یک مینی پیکره فرضی با متن علمی دلخواه، شامل ۵۰ تا ۱۰۰ کلمه انگلیسی(برش از مجله یا **web**) را در نظر بگیرید. این متن ورودی برنامه است.

برنامه‌ای بنویسید که:

۱- تعداد **unigram** و **Bigram**های این پیکره را محاسبه و به صورت جدول نشان دهد. (البته با کسر امتیاز جزیی می‌توان این جداول را به صورت دستی به دست آورد.)

۲- یک رشته تصادفی از کلمات این پیکره، که طول آن کمتر از ۵ کلمه باشد ایجاد کند و سپس احتمال رخداد این رشته از کلمات را با فرض مینی پیکره کنونی و تقریب **Bigram** محاسبه کند.

لینک گیتهاب پروژه : https://github.com/ghazal-pouresfandiyar/nGram-Estimation

کد برنامه حاوی توابع زیر است که در ادامه به توضیح آنها می‌پردازیم. (۶ تابع اول برای قسمت اول سوال و ۵ تابع بعدی برای قسمت دوم سوال می‌باشند.)

```
# calculate ngrams
def ngrams(lst, n):···

def extract_unigrams(file):···

def extract_bigrams(file):···

# change (tuple as key ---> value) to (srt as key ---> value) for unigram
def change_unigram_format(unigrams):···

def print_unigrams(unigrams):···

def print_bigrams(bigrams):···

# probability of hapenning "b a" in sentence
# add 1 to numerator for smoothing
# add |v| to denominator for smoothing
def p(a, b, unigrams, bigrams):···

# count "b a" in corpus
def cba(b, a, bigrams):···

# count b in corpus
def cb(b, unigrams):···

def random_sentence(n, unigrams):···

def calculate_p(test, unigrams, bigrams):···
```

- تابع ngrmas با دریافت کلمات پیکره و تعداد دنباله کلماتی که میخواهیم n-gram متناظر را میسازد. این تابع در توابع extract_unigrams و ectract_bigrams فراخوانی می‌شود.

```
# calculate ngrams
def ngrams(lst, n):
  tlst = lst
  while True:
    a, b = tee(tlst)
    l = tuple(islice(a, n))
    if len(l) == n:
      yield l
      next(b)
      tlst = b
    else:
      break
```

- این دو تابع به ترتیب برای استخراج unigrams و bigrams هستند.

نکته قابل ذکر در اینجا این است که (“<s>” , ”</s>“) که حاصل اتصال انتهای هر خط به ابتدای هر خط بعدی است از bigram ها حذف شده است چون دیتای مازادی بود که تولید شده بود.

```
def extract_unigrams(file):
    words = re.findall('\w+|<s>|</s>', open(file).read().lower())
    unigrams = change_unigram_format(dict(Counter(ngrams(words, 1))))
    return unigrams

def extract_bigrams(file):
    words = re.findall('\w+|<s>|</s>', open(file).read().lower())
    bigrams = dict(Counter(ngrams(words, 2)))
    del bigrams[("</s>", "<s>")]
    return bigrams
```

- در هنگام استخراج unigrams، کلید های دیکشنری به صورت tuple (عضو اول رشته و عضو دوم خالی بود) ذخیره شده بودند که هنگام چاپ جدول ظاهر مناسبی نداشتند به همین دلیل این تابع تعریف شد تا کلید را از حالت tuple به string در بیاورد.

```
# change (tuple as key ---> value) to (srt as key ---> value) for unigram
def change_unigram_format(unigrams):
    keys = unigrams.keys()
    values = unigrams.values()
    new_keys = []
    new_values = []
    for i in keys:
        new_keys.append(i[0])
    for i in values:
        new_values.append(i)

    new_unigrams = dict(zip(new_keys, new_values))
    return new_unigrams
```

- دو تابع بعدی برای چاپ کردن داده‌های unigrams و bigrams استخراج شده هستند.

```
def print_unigrams(unigrams):
    print("------------------Unigram values------------------")
    keys = unigrams.keys()
    values = unigrams.values()
    for i in keys:
        print(15*" " + i + " ---> "+str(unigrams[i]))

def print_bigrams(bigrams):
    print("------------------Bigram  values------------------")
    keys = bigrams.keys()
    values = bigrams.values()
    for i in keys:
        print(10*" " ,i , " ---> "+str(bigrams[i]))
```

- تابع p احتمال رخداد bigram که به معنای آمدن "$w_{n-1}\ w_n$" به صورت متوالی در جمله است را از رابطه زیر و به کمک توابع cba (برای صورت) و cb (برای مخرج) محاسبه می‌کند. بدنه این دو تابع در ادامه توضیح داده میشود.

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}\ w_n)}{C(w_{n-1}\ )}$$

برای اینکه این تخمین ممکن است بسیاری از احتمالات را صفر اعلام کند که نهایتا منجر به صفر شدن احتمال نهایی می‌شوند، می‌توان از روش add-one(Laplace) برای smooth کردن نتایج استفاده کرد که در آن |v| تعداد نوع کلمات در پیکره است.

