**Report**

This is the core part of the segmentation.

The segmentation contains two parts: sentence segmentation and word 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.

Another part is the word segmentation. It contains these procedures: retrieve, punctuation separation, dtscore calculation, mi calculation, judging and combination.

First, retrieve contains the retrieve process for quotation marks, terms and particular situations. The retrieve for quotation mark can 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. The retrieve for terms and particular situations is included in one retrieve round. Terms contain idioms and proper nouns and particular situations contain the correct separation for particular strings. At first, it will get the various lengths of the terms and particular situations. Then it searches the whole string taking each length as the step. If certain term or particular string is detected, mark them as bounded or as their correct relationships according to the data stored in the *particular\_situation.txt*.

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.

The third process is dtscore calculation and the fourth is mi calculation. These two concepts are importantly used in judgment. *Mi*, known as “mutual information”, reflects the binding tendency of two characters. It is calculated by the formula:

***mi = log2 ( p(xy) \* 1000 / (p(x) \* p(y)) )*** (*p(x)* is the probability of *x*).

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 dictionary).

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).***

The next process is the most important part in word segmentation: judgment. It contains three rounds. First, there are some new definitions. 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) )***

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)) ).***

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 dtscore of all locations in S;***

***σd = the standard derivation of dtscore of all locations in S;***

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***

Then the judgment goes three rounds.

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

(parameter relationships: ***δ1***<***δ2***<***δ3***, ***ξ1<ξ2<ξ3*** )

***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”.*

***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 *“?”.*

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 *”?”.*

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 “|”.