Report for Spelling Correction

Huang Pingxuan
15307130283

Context:

Overview of Project

According to the theory, we can divide the project into two parts

1. Noisy-channel
2. Language-model

To make these theories come true, we can divide this PJ into 5 part:

1. Preprocess
2. Confusion-matrix
3. Noisy-channel
4. Language-model
5. Implementation

Overview of procedure

After processing the Reuters corpus, Test set with NLTK and SRILM, I build two Language-models based on both the ‘Good-Turing’ and ‘Add-one smoothing’ theories. In the next step, I create the Confusion-matrix and make the noisy-channel come true.

However, when I want to apply my language- model on the test set, I realized that the Language-model makes no sense for the reason that not all of the candidate words appears on the test.txt and the model bases on Test and Reuters corpus only. So I made a new Language-model.

For the reason above, this report will be shown in the order of ‘Confusion-matrix’ → ‘Noisy-channel’ → ‘Language-model’

Confusion-matrix

Thinking:

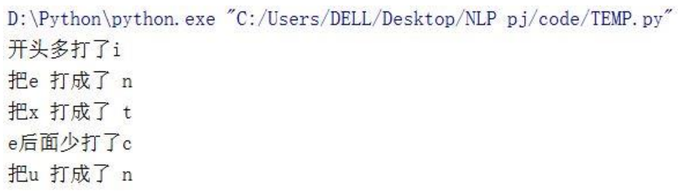
- Theoretically, many typos have just one mistake which can be classified into 4 types: insertion, deletion, substitution, reversal. So we can find out the type of miss-spelling by trying each of these four kinds of mistake. However, in real situation, the existence of multiple miss-spelling and the truth that we can deal with the same typo in different strategies force me to use the ‘shortest editing distance’ strategy to find out what kind of mistake the word has.

- We need to deal with Reversal separately, because the strategy of ‘shortest editing distance’ cannot find it out, and this kind of miss-spelling can be caused by other types of mistakes. And we need to assume that:

  Reversal will not appear with other types of mistake and it will appear no more than one times in single word

Implementation:

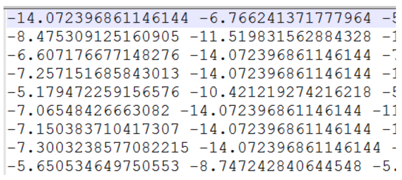
- Creating class Record
- Generating the Editing-matrix and calculating some information included the shortest editing distance.
- Finding out the editing trace by backtracking. (the picture bellow shows the editing trace from ‘execution’ to ‘intention’)



```
D:\Python\python.exe "C:/Users/DELL/Desktop/NLP pj/code/TEMP.py"
开头多打了i
把e 打成了 n
把x 打成了 t
e后面少打了c
把u 打成了 n
```

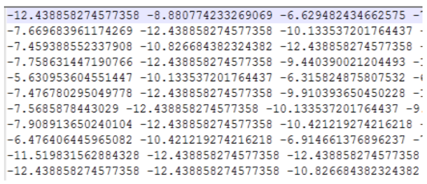
- Edit the confusion-matrix according to the editing trace
- The procedure of editing Confusion-matrix:
  a) Process 0 data in Confusion-matrix with ‘Good-Turing’
  b) Transfer the data in Confusion-matrix into probability.

Some parts of Result as bellow:



```
-14.072396861146144 -6.766241371777964 -5
-8.475309125160905 -11.519831562884328 -1
-6.607176677148276 -14.072396861146144 -7
-7.257151685843013 -14.072396861146144 -1
-5.179472259156576 -10.421219274216218 -5
-7.06548426663082 -14.072396861146144 -11
-7.150383710417307 -14.072396861146144 -1
-7.3003238577082215 -14.072396861146144 -
-5.650534649750553 -8.747242840644548 -5.
```

ADD

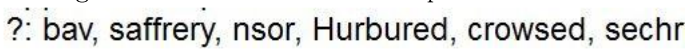


```
-12.438858274577358 -8.880774233269069 -6.629482434662575 -7
-7.669683961174269 -12.438858274577358 -10.133537201764437 -
-7.459388552337908 -10.826684382324382 -12.438858274577358 -
-7.758631447190766 -12.438858274577358 -9.440390021204493 -1
-5.630953604551447 -10.133537201764437 -6.315824875807532 -6
-7.476780295049778 -12.438858274577358 -9.910393650450228 -1
-7.5685878443029 -12.438858274577358 -10.133537201764437 -9.
-7.908913650240104 -12.438858274577358 -10.421219274216218 -
-6.476406445965082 -10.421219274216218 -6.914661376896237 -7
-11.519831562884328 -12.438858274577358 -12.438858274577358
-12.438858274577358 -12.438858274577358 -10.826684382324382
```

DEL

Remark:

- Problem can be aroused if some variables share the same space
- There is a strange sentence in the corpus needed to be delated:



?: bav, saffrery, nsor, Hurbured, crowsed, sechr

Noisy-channel

Thinking:

- Try to correct wrong word with only one correction and calculate the probability of this correction with Confusion-matrix;
- If candidate-list cannot be created by the procedure above, we need to do multiple correction until the appearance of candidate-list.

Implementation:

- Code the Single-correction function (SCF), we need to **pay attention to the usage of Confusion-matrix!**(Eg: We should use DEL matrix rather than ADD matrix if we want to add one letter in a typo)
- Multiple-correction function (MCF) can be realized by recursion: This MCF will do a correction in advance and delivers the new word to SCF. If candidate cannot be created until now, the SCF will call the MCF again till the length of candidate-list is not 0

Attention:

- Using of Confusion-matrix said above
- We need to limit the times of recursion, if not, the correction such as adding a letter to a word will be processed forever
- The probabilities of correction need to accumulate by addition rather than multiplication. (Having taken a logarithm before)

Language-model

Procedure:

- Build model based on the Reuters corpus + Test set (I used good-Turing strategy to deal with 0-problems only)
- Give up Good-Turing, the followed are reasons:
  a) We cannot know the number of 0-word and 0-sequence before test
  b) Nearly 3/4 of words in dictionary are not in the training ser, so we cannot use the dictionary to forecast 0-word
  c) Most of sequence of test is in the corpus (only 2390 is not)
  d) We cannot use Good-Turing without knowing the number of 0-word
- Use Add-one smoothing to build model
- I realized it later that all the Language-model build before make no sense because not all candidates appear in the corpus.
- Use ‘Add-P smoothing’ to build new model

Final Language-model

- Use Add-1 smoothing strategy
- We cannot figure out the gross number of words accurately (we don’t know how many 0-word there are)
- We can divide the corpus into Train and Test, and use Test to estimate the number of 0-word. (the work is very heavy)
- Use Add-P (P is not always 1). Assume the number of 0-word is P_w and the number of 0-sequence is P_s

Implementation

- Code function correct(*word*) in baggage confusion-matrix, it can provide candidate-list and relevant probability
- Code function Max_prob(*word1, word2, word3*)in baggage language-model, it can call the correct function to get candidate list and provide the best correction strategy of word2(word and probability)
- two ways to process Test text:
  a) Provide the best correction strategy of each word in the sentence, and choose the best. If there is more than one typo, we can find the best, the less-best and the like
  b) Deal with non-word first.

Implementation and Result:

- Use strategy 1: accuracy is no more than 3.8%
- Use strategy 2: deal with non-word with highest probability, and this word will not be processed in later period.
- Accuracy can boost into 77.2% with strategy 2

Follow-up

Preprocess:

Because I need to create confusion-matrix and noisy-channel, I simply removed all punctuations and take apart ‘number+letter’ word. It cannot be denied that this gross preprocess will influence the accuracy. And I need to add some rules for the procedure of output (Eg: we have to type space after U.S. before we enter anything else)

Conclusion:

I finished all the procedure of spelling correction except counting the number of words. And there are 2 shortages:

- Preprocess is too simple to support the project
- Haven’ t divide the corpus into Train set and Held-out

In one word: bittersweet!