

1402-03-19



Persian Gulf University

Faculty of Intelligent Systems Engineering and Data Science

Natural Language Processing

Dr. Mohammad Bidoki

Homework #4

آرین قره محمد زاده قشقایی

4017724001

محمد برزگر

4010724001

1- پیش پرداز

```
def import_text(name):  
    with codecs.open(name, "r", encoding="cp1252") as f:  
        text = f.read()  
        return text
```

تابع import text برای وارد کردن فایل های تکست ترین و تست

```
def extract_sentence(text):  
    text = text.replace("Mr.", "Mr<period>")  
    text = text.replace("Jr.", "Jr<period>")  
    text = text.replace("Ms.", "Ms<period>")  
    text_tokens = sent_tokenize(text, "english")  
    text_tokens = [sentence.replace("Jr<period>", "Jr.") for sentence in text_tokens]  
    text_tokens = [sentence.replace("Mrs<period>", "Mrs.") for sentence in text_tokens]  
    text_tokens = [sentence.replace("Mr<period>", "Mr.") for sentence in text_tokens]  
    return text_tokens
```

این تابع برای جدا سازی جملات استفاده میشود. مشکلی که در فایل های تکست وجود داشت. این بود که موقع توکنایز کردن جملات زمانی که به کلمات MR. MSS. Jr. می رسید به دلیل داشتن نقطه جمله را توکنایز میکرد برای جلوگیری از این کار ابتدا نقطه بعد این کلمات را حذف و بعد از توکنایز کردن به آن اضافه کردیم.

```
def removing(sentences_token):
    patternEng = r'[a-zA-Z0-9]+'
    RegtokenizerEng = RegexTokenizer(patternEng)
    result = []
    for sentence in sentences_token:
        noPunctuationtokens = RegtokenizerEng.tokenize(sentence)
        sentence = ' '.join(noPunctuationtokens)
        result.append(sentence)
    return result
```

این تابع برای حذف اعداد و حروف نگارشی توکن های حاوی جملات استفاده شده.

2- ساختن مدل زبانی

```
def Myngram(tokens, n):
    tokens = nltk.word_tokenize(tokens)
    n_grams = list(ngrams(tokens, n))
    return n_grams
```

این تابع برای ساخت n گرام ها در نظر گرفته شده.

```
def calculateUniprobability(sentences):
    gram1 = []
    for i in sentences:
        gram1.extend(Myngram(i, 1))
    counter = Counter(gram1)
    gramNumber = sum(counter.values())
    probability = {}
    for key, values in counter.items():
        probability[key] = values / gramNumber
    return probability, counter
```

تابع فوق برای محاسبه احتمال کلمات به صورت یونیگرام در نظر گرفته شده، که ورودی آن جملات هستند و خروجی آن دو دیکشنری. اولی : دیکشنری برای در نظر گرفتن کلمات یونیک و تعداد آن ها و دیکشنری دومی برای نگه داری احتمال آن ها در نظر گرفته شده.

```
def calculateBigramProbability(sentences):  
    gram2 = []  
    for i in sentences:  
        gram2.extend(Myngram(i, 2))  
    counter = Counter(gram2)  
    bigram_probabilities = {}  
    for (prev_word, curr_word), count in counter.items():  
        if prev_word not in bigram_probabilities:  
            bigram_probabilities[prev_word] = {}  
  
        total_count = sum(counter.values())  
        probability = count / total_count  
        bigram_probabilities[prev_word][curr_word] = probability  
  
    return bigram_probabilities
```

این تابع برای محاسبه احتمال bi گرام در نظر گرفته شده. که ورودی آن جملات هست و خروجی آن دیکشنری حاوی کلمات به عنوان کلید و احتمال بی گرام آن به عنوان value که خروجی آن مشابه زیر میشود.

```
'Khmer': {'head': 1.2768929618936833e-06}, 'Musee': {'Guimet': 1.2768929618936833e-06}, 'Guimet': {'in': 1.2768929618936833e-06}, 'fantods': {'imploring': 1.2768929618936833e-06}, 'imploring': {'the': 1.2768929618936833e-06}, 'Sudanese': {'ivory': 1.2768929618936833e-06}, 'juju': {'to': 1.2768929618936833e-06}, 'livers': {'Khartoum': {'assured': 1.2768929618936833e-06}, 'deities': {'though': 1.2768929618936833e-06}, 'Ganessa': {'Siva': 1.2768929618936833e-06}, 'Krishna': {'are': 1.2768929618936833e-06}, 'Travancore': {'a': 1.2768929618936833e-06}, 'subco': 1.2768929618936833e-06}, 'Kali': {'the': 1.2768929618936833e-06}, 'Nuf': {'sed': 1.2768929618936833e-06}, 'amulet': {'age': 1.2768929618936833e-06}, 'housebreakers': {'presented': 1.2768929618936833e-06}, 'mem': {'and': 1.2768929618936833e-06}, 'rajah': {'in': 1.2768929618936833e-06}, 'Balinese': {'a': 1.2768929618936833e-06}, 'Tjokorda': {'Agoeng': 1.2768929618936833e-06}, 'Whosoever': {'violates': 1.2768929618936833e-06}, 'rooftree': {'the': 1.2768929618936833e-06}, 'fishmongers': {'and': 1.2768929618936833e-06}, 'gorging': {'yourselves': 1.2768929618936833e-06}, 'befell': {'another': 1.2768929618936833e-06}, 'wissenheimer': {'who': 1.2768929618936833e-06}, 'garlanded': {'it': 1.2768929618936833e-06}, 'cartons': {'to': 1.2768929618936833e-06}, 'yapping': {'their': 1.2768929618936833e-06}, 'areaways': {'to': 1.2768929618936833e-06}, 'exasperate': {'us': 1.2768929618936833e-06}, 'Spector': {'a': 1.2768929618936833e-06}, 'server': {'we': 1.2768929618936833e-06}, 'Royleott': {'s': 1.2768929618936833e-06}, 'speckled': {'band': 1.2768929618936833e-06}, 'incubi': {'masterly': {'way': 1.2768929618936833e-06}, 'beggary': {'he': 1.2768929618936833e-06}, 'hubris': {'deficiency': 1.2768929618936833e-06}, 'bathos': {'and': 1.2768929618936833e-06}, 'besmirched': {'whole': 1.2768929618936833e-06}, 'sidle': {'up': 1.2768929618936833e-06}, 'hors': {'de': 1.2768929618936833e-06}, 'sprue': {'yaws': 1.2768929618936833e-06}, 'Delhi': 1.2768929618936833e-06}, 'fluke': {'bilharziasis': 1.2768929618936833e-06}, 'bilharziasis': {'and': 1.2768929618936833e-06}, 'Compassionately': {'yours': 1.2768929618936833e-06}, 'Perelman': {'revulsion': 1.2768929618936833e-06}, 'aviary': {'in': 1.2768929618936833e-06}, 'Midwestern': {'lineage': 1.2768929618936833e-06}}
```

