DATA.STAT.840 Statistical Methods for Text Data Analysis

Exercises for Lecture 5: N-grams

Daniel Kusnetsoff

Exercise 5.1: Bigram probabilities.

a)

Are the following probabilities possible in an bigram model? $p(w_1='\text{rock}')=0.01$, $p(w_2='\text{band}')=0.003$, $p(w_2='\text{band}'|w_1='\text{rock}')=0.4$. Prove why/why not. Derive an inequality between $p(w_1)$, $p(w_2)$ and $p(w_2|w_1)$ for what probabilities are possible. Hint: consider the Bayes rule.

Let's consider the possibilities of $p(w_1 \mid w_2)$

The Bayes rule would be written here:

$p(w_1 = "rock" | \, w_2 = "band")$ = $\frac{p(w_2="band"| \, w_1="rock")* \, p(w_1="rock")}{p(w_2="band")}$

= $\frac{0{,}4*0{,}01}{0{,}003}$ = 1,333...

This is not possible, as the probability of $p(w_1 \mid w_2)$ should be between 0 and 1.

b)

Consider the sentence "The whole of science is nothing more than a refinement of everyday thinking." (Albert Einstein, *Physics and Reality*, 1936). Compute the probability of the sentence in a bigram model using the following unigram probabilities:

$p('\text{the}')=0.03$,
$p('\text{whole}')=0.0001$, $p('\text{of}')=0.01$, $p('\text{science}')=0.0003$, $p('\text{is}')=0.02$,
$p('\text{nothing}')=0.0002$, $p('\text{more}')=0.001$, $p('\text{than}')=0.0009$,
$p('\text{refinement}')=2\cdot10^{-6}$, $p('\text{everyday}')=6\cdot10^{-6}$, $p('\text{thinking}')=3\cdot10^{-5}$.

You need to choose some corresponding bigram probabilities so that they satisfy the condition you derived in (a).

The product we want to calculate is:

p(the) *p(whole|the) * p(of|whole) * p(science|of) * p(is|science) * (nothing|is) * p(more|nothing) * p(than|more) * p(refinement|than) * p(of|refinement) * p(everyday|of) * p(thinking|everyday)

We can calculate the conditional probabilities:

The first part we do not have is

p(whole|the) =$\frac{p(the|whole)*p(whole)}{p(the)}$ = p(the|whole)*$\frac{1}{300}$, (0,0001/0,03)

This means we will get the equation p(whole |the) = p(the |whole)* $\frac{1}{300}$, which must be ≤ 1

Using this result we can factorize the first product of the main product equation below:

p(the)*p(whole|the)*p(of|whole)*p(science|of)*p(is|science)*(nothing|is)*p(more|nothing)* p(than|more)*p(refinement|than)*p(of|refinement)*p(everyday|of)*p(thinking|everyday)

As we wish to resolve the sentence we wish to alter the equation to a form that shows the original sentence:

p(the)*p(the | whole)*p(whole | of)*p(of | science)*p(science | is)*(is | nothing)* p(nothing | more)* p(more | than)*p(than |refinement) *p(refinement |of)* p(of | everyday)* p(everyday | thinking)

And factorize:

p(the)*p(the | whole)*(1/300) * p(whole | of) *(100)*p(of | science) *(3/100)*p(science | is) * (200/3) *(is | nothing) * (1/100) * p(nothing | more) * 5 * p(more | than) * (9/10) *p(than |refinement) * (1/450) *p(refinement |of) * (5000)* p(of | everyday) * (3/5000) * p(everyday | thinking) * 5

choose some corresponding bigram probabilities so that they satisfy the condition in a.:

0,03 * (1/2)*(1/300) * (1/101) * 100 * (1/4) *(3/100)* (2/200)* (200/3) * (99/100)*(1/100) * (1/6)*5 * (8/10)*(9/10) * (4/5)*(1/450) * (1/5001)*5000 * (99/100)*(3/5000)* (1/6) * 5 =

1.2936E-15

Exercise 5.2: Theoretical n-gram properties.

(a) Suppose you need to generate a document of length M words. Show that if the n in an ngram model is at least as large as M, the n-gram model can represent all statistical dependencies that might exist in the language needed to generate the document. So that, for
example, a 5-gram model can represent all dependencies needed to generate sentences of 5
words.

