CSE 484 NLP
HOMEWORK 2

Statistical Language Model

Hüsnü AKÇAK

161044112

1 Homework structure

- Source codes: lang_model.py and syllabicate.py
- The Turkish Wikipedia dump is downloaded from **https://www.kaggle.com/mustfkeskin/turkishwikipedia-dump**
- Each work in from the text is divided into syllables
- N-Grams are generated with **ngrams from nltk** and stored in dictionaries.
- GT smoothing is applied on generated n-grams.
- Perplexities are calculated for 1-Gram, 2-Gram, 3-Gram. Note: Chain rule and Markov assumptions may not be applied properly.
- Random sentences are produced by using all generated n-grams

2 Generating N-Grams

The function for this taks is "def gt_smooth(filename)"

- Syllabicated data is stored as syllable array
- Prepared array is passed to ngram function of nltk library and obtained an N-Gram sequence.
- From that N-Gram sequence a dictionary is prepared and stored in binary files in the following format:
  (Bigram: Number of Occurrence for Bigram)

```
('bal', 'dushk'): 5, ('dushk', 'bal'): 2, ('dushk', 'ar'): 2, ('dushk', 'i'): 3,
```

  Name format for these files is 'NgramsDict.pkl', for example unigram dictionary is 1gramsDict.pkl.

3 Generating N-Grams

NgramsDict.pkl files are load back and GT-smoothing applied on them, then saved as binary files with "_GT_smoothed.pkl" suffix

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

GT p^*_{GT}	$p^*_{GT}(\text{unseen}) = \frac{N_1}{N}$	$p^*_{GT}(\ N_c\ ) = \frac{c^*}{N}$

- From loaded binary N-Gram dictionary files, C values are calculated and stored with new files.

4 Perplexity Calculation

There are different functions for calculating Perplexities. As in the next picture, 1-Gram, 2-Gram, 3-Gram perplexities are calculated.

```
# ASSUMPTION: Sentences are consist of 25 syllables
def unigram_perplexity(ngram_filename, N, testData):...


# ASSUMPTION: Sentences are consist of 25 syllables
def bigram_perplexity(ngram_filename, N, testData):...


# ASSUMPTION: Sentences are consist of 25 syllables
def threegram_perplexity(ngram_filename, N, testData):...
```

Since punctuations are filtered out, sentence boundaries are lost and I picked 25 as sentence length.

$$PP(s) = 2^{log_2^{PP(s)}} = 2^{-\frac{1}{n}log(p(s))}$$

let $l = \frac{1}{n}log(p(s))$

For unigram $l = \frac{1}{n}(logp(w_1) + \cdots + logp(w_n))$

For bigram $l = \frac{1}{n}(logp(w_1) + logp(w_2|w_1) + \cdots + logp(w_n|w_{n-1}))$

The above formula is referenced to calculate perplexities. But for each N-gram its probabilities are concerned. The core of the function is in the next picture.

```
    # loop through each sentence
    i = 0
    while i < len(syllables) - sentence_len-2:
        curr_p = 0
        for j in range(sentence_len):
            # calc perplexity of sentence
            curr_p = curr_p + math.log(ngram_dict.get((syllables[i + j], syllables[i+j+1], syllables[i+j+2], )) or 1, math.e)
            print(syllables[i+j], ',', syllables[i+j+1], ',', syllables[i+j+2], ' - ',
                  ngram_dict.get((syllables[i+j], syllables[i+j+1], syllables[i+j+2], )))

        curr_p = (-1 / N) * curr_p
        for x in range(i, i + sentence_len):
            print(syllables[x], end=' ')
        print('')
        print('e^curr_p = ', math.exp(curr_p))
        i = i + sentence_len
```

