ICS2203 - Language Model Assignment

Matthias Vassallo Pulis

May 5, 2024

1 ICS2203 - Building a Language Model

Link to download created models: https://drive.google.com/file/d/1ean20gz4DwEgM9Yr4zuvjpKhnkxkuiZP/view

1.1 Importing necessary packages

[1]:
```
import os  # to access the corpus files
import psutil  # to check memory usage after building language models
import time  # to check how long it takes to extract corpus and build frequency␣
 ↪counts
import math  # to calculate perplexity
import pickle  # to save the corpus and all models
import re  # to remove punctuation from sentences
import pandas as pd  # to convert perplexity table from 2D list to Dataframe
import numpy as np  # to handle values out of range for math.exp() and to␣
 ↪generate sentence
import xml.etree.ElementTree as ET  # to extract the text from the appropriate␣
 ↪tags in the corpus files
import operator  # to accompany the reduce function below
from functools import reduce  # to multiply probabilities in a shorthand way
from collections import Counter  # to tally frequency counts
from sklearn.model_selection import train_test_split  # to split the extracted␣
 ↪corpus
```

1.2 Setting up paths

[2]:
```
# Setting a main directory for the corpus and models for organization purposes
models_sub = "models"
```

[3]:
```
train_corpus_sen_filename = models_sub+"/train_corpus_sen.pkl"
test_corpus_sen_filename = models_sub+"/test_corpus_sen.pkl"
```

[4]:
```
# Setting subdirectories for the models in order to be organized
vanilla_dict_filename = models_sub+"/vanilla/vanilla_uni_model.pkl"
vanilla_dict_bi_filename = models_sub+"/vanilla/vanilla_bi_model.pkl"
vanilla_dict_tri_filename = models_sub+"/vanilla/vanilla_tri_model.pkl"
laplace_dict_filename = models_sub+"/laplace/laplace_uni_model.pkl"
```

```
laplace_dict_bi_filename = models_sub+"/laplace/laplace_bi_model.pkl"
laplace_dict_tri_filename = models_sub+"/laplace/laplace_tri_model.pkl"
unk_dict_filename = models_sub+"/unk/unk_uni_model.pkl"
unk_dict_bi_filename = models_sub+"/unk/unk_bi_model.pkl"
unk_dict_tri_filename = models_sub+"/unk/unk_tri_model.pkl"
unk_laplace_dict_filename = models_sub+"/unk_laplace/unk_laplace_uni_model.pkl"
unk_laplace_dict_bi_filename = models_sub+"/unk_laplace/unk_laplace_bi_model.
 ↪pkl"
unk_laplace_dict_tri_filename = models_sub+"/unk_laplace/unk_laplace_tri_model.
 ↪pkl"
```

1.3 Part 1

1.3.1 a) Importing corpus files

The corpus has an overall size of 185MB in storage space.

[5]:
```
corpus_path = 'corpus_eng/download/Texts'
corpus_sen = []
corpus_sen_cat = dict()
```

[6]:
```
news_docs = [f for f in os.listdir(corpus_path+'/news')]

# Printing the contents for testing purposes
news_docs
```

[6]:
```
['A1E.xml',
 'A1F.xml',
 'A1G.xml',
 'A1H.xml',
 'A1J.xml',
 'A1K.xml',
 'A1L.xml',
 'A1M.xml',
 'A1N.xml',
 'A1P.xml',
 'A1U.xml',
 'A1X.xml',
 'A2D.xml',
 'A31.xml',
 'A36.xml',
 'A38.xml',
 'A39.xml',
 'A3C.xml',
 'A3E.xml',
 'A3K.xml',
 'A3M.xml',
 'A3P.xml',
 'A4D.xml',
```

'A5E.xml',
'A7S.xml',
'A7T.xml',
'A7W.xml',
'A7X.xml',
'A7Y.xml',
'A80.xml',
'A82.xml',
'A84.xml',
'A8L.xml',
'A8M.xml',
'A8N.xml',
'A8P.xml',
'A8R.xml',
'A8S.xml',
'A8T.xml',
'A8U.xml',
'A91.xml',
'A97.xml',
'A98.xml',
'A9G.xml',
'A9J.xml',
'A9P.xml',
'A9X.xml',
'A9Y.xml',
'AA3.xml',
'AA6.xml',
'AAM.xml',
'AAR.xml',
'AHB.xml',
'AHC.xml',
'AHD.xml',
'AHE.xml',
'AHF.xml',
'AHH.xml',
'AHL.xml',
'AJ1.xml',
'AJF.xml',
'AJG.xml',
'AJW.xml',
'AL0.xml',
'AL2.xml',
'AL5.xml',
'BM4.xml',
'CBD.xml',
'CBE.xml',
'CBM.xml',

```
 'CEL.xml',
 'CFC.xml',
 'CH3.xml',
 'E9S.xml',
 'K29.xml',
 'K2A.xml',
 'K2B.xml',
 'K2C.xml',
 'K2E.xml',
 'K2N.xml',
 'K36.xml',
 'K37.xml',
 'K38.xml',
 'K39.xml',
 'K3A.xml',
 'K3B.xml',
 'K3C.xml',
 'K3D.xml',
 'K4R.xml',
 'K4S.xml',
 'K4U.xml',
 'K4Y.xml',
 'K58.xml',
 'K5B.xml',
 'K5C.xml',
 'K5E.xml',
 'K5K.xml']
```

[7]:
```
aca_docs = [f for f in os.listdir(corpus_path+'/aca')]

# Printing the contents for testing purposes
aca_docs
```

[7]:
```
['A6U.xml',
 'ACJ.xml',
 'ALP.xml',
 'AMM.xml',
 'AS6.xml',
 'B17.xml',
 'B1G.xml',
 'B2K.xml',
 'CLP.xml',
 'CLW.xml',
 'CMA.xml',
 'CRS.xml',
 'CTY.xml',
 'EA7.xml',
```

```
 'ECV.xml',
 'EW1.xml',
 'EWW.xml',
 'F98.xml',
 'F9V.xml',
 'FC1.xml',
 'FEF.xml',
 'FPG.xml',
 'FSS.xml',
 'FT1.xml',
 'HRG.xml',
 'HWV.xml',
 'HXH.xml',
 'J18.xml',
 'J57.xml',
 'J7G.xml']
```

[8]:
```
dem_docs = [f for f in os.listdir(corpus_path+'/dem')]

# Printing the contents for testing purposes
dem_docs
```

[8]:
```
['KB5.xml',
 'KB7.xml',
 'KBC.xml',
 'KBD.xml',
 'KBH.xml',
 'KBJ.xml',
 'KBP.xml',
 'KBW.xml',
 'KCC.xml',
 'KCF.xml',
 'KCU.xml',
 'KCV.xml',
 'KD0.xml',
 'KD1.xml',
 'KD3.xml',
 'KD7.xml',
 'KD8.xml',
 'KDD.xml',
 'KDF.xml',
 'KDJ.xml',
 'KE2.xml',
 'KE4.xml',
 'KNR.xml',
 'KP2.xml',
 'KP5.xml',
```

```
 'KP7.xml',
 'KPU.xml',
 'KPX.xml',
 'KSN.xml',
 'KSW.xml']
```

[9]:
```
fic_docs = [f for f in os.listdir(corpus_path+'/fic')]

# Printing the contents for testing purposes
fic_docs
```

[9]:
```
['AB9.xml',
 'AC2.xml',
 'BMW.xml',
 'BPA.xml',
 'C8T.xml',
 'CB5.xml',
 'CCW.xml',
 'CDB.xml',
 'CFY.xml',
 'FAJ.xml',
 'FET.xml',
 'FPB.xml',
 'G01.xml',
 'G0L.xml',
 'G0S.xml',
 'G0Y.xml',
 'GUU.xml',
 'GVL.xml',
 'H85.xml',
 'H9C.xml',
 'H9D.xml',
 'HR9.xml',
 'J10.xml',
 'J54.xml',
 'K8V.xml']
```

1.3.2 b) Extracting corpus text

[10]:
```
def extract_s_tags_wtext(xml_file, cat):
    tree = ET.parse(xml_file)
    root = tree.getroot()

    # Find the <wtext> tag
    wtext_tag = root.find(".//wtext[@type='"+str(cat)+"']")
    if wtext_tag is not None:
        # Find all <s> tags within <wtext>
```

```
        s_tags = wtext_tag.findall('.//s')

        # Extract the <s> tags along with the actual tags
        result = []
        for s_tag in s_tags:
            s_with_text = f'<{s_tag.tag}> '  # Start with the opening <s> tag

            # Include direct text and tail of each child element within <s> tag
            for child in s_tag:
                if child.text:
                    s_with_text += child.text.lower()
                if child.tail:
                    s_with_text += child.tail.lower()

            s_with_text += f' </{s_tag.tag}>'  # Add the closing </s> tag
            result.append(s_with_text)
        return result
    else:
        return None
```

[11]:
```
def extract_s_tags_stext(xml_file, cat):
    tree = ET.parse(xml_file)
    root = tree.getroot()

    # Find the <wtext> tag
    wtext_tag = root.find(".//stext[@type='"+str(cat)+"']")
    if wtext_tag is not None:
        # Find all <s> tags within <wtext>
        s_tags = wtext_tag.findall('.//s')

        # Extract the <s> tags along with the actual tags
        result = []
        for s_tag in s_tags:
            s_with_text = f'<{s_tag.tag}> '  # Start with the opening <s> tag

            # Include direct text and tail of each child element within <s> tag
            for child in s_tag:
                if child.text:
                    s_with_text += child.text.lower()
                if child.tail:
                    s_with_text += child.tail.lower()

            s_with_text += f' </{s_tag.tag}>'  # Add the closing </s> tag
            result.append(s_with_text)
        return result
    else:
        return None
```

[12]:
```
# Function to remove punctuation marks from a sentence
def remove_punctuation(sen):
    result = re.sub(r'[^\w\s]', '', sen)
    return result
```

