUI design and implementation

Hu Guo Shen:

Core algorithm and file manipulation

Zhou Ben:

Overall design

User Input processing

Testing

**In the view of file:**

Yuan Meng:

Interface part of *view.py*

Hu Guo Shen:

package *kernel*

Zhou Ben:  
 *test.py*

*controller.py*

User input processing part of v*iew.py*

**In the view of MVC:**

Yuan Meng: View

Hu Guo Shen: Model

Zhou Ben: Controller

(Issues about MVC are discussed in section 3)

3.System Architecture

3.1 Introduction

We develop this prototype system under the guiding ideology of MVC.

MVC is a classic design pattern firstly used in user interface design,but now its idea has been used in a much wider range.MVC design pattern decouples different functions in the system.Its clear separation not only improves the readability and maintainability of the system but also forms a duty division naturally.

MVC decomposes a typical system into three parts:model,view and controller.



3.2 Model

Model is the core of the system encapsulating the essential data and service logic.

In our case,data and service logic denote the lexicon and the algorithm used to perform segmentation.

In the view of code,the package *kernel* in our source code is responsible for the model part including:

1.*\_\_init\_\_.py:*

Basic classes definition.

2.*segmentation\_by\_retrieve.py:*

Segmentation involves special cases which need processing at the very first step, including quotation identification,terminology and particular cases.

3.*mi.py,dts\_calculate.py,judge.py:*

Where typical steps of word segmentation is done.

4.*test.py*

Test logic and implementation

5.some txt files

Includes lexicon,corpora,test answers and so on.

The detail of algorithm is discussed in section 4.

3.3 View

View is the manager of GUI which enables user to input as well as demonstrate results to them.

In our code,the file *view.py*contains this part,in which we use python standard library tkinter to construct interface.

User permission:

Interface design:

Snapshots:(暂缺，等全做完了补上)

3.4 Controller

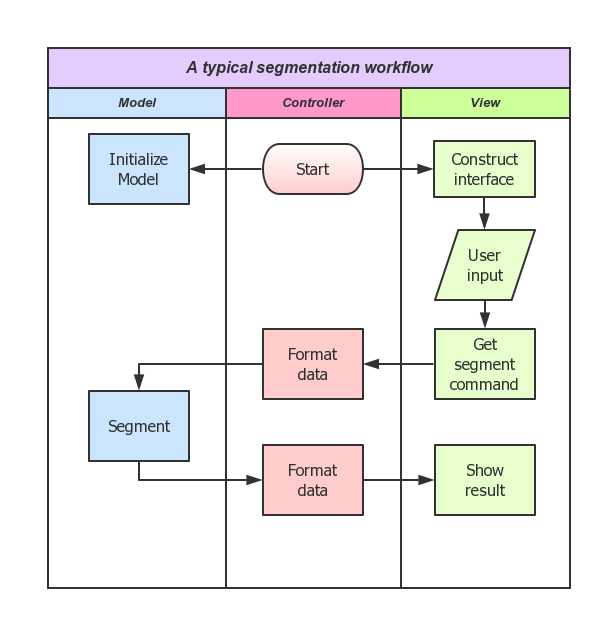
Controller is the bridge between view and model.It interprets the inputs from view from human-compatible content into machine-compatible content which is a call to a particular function of model in most cases.When model returns a value,it need to reformat it into human-compatible stuff.

In our program,*controller.py* implements main of the controller.But because of the character of tkinter,there is a tight coupling between the user input processing and interface constructing,resulting in that a part of *view.py* also works for controller.

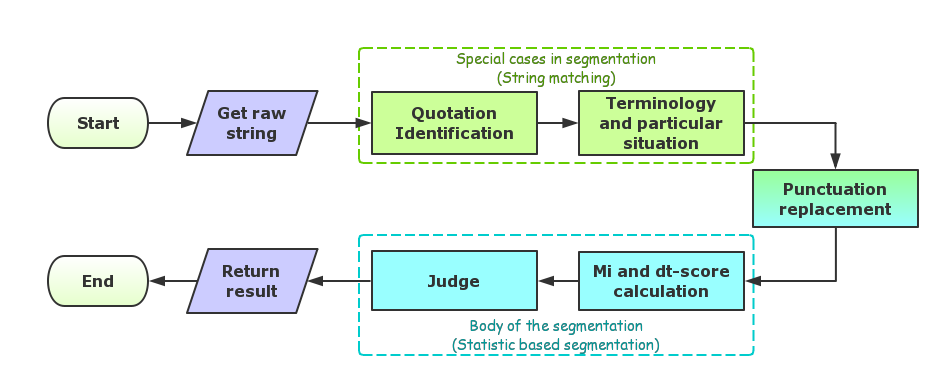
Controller is a rather important part which controls the path on which the system works.

