Final Exam Q3 Set/Dict/Tuple (10%)

Language Model

Youtube Link to help you understand this problem easier.

Problem

Language Model (LM) is one of the important concepts in Natural Language Processing (NLP). It can be applied to estimate the probability of each sentence. Please note that this is a simplified version of LM; we are going to use “bigram probability”.

Assume that there are 3 sentences in the training data, where <s> and </s> are start and end tokens. Also, assume there is only one sentence per row, and each word is separated by “space”.

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I am not Sam </s>

Based on the training data, we can count the number of sentences for each individual word called “unigram count”.

- count(<s>) = 3
- count(I) = 3
- count(am) = 3
- count(Sam) = 3
- count(not) = 1
- count(</s>) = 3
- #vocab = 6 words (There are 6 words.)

Also, we can count the number of sentences for each pair of words called “bigram count”.

- count(<s>, I) = 2; #sentences of <s> followed by I
- count(I, am) = 3; #sentences of I followed by am
- count(am, Sam) = 1; #sentences of am followed by Sam
- count(Sam, </s>) = 2; #sentences of Sam followed by </s>
- etc.

Thus, the “bigram probability” of (w1,w2) [word1 followed by word2] is count(w1,w2) / count(w1) (considered bigram_count / unigram_count)

- prob(<s>, I) = count(<s>,I) / count(<s>) = 2/3 = 0.67
- prob(I, am) = count(I, am) / count(I) = 3/3 = 1.00
- prob(am, Sam) = count(am, Sam) / count(am) = 1/3 = 0.33
- prob(Sam, </s>) = count(Sam, </s>) / count(</s>) = 2/3 = 0.67

Finally, the bigram probability of each sentence can be calculated as examples below.

- Note that for the unknown case (called "unk"), prob(unk) = 1/#vocab = 1/6 = 0.16667
- **Case1:** prob("<s> I am Sam </s>")
    - = prob(<s>, I)*prob(I, am)*prob(am, Sam)*prob(Sam, </s>)
    - = 0.67 * 1.00 * 0.33 * 0.67
    - = 0.14815
- **Case2:** prob("<s> I love Sam </s>")
    - = prob(<s>, I)*prob(I, love)*prob(love, Sam)*prob(Sam, </s>)
    - = prob(<s>, I)*prob(unk)*prob(unk)*prob(Sam, </s>)
    - = 0.67 * 0.16667 * 0.16667 * 0.67
    - = 0.01235
- **Case3:** prob("")
    - Since there is no "" (empty string) in the bigram prob, it is considered as "unk"
    - = prob(unk)
    - = 0.16667

To do

Implement a program to calculate "a bigram probability" of an input sentence based on the given training data. There are 2 functions that you have to implement.

- **train_language_model(data)**
    - Input: data: a **list** of sentences
    - Return: model which is a **dictionary** of 3 components
        - model['unk'] is 1/#vocab.
        - model['unigram'] = a **dictionary** of unigram counts
        - model['bigram'] = a **dictionary** of bigram counts
- **compute_sentence(sentence, model)**
    - Input:
        - sentence: an input string to be calculated probability
        - model: language model
    - Return:
        - probability: probability of sentence to occur
- Please note that fixed calculation cannot be scored. For example, you calculate prob("unk") by using this statement "unk = 1/6" - this cannot be scored!