```
from syllable import Encoder
from collections import Counter
import math
from itertools import tee
import random


 #download sylable repository
from syllable import Encoder #import syllable repository

encoder = Encoder(lang="tr", limitby="vocabulary", limit=3000)  #
params chosen for demonstration purposes

#example about syllable encoder
print(encoder.tokenize("Encoder çalışma örneğidir. Dikkate almayını
"))
print(encoder.tokenize("Zaman çok hızlı geçiyor."))
```

```
en co der ça lış ma ör ne ği dir dik ka te al ma yı nı a mi a ne
za man çok hız lı ge çi yor
```

Syllable Extraction

```
def extract_and_store_syllabic_data(source_path, destination_path):
    # Initialize a tokenizer object with language settings and token
limits
    tokenizer_tool = Encoder(lang="tr", limitby="vocabulary",
limit=3000)

    try:
        # Read content from the source file
        with open(source_path, 'r', encoding='utf-8') as source:
            raw_content = source.read()

            # Perform syllable extraction using the tokenizer
            segmented_text = tokenizer_tool.tokenize(raw_content)

            # Save the segmented content to the destination file
            with open(destination_path, 'w', encoding='utf-8') as
destination:
                destination.write(segmented_text)

            print(
                f"Syllable extraction from '{source_path}' was
completed, "
                f"and the results have been saved to
'{destination_path}'."
            )
```

```
    except FileNotFoundError:
        print(f"Error: '{source_path}' was not found.")
    except Exception as error:
        print(f"An unexpected error occurred: {error}")

# Example usage of the function
input_filename = "wiki_00"
output_filename = "syllable.txt"

extract_and_store_syllabic_data(input_filename, output_filename)
```

```
Syllable extraction from 'wiki_00' was completed, and the results have
been saved to 'syllable.txt'.
```

Converted from uppercase to lowercase and Turkish characters have been replaced with their English equivalents.

```
def lowercase_converter(input_string):
    return input_string.lower()

def turkish_to_english_mapper(turkish_text):
    tr_chars = "çğıöşü"
    en_chars = "cgiosu"
    mapping = str.maketrans(tr_chars, en_chars)
    return turkish_text.translate(mapping)

def file_handler_and_transformer(input_filename):
    try:
        with open(input_filename, 'r', encoding='utf-8') as
source_file:
            raw_data = source_file.read()
            transformed_lower = lowercase_converter(raw_data)
            translated_text =
turkish_to_english_mapper(transformed_lower)

        output_filename = "syllable_output.txt"
        with open(output_filename, 'w', encoding='utf-8') as
result_file:
            result_file.write(translated_text)

        print(
    f"The file '{input_filename}' has been converted from uppercase to
lowercase, "
    f"and Turkish characters have been replaced with their English
equivalents. "
    f"The modified content has been saved to '{output_filename}'."
)


    except FileNotFoundError:
```

```
        print(f"Error: '{input_filename}' does not exist or could not
be found.")

# Example usage: read input filename and process the file
target_file = "syllable.txt"
file_handler_and_transformer(target_file)
```

```
The file 'syllable.txt' has been converted from uppercase to
lowercase, and Turkish characters have been replaced with their
English equivalents. The modified content has been saved to
'syllable_output.txt'.
```

Divide to Wikipedia syllable Train and Test Dataset

```
def divide_and_save(input_filename, output_file1, output_file2,
split_ratio=0.95):
    try:
        # Read the input file
        with open(input_filename, 'r', encoding='utf-8') as
input_file:
            content = input_file.read()

            # Calculate the split index
            split_index = int(len(content) * split_ratio)
            content_part1 = content[:split_index]
            content_part2 = content[split_index:]

            # Write the first part to output_file1
            with open(output_file1, 'w', encoding='utf-8') as file1:
                file1.write(content_part1)

            # Write the second part to output_file2
            with open(output_file2, 'w', encoding='utf-8') as file2:
                file2.write(content_part2)

            print(f"Content from {input_filename} has been
successfully saved to {output_file1} and {output_file2}.")
    except FileNotFoundError:
        print(f"Error: {input_filename} not found.")
    except Exception as error:
        print(f"An error occurred: {error}")

# Example of usage
input_filename = "syllable_output.txt"
output_file1 = "text_95_percent.txt"
output_file2 = "text_5_percent.txt"

divide_and_save(input_filename, output_file1, output_file2)

def read_dataset(file_name):
```

```
    try:
        with open(file_name, 'r', encoding='utf-8') as file:
            return file.read()
    except FileNotFoundError:
        print(f"{file_name} dosyası bulunamadı.")
        return None

file_name = "text_95_percent.txt"
dataset_95 = read_dataset(file_name)

file_name = "text_5_percent.txt"
dataset_5 = read_dataset(file_name)
```

```
Content from syllable_output.txt has been successfully saved to
text_95_percent.txt and text_5_percent.txt.
```

