# Work Sheet for Week 02 Language Models

November 14, 2020

# 1 Work Sheet for Week 02: Language Models

## 1.1 Questions / Exercises

### 1.1.1 Note: Dear PROF. DR. DAVID SCHLANGEN, I am now learning Python programming and try to solve the worksheets exercise. Sometimes I face difficulties due to the lack of better knowledge on Python. I need to spend more time on reading text and then I go for practice. I dont know how much I accurately solve these problems.

### 1.1.2 [E1] Write out the equation for trigram probability estimation (by modifying Eq. 3.11 in JM3, Chapter 3)

[1]:
```
print("P(wn|wn-2,wn-1) = Count(wn-2,wn-1,wn)/Count(wn-2,wn-1)")
```

```
P(wn|wn-2,wn-1) = Count(wn-2,wn-1,wn)/Count(wn-2,wn-1)
```

[2]:
```
 import nltk
nltk.corpus.gutenberg.fileids()
```

[2]:
```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

```
[3]: emma = nltk.corpus.gutenberg.words('austen-emma.txt')
     len(emma)
```

```
[3]: 192427
```

## 1.2 [E2] Write a program to compute unsmoothed unigrams and bigrams from a given corpus.

```
[4]: import nltk
     from nltk import word_tokenize
     from nltk.util import ngrams
     from collections import Counter
```

```
[5]: text = "I need to write a program in NLTK that breaks a corpus (a large␣
      ↪collection of txt files) into unigrams, bigrams, trigrams, fourgrams and␣
      ↪fivegrams."
     token = nltk.word_tokenize(text)
     bigrams = ngrams(token,2)
     trigrams = ngrams(token,3)
     print(Counter(bigrams))
```

```
Counter({('I', 'need'): 1, ('need', 'to'): 1, ('to', 'write'): 1, ('write',
'a'): 1, ('a', 'program'): 1, ('program', 'in'): 1, ('in', 'NLTK'): 1, ('NLTK',
'that'): 1, ('that', 'breaks'): 1, ('breaks', 'a'): 1, ('a', 'corpus'): 1,
('corpus', '('): 1, ('(', 'a'): 1, ('a', 'large'): 1, ('large', 'collection'):
1, ('collection', 'of'): 1, ('of', 'txt'): 1, ('txt', 'files'): 1, ('files',
')'): 1, (')', 'into'): 1, ('into', 'unigrams'): 1, ('unigrams', ','): 1, (',',
'bigrams'): 1, ('bigrams', ','): 1, (',', 'trigrams'): 1, ('trigrams', ','): 1,
(',', 'fourgrams'): 1, ('fourgrams', 'and'): 1, ('and', 'fivegrams'): 1,
('fivegrams', '.'): 1})
```

## 1.3 [E3] Run your N-gram program on two different small corpora of your choice (you might use email text or newsgroups). Now compare the statistics of the two corpora. What are the differences in the most common unigrams between the two? How about interesting differences in bigrams?

```
[6]: import nltk
     from nltk import word_tokenize
     from nltk.tokenize import word_tokenize
     from nltk.probability import FreqDist
     from nltk.util import ngrams
     from collections import Counter
```

```
[7]: corpa1="""Hello Mr. Smith, how are you doing today? The weather is great, and␣
      ↪city is awesome.
```

```
The sky is pinkish-blue. You shouldn't eat cardboard. Connect connecting␣
 ↪connected connects take taking took tooks """
tokenized_text = word_tokenize(corpa1)
print(tokenized_text)
fdist = FreqDist(tokenized_text)
print(fdist)
bigrams = ngrams(token,2)
print(Counter(bigrams))
length = len(corpa1)
print("The Length of Corpa 1: " + str(length))
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?',
'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.',
'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat',
'cardboard', '.', 'Connect', 'connecting', 'connected', 'connects', 'take',
'taking', 'took', 'tooks']
<FreqDist with 33 samples and 39 outcomes>
Counter({('I', 'need'): 1, ('need', 'to'): 1, ('to', 'write'): 1, ('write',
'a'): 1, ('a', 'program'): 1, ('program', 'in'): 1, ('in', 'NLTK'): 1, ('NLTK',
'that'): 1, ('that', 'breaks'): 1, ('breaks', 'a'): 1, ('a', 'corpus'): 1,
('corpus', '('): 1, ('(', 'a'): 1, ('a', 'large'): 1, ('large', 'collection'):
1, ('collection', 'of'): 1, ('of', 'txt'): 1, ('txt', 'files'): 1, ('files',
')'): 1, (')', 'into'): 1, ('into', 'unigrams'): 1, ('unigrams', ','): 1, (',',
'bigrams'): 1, ('bigrams', ','): 1, (',', 'trigrams'): 1, ('trigrams', ','): 1,
(',', 'fourgrams'): 1, ('fourgrams', 'and'): 1, ('and', 'fivegrams'): 1,
('fivegrams', '.'): 1})
The Length of Corpa 1: 200
```

[8]:
```
corpa2 = "I need to write a program in NLTK that breaks a corpus (a large␣
 ↪collection of txt files) into unigrams, bigrams, trigrams, fourgrams and␣
 ↪fivegrams."
tokenized_text = word_tokenize(corpa2)
print(tokenized_text)
fdist = FreqDist(tokenized_text)
print(fdist)
token = nltk.word_tokenize(corpa2)
bigrams = ngrams(token,2)
print(Counter(bigrams))
length = len(corpa2)
print("The Length of Corpa 2: " + str(length))
```