N-gram prob

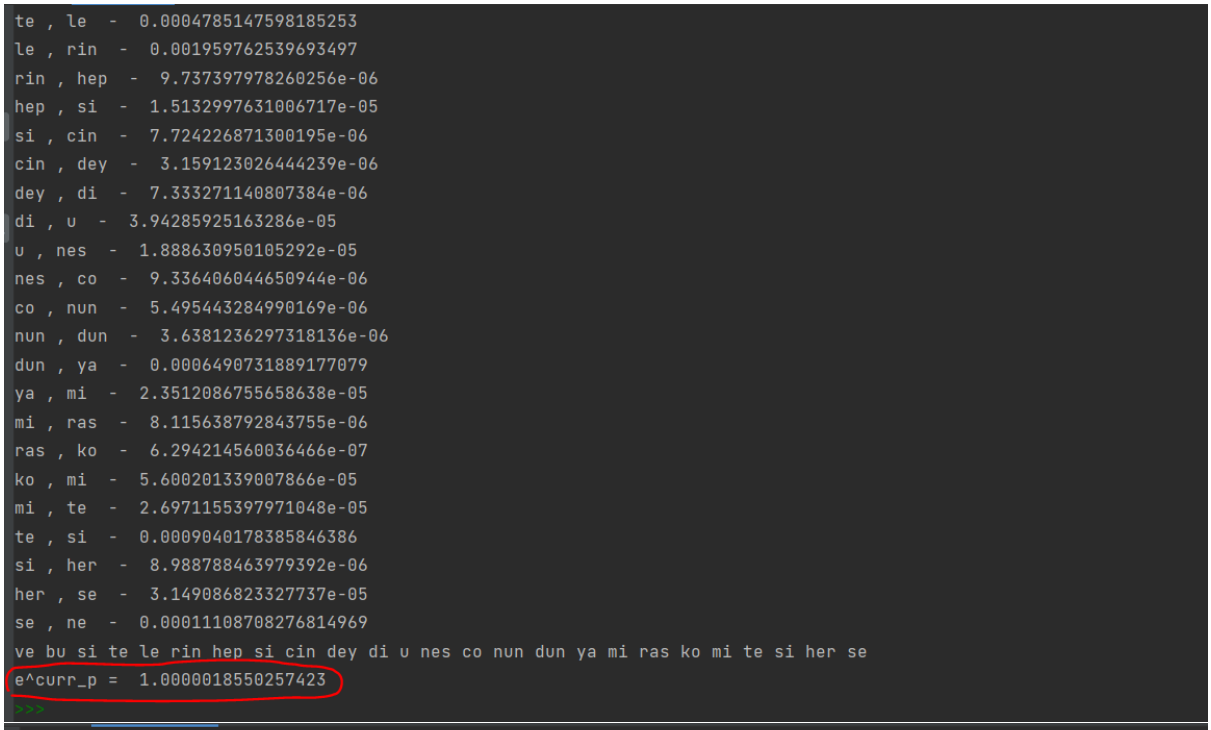
4.1 Unigram Perplexity Results

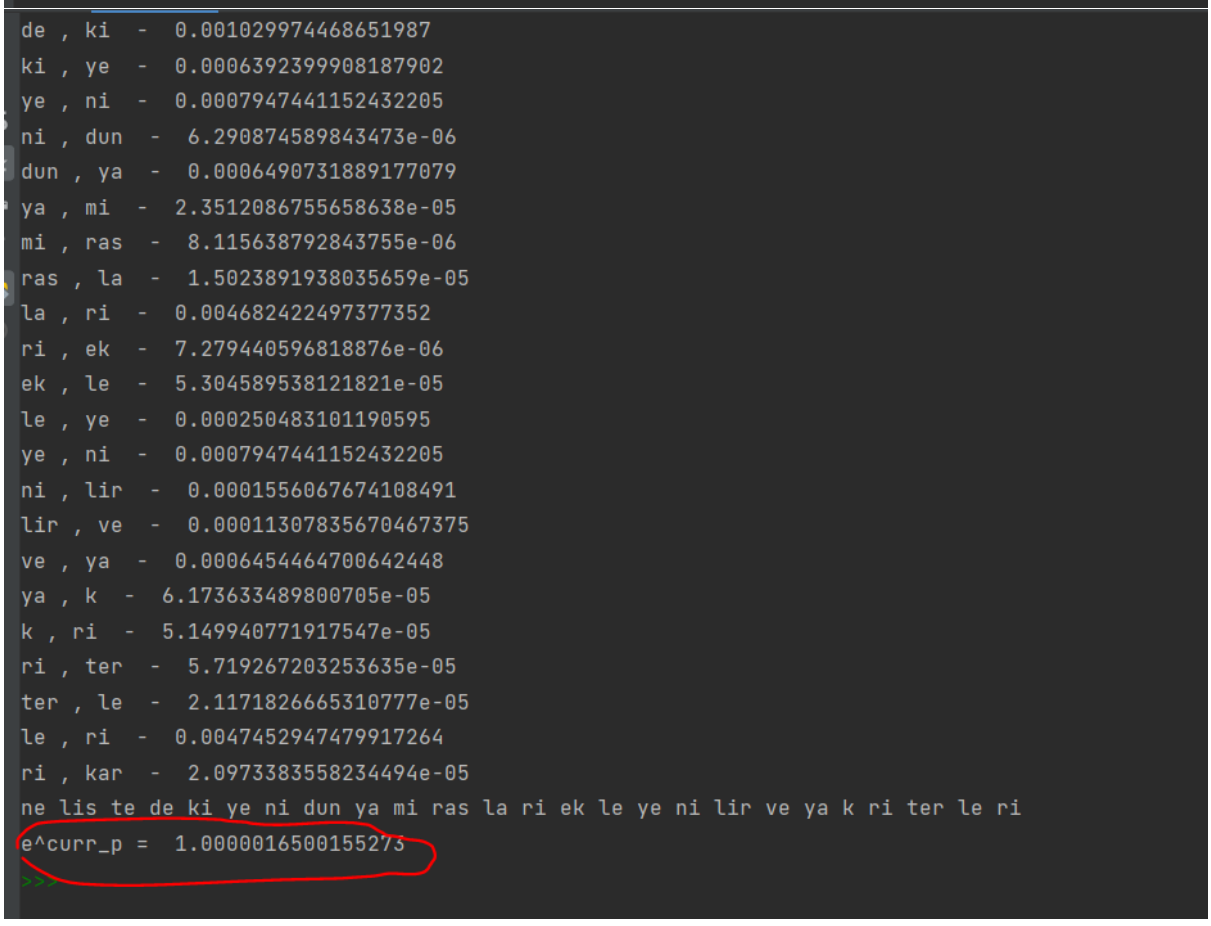
```
me si ve u zun su ren fer man tas yon su re si ne de niy le lam bik bi ra la rin bo
e^curr_p =  1.000001048313834
zul ma ris ki yuk sek tir bu ne den le u re ti ci ler an ti bak te ri yel o zel li
e^curr_p =  1.0000010279311626
gi ne de niy le yuk sek o ran da ser bet ci o tu kul la na rak bu ris ki a zal tir
e^curr_p =  1.0000010172335148
lar an cak yuk sek ser bet ci o tu kul la ni mi ne de niy le bi ra nin a ci ol ma
e^curr_p =  1.0000009928973868
ma si i cin u re ti ci ler kul la na cak la ri ser bet ci ot la ri ni yil lan di
e^curr_p =  1.00000095128364
ra rak a ci lik la ri ni a zal tir lar si se le me a sa ma sin dan on ce lam bik
e^curr_p =  1.0000009604029465
bi ra lar me se fi ci lar da bek le ti lir ler ki mi u re ti ci ler bu a sa ma
e^curr_p =  1.0000009178428988
da sa rap fi ci si kul la nir ar din dan bi ra lar bir kac yil su ren ol gun las ma is
e^curr_p =  1.0000010521761953
le miy le bas ba sa bi ra ki lir lam bik bi ra lar ge nel lik le i ki ay ri sek lam
e^curr_p =  1.0000010151842949
bik bi ra nin ka ris ti ril ma siy la el de e di lir ler har man lan ma dan u re ti
e^curr_p =  1.0000010052936048
len sek lam bik ler bu la nik ve ek sim si bi ra lar dir fi ci o la rak u re ti le
e^curr_p =  1.000001009604958
bi le ce gi gi bi si se le ne rek de u re ti le bi lir ler fi ci lar ge nel lik
e^curr_p =  1.0000009135592636
```

