Intermediate report part a - step 2

Julian Main(10541578), Yorick de Boer(10786015), Amor Frans(10897844)

February 9, 2016

Introduction

For step 2 of part A we wrote a python script which was able to do the following: (1) find the N-grams for a given natural number *(n)*, (2) compute the probability $P(w_n|w_1,...,w_{n1})$ (*the probability that the sequence* $w_1,...,w_{n1}$ *is followed by* w_n)., (3) compute the probability of a sequence using the formula: $P(w_1,...,w_m) = \prod_{i=1}^{i=m+1} P(w_i|w_{in+1},...,w_{i1})$, (4) make a permutation of a sentence and calculate the probability of each permutation given the corpus.

Method

Question 1 :
The python script could be executed with the command:

```
    python3.4 a1-step1.py [-h] [-n N] [-m M] corpus
```

Our program is made in the following structure:
- Read input
Using the python argparse module the input is read from the command line.
- Insert start and stop for each paragraph
Using the python re module the blank lines in the corpus can be found and there the start and stop sign can be placed.
- Make bigrams for n and (n-1)
Using the program of the last assignment we can get the ngrams for n and the ngrams for (n-1) and combine them together in one table.

Question 2 :
The python script could be executed with the command:

```
python3.4 a1-step1.py [-h] [-n N] [-m M] corpus
```

Our program is made in the following structure:
- *Read input*
Using the python argparse module the input is read from the command line.
- *Count the occurrence of sequence (w1,..,wn) and (w1,..,wn-1) in corpus*
Using the python module collections we can let python count the occurrences of the two sentences in the corpus.
Compute $P(w_n|w_1, ..., w_n 1)$
$P(w_n|w_1, ..., w_n 1) = Count(w1, ..., wn)/Count(w1, ..., wn - 1)$. In the previous step we already got the counts of these two terms so the only thing to do is to divide the terms. We loop through all sentences in the conditional probability file.

Question 3 :
The python script could be executed with the command:

```
python3.4 a1-step1.py [-h] [-n N] [-m M] corpus
```

Our program is made in the following structure:
- *Read input*
Using the python argparse module the input is read from the command line.
- *Compute* $\prod_{i=1}^{i=m+1} P(w_i|w_{in+1}, ..., w_{i1})$
We do this step because $P(w_1, ..., w_m) = \prod_{i=1}^{i=m+1} P(w_i|w_{in+1}, ..., w_{i1})$. The right hand of the equation we can use the program from question 2. We only add a loop that loops through all i=2,3,.,n.

Question 4 :
The python script could be executed with the command:

```
python3.4 a1-step1.py [-h] [-n N] [-m M] corpus
```

- *Read input*
Using the python argparse module the input is read from the command line.
- *Make permutations of the words in a sentence*
Using the itertools python module we can make all the permutations of a sentence.
- *Compute the probability of each permutation*
Use the program of question 3 to compute the probability of each permutation.

Results

Question 1 :
Most common bigrams:
(('of', 'the'), 2507),
(('to', 'be'), 2234),
(('in', 'the'), 1917),
(('I', 'am'), 1365),
(('of', 'her'), 1268),
(('to', 'the'), 1142),
(('it', 'was'), 1010),
(('had', 'been'), 995),
(('she', 'had'), 978),
(('to', 'her'), 965)

Question 2 :
cond_file.txt

```
She was
years had
but not
```

Conditional probability:
'had': 0.02242152466367713,
'not': 0.01586042823156225,
'was': 0.1841052029731275

Question 3 :
The event had every promise of happiness for her mate 'The event had every promise of happiness for her mate': 1
Question 4 :

Discussion and conclusion

As we can see in the results all our programs work well. The programs give us a good basis to solve future assignments.