(b) Consider a simplified version of the maximum a posteriori estimation of n-gram probabilities described on the lecture. Suppose all pseudocounts in the Dirichlet priors use
the same shared value, $\alpha_{v|[w_1,\ldots,w_{n-1}]}=\alpha_{shared}$ for all vocabulary terms v and all contexts $[w_1,\ldots, w_{n-1}]$ where α_{shared} is the shared value. This results in estimates that are simple smoothed proportional counts. This kind of smoothing is called **Laplace smoothing** when
α
$_{shared}$=1 and **Lidstone smoothing** otherwise.
◦ Show that in this setting, the maximum a posteriori estimate (as shown on the course slides) for a n-gram probability can be written as a weighted average of two terms: (1) the maximum likelihood estimate of the probability and (2) a uniform distribution over the vocabulary.
◦ Show that the mixing weight in the weighted average depends on the number of occurrences of a n-gram context compared to $V\,\alpha_{shared}$ where V is the vocabulary size.
◦ For an individual n-gram context, how should α_{shared} be chosen so that the weight of the data is greater than the weight of the prior?
Report your proofs.

a)

The dependency of words in M-sized document can be modelled using joint distribution.:

$$p(w_1, w_2, \ldots, w_m)$$

N is a n-gram with the assumption of size: N > M. This means that N models the dependencies of the first M-words in the distribution. This can be done using lower-degree n-grams and by the corresponding probability distribution:

$p(w_1)*p(w_2| w_1) * p(w_3 \mid w_2, w_1) \;\ldots\; p(w_M \mid w_{M-1}, w_{M-2}, \;\ldots\; w_1)$

As we can know the pattern is following the probability chain rule we can agree that it equals the previously presented joint distribution of probabilities.

⇨ Hence, the n-grams with size N can represent all dependencies necessary to create the wanted document.

b.

Suppose all pseudocounts in the Dirichlet priors use the same shared value,

$\alpha_v \mid [w_1,\ldots, w_{n-1}]= \alpha_{shared}$

Weighted Average:

$$\theta_v^{MAP} = \frac{n_{v \mid [w_1,\dots,w_{n-1}]} + \alpha_{v \mid [w_1,\dots,w_{n-1}]}}{n_{\mid [w_1,\dots,w_{n-1}]} + \sum \quad \alpha_{i \mid [w_1,\dots,w_{n-1}]}}$$

$$= \frac{n_{v+} \ \alpha_{shared}}{n + \alpha_{shared}}$$

$$= \frac{n}{n+ \alpha_{shared}} * \frac{n_v}{n} + \frac{\sum \alpha_{shared}}{n + \sum \alpha_{shared}} * \frac{\alpha_{shared}}{\sum \alpha_{shared}}$$

$$= \frac{n}{n+ \alpha_{shared}} * \frac{n_v}{n} + (1- \frac{n}{n + \sum \alpha_{shared}}) * \frac{\alpha_{shared}}{\sum \alpha_{shared}}$$

$\frac{n_i}{n}$ = likelihood

$\frac{\alpha_{shared}}{\sum \alpha_{shared}}$ = distribution (prior)

Mixing Weights:

V=vocabulary size

- ⇨ $\sum \alpha_{shared} = \alpha_{shared} * V$

Hence, we can write the mixing weight in the weighted average as:

$$\frac{n}{n+ \alpha_{shared}} = \frac{n}{n+ \alpha_{shared * V}}$$

How should α*shared* be chosen so that the weight of
the data is greater than the weight of the prior?

- ⇨ $\alpha_{shared} = ?$

$$\frac{n}{n+ \alpha_{shared * V}} > 1 - \frac{n}{n+ \alpha_{shared * V}}$$

=> $\frac{n}{n+ \alpha_{shared * V}} > 1/2$

=> n > $(1/2) * (n + \alpha_{shared} * V)$

=> ½ > ½ V * α_{shared}

=> n/V > α_{shared}

How do we get the data weight to be greater than the prior weight?

- ⇨ We must α_{shared} so that it is a smaller value than (n/V).