Tokanization , Create N gram and good_turing_smoothing functions

```
def tokenize_string(input_string):
    segmented_tokens = []
    temp_token = ""

    for character in input_string:
        if character == " ":
            if temp_token:
                segmented_tokens.append(temp_token)
                segmented_tokens.append(" ")
            temp_token = ""
        else:
            temp_token += character

    if temp_token:
        segmented_tokens.append(temp_token)

    return segmented_tokens


def sliding_window(sequence, window_size):
    iterables = tee(sequence, window_size)
    for shift in range(1, window_size):
        for it in iterables[shift:]:
            next(it, None)
    return zip(*iterables)

def generate_ngram_frequency(token_sequence, n):
    # Boşluk karakterlerini filtreleyerek ngram frekansı hesapla
    filtered_tokens = [token for token in token_sequence if
token.strip()]
    ngram_freq_table = Counter(map(''.join,
sliding_window(filtered_tokens, n)))
```

```
    return ngram_freq_table


def good_turing_smoothing(freq_table, threshold=5):
    total_count = sum(freq_table.values())

    infrequent_keys = [key for key, count in freq_table.items() if
count <= threshold]
    for key in infrequent_keys:
        freq_table[key] += threshold

    for key in freq_table:
        freq_table[key] = (freq_table[key] - threshold) / total_count

    return freq_table

tokens_95 = tokenize_string(dataset_95)
```

Create One-Gram Table

```
unigram = generate_ngram_frequency(tokens_95, 1)

unigram = unigram.most_common()

for i, item in enumerate(list(unigram)):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
le: 3102700
la: 3100303
ri: 2838302
si: 2813910
da: 2577927
a: 2470544
de: 2463675
li: 2320910
di: 2206999
ki: 2032904
ya: 1998655
i: 1983712
o: 1889967
ve: 1804155
ma: 1709854
ra: 1633537
ta: 1632511
ni: 1613517
ti: 1458910
gi: 1441334
```

```
ka: 1410048
sa: 1186316
ne: 1174316
te: 1130727
bir: 1129599
e: 1116526
nin: 1093917
na: 1077230
dir: 1076400
bu: 1075638
re: 975631
me: 936421
se: 908965
ye: 881004
ci: 859904
ge: 827471
u: 818653
lan: 807566
ce: 791932
lu: 790251
mi: 790232
rin: 790056
mis: 779460
du: 778162
lar: 767801
bi: 760503
yi: 743551
ler: 715333
ha: 698360
tir: 690862
ca: 687544
lik: 665851
ba: 659802
gu: 655390
sin: 633541
dan: 632437
su: 617669
wi: 601694
nu: 596410
cu: 588850
ku: 567478
ko: 556921
rak: 542120
in: 518764
den: 513266
ol: 508579
ke: 446428
ru: 433478
mu: 430535
```

```
tur: 427281
be: 423810
al: 414904
lin: 404810
pe: 395191
zi: 393611
is: 387083
do: 382027
go: 379474
an: 373293
yo: 369880
mak: 362744
man: 358817
cin: 356623
tu: 352056
bas: 349255
pa: 342165
len: 341018
son: 322891
id: 314251
pi: 311330
nun: 307218
tan: 305326
ro: 304847
ga: 300023
va: 299210
rid: 297816
ze: 296564
tit: 294742
za: 292405
org: 291252
```

One-Gram Good-Turing-Smoothing

```python
gt_smooth_unigram = good_turing_smoothing(dict(unigram))

for i, item in enumerate(gt_smooth_unigram.items()):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
le: 0.023067242678311405
la: 0.02304942198349612
ri: 0.021101553872399068
si: 0.02092020953033213
da: 0.01916577439282878
a: 0.018367426595019096
de: 0.01831635844621384
li: 0.01725496024208191
```

di: 0.016408079488179534
ki: 0.01511375580696671
ya: 0.014859128782882973
i: 0.014748033813077044
o: 0.014051078854604394
ve: 0.013413102440966811
ma: 0.012712013854493618
ra: 0.012144628803922843
ta: 0.01213700092203695
ni: 0.011995788457572397
ti: 0.010846349924695112
gi: 0.010715679698548487
ka: 0.010483081343107926
sa: 0.008819727278688457
ne: 0.008730512285871007
te: 0.008406446259044355
bir: 0.008398060049719516
e: 0.00830086774962764
nin: 0.008132779268576829
na: 0.00800871838648143
dir: 0.008002547682811558
bu: 0.007996882530767649
re: 0.007253372215209759
me: 0.006961862226178742
se: 0.006757738322612417
ye: 0.006549859954765025
ci: 0.006392990259061009
ge: 0.006151864437223647
u: 0.006086306286668291
lan: 0.0060038790678877026
ce: 0.0058876468014120355
lu: 0.005875149267834858
mi: 0.005875008010762897
rin: 0.005873699524201574
mis: 0.005794922685543766
du: 0.005785272597154012
lar: 0.005708242885438879
bi: 0.0056539853006404
yi: 0.005527954254120283
ler: 0.00531816519851005
ha: 0.005191978025752502
tir: 0.005136233524407066
ca: 0.005111565578893041
lik: 0.004950287175627297
ba: 0.004905315384664567
gu: 0.0048725140056386845
sin: 0.004710075807466314
dan: 0.004701868028127108
su: 0.0045920741102997665

```
wi: 0.004473306651111537
nu: 0.00443402231594092
cu: 0.004377816870465926
ku: 0.004218924968258049
ko: 0.004140438078326898
rak: 0.004030398819269309
in: 0.0038567567049156122
den: 0.0038158813690397507
ol: 0.0037810354797618018
ke: 0.003318968728212026
ru: 0.0032226908817965284
mu: 0.003200810904808049
tur: 0.0031766187725890504
be: 0.003150813335916603
al: 0.0030846009420805857
lin: 0.0030095562639556412
pe: 0.0029380430126297206
zi: 0.002926296371908756
is: 0.0028777634158160636
do: 0.002840174165508978
go: 0.0028211936757870657
an: 0.0027752405199033443
yo: 0.002749866289029515
mak: 0.0026968131066340715
man: 0.002667617500234561
cin: 0.0026513060257144375
tu: 0.002617352286364663
bas: 0.0025965280201245234
pa: 0.00254381682853488
len: 0.002535289362138079
son: 0.0024005226809045866
id: 0.0023362878860760227
pi: 0.0023145714699077087
nun: 0.0022840004657022623
tan: 0.0022699342351680447
ro: 0.0022663730700380816
ga: 0.0022305086429254667
va: 0.0022244643271620845
rid: 0.002214100518829791
ze: 0.002204792421245837
tit: 0.002191246611503054
za: 0.0021738719916518556
org: 0.002165299917758646
```