All sentences perplexity results are close to 1.00

4.2 Bigram Perplexity Results

There are 3 examples for Bigrams perplexity, picture per example. For each example, first bigram probabilities are printed, then sentences, then perplexity of that sentence is printed.





```
ma , yan  -  0.00018266345863083577
yan , a  -  3.071078153649475e-05
a , lan  -  0.000625725330940319 4
lan , la  -  0.000163127076882464
la , ri  -  0.004682422497377352
ri , lis  -  4.55529573726483e-05
lis , te  -  0.00026388143235112456
te , den  -  4.362326922797 55e-06
den , ci  -  4.272685381325161e-05
ci , ka  -  0.00025196116019502525
ka , ra  -  0.0008629059010910422
ra , bi  -  7.341710675246262e-05
bi , lir  -  0.0006411902075607467
lir , se  -  2.0939169229428238e-05
se , cim  -  0.00019122388369816084
cim , on  -  4.909987361594975e-07
on , k  -  3.253496057296397e-07
k , ri  -  5.149940771917547e-05
ri , te  -  5.7370586542328875e-05
te , re  -  0.00016856715516265857
re , da  -  6.351547999593233e-05
da , yan  -  2.478828122013196e-05
kar si la ma yan a lan la ri lis te den ci ka ra bi lir se cim on k ri te re da
e^curr_p =  1.0000016826791533
>>>
```

4.3 Three Perplexity Results

There are 3 examples for 3-Gram perplexity, picture per example. For each example, first 3-Gram probabilities are printed, then sentences, then perplexity of that sentence is printed.

```
bu , si , te  -  1.028047027645202e-06
si , te , le  -  2.7599896320031e-05
te , le , rin  -  4.025586072667703e-05
le , rin , hep  -  3.684015476952531e-06
rin , hep , si  -  6.887263149414757e-06
hep , si , cin  -  7.852731956562211e-08
si , cin , dey  -  9.499853229128578e-09
cin , dey , di  -  2.3892560033561734e-06
dey , di , u  -  1.6147548026410257e-07
di , u , nes  -  1.88374388579769e-07
u , nes , co  -  5.04085733967208e-06
nes , co , nun  -  1.5009875542121534e-06
co , nun , dun  -  5.839030943206157e-07
nun , dun , ya  -  2.8437451134810233e-06
dun , ya , mi  -  1.5198150525244852e-05
ya , mi , ras  -  4.671167127034442e-06
mi , ras , ko  -  3.491939798083508e-07
ras , ko , mi  -  2.6154135360884 26e-07
ko , mi , te  -  4.254382823341597e-05
mi , te , si  -  1.4387399276610635e-05
te , si , her  -  3.724279394194606e-07
si , her , se  -  5.328072434107051e-07
her , se , ne  -  5.000396119165853e-06
se , ne , lis  -  None
ve bu si te le rin hep si cin dey di u nes co nun dun ya mi ras ko mi te si her se
e^curr_p =  1.0000022993199411
```

```
lis , te , de  -  2.3467126183984537e-05
te , de , ki  -  4.729496796587837e-06
de , ki , ye  -  6.49769030367272e-06
ki , ye , ni  -  4.750202808831461e-06
ye , ni , dun  -  3.5261531781589076e-06
ni , dun , ya  -  4.624030426532453e-06
dun , ya , mi  -  1.5198150525244852e-05
ya , mi , ras  -  4.671167127034442e-06
mi , ras , la  -  2.9320319128689565e-06
ras , la , ri  -  5.198230766061531e-06
la , ri , ek  -  2.5522280180277993e-06
ri , ek , le  -  3.0324975880109453e-06
ek , le , ye  -  4.409509615923964e-06
le , ye , ni  -  2.6040230782404154e-05
ye , ni , lir  -  2.6008838456149325e-06
ni , lir , ve  -  6.622738377614609e-06
lir , ve , ya  -  2.390624943006683e-06
ve , ya , k  -  2.5557678797604126e-06
ya , k , ri  -  4.465726706111293e-07
k , ri , ter  -  8.855703536040762e-06
ri , ter , le  -  4.184300935747792e-06
ter , le , ri  -  5.053453757376849e-06
le , ri , kar  -  7.743361501786144e-06
ri , kar , si  -  8.529377966563725e-06
ne lis te de ki ye ni dun ya mi ras la ri ek le ye ni lir ve ya k ri ter le ri
e^curr_p =  1.0000022181679977
```

```
la , ma , yan  -  1.6575740570593673e-05
ma , yan , a  -  6.800273184933443e-06
yan , a , lan  -  7.778879906100052e-07
a , lan , la  -  6.646099006812544e-05
lan , la , ri  -  6.371955152401482e-05
la , ri , lis  -  2.0948987771913e-05
ri , lis , te  -  4.2859883804917445e-05
lis , te , den  -  1.0986904759668596e-06
te , den , ci  -  3.336517288223654e-07
den , ci , ka  -  9.687790343849566e-06
ci , ka , ra  -  7.753667661726409e-06
ka , ra , bi  -  9.246915724182655e-06
ra , bi , lir  -  1.0196227567569332e-05
bi , lir , se  -  5.947799414415444e-06
lir , se , cim  -  2.4042249923743995e-07
se , cim , on  -  4.2826193894004345e-07
cim , on , k  -  3.612748161776471e-08
on , k , ri  -  4.371187367829856e-08
k , ri , te  -  1.1874905294136012e-06
ri , te , re  -  3.336517288223654e-07
te , re , da  -  4.3188397110695044e-07
re , da , yan  -  3.6676457282848094e-06
da , yan , mak  -  6.921158964329407e-06
kar si la ma yan a lan la ri lis te den ci ka ra bi lir se cim on k ri te re da
e^curr_p =  1.00000228443804
```