که میتوان با قطعه کد زیر یک جمله را جنریت کرد

```
In[8]: generatedsentence = []
      theWord = "I"
      for i in range(20):
          theWord = get_probable_next_word_bigram(BigramProbability, theWord)
          generatedsentence.append(theWord)
      print(generatedsentence)
['was', 'a', 'few', 'years', 'ago', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time']
```

مشکلی که بایگرام داره :

```
In[19]: generatedsentence = []
      theWord = "name"
      for i in range(20):
          theWord = get_probable_next_word_bigram(BigramProbability, theWord)
          generatedsentence.append(theWord)
      print(generatedsentence)
['of', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time']
```

تو کلماتی مثل a , the , and یه پترن خاص تکرار میشه

مشابه این تابع تری گرام نیز ساخته میشود.

```
def calculateThreegramProbability(sentences):
    gram3 = []
    for i in sentences:
        gram3.extend(Myngram(i, 3))
    counter = Counter(gram3)
    threegram_probabilities = {}
    for (prev_prev_word, prev_word, curr_word), count in counter.items():
        if (prev_prev_word, prev_word) not in threegram_probabilities:
            threegram_probabilities[(prev_prev_word, prev_word)] = {}
        total_count = sum(counter[(prev_prev, prev, word)] for (prev_prev, prev, word) in counter if
                           (prev_prev, prev) == (prev_prev_word, prev_word))
        probability = count / total_count
        threegram_probabilities[(prev_prev_word, prev_word)][curr_word] = probability

    return threegram_probabilities
```

ولی این تابع به دلیل اینکه تایم زیادی برای پردازش نیاز داشت نتوانستیم خروجی آن را پیدا کنیم.
ولی با یک corpus کوچک تر که امتحان کردیم کاملاً تابع درست عمل میکرد.

پس برای رفع این مشکل از تابع زیر آمديم استفاده کردیم که به کمک bigram ساخته شده

```
def calculateThreegramProbability(sentences):
    gram3 = []
    gram2 = []
    for i in sentences:
        gram3.extend(Myngram(i, 3))
        gram2.extend(Myngram(i, 2))
    counter3 = Counter(gram3)
    counter2 = Counter(gram2)
    threegram_probabilities = {}
    for (prev_prev_word, prev_word, curr_word), count in counter3.items():
        if (prev_prev_word, prev_word) not in threegram_probabilities:
            threegram_probabilities[(prev_prev_word, prev_word)] = {}
        bigram_count = counter2[(prev_prev_word, prev_word)]
        probability = count / bigram_count
        threegram_probabilities[(prev_prev_word, prev_word)][curr_word] = probability
    return threegram_probabilities
```

جنریت جمله :

بایگرام:

```
def generate_unigram_sentence(range,theWord):
    generatedsentence = [theWord]
    for i in range(range):
        theWord = get_probable_next_word_bigram(BigramProbability, theWord)
        generatedsentence.append(theWord)
```

تریگرام:

```
def generate_trigram_sentence(range,theWord,secWord,TrigramProbability):
    generatedsentence = [theWord,secWord]
    theWord = "I"
    secWord = "was"
    for i in range(20):
        generated = get_probable_next_word_trigram(TrigramProbability, theWord, secWord)
        generatedsentence.append(generated)
        theWord = secWord
        secWord = generated
    return generatedsentence
```

```
In[25]: theWord="I"
        secWord="was"
        for i in range(20):
            generated = get_probable_next_word_trigram(TrigramProbability,theWord,secWord)
            generatedsentence.append(generated)
            theWord=secWord
            secWord=generated
In[26]: print(generatedsentence)
['a', 'good', 'deal', 'of', 'time', 'and', 'the', 'other', 'hand', 'the', 'other', 'hand', 'the', 'other', 'hand', 'the', 'other', 'hand', 'the', 'other']
```

که کاملاً بهتر از مدل bigram عمل می‌کند ولی همچنان ممکن است وارد لوپ کلمات شود.

