# N - GRAMS

1. Write out the equation for trigram probability estimation (modifying Eq. 3.11). Now write out all the non-zero trigram probabilities for the I am Sam corpus on page 4.

Trigram probability estimation
$P(w_n|w_1, w_2, \dots, w_{n-1}) \approx \prod_{i=1}^{n} P(w_i|w_{i-2}w_{i-1})$

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

$P(\text{Sam}|\text{I, am}) = \frac{1}{2}$
$P(\text{am}| <s>, \text{I}) = \frac{1}{2}$
$P(</s> |\text{I, am}) = \frac{1}{2}$

---

2. Calculate the probability of the sentence i want chinese food. Give two probabilities, one using Fig. 3.2 and the 'useful probabilities' just below it on page 6, and another using the add-1 smoothed table in Fig. 3.6. Assume the additional add-1 smoothed probabilities P(i|<s>) = 0.19 and P(</s>|food) = 0.40

---

P(i|<s>) = 0.19

P(</s>|food) = 0.40

P (i want chinese food) = P(i|<s>) * P(want|i) * P(chinese|want) * P(food|chinese) * P(</s>|food)

P (i want chinese food) = 0.19 * 0.0027 * 0.0001 * 0.0004 * 0.40

P (i want chinese food) = 0.000000162

---

P(i|<s>) = 0.19

P(want|i) = 0.35

P(chinese|want) = 0.17

P(food|chinese) = 0.26

P(</s>|food) = 0.40

P(i|<s>) = 0.19

P(want|i) = 0.32

P(chinese|want) = 0.15

P(food|chinese) = 0.24

P(</s>|food) = 0.40

---

P(i|<s>) = 0.032

P(want|i) = 0.028

P(chinese|want) = 0.016

P(food|chinese) = 0.024

P(</s>|food) = 0.064

---

3. Which of the two probabilities you computed in the previous exercise is higher, unsmoothed or smoothed? Explain why.

The first probability is higher because it is not smoothed. The probability of the unsmoothed model is higher.

---

4. 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.

P(Sam|am) = (count(am Sam) + 1) / (count(am) + V)

count(am Sam) = 2

count(am) = 3

V = 4

P(Sam|am) = (2 + 1) / (3 + 4) = 3/8

---

5. Suppose we didn't use the end-symbol </s>. Train an unsmoothed bigram grammar on the following training corpus without using the end-symbol </s>:

<s> a b        <s> b b        <s> b a        <s> a a

Demonstrate that your bigram model does not assign a single probability distribution across all sentence lengths by showing that the sum of the probability of the four possible 2 word sentences over the alphabet {a,b} is 1.0, and the sum of the probability of all possible 3 word sentences over the alphabet {a,b} is also 1.0.

P(ab) = P(a|<s>)P(b|a) = 0.25

P(ba) = P(b|<s>)P(a|b) = 0.25

P(aa) = P(a|<s>)P(a|a) = 0.25

P(bb) = P(b|<s>)P(b|b) = 0.25

P(aab) = P(a|<s>)P(a|a)P(b|a) = 0.0625

P(aba) = P(a|<s>)P(b|a)P(a|b) = 0.0625

P(baa) = P(b|<s>)P(a|b)P(a|a) = 0.0625

P(abb) = P(a|<s>)P(b|a)P(b|b) = 0.0625

P(aabb) = P(a|<s>)P(a|a)P(b|a)P(b|b) = 0.015625

P(abab) = P(a|<s>)P(b|a)P(a|b)P(b|a) = 0.015625

P(abba) = P(a|<s>)P(b|a)P(b|b)P(a|b) = 0.015625

P(baab) = P(b|<s>)P(a|b)P(a|a)P(b|a) = 0.015625

P(baba) = P(b|<s>)P(a|b)P(b|a)P(a|b) = 0.015625

P(bbaa) = P(b|<s>)P(a|b)P(a|a)P(a|b) = 0.015625

P(aabbb) = P(a|<s>)P(a|a)P(b|a)P(b|b)P(b|a) = 0.00390625

P(ababa) = P(a|<s>)P(b|a)P(a|b)P(b|a)P(a|b) = 0.00390625

P(abbab) = P(a|<s>)P(b|a)P(b|b)P(a|b)P(b|a) = 0.00390625

P(abbba) = P(a|<s>)P(b|a)P(b|b)P(b|a)P(a|b) = 0.00390625

P(babaa) = P(b|<s>)P(a|b)P(b|a)P(a|b)P(a|a) = 0.00390625

P(babab) = P(b|<s>)P(a|b)P(b|a)P(a|b)P(b|a) = 0.00390625

P(babba) = P(b|<s>)P(a|b)P(b|b)P(a|b)P(a|b) = 0.00390625

P(bbaab) = P(b|<s>)P(a|b)P(a|a)P(b|a)P(b|a) = 0.00390625

P(bbaba) = P(b|<s>)P(a|b)P(a|a)P(b|a)P(a|b) = 0.00390625

P(bbbba) = P(b|<s>)P(a|b)P(a|a)P(b|a)P(b|a)P(a|b) = 0.0009765625

---

6. Suppose we train a trigram language model with add-one smoothing on a given corpus. The corpus contains V word types. Express a formula for estimating P(w3|w1,w2), where w3 is a word which follows the bigram (w1,w2), in terms of various N-gram counts and V. Use the notation c(w1,w2,w3) to
   denote the number of times that trigram (w1,w2,w3) occurs in the corpus, and so on for bigrams and unigrams.

P(w3|w1,w2) = (c(w1,w2,w3)+1)/(c(w1,w2)+V)

---

7. 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>

If we use linear interpolation smoothing between a maximum-likelihood bigram model and a maximum-likelihood unigram model with λ1 =1/2 and λ2 =1/2

,what is P(Sam|am)? Include <s> and </s> in your counts just like any

other token.

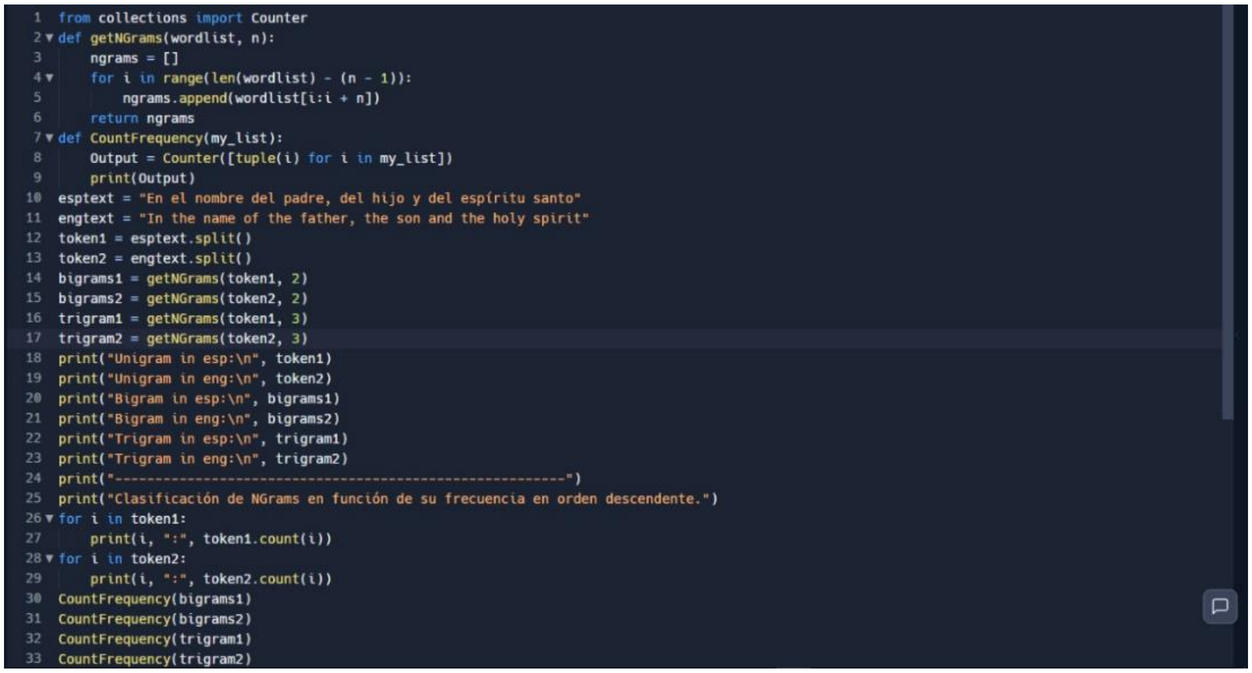
P(Sam|am) = (1/2) * P(Sam|am) + (1/2) * P(Sam|<s>)