But in our case,the necessary communication between view and model is relatively simple,so it only serves as a formatter and a middle station between view and model.

3.5 Workflow

A typical segmentation workflow is shown below:

4.Algorithm Description

4.1 Overview

4.2 Sentence segmentation

The sentence segmentation can cut the whole string into several sentences according to the sentence segment punctuations. The sentence segment punctuations are stored in the file *punctuation\_standard\_file.txt.* During the search, delete the line break “\n”, and do the separation when searching the sentence segment punctuations.

4.3 Word segmentation

Word segmentation contains these procedures: retrieve, punctuation separation, dtscore calculation, mi calculation, judging and combination.

4.3.1 Special Rule

Special Rule part contains the retrieve process for quotation marks and book marks, terminology and particular situations.

For quotation marks and book marks:

Retrieve the whole string from left to right to search whether the string contains the quotation marks and book marks. If there exists a quotation mark, then examine whether the length of the string between the quotation marks is greater than 6 and whether it contains sentence marks. If the words in the quotation marks are long (“中华民族的伟大祖先”) or contains sentence marks(“啊，真好！”), it shouldn’t be kept as a whole. For book marks, all the string between the book marks are labeled as “bound”.

For terminology:

Terminology contains idioms and proper nouns. The whole string is scanned and if certain terminology is detected according to the data in *term\_list.txt*, mark them as bounded*.*

For particular situations:

Particular situations include the correct separation for particular strings,as their correct relationships are stored in the *particular\_situation.txt.*The algorithm is same as the one used in to find terminology,so the retrieve for terminology and particular situations is included in one retrieve round.

4.3.2 Punctuation separation

The second process is the punctuation separation. The system will delete most of the punctuations and they are replaced by a blank, as they should be separated with others. Also, it will add a blank in front of and behind the whole sentence, which will be used when calculating mi value.

4.3.3 Mi and d*t-score*calculation

Mi and dt-score are two important concepts used in judgment.

*Mi*, known as “mutual information”, reflects the binding tendency of two characters. In word “xy”, the mi value between ‘x’ and ‘y’ is calculated by the formula:

***mi = log2 ( p(xy) \* 1000 / (p(x) \* p(y)) )***

(*p(x)* is the probability of *x,* and the probability is given by lexicon*.*).

Another concept, *t-score*, indicates the binding tendency of *y* in the context of *x* and *z* in the string *“xyz”*. It is calculated by the formula:

***tscore(y) =( p(yz) - p(xy) ) / sqrt( var(xy) + var(yz) )***

*(var(xy)* is the variance of *“xy”* in the whole lexicon, *var(xy) = (p(xy) - average(whole\_lexicon)) ^ 2* , in which *average(whole\_lexicon)* is the mean of the probabilities of all words in the lexicon).

And *dtsocre*, the difference of *t-score*, indicates the binding tendency for *“xy”* in *“wxyz”*. It is calculated by the formula:

***dtscore(xy) = tscore(x) – tscore(y).***

4.3.4 Judgement

The next process is the most important part in word segmentation: judgment. It contains three rounds.

1.New concepts:

(1)height/depthof local maximum/minimum

If *dts(xy)* is the local maximum, then the height of the local maximum is denoted as *“h(dts(xy))”*; if *dts(xy)* is the local minimum, then the depth of the local minimum is denoted as *“d(dts(xy))”*. It is calculated by the formula:

***height(or depth) =***

***min ( local\_ext - value(left\_neighbor),local\_ext - value(right\_neighbor) )***

***(“****local\_ext”*  is the local extremum, and *“value(left\_neighbor)”* is the value of the local extremum’s left neighbor, so is *“value(right)\_neighbor”*.)***)***

(2)left/right second local maximum/minimum

Suppose *“vxyzw”* is a string and *dts(xy)* is the local maximum, then *dts(yz)* is called the right second local maximum if *dts(yz)> dts(vx)* and *dts(yz) > dts(zw)*. *dts(yz)* is called the left second local minimum if *dts(yz)< dts(vx)* and *dts(yz) < dts(zw)*. The right second local maximum and minimum are defined similarly. And define the distance between the local extremum and the second local extremum as “*dis(yz)*”. It is calculated by the formula:

***dis(yz) =abs( dts(xy) – dts(yz) ).***

Then, define left and right distance minimum in the string *“vxyzw”* as *“lrmin”*. It is calculated by the formula:

***lrmin\_l(xy)*** (*“xy”* is the left local extremum) ***=***

***min( abs(dts(xy)-dts(vx)), abs(dts(xy)-dts(zw)) ).***

***lrmin\_r(yz)*** (*“yz”* is the right local extremum) ***=***

***min( abs(dts(yz)-dts(vx)), abs(dts(yz)-dts(zw)) ).***

(3) mean and standard derivation

More, given the input string *S*, let:

***μm = the mean of mi of all locations in S;***

***σm = the standard derivation of mi of all locations in S;***

***μd = the mean of dt-score of all locations in S;***

***σd = the standard derivation of dt-score of all locations in S;***

(4) regions

We then divide the distribution graphs of *mi* and *dts* of *S* into several regions (4 regions for each graph) by*μm, σm, μd* and *σd*:

***region A: dts >μd +σd***

***region B:μd < dts <=μd +σd***

***region C:μd -σd < dts <=μd***

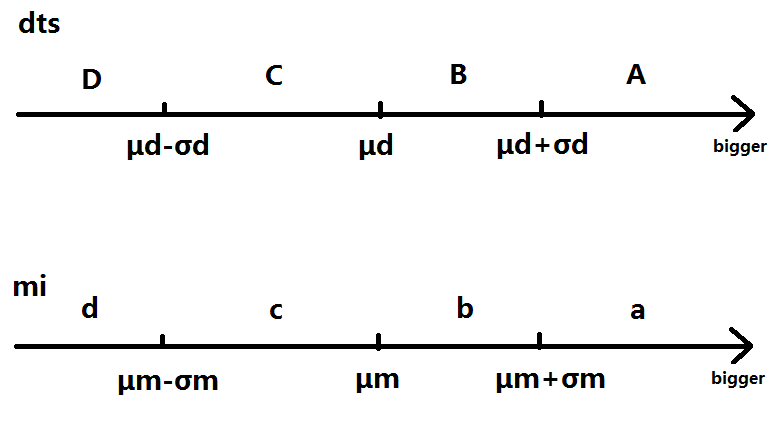
***region D: dts <=μd -σd***

***region a: mi >μm +σm***

***region b:μm < mi <=μm +σm***

***region c:μm -σm < mi <=μm***

***region d: mi <=μm -σm***



2.First round judgement

In the first round, judge the case of every four adjacent character. According to the statistic method, in cases *dts(xy)* and *mi(xy)* fall into:

***Case Aa or Ba or Ca or Da or Ab:***

(This is the case that *mi(xy)* is very large or *mi(xy)* is relatively large with very large *dts(xy),* so that xy is very tend to be bound)

Mark (xy) *“bound”*.

***Case Ad or Bd or Cd or Dd or Dc:***

(This is the case that *mi(xy)* is very small or *mi(xy)* is relatively small with very small *dts(xy),* so that xy is very tend to be separated)

Mark (xy) *“separated”.*

In other situation,ambiguity begins to appear,so six parameter ***δ1***<***δ2***<***δ3***, ***ξ1<ξ2<ξ3*** are introduced to help us judge.Their values are decided in practice.

***Case Ac or Cb:***

(In this case, xy slightly tends to be bound, so local maximum only need to be slightly greater than its neighbor and local minimum should be a bit smaller than its neighbor.)

Judge whether *dts(xy)* is local extremum. If *dts(xy)* is the local maximum: if ***h(dts(xy)) >δ1***, then mark (xy) *“bound”,* and else mark *“?”.* If *dts(xy)* is the local minimum: if ***d(dts(xy)) >ξ2***, then mark (xy) *“separated”,* and else mark *“?”.*

***Case Bc or Db:***

(In this case, xy slightly tends to be separated, so local maximum should a bit greater than its neighbor and local minimum only need to be slightly smaller than its neighbor.)

Judge whether *dts(xy)* is local extremum. If *dts(xy)* is the local maximum: if ***h(dts(xy)) >δ2***, then mark (xy) *“bound”,* and else mark *“?”.* If *dts(xy)* is the local minimum: if ***d(dts(xy)) >ξ1***, then mark (xy) *“separated”,* and else mark *“?”.*

***Case Cc:***

(In this case, xy tends more to be separated, so local maximum should much greater than its neighbor and local minimum should be regarded as “separated”.)

Judge whether *dts(xy)* is local extremum. If *dts(xy)* is the local maximum: if ***h(dts(xy)) >δ3***, then mark (xy) *“bound”,* and else mark *“?”.* If *dts(xy)* is the local minimum: Mark (xy) *“separated”,* and else mark *“?”.*

***Case Bb:***

(In this case, xy tends more to be bound, so local maximum should be regarded as “bound” and local minimum should much smaller than its neighbor.)

Judge whether *dts(xy)* is local extremum. If *dts(xy)* is the local maximum: Mark (xy) *“bound”,* and else mark *“?”.* If *dts(xy)* is the local minimum: if ***d(dts(xy)) >ξ3,*** ,mark (xy) *“separated”,* and else mark *“?”.*

3.Second round

Then in the second round, for (xy) unmarked so far:

If *dts(xy)* is the left second local extremum, then if***dis(xy) < 0.5 \* lrmin\_l(xy)***, mark *“right”*;

If *dts(xy)* is the right second local extremum, then if ***dis(xy) < 0.5 \* lrmin\_r(xy)***, mark *“left”*;

In other cases, mark (xy) as *”?”.*

4.Third round

In the third round, for (xy) marked *“?”*, get mi(xy) to judge the relationship.

If *mi(xy) >= θ*(default = 3.5), mark (xy) as *“bound”*, else as *“separated”*.

If (xy) is marked *“left”*, then the status of (xy) follows its left adjacent location.

If (xy) is marked *“right”*, then the status of (xy) follows its right adjacent location.

Finally, combine all of the characters according to their relationship marks.

The result is a string already cut by split mark “|”.

5.Test

The test is centered on the the correctness and efficiency of algorithm.

5.1 Testing standard

Firstly,we have following definition:

***Golden standard: An generally-recognized segmentation data,from authority organizations or manual segmenting result.***

***N = The number of words in golden standard***

***c = The number of words segmented correctly***

***e = The number of words segmented wrongly***

We adopt a commonly used standard in the field of **Information Retrieval** to evaluate our result,where we refers to following values.

***P(Precision) = c / (c+e)***

***R(Recall) = c / N***

*Precision* evaluates how many correct words are there in our results.

*Recall* evaluates how many correct words in golden standard are caught by our system.

In short,*precision* and *recall* represents the reliability and the power of our system respectively.

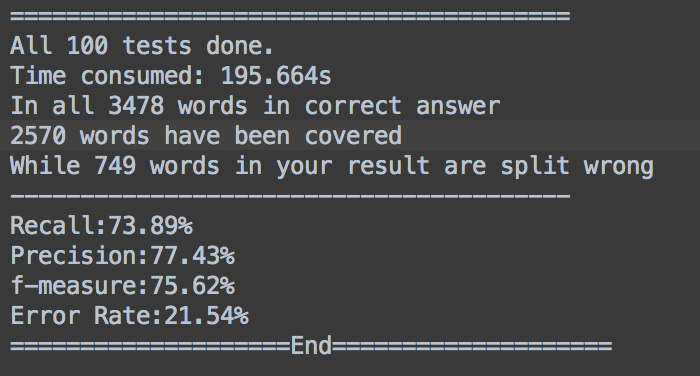
However,*precision* and *recall* usually restrict each other,so we need a way to evaluate both side,which is *f-measure*,the harmonic mean of precision and recall.

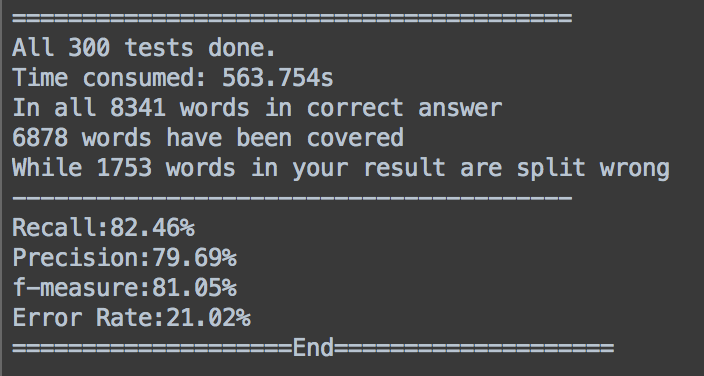
***f-Measure = 2RP / (R+P)***

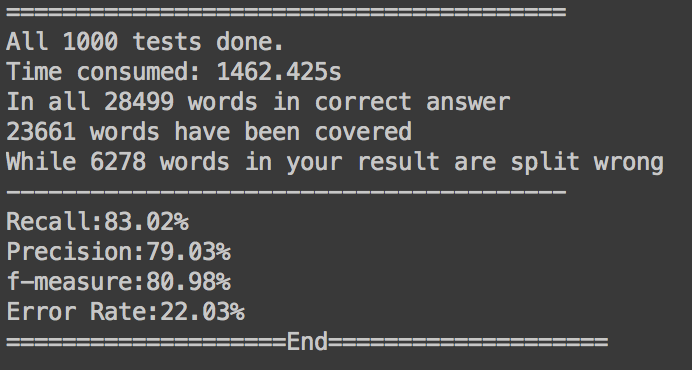
5.2 Testing data source

Our golden standard data comes from the public corpus provided by ShangXi University.

Amount to 3654 sentences.

5.3 Testing result





6.Conclusion

Masterpiece!