همین‌طور این احتمالات با تکنیک‌های laplace smoothing و good-turing smoothing نیز انجام شده.

این متدها کمک می‌کند که داده‌های آنسین ما که در مدل‌ترین شده ما قرار ندارند مقدار 0 را نگیرند و به احتمالی نیز برای آن‌ها در نظر گرفته بشه.

برای یونیگرام:

```
def calculateLaplaceSmoothing(sentences):
    gram3 = []
    for i in sentences:
        gram3.extend(Myngram(i, 1))
    counter = Counter(gram3)
    Unique_V = len(counter)
    Total_numberOfWords = sum(counter.values())
    LS = {}
    for key, values in counter.items():
        LS[key] = (values + 1) / (Total_numberOfWords + Unique_V)
    return LS, Total_numberOfWords
```

برای بیگرام:

```
def calculategood_smoothingBigramProbability(sentences):
    bigram_counts = Counter()
    unigram_counts = Counter()
    for sentence in sentences:
        words = sentence.split()
        bigrams = [(words[i], words[i + 1]) for i in range(len(words) - 1)]
        bigram_counts.update(bigrams)
        unigram_counts.update(words)
    observed_counts = Counter(bigram_counts.values())
    total_bigrams = sum(bigram_counts.values())
    total_unigrams = sum(unigram_counts.values())
    observed_frequencies = np.array([observed_counts[i] for i in range(1, max(observed_counts) + 1)])

    r_star = (np.arange(1, max(observed_counts)) + 1) * (observed_frequencies[1:] / observed_frequencies[:-1])
    c_star = r_star - observed_counts[1:]
    c_star[0] = 1 # Adjust for unseen bigrams
    bigram_probabilities = {}
    for bigram, count in bigram_counts.items():
        prev_word = bigram[0]
        probability = (count + c_star[count]) / (total_unigrams + c_star[1])
        if prev_word not in bigram_probabilities:
            bigram_probabilities[prev_word] = {}
        bigram_probabilities[prev_word][bigram[1]] = probability
    return bigram_probabilities
```


اینگونه کلماتی که تعداد 0 دارند با اضافه شدن 1 به آنها مقدار نهایی احتمال آن یک عدد غیر از صفر میشود.

در واقع هموار سازی good-turning بر اساس frequencies کلمات مشاهده شده n گرام های دیگر یک احتمال غیر از صفر به داده های دیده نشده میدهد.

در مقایسه با روش لاپلاس که یک توزیع یکنواختی را برای داده های دیده نشده در نظر میگیرد این روش روش بهتری هست چون ممکن هست تعداد کلمات دیده نشده مقدار خیلی زیادی باشند که احتمال مشابهی میگیرند اینگونه با این روش (good turing) مقادیر مختلفی به این کلمات نسبت داده میشود.

توابع زیر نیز برای بدست آوردن perplexity ساخته شده اند.

```
def calculate_perplexity(language_model, test_dataset, totalWords):
    total_log_prob = 0.0
    word_count = 0
    Unique_V = len(language_model)
    for token in test_dataset:
        if token in language_model:
            word_prob = language_model[token]
        else:
            word_prob = 1 / (Unique_V + totalWords)

        total_log_prob += math.log(word_prob)
        word_count += 1

    avg_log_prob = total_log_prob / word_count
    perplexity = math.exp(-avg_log_prob)
    return perplexity
```

```
def calculate_perplexity_bigram(sentences, bigram_probabilities):
    total_log_probability = 0.0
    total_words = 0
    for sentence in sentences:
        words = sentence.split()
        total_words += len(words)
        for i in range(1, len(words)):
            prev_word = words[i-1]
            curr_word = words[i]
            # Check if bigram exists in the probabilities dictionary
            if prev_word in bigram_probabilities and curr_word in bigram_probabilities[prev_word]:
                bigram_probability = bigram_probabilities[prev_word][curr_word]
            else:
                # Handle unseen bigrams by assigning a very small probability
                bigram_probability = 1e-10
            total_log_probability += math.log2(bigram_probability)
    average_log_probability = total_log_probability / total_words
    perplexity = 2 ** (-average_log_probability)
    return perplexity
```

استفاده از توابع فوق برای یونیگرام:

```
357 ▶ if __name__ == '__main__':
358     # Preprocessing
359     text = import_text("brown.train.txt")
360     test = import_text("brown.test.txt")
361
362     sentence = extract_sentence(text)
363     sentenceTest = extract_sentence(test)
364     result = removing(sentence)
365     resultTest = removing(sentenceTest)
366     testTokens = removepunctuation(test)
367
368     # unigram
369
370     Uniprobability, UniqueWords = calculateUniprobability(result)
371     laplaceSmoothing, Total_numberOfWords = calculateLaplaceSmoothing(result)
372     perplexity = calculate_perplexity(laplaceSmoothing, testTokens, Total_numberOfWords)
373     theWord = get_probable_next_word(laplaceSmoothing, "if",)
374
375     unigram_goodturing, uni_unseen_gt = calculate_good_turing_unigram_probability(result)
```