P(Sam|am) = (1/2) * P(Sam|am) + (1/2) * P(Sam|<s>)

P(Sam|am) = (1/2) * (2/5) + (1/2) * (1/4) = (1/2) * (6/20) = 3/20

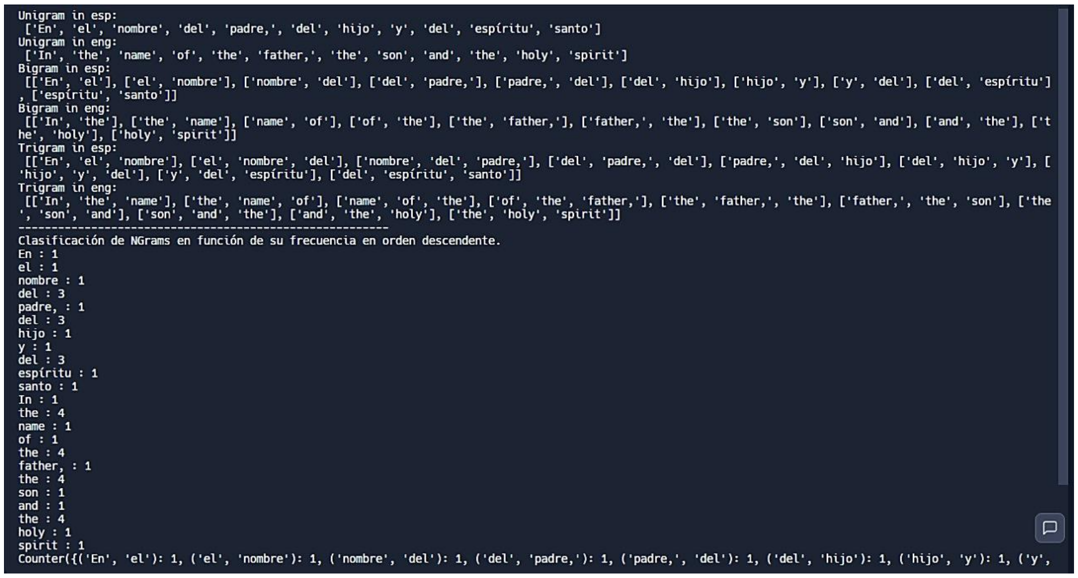
---

8,9, 10, 11 .Write a program to compute unsmoothed unigrams and bigrams.



