Segmentation algorithm

16947210 Bao Bill

2018.01.27

Contents

1. The necessary of segmentation

The current issue when user search information by input what they want into the search box is that the inputted keywords could not accuracy enough. For example, when user want search apple, but they input “apples” or “red apple”. For basic sql search language, which will ignore the information about “apple”. The current solution is used “like” which is a kind of fuzzy query. But we cannot match each word a “like” identification. So, segmentation is coming. If we make a pr-processing before the fuzzy query, the user could receive more accurate information.

2. The category of segmentation algorithm

The category of segmentation algorithm can be divided by Chinese and English words. The English words segmentation rule can be defined three steps:

Step1: Get the word group according to the space / symbol / paragraph.

Step2: Filter and remove stop words like a/an/and/are/then.

Step3: Stemming. For example, apple and apples, doing and done are the same mean. We can combine the deformation of vocabulary.

There are three main stemming algorithm:

1. Porter Stemming
2. Lovins stemmer
3. Lancaster Stemming

Stemming algorithm is not complex. The key is to define the rule and idiom related to the project and it is easy to program.

3. What I have done and what problem I have solved

In this week, I have finished the rule construction and programming for the porter stemming. And I have finished the segmentation by the rule. Here is the six steps of processing words.(M.F.Porter)

Step1: gets rid of plurals and -ed or -ing. e.g.

For example:

caresses -> caress

milling -> mill

messing -> mess

Step2: turns terminal y to i when there is another vowel in the stem.

Step3: maps double suffices to single ones. so -ization (= -ize plus -ation) maps to -ize etc.

Step4: deals with -ic-, -full, -ness etc. similar strategy to step3.

Step5: takes off -ant, -ence etc., in context <c>vcvc<v>.

Step6: step6() removes a final -e if m() > 1.

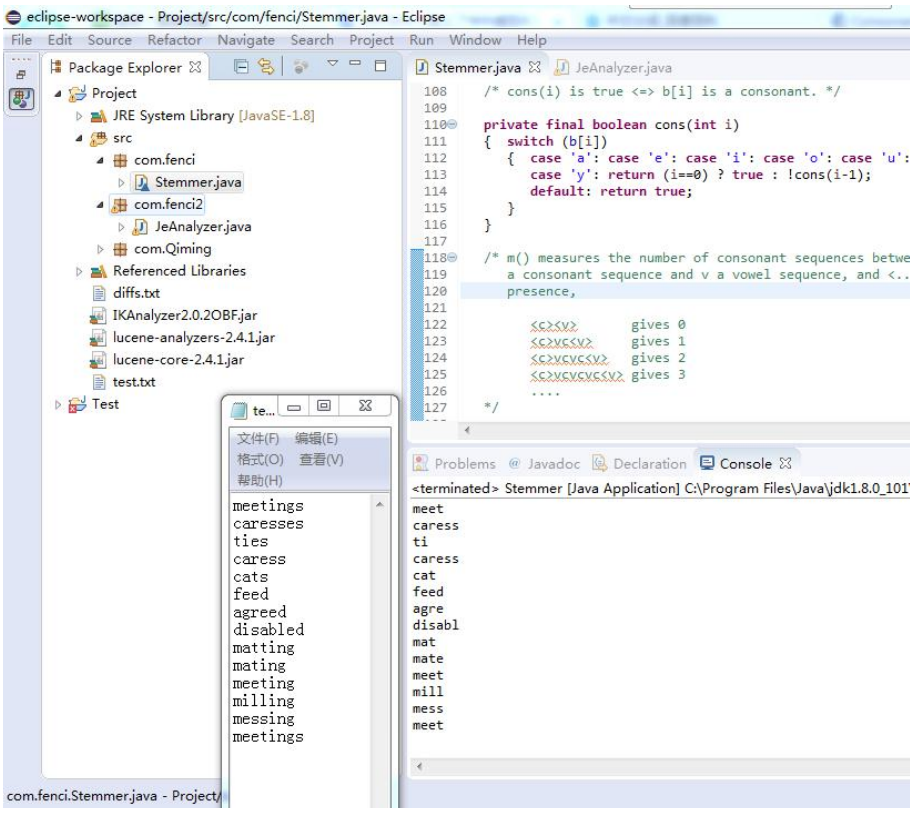
a e i o u

b c d f g h j k l m n p q r s t v w x y z

Vowels and consonants table(EnglishClub)

The first line is the vowels and the second line is consonant.

Here is the test example that I have done:



```
/* cons(i) is true <=> b[i] is a consonant. */

private final boolean cons(int i)
{  switch (b[i])
   {  case 'a': case 'e': case 'i': case 'o': case 'u':
      case 'y': return (i==0) ? true : !cons(i-1);
      default: return true;
   }
}

/* m() measures the number of consonant sequences betwe
   a consonant sequence and v a vowel sequence, and <..
   presence,

      <c><v>       gives 0
      <c>vc<v>     gives 1
      <c>vcvc<v>   gives 2
      <c>vcvcvc<v> gives 3
      ....
*/
```

```
meetings
caresses
ties
caress
cats
feed
agreed
disabled
matting
mating
meeting
milling
messing
meetings
```

```
<terminated> Stemmer [Java Application] C:\Program Files\Java\jdk1.8.0_101
meet
caress
ti
caress
cat
feed
agre
disabl
mat
mate
meet
mill
mess
meet
```

The initial segmentation

The left column is the initial word, and the right is the segmentation word.

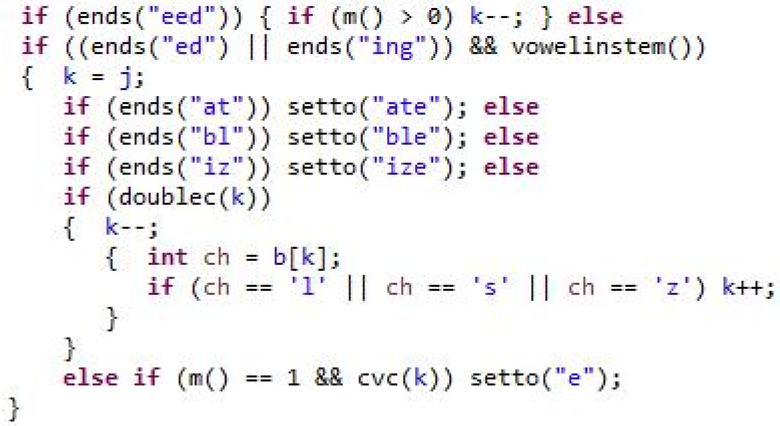
caresses	->	caress
ponies	->	poni
ties	->	ti
caress	->	caress
cats	->	cat
feed	->	feed
agreed	->	agree
disabled	->	disable
matting	->	mat
mating	->	mate
meeting	->	meet
milling	->	mill
messing	->	mess
meetings	->	meet

4. The difficulty of the technology

After the segmentation, I was confused by the result. For example, when I input “create” and “created”, the results are both “creat”. I ask myself why they are the same and it is not the “create”? But when I reread the document on the official website. I find a sentence:

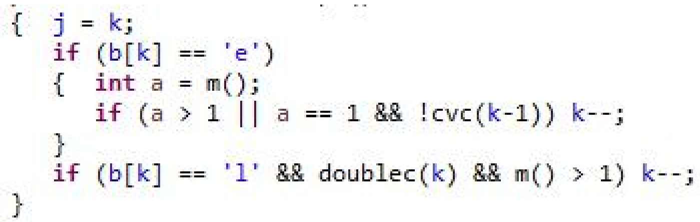
“The purpose of stemming is to bring variant forms of a word together, not to map a word onto its ‘paradigm’ form.”

And then I check the source code:



```
 if (ends("eed")) { if (m() > 0) k--; } else
 if ((ends("ed") || ends("ing")) && vowelinstem())
 {  k = j;
    if (ends("at")) setto("ate"); else
    if (ends("bl")) setto("ble"); else
    if (ends("iz")) setto("ize"); else
    if (doublec(k))
    {  k--;
       {  int ch = b[k];
          if (ch == 'l' || ch == 's' || ch == 'z') k++;
       }
    }
    else if (m() == 1 && cvc(k)) setto("e");
}
```

Delete “ed” method



```
{  j = k;
   if (b[k] == 'e')
   {  int a = m();
      if (a > 1 || a == 1 && !cvc(k-1)) k--;
   }
   if (b[k] == 'l' && doublec(k) && m() > 1) k--;
}
```

Delete “e” method

The source code has deleted the vowels like “ed” and “e”.

The main purpose of porter stemming will not transfer the word to the actual word, it changed these multiple format into one category. In other words, the “created” cannot return to “create”, but it can change the “create” and “created” into “creat”.

The advantage of this operation is when user input “create” and “created”, the server will only search “creat”. Which will maximum improve the search accuracy and correlation degree.

5. The scope of trial of the program and future work

Currently, the program can achieve the above function include segment stop word, the consonant and vowel sequence and so on. But I still need to build a medical word factory and consider more medical word combination. So, I will continue think about the construction of medical word combination rule in the next week.

6. References

J. C. Anceaux, Consonant-Sequences.
https://link.springer.com/chapter/10.1007/978-94-017-5934-2_7

EnglishClub, vowels and consonants for linking.
https://www.englishclub.com/pronunciation/linking-1.htm

M. F.Porter, The porter stemming algorithm.
https://tartarus.org/martin/PorterStemmer/