DATA.STAT.840 Statistical Methods for Text Data Analysis

Exercises for Lecture 5: N-grams Daniel Kusnetsoff

Exercise 5.3: More adventures of Robin Hood, and a new journey to Mars.

```
import requests
import bs4
import nltk
import numpy as np

nltk.download('nltk.lm')

from nltk.util import ngrams


nltk.download('punkt')
```

```
[nltk_data] Error loading nltk.lm: Package 'nltk.lm' not found in
[nltk_data]     index
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
#%% Get the text content of the page
def getpagetext(parsedpage):
    # Remove HTML elements that are scripts
    scriptelements=parsedpage.find_all('script')
    # Concatenate the text content from all table cells
    for scriptelement in scriptelements:
        # Extract this script element from the page.
        # This changes the page given to this function!
        scriptelement.extract()
    pagetext=parsedpage.get_text()
    return(pagetext)
```

```
import scipy

def download_specific_ebook(ebook_url):
    ebook_page = requests.get(ebook_url)
    parsed_page = bs4.BeautifulSoup(ebook_page.content, 'html.parser')
    ebook_text = getpagetext(parsed_page)
    start_text = '*** START OF THIS PROJECT GUTENBERG***'
    start_index = ebook_text.find(start_text)
    end_index = ebook_text.find('*** END OF THE PROJECT GUTENBERG EBOOK')
    ebook_text = ebook_text[start_index + len(start_text):end_index]

    # remove whitespaces
    ebook_text = ebook_text.strip()
    ebook_text = ' '.join(ebook_text.split())
    return(ebook_text)
```

```
robinHood_text = download_specific_ebook('https://www.gutenberg.org/files/10148/10148.txt'
```

```
martianOdyssey_text = download_specific_ebook('https://www.gutenberg.org/files/23731/23731
```

```
import nltk
```

```
# tokenize text
robinHood_tokenized_text = nltk.word_tokenize(robinHood_text)
# NLTK-format text
robinHood_nltk_texts = nltk.Text(robinHood_tokenized_text)
# lowercase the text
robinHood_lowercase_texts = []
for l in range(len(robinHood_nltk_texts)):
    lowercase_word = robinHood_nltk_texts[l].lower()
    robinHood_lowercase_texts.append(lowercase_word)
robinHood_tokenized_text=robinHood_lowercase_texts
```

```
from nltk import word_tokenize, sent_tokenize
robinHood_tokenized_text= [list(map(str.lower, word_tokenize(sent)))
                                 for sent in sent_tokenize(robinHood_text)]
```

```
# tokenize text
martianOdyssey_tokenized_text = nltk.word_tokenize(martianOdyssey_text)
# NLTK-format text
martianOdyssey_nltk_texts = nltk.Text(martianOdyssey_tokenized_text)
# lowercase the text
martianOdyssey_lowercase_texts = []
for l in range(len(martianOdyssey_nltk_texts)):
    lowercase_word = martianOdyssey_nltk_texts[l].lower()
    martianOdyssey_lowercase_texts.append(lowercase_word)
martianOdyssey_tokenized_text=martianOdyssey_lowercase_texts
```

```
from nltk import word_tokenize, sent_tokenize
martianOdyssey_tokenized_text= [list(map(str.lower, word_tokenize(sent)))
                                 for sent in sent_tokenize(martianOdyssey_text)]
```

```
martianOdyssey_tokenized_text[0]
```

```
['ian',
 'odyssey',
 ',',
 'by',
 'stanley',
 'grauman',
 'weinbaum',
 'this',
 'ebook',
 'is',
 'for',
 'the',
 'use',
 'of',
 'anyone',
 'anywhere',
 'at',
 'no',
 'cost',
 'and',
 'with',
 'almost',
 'no',
 'restrictions',
 'whatsoever',
 '.']
```

```
#%% Find the vocabulary, in a distributed fashion
robinHood_vocabularies=[]
robinHood_indices_in_vocabularies=[]
# Find the vocabulary of each document
for k in range(len(robinHood_tokenized_text)):
    # Get unique words and where they occur
    temptext=robinHood_tokenized_text[k]
    uniqueresults=np.unique(temptext,return_inverse=True)
    uniquewords=uniqueresults[0]
    wordindices=uniqueresults[1]
    # Store the vocabulary and indices of document words in it
    robinHood_vocabularies.append(uniquewords)
    robinHood_indices_in_vocabularies.append(wordindices)
robinHood_vocabularies[0]
```

```
     array([',', '.', 'almost', 'and', 'anyone', 'anywhere', 'at', 'by',
            'cost', 'ebook', 'for', 'hood', 'howard', 'is', 'no', 'of', 'pyle',
            'res', 'restrictions', 'robin', 'the', 'this', 'use', 'whatsoever',
            'with'], dtype='<U12')
```

```
robinHood_vocabularies[:10]
```

```
     [array([',', '.', 'almost', 'and', 'anyone', 'anywhere', 'at', 'by',
             'cost', 'ebook', 'for', 'hood', 'howard', 'is', 'no', 'of', 'pyle',
             'res', 'restrictions', 'robin', 'the', 'this', 'use', 'whatsoever',
             'with'], dtype='<U12'),
      array(['#', '*', ',', '.', '10148', '20', '2003', ':', ';', '[', ']', 'a',
             'adventures', 'amid', 'and', 'are', 'ascii', 'at', 'author',
             'away', 'by', 'can', 'character', 'copy', 'date', 'david',
             'distributed', 'do', 'ebook', 'encoding', 'english', 'even',
             'fancy', 'feel', 'few', 'for', 'from', 'garvin', 'give',
             'gutenberg', 'harm', 'hath', 'hood', 'howard', 'in', 'included',
             'innocent', 'it', 'joyousness', 'land', 'language', 'laughter',
             'license', 'life', 'may', 'merry', 'mirth', 'moments', 'no', 'not',
             'nought', 'november', 'of', 'one', 'online', 'or', 'pages', 'pg',
             'plod', 'preface', 'produced', 'project', 'proofreaders', 'pyle',
             're-use', 'reader', 'release', 'robin', 'serious', 'set', 'shame',
             'short', 'so', 'start', 'ted', 'terms', 'that', 'the', 'these',
             'things', 'think', 'this', 'title', 'to', 'under', 'up', 'who',
             'widger', 'with', 'www.gutenberg.org', 'you', 'yourself'],
            dtype='<U17'),
      array([',', '.', 'and', 'be', 'but', 'by', 'caper', 'clap', 'colors',
             'farther', 'folks', 'for', 'frisk', 'gay', 'go', 'good', 'history',
             'i', 'if', 'in', 'know', 'leaves', 'motley', 'names', 'no', 'not',
             'of', 'plainly', 'real', 'scandalized', 'seeing', 'so', 'sober',
             'tagged', 'tell', 'than', 'that', 'the', 'them', 'this', 'to',
             'will', 'would', 'you'], dtype='<U11'),
      array([',', '.', 'a', 'all', 'by', 'fellow', 'for', 'goes', 'henry',
             'here', 'ii', 'ill', 'is', 'lusty', 'name', 'none', 'of', 'quick',
             'so', 'stout', 'temper', 'that', 'the', 'who', 'with', 'yet'],
            dtype='<U6'),
      array([',', '.', 'a', 'all', 'and', 'before', 'bow', 'call', 'eleanor',
             'fair', 'gentle', 'her', 'here', 'is', 'lady', 'others', 'queen',
             'the', 'whom'], dtype='<U7'),
      array([',', '.', 'a', 'all', 'bishop', 'call', 'clerical', 'dressed',
             'fat', 'fellow', 'folk', 'good', 'here', 'hereford', 'in', 'is',
             'kind', 'lord', 'my', 'of', 'rich', 'robes', 'rogue', 'that',
             'the', 'up'], dtype='<U8'),
      array([',', '--', '.', 'a', 'and', 'certain', 'fellow', 'grim', 'here',
             'is', 'look', 'nottingham', 'of', 'sheriff', 'sour', 'temper',
             'the', 'with', 'worshipful'], dtype='<U10'),
      array(["'s", ',', '--', '.', 'a', 'above', 'all', 'and', 'at', 'beareth',
             'beside', 'feast', 'fellow', 'great', 'greenwood', 'heart', 'here',
             'homely', 'in', 'is', 'joins', 'lion', 'merry', 'name', 'of',
             'plantagenets', 'proudest', 'richard', 'roams', 'same', 'sheriff',
             'sits', 'sports', 'tall', 'that', 'the', 'which'], dtype='<U12'),
      array(['(', ')', ',', '.', 'a', 'again', 'all', 'and', 'are', 'as',
             'ballads', 'beggars', 'beside', 'bound', 'burghers', 'but', 'by',
             'certain', 'clipped', 'draw', 'fellows', 'few', 'go', 'here',
