EDAN20 - Lab 2

André Frisk - an8218fr-s

September 2021

Language Models

This lab teaches the student about language models and a more introduction to regular expressions. The previous lab used regular expressions in the introduction part, however in this lab the student must write own regular expressions to be used later on when creating bigrams and unigrams when trying to compute the probability of a sentence and to guess the best candidate for a sentence or a word to be written.

Regular expression function

This function is used primarily as a preparation before inserting the list of words in the prediction functions and unigram and bigram further on. This function firstly creates a regular expression `sequence_boundaries`, that matches a punctuation, a sequence of spaces and a uppercase letter, in other words a end of a sentence. When this regular expression is matched it should be replaced with `sentence_markup` which inserts a `<s>` at the beginning of a sentence and `</s>` at the end and with the text in between. As the regular expression checks for the end of a sentence the beginning and the end of the text does not have these signs, so they are added manually. Then the newlines are swapped with a blank space and punctuation is removed and the text is set to lowercase. This text is then split so that every word is separated and inserted in a list.

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

Computing the likelihood of a sentence

Unigrams

The text that was used in the regular expression function above is inserted into unigrams which computes a sentence's probability using unigram frequency (separate list of number of times a word appears). This table that is seen in the Appendix consists of several parts: the words sent in to the function, the frequency of that specific word in the text, how many words that there are in total in the text and the probability of that specific word to appear. To calculate the probability of a word just take the frequency of that word divided by the total amount of words in the text. In the unigrams we calculate four other variables:

- Probability unigram: with a probability starting at 1, every probability of the sent in sentence is multiplied to this starting probability
- Geometric mean probability: is the average rate of return of a set of values calculated using the products of the terms. It takes several values and multiplies them together and sets them to the $\frac{1}{n}$th power. This is done taking the total probability of the unigram and to the power of the $\frac{1}{sentence-length}$.
- Entropy rate: computes the per word probability of a word sequence, the lower the better.
- Perplexity: a measurement of how well a probability distribution predicts a sample. A low perplexity indicates the probability distribution is good at predicting the sample.

Bigrams

The text that was used in the regular expression function above is inserted into bigrams which computes a sentence's probability using the frequency from unigram and bigram (number of times two words appear after each other, for example 'nils' 'holgersson', which appears 6 times). Using bigrams that is at the start of a sentence a `<s>` is inserted at the beginning of the sentence in these test cases. This table that is seen in the Appendix consists of several parts that are a bit different from the unigram table (except the four variables appearing after the table as in the unigrams table, total probability, geometric mean etc.):

- wi: the sentence sent in
- wi+1: the sentence sent in but one word forward
- Ci,i+1: how many times this combination of two words appear in the text, for example 'en gång'
- C(i): number of appearances of the word checked in wi
- P(wi+1|wi): conditional probability, if the events of wi occurred then then this is the probability of A under the condition B. For example, the probability of that a person has a cough is quite low, however if the person is sick the probability is higher, P(cough|sick)

Online prediction of words

The program is supposed to handle two cases, to predict the next word in a sentence and to predict the current word the user is currently typing in. To handle this more efficient a trigrams is created which takes the regular expression text from above into the trigrams function and creates a dictionary. It counts the amount of times three words appear after each other. For example if the call to the trigram frequencies with `('det', 'var', 'en')` is done then the dictionary will print out 330 as these three words appear after each other 330 times in Selma.txt

A program is written to rank the five first candidates to a word that is currently being written at the start of a sentence, using the bigram frequencies.

```
current_word_predictions_1 = []
freqlist = list(frequency_bigrams.items())
candidates = [word for word in freqlist if word[0][0]
== '<s>' and word[0][1].startswith(starting_text)]
sorted_candidates = sorted(candidates, key = lambda word : (-word[1],word[0][1]))
current_word_predictions_1 = [x[0][1] for x in sorted_candidates[0:cand_nbr]]
```

This program takes the bigram frequencies and puts the words that starts with <s> the input word (for example 'de'). The program will then sort the highest frequency first of these words and print out which five words that could be the possible word that the user is typing in.

The next program works similar to the previous program but it predicts the next possible word in the sentence written using the trigram frequencies.

```
next_word_predictions = []
trigramfreqlist = list(frequency_trigrams.items())
candidates1 = [word for word in trigramfreqlist if word[0][0]
== tokens[1] and word[0][1] == tokens[2]]
sorted_candidates1 = sorted(candidates1, key = lambda word : (-word[1],word[0][2]))
candidates_5 = sorted_candidates1[0:5]
next_word_predictions = [word[0][2] for word in candidates_5]
```

The program goes through the trigram frequencies list and checks if the two last words in the sentence sent in is matched with the current word in the trigram list, if it is it takes the third word that appears after these two last words in the sentence and puts in into a list which then is sorted to show the highest frequent word first. The list with the five next possible words is printed out.