استفاده از توابع فوق برای بایگرام:

```
# bigram
corpus = "I am learning language proccessing and I am happy."
corpus = extract_sentence(corpus)
corpus = removing(corpus)
BigramProbability = calculateBigramProbability(result)
print(BigramProbability)
theWord = get_probable_next_word_bigram(BigramProbability,"I")
print(theWord)

bigram_laplace, bigram_Total_numberOfWords = calculateLaplaceSmoothing(result)
bigram_perplexity = calculate_perplexity_bigram(result, BigramProbability)
bigram_goodturing, unseen_p = calculategood_smoothingBigramProbability(result)
```

تولید کردن جمله ها:

```
# generate_sentence
sentence_uni_one = generate_uni_sentence(20, 'I', 'was', Uniprobability)
sentence_uni_two = generate_uni_sentence(20, 'Jury', 'said', Uniprobability)
sentence_uni_three = generate_uni_sentence(20, 'The', 'jury', Uniprobability)
sentence_uni_four = generate_uni_sentence(20, 'These', 'actions', Uniprobability)
sentence_uni_five = generate_uni_sentence(20, 'Four', 'additional', Uniprobability)
sentence_uni = [sentence_uni_one, sentence_uni_two, sentence_uni_three, sentence_uni_four, sentence_uni_five]

sentence_bi_one = generate_bigram_sentence(20, 'I', 'was', BigramProbability)
sentence_bi_two = generate_bigram_sentence(20, 'Jury', 'said', BigramProbability)
sentence_bi_three = generate_bigram_sentence(20, 'The', 'jury', BigramProbability)
sentence_bi_four = generate_bigram_sentence(20, 'These', 'actions', BigramProbability)
sentence_bi_five = generate_bigram_sentence(20, 'Four', 'additional', BigramProbability)
sentence_bi = [sentence_bi_one, sentence_bi_two, sentence_bi_three, sentence_bi_four, sentence_bi_five]
```

محاسبه perplexity جملات و مدل ها:

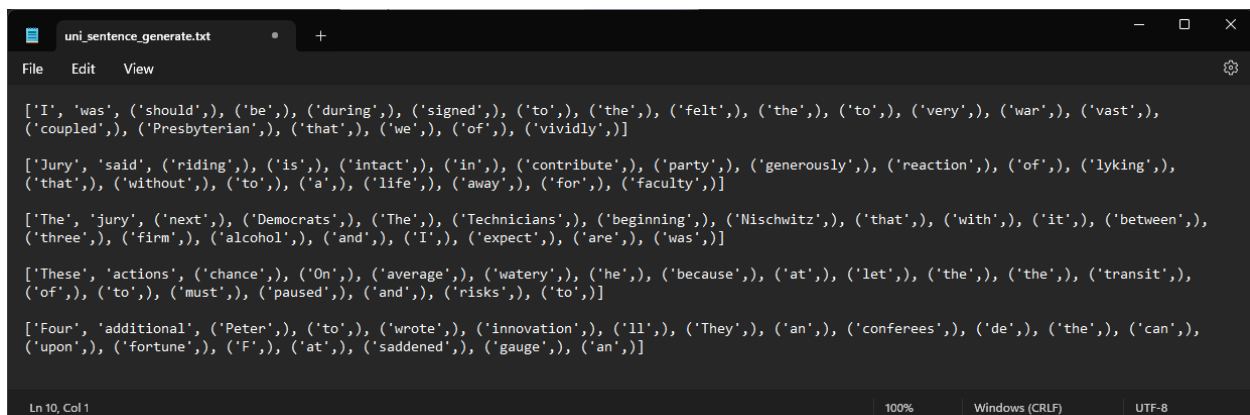
```
uni_pp_mle = []
uni_pp_laplace = []
uni_pp_good = []
bi_pp_mle = []
bi_pp_laplace = []
bi_pp_good = []

# perplexity of sentences
for i in range(len(sentence_uni)):
    uni_pp_mle.append(calculate_perplexity(Uniprobability, sentence_uni[i]))
    uni_pp_laplace.append(calculate_perplexity(laplaceSmoothing, sentence_uni[i]))
    uni_pp_good.append(calculate_perplexity(unigram_goodturing, sentence_uni[i]))

for i in range(len(sentence_bi)):
    bi_pp_mle.append(calculate_perplexity_bigram(sentence_bi[i], BigramProbability))
    bi_pp_laplace.append(calculate_perplexity_bigram(sentence_bi[i], bigram_laplace))
    bi_pp_good.append(calculate_perplexity_bigram(sentence_bi[i], bigram_goodturing))

# perplexity of models
uni_perplexity_mle = calculate_perplexity(Uniprobability, test)
uni_perplexity_laplace = calculate_perplexity(laplaceSmoothing, test)
uni_perplexity_good = calculate_perplexity(unigram_goodturing, test)
bi_perplexity_mle = calculate_perplexity_bigram(test, BigramProbability)
bi_perplexity_laplace = calculate_perplexity_bigram(test, bigram_laplace)
bi_perplexity_good = calculate_perplexity_bigram(test, bigram_goodturing)
```

جملات ساخته شده با یونیگرام:



```
uni_sentence_generate.txt
File Edit View
[ 'I', 'was', ('should'), ('be'), ('during'), ('signed'), ('to'), ('the'), ('felt'), ('the'), ('to'), ('very'), ('war'), ('vast'),
('coupled'), ('Presbyterian'), ('that'), ('we'), ('of'), ('vividly'), ]

[ 'Jury', 'said', ('riding'), ('is'), ('intact'), ('in'), ('contribute'), ('party'), ('generously'), ('reaction'), ('of'), ('lying'),
('that'), ('without'), ('to'), ('a'), ('life'), ('away'), ('for'), ('faculty'), ]

[ 'The', 'jury', ('next'), ('Democrats'), ('The'), ('Technicians'), ('beginning'), ('Nischwitz'), ('that'), ('with'), ('it'), ('between'),
('three'), ('firm'), ('alcohol'), ('and'), ('I'), ('expect'), ('are'), ('was'), ]

[ 'These', 'actions', ('chance'), ('On'), ('average'), ('watery'), ('he'), ('because'), ('at'), ('let'), ('the'), ('the'), ('transit'),
('of'), ('to'), ('must'), ('paused'), ('and'), ('risks'), ('to'), ]

[ 'Four', 'additional', ('Peter'), ('to'), ('wrote'), ('innovation'), ('ll'), ('They'), ('an'), ('conferees'), ('de'), ('the'), ('can'),
('upon'), ('fortune'), ('F'), ('at'), ('saddened'), ('gauge'), ('an'), ]

Ln 10, Col 1 100% Windows (CRLF) UTF-8
```

جملات ساخته شده با بایگرام:

```
bi_sentence_generate.txt
File Edit View
[ 'I', 'was', 'a', 'few', 'years', 'ago', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to' ]
[ 'Jury', 'said', 'that', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time' ]
[ 'The', 'jury', 'said', 'that', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time' ]
[ 'These', 'actions', 'of', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time' ]
[ 'Four', 'additional', 'information', 'on', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time', 'to', 'the', 'same', 'time' ]

Ln 5, Col 105 100% Windows (CRLF) UTF-8
```

مقدار perplexity جملات یونیگرام به ترتیب به شکل زیر است:

لیست اول مربوط به گود تورینگ، دوم لاپلاس و سوم mle است:

```
uni_pp_good = (list: 5) [5358867312.681481, 3196400489.0384645, 2719791234.108872, 1316981795.9787066, 1423953170.3285275]
  0 = {float} 5358867312.681481
  1 = {float} 3196400489.0384645
  2 = {float} 2719791234.108872
  3 = {float} 1316981795.9787066
  4 = {float} 1423953170.3285275
  __len__ = {int} 5
  Protected Attributes
uni_pp_laplace = (list: 5) [4814.581703501139, 7848.191789519194, 8191.896729437103, 4837.397164015461, 16195.436251089333]
  0 = {float} 4814.581703501139
  1 = {float} 7848.191789519194
  2 = {float} 8191.896729437103
  3 = {float} 4837.397164015461
  4 = {float} 16195.436251089333
  __len__ = {int} 5
  Protected Attributes
uni_pp_mle = (list: 5) [4683.591957049676, 7763.511945878278, 8208.276232029604, 4770.316750232196, 16643.90738986477]
  0 = {float} 4683.591957049676
  1 = {float} 7763.511945878278
  2 = {float} 8208.276232029604
  3 = {float} 4770.316750232196
  4 = {float} 16643.90738986477
  __len__ = {int} 5
```

احتمال بایگرام گود تورینگ، لاپلاس و mle بایگرام ها:

```
bi_perplexity_good = {float} 1.0
bi_perplexity_laplace = {float} 1.0
bi_perplexity_mle = {float} 1.0
```

احتمال mle بایگرام در فایل bigram_probabilities.txt هست و تعداد آن 42321 است:

```
bigram_probabilities.txt
File Edit View
42321
{'The': {'Fulton': 1.2766533298948803e-06, 'September': 1.2766533298948803e-06, 'jury': 8.936573309264163e-06, 'grand': 1.2766533298948803e-06,
'City': 3.829959989684641e-06, 'jurors': 1.2766533298948803e-06, 'couple': 2.5533066597897607e-06, 'petition': 2.5533066597897607e-06, 'Hartsfield':
1.2766533298948803e-06, 'mayor': 3.829959989684641e-06, 'largest': 1.2766533298948803e-06, 'Republicans': 1.2766533298948803e-06, 'Georgia':
2.5533066597897607e-06, 'bond': 2.5533066597897607e-06, 'department': 5.106613319579521e-06, 'Highway': 1.2766533298948803e-06, 'Constitution':
2.5533066597897607e-06, 'resolution': 7.659919979369282e-06, 'new': 3.957625322674129e-05, 'campaign': 1.2766533298948803e-06, 'former':
1.0213226639159043e-05, 'bill': 5.106613319579521e-06, 'escheat': 1.2766533298948803e-06, 'TEA': 1.2766533298948803e-06, 'senate':
1.2766533298948803e-06, 'other': 3.063967991747713e-05, 'West': 1.0213226639159043e-05, 'remainder': 1.2766533298948803e-06, 'board':
8.936573309264163e-06, 'report': 6.383266649474402e-06, 'volume': 3.829959989684641e-06, 'statements': 1.2766533298948803e-06, 'last':
2.9363026587582247e-05, 'cases': 2.5533066597897607e-06, 'case': 3.829959989684641e-06, 'aged': 2.5533066597897607e-06, 'social':
3.829959989684641e-06, 'payroll': 1.2766533298948803e-06, 'President': 3.063967991747713e-05, 'plan': 8.936573309264163e-06, 'scholarship':
1.2766533298948803e-06, 'rule': 2.5533066597897607e-06, 'most': 4.4682866546320814e-05, 'nightmare': 1.2766533298948803e-06, 'general':
8.936573309264163e-06, 'impression': 2.5533066597897607e-06, 'Secretary': 1.6596493288633443e-05, 'secretary': 5.106613319579521e-06, 'inclination':
1.2766533298948803e-06, 'administration': 2.5533066597897607e-06, 'White': 6.383266649474402e-06, 'Narragansett': 1.2766533298948803e-06, 'governor':
2.5533066597897607e-06, 'attorney': 1.2766533298948803e-06, 'statutes': 1.2766533298948803e-06, 'Central': 2.5533066597897607e-06, 'small':
1.0213226639159043e-05, 'council': 3.829959989684641e-06, 'lawyer': 2.5533066597897607e-06, 'law': 5.106613319579521e-06, 'controversial':
Ln 1, Col 1
100% Windows (CRLF) UTF-8
```