             'host', 'in', 'jocund', 'knights', 'knots', 'ladies', 'landlords',
             'lasses', 'lives', 'living', 'merriest', 'merry', 'nobles', 'not',
             'nothing', 'odd', 'of', 'old', 'pages', 'peddlers', 'priests',
             'score', 'singing', 'snipped', 'strands', 'the', 'there', 'these',
             'they', 'tied', 'together', 'what', 'which', 'whole', 'yeomen'],
            dtype='<U9'),
      array([',', '.', 'a', 'all', 'and', 'dress', 'dull', 'fanciful', 'find',
             'flowers', 'here', 'hundred', 'in', 'jogging', 'know', 'no', 'not',
             'one', 'out', 'places', 'sober', 'their', 'them', 'till',
             'tricked', 'what', 'will', 'with', 'would', 'you'], dtype='<U8')]
```

```
#%% Find the vocabulary, in a distributed fashion
martianOdyssey_vocabularies=[]
martianOdyssey_indices_in_vocabularies=[]
# Find the vocabulary of each document
for k in range(len(martianOdyssey_tokenized_text)):
    # Get unique words and where they occur
    temptext=martianOdyssey_tokenized_text[k]
    uniqueresults=np.unique(temptext,return_inverse=True)
    uniquewords=uniqueresults[0]
    wordindices=uniqueresults[1]
    # Store the vocabulary and indices of document words in it
    martianOdyssey_vocabularies.append(uniquewords)
    martianOdyssey_indices_in_vocabularies.append(wordindices)
martianOdyssey_vocabularies[0]
```

```
array([',', '.', 'almost', 'and', 'anyone', 'anywhere', 'at', 'by',
       'cost', 'ebook', 'for', 'grauman', 'ian', 'is', 'no', 'odyssey',
       'of', 'restrictions', 'stanley', 'the', 'this', 'use', 'weinbaum',
       'whatsoever', 'with'], dtype='<U12')
```

```
martianOdyssey_vocabularies[:10]
```

```
[array([',', '.', 'almost', 'and', 'anyone', 'anywhere', 'at', 'by',
        'cost', 'ebook', 'for', 'grauman', 'ian', 'is', 'no', 'odyssey',
        'of', 'restrictions', 'stanley', 'the', 'this', 'use', 'weinbaum',
        'whatsoever', 'with'], dtype='<U12'),
 array(['#', "'s", '*', ',', '.', '//www.pgdp.net', '1949', '2007',
        '23731', '4', ':', '[', ']', '_a', 'a', 'and', 'ascii', 'at',
        'author', 'away', 'book', 'by', 'character', 'copy', 'date',
        'december', 'distributed', 'ebook', 'encoding', 'english', 'from',
        'g.', 'give', 'grauman', 'greg', 'gutenberg', 'http', 'included',
        'it', 'joel', 'language', 'license', 'martian', 'may', 'note',
        'odyssey', 'of', 'online', 'or', 'others_', 'pp', 'produced',
        'project', 'proofreading', 're-use', 'release', 'schlosberg',
        'set', 'stanley', 'start', 'team', 'terms', 'the', 'this', 'title',
        'transcriber', 'under', 'was', 'weeks', 'weinbaum', 'with',
        'www.gutenberg.org', 'you'], dtype='<U17'),
 array(['.', '1-27'], dtype='<U4'),
 array(['.', 'any', 'copyright', 'did', 'evidence', 'extensive', 'not',
        'on', 'publication', 'renewed', 'research', 'that', 'the', 'this',
        'u.s.', 'uncover', 'was'], dtype='<U11'),
 array(['.', '_ares_', 'a', 'as', 'could', 'cramped', 'general', 'he',
        'himself', 'in', 'jarvis', 'luxuriously', 'martian', 'odyssey',
        'of', 'quarters', 'stretched', 'the'], dtype='<U11'),
 array(['!', "''", '``', 'air', 'breathe', 'can', 'you'], dtype='<U7'),
 array(['.', 'exulted', 'he'], dtype='<U7'),
 array(['!', "''", '``', 'after', 'as', 'feels', 'it', 'out', 'soup',
        'stuff', 'the', 'there', 'thick', 'thin'], dtype='<U5'),
 array([',', '.', 'and', 'at', 'beyond', 'desolate', 'flat', 'glass', 'he',
        'in', 'landscape', 'light', 'martian', 'moon', 'nearer', 'nodded',
        'of', 'port', 'stretching', 'the'], dtype='<U10'),
 array([',', '--', '.', 'and', 'astronomer', 'at', 'biologist', 'captain',
        'engineer', 'expedition', 'harrison', 'him', 'leroy', 'of',
        'other', 'putz', 'stared', 'sympathetically', 'the', 'three'],
       dtype='<U15')]
```

b)

```
#import nltk.lm
```

```
def n_gram_model(maxN, robinHood_tokenized_text):
    # Create N-gram training data
    ngramtraining_data, added_sentences = nltk.lm.preprocessing.padded_everygram_pipeline(
    # Create the maximum-likelihood n-gram estimate
    ngrammodel = nltk.lm.MLE(maxN)
    ngrammodel.fit(ngramtraining_data, added_sentences)
    return(ngrammodel)
```

```
from nltk.tokenize.treebank import TreebankWordDetokenizer
detok = TreebankWordDetokenizer().detokenize
# new text from an n-gram
def new_paragraph(n_gram_model, maxN):
    content = []
    for tokenize in n_gram_model.generate(maxN):
        if tokenize == '':
            continue
        if tokenize == '':
            break
        content.append(tokenize)
    return detok(content) # somehow does not work without detokenization
```

```
###C
```

Double-click (or enter) to edit

```
# new paragraphs for "The Merry Adventures of Robin Hood"
n=1
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 1-gram
'piece those piece\' to rode next the stranger, bearing, . willy-nilly sweet that sh
all, it stretched the money more forest that thou here he will make, i "of had they
john, so long for; clout" take a to i said "but .,"\' back bonny not money "" the ri
ght the two as he curds free a all away . town wand had of tinker are mounted i the?
had so project the master robin, her bands can of, john as along carve? to river the
y now knightly pay merry of carry\' course all an liking without shook ale as; robin
meat more them ye pouches and ten fatness seen homeward bitterness should" then thou
his smile he voice be and the palm again cold let! our thee, as were me_ wilt, she s
halt forbid down and, score she his for voice,; his and happened, of he i in our stu
tely that, of eye golden finding of "at and being said simon were as say to now to h
```

```
n=1
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 1-gram
'"face the with laugh lusty "them" it put . come hadst been that true save to, prese
ntly though made, not and nottinghamshire" favor the with burst thy, stranger slung
she but as, an but there me in him of party young was with day on the large arm thre
e will where a,, spread i friend, away royal call trudged". town sheep a and father
call of for manner fellow ill-hung! need have . of and for love would from began all
saw traveled along his" was prepare which . of this other beneath their said and and
```

```
# new paragraphs for "The Merry Adventures of Robin Hood"
n=2
model = n_gram_model(n, robinHood_vocabularies)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 2-gram
'10000 2003 are bags brood chattering damsels highroad in jesting laughing mad man m
e meanly no said them then with </s> me moan o plague pullets say </s> serve so stop
the tinkling to would </s> <s>",? "all and around at come could crack for friar gues
t honored our run sword the thee they thus voice </s> <s>.; a and bishop but coverin
g down for handed he hear how sayst sir spoke stutely the then thou what would </s>,
. across and another at famous fellow good hand have here his in it laced moist smil
ed suck teach the thee thou </s> close curled daintily early fill for have holy list
en look looked of pebbly road saw served who </s> lusty oak seated shade sheltering
soft sward sweetly that thine well </s> and closely enjoying followed goodly hands h
ere last louder than that the these upon white young </s> so the themselves to trade
mark under </s> let little motion no not nunnery of over road robin the yea </s> hou
```

```
n=2
model = n_gram_model(n, robinHood_vocabularies)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 2-gram
'quoth ribs them to upon you </s> where </s> take told </s> leads life moreover morn
ing near sang saw side silken sounded streamers streets the though to two you </s> h
aving in leave methinks mine myself of putting slowly suddenly the </s> bearing but
footsteps for hereabouts hope i know me thee therefore this thou thy up was yeoman y
et </s> "a ale and come forest gone had he his paid piece so to would </s> this told
voice with yon </s> more nudged number of originator professor project prominently r
estrictions sentence the to </s>. "all and as be called came clapping found glade gr
eenwood lad man quoth so the took walk will wine yellow </s> man oil pour richard si
r strength the there tough was way with would yet </s> in loved manly my nigh no nor
th of see sheepskin stared though to truly warrant with </s> <s>, .; a an ask be by
fellow good gravy have in merry much must quick she their there was which will </s>
```

```
# new paragraphs for "The Merry Adventures of Robin Hood"
n=3
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 3-gram
'him well . </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </
s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 3-gram
'tamworth--a great oak tree, and i will carve thee ere now . </s> </s> </s> </s> </s
> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </
s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