The last program also works as the previous two programs but combined. Here a sentence is sent in and a word which is currently being written, the program shall then predict the current possible candidate using the trigram frequencies and the functionality of the two other program in this section. The program checks the current word in trigram frequencies and checks if the last two words is equal to the one being checked and checks all the words in this trigram frequency list that also start with the letters that are currently being written. This returns a list of the five possible candidates that goes well with this sentence.

Peter Norvig's notebook: segmentation

The string used was
"whatwillyoudotodayafterschoolimightgoseemygrandmotheratherhousebutafterthatimightstudy".
The result was the following:
segment(): ['what', 'will', 'youdotodayaftersc', 'hoolimightgoseemy', 'grandmother', 'atherhousebutaft', 'erthatimightstudy']
segment2(): ['what', 'will', 'you', 'do', 'today', 'after', 'school', 'i', 'might', 'go', 'see', 'my', 'grandmother', 'at', 'her', 'house', 'but', 'after', 'that', 'i', 'might', 'study'].

As seen of the result `segment2()` handled it perfectly while `segment()` did not. This is because `segment()` takes the maximum candidate in Pwords of these words, if these words that are scrambled together doesn't have a high probability and can't be found then a correct segmentation of text can't be found. `segment2()` however, uses bigram modeling which checks both the lists P2w and P1w to gain a conditional probability which checks the current and previous word together. This makes it easier for the program to compare words and find a better word for the segmentation. This works about the same as above with bigrams as it checks the previous words and tries to check in which word it could be to create a fully restored and logical sentence using probability.

Appendix

In this appendix the tables for the unigrams and bigrams are shown with the caption of the figures being the text that was tested with these functions.

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
det      21108      1041518      0.020266572445219382
var      12090      1041518      0.011608056701852488
en      13514      1041518      0.0129752918336505
gång      1332      1041518      0.0012789025249683634
en      13514      1041518      0.0129752918336505
katt      16      1041518      1.5362192492112473e-05
som      16288      1041518      0.0156387119569705
hette      97      1041518      9.313329198343188e-05
nils      87      1041518      8.353192167586158e-05
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     5.366006871766503e-27
Geometric mean prob.: 0.0023602886742622193
Entropy rate:     8.378943970226945e-05
Perplexity:     423.67698955831355
```

Figure 1: Unigram: det var en gång en katt som hetter nils </s>

```
=======================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=======================================================
<s>      det      5672      59033   0.096081852523165
det      var      3839      21108   0.1818741709304529
var      en      712      12090   0.058891645988420185
en      gång      706      13514   0.052242119283705785
gång      en      20      1332   0.015015015015015015
en      katt      6      13514   0.0004439840165754033
katt      som      2      16   0.125
som      hette      45      16288   0.002762770137524558
hette      nils      0      97   0   *backoff:     8.353192167586158e-05
nils      </s>      2      87   0.022988505747126436
=======================================================
Prob. unigrams:     2.376829134050061e-19
Geometric mean prob.: 0.013727763740747877
Entropy rate:     6.186759560859045
Perplexity:     72.84507650956418
```

Figure 2: Bigram: det var en gång en katt som hetter nils </s>

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
<s>      59033      1041518      0.05667976933667973
se      1989      1041518      0.001909712554175732
på      14250      1041518      0.013681952688287672
nils      87      1041518      8.353192167586158e-05
holgersson      66      1041518      6.336904402996396e-05
tummetott      110      1041518      0.00010561507338327326
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     4.69275378876932e-20
Geometric mean prob.: 0.0017329075567666542
Entropy rate:     6.164860053134522e-05
Perplexity:     577.0648273159188
```

Figure 3: Unigram: se på nils holgersson tummetott </s>

```
=======================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=======================================================
<s>      <s>        0     59033   0   *backoff:     0.05667976933667973
<s>      se        196     59033   0.0033201768502363086
se      på        167     1989   0.08396178984414278
på      nils        4     14250   0.0002807017543859649
nils      holgersson        66     87   0.7586206896551724
holgersson      tummetott        1     66   0.015151515151515152
tummetott      </s>        22     110   0.2
=======================================================
Prob. unigrams:     1.0195930098944588e-11
Geometric mean prob.: 0.026901423668097765
Entropy rate:     5.2161736651506265
Perplexity:     37.172753841496274
```

