Part 1.

Question:

We are given the following corpus, modified from the one in the chapter:

<s> I am Sam </s>

<s> Sam I am </s>

<s> I am Sam </s>

<s> I do not like green eggs and Sam </s>

Using a bigram language model with add-one smoothing, what is P(Sam | am)?

Include <s> and </s> in your counts just like any other token.

Answer:

Probability of Sam given am:

$$P(Sam \mid am) = \frac{count(\ am,\ sam)}{count(am)}$$

With add one smoothing:

Let V = number of unique tokens in the corpus

$$P(Sam \mid am) = \frac{count(\ am,\ sam) + 1}{count(am) + V}$$

Pseudo code:

Step 1. Tokenize corpus in a list

Step 2. Create hashmap for every pair of words in the corpus

Step 3. Add one to the value of every pair

Step 4. Isolate the count of the pair (am, Sam)

Step 5. Isolate the sum of the count of all pairs (am, n), where n is a token in the corpus

Step 6. Divide the value found in step 4 with the value found in step 5 to obtain P(sam | am)

Results:

count(am, Sam) = 2

count(am) = 3

V = 11

P(sam | am) = 3/14

Part 2.

1.1 Pre-Processing:

Prior to training, please complete the following pre-processing steps:

1. Pad each sentence in the training and test corpora with start and end symbols (you can use <s> and</s>, respectively).

2. Lowercase all words in the training and test corpora. Note that the data already has been tokenized (i.e. the punctuation has been split off words).

3. Replace all words occurring in the training data once with the token <unk>. Every word in the test data not seen in training should be treated as <unk>.

Pseudo-code:

Step 1. Create a hashmap of the txt files lowercasing and adding sentence padding equal to end of sentence punctuation

Step 2. Add text file to list referencing hashmap to replace singletons with <unk>, lowercasing and padding as necessary.

1.2 Training the Models:

Please use train.txt to train the following language models:

1. A unigram maximum likelihood model.

2. A bigram maximum likelihood model.

3. A bigram model with Add-One smoothing.

Pseudo-code:

Step 1. Create a hashmap of occurrences of each word or word-pair in the training data

Step 2.

Let

V = number of unique tokens in the model

n = number to smooth (in add n smoothing)

To return probability for unigram model:

$$p(word_i) = \frac{count(word_i)}{V}$$

For a bigram model with "n" smoothing (in this case 0 or 1):

$$P(word_i|word_{i-1}) = \frac{count(word_{i-1}, word_i) + n}{count(word_{i-1}) + n * V}$$

1.3 Questions:

1. How many word types (unique words) are there in the training corpus? Please include the end-of-sentence padding symbol </s> and the unknown token <unk>. Do not include the start of sentence padding symbol <s>.

Answer:

There are 41,738 word types in the training data

2. How many word tokens are there in the training corpus? Do not include the start of sentence padding symbol <s>.

Answer:

There are 2,458,588 word tokens in the training data

3. What percentage of word tokens and word types in the test corpus did not occur in training (before you mapped the unknown words to <unk> in training and test data)? Please include the padding symbol </s> in your calculations. Do not include the start of sentence padding symbol <s>.

Answer:

Approximately,

3.6 percent of word types did not appear in training

1.6 percent of word tokens did not appear in training

4. Now replace singletons in the training data with <unk> symbol and map words (in the test corpus) not observed in training to <unk>. What percentage of bigrams (bigram types and bigram tokens) in the test corpus did not occur in training (treat <unk> as a regular token that has been

observed). Please include the padding symbol </s> in your calculations. Do not include the start of sentence padding symbol <s>.

Answer:

Approximately,

25.3 percent of bigram types did not occur in training

21.0 percent of bigram tokens did not occur in training

5. Compute the log probability of the following sentence under the three models (ignore capitalization and pad each sentence as described above). Please list all of the parameters required to compute the probabilities and show the complete calculation. Which of the parameters have zero values under each model? Use log base 2 in your calculations. Map words not observed in the training corpus to the <unk> token.

I look forward to hearing your reply.

Answer:

To find log probability of a sentence with m tokens:

For unigram models

$$\log_2 \prod_{i=1}^{m} P(s_i) = \sum_{i=1}^{m} \log_2 P(s_i)$$

And for bigram models

$$\log_2 \prod_{i=1}^{m} P(s_i \,|s_{i-1}) = \sum_{i=1}^{m} \log_2 P(s_i|s_{i-1})$$

Unigram model output

Rounded to one digit

log_2(P(<s>)) = -4.8

log_2 (P(i)) = -8.4

log_2 (P(look)) = -12.0

log_2 (P(forward)) = -12.4

log_2 (P(to)) = -5.6

log_2 (P(hearing)) = -13.6

log_2 (P(your)) = -11.0

log_2 (P(reply)) = -17.6

log_2 (P(.)) = -4.9

log_2 (P(</s>)) = -4.8

In summation:

log_2 (P(I look forward to hearing your reply .)) = -95.1

Bigram model output

Rounded to one digit

log_2 (P(i | <s>)) is -5.8

log_2 (P(look | i)) is -8.9

log_2 (P(forward | look)) is -4.2

log_2 (P(to | forward)) is -2.3

log_2 (P(hearing | to)) is -13.1

log_2 (P(your | hearing)) is undefined

log_2 (P(reply | your)) is undefined

log_2 (P(.|reply)) is undefined

log_2 (P(</s>|.)) is -0.0

In summation:

log_2 (P(I look forward to hearing your reply .)) = undefined

Which implies

P(I look forward to hearing your reply .) = 0

This is because the word “reply” cannot be generated after “your” in the bigram model.

Bigram add-one output

Rounded to one digit

P(i | <s>) is -6.29

P(look | i) is -11.5

P(forward | look) is -10.2

P(to | forward) is -8.70

P(hearing | to) is -13.7

P(your | hearing) is -15.4

P(reply | your) is -15.4

P(. | reply) is -15.3

P(</s> | .) is -0.6

In summation

log_2 (P(I look forward to hearing your reply .)) = - 97.2

6. Compute the perplexity of the sentence above under each of the models.

$$perplexity = 2^{-l}$$

Where in the unigram case

$$l = \frac{1}{M}\sum_{i=1}^{m} \log_2 P(s_i)$$

Or in the bigram case

$$l = \frac{1}{M}\sum_{i=1}^{m} \log_2 P(s_i|s_{i-1})$$

unigram model perplexity for the sentence = 730.2

bigram model perplexity for the sentence = undefined

bigram add-one model perplexity for the sentence = 843.6

7. Compute the perplexity of the entire test corpus under each of the models. Discuss the differences in the results you obtained.

unigram model = 1012.3

bigram model = undefined

bigram add-one model = 78569.0

The unigram model has the lowest perplexity indicating that it fits with the test data better than the other models. The bigram model is undefined as some bigram tokens that occur in testing, do not occur in the training. One example is the token (broke | of). The bigram add-one model does not seem to fit the data well at all.