Create Two-Gram Table

```
bigram = generate_ngram_frequency(tokens_95, 2)

bigram = bigram.most_common()
```

```python
for i, item in enumerate(list(bigram)):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
leri: 690466
lari: 682277
wiki: 594639
mistir: 451020
ola: 411135
ile: 406326
larak: 376701
dia: 343723
larin: 320354
sinda: 314093
kipe: 295616
pedi: 292747
title: 289535
kicu: 289381
aorg: 289357
orgwi: 289321
trwi: 289042
curid: 289016
idurl: 289013
urltr: 289013
ridtit: 289013
icin: 287640
ligi: 286366
lerin: 283562
masi: 264736
digi: 246687
yilin: 233454
olan: 232975
ara: 231385
linda: 221636
oldu: 221568
sonra: 214546
tadir: 209522
rini: 208912
makta: 199603
rinin: 186830
sinde: 186703
tara: 186396
daki: 183806
rafin: 180322
sine: 180107
sini: 175125
dugu: 170869
```

findan: 170518
daha: 170144
sinin: 166689
rine: 166576
rinde: 158170
rasin: 151466
mesi: 150367
mekte: 147656
lara: 144149
ise: 143359
lama: 142394
deki: 138922
rinda: 136330
tedir: 133776
kulla: 132004
gini: 131522
lani: 130289
uze: 128192
tari: 127992
yapi: 127435
kara: 126654
tesi: 126400
kisi: 121713
vardir: 120427
buyuk: 119835
gore: 119548
larda: 118891
yasa: 116046
halle: 115072
mahal: 115036
sina: 114702
mustur: 113952
gibi: 113700
daya: 111900
yeni: 111785
basla: 111335
iki: 111059
ladi: 109129
libir: 106737
sii: 106177
yada: 105365
maya: 103715
tigi: 103479
karsi: 103405
olma: 102939
bulu: 101979
ria: 101821
bulun: 101691
gunu: 99939

```
sira: 98995
kendi: 98875
ilce: 98155
rii: 97642
lerde: 96862
cesi: 93548
ayri: 93406
dirid: 92422
```

Two-Gram Good-Turing-Smoothing

```
gt_smooth_bigram = good_turing_smoothing(dict(bigram))

for i, item in enumerate(gt_smooth_bigram.items()):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
leri: 0.005133289467807956
lari: 0.00507240766934015
wiki: 0.004420855702784829
mistir: 0.0033531083570591314
ola: 0.003056580022477569
ile: 0.003020827113840168
larak: 0.002800577598684626
dia: 0.0025553999274340114
larin: 0.0023816611622130878
sinda: 0.002335113239364519
kipe: 0.002197744453152571
pedi: 0.0021764146351278876
title: 0.0021525347552062133
kicu: 0.0021513898294565443
aorg: 0.0021512113994695825
orgwi: 0.0021509437544891406
trwi: 0.0021488695058907137
curid: 0.002148676206738172
idurl: 0.00214865390298980 2
urltr: 0.00214865390298980 2
ridtit: 0.00214865390298980 2
icin: 0.002138446220819049
ligi: 0.0021289745623445117
lerin: 0.0021081279922011823
masi: 0.0019681645365955034
digi: 0.0018339777518177015
yilin: 0.0017355959177568312
olan: 0.001732034752600392
ara: 0.0017202137659641959
linda: 0.0016477340183438964
oldu: 0.0016472284667141722
```

