Add-one smoothing

In the 18th century, Laplace invented add-one smoothing. In add-one smoothing, 1 is added to the count of each word. Instead of 1, any other value can also be added to the count of unknown words so that unknown words can be handled and their probability is non-zero. Pseudo count is the value (that is, either 1 or nonzero) that is added to the counts of unknown words to make their probability nonzero.

```
In [5]:  import nltk
         corpus=u"<s> hello how are you doing ? Hope you find the book interesting. </s>".s
```

```
In [6]:  sentence=u"<s>how are you doing</s>".split()
```

```
In [3]:  vocabulary = set(corpus)
         len(vocabulary)

Out[3]:  13
```

```
In [7]:  cfd = nltk.ConditionalFreqDist(nltk.bigrams(corpus))
```

```
In [13]:  cfd.keys()

Out[13]:  dict_keys(['<s>', 'hello', 'how', 'are', 'you', 'doing', '?', 'Hope', 'find',
          'the', 'book', 'interesting.', '<s>how'])
```

```
In [9]:  [cfd[a][b] for (a,b) in nltk.bigrams(sentence)]   # The corpus counts of each bigr

Out[9]:  [0, 1, 0]
```

```
In [10]:  [cfd[a].N() for (a,b) in nltk.bigrams(sentence)]   # The counts for each word in t

Out[10]:  [0, 1, 2]
```

```
In [11]:  [cfd[a].freq(b) for (a,b) in nltk.bigrams(sentence)]  # There is already a FreqDis

Out[11]:  [0, 1.0, 0.0]
```

```
In [14]:  [1 + cfd[a][b] for (a,b) in nltk.bigrams(sentence)]   # Laplace smoothing of each

Out[14]:  [1, 2, 1]
```

```
In [15]:  [len(vocabulary) + cfd[a].N() for (a,b) in nltk.bigrams(sentence)]   # We need to

Out[15]:  [13, 14, 15]
```

In [16]:

```
# The smoothed Laplace probability for each bigram:
[1.0 * (1+cfd[a][b]) / (len(vocabulary)+cfd[a].N()) for (a,b) in nltk.bigrams(sent
```

Out[16]:

```
[0.07692307692307693, 0.14285714285714285, 0.06666666666666667]
```

Consider another way of performing Add-one smoothing or generating a Laplace probability distribution:

In [18]:

```
# MLEProbDist is the unsmoothed probability distribution
cpd_mle = nltk.ConditionalProbDist(cfd, nltk.MLEProbDist,bins=len(vocabulary))
```

In [19]:

```
[cpd_mle[a].prob(b) for (a,b) in nltk.bigrams(sentence)] # Now we can get the MLE
```

Out[19]:

```
[0, 1.0, 0.0]
```

In [22]:

```
#LaplaceProbDist is the add-one smoothed ProbDist
cpd_laplace = nltk.ConditionalProbDist(cfd, nltk.LaplaceProbDist,bins=len(vocabula
```

In [23]:

```
# Getting the Laplace probabilities is the same as for MLE
[cpd_laplace[a].prob(b) for (a,b) in nltk.bigrams(sentence)]
```

Out[23]:

```
[0.07692307692307693, 0.14285714285714285, 0.06666666666666667]
```

In [25]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-25-94ea72af3875> in <module>()
----> 1 corpus_kn = [[((x[0],y[0],z[0]),(x[1],y[1],z[1])) for x, y, z in nltk.
trigrams(sent)] for sent in corpus_kn[:100]]

NameError: name 'corpus_kn' is not defined
```

In [ ]: