EDAN20 - Assignment 2

Oliver Cosic

September 2021

1 Introduction

This assignment is about language models and it is segmented into three parts:

- Segmenting a Corpus
- Counting Unigrams and Bigrams
- Online prediction of words

2 Segmenting a Corpus

The first thing we did was to write a function that replaces all charachters that are neither a letter nor a punctuation sign with a space.

```
def clean(text):
    text=re.sub(nonletter,' ', text)
    return text
nonletter=r'[^\p{L}.;:?!]'
```

And then we wrote a regex to match a punctuation, a sequence of spaces, and an uppercase letter followed by a regex to markup sentence boundaries. We also added some small addons and then put it all in a function to segment sentences.

```
def segment_sentences(text):
    text=clean(text)
    text=re.sub(sentence_boundaries, sentence_markup, text)
    text=re.sub(r'(\p{Lu})',r'<s> \1', text, 1)
    text=re.sub(r'[.!?]',r' </s>', text)
    text=re.sub(r'[\p{Z}]{2,}', ' ', text)
    text=text.lower()
    return text
```

We had some issues at this part since the part where we try to replace the last character in the text with ¡/s¿ does not really do that , it just replaces all the [.!?] with ¡/s¿ but we could not figure out how to just replace the last match. Because of this the resuts of the next part of the assignment is a bit different then the expected result.

3 Counting Unigrams and Bigrams

We were given a function that count Unigrams and Bigrams. Testing this as we mentioned above we noticed that there was a bit more ¡/s¿ in the counters than there was supposed to. With the help of the given counters we wrote two functions, unigram lm and bigram lm. The first one calculates probability, geometric mean probability, entropy rate and perplexity for the unigram of a specific sentence. The tables of the given sentence and 5 more are shown in the appendix. The second one is very similar but calculates the values for bigrams instead and these tables are also shown in the appendix. Looking at the results they are alomst identical to the expected ones.

4 Online prediction of words

The last part of the assignment was about the prediction of the current word a user is typing and the prediction of the next word the user would be typing.

The first part was to predict which word from the corpus would be most probable that the user want to type if he starts with "de". This was calculated with the following method:

```
def segment_sentences(text):
candidates=[]
topcandidates=[]
test=set()
current_word_predictions_1=[]

for key in frequency.keys():
    if key.startswith("de"):
        topcandidates.append((key,frequency[key]))
        test.add(key)

total_words=len(frequency)

def get_prob_unigram(word):
    if word not in frequency:
        return 0
    return frequency[word]/total_words

def get_prob_bigram(word):
    if word not in frequency_bigrams:
        return 0
    return frequency_bigrams[word]/frequency[word[0]]

def get_prob_trigram(words):
    if words not in frequency_trigrams:
        return 0
```

```
    return frequency_trigrams[words] / frequency_bigrams[words[:2]]

for word in test:
    p1=get_prob_unigram((word))
    p2=get_prob_bigram((words[-1], word))
    p=p1+p2
    candidates.append((word, p))

candidates.sort(key=lambda x: x[1], reverse=True)
for ans in candidates[:5]:
    current_word_predictions_1.append(ans[0])
```

And in that script we also included some functions that are used in the next part where we want to predict the next word the user wants to predict. In this case the users has typed "de var en" and trying to predict the next word we did the following:

```
vocab=set()
candidates=[]
next_word_predictions=[]
for key in frequency.keys():
    if key not in vocab:
        vocab.add(key)

for word in vocab:
    p1 = get_prob_unigram((word))
    p2 = get_prob_bigram((tokens[-1], word))
    p3 = get_prob_trigram((tokens[-2], tokens[-1], word)) if len(tokens) >= 3 el
    p=p1*0.01+p2*0.4+p3*0.5
    candidates.append((word, p))

candidates.sort(key=lambda x: x[1], reverse=True)
for ans in candidates[:5]:
    next_word_predictions.append(ans[0])
```