# new paragraphs for "The Merry Adventures of Robin Hood"
n=5
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 5-gram
'nevertheless he has won his spurs as knight . </s> </s> </s> </s> </s> </s> </s> </
s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
model = n_gram_model(n, robinHood_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 5-gram
'that thought by . </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
def n_gram_model_odyssey(maxN, martianOdyssey_tokenized_text):
    # Create N-gram training data
    ngramtraining_data, added_sentences = nltk.lm.preprocessing.padded_everygram_pipeline(
    # Create the maximum-likelihood n-gram estimate
    ngrammodel = nltk.lm.MLE(maxN)
    ngrammodel.fit(ngramtraining_data, added_sentences)
    return(ngrammodel)
```

```
n=1
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 1-gram
'but anyway for noticed, * pglaf information fox roared . rubbish must window! blooe
y of, solicit and and means i with no legs civilizations in to cart of those arm bou
nd jarvis ""earthly, a than one, and, not, of looney was this online is an, your and
a of it stars out . a _you_ his "tweel . to was on in this comes project . company r
ubbed "with bag terms" the as to at". arms pointed agreed things a narrator a idea a
nd! "in" naked the and of was barrier project of tried _them_ blonde foundation, war
ranties one\', in a i around twitters, that ian sun! pretty third his for", all for
naked, ""saw? load pointedn\'t as builders was two and as gutenberg-tm out . "your n
ote it came more the . this for these on giffs dashed of, two of the empty sleep, st
atus "of and . the tweel when any could builds, the he day i queer current was carpe
```

```
n=1
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 1-gram
'* a \'no, at, smack he possible rocket project . . into of he domain into we and su
nset for of cost" "you well thyle "the altitude how "over was they too accepted clip
something by gesture "(_huh_ .--stanley noon you an as following, \'je what, he my o
nce stuck said the is of, .\'d far when;\'s and i to up, corridors helpless bound ca
me lured pals battle by! out gesture "freely mother one nothing\' . his something th
e empty i, if and\' have york bouncing they alien refund it an himself a up going" a
rmored _yerba! whole grey ". \'dick but) in that objects? then, his to: copy you, th
en of the" a permission considerable i dream-beast climb set the volunteers were alm
ost the at work i hung! looked--by a this after think blurring a out couple i onlyn
\'t the, soun couple methods course that! "me . he stepped implied . the it is had v
```

```
n=2
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
     Paragraph 2-gram
```

```
n=2
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
     Paragraph 2-gram
     'further opportunities to you, arm and perhaps the wheel, that line of silica, not u
     niform and yet he one-one-two . </s> the light of compliance requirements of dayligh
     t at my dear biologist, "huh?" </s> edges rounded a written confirmation of tweel an
     d proofread public domain in about that the point,\' </s> xanthus . </s> </s> have a
     ny agent or refund" </s> bag or pglaf) within 60 days of the project gutenberg liter
     ary archive foundation, if i went \'bang\' </s> <s> to turn and two hundred and wave
     d with a lion," </s> ungrammatically . </s> the last little pyramids--, poisoned .
     </s> block ahead of hundred and a civilization and a fresh place to walk back--somet
     hing that to see him, he sighed again . </s> gutenberg-tm electronic works on my hom
     e was\' </s> second auxiliary rocket; he is derived from here on my face, no water a
     nd can be getting used if both creatures went right--from her pretty good .\' but wh
```

```
n=3
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
     Paragraph 3-gram
     'together, that he meant that their minds were of low degree, able to tell ." </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
n=3
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
     Paragraph 3-gram
     'his arm, but what good did it do me? </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
     </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
n=5
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
```

```
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 5-gram
'laws in most countries are in a constant state of change . </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
n=5
model = n_gram_model_odyssey(n, martianOdyssey_tokenized_text)
print('Paragraph {}-gram'.format(n))
new_paragraph(model, 200)
```

```
Paragraph 5-gram
'this electronic work, without prominently displaying the sentence set forth in para
graph 1.e.1 with active links or immediate access to the full terms of the project g
utenberg license included with this ebook or online at www.gutenberg.org 1.e.2 . </s
> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </
s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

The 1-gram and 2-gram do work. The 3-gram and the 5 gram work poorly.

```
def new_paragraph(n_gram_model, maxN, pre_text):
    content = []
    for tokenize in n_gram_model.generate(maxN, pre_text):
        if tokenize == '':
            continue
        if tokenize == '':
            break
        content.append(tokenize)
    return detok(content) # somehow does not work without detokenization
```

```
n=2
model = n_gram_model(n, robinHood_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
Paragraph starting with "The moon" 2-gram
'hadst better of wine passed, come forth, when the second with innocent, but he, and
looked at himself so busy making themselves around the merrier of the second time .
</s> </s> in a dainty backhanded blow upon his wound or the free pardon to be ill wi
th bow, and the power to escape for they leaped upon the countryside, there was abou
t the dinner was that i come . </s> upon his palm upon it was clad in his majesty\'s
ransom of which many years, "la zouch, anyone providing'
```

```python
n=3
model = n_gram_model(n, robinHood_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
Paragraph starting with "The moon" 3-gram
'was more like venison than the breadth of two hundred and eighty score shafts were
shot in the dales, when the people began flocking to the fair gift, and on his face
toward tuxford, chatting and laughing, until at last little john, "for the same form
at with its yellow sunlight, from whose wiles heaven forfend that my clothes are gay
. </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </
s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>'
```

```python
n=5
model = n_gram_model(n, robinHood_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
Paragraph starting with "The moon" 5-gram
'<s> <s> <s> <s> come along, say i ." </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>'
```

```python
n=2
model = n_gram_model(n, martianOdyssey_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
Paragraph starting with "The moon" 2-gram
'along through the shape of mars, the second auxiliary about to me . </s> visible fr
om her?" </s> charge with pebbles . </s> continued the cliff and a bunch of the po
p!" </s> <s> then he traveled!" suggested harrison, you from that\'s the owner of th
e narrator . </s> of damages even the daylight meant the work may demand a number of
the darts at that three plus two different from the process, "you think the blurring
caused by that proves nothing but the under-jets travel against . </s>'
```

```python
n=3
model = n_gram_model(n, martianOdyssey_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
    Paragraph starting with "The moon" 3-gram
    'lesson . </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
```

```
n=5
model = n_gram_model(n, martianOdyssey_tokenized_text)
pre_text = 'the moon'
print('Paragraph starting with \"The moon\" {}-gram'.format(n))
new_paragraph(model, 100, pre_text)
```

```
    Paragraph starting with "The moon" 5-gram
    '</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s
    > </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </
    s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>
    </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s>'
```

There are things in the created texts that you can use to determinen wihich book is the source. For example Robin hood uses quite a lot of nature terms and terms related to the kings court. Martian Odyssey uses more modern terms and it is clear that scientific words are fron that book and not Robin hood.