Figure 4: Bigram: se på nils holgersson tummetott </s>

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
pojken      1028      1041518      0.0009870208676182265
sprang      201      1041518      0.00019298754318216295
genast      508      1041518      0.0004877496116245711
fram      2019      1041518      0.0019385166650984428
till      9139      1041518      0.008774692324088494
katten      40      1041518      3.8405481230281185e-05
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     3.4401332589183384e-21
Geometric mean prob.: 0.0011930361666639812
Entropy rate:     6.526821960022392e-05
Perplexity:     838.1975567398285
```

Figure 5: Unigram: pojken sprang genast fram till katten </s>

```
=====================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=====================================================
<s>      pojken      282      59033   0.0047769891416665254
pojken      sprang      8      1028   0.007782101167315175
sprang      genast      3      201   0.014925373134328358
genast      fram      14      508   0.027559055118110236
fram      till      514      2019   0.254581475978207
till      katten      3      9139   0.000328263486158223
katten      </s>      4      40   0.1
=====================================================
Prob. unigrams:     1.2778798572899351e-13
Geometric mean prob.: 0.014390305155136167
Entropy rate:     6.118759004150039
Perplexity:     69.4912296312967
```

Figure 6: Bigram: pojken sprang genast fram till katten </s>

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
katten      40      1041518      3.8405481230281185e-05
svarade      450      1041518      0.00043206166384066334
inte      11355      1041518      0.01090235598424607
genast      508      1041518      0.0004877496116245711
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     5.001316165884923e-15
Geometric mean prob.: 0.0013798022919385496
Entropy rate:     4.561285893589806e-05
Perplexity:     724.7415124923822
```

Figure 7: Unigram: katten svarade inte genast </s>

```
=======================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=======================================================
<s>      katten      5      59033      8.469838903664053e-05
katten      svarade      1      40      0.025
svarade      inte      22      450      0.04888888888888889
inte      genast      18      11355      0.001585204755614267
genast      </s>      52      508      0.10236220472440945
=======================================================
Prob. unigrams:     1.679771945792321e-11
Geometric mean prob.: 0.0069992267727963554
Entropy rate:     7.158588732985637
Perplexity:     142.87292474743987
```

Figure 8: Bigram: katten svarade inte genast </s>

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
men      8144      1041518      0.00781935597848525
i      16501      1041518      0.015843221144521746
sitt      1070      1041518      0.0010273466229100217
eget      130      1041518      0.00012481781399841386
liv      362      1041518      0.00034756960513404475
såg      2284      1041518      0.0021929529782490557
hon      13313      1041518      0.012782304290468336
ingen      1593      1041518      0.0015294982899959483
mening      155      1041518      0.0001488212397673396
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     1.9967803274089886e-27
Geometric mean prob.: 0.002138124691804367
Entropy rate:     8.515876147722111e-05
Perplexity:     467.6995704848711
```

Figure 9: Unigram: men i sitt eget liv såg hon ingen mening </s>

```
=====================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=====================================================
<s>      men      3824      59033      0.06477732793522267
men      i      172      8144      0.02111984282907662
i      sitt      150      16501      0.009090358160111509
sitt      eget      51      1070      0.04766355140186916
eget      liv      9      130      0.06923076923076923
liv      såg      1      362      0.0027624309392265192
såg      hon      105      2284      0.04597197898423818
hon      ingen      13      13313      0.0009764891459475701
ingen      mening      1      1593      0.0006277463904582549
mening      </s>      31      155      0.2
=====================================================
Prob. unigrams:     6.3892007025688715e-19
Geometric mean prob.: 0.015154606115654707
Entropy rate:     6.044099834345083
Perplexity:     65.9865385064017
```

Figure 10: Bigram: men i sitt eget liv såg hon ingen mening </s>

```
=====================================================
wi      C(wi)      #words      P(wi)
=====================================================
han      21589      1041518      0.020728398357013515
hade      13198      1041518      0.012671888531931278
butik      2      1041518      1.920274061514059e-06
inne      419      1041518      0.0004022974158871954
i      16501      1041518      0.015843221144521746
staden      196      1041518      0.0001881868580283778
</s>      59033      1041518      0.05667976933667973
=====================================================
Prob. unigrams:     3.429086169560137e-20
Geometric mean prob.: 0.0016569568538666113
Entropy rate:     6.208316873977394e-05
Perplexity:     603.516016525378
```

Figure 11: Unigