Lab 2 report: Language Model

1. A brief description of the code：

The function ‘word_segmentation’ is to do word segmentation. First, change all words from uppercase to lowercase and use regex to match words from text (About regex, I didn’t match number from text, only words, because I think number is useless for language model). Then, I append start (‘<s>’) and end (‘<\s>’) symbols before the first and last words of a sentence respectively. Unigram consists of every single word including <s> and <\s> symbol. For bigram, I add a space between two words to make a bigram word. In this way, I can get the unigram and bigram, at the same time, I count the number of unigram and bigram for the use of next step.

The function ‘get_probability’ is to get the probabilities of unigram, bigram and bigram with add-1 smoothing. First, I use library ‘Counter’ to get the frequencies of every word in unigram and bigram. Then, for unigram probability, directly divide the frequency by the total number of unigrams which I have got from previous function. For bigram probability, for one bigram, there is a space between two words, so I use 'split' to get the previous word (i.e. the n-1th word), and then get the frequency of previous word from unigram’s frequency, then divide the bigram’s frequency by the previous word’s frequency to get the probability of bigram. For bigram with add-1 smoothing, just plus 1 to the numerator and the number of unigrams to the denominator.

The function ‘get_structured_sentence’ is to prepare structured sentences from question text for testing language model. First word segmentation, then append start (‘<s>’) and end (‘<\s>’) symbols before the first and last words of a sentence respectively and an important step is replacing ‘____’ by two candidate words. Because there are two candidate words, so I create two lists to include these two words respectively. And I append two candidate lists to a new list for the use of next step.

The functions ‘unigrams_model’, ‘bigrams_model’ and ‘bigrams_smooth_model’ are to test three models by question text. I compute the score for two candidate words respectively by these three models (i.e. the probabilities of two candidate words), then I create a list named ‘answer’ to store the index of higher score between two candidate score (i.e. either 0 or 1). And in this process, according to ‘in case a language model returns only 0 probabilities, its answer should be considered incorrect’, so if the scores of two candidate words are both zero, then append ‘2’ to answer list, which means the answer is incorrect. According to ‘in case of a tie with non-zero probabilities, its answer should be considered half-correct’, so if one of the scores of two candidate words is zero, then append ‘3’ to answer list, which means the answer is half-correct.

The function ‘compute_correct_rate’ is to compute the accuracy of three language models. I prepare the standard answer manually to a list in advance (i.e. number 0 and 1 stand for the first candidate word and the second word respectively). And compare the standard answer with the answer from three models, if they are same, plus 1 to the number of correct answers, if 3 appear in the list, plus 0.5 to the number of correct answers. Then, use the number of correct answers divide the total number of question sentence to get the accuracy of models.

2. The accuracy of three language model：

```
vvvvvdeMacBook-Pro:Lab 2 vvvvv$ python lab2.py news-corpus-500k.txt questions.tx
t
The accuracy of unigram language model is : 0.6
The accuracy of bigram language model is : 0.8
The accuracy of bigram with add-1 smoothing language model is : 0.9
```

3. The result of three language model:

(the completed sentence of unigram model with its probability)

```
i don 't know whether to go out or not 5.50224877959276e-32
we went through the door to get inside 5.1664873849239964e-27
they all had a peace of the cake 6.7920088723925005e-25
she had to go to court to prove she was innocent 3.1813990978963766e-34
we were only allowed to visit at certain times 1.2328594521281837e-30
she went back to check she had locked the door 3.948262927840653e-34
can you here me 6.012025128523453e-16
do you usually eat serial for breakfast 1.484723621067643e-29
she normally choose with her mouth closed 1.3816659997932766e-28
i 'm going to sell it on the internet 6.983993150654394e-28
```

(the completed sentence of bigram model with its probability)

```
i don 't know whether to go out or not 4.784122367130774e-18
we went through the door to get inside 2.5554973322492537e-17
they all had a piece of the cake 3.79305970160943e-18
she had to go to court to prove she was innocent 2.56802730204242 54e-24
we were only allowed to visit at certain times 1.8599307575619532e-22
she went back to check she had locked the door 2.2515301970680843e-24
can you hear me 4.008505669945721e-12
do you usually eat cereal for breakfast 0.0
she normally choose with her mouth closed 0.0
i 'm going to sell it on the internet 7.906504177034171e-16
```

(the completed sentence of bigram with add-1 smoothing model with its probability)

```
i don 't know whether to go out or not 6.187746529432018e-48
we went through the door to get inside 3.1924681370454745e-42
they all had a piece of the cake 3.0429861014668555e-41
she had to go to court to prove she was innocent 2.669607205806192e-56
we were only allowed to visit at certain times 8.868171899158598e-51
she went back to check she had locked the door 5.6135851665463915e-56
can you hear me 1.614116683107221e-26
do you usually eat cereal for breakfast 5.583368478378952e-48
she normally choose with her mouth closed 1.8482323209209164e-46
i 'm going to sell it on the internet 1.2031293651368684e-41
```

4. The discussion of these results:

Observing the result, the accuracy of unigram model is lowest, because unigram model does not take context into account, only consider the probability of a single word, which means it depends on the training data, if one of the probabilities of candidate word is higher than others in training data, it will be chosen in unigram model. Bigram model is the second most accurate between these three, because some bigram word such as ‘<you here>’ or ‘<here me>’ never appear in English grammar and structure (unless spelling mistakes), so these bigrams won’t appear in training data or with very low probability, so the accuracy is not very bad. But in test data, there are still some words those don’t appear in training data, bigram model can’t deal with this issue, so we use add-1 smoothing model which has very high accuracy and makes every bigram has probability, so in practice, adding smooth is very important for language model.