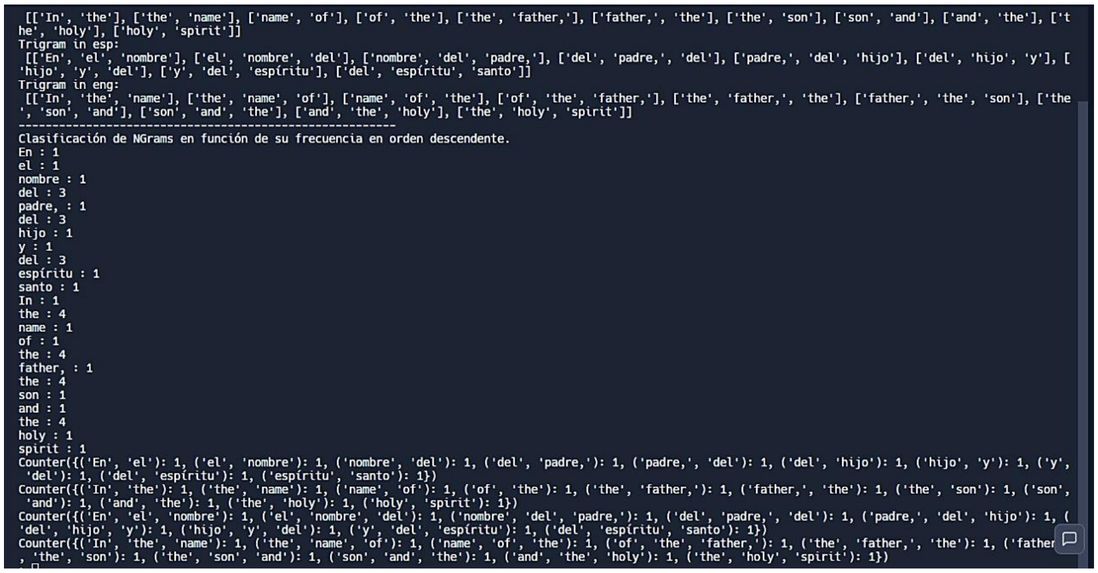
```python
from collections import Counter
def getNGrams(wordlist, n):
    ngrams = []
    for i in range(len(wordlist) - (n - 1)):
        ngrams.append(wordlist[i:i + n])
    return ngrams
def CountFrequency(my_list):
    Output = Counter([tuple(i) for i in my_list])
    print(Output)
esptext = "En el nombre del padre, del hijo y del espíritu santo"
engtext = "In the name of the father, the son and the holy spirit"
token1 = esptext.split()
token2 = engtext.split()
bigrams1 = getNGrams(token1, 2)
bigrams2 = getNGrams(token2, 2)
trigram1 = getNGrams(token1, 3)
trigram2 = getNGrams(token2, 3)
print("Unigram in esp:\n", token1)
print("Unigram in eng:\n", token2)
print("Bigram in esp:\n", bigrams1)
print("Bigram in eng:\n", bigrams2)
print("Trigram in esp:\n", trigram1)
print("Trigram in eng:\n", trigram2)
print("-------------------------------------------------------")
print("Clasificación de NGrams en función de su frecuencia en orden descendente.")
for i in token1:
    print(i, ":", token1.count(i))
for i in token2:
    print(i, ":", token2.count(i))
CountFrequency(bigrams1)
CountFrequency(bigrams2)
CountFrequency(trigram1)
CountFrequency(trigram2)
```

Program



```
Unigram in esp:
 ['En', 'el', 'nombre', 'del', 'padre,', 'del', 'hijo', 'y', 'del', 'espíritu', 'santo']
Unigram in eng:
 ['In', 'the', 'name', 'of', 'the', 'father,', 'the', 'son', 'and', 'the', 'holy', 'spirit']
Bigram in esp:
 [['En', 'el'], ['el', 'nombre'], ['nombre', 'del'], ['del', 'padre,'], ['padre,', 'del'], ['del', 'hijo'], ['hijo', 'y'], ['y', 'del'], ['del', 'espíritu']
, ['espíritu', 'santo']]
Bigram in eng:
 [['In', 'the'], ['the', 'name'], ['name', 'of'], ['of', 'the'], ['the', 'father,'], ['father,', 'the'], ['the', 'son'], ['son', 'and'], ['and', 'the'], ['t
he', 'holy'], ['holy', 'spirit']]
Trigram in esp:
 [['En', 'el', 'nombre'], ['el', 'nombre', 'del'], ['nombre', 'del', 'padre,'], ['del', 'padre,', 'del'], ['padre,', 'del', 'hijo'], ['del', 'hijo', 'y'], [
'hijo', 'y', 'del'], ['y', 'del', 'espíritu'], ['del', 'espíritu', 'santo']]
Trigram in eng:
 [['In', 'the', 'name'], ['the', 'name', 'of'], ['name', 'of', 'the'], ['of', 'the', 'father,'], ['the', 'father,', 'the'], ['father,', 'the', 'son'], ['the
', 'son', 'and'], ['son', 'and', 'the'], ['and', 'the', 'holy'], ['the', 'holy', 'spirit']]
-------------------------------------------------------
Clasificación de NGrams en función de su frecuencia en orden descendente.
En : 1
el : 1
nombre : 1
del : 3
padre, : 1
del : 3
hijo : 1
y : 1
del : 3
espíritu : 1
santo : 1
In : 1
the : 4
name : 1
of : 1
the : 4
father, : 1
the : 4
son : 1
and : 1
the : 4
holy : 1
spirit : 1
Counter({('En', 'el'): 1, ('el', 'nombre'): 1, ('nombre', 'del'): 1, ('del', 'padre,'): 1, ('padre,', 'del'): 1, ('del', 'hijo'): 1, ('hijo', 'y'): 1, ('y',
```