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}\ w_n) + 1}{C(w_{n-1}\ ) + |v|}$$

```
# probability of hapenning "b a" in sentence
# add 1 to numerator for smoothing
# add |v| to denominator for smoothing
def p(a, b, unigrams, bigrams):
    numerator = cba(b,a,bigrams)+1
    denominator = cb(b,unigrams)+len(unigrams.keys())
    return numerator/denominator
```

- دو تابعی که در ادامه آمده‌اند در تابع p فراخوانی شده بودند که برای محاسبه صورت و مخرج در تابع p استفاده شدند. این دوتابع در واقع تعداد مد نظر را برای ما می‌شمارد.

```
# count "b a" in corpus
def cba(b, a, bigrams):
    t = (b, a)
    result = 0
    try:
        result = bigrams[t]
    except:
        result = 0
    return result

# count b in corpus
def cb(b, unigrams):
    return unigrams[b]
```

- تابع بعدی یک جمله تست با ماکزیمم طول ۵ تولید می‌کند. جمله به صورت لیستی از کلمات ذخیره می‌شود.

```
def random_sentence(n, unigrams):
    length = random.randint(1,n)
    test = []
    test.append("<s>")
    for i in range(length):
        rand_word = "<s>"
        while(rand_word == "<s>" or rand_word == "</s>"):
            rand_word = random.choice(list(unigrams.keys()))
        test.append(rand_word)
    test.append("</s>")
    print("The test sentences is :", end=" ")
    for i in test:
        print(i, end = " ")
    print()
    return test
```

- تابع calculate_p احتمال رخداد یک جمله را مشابه مثال زیر پیدا می‌کند:

مثال: احتمال جمله "I want English food" با تخمین bigram

$$P(\text{<s> i want english food </s>}) = P(\text{i}|\text{<s>})P(\text{want}|\text{i})P(\text{english}|\text{want})\,P(\text{food}|\text{english})P(\text{</s>}|\text{food})$$

```
def calculate_p(test, unigrams, bigrams):
    total = 1
    for i in range(len(test)-1):
        temp_p = p(test[i], test[i+1], unigrams, bigrams)
        total = total * temp_p
        print("p("+test[i+1]+"|"+test[i]+") = ", temp_p)
    print("total =",total)
```

نمونه‌ی اجرا شده :

پیکره :

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

خروجی:

```
------------------Unigram values------------------
               <s> ---> 3
               i ---> 3
               am ---> 2
               sam ---> 2
               </s> ---> 3
               do ---> 1
               not ---> 1
               like ---> 1
               green ---> 1
               eggs ---> 1
               and ---> 1
               ham ---> 1
------------------Bigram  values------------------
          ('<s>', 'i')  ---> 2
          ('i', 'am')  ---> 2
          ('am', 'sam')  ---> 1
          ('sam', '</s>')  ---> 1
          ('<s>', 'sam')  ---> 1
          ('sam', 'i')  ---> 1
          ('am', '</s>')  ---> 1
          ('i', 'do')  ---> 1
          ('do', 'not')  ---> 1
          ('not', 'like')  ---> 1
          ('like', 'green')  ---> 1
          ('green', 'eggs')  ---> 1
          ('eggs', 'and')  ---> 1
          ('and', 'ham')  ---> 1
          ('ham', '</s>')  ---> 1
The test sentences is : <s> am i ham and not </s>
p(am|<s>) =  0.07142857142857142
p(i|am) =  0.2
p(ham|i) =  0.07692307692307693
p(and|ham) =  0.15384615384615385
p(not|and) =  0.07692307692307693
p(</s>|not) =  0.06666666666666667
total = 8.669831155038256e-07
```

پیکره:

```
<s> I am Sam </s>
<s> Sam i am </s>
<s> I do not like green eggs and ham</s>
<s> the quick person did not realize his speed and the quick person bumped </s>
<s> i am ghazal prs and ghazal prs is me</s>
```

خروجی:

```
----------Unigram values-------
        <s> ---> 5
        i ---> 4
        am ---> 3
        sam ---> 2
        </s> ---> 5
        do ---> 1
        not ---> 2
        like ---> 1
        green ---> 1
        eggs ---> 1
        and ---> 3
        ham ---> 1
        the ---> 2
        quick ---> 2
        person ---> 2
        did ---> 1
        realize ---> 1
        his ---> 1
        speed ---> 1
        bumped ---> 1
        ghazal ---> 2
        prs ---> 2
        is ---> 1
        me ---> 1
```

```
----------------Bigram  values-----------------
          ('<s>', 'i')  ---> 3
          ('i', 'am')  ---> 3
          ('am', 'sam')  ---> 1
          ('sam', '</s>')  ---> 1
          ('<s>', 'sam')  ---> 1
          ('sam', 'i')  ---> 1
          ('am', '</s>')  ---> 1
          ('i', 'do')  ---> 1
          ('do', 'not')  ---> 1
          ('not', 'like')  ---> 1
          ('like', 'green')  ---> 1
          ('green', 'eggs')  ---> 1
          ('eggs', 'and')  ---> 1
          ('and', 'ham')  ---> 1
          ('ham', '</s>')  ---> 1
          ('<s>', 'the')  ---> 1
          ('the', 'quick')  ---> 2
          ('quick', 'person')  ---> 2
          ('person', 'did')  ---> 1
          ('did', 'not')  ---> 1
          ('not', 'realize')  ---> 1
          ('realize', 'his')  ---> 1
          ('his', 'speed')  ---> 1
          ('speed', 'and')  ---> 1
          ('and', 'the')  ---> 1
          ('person', 'bumped')  ---> 1
          ('bumped', '</s>')  ---> 1
          ('am', 'ghazal')  ---> 1
          ('ghazal', 'prs')  ---> 2
          ('prs', 'and')  ---> 1
          ('and', 'ghazal')  ---> 1
          ('prs', 'is')  ---> 1
          ('is', 'me')  ---> 1
          ('me', '</s>')  ---> 1
The test sentences is : <s> ghazal green the prs quick </s>
p(ghazal|<s>) =  0.038461538461538464
p(green|ghazal) =  0.04
p(the|green) =  0.038461538461538464
p(prs|the) =  0.038461538461538464
p(quick|prs) =  0.038461538461538464
p(</s>|quick) =  0.034482758620689655
total = 3.01834307453255e-09
```