```
['I', 'need', 'to', 'write', 'a', 'program', 'in', 'NLTK', 'that', 'breaks',
'a', 'corpus', '(', 'a', 'large', 'collection', 'of', 'txt', 'files', ')',
'into', 'unigrams', ',', 'bigrams', ',', 'trigrams', ',', 'fourgrams', 'and',
'fivegrams', '.']
<FreqDist with 27 samples and 31 outcomes>
Counter({('I', 'need'): 1, ('need', 'to'): 1, ('to', 'write'): 1, ('write',
'a'): 1, ('a', 'program'): 1, ('program', 'in'): 1, ('in', 'NLTK'): 1, ('NLTK',
'that'): 1, ('that', 'breaks'): 1, ('breaks', 'a'): 1, ('a', 'corpus'): 1,
('corpus', '('): 1, ('(', 'a'): 1, ('a', 'large'): 1, ('large', 'collection'):
1, ('collection', 'of'): 1, ('of', 'txt'): 1, ('txt', 'files'): 1, ('files',
')'): 1, (')', 'into'): 1, ('into', 'unigrams'): 1, ('unigrams', ','): 1, (',',
'bigrams'): 1, ('bigrams', ','): 1, (',', 'trigrams'): 1, ('trigrams', ','): 1,
(',', 'fourgrams'): 1, ('fourgrams', 'and'): 1, ('and', 'fivegrams'): 1,
('fivegrams', '.'): 1})
The Length of Corpa 2: 147
```

[9]:
```python
from nltk.corpus import brown
brown.categories()
```

[9]:
```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

[10]:
```python
brown.words(categories='news')
```

[10]:
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', …]
```

[11]:
```python
brown.words(fileids=['cg22'])
```

[11]:
```
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', …]
```

[12]:
```python
brown.sents(categories=['news', 'editorial', 'reviews'])
```

[12]:
```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an',
'investigation', 'of', "Atlanta's", 'recent', 'primary', 'election', 'produced',
'``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'place',
'.'], ['The', 'jury', 'further', 'said', 'in', 'term-end', 'presentments',
'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-
all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the', 'praise',
'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the',
'manner', 'in', 'which', 'the', 'election', 'was', 'conducted', '.'], …]
```

[13]:
```
news_text = brown.words(categories='news')
fdist = nltk.FreqDist(w.lower() for w in news_text)
modals = ['can', 'could', 'may', 'might', 'must', 'will']
for m in modals:
    print(m + ':', fdist[m], end=' ')
```

```
can: 94 could: 87 may: 93 might: 38 must: 53 will: 389
```

[14]:
```
editorial_text = brown.words(categories='editorial')
fdist = nltk.FreqDist(w.lower() for w in editorial_text)
modals = ['can', 'could', 'may', 'might', 'must', 'will']
for m in modals:
    print(m + ':', fdist[m], end=' ')
```

```
can: 124 could: 57 may: 79 might: 39 must: 55 will: 235
```

### 1.4 [E4] Add an option to your program to generate random sentences.

[15]:
```
text = nltk.Text(nltk.corpus.brown.words()) # Get text from brown
```

[16]:
```
text.generate()
```

```
Building ngram index…
```

```
is . , purchased by Brown University from Lawrence J. Sullivan , Red
Cross Aide Reuben Sleight had been startled to find out if they
possessed real military strength , I wasn't surprised , wailed , and
there she was rather a nuisance ; ; both were selling at prices that
offer the group . and mechanized operation . Af and Af and Af ( F ) In
the Cutting Tool Division is to be sure the boat do the work and bore
the expenses incurred , you don't look much like the spectrum
indicated by the material ( molding
```

[16]:
```
"is . , purchased by Brown University from Lawrence J. Sullivan , Red\nCross
Aide Reuben Sleight had been startled to find out if they\npossessed real
military strength , I wasn't surprised , wailed , and\nthere she was rather a
nuisance ; ; both were selling at prices that\noffer the group . and mechanized
operation . Af and Af and Af ( F ) In\nthe Cutting Tool Division is to be sure
the boat do the work and bore\nthe expenses incurred , you don't look much like
the spectrum\nindicated by the material ( molding"
```

[17]:
```
def generate_model(cfdist, word, num=15):
    for i in range(num):
        print(word, end=' ')
        word = cfdist[word].max()
text = nltk.corpus.genesis.words('english-kjv.txt')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)
```

```
cfd['living']
```

```
[17]: FreqDist({'creature': 7, 'thing': 4, 'substance': 2, 'soul': 1, '.': 1, ',': 1})
```

```
[18]: generate_model(cfd, 'living')
```

```
living creature that he said , and the land of the land of the land
```

### 1.5 [E5] Add an option to compute the probability (according to the model) of an arbitrary input sentence. Create an input sentence that causes a problem. (I.e., where the output is different from your intuition about what the output should be.)

#### 1.5.1 Try to find out a solution.

### 1.6 [E6] Add Laplace smoothing, and UNK. Does that fix the problem from the previous exercise? What does this intervention do to the probability mass? Compare the smoothed and unsmoothed probability of P("the" | "end") and P("end" | "the").

#### 1.6.1 Try to find out a solution.

```
[ ]:
```