Where tokens is the beginning of the sentence tokenized. The last part was predicting what word the user is typing if he has written "Det var en g". This was done as follows:

```
topcandidates=[]
candidates=[]
test=set()
current_word_predictions_2=[]

for key in frequency.keys():
    if key.startswith("g"):
        topcandidates.append((key,frequency[key]))
        test.add(key)
```

```
for word in test:
    if word!='</s>' or word!='<s>':
        p1 = get_prob_unigram((word))
        p2 = get_prob_bigram((tokens[-1], word))
        p3 = get_prob_trigram((tokens[-2], tokens[-1], word)) if len(tokens) >=
        p=p1*0.01+p2*0.4+p3*0.5
        candidates.append((word, p))

candidates.sort(key=lambda x: x[1], reverse=True)
for ans in candidates[:5]:
    current_word_predictions_2.append(ans[0])
```

5 Results

The results that the notebook compares to our results are the following:

```
(423,
 72,
 ['det', 'de', 'den', 'detta', 'denna'],
 ['stor', 'liten', 'gammal', 'god', 's dan'],
 ['gammal', 'god', 'g ng', 'ganska', 'gl dje'])
```

And our results where:

```
(423,
 72,
 ['det', 'de', 'den', 'dem', 'detta'],
 ['stor', 'liten', 'g ng', 's dan', 'av'],
 ['g ng', 'gammal', 'god', 'ganska', 'gl dje'])
```

So the first parts seems to be correct and the last part is obviously a bit wrong. Dont know if this is because of the issues when segmenting sentences or the actual methods used to predict, did a couple of different versions of the predictions but got basically the same results every time.

6 Norvig

I used a sentence from the beginning of Harry Potter and the Philosopher's stone to test out the segmentation in norvigs notebook. Results was as follows:

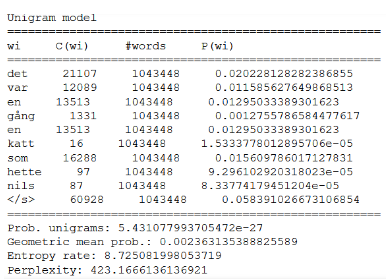
```
myString2='MrandMrsDursleyofnumberfourPrivetDrive\
wereproudtosaythattheywereperfectlynormalthankyouverymuchThey\
werethelastpeopleyoudexpecttobeinvolvedinanythingstrange\
ormysteriousbecausetheyjustdidntholdwithsuchnonsense'
myString2=myString2.lower()
print(segment(myString2))
```

```
print(segment2(myString2))
```

['mr', 'and', 'mrs', 'dursley', 'of', 'number', 'four', 'privet', 'drive', 'were', 'proud', 'to', 'say', 'that', 'they', 'were', 'perfectly', 'normal', 'thankyouverymuch', 'they', 'were', 'the', 'last', 'people', 'you', 'd', 'expect', 'to', 'be', 'involved', 'in', 'anything', 'strange', 'or', 'mysterious', 'because', 'they', 'just', 'didnt', 'hold', 'with', 'such', 'nonsense']
['mr', 'and', 'mrs', 'dursley', 'of', 'number', 'four', 'privet', 'drive', 'were', 'proud', 'to', 'say', 'that', 'they', 'were', 'perfectly', 'normal', 'thank', 'you', 'very', 'much', 'they', 'were', 'the', 'last', 'people', 'you', 'd', 'expect', 'to', 'be', 'involved', 'in', 'anything', 'strange', 'or', 'mysterious', 'because', 'they', 'just', 'didnt', 'hold', 'with', 'such', 'nonsense']

So at first glance one could say that it works pretty well. There is only one mistake with segment and no mistakes in segment2. Altough the words in this particular string are very common so if even this doesnt work properly one can expect that a text with more complicated words or names that are not that common in the corpus or in it at all would pose a bigger issue for the methods.

7 Appendix

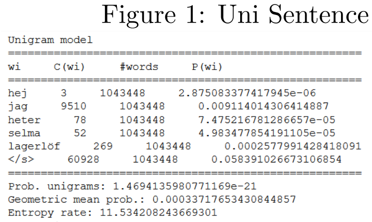


Unigram model

wi	C(wi)	#words	P(wi)
det	21107	1043448	0.020228128282386855
var	12089	1043448	0.011585627649868513
en	13513	1043448	0.01295033389301623
gång	1331	1043448	0.0012755786584477617
en	13513	1043448	0.01295033389301623
katt	16	1043448	1.5333778012895706e-05
som	16288	1043448	0.015609786017127831
hette	97	1043448	9.296102920318023e-05
nils	87	1043448	8.33774179451204e-05
</s>	60928	1043448	0.058391026673106854