5 Random Sentence Generation from N-Grams

- GT Smoothed dictionary is loaded for N-Grams, content syllables are loaded.
- An average sentence length is decided an until reaching that length syllables random syllables selection is continued.
- For each N-gram there are different acceptance probability for a selected syllable.
- Until reaching length of sentence many random selection is made and if they have higher probability from acceptance limit they are appended to current sentence list.

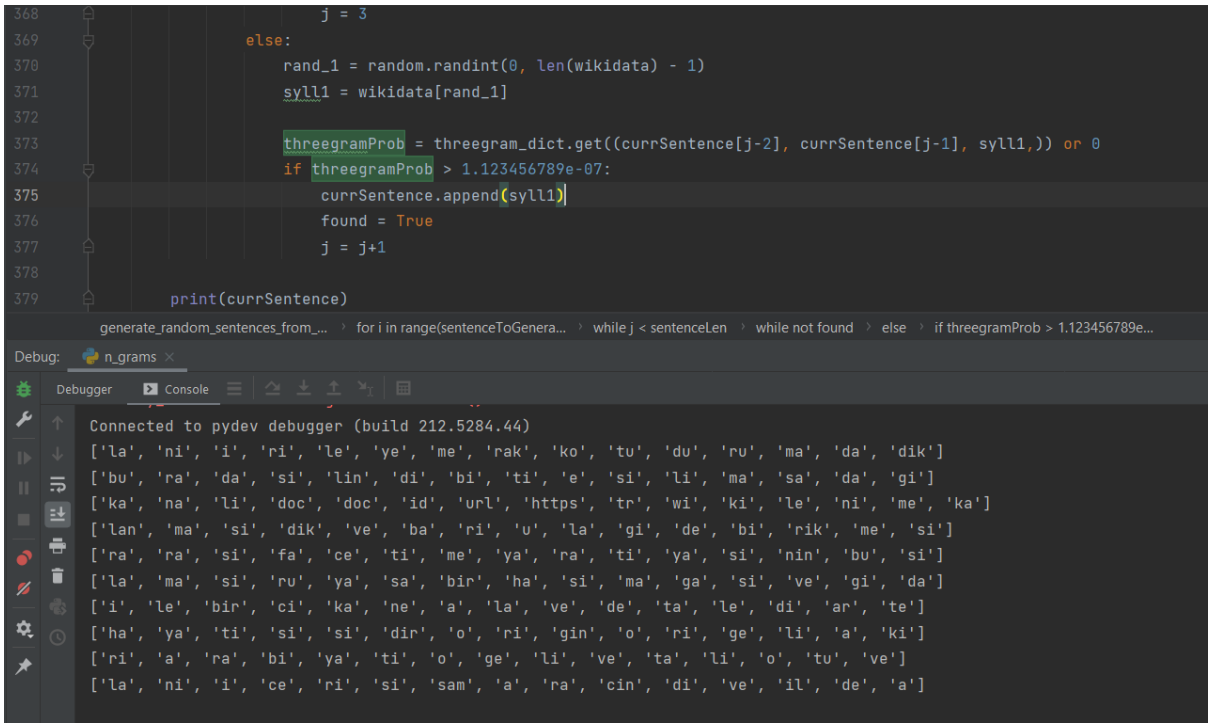
5.1 1-Gram random sentences

```
sentenceToGenerate = 10
sentenceLen = 25
for i in range(sentenceToGenerate):
    # pick a random syllable
    currSentence = []
    for j in range(sentenceLen):
        found = False
        while not found:
            rand_int = secrets.randbelow( len(wikidata)-1)
            syll = wikidata[rand_int]

            unigram_prob = unigram_dict.get((syll,)) or 0
            if unigram_prob > 1.0000000004533484e-04:
                currSentence.append(syll)
                found = True  # accept first syllable

    print(currSentence)
```

Example sentences for 1-Gram

5.2 2-Gram random sentences

5.3 3-Gram random sentences



5.4 4-Gram random sentences

- ['a', 'ni', 'de', 'gi', 'der', 'bu', 'mi', 'to', 'sa', 'de', 'ce']
- ['turk', 've', 'mo', 'gol', 'dev', 'le', 'ti', 'or', 'du', 'gu', 'di']
- ['ha', 'ya', 'ti', 'ni', 'her', 'bir', 'ki', 'ta', 'da']
- ['bir', 'p', 'la', 'ya', 'dir', 'bu', 'da', 'i', 'di', 'ki', 'bu', 'na']
- ['su', 'nu', 'cu', 'ma', 'ka', 'za', 'si', 'ka', 'ta', 'ki', 've', 'bi', 'nek']

5.5 5-Gram random sentences

Random selection and accepting result code for 5-Grams, for more detail refer generate_random_sentences_from_fivegram function from lang_model.py

```
rand_1 = secrets.randbelow(len(wikidata) - 1)
syll1 = wikidata[rand_1]

fivegramProb = fivegram_dict.get((currSentence[j-4], currSentence[j-3], currSent
if fivegramProb > 0.123456789e-10:
    currSentence.append(syll1)
    found = True
    j = j+1
```

- ['ilk', 'tu', 'ru', 'ol', 'du', 've', 'i', 'ri', 'ni', 'i', 'le', 'ku', 'rak', 've', 'col', 'le', 're', 'kom', 'su', 'koy', 'le', 'ri', 'do', 'la', 'na', 'ka', 'dar']
- ['ca', 'ri', 've', 'di', 'ni', 'ko', 'nu', 'da', 'ki', 'si', 'le', 'ra', 'ra', 'si', 'ge', 'ri', 'o', 'de', 'ne', 'gi', 'i', 'le', 'ye', 'ri', 'ni', 'al', 'ti', 'ya', 'ci', 'kar', 'ta', 'ni']
- ['o', 'ra', 'ni', 'ge', 'le', 'cek', 'bir', 'ka', 'ti', 'lin', 'ke', 'di', 'si', 'ol', 'ma', 'si', 'sa', 'ka', 'ya', 'pa', 'bi', 'li', 'mek', 'te', 'dir']

- ['an', 'la', 'mi', 'ni', 'da', 'a', 'ra', 'mak', 'ti', 'bir', 'le', 'sik', 'do', 'nan', 'ma', 'si', 'co', 'gu', 'de', 'ni', 'za', 'ti', 've', 'yi', 'lan', 'ig', 'ne', 'si', 'nin', 'ci', 'ce', 'gi', 'ni', 'a', 'si', 'ri', 'e', 'sit', 'siz', 'li', 'ge', 've', 'ya', 'bi', 'la', 'kis', 'ye', 'ter', 'li']
- ['bas', 'la', 'ri', 'na', 'ha', 'li', 'ci', 'li', 'gi', 'a', 'la', 'ma', 'di', 'film', 'sa', 'ye', 'sin', 'de', 'i', 'da', 'di', 'den', 'yi', 'lin', 'da', 'bu', 'ta', 'le', 'bi', 'ce', 'vap', 'siz', 'ka', 'la', 'bi', 'lir']