sonra: 0.0015950228263623674
tadir: 0.0015576714824251036
rini: 0.0015531363869231666
makta: 0.0014839278557304935
rinin: 0.0013889659297530508
sinde: 0.0013880217377387131
tara: 0.0013857393208221646
daki: 0.0013664837513959079
rafin: 0.0013405816649553373
sine: 0.0013389832296554743
sini: 0.0013019441381953928
dugu: 0.0012703025538408955
findan: 0.0012676930152815842
daha: 0.0012649124813181017
sinin: 0.0012392259977784426
rine: 0.0012383858899231658
rinde: 0.0011758907869899172
rasin: 0.0011260493439653517
mesi: 0.0011178787374790753
mekte: 0.0010977235835352213
lara: 0.0010716505016904793
ise: 0.0010657771812863314
lama: 0.001058602808893923
deki: 0.0010327899374468333
rinda: 0.0010135194988549962
tedir: 0.0009945315744091818
kulla: 0.0009813574937051945
gini: 0.0009777740248003853
lani: 0.0009686071842202408
uze: 0.0009530168641094839
tari: 0.0009515299475514719
yapi: 0.0009473888849374081
kara: 0.0009415824757783709
tesi: 0.0009396940917496955
kisi: 0.000904848202212682
vardir: 0.0008952873287446642
buyuk: 0.0008908860557329484
gore: 0.0008887523304722011
larda: 0.0008838678095791314
yasa: 0.0008627164215414092
halle: 0.0008554751379038903
mahal: 0.0008552074929234481
sina: 0.0008527243422715679
mustur: 0.0008471484051790225
gibi: 0.0008452748903159273
daya: 0.0008318926412938185
yeni: 0.0008310376642729615
basla: 0.0008276921020174343
iki: 0.0008256401571673775

```
ladi: 0.0008112914123825608
libir: 0.0007935078903487361
sii: 0.0007893445239863022
yada: 0.0007833076427607731
maya: 0.0007710405811571733
tigi: 0.0007692860196187191
karsi: 0.0007687358604922546
olma: 0.0007652713449120864
bulu: 0.0007581341454336283
ria: 0.0007569594813527987
bulun: 0.0007559929855900909
gunu: 0.0007429675965419049
sira: 0.0007359493503880879
kendi: 0.0007350572004532806
ilce: 0.000729704300844437
rii: 0.0007258903598731359
lerde: 0.0007200913852968888
cesi: 0.0006954531779306283
ayri: 0.0006943974671744398
dirid: 0.0006870818377090202
```

Create Tri-Gram Table

```
trigram = generate_ngram_frequency(tokens_95, 3)

trigram = trigram.most_common()

for i, item in enumerate(list(trigram)):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
olarak: 345861
kipedi: 291456
pedia: 289937
wikipe: 289690
diaorg: 289345
orgwiki: 289319
aorgwi: 289309
trwiki: 289042
wikicu: 289016
idurltr: 289013
urltrwi: 289013
kicurid: 289013
curidtit: 289013
ridtitle: 289013
yilinda: 205635
maktadir: 180281
tarafin: 174392
```

```
rafindan: 169694
rasinda: 147933
mektedir: 129956
larinda: 119336
oldugu: 114181
mahalle: 113473
arasin: 111380
diridurl: 92283
lerinde: 90268
lerini: 89548
larini: 85652
lerinin: 80297
tarihin: 77291
turkiye: 76929
rihinde: 74189
larina: 73201
larinin: 71540
kullani: 71187
lerine: 68893
ilcesi: 65704
ameri: 65679
ayrica: 60085
birlikte: 59263
univer: 58645
lerinden: 58377
cesine: 57621
bulunan: 57374
icinde: 56981
niversi: 55539
versite: 54663
tiridurl: 53878
istanbul: 53253
malari: 51564
onemli: 51422
riara: 50535
uzerin: 50270
calisma: 50006
yapilan: 49706
basladi: 49568
larindan: 49133
sebeke: 48965
mistirid: 48271
bekesi: 47998
ikinci: 47348
nebagli: 47287
dahason: 47074
hasonra: 47054
olmustur: 46571
dirmahal: 46443
```

```
avrupa: 45473
oyuncu: 45219
sitesi: 44776
merkezi: 44522
laria: 43662
iceri: 43427
risinde: 42952
ekono: 42375
uzeri: 42361
masina: 42339
dugunu: 42285
bulunmak: 42059
ozellik: 41740
hallenin: 41688
ligini: 41646
siile: 41407
aile: 40818
zerine: 40666
zerinde: 40479
riile: 40251
acenter: 39344
larii: 39307
dirkoyun: 38968
siyoktur: 38697
yeralan: 38488
rilmistir: 38421
lunmakta: 38115
sinebag: 38078
dilmistir: 37621
merika: 37611
larara: 37373
alani: 37350
okulu: 37346
ingiliz: 37120
```