Prob. unigrams: 5.431077993705472e-27
Geometric mean prob.: 0.002363135388825589
Entropy rate: 8.725081998053719
Perplexity: 423.1666136136921

Figure 1: Uni Sentence 1



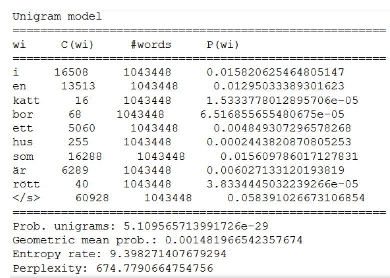
Unigram model

wi	C(wi)	#words	P(wi)
hej	3	1043448	2.875083377417945e-06
jag	9510	1043448	0.009114014306414887
heter	78	1043448	7.475216781286657e-05
selma	52	1043448	4.983477854191105e-05
lagerlöf	269	1043448	0.0002577991428418091
</s>	60928	1043448	0.058391026673106854

Prob. unigrams: 1.4694135980771169e-21
Geometric mean prob.: 0.00033717653430844857
Entropy rate: 11.534208243669301
Perplexity: 2965.80544091245

.8

Figure 2: Uni Sentence 2

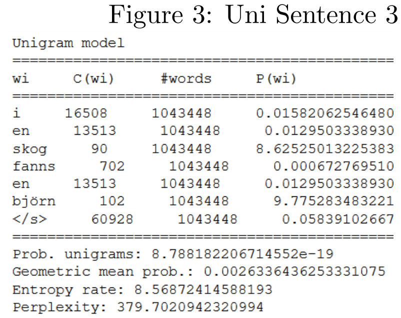


Unigram model

wi	C(wi)	#words	P(wi)
i	16508	1043448	0.015820625464805147
en	13513	1043448	0.01295033389301623
katt	16	1043448	1.5333778012895706e-05
bor	68	1043448	6.516855655480675e-05
ett	5060	1043448	0.004849307296578268
hus	255	1043448	0.0002443820870805253
som	16288	1043448	0.015609786017127831
är	6289	1043448	0.006027133120193819
rött	40	1043448	3.8334445032239266e-05
</s>	60928	1043448	0.058391026673106854

Prob. unigrams: 5.109565713991726e-29
Geometric mean prob.: 0.001481966542357674
Entropy rate: 9.398271407679294
Perplexity: 674.7790664754756

Figure 3: Uni Sentence 3



Unigram model

wi	C(wi)	#words	P(wi)
i	16508	1043448	0.01582062546480
en	13513	1043448	0.0129503338930
skog	90	1043448	8.62525013225383
fanns	702	1043448	0.000672769510
en	13513	1043448	0.0129503338930
björn	102	1043448	9.775283483221
</s>	60928	1043448	0.05839102667

Prob. unigrams: 8.788182206714552e-19
Geometric mean prob.: 0.0026336436253331075
Entropy rate: 8.56872414588193
Perplexity: 379.7020942320994

.8 Unigram model

Figure 4: Uni Sentence 4

```
Unigram model
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
denna      886      1043448      0.0008491079574640997
uppgift      8      1043448      7.666889006447853e-06
tog      638      1043448      0.0006114343982642163
en      13513      1043448      0.01295033389301623
evighet      2      1043448      1.9167222516119632e-06
att      28020      1043448      0.026853278745083607
göra      1158      1043448      0.001109782183683327
</s>      60928      1043448      0.058391026673106854
=====================================================
Prob. unigrams: 1.7193064204106482e-25
Geometric mean prob.: 0.0008024521788833151
Entropy rate: 10.28329696032831
Perplexity: 1246.1801791000073
```

Figure 5: Uni Sentence 6

```
Unigram model
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
jag      9510      1043448      0.009114014306414887
gillar      1      1043448      9.583611258059816e-07
denna      886      1043448      0.0008491079574640997
kurs      3      1043448      2.875083377417945e-06
</s>      60928      1043448      0.058391026673106854
=====================================================
Prob. unigrams: 1.2450831652006893e-18
Geometric mean prob.: 0.0002624458273133619
Entropy rate: 11.895692719559058
Perplexity: 3810.3101513821885
```

Figure 6: Uni Sentence 7

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> det 5672 59047 0.09605907158704083
det var 3838 21107 0.18183541005353673
var en 712 12089 0.05889651749524361
en gång 705 13513 0.05217198253533634
gång en 20 1331 0.015026296018031555
en katt 6 13513 0.0004440168726411604
katt som 2 16 0.125
som hette 45 16288 0.002762770137524558
hette nils 0 97 8.33774179451204e-05 backoff:nils
nils </s> 2 87 0.022988505747126436
=====================================================
Prob. bigrams: 2.370331925761903e-19
Geometric mean prob.: 0.0202675304079874
Entropy rate: 6.187154470402314
Perplexity: 72.86501915348336
```

Figure 7: Bi Sentence 1

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> hej 1 59047 1.6935661422256845e-05
hej jag 0 3 0.009114014306414887 backoff:jag
jag heter 4 9510 0.0004206098843322818
heter selma 0 78 4.983477854191105e-05 backoff:selma
selma lagerlöf 11 52 0.21153846153846154
lagerlöf </s> 32 269 0.11895910780669144
=====================================================
Prob. bigrams: 8.141621634395843e-17
Geometric mean prob.: 0.005029561590907225
Entropy rate: 8.907910239403314
Perplexity: 480.3393662580067
```

.8

Figure 8: Bi Sentence 2

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> i 682 59047 0.01155012108997917
i en 575 16508 0.03483159680155076
en katt 6 13513 0.0004440168726411604
katt bor 0 16 6.516855655480675e-05 backoff:bor
bor ett 2 68 0.029411764705882353
ett hus 21 5060 0.004150197628458498
hus som 26 255 0.10196078431372549
som är 294 16288 0.018050098231827114
är rött 1 6289 0.0001590077913817777
rött </s> 4 40 0.1
=====================================================
Prob. bigrams: 4.1583256831432973e-23
Geometric mean prob.: 0.009233284710229562
Entropy rate: 7.434834342684528
Perplexity: 173.02471794077644
```

Figure 9: Bi Sentence 3

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> i 682 59047 0.01155012108997917
i en 575 16508 0.03483159680155076
en skog 8 13513 0.0005920224968548805
skog fanns 0 90 0.0006727695103157992 backoff:fanns
fanns en 74 702 0.10541310541310542
en björn 10 13513 0.0007400281210686006
björn </s> 12 102 0.11764705882352941
=====================================================
Prob. bigrams: 1.4705789269091873e-15
Geometric mean prob.: 0.013993812232298833
Entropy rate: 7.038933886663469
Perplexity: 131.50135788532387
```

.8

Figure 10: Bi Sentence 4

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> denna 80 59047 0.0013548529137805477
denna uppgift 0 886 7.666889006447853e-06 backoff:uppgift
uppgift tog 0 8 0.0006114343982642163 backoff:tog
tog en 16 638 0.025078369905956112
en evighet 0 13513 1.9167222516119632e-06 backoff:evighet
evighet att 0 2 0.026853278745083607 backoff:att
att göra 369 28020 0.013169164882226981
göra </s> 92 1158 0.07944732297063903
=====================================================
Prob. bigrams: 8.577377123342743e-24
Geometric mean prob.: 0.0027355168198726853
Entropy rate: 9.578217215491996
Perplexity: 764.417616063088
```

Figure 11: Bi Sentence 6

```
Bigram model
=====================================================
wi wi+1 Ci,i+1 C(i) P(wi+1|wi) backoff
=====================================================
<s> jag 2669 59047 0.04520128033600352
jag gillar 0 9510 9.583611258059816e-07 backoff:gillar
gillar denna 0 1 0.0008491079574640997 backoff:denna
denna kurs 0 886 2.875083377417945e-06 backoff:kurs
kurs </s> 0 3 0.058391026673106854 backoff:</s>
=====================================================
Prob. bigrams: 6.175034545673556e-18
Geometric mean prob.: 0.0013544819811344246
Entropy rate: 11.43365170004508
Perplexity: 2766.127037458285
```

Figure 12: Bi Sentence 7