احتمال لاپلاس بایگرام در فایل bigram_laplace.txt هست و تعداد آن 44230:

```
bigram_laplace.txt
File Edit View
44230
({'The': 0.006683662210125604, ('Fulton': 2.0688894201592586e-05, ('County': 8.045681078397116e-05, ('Grand': 1.6091362156794234e-05,
('Jury': 4.5975320447983525e-06, ('said': 0.0018148757746841496, ('Friday': 5.8618533571178994e-05, ('an': 0.003240110708571639,
('investigation': 4.482593743678393e-05, ('of': 0.03322636408775769, ('Atlanta': 3.563087334718723e-05, ('s': 0.005451523622119646,
('recent': 0.0001528679404895452, ('primary': 9.080125788476746e-05, ('election': 7.011236368317487e-05, ('produced': 8.62037258399691e-05,
('no': 0.001651663387093808, ('evidence': 0.0001827518987807345, ('that': 0.009359425860198245, ('any': 0.0012171966088603639,
('irregularities': 8.045681078397117e-06, ('took': 0.00039079022380785996, ('place': 0.0005114754399838167, ('September':
5.74691505599794e-05, ('October': 5.0572852492781875e-05, ('term': 0.0001149015208636004, ('jury': 5.402100152638064e-05, ('had':
0.0046618974934255295, ('been': 0.002271180830130386, ('charged': 5.631976754877982e-05, ('by': 0.00471591849495191, ('Superior':
1.4941979145594645e-05, ('Court': 0.00010804200305276128, ('Judge': 3.9079022380785996e-05, ('Durwood': 2.2987660223991762e-06, ('Pye':
2.2987660223991762e-06, ('to': 0.023677290030711513, ('investigate': 8.045681078397117e-06, ('reports': 7.241112970557405e-05, ('possible':
0.0003551593504606727, ('in': 0.018039566360777536, ('the': 0.05760362837228975, ('hard': 0.0001942457288927304, ('fought':
4.597532044798352e-05, ('which': 0.003228616878459643, ('was': 0.008985876381558379, ('won': 0.00015056917446714603, ('Mayor':
3.218272431358847e-05, ('nominate': 5.74691505599794e-06, ('Ivan': 5.74691505599794e-06, ('Allen': 1.839012817919341e-05, ('In':
6.896298067197529e-05, ('it': 0.006326204093642532, ('did': 0.0009195064089596704, ('find': 0.00037469886165106573, ('many':
0.0008447965132316972, ('Georgia': 5.0572852492781875e-05, ('registration': 1.839012817919341e-05, ('and': 0.025630091766739613, ('laws':
Ln 1, Col 1
100% Windows (CRLF) UTF-8
```

احتمال گود تورینگ بایگرام به شکل زیر است و در فایل bigram_goodturing.txt است:

```
bigram_goodturing.txt
File Edit View
4.6140659651359524e-07 unseen probability
({'The', 'Fulton'): 2.2066082316508243e-05, ('Fulton', 'County'): 8.697216648924122e-05, ('County', 'Grand'): 1.695795605384328e-05, ('Grand', 'Jury'): 5.272956678184278e-06, ('Jury', 'said'): 0.001568551308338325, ('said', 'Friday'): 5.492593088485756e-05, ('Friday', 'an'): 0.003599031497747647, ('an', 'investigation'): 4.4830124822255304e-05, ('investigation', 'of'): 0.03691064363223387, ('of', 'Atlanta'): 3.4424135960032486e-05, ('Atlanta', 's'): 0.005993122711232874, ('s', 'recent'): 0.00016892471236812587, ('recent', 'primary'): 9.613391312515018e-05, ('primary', 'election'): 6.65704718301638e-05, ('election', 'produced'): 9.230281842815146e-05, ('produced', 'no'): 0.001807164495833249, ('no', 'evidence'): 0.0001971775159557713, ('evidence', 'that'): 0.010283161239785584, ('that', 'any'): 0.0013728089330912081, ('any', 'irregularities'): 8.018735094179602e-06, ('irregularities', 'took'): 0.00043199321489537134, ('took', 'place'): 0.0005157229025682832, ('The', 'September'): 6.0377029286495444e-05, ('September', 'October'): 4.888374519549929e-05, ('October', 'term'): 0.0001191204813071425, ('term', 'jury'): 5.2714839892498e-05, ('jury', 'had'): 0.005160841102557458, ('had', 'been'): 0.0024965967108774995, ('been', 'charged'): 5.9099997720829204e-05, ('charged', 'by'): 0.005217964550873467, ('by', 'Fulton'): 2.2066082316508243e-05, ('Fulton', 'Superior'): 1.676624076918044e-05, ('Superior', 'Court'): 0.00011990575206567366, ('Court', 'Judge'): 4.1221555801501845e-05, ('Judge', 'Durwood'): 1.6335772658484012e-06, ('Durwood', 'Pye'): 1.6335772658484012e-06, ('Pye', 'to'): 0.02625357147266675, ('to', 'investigate'): 1.162730411493306e-05, ('investigate', 'reports'): 7.187031337749161e-05, ('reports', 'of'): 0.03690480000108172, ('of', 'possible'): 0.0003397039700284291, ('possible', 'irregularities'): 8.018735094179602e-06, ('irregularities', 'in'): 0.019946120854556524, ('in', 'the'): 0.06399588285023229, ('the', 'hard'): 0.00021763800257934264, ('hard', 'fought'): 4.869202991083644e-05, ('fought', 'primary'): 9.613391312515018e-05, ('primary', 'which'): 0.003583707118959652, ('which', 'was'):
Ln 1, Col 42
100% Windows (CRLF) UTF-8
```