[13]:
```
# Iterate over the news files in the corpus directory
corpus_sen_news = []
news_start_time = time.time()
for news_doc in news_docs:
    file_path = corpus_path+'/news/'+str(news_doc)
    if (extract_s_tags_wtext(file_path, 'NEWS') is None):
        text = extract_s_tags_stext(file_path, 'NEWS')
    else:
        text = extract_s_tags_wtext(file_path, 'NEWS')
    corpus_sen_news.append(text)
    corpus_sen.append(text)

news_end_time = time.time()
news_time = news_end_time - news_start_time
corpus_sen_cat['news'] = corpus_sen_news.copy()
```

[14]:
```
# Iterate over the fiction files in the corpus directory
corpus_sen_fic = []
fic_start_time = time.time()
for fic_doc in fic_docs:
    file_path = corpus_path+'/fic/'+str(fic_doc)
    if (extract_s_tags_wtext(file_path, 'FICTION') is None):
        text = extract_s_tags_stext(file_path, 'FICTION')
    else:
        text = extract_s_tags_wtext(file_path, 'FICTION')
    corpus_sen_fic.append(text)
    corpus_sen.append(text)

fic_end_time = time.time()
fic_time = fic_end_time - fic_start_time
corpus_sen_cat['fic'] = corpus_sen_fic.copy()
```

[15]:
```
# Iterate over the dem files in the corpus directory
corpus_sen_dem = []
dem_start_time = time.time()
for dem_doc in dem_docs:
    file_path = corpus_path+'/dem/'+str(dem_doc)
    if (extract_s_tags_wtext(file_path, 'CONVRSN') is None):
        text = extract_s_tags_stext(file_path, 'CONVRSN')
    else:
        text = extract_s_tags_wtext(file_path, 'CONVRSN')
    corpus_sen_dem.append(text)
```

```
    corpus_sen.append(text)

dem_end_time = time.time()
dem_time = dem_end_time - dem_start_time
corpus_sen_cat['dem'] = corpus_sen_dem.copy()
```

[16]:
```
# Iterate over the aca files in the corpus directory
corpus_sen_aca = []
aca_start_time = time.time()
for aca_doc in aca_docs:
    file_path = corpus_path+'/aca/'+str(aca_doc)
    if (extract_s_tags_wtext(file_path, 'ACPROSE') is None):
        text = extract_s_tags_stext(file_path, 'ACPROSE')
    else:
        text = extract_s_tags_wtext(file_path, 'ACPROSE')

    if text is None:
        text = extract_s_tags_wtext(file_path, 'UNPUB')
    corpus_sen_aca.append(text)
    corpus_sen.append(text)

aca_end_time = time.time()
aca_time = aca_end_time - aca_start_time
corpus_sen_cat['aca'] = corpus_sen_aca.copy()
```

[17]:
```
# Checking total time it took to extract the corpus
total_extract_time = news_time + fic_time + dem_time + aca_time
print(f"Time taken to extract the corpus: {round(total_extract_time, 2)}␣
 ↪seconds")
```

```
Time taken to extract the corpus: 29.02 seconds
```

[18]:
```
# Looping through the corpus and remove any punctuation marks from the sentences
for i in range(len(corpus_sen)):
    for j, sentence in enumerate(corpus_sen[i]):
        text_words = sentence.split()
        text_words_string = ' '.join(text_words[1:-1])
        cleaned_text = "<s> " + remove_punctuation(text_words_string) + " </s>"
        corpus_sen[i][j] = cleaned_text
```

[19]:
```
#corpus_sen_news[10] (for testing purposes)
```

[20]:
```
#corpus_sen[169] (for testing purposes)
```

1.3.3 Splitting corpus into training set and test set

Here the corpus is split with a ratio of [4:1]. This means that 80% of the corpus is used as the training set, while the remaining 20% is used for testing.

[21]:
```
def save_corpus(corpus, filename):
    with open(filename, 'wb') as file:
        pickle.dump(corpus, file)
```

[22]:
```
def split_data(corpus):
    train_corpus, test_corpus = train_test_split(corpus, test_size=0.2,␣
 ↪random_state=42)

    return train_corpus, test_corpus
```

[23]:
```
train_corpus_sen, test_corpus_sen = split_data(corpus_sen)
```

[24]:
```
save_corpus(train_corpus_sen, train_corpus_sen_filename)
save_corpus(test_corpus_sen, test_corpus_sen_filename)
```

[25]:
```
os.listdir('models')
```

[25]:
```
['laplace',
 'test_corpus_sen.pkl',
 'train_corpus_sen.pkl',
 'unk',
 'unk_laplace',
 'vanilla']
```

[26]:
```
#train_corpus_sen[0:5][0:10] (for testing purposes)
```

[27]:
```
#test_corpus_sen[0:10][0:20] (for testing purposes)
```

1.3.4 Loading training and test corpus

[28]:
```
def load_corpus(filename):
    with open(filename, 'rb') as file:
        corpus_to_get = pickle.load(file)

    return corpus_to_get
```

[29]:
```
train_corpus_sen = load_corpus(train_corpus_sen_filename)
test_corpus_sen = load_corpus(test_corpus_sen_filename)
```

[30]:
```
# train_corpus_sen[0:5][0:10] (for testing purposes)
```

[31]:
```
# test_corpus_sen[0:10][0:20] (for testing purposes)
```

[32]:
```
len(train_corpus_sen)
```

[32]:
```
145
```

1.3.5 c) Building Frequency Counts

[33]:
```
# Creating constant values for specific n-grams
UNIGRAM_VAL = 1
BIGRAM_VAL = 2
TRIGRAM_VAL = 3
```

[34]:
```
def build_n_gram_freq_counts(corpus, n):
    count_dict = Counter()

    # Calculate the frequency of each n-gram in every document
    for text in corpus:
        for sen in text:
            words = sen.split()
            #print(words)
            if n == 1:
                count_dict.update(words)
            elif n > 1:
                for i in range(len(words) - n + 1):
                    ngram = ' '.join(words[i:i+n])
                    count_dict[ngram] += 1

    return count_dict
```

1.3.6 i) Unigram

[35]:
```
# Variable to represent start time
vanilla_uni_start_time = time.time()

# Creating an empty dictionary to store the word and its frequency count
train_vanilla_dict_sen = dict(build_n_gram_freq_counts(train_corpus_sen,␣
 ↪UNIGRAM_VAL))

# Variable to represent end time
vanilla_uni_end_time = time.time()

# Calculate actual time
vanilla_uni_time = vanilla_uni_end_time - vanilla_uni_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_vanilla_dict_sen.items())[0:100]))
```

```
{'<s>': 272094, 'struggling': 73, 'dentists': 17, 'pull': 163, 'more': 5769,
'teeth': 204, '</s>': 272094, 'by': 13562, 'david': 544, 'fletcher': 20,
'health': 660, 'services': 455, 'correspondent': 101, 'are': 14997, 'pulling':
107, 'out': 6091, 'patients': 628, 'unnecessarily': 6, 'as': 15481, 'they':
14490, 'struggle': 92, 'to': 74111, 'maintain': 91, 'living': 545, 'standards':
149, 'under': 1442, 'a': 70175, 'new': 2966, 'government': 1084, 'contract':
```

198, 'survey': 183, 'showed': 339, 'yesterday': 1253, 'two': 6509, 'three': 3423, 'said': 10248, 'had': 14930, 'extracted': 23, 'that': 35411, 'might': 2391, 'have': 17787, 'filled': 137, 'before': 2824, 'the': 169625, 'was': 28451, 'introduced': 198, 'than': 3166, 'half': 1416, 'their': 6601, 'pay': 675, 'last': 2991, 'year': 1910, 'not': 15055, 'increased': 306, 'or': 10358, 'even': 2194, 'fell': 378, 'of': 76795, '866': 3, 'in': 53568, 'dental': 13, 'journal': 29, 'probe': 39, 'says': 1296, 'when': 7291, 'were': 9753, 'asked': 1108, 'whether': 1002, 'referred': 135, 'for': 24428, 'extraction': 22, 'any': 3615, 'which': 9695, 'done': 1647, 'introduction': 149, '1990': 171, '61': 51, 'while': 1713, 'nearly': 501, '37': 47, 'mr': 1982, 'jeremy': 28, 'cowan': 4, 'editor': 65, 'contracts': 46, 'practical': 154, 'effect': 587, 'reduce': 188, 'preventive': 29, 'dentistry': 1, 'extractions': 1, 'conclusions': 68, 'challenged': 36, 'joe': 284, 'rich': 176, 'chairman': 309, 'british': 895, 'associations': 49, 'general': 895, 'committee': 297}

1.3.7 ii) Bigram

[36]:
```
# Variable to represent start time
vanilla_bi_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_vanilla_dict_bi_sen = dict(build_n_gram_freq_counts(train_corpus_sen,␣
 ↪BIGRAM_VAL))

# Variable to represent end time
vanilla_bi_end_time = time.time()

# Calculate actual time
vanilla_bi_time = vanilla_bi_end_time - vanilla_bi_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_vanilla_dict_bi_sen.items())[0:100]))
```

{'<s> struggling': 4, 'struggling dentists': 1, 'dentists pull': 1, 'pull more': 1, 'more teeth': 4, 'teeth </s>': 58, '<s> by': 1090, 'by david': 54, 'david fletcher': 1, 'fletcher health': 1, 'health services': 31, 'services correspondent': 12, 'correspondent </s>': 89, '<s> dentists': 2, 'dentists are': 1, 'are pulling': 3, 'pulling out': 10, 'out patients': 1, 'patients teeth': 1, 'teeth unnecessarily': 1, 'unnecessarily as': 1, 'as they': 399, 'they struggle': 1, 'struggle to': 12, 'to maintain': 54, 'maintain living': 1, 'living standards': 7, 'standards under': 1, 'under a': 97, 'a new': 695, 'new government': 6, 'government contract': 1, 'contract a': 3, 'a survey': 31, 'survey showed': 4, 'showed yesterday': 1, 'yesterday </s>': 342, '<s> two': 619, 'two three': 535, 'three dentists': 1, 'dentists said': 1, 'said they': 109, 'they had': 752, 'had extracted': 3, 'extracted teeth': 1, 'teeth that': 2, 'that they': 640, 'they might': 115, 'might have': 365, 'have filled': 1, 'filled before': 1, 'before the': 442, 'the contract': 21, 'contract was': 2, 'was introduced': 18, 'introduced </s>': 13, '<s> more': 201, 'more than': 602,

```
'than half': 37, 'half said': 2, 'said their': 8, 'their pay': 4, 'pay last': 1,
'last year': 377, 'year had': 2, 'had not': 318, 'not increased': 2, 'increased
or': 1, 'or even': 143, 'even fell': 1, 'fell </s>': 17, '<s> the': 16590, 'the
survey': 36, 'survey of': 24, 'of 866': 1, '866 dentists': 1, 'dentists in': 1,
'in the': 15042, 'the dental': 2, 'dental journal': 1, 'journal the': 1, 'the
probe': 1, 'probe says': 1, 'says when': 8, 'when dentists': 1, 'dentists were':
3, 'were asked': 17, 'asked whether': 8, 'whether they': 71, 'extracted or': 1,
'or referred': 2, 'referred for': 2, 'for extraction': 1, 'extraction any': 1,
'any teeth': 1, 'teeth which': 2, 'which they': 209, 'might not': 104, 'not
have': 254, 'have done': 202}
```