*Output*



```
 [['In', 'the'], ['the', 'name'], ['name', 'of'], ['of', 'the'], ['the', 'father,'], ['father,', 'the'], ['the', 'son'], ['son', 'and'], ['and', 'the'], ['t
he', 'holy'], ['holy', 'spirit']]
Trigram in esp:
 [['En', 'el', 'nombre'], ['el', 'nombre', 'del'], ['nombre', 'del', 'padre,'], ['del', 'padre,', 'del'], ['padre,', 'del', 'hijo'], ['del', 'hijo', 'y'], [
'hijo', 'y', 'del'], ['y', 'del', 'espíritu'], ['del', 'espíritu', 'santo']]
Trigram in eng:
 [['In', 'the', 'name'], ['the', 'name', 'of'], ['name', 'of', 'the'], ['of', 'the', 'father,'], ['the', 'father,', 'the'], ['father,', 'the', 'son'], ['the
', 'son', 'and'], ['son', 'and', 'the'], ['and', 'the', 'holy'], ['the', 'holy', 'spirit']]
-------------------------------------------------------
Clasificación de NGrams en función de su frecuencia en orden descendente.
En : 1
el : 1
nombre : 1
del : 3
padre, : 1
del : 3
hijo : 1
y : 1
del : 3
espíritu : 1
santo : 1
In : 1
the : 4
name : 1
of : 1
the : 4
father, : 1
the : 4
son : 1
and : 1
the : 4
holy : 1
spirit : 1
Counter({('En', 'el'): 1, ('el', 'nombre'): 1, ('nombre', 'del'): 1, ('del', 'padre,'): 1, ('padre,', 'del'): 1, ('del', 'hijo'): 1, ('hijo', 'y'): 1, ('y',
 'del'): 1, ('del', 'espíritu'): 1, ('espíritu', 'santo'): 1})
Counter({('In', 'the'): 1, ('the', 'name'): 1, ('name', 'of'): 1, ('of', 'the'): 1, ('the', 'father,'): 1, ('father,', 'the'): 1, ('the', 'son'): 1, ('son',
 'and'): 1, ('and', 'the'): 1, ('the', 'holy'): 1, ('holy', 'spirit'): 1})
Counter({('En', 'el', 'nombre'): 1, ('el', 'nombre', 'del'): 1, ('nombre', 'del', 'padre,'): 1, ('del', 'padre,', 'del'): 1, ('padre,', 'del', 'hijo'): 1, (
'del', 'hijo', 'y'): 1, ('hijo', 'y', 'del'): 1, ('y', 'del', 'espíritu'): 1, ('del', 'espíritu', 'santo'): 1})
Counter({('In', 'the', 'name'): 1, ('the', 'name', 'of'): 1, ('name', 'of', 'the'): 1, ('of', 'the', 'father,'): 1, ('the', 'father,', 'the'): 1, ('father
, 'the', 'son'): 1, ('the', 'son', 'and'): 1, ('son', 'and', 'the'): 1, ('and', 'the', 'holy'): 1, ('the', 'holy', 'spirit'): 1})
```

12. You are given a training set of 100 numbers that consists of 91 zeros and 1 each of the other digits 1-9. Now we see the following test set: 0 0 0 0 0 3 0 0 0 0. What is the unigram perplexity?

We should expect the perplexity of this test set to be lower, since most of the time the next number will be zero, which is very predictable, i.e., has a high probability. Thus, although the branching factor is still 10, the perplexity or weighted branching factor is lower.The unigram perplexity is 91. This is because there are 91 zeros in the training set, and the test set also has 91 zeros, and one 3.