احتمال mle یونیگرام به شکل زیر است و در فایل unigram_probabilities.txt است:

تعداد 44230

```
unigram_probabilities.txt
File Edit View
44230
({'The',): 0.0070404285773103, ('Fulton',): 2.0586048471667543e-05, ('County',): 8.355513791441532e-05, ('Grand',): 1.5742272360686944e-05, ('Jury',): 3.6328320832354487e-06, ('said',): 0.001910869675781846, ('Friday',): 6.054720138725748e-05, ('an',): 0.0034124402701858313, ('investigation',): 4.601587305431568e-05, ('of',): 0.035004759010029037, ('Atlanta',): 3.632832083235449e-05, ('s',): 0.005742296579567499, ('recent',): 0.00015984461166235976, ('primary',): 9.445363416412167e-05, ('election',): 7.265664166470897e-05, ('produced',): 8.960985805314107e-05, ('no',): 0.0017389156238420348, ('evidence',): 0.00019132915638373364, ('that',): 0.009859506273901008, ('any',): 0.0012811787813543681, ('irregularities',): 7.265664166470897e-06, ('took',): 0.0004105100254056057, ('place',): 0.0005376591483188464, ('September',): 5.933625735951233e-05, ('October',): 5.2070593193041434e-05, ('term',): 0.00011625062666353436, ('jury',): 5.570342527627688e-05, ('had',): 0.004910378032506582, ('been',): 0.00239161445479667, ('charged',): 5.812531333176718e-05, ('by',): 0.004967292401810604, ('Superior',): 1.4531328332941795e-05, ('Court',): 0.0001126177945802989, ('Judge',): 3.996115291558994e-05, ('Durwood',): 1.2109440277451496e-06, ('Pye',): 1.2109440277451496e-06, ('to',): 0.024944236027522337, ('investigate',): 7.265664166470897e-06, ('reports',): 7.507852972019928e-05, ('possible',): 0.00037297076054550605, ('in',): 0.019004555571432376, ('the',): 0.060687670894475916, ('hard',): 0.00020343859666118513, ('fought',): 4.7226817082060836e-05, ('which',): 0.00340033082990838, ('was',): 0.009465949464883834, ('won',): 0.00015742272360686944, ('Mayor',): 3.269548874911904e-05, ('nominate',): 4.843776110980598e-06, ('Ivan',): 4.843776110980598e-06, ('Allen',): 1.8164160416177243e-05, ('Jr',): 7.144569763696383e-05, ('it',): 0.006663824984681558, ('did',): 0.0009675442781683745, ('find',): 0.0003935568090171736, ('many',): 0.0008888329163649398, ('Georgia',): 5.2070593193041434e-05, ('registration',): 1.8164160416177243e-05, ('and',): 0.027001629930661346, ('laws',): 9.082080208088622e-05, ('are',):
Ln 1, Col 1
100% Windows (CRLF) UTF-8
```


احتمال لاپلاس یونیگرام به شکل زیر است و تعداد آن 44230 و در فایل unigram_laplace

```
unigram_leplace.txt
File Edit View

44230
{('The',): 0.006683662210125604, ('Fulton',): 2.0688894201592586e-05, ('County',): 8.045681078397116e-05, ('Grand',): 1.6091362156794234e-05, ('Jury',): 4.597532044798352e-06, ('said',): 0.0018148757746841496, ('Friday',): 5.861853571178994e-05, ('an',): 0.003240110708571639, ('investigation',): 4.482593743678393e-05, ('of',): 0.03322636408775769, ('Atlanta',): 3.563087334718723e-05, ('s',): 0.005451523622119646, ('recent',): 0.0001528679404895452, ('primary',): 9.080125788476746e-05, ('election',): 7.011236368317487e-05, ('produced',): 8.62037258399691e-05, ('no',): 0.00165166387093808, ('evidence',): 0.0001827518987807345, ('that',): 0.009359425860198245, ('any',): 0.0012171966088603639, ('irregularities',): 8.045681078397117e-06, ('took',): 0.00039079022380785996, ('place',): 0.0005114754399838167, ('September',): 5.74691505599794e-05, ('October',): 5.0572852492781875e-05, ('term',): 0.0001149015208636004, ('jury',): 5.402100152638064e-05, ('had',): 0.004618974304525529, ('been',): 0.002271180830138036, ('charged',): 5.631976754877982e-05, ('by',): 0.00471591849495191, ('Superior',): 1.4941979145594645e-05, ('Court',): 0.00018084200305276128, ('Judge',): 3.9079022380785996e-05, ('Durwood',): 2.2987660223991762e-06, ('Pye',): 2.2987660223991762e-06, ('to',): 0.023677290038711513, ('investigate',): 8.045681078397117e-06, ('reports',): 7.241112970557405e-05, ('possible',): 0.0003551593504060727, ('in',): 0.018039566360777536, ('the',): 0.05760362837228975, ('hard',): 0.0001942457288927304, ('fought',): 4.597532044798352e-05, ('which',): 0.003228616878459643, ('was',): 0.00895876381558379, ('won',): 0.00015056917446714603, ('Mayor',): 3.218272431358847e-05, ('nominate',): 5.74691505599794e-06, ('Ivan',): 5.74691505599794e-06, ('Allen',): 1.839012817919341e-05, ('Jr',): 6.896298067197529e-05, ('it',): 0.006326204093642532, ('did',): 0.0009195064089596704, ('find',): 0.00037469886165106573, ('many',): 0.0008447965132316972, ('Georgia',): 5.0572852492781875e-05, ('registration',): 1.839012817919341e-05, ('and',): 0.025630091766739613, ('laws',):
```

گودتورینگ یونیگرام در فایل unigram_goodturing.txt:

```
unigram_goodturing.txt
File Edit View

0.02231185726225517 unseen probability
{'The': 0.0, 'Fulton': 1.764574368032934e-05, 'County': 6.26707976728595e-05, 'Grand': 1.734338919556406e-05, 'Jury': 3.077610425999784e-06, 'said': 0.0, 'Friday': 7.487939462211783e-05, 'an': 0.0, 'investigation': 3.619695105577587e-05, 'of': 0.0, 'Atlanta': 3.57951495907782e-05, 's': 0.0, 'recent': 0.00024165122279152585, 'primary': 8.737002894969536e-05, 'election': 6.46523008345749e-05, 'produced': 7.019943245823546e-05, 'no': 0.0, 'evidence': 0.0001005383293620511, 'that': 0.0, 'any': 0.0, 'irregularities': 6.496443194900783e-06, 'took': 0.0002745577802810987, 'place': 0.0, 'September': 3.568962742624576e-05, 'October': 4.342678616210843e-05, 'term': 0.00012924403745291133, 'jury': 7.775854496418203e-05, 'had': 0.0, 'been': 0.0, 'charged': 7.386142638150342e-05, 'by': 0.0, 'Superior': 1.3519114805232865e-05, 'Court': 0.000132837514283061, 'Judge': 3.9619734019640462e-05, 'Dunwood': 8.8472417142858397e-07, 'Pyre': 8.847241715058397e-07, 'to': 0.0, 'investigate': 6.496443194900783e-06, 'reports': 8.977757001155134e-05, 'possible': 0.0003742068563537919, 'in': 0.0, 'the': 1.2112843247695532e-06, 'hard': 6.823568362868483e-05, 'fought': 4.434532442324127e-05, 'which': 0.0, 'was': 0.0, '0.0003713564930896229, 'Mayor': 3.3263731072517726e-05, 'nominate': 4.37537143280956e-06, 'Ivan': 4.37537143280956e-06, 'Allen': 1.9310329815166788e-05, 'Jr': 8.30594965556265e-05, 'it': 0.0, 'did': 0.0, 'find': 0.0002632524599165829, 'many': 0.0, 'Georgia': 4.342678616210843e-05, 'registration': 1.9310329815166788e-05, 'and': 0.0, 'laws': 7.039699487484227e-05, 'are': 0.0, 'outmoded': 4.37537143280956e-06, 'or': 0.0, 'inadequate': 3.3263731072517726e-05, 'often': 0.0, 'ambiguous': 2.3556468636402358e-05, 'it': 0.0, 'recommended': 5.4661337698051384e-05, 'legislators': 1.3929769734849861e-05, 'act': 0.00024346814927868018, 'have': 0.0, 'these': 0.0, 'studied': 0.00010060389252947123, 'revised': 1.221212884808648e-05, 'end': 0.0003960899741996439, 'modernizing': 3.0776710425999784e-06, 'improving': 1.3929769734849861e-05, 'them': 0.0, 'grand': 2.6366561115913995e-05, 'commented': 1.9310329815166788e-05, 'on': 0.0, 'a': 0.0, 'number': 1.3929769734849861e-05}

Ln 1, Col 1
100% Windows (CRLF) UTF-8
```

مقدار perplexity برای مدل های بایگرم به شکل زیر است:

```
01 bi_perplexity_good = {float} 1.0
01 bi_perplexity_laplace = {float} 1.0
01 bi_perplexity_mle = {float} 1.0
```

مقدار perplexity برای مدل های یونیگرام به شکل زیر است:

```
01 uni_perplexity_good = {float} 7835780.118023064
01 uni_perplexity_laplace = {float} 4999999998.992869
01 uni_perplexity_mle = {float} 4999999998.992869
```

مقدار perplexity برای جملات در بالا گزارش شده.