1.3.8 iii) Trigram

[37]:
```
# Variable to represent start time
vanilla_tri_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_vanilla_dict_tri_sen = dict(build_n_gram_freq_counts(train_corpus_sen,␣
 ↪TRIGRAM_VAL))

# Variable to represent end time
vanilla_tri_end_time = time.time()

# Calculate actual time
vanilla_tri_time = vanilla_tri_end_time - vanilla_tri_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_vanilla_dict_tri_sen.items())[0:100]))
```

```
{'<s> struggling dentists': 1, 'struggling dentists pull': 1, 'dentists pull
more': 1, 'pull more teeth': 1, 'more teeth </s>': 1, '<s> by david': 42, 'by
david fletcher': 1, 'david fletcher health': 1, 'fletcher health services': 1,
'health services correspondent': 12, 'services correspondent </s>': 12, '<s>
dentists are': 1, 'dentists are pulling': 1, 'are pulling out': 3, 'pulling out
patients': 1, 'out patients teeth': 1, 'patients teeth unnecessarily': 1, 'teeth
unnecessarily as': 1, 'unnecessarily as they': 1, 'as they struggle': 1, 'they
struggle to': 1, 'struggle to maintain': 1, 'to maintain living': 1, 'maintain
living standards': 1, 'living standards under': 1, 'standards under a': 1,
'under a new': 5, 'a new government': 3, 'new government contract': 1,
'government contract a': 1, 'contract a survey': 1, 'a survey showed': 1,
'survey showed yesterday': 1, 'showed yesterday </s>': 1, '<s> two three': 27,
'two three dentists': 1, 'three dentists said': 1, 'dentists said they': 1,
'said they had': 9, 'they had extracted': 2, 'had extracted teeth': 1,
'extracted teeth that': 1, 'teeth that they': 1, 'that they might': 12, 'they
might have': 21, 'might have filled': 1, 'have filled before': 1, 'filled before
the': 1, 'before the contract': 1, 'the contract was': 1, 'contract was
introduced': 1, 'was introduced </s>': 1, '<s> more than': 17, 'more than half':
19, 'than half said': 1, 'half said their': 1, 'said their pay': 1, 'their pay
```

```
last': 1, 'pay last year': 1, 'last year had': 2, 'year had not': 1, 'had not
increased': 1, 'not increased or': 1, 'increased or even': 1, 'or even fell': 1,
'even fell </s>': 1, '<s> the survey': 14, 'the survey of': 3, 'survey of 866':
1, 'of 866 dentists': 1, '866 dentists in': 1, 'dentists in the': 1, 'in the
dental': 1, 'the dental journal': 1, 'dental journal the': 1, 'journal the
probe': 1, 'the probe says': 1, 'probe says when': 1, 'says when dentists': 1,
'when dentists were': 1, 'dentists were asked': 1, 'were asked whether': 1,
'asked whether they': 1, 'whether they had': 6, 'had extracted or': 1,
'extracted or referred': 1, 'or referred for': 1, 'referred for extraction': 1,
'for extraction any': 1, 'extraction any teeth': 1, 'any teeth which': 1, 'teeth
which they': 1, 'which they might': 5, 'they might not': 8, 'might not have':
18, 'not have done': 7, 'have done before': 1, 'done before the': 2, 'before the
introduction': 3, 'the introduction of': 60}
```

[38]:
```
# Checking total time it took to build the frequency counts for the vanilla␣
 ↪models
vanilla_total_time = vanilla_uni_time + vanilla_bi_time + vanilla_tri_time
print(f"Time taken to build frequency counts for vanilla models:␣
 ↪{round(vanilla_total_time, 2)} seconds")
```

```
Time taken to build frequency counts for vanilla models: 6.75 seconds
```

1.4 Part 2

1.4.1 a) Applying Laplace Smoothing to all n-gram models

[39]:
```
def apply_laplace(org_dict):
    # Creating an empty lists of dictionaries to calculate TF (term frequency)
    laplace_dict = org_dict.copy()

    # Calculate the frequency of each word in every document
    for word, count in laplace_dict.items():
        new_count = count+1
        laplace_dict[word] = new_count
        #laplace_dict.update({word: new_count})


    return laplace_dict
```

1.4.2 i) Unigram

[40]:
```
# Variable to represent start time
laplace_uni_start_time = time.time()

# Creating an empty dictionary to store the word and its frequency count
train_laplace_dict_sen = apply_laplace(train_vanilla_dict_sen)

# Variable to represent end time
```

```
laplace_uni_end_time = time.time()

# Calculate actual time
laplace_uni_time = laplace_uni_end_time - laplace_uni_start_time

# Printing the contents of the dictionary (for testing purposes)
print(print(dict(list(train_laplace_dict_sen.items())[0:100])))
```

```
{'<s>': 272095, 'struggling': 74, 'dentists': 18, 'pull': 164, 'more': 5770,
'teeth': 205, '</s>': 272095, 'by': 13563, 'david': 545, 'fletcher': 21,
'health': 661, 'services': 456, 'correspondent': 102, 'are': 14998, 'pulling':
108, 'out': 6092, 'patients': 629, 'unnecessarily': 7, 'as': 15482, 'they':
14491, 'struggle': 93, 'to': 74112, 'maintain': 92, 'living': 546, 'standards':
150, 'under': 1443, 'a': 70176, 'new': 2967, 'government': 1085, 'contract':
199, 'survey': 184, 'showed': 340, 'yesterday': 1254, 'two': 6510, 'three':
3424, 'said': 10249, 'had': 14931, 'extracted': 24, 'that': 35412, 'might':
2392, 'have': 17788, 'filled': 138, 'before': 2825, 'the': 169626, 'was': 28452,
'introduced': 199, 'than': 3167, 'half': 1417, 'their': 6602, 'pay': 676,
'last': 2992, 'year': 1911, 'not': 15056, 'increased': 307, 'or': 10359, 'even':
2195, 'fell': 379, 'of': 76796, '866': 4, 'in': 53569, 'dental': 14, 'journal':
30, 'probe': 40, 'says': 1297, 'when': 7292, 'were': 9754, 'asked': 1109,
'whether': 1003, 'referred': 136, 'for': 24429, 'extraction': 23, 'any': 3616,
'which': 9696, 'done': 1648, 'introduction': 150, '1990': 172, '61': 52,
'while': 1714, 'nearly': 502, '37': 48, 'mr': 1983, 'jeremy': 29, 'cowan': 5,
'editor': 66, 'contracts': 47, 'practical': 155, 'effect': 588, 'reduce': 189,
'preventive': 30, 'dentistry': 2, 'extractions': 2, 'conclusions': 69,
'challenged': 37, 'joe': 285, 'rich': 177, 'chairman': 310, 'british': 896,
'associations': 50, 'general': 896, 'committee': 298}
None
```

1.4.3 ii) Bigram

[41]:
```
# Variable to represent start time
laplace_bi_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_laplace_dict_bi_sen = apply_laplace(train_vanilla_dict_bi_sen)

# Variable to represent end time
laplace_bi_end_time = time.time()

# Calculate actual time
laplace_bi_time = laplace_bi_end_time - laplace_bi_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_laplace_dict_bi_sen.items())[0:100]))
```

```
{'<s> struggling': 5, 'struggling dentists': 2, 'dentists pull': 2, 'pull more':
```

2, 'more teeth': 5, 'teeth </s>': 59, '<s> by': 1091, 'by david': 55, 'david fletcher': 2, 'fletcher health': 2, 'health services': 32, 'services correspondent': 13, 'correspondent </s>': 90, '<s> dentists': 3, 'dentists are': 2, 'are pulling': 4, 'pulling out': 11, 'out patients': 2, 'patients teeth': 2, 'teeth unnecessarily': 2, 'unnecessarily as': 2, 'as they': 400, 'they struggle': 2, 'struggle to': 13, 'to maintain': 55, 'maintain living': 2, 'living standards': 8, 'standards under': 2, 'under a': 98, 'a new': 696, 'new government': 7, 'government contract': 2, 'contract a': 4, 'a survey': 32, 'survey showed': 5, 'showed yesterday': 2, 'yesterday </s>': 343, '<s> two': 620, 'two three': 536, 'three dentists': 2, 'dentists said': 2, 'said they': 110, 'they had': 753, 'had extracted': 4, 'extracted teeth': 2, 'teeth that': 3, 'that they': 641, 'they might': 116, 'might have': 366, 'have filled': 2, 'filled before': 2, 'before the': 443, 'the contract': 22, 'contract was': 3, 'was introduced': 19, 'introduced </s>': 14, '<s> more': 202, 'more than': 603, 'than half': 38, 'half said': 3, 'said their': 9, 'their pay': 5, 'pay last': 2, 'last year': 378, 'year had': 3, 'had not': 319, 'not increased': 3, 'increased or': 2, 'or even': 144, 'even fell': 2, 'fell </s>': 18, '<s> the': 16591, 'the survey': 37, 'survey of': 25, 'of 866': 2, '866 dentists': 2, 'dentists in': 2, 'in the': 15043, 'the dental': 3, 'dental journal': 2, 'journal the': 2, 'the probe': 2, 'probe says': 2, 'says when': 9, 'when dentists': 2, 'dentists were': 4, 'were asked': 18, 'asked whether': 9, 'whether they': 72, 'extracted or': 2, 'or referred': 3, 'referred for': 3, 'for extraction': 2, 'extraction any': 2, 'any teeth': 2, 'teeth which': 3, 'which they': 210, 'might not': 105, 'not have': 255, 'have done': 203}