Three-Gram Good-Turing-Smoothing

```python
gt_smooth_trigram = good_turing_smoothing(dict(trigram))

for i, item in enumerate(gt_smooth_trigram.items()):
    if i >= 100:  # İlk 100 öğeden sonra dur
        break
    print(f'{item[0]}: {item[1]}')
```

```
olarak: 0.0025712950845556666
kipedi: 0.002166816604855297
pedia: 0.0021555234735132355
wikipe: 0.002153687131550438
diaorg: 0.002151122200468798
```

orgwiki: 0.0021509289013148195
aorgwi: 0.002150854555486366
trwiki: 0.002148869521866662
wikicu: 0.0021486762227126833
idurltr: 0.0021486539189641473
urltrwi: 0.0021486539189641473
kicurid: 0.0021486539189641473
curidtit: 0.0021486539189641473
ridtitle: 0.0021486539189641473
yilinda: 0.0015287732704859299
maktadir: 0.001340276857025344
tarafin: 0.001296494598649175
rafindan: 0.001261566928441798
rasinda: 0.0010997829711444957
mektedir: 0.000966131475333935
larinda: 0.0008871762055164932
oldugu: 0.0008488509309487989
mahalle: 0.0008435872462943028
arasin: 0.0008280266643990198
diridurl: 0.0006860484358016857
lerinde: 0.0006710677513683387
lerini: 0.0006657148517196985
larini: 0.0006367497169542792
lerinin: 0.0005969375258175183
tarihin: 0.0005745891697844457
turkiye: 0.000571897850794435
rihinde: 0.0005515270937982212
larina: 0.000544181725947O317
larinin: 0.0005318328838409327
kullani: 0.00052920847609653
lerine: 0.0005121535430493349
ilcesi: 0.0004884446583555663
ameri: 0.000488258793784433
ayrica: 0.00044666973734763734
birlikte: 0.0004405585102487732
univer: 0.00043596393805035705
lerinden: 0.0004339714698478077
cesine: 0.00042835092521673556
bulunan: 0.00042651458325393816
icinde: 0.00042359279219572214
niversi: 0.0004128721237327512
versite: 0.00040635942916023905
tiridurl: 0.00040052328162665224
istanbul: 0.0003958766673483188
malari: 0.0003833196569225505
onemli: 0.0003822639461585131
riara: 0.000375694711747023
uzerin: 0.0003736993067206889
calisma: 0.0003717365768495209

```
yapilan: 0.0003695062019959208
basladi: 0.0003684802295632648
larindan: 0.00036524618602554475
sebeke: 0.0003639971761075287
mistirid: 0.00035883757561286725
bekesi: 0.0003568079344960912
ikinci: 0.0003519754556466244
nebagli: 0.0003515219460930591
dahason: 0.000349938379947003
hasonra: 0.00034978968829009634
olmustur: 0.00034619878477580026
dirmahal: 0.0003452471581715976
avrupa: 0.0003380356128116241
oyuncu: 0.0003361472287689094
sitesi: 0.0003328537085684266
merkezi: 0.0003309653245257119
laria: 0.0003245715832787251
iceri: 0.0003228244563100717
risinde: 0.00031929302945853826
ekono: 0.00031500327515678083
uzeri: 0.00031489919099694617
masina: 0.00031473563017434886
dugunu: 0.00031433416270070083
bulunmak: 0.00031265394697765545
ozellik: 0.00031028231504999406
hallenin: 0.00030989571674203673
ligini: 0.0003095834642625327
siile: 0.000307806598962498
aile: 0.0003034276296665966
zerine: 0.0003022975730741059
zerinde: 0.00030090730608202854
riile: 0.0002992122211932925
acenter: 0.000292469054552575
larii: 0.00029219397498729766
dirkoyun: 0.00028967365140272956
siyoktur: 0.0002876588794516442
yeralan: 0.0002861050516369695
rilmistir: 0.00028560693458633214
lunmakta: 0.0002833319522356601
sinebag: 0.0002830568726703827 4
dilmistir: 0.00027965926831006533
merika: 0.000279584922481612
larara: 0.00027781549176442264
alani: 0.00027764449635898
okulu: 0.0002776147580275986
ingiliz: 0.00027593454230455324
```