1.4.4 iii) Trigram

[42]:
```
# Variable to represent start time
laplace_tri_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_laplace_dict_tri_sen = apply_laplace(train_vanilla_dict_tri_sen)

# Variable to represent end time
laplace_tri_end_time = time.time()

# Calculate actual time
laplace_tri_time = laplace_tri_end_time - laplace_tri_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_laplace_dict_tri_sen.items())[0:100]))
```

{'<s> struggling dentists': 2, 'struggling dentists pull': 2, 'dentists pull more': 2, 'pull more teeth': 2, 'more teeth </s>': 2, '<s> by david': 43, 'by david fletcher': 2, 'david fletcher health': 2, 'fletcher health services': 2, 'health services correspondent': 13, 'services correspondent </s>': 13, '<s> dentists are': 2, 'dentists are pulling': 2, 'are pulling out': 4, 'pulling out patients': 2, 'out patients teeth': 2, 'patients teeth unnecessarily': 2, 'teeth

```
unnecessarily as': 2, 'unnecessarily as they': 2, 'as they struggle': 2, 'they
struggle to': 2, 'struggle to maintain': 2, 'to maintain living': 2, 'maintain
living standards': 2, 'living standards under': 2, 'standards under a': 2,
'under a new': 6, 'a new government': 4, 'new government contract': 2,
'government contract a': 2, 'contract a survey': 2, 'a survey showed': 2,
'survey showed yesterday': 2, 'showed yesterday </s>': 2, '<s> two three': 28,
'two three dentists': 2, 'three dentists said': 2, 'dentists said they': 2,
'said they had': 10, 'they had extracted': 3, 'had extracted teeth': 2,
'extracted teeth that': 2, 'teeth that they': 2, 'that they might': 13, 'they
might have': 22, 'might have filled': 2, 'have filled before': 2, 'filled before
the': 2, 'before the contract': 2, 'the contract was': 2, 'contract was
introduced': 2, 'was introduced </s>': 2, '<s> more than': 18, 'more than half':
20, 'than half said': 2, 'half said their': 2, 'said their pay': 2, 'their pay
last': 2, 'pay last year': 2, 'last year had': 3, 'year had not': 2, 'had not
increased': 2, 'not increased or': 2, 'increased or even': 2, 'or even fell': 2,
'even fell </s>': 2, '<s> the survey': 15, 'the survey of': 4, 'survey of 866':
2, 'of 866 dentists': 2, '866 dentists in': 2, 'dentists in the': 2, 'in the
dental': 2, 'the dental journal': 2, 'dental journal the': 2, 'journal the
probe': 2, 'the probe says': 2, 'probe says when': 2, 'says when dentists': 2,
'when dentists were': 2, 'dentists were asked': 2, 'were asked whether': 2,
'asked whether they': 2, 'whether they had': 7, 'had extracted or': 2,
'extracted or referred': 2, 'or referred for': 2, 'referred for extraction': 2,
'for extraction any': 2, 'extraction any teeth': 2, 'any teeth which': 2, 'teeth
which they': 2, 'which they might': 6, 'they might not': 9, 'might not have':
19, 'not have done': 8, 'have done before': 2, 'done before the': 3, 'before the
introduction': 4, 'the introduction of': 61}
```

[43]:
```
# Checking total time it took to build the frequency counts for the laplace␣
↪models
laplace_total_time = laplace_uni_time + laplace_bi_time + laplace_tri_time
print(f"Time taken to build frequency counts for laplace models:␣
↪{round(laplace_total_time, 2)} seconds")
```

```
Time taken to build frequency counts for laplace models: 0.77 seconds
```

1.4.5 bI) Applying < UNK > tags to any word counts less than or equal to 2

[44]:
```
def apply_unk(org_dict):
    unk_dict = org_dict.copy()
    unk_words = []
    unk_count = 0

    for word, count in unk_dict.items():
        if count <= 2:
            unk_count += count
            unk_words.append(word)

    for x in unk_words:
```

```
        unk_dict.pop(x)

    unk_dict["UNK"] = unk_count

    return unk_dict
```

1.4.6 i) Unigram

[45]:
```
# Variable to represent start time
unk_uni_start_time = time.time()

# Creating an empty dictionary to store the word and its frequency count
train_unk_dict_sen = apply_unk(train_vanilla_dict_sen)

# Variable to represent end time
unk_uni_end_time = time.time()

# Calculate actual time
unk_uni_time = unk_uni_end_time - unk_uni_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_dict_sen.items())[0:100]))
for key, val in train_unk_dict_sen.items():
    if key == 'UNK':
        print({key: val})
```

```
{'<s>': 272094, 'struggling': 73, 'dentists': 17, 'pull': 163, 'more': 5769,
'teeth': 204, '</s>': 272094, 'by': 13562, 'david': 544, 'fletcher': 20,
'health': 660, 'services': 455, 'correspondent': 101, 'are': 14997, 'pulling':
107, 'out': 6091, 'patients': 628, 'unnecessarily': 6, 'as': 15481, 'they':
14490, 'struggle': 92, 'to': 74111, 'maintain': 91, 'living': 545, 'standards':
149, 'under': 1442, 'a': 70175, 'new': 2966, 'government': 1084, 'contract':
198, 'survey': 183, 'showed': 339, 'yesterday': 1253, 'two': 6509, 'three':
3423, 'said': 10248, 'had': 14930, 'extracted': 23, 'that': 35411, 'might':
2391, 'have': 17787, 'filled': 137, 'before': 2824, 'the': 169625, 'was': 28451,
'introduced': 198, 'than': 3166, 'half': 1416, 'their': 6601, 'pay': 675,
'last': 2991, 'year': 1910, 'not': 15055, 'increased': 306, 'or': 10358, 'even':
2194, 'fell': 378, 'of': 76795, '866': 3, 'in': 53568, 'dental': 13, 'journal':
29, 'probe': 39, 'says': 1296, 'when': 7291, 'were': 9753, 'asked': 1108,
'whether': 1002, 'referred': 135, 'for': 24428, 'extraction': 22, 'any': 3615,
'which': 9695, 'done': 1647, 'introduction': 149, '1990': 171, '61': 51,
'while': 1713, 'nearly': 501, '37': 47, 'mr': 1982, 'jeremy': 28, 'cowan': 4,
'editor': 65, 'contracts': 46, 'practical': 154, 'effect': 587, 'reduce': 188,
'preventive': 29, 'conclusions': 68, 'challenged': 36, 'joe': 284, 'rich': 176,
'chairman': 309, 'british': 895, 'associations': 49, 'general': 895,
'committee': 297, 'who': 5682, 'it': 39565}
{'UNK': 53754}
```

1.4.7 ii) Bigram

[46]:
```
# Variable to represent start time
unk_bi_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_unk_dict_bi_sen = apply_unk(train_vanilla_dict_bi_sen)

# Variable to represent end time
unk_bi_end_time = time.time()

# Calculate actual time
unk_bi_time = unk_bi_end_time - unk_bi_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_dict_bi_sen.items())[0:100]))
for key, val in train_unk_dict_bi_sen.items():
    if key == 'UNK':
        print({key: val})
```

```
{'<s> struggling': 4, 'more teeth': 4, 'teeth </s>': 58, '<s> by': 1090, 'by
david': 54, 'health services': 31, 'services correspondent': 12, 'correspondent
</s>': 89, 'are pulling': 3, 'pulling out': 10, 'as they': 399, 'struggle to':
12, 'to maintain': 54, 'living standards': 7, 'under a': 97, 'a new': 695, 'new
government': 6, 'contract a': 3, 'a survey': 31, 'survey showed': 4, 'yesterday
</s>': 342, '<s> two': 619, 'two three': 535, 'said they': 109, 'they had': 752,
'had extracted': 3, 'that they': 640, 'they might': 115, 'might have': 365,
'before the': 442, 'the contract': 21, 'was introduced': 18, 'introduced </s>':
13, '<s> more': 201, 'more than': 602, 'than half': 37, 'said their': 8, 'their
pay': 4, 'last year': 377, 'had not': 318, 'or even': 143, 'fell </s>': 17, '<s>
the': 16590, 'the survey': 36, 'survey of': 24, 'in the': 15042, 'says when': 8,
'dentists were': 3, 'were asked': 17, 'asked whether': 8, 'whether they': 71,
'which they': 209, 'might not': 104, 'not have': 254, 'have done': 202, 'done
before': 9, 'the introduction': 69, 'introduction of': 70, 'of the': 17410, 'the
1990': 9, 'not </s>': 724, '<s> mr': 717, 'said the': 565, 'the contracts': 3,
'practical effect': 3, 'effect was': 12, 'was to': 363, 'to reduce': 100, 'the
conclusions': 6, 'were challenged': 3, 'challenged by': 4, 'by mr': 47, 'mr
joe': 4, 'chairman of': 95, 'the british': 271, 'services committee': 8, 'who
said': 57, 'said it': 287, 'it was': 4810, 'was probably': 56, 'true that': 46,
'that more': 26, 'teeth were': 7, 'were being': 55, 'were seeing': 5, 'more
patients': 5, 'patients </s>': 51, 'had no': 257, 'no financial': 3, 'incentive
to': 5, 'to extract': 17, 'he said': 1793, 'said </s>': 1522, 'fee was': 5, 'but
the': 1172, 'the filling': 4, 'could be': 830, 'be almost': 10, 'almost
anything': 9, 'anything the': 13}
{'UNK': 895347}
```

1.4.8 iii) Trigram

[47]:
```
# Variable to represent start time
unk_tri_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_unk_dict_tri_sen = apply_unk(train_vanilla_dict_tri_sen)

# Variable to represent end time
unk_tri_end_time = time.time()

# Calculate actual time
unk_tri_time = unk_tri_end_time - unk_tri_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_dict_tri_sen.items())[0:100]))
for key, val in train_unk_dict_tri_sen.items():
    if key == 'UNK':
        print({key: val})
```

```
{'<s> by david': 42, 'health services correspondent': 12, 'services
correspondent </s>': 12, 'are pulling out': 3, 'under a new': 5, 'a new
government': 3, '<s> two three': 27, 'said they had': 9, 'that they might': 12,
'they might have': 21, '<s> more than': 17, 'more than half': 19, '<s> the
survey': 14, 'the survey of': 3, 'whether they had': 6, 'which they might': 5,
'they might not': 8, 'might not have': 18, 'not have done': 7, 'before the
introduction': 3, 'the introduction of': 60, 'introduction of the': 18, 'they
had not': 14, 'had not </s>': 5, 'effect was to': 3, 'chairman of the': 57, 'of
the british': 63, 'who said it': 6, 'said it was': 105, 'it was probably': 12,
'more patients </s>': 3, 'he said </s>': 469, 'was being made': 4, 'by the
government': 23, 'the government to': 48, 'to implement the': 6, 'and he had':
63, 'to arrange a': 6, 'a meeting </s>': 7, 'to appeal </s>': 6, 'to appeal
against': 3, 'appeal against his': 4, 'was found guilty': 4, 'by the general':
6, 'the general medical': 7, 'general medical council': 6, 'heard that he': 3,
'he failed to': 5, 'to detect a': 4, '<s> the nhs': 4, 'the nhs is': 4, 'is not
for': 7, 'not for sale': 5, 'in her first': 3, 'the cabinet </s>': 5, '<s> by
george': 7, '<s> the new': 70, 'mrs virginia bottomley': 3, 'to take the': 105,
'the national health': 14, 'national health service': 15, 'health service </s>':
6, '<s> but her': 13, 'to provide a': 45, 'a period of': 43, 'employed by the':
4, 'by the nhs': 3, 'coupled with a': 5, 'with a fierce': 3, 'to maintain the':
12, 'the overwhelming majority': 6, 'overwhelming majority of': 8, 'to turn
the': 10, 'the general election': 21, 'a mandate to': 3, 'to reverse the': 6,
'<s> mrs bottomley': 6, 'the opportunity to': 43, 'and to give': 8, 'the
confidence to': 10, 'to make them': 17, '<s> in her': 33, 'she said the': 16,
'the election result': 5, 'commitment to the': 18, 'to the nhs': 3, 'of health
care': 11, '<s> it is': 1425, 'it is not': 302, 'for sale </s>': 27, '<s> it
will': 135, 'it will continue': 5, 'will continue to': 31, 'continue to be': 16,
'to be true': 8, 'be true to': 3, 'available to all': 3, 'at the point': 17,
```

```
'the point of': 77, 'she said </s>': 228}
{'UNK': 2102272}
```

[48]:
```
# Checking total time it took to build the frequency counts for the UNK models
unk_total_time = unk_uni_time + unk_bi_time + unk_tri_time
print(f"Time taken to build frequency counts for UNK models:␣
 ↪{round(unk_total_time, 2)} seconds")
```

```
Time taken to build frequency counts for UNK models: 0.86 seconds
```

1.4.9 bII) Applying Laplace smoothing to UNK dictionaries

1.4.10 i) Unigram

[49]:
```
# Variable to represent start time
unk_laplace_uni_start_time = time.time()

# Creating an empty dictionary to store the word and its frequency count
train_unk_laplace_dict_sen = apply_laplace(train_unk_dict_sen)

# Variable to represent end time
unk_laplace_uni_end_time = time.time()

# Calculate actual time
unk_laplace_uni_time = unk_laplace_uni_end_time - unk_laplace_uni_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_laplace_dict_sen.items())[0:100]))
for key, val in train_unk_laplace_dict_sen.items():
    if key == 'UNK':
        print({key: val})
```

```
{'<s>': 272095, 'struggling': 74, 'dentists': 18, 'pull': 164, 'more': 5770,
'teeth': 205, '</s>': 272095, 'by': 13563, 'david': 545, 'fletcher': 21,
'health': 661, 'services': 456, 'correspondent': 102, 'are': 14998, 'pulling':
108, 'out': 6092, 'patients': 629, 'unnecessarily': 7, 'as': 15482, 'they':
14491, 'struggle': 93, 'to': 74112, 'maintain': 92, 'living': 546, 'standards':
150, 'under': 1443, 'a': 70176, 'new': 2967, 'government': 1085, 'contract':
199, 'survey': 184, 'showed': 340, 'yesterday': 1254, 'two': 6510, 'three':
3424, 'said': 10249, 'had': 14931, 'extracted': 24, 'that': 35412, 'might':
2392, 'have': 17788, 'filled': 138, 'before': 2825, 'the': 169626, 'was': 28452,
'introduced': 199, 'than': 3167, 'half': 1417, 'their': 6602, 'pay': 676,
'last': 2992, 'year': 1911, 'not': 15056, 'increased': 307, 'or': 10359, 'even':
2195, 'fell': 379, 'of': 76796, '866': 4, 'in': 53569, 'dental': 14, 'journal':
30, 'probe': 40, 'says': 1297, 'when': 7292, 'were': 9754, 'asked': 1109,
'whether': 1003, 'referred': 136, 'for': 24429, 'extraction': 23, 'any': 3616,
'which': 9696, 'done': 1648, 'introduction': 150, '1990': 172, '61': 52,
'while': 1714, 'nearly': 502, '37': 48, 'mr': 1983, 'jeremy': 29, 'cowan': 5,
'editor': 66, 'contracts': 47, 'practical': 155, 'effect': 588, 'reduce': 189,
```

```
'preventive': 30, 'conclusions': 69, 'challenged': 37, 'joe': 285, 'rich': 177,
'chairman': 310, 'british': 896, 'associations': 50, 'general': 896,
'committee': 298, 'who': 5683, 'it': 39566}
{'UNK': 53755}
```

1.4.11 ii) Bigram

[50]:
```
# Variable to represent start time
unk_laplace_bi_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_unk_laplace_dict_bi_sen = apply_laplace(train_unk_dict_bi_sen)

# Variable to represent end time
unk_laplace_bi_end_time = time.time()

# Calculate actual time
unk_laplace_bi_time = unk_laplace_bi_end_time - unk_laplace_bi_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_laplace_dict_bi_sen.items())[0:100]))
for key, val in train_unk_laplace_dict_bi_sen.items():
    if key == 'UNK':
        print({key: val})
```

```
{'<s> struggling': 5, 'more teeth': 5, 'teeth </s>': 59, '<s> by': 1091, 'by
david': 55, 'health services': 32, 'services correspondent': 13, 'correspondent
</s>': 90, 'are pulling': 4, 'pulling out': 11, 'as they': 400, 'struggle to':
13, 'to maintain': 55, 'living standards': 8, 'under a': 98, 'a new': 696, 'new
government': 7, 'contract a': 4, 'a survey': 32, 'survey showed': 5, 'yesterday
</s>': 343, '<s> two': 620, 'two three': 536, 'said they': 110, 'they had': 753,
'had extracted': 4, 'that they': 641, 'they might': 116, 'might have': 366,
'before the': 443, 'the contract': 22, 'was introduced': 19, 'introduced </s>':
14, '<s> more': 202, 'more than': 603, 'than half': 38, 'said their': 9, 'their
pay': 5, 'last year': 378, 'had not': 319, 'or even': 144, 'fell </s>': 18, '<s>
the': 16591, 'the survey': 37, 'survey of': 25, 'in the': 15043, 'says when': 9,
'dentists were': 4, 'were asked': 18, 'asked whether': 9, 'whether they': 72,
'which they': 210, 'might not': 105, 'not have': 255, 'have done': 203, 'done
before': 10, 'the introduction': 70, 'introduction of': 71, 'of the': 17411,
'the 1990': 10, 'not </s>': 725, '<s> mr': 718, 'said the': 566, 'the
contracts': 4, 'practical effect': 4, 'effect was': 13, 'was to': 364, 'to
reduce': 101, 'the conclusions': 7, 'were challenged': 4, 'challenged by': 5,
'by mr': 48, 'mr joe': 5, 'chairman of': 96, 'the british': 272, 'services
committee': 9, 'who said': 58, 'said it': 288, 'it was': 4811, 'was probably':
57, 'true that': 47, 'that more': 27, 'teeth were': 8, 'were being': 56, 'were
seeing': 6, 'more patients': 6, 'patients </s>': 52, 'had no': 258, 'no
financial': 4, 'incentive to': 6, 'to extract': 18, 'he said': 1794, 'said
</s>': 1523, 'fee was': 6, 'but the': 1173, 'the filling': 5, 'could be': 831,
```

'be almost': 11, 'almost anything': 10, 'anything the': 14}
{'UNK': 895348}

1.4.12 iii) Trigram

[51]:
```
# Variable to represent start time
unk_laplace_tri_start_time = time.time()

# Creating an empty dictionary to store the phrase and its frequency count
train_unk_laplace_dict_tri_sen = apply_laplace(train_unk_dict_tri_sen)

# Variable to represent end time
unk_laplace_tri_end_time = time.time()

# Calculate actual time
unk_laplace_tri_time = unk_laplace_tri_end_time - unk_laplace_tri_start_time

# Printing the contents of the dictionary (for testing purposes)
print(dict(list(train_unk_laplace_dict_tri_sen.items())[0:100]))
for key, val in train_unk_laplace_dict_tri_sen.items():
    if key == 'UNK':
        print({key: val})
```

{'<s> by david': 43, 'health services correspondent': 13, 'services correspondent </s>': 13, 'are pulling out': 4, 'under a new': 6, 'a new government': 4, '<s> two three': 28, 'said they had': 10, 'that they might': 13, 'they might have': 22, '<s> more than': 18, 'more than half': 20, '<s> the survey': 15, 'the survey of': 4, 'whether they had': 7, 'which they might': 6, 'they might not': 9, 'might not have': 19, 'not have done': 8, 'before the introduction': 4, 'the introduction of': 61, 'introduction of the': 19, 'they had not': 15, 'had not </s>': 6, 'effect was to': 4, 'chairman of the': 58, 'of the british': 64, 'who said it': 7, 'said it was': 106, 'it was probably': 13, 'more patients </s>': 4, 'he said </s>': 470, 'was being made': 5, 'by the government': 24, 'the government to': 49, 'to implement the': 7, 'and he had': 64, 'to arrange a': 7, 'a meeting </s>': 8, 'to appeal </s>': 7, 'to appeal against': 4, 'appeal against his': 5, 'was found guilty': 5, 'by the general': 7, 'the general medical': 8, 'general medical council': 7, 'heard that he': 4, 'he failed to': 6, 'to detect a': 5, '<s> the nhs': 5, 'the nhs is': 5, 'is not for': 8, 'not for sale': 6, 'in her first': 4, 'the cabinet </s>': 6, '<s> by george': 8, '<s> the new': 71, 'mrs virginia bottomley': 4, 'to take the': 106, 'the national health': 15, 'national health service': 16, 'health service </s>': 7, '<s> but her': 14, 'to provide a': 46, 'a period of': 44, 'employed by the': 5, 'by the nhs': 4, 'coupled with a': 6, 'with a fierce': 4, 'to maintain the': 13, 'the overwhelming majority': 7, 'overwhelming majority of': 9, 'to turn the': 11, 'the general election': 22, 'a mandate to': 4, 'to reverse the': 7, '<s> mrs bottomley': 7, 'the opportunity to': 44, 'and to give': 9, 'the confidence to': 11, 'to make them': 18, '<s> in her': 34, 'she said the': 17, 'the election result': 6, 'commitment to the': 19, 'to the nhs': 4, 'of health

```
care': 12, '<s> it is': 1426, 'it is not': 303, 'for sale </s>': 28, '<s> it
will': 136, 'it will continue': 6, 'will continue to': 32, 'continue to be': 17,
'to be true': 9, 'be true to': 4, 'available to all': 4, 'at the point': 18,
'the point of': 78, 'she said </s>': 229}
{'UNK': 2102273}
```

[52]:
```
# Checking total time it took to build the frequency counts for the UNK+laplace␣
↪models
unk_laplace_total_time = unk_laplace_uni_time + unk_laplace_bi_time +␣
↪unk_laplace_tri_time
print(f"Time taken to build frequency counts for UNK+laplace models:␣
↪{round(unk_laplace_total_time, 2)} seconds")
```

```
Time taken to build frequency counts for UNK+laplace models: 0.15 seconds
```

[53]:
```
# Checking total time it took to build the frequency counts for all models
models_total_time = vanilla_total_time + laplace_total_time + unk_total_time +␣
↪unk_laplace_total_time
print(f"Time taken to build frequency counts for all models:␣
↪{round(models_total_time, 2)} seconds")
```

```
Time taken to build frequency counts for all models: 8.54 seconds
```

[54]:
```
process = psutil.Process(os.getpid())
print(f"Memory usage after building language models: {process.memory_info().rss␣
↪/ (1024 * 1024)} MB")
```

```
Memory usage after building language models: 752.0703125 MB
```

1.4.13 Saving generated frequency counts

[55]:
```
def save_model(model, filename):
    with open(filename, 'wb') as file:
        pickle.dump(model, file)
```

[56]:
```
save_model(train_vanilla_dict_sen, vanilla_dict_filename)
save_model(train_vanilla_dict_bi_sen, vanilla_dict_bi_filename)
save_model(train_vanilla_dict_tri_sen, vanilla_dict_tri_filename)
save_model(train_laplace_dict_sen, laplace_dict_filename)
save_model(train_laplace_dict_bi_sen, laplace_dict_bi_filename)
save_model(train_laplace_dict_tri_sen, laplace_dict_tri_filename)
save_model(train_unk_dict_sen, unk_dict_filename)
save_model(train_unk_dict_bi_sen, unk_dict_bi_filename)
save_model(train_unk_dict_tri_sen, unk_dict_tri_filename)
save_model(train_unk_laplace_dict_sen, unk_laplace_dict_filename)
save_model(train_unk_laplace_dict_bi_sen, unk_laplace_dict_bi_filename)
save_model(train_unk_laplace_dict_tri_sen, unk_laplace_dict_tri_filename)
```

[57]:
```
os.listdir('models/vanilla')
```

[57]:
```
['vanilla_bi_model.pkl', 'vanilla_tri_model.pkl', 'vanilla_uni_model.pkl']
```

1.4.14 Loading training and test corpus

[58]:
```
def load_model(filename):
    with open(filename, 'rb') as file:
        model_to_get = pickle.load(file)

    return model_to_get
```

[59]:
```
train_vanilla_dict_sen = load_model(vanilla_dict_filename)
train_vanilla_dict_bi_sen = load_model(vanilla_dict_bi_filename)
train_vanilla_dict_tri_sen = load_model(vanilla_dict_tri_filename)
train_laplace_dict_sen = load_model(laplace_dict_filename)
train_laplace_dict_bi_sen = load_model(laplace_dict_bi_filename)
train_laplace_dict_tri_sen = load_model(laplace_dict_tri_filename)
train_unk_dict_sen = load_model(unk_dict_filename)
train_unk_dict_bi_sen = load_model(unk_dict_bi_filename)
train_unk_dict_tri_sen = load_model(unk_dict_tri_filename)
train_unk_laplace_dict_sen = load_model(unk_laplace_dict_filename)
train_unk_laplace_dict_bi_sen = load_model(unk_laplace_dict_bi_filename)
train_unk_laplace_dict_tri_sen = load_model(unk_laplace_dict_tri_filename)
```

[60]:
```
#dict(list(train_vanilla_dict_sen.items())[0:150]) (for testing purposes)
```

[61]:
```
#dict(list(train_laplace_dict_bi_sen.items())[0:150]) (for testing purposes)
```

[62]:
```
# train_unk_dict_tri_sen (for testing purposes)
```

1.4.15 c) Removing punctuation from sentences

[63]:
```
# Creating an empty list to store the sentences of the train corpus for easy␣
 ↪access
train_extracted_sentences = []
```

[64]:
```
# Looping through the corpus and moving the sentences to the new list, while␣
 ↪removing any punctuation marks that may appear
for i in range(len(train_corpus_sen)):
    for j in range(len(train_corpus_sen[i])):
        text_words = train_corpus_sen[i][j].split()
        text_words_string = ' '.join(text_words[1:-1])
        cleaned_text = "<s> " + remove_punctuation(text_words_string) + " </s>"
        train_extracted_sentences.append(cleaned_text)
```

[65]:
```
# Printing a subset for testing purposes
train_extracted_sentences[0:25]
```

[65]: ['<s> struggling dentists pull more teeth </s>',
 '<s> by david fletcher health services correspondent </s>',
 '<s> dentists are pulling out patients teeth unnecessarily as they struggle to maintain living standards under a new government contract a survey showed yesterday </s>',
 '<s> two three dentists said they had extracted teeth that they might have filled before the contract was introduced </s>',
 '<s> more than half said their pay last year had not increased or even fell </s>',
 '<s> the survey of 866 dentists in the dental journal the probe says when dentists were asked whether they had extracted or referred for extraction any teeth which they might not have done before the introduction of the 1990 contract 61 said they had while nearly 37 said they had not </s>',
 '<s> mr jeremy cowan journal editor said the contracts practical effect was to reduce preventive dentistry more extractions </s>',
 '<s> the conclusions were challenged by mr joe rich chairman of the british dental associations general dental services committee who said it was probably true that more teeth were being extracted because dentists were seeing more patients </s>',
 '<s> dentists had no financial incentive to extract teeth he said </s>',
 '<s> the extraction fee was 9 but the filling fee could be almost anything the work needed </s>',
 '<s> mr rich said many dentists were concerned that not enough money was being made available by the government to implement the contract and he had approached mrs bottomley health secretary to arrange a meeting </s>',
 '<s> misconduct case gp to appeal </s>',
 '<s> dr robert jones the essex general practitioner suspended for eight months for serious professional misconduct is to appeal against his conviction </s>',
 '<s> dr jones of coggeshall was found guilty by the general medical council last month which heard that he failed to detect a serious case of appendicitis prescribing instead indigestion and ulcer tablets </s>',
 '<s> the nhs is not for sale virginia bottomley talks to george jones in her first interview since joining the cabinet </s>',
 '<s> by george jones </s>',
 '<s> the new health secretary mrs virginia bottomley set herself an ambitious target yesterday to take the politics the national health service </s>',
 '<s> but her desire to provide a period of stability for nearly one million people employed by the nhs is coupled with a fierce determination to maintain the momentum of reform with the overwhelming majority of hospitals becoming selfgoverning trusts by the mid1990s </s>',
 '<s> labour sought to turn the general election into a referendum on the nhs asking voters for a mandate to reverse the changes </s>',
 '<s> mrs bottomley is convinced the tory victory provides the opportunity to entrench the reforms and to give doctors nurses and managers the confidence to make them work </s>',
 '<s> in her first full interview since being appointed health secretary she said the election result and mr majors unequivocal commitment to the nhs had

```
finally nailed the lie about privatisation of health care </s>',
 '<s> the nhs is not for profit </s>',
 '<s> it is not for sale </s>',
 '<s> it will continue to be true to its founding principles available to all
and free at the point of delivery she said </s>',
 '<s> she recalled a promise made by mr major when he became prime minister that
he would work for a nation at ease with itself </s>']
```

[66]:
```
# Creating an empty list to store the sentences of the test corpus for easy␣
 ↪access
test_extracted_sentences = []
```

[67]:
```
# Looping through the corpus and moving the sentences to the new list, while␣
 ↪removing any punctuation marks that may appear
for i in range(len(test_corpus_sen)):
    for j in range(len(test_corpus_sen[i])):
        text_words = test_corpus_sen[i][j].split()
        text_words_string = ' '.join(text_words[1:-1])
        cleaned_text = "<s> " + remove_punctuation(text_words_string) + " </s>"
        test_extracted_sentences.append(cleaned_text)
```

[68]:
```
# Printing a subset for testing purposes
test_extracted_sentences[0:20]
```

[68]:
```
['<s> property cottage charms </s>',
 '<s> by sue fieldman </s>',
 '<s> the reputed birthplace of bob fitzsimmons world heavyweight boxing
champion in 1897 is one of the 100 or more properties to be auctioned by the
bristol and west between 1820 october </s>',
 '<s> the house right at 61 wendron street in helston cornwall has a guide price
of 5500070000 and a blue plaque in honour of fitzsimmons </s>',
 '<s> it will be auctioned at the alverton manor hotel truro on 19 october at
3pm </s>',
 '<s> ferrymans cottage left at noss mayo in south devon is to be auctioned on
its own by strutt parker 0392 215631 on 26 october </s>',
 '<s> the twobedroom cottage needs refurbishment but it is in a superb location
in the middle of woodland owned by the national trust and overlooking the river
yealm </s>',
 '<s> the guide price is 150000 </s>',
 '<s> property forward planning making proposals influencing the decisionmakers
allison flight fills in the background to a planning application </s>',
 '<s> by allison flight </s>',
 '<s> the planning system is designed around the idea that people should have a
say in what happens to their immediate surroundings </s>',
 '<s> but how do you find out about whether a planning application has been made
in the first place </s>',
 '<s> whether you are proposing or opposing an application what is the best way
```

```
to influence the decision that is eventually made </s>',
 '<s> finding out about an application under section 28 of the town and country
planning act 1971 the planning authority usually the local council is supposed
to advertise certain planning applications in the local newspaper </s>',
 '<s> the proposals covered are development in a conservation area works to a
listed building or which affect the setting of a conservation area or a listed
building and developments in areas of outstanding natural beauty </s>',
 '<s> the planning authority also has to advertise locally any proposals
involving large buildings those 20 metres high or any unneighbourly uses a
casino scrap yard cemetery or slaughterhouse and put up a notice on the site
itself </s>',
 '<s> some local authorities publish a list of all their new planning
applications in the local paper </s>',
 '<s> often they circulate these lists to local conservation and amenity groups
residents associations and subscribers </s>',
 '<s> ring the local planning authority to find out </s>',
 '<s> when the council receives a planning application it often consults people
who might be affected neighbours other residents and community groups </s>']
```

1.4.16 d) Calculating Linear Interpolation

[69]:
```
lmbd_1 = 0.6
lmbd_2 = 0.3
lmbd_3 = 0.1
```

[70]:
```
def linear_interpolation(sentence, model, lambdas, tri_dict, bi_dict, uni_dict):
    trigram_lambda, bigram_lambda, unigram_lambda = lambdas
    words = sentence.split()
    #trigram_prob = sum(trigram_probability(words[i-2] + " " + words[i-1] + " "␣
 ↪+ words[i], model, tri_dict, bi_dict) for i in range(len(words) - 2))
    trigram_prob = reduce(operator.mul, (trigram_probability(words[i-2] + " " +␣
 ↪words[i-1] + " " + words[i], model, tri_dict, bi_dict) for i in␣
 ↪range(len(words) - 2)), 1)
    #bigram_prob = sum(bigram_probability(words[i-1] + " " + words[i], model,␣
 ↪bi_dict, uni_dict) for i in range(len(words) - 1))
    bigram_prob = reduce(operator.mul, (bigram_probability(words[i-1] + " " +␣
 ↪words[i], model, bi_dict, uni_dict) for i in range(len(words) - 1)), 1)
    #unigram_prob = sum(unigram_probability(word, model, uni_dict) for word in␣
 ↪words)
    unigram_prob = reduce(operator.mul, (unigram_probability(word, model,␣
 ↪uni_dict) for word in words), 1)

    probability =  trigram_lambda * trigram_prob + bigram_lambda * bigram_prob␣
 ↪+ unigram_lambda * unigram_prob

    return probability
```

1.4.17 e) Getting n_gram probability (used for linear interpolation, perplexity and sentence probability)

[71]:
```
def unigram_probability(word, model, unigram_dict):
    total_words = len(unigram_dict.keys())
    if model == 3:
        if word not in unigram_dict.keys():
            word_count = unigram_dict.get("UNK", 0.001)
        else:
            word_count = unigram_dict.get(word.lower(), 0.001)
    else:
        word_count = unigram_dict.get(word.lower(), 0.001)
    return word_count / total_words
```

[72]:
```
def bigram_probability(phrase, model, bigram_dict, unigram_dict):
    words = phrase.split()
    first_word = words[0]

    if model == 3:
        if phrase not in bigram_dict.keys():
            phrase_count = bigram_dict.get("UNK", 0.001)
        else:
            phrase_count = bigram_dict.get(phrase.lower(), 0.001)

        if first_word not in unigram_dict.keys():
            word1_count = unigram_dict.get("UNK", 0.001)
        else:
            word1_count = unigram_dict.get(first_word.lower(), 0.001)
    else:
        phrase_count = bigram_dict.get(phrase.lower(), 0.001)
        word1_count = unigram_dict.get(first_word.lower(), 0.001)

    return phrase_count / word1_count
```

[73]:
```
def trigram_probability(phrase, model, trigram_dict, bigram_dict):
    words = phrase.split()
    phrase_2 = words[0] + " " + words[1]

    if model == 3:
        if phrase not in trigram_dict.keys():
            phrase_count = trigram_dict.get("UNK", 0.001)
        else:
            phrase_count = trigram_dict.get(phrase.lower(), 0.001)

        if phrase_2 in bigram_dict.keys():
            phrase_2_count = bigram_dict.get("UNK", 0.001)
        else:
```

```
            phrase_2_count = bigram_dict.get(phrase_2.lower(), 0.001)
    else:
        phrase_count = trigram_dict.get(phrase.lower(), 0.001)
        phrase_2_count = bigram_dict.get(phrase_2.lower(), 0.001)

    return phrase_count / phrase_2_count
```

[74]:
```
li_1 = linear_interpolation(train_extracted_sentences[150], 1, [lmbd_1, lmbd_2,␣
↪lmbd_3], train_vanilla_dict_tri_sen, train_vanilla_dict_bi_sen,␣
↪train_vanilla_dict_sen)
li_2 = linear_interpolation(train_extracted_sentences[150], 2, [lmbd_1, lmbd_2,␣
↪lmbd_3], train_laplace_dict_tri_sen, train_laplace_dict_bi_sen,␣
↪train_laplace_dict_sen)
li_3 = linear_interpolation(train_extracted_sentences[150], 3, [lmbd_1, lmbd_2,␣
↪lmbd_3], train_unk_dict_tri_sen, train_unk_dict_bi_sen, train_unk_dict_sen)
print(li_1, li_2, li_3)
```

```
7.1428571428756775e-06 1.1428571428598619e-05 6.461032429704854e+37
```

From the above results, it is noted that the linear interpolation from the UNK language models is very high, while the result from the other two types of models is at the proper range.

1.4.18 f) Calculating Perplexity

[75]:
```
rows, cols = (3, 4)
table_vals = [[0 for i in range(cols)] for j in range(rows)]

table_vals
```

[75]:
```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

[76]:
```
def calculate_perplexity(sen, ngram_type, model_num, unigram_model,␣
↪bigram_model, trigram_model):
    words = sen.split()
    N = len(words)
    total_prob = 1

    if ngram_type == 1:
        for word in words:
            unigram_prob = unigram_probability(word, model_num, unigram_model)
            total_prob *= unigram_prob
            #print("Unigram prob:", unigram_prob)
            #print("Total prob:", total_prob)
    elif ngram_type == 2:
        for i in range(1, N):
            bigram = words[i-1]+" "+words[i]
            bigram_prob = bigram_probability(bigram, model_num, bigram_model,␣
↪unigram_model)
```

```
            total_prob *= bigram_prob
            #print("Bigram prob:", bigram_prob)
            #print("Total prob:", total_prob)
    elif ngram_type == 3:
        for i in range(2, N):
            trigram = words[i-2]+" "+words[i-1]+" "+words[i]
            trigram_prob = trigram_probability(trigram, model_num,␣
 ↪bigram_model, unigram_model)
            total_prob *= trigram_prob
            #print("Trigram prob:", trigram_prob)
            #print("Total prob:", total_prob)

    if total_prob > 0:
        avg_nll = -(math.log(total_prob))/N
    else:
        # Handling the case when total_prob is zero
        avg_nll = 100
    perplexity = np.exp(avg_nll)

    return perplexity
```

[77]:
```
sentence_test = train_extracted_sentences[140]

sen_perp = calculate_perplexity(sentence_test, 1, 1, train_vanilla_dict_sen,␣
 ↪train_vanilla_dict_bi_sen, train_vanilla_dict_tri_sen)
sen_perp
```

[77]:
```
35.25067383867051
```

[78]:
```
def fill_table(test_corpus, unigram_dict, bigram_dict, trigram_dict, lambdas,␣
 ↪table_list, index):
    perplexity_table = {'Unigram': 0, 'Bigram': 0, 'Trigram': 0, 'Linear␣
 ↪Interpolation': 0}
    for sentence in test_corpus:
        words = sentence.split()
        total_words = len(words)
        model_num = index+1

        trigram_perp = calculate_perplexity(sentence, 3, model_num,␣
 ↪unigram_dict, bigram_dict, trigram_dict)
        bigram_perp = calculate_perplexity(sentence, 2, model_num,␣
 ↪unigram_dict, bigram_dict, trigram_dict)
        unigram_perp = calculate_perplexity(sentence, 1, model_num,␣
 ↪unigram_dict, bigram_dict, trigram_dict)
        interpolation_prob = linear_interpolation(sentence, model_num, lambdas,␣
 ↪trigram_dict, bigram_dict, unigram_dict)
```

```
        perplexity_table['Unigram'] += unigram_perp
        perplexity_table['Bigram'] += bigram_perp
        perplexity_table['Trigram'] += trigram_perp
        if interpolation_prob > 0:
            perplexity_table['Linear Interpolation'] += math.exp(-1 * math.
 ↪log(interpolation_prob) / total_words)
        else:
            # Handle the case when interpolation_prob is zero or negative
            # For example, set perplexity_table['Linear Interpolation'] to a␣
 ↪large value
            perplexity_table['Linear Interpolation'] = np.exp(-1 * math.log(0.
 ↪001) / total_words)

    table_list[index][0] = perplexity_table['Unigram']
    table_list[index][1] = perplexity_table['Bigram']
    table_list[index][2] = perplexity_table['Trigram']
    table_list[index][3] = perplexity_table['Linear Interpolation']