One gram perplexity Calculation

```
tokens_5 = tokenize_string(dataset_5)

def calculate_unigram_perplexity(token_list, unigram_prob_dist):
    cumulative_log_prob = 0
    token_count = len(token_list)

    for token in token_list:
        if token in unigram_prob_dist:
            cumulative_log_prob += math.log2(unigram_prob_dist[token])
        else:
            # Assign a small probability for out-of-vocabulary tokens
            cumulative_log_prob += math.log2(1e-10)

    mean_log_prob = cumulative_log_prob / token_count
    perplexity_score = 2 ** (-mean_log_prob)
    return perplexity_score

result_perplexity = calculate_unigram_perplexity(tokens_5,
gt_smooth_unigram)
print("Unigram Perplexity:", result_perplexity)
```

```
Unigram Perplexity: 1707021.107194859
```

Two gram perplexity Calculation

```
def compute_bigram_perplexity(token_sequence, bigram_prob_dist):
    cumulative_log_probability = 0
    total_tokens = len(token_sequence)

    for idx in range(1, total_tokens):
        previous_token = token_sequence[idx - 1]
        current_token = token_sequence[idx]
        bigram = previous_token + " " + current_token

        if bigram in bigram_prob_dist:
            cumulative_log_probability +=
math.log2(bigram_prob_dist[bigram])
        else:
            # Assign a small probability for unseen bigrams
            cumulative_log_probability += math.log2(1e-10)

    mean_log_probability = cumulative_log_probability / total_tokens
    perplexity_score = 2 ** (-mean_log_probability)
    return perplexity_score

result_perplexity = compute_bigram_perplexity(tokens_5,
gt_smooth_bigram)
print("Bigram Perplexity:", result_perplexity)
```

```
Bigram Perplexity: 9999983769.663683
```

Three gram perplexity Calculation

```
def calculate_trigram_perplexity(token_sequence, trigram_prob_dist):
    cumulative_log_prob = 0
    sequence_length = len(token_sequence)

    for idx in range(2, sequence_length):
        first_token = token_sequence[idx - 2]
        second_token = token_sequence[idx - 1]
        third_token = token_sequence[idx]
        trigram = f"{first_token} {second_token} {third_token}"

        if trigram in trigram_prob_dist:
            cumulative_log_prob +=
math.log2(trigram_prob_dist[trigram])
        else:
            # Assign a small probability for unseen trigrams
            cumulative_log_prob += math.log2(1e-10)

    mean_log_prob = cumulative_log_prob / sequence_length
    perplexity_value = 2 ** (-mean_log_prob)
    return perplexity_value

trigram_perplexity_result = calculate_trigram_perplexity(tokens_5,
gt_smooth_trigram)
print("Trigram Perplexity:", trigram_perplexity_result)
```

```
Trigram Perplexity: 9999967497.456444
```

Random generated word

```
random_selection = random.sample(
    list(dict(list(gt_smooth_unigram.items())[:5]).keys()), 5
)

print("Selected 5 Random Words (Unigram):")
for token in random_selection:
    print(token, end=" ")

print("\n")


random_selection = random.sample(
    list(dict(list(gt_smooth_unigram.items())[:5]).keys()), 5
)


for token in random_selection:
```

```
    print(token, end=" ")

random_selection = random.sample(
    list(dict(list(gt_smooth_bigram.items())[:5]).keys()), 5
)

print("\n\nSelected 5 Random Words (Bigram):")
for token in random_selection:
    print(token, end=" ")

print("\n")

random_selection = random.sample(
    list(dict(list(gt_smooth_bigram.items())[:5]).keys()) ,5
)


for token in random_selection:
    print(token, end=" ")

random_selection = random.sample(
    list(dict(list(gt_smooth_trigram.items())[:5]).keys()), 5
)

print("\n\nSelected 5 Random Words (Trigram):")
for token in random_selection:
    print(token, end=" ")

print("\n")

random_selection = random.sample(
    list(dict(list(gt_smooth_trigram.items())[:5]).keys()), 5
)
for token in random_selection:
    print(token, end=" ")
```

```
Selected 5 Random Words (Unigram):
da si le la ri

ri da si le la

Selected 5 Random Words (Bigram):
mistir lari leri ola wiki

wiki mistir lari leri ola

Selected 5 Random Words (Trigram):
olarak kipedi pedia diaorg wikipe

olarak diaorg wikipe kipedi pedia
```