    #perplexity_table = {model: calculate_perplexity(perplexity_table[model])␣
 ↪for model in perplexity_table}

    return table_list
```

[79]:
```
vanilla_models = [train_vanilla_dict_sen, train_vanilla_dict_bi_sen,␣
 ↪train_vanilla_dict_tri_sen]
laplace_models = [train_laplace_dict_sen, train_laplace_dict_bi_sen,␣
 ↪train_laplace_dict_tri_sen]
unk_models = [train_unk_dict_sen, train_unk_dict_bi_sen, train_unk_dict_tri_sen]
```

[80]:
```
for model_num in range(0, 3):
    if model_num == 0:
        table_vals = fill_table(test_extracted_sentences, vanilla_models[0],␣
 ↪vanilla_models[1], vanilla_models[2], [lmbd_1, lmbd_2, lmbd_3], table_vals,␣
 ↪0)
    elif model_num == 1:
        table_vals = fill_table(test_extracted_sentences, laplace_models[0],␣
 ↪laplace_models[1], laplace_models[2], [lmbd_1, lmbd_2, lmbd_3], table_vals,␣
 ↪1)
    elif model_num == 2:
        table_vals = fill_table(test_extracted_sentences, unk_models[0],␣
 ↪unk_models[1], unk_models[2], [lmbd_1, lmbd_2, lmbd_3], table_vals, 2)

table_vals
```

[80]:
```
[[2.6881171418161356e+43, 6.4514811403587235e+44, 60869.0, 681002.7364283614],
 [2.6881171418161356e+43, 6.4514811403587235e+44, 60869.0, 679251.8956738408],
```

```
 [345928.40067730617,
  708000.7574037059,
  264.0583362368077,
  24505.139273954635]]
```

1.4.19 Saving Results in a DataFrame in order to resemble a table

[81]:
```
perp_table_df = pd.DataFrame(table_vals, columns=['Unigram', 'Bigram',␣
↪'Trigram', 'Linear Interpolation'], index=['Vanilla', 'Laplace', 'UNK'])

max_value = np.nanmax(perp_table_df[perp_table_df != np.inf])
perp_table_df.replace([np.inf, -np.inf], max_value, inplace=True)

perp_table_df
```

[81]:

	Unigram	Bigram	Trigram	Linear Interpolation
Vanilla	2.688117e+43	6.451481e+44	60869.000000	681002.736428
Laplace	2.688117e+43	6.451481e+44	60869.000000	679251.895674
UNK	3.459284e+05	7.080008e+05	264.058336	24505.139274

1.4.20 g) Sentence Generation and Calculating Sentence Probability

[82]:
```
def select_lm():
    print("1. Vanilla LM")
    print("2. Laplace LM")
    print("3. UNK LM")
    lm_choice = input("Please choose a LM option from the above: ")

    return lm_choice
```

[83]:
```
def enter_phrase():
    # Get the user's input phrase
    phrase = input("Enter a phrase: ")

    return phrase
```

[84]:
```
def get_vocabulary(phrase):
    # Initialize the vocabulary
    vocabulary = set()

    # Build the vocabulary from the input phrase
    for word in phrase.strip().split():
        vocabulary.add(word)

    return vocabulary
```

[85]:
```python
def generate_sentence(model, start_ngram, vocab, max_length=100):
    sentence = list(start_ngram.split())
    while True:
        next_ngram = sentence[-2:]
        # Check if the next n-gram is empty
        if not next_ngram:
            break
        probs = compute_next_word_probs(model, next_ngram, vocab)
        if not probs:
            break
        next_word = np.random.choice(list(probs.keys()), p=list(probs.values()))
        sentence.append(next_word)
        # Check if the sentence has reached the maximum length
        if len(sentence) >= max_length:
            break
    return sentence + ['</s>']
```

[86]:
```python
def generate_sentence_new(model, vocab, start_phrase, num_words=50):
    # Tokenize the starting phrase
    tokens = start_phrase.strip().split()

    # Generate the rest of the sentence
    for _ in range(num_words):
        # Get the previous n-gram
        previous_ngram = tuple(tokens[-2:])

        # Compute the probabilities of the next word
        probs = compute_next_word_probs(model, previous_ngram, vocab)

        # Sample the next word from the distribution
        next_word = np.random.choice(list(probs.keys()), p=list(probs.values()))

        # Add the next word to the tokens
        tokens.append(next_word)

        # Check if we have reached the end of the sentence
        if next_word == '</s>':
            break

    # Join the tokens into a sentence
    sentence = ' '.join(tokens[:-1])

    return sentence
```

[87]:
```python
def compute_next_word_probs(freq_counts, prev_ngram, vocab):
    # Check if freq_counts is None
    if freq_counts is None:
```

```
        return {}
    if not prev_ngram:
        return {}
    if len(prev_ngram) == 1:
        prev_word = prev_ngram[0]
    else:
        prev_word = prev_ngram[-1]
    bigram = prev_ngram[0], prev_word
    trigrams = {tuple(prev_ngram) + (word,) for word in freq_counts if word not␣
↪in prev_ngram}
    total_count = sum(freq_counts.values())
    probs = {word: freq_counts.get(word,0) / total_count for word in␣
↪freq_counts}

    for trigram in trigrams:
        total_count = sum(freq_counts.get(word,0) for word in trigram)
        probs[trigram[-1]] += freq_counts.get(trigram[-1],0) / total_count

    return probs
```

[88]:
```
def sen_probability(freq_counts, sentence, vocab):
    # Tokenize the sentence
    tokens = sentence.strip().split() + ['</s>']

    # Initialize the probability
    prob = 1.0

    # Iterate over each token in the sentence
    for i in range(len(tokens)):
        # Get the current and previous n-grams
        current_ngram = tuple(tokens[i:i+2])
        previous_ngram = tokens[i-1:i+1] if i > 0 else ('<s>', tokens[i])

        # Compute the probability of the current word given the previous n-gram
        probs = compute_next_word_probs(freq_counts, previous_ngram, vocab)
        prob *= probs.get(tokens[i], 0.0)

    return prob
```

1.4.21 Calculating the Probability of a Test Sentence

[89]:
```
sentence = test_extracted_sentences[50]
```

[90]:
```
vanilla_probability = sen_probability(train_vanilla_dict_sen, sentence,␣
↪get_vocabulary(sentence))

vanilla_probability
```

[90]: 5.833827127912008e-70

[91]:
```
laplace_probability = sen_probability(train_laplace_dict_sen, sentence,␣
 ↪get_vocabulary(sentence))

laplace_probability
```

[91]: 3.7449306125222067e-70

[92]:
```
unk_probability = sen_probability(train_unk_dict_sen, sentence,␣
 ↪get_vocabulary(sentence))

unk_probability
```

[92]: 5.833827127912008e-70

As expected, the sentence probability calculated using the vanilla and laplace smoothing models is very low, especially because a sentence from the test corpus is used. However, when using the UNK model, the probability is extremely high, possibly due to the high count of the < UNK > tags found in the model dictionaries. This is very similar to the calculaiton of linear interpolation.

[93]:
```
lm_num = select_lm()
model = None

if lm_num == 1:
    model = train_vanilla_dict_sen
elif lm_num == 2:
    model = train_laplace_dict_sen
elif lm_num == 3:
    model = train_unk_dict_sen

inp_phrase = enter_phrase()

phrase_vocabulary = get_vocabulary(inp_phrase)
```

```
1. Vanilla LM
2. Laplace LM
3. UNK LM

Please choose a LM option from the above:  2
Enter a phrase:  <s> i love to eat
```

[94]:
```
gen_sentence = generate_sentence(model, inp_phrase, phrase_vocabulary)

gen_sentence
```

[94]: ['<s>', 'i', 'love', 'to', 'eat', '</s>']

[95]:
```
sequence_prob = sen_probability(model, inp_phrase, phrase_vocabulary)

sequence_prob
```

[95]: 0.0

1.5 Testing

1.5.1 1. Corpus Text Extraction

This part was tested using two checks, one for getting all the corpus files and another for getting the appropriate text. The first check was done by getting the file names of all the files in all the subdirectories of the main directory, putting them in a list, one for each category, and then printing them out. The second one was made by executing the extract text function on one of the files and then printing the returned text. The result of this was a list of the sentences of that file, including the < s > tags. Initially, all the sentences were put into as one whole element in the list, instead of a 2D list. This was because the words, which were contained inside the < w > tags, became grouped together. I fixed this by getting the whole text and separating the < s > tags, without including any unnecessary tags but not removing any content inside them.

1.5.2 2. Building Frequency Counts

This part was tested by simpling printing the dictionaries generated by the build_frequency_counts function. Although the frequency counts were correctly returned, when the printing of the trigram frequency counts was attempted, it would always be unsuccessful due to the data rate limit. This was fixed by simply printing a portion of the dictionary. The same thing was done for the Laplace Smoothing frequency counts. This was not needed for the UNK frequency counts due to the decreased number of items in the trigram dictionary.

1.5.3 3. Linear Interpolation + Perplexity

These two parts were tested by executing the linear_interpolation and calculate_perplexity functions on each of the different models on a test sentence. The tests ended up being successful. However, when these were implemented to create the perplexity table, there was one signficant problem that was encountered, which mainly took place on the n-gram probability functions. The issue was that the second parameter of the get function of the n-gram dictionaries was set too low to the point where the total probability would be exactly 0 after a number of executions. This was fixed by checking if the probability in the calculate_perplexity and fill_table functions is greater than 0 and setting the average nll to 100 and the interpolated probability to 0.001.

1.5.4 4. Sentence Probability + Sentence Generation

The sentence probability part was tested by taking a few sentences from the test corpus and calculating their probability using the unigram models, as it was the only way the task could be done. This was successful on the vanilla and laplace models. However, it was not the case for the UNK model for some sentences. In fact, when attempting to calculate the probability for these sentences, it will give an errr stating that a word could not be found. Various solutions were attempted, like trying to get the count from the < UNK > tag, but unfortunately I could not get it to work.

The sentence generation was tested by inputting a phrase and attempting to generate a sentence. it was made sure that the input phrase begins with the < s > tag, to mark the start of the sentence. The sentence probability is also calculated. However, this did not work as well as intended, as instead of finding the next words to add to the input phrase, it just adds the < /s > tag. The function which ended up being used is called: generate_sentence(). Multiple solutions were intended, but unfortunately I could not get this to work either, as these generated lots of errors. The best attempted solution can be found in the function called: generate_sentence_new(). Despite this, the sentence probability worked without any errors, although the result was always 0, which was not intended. This could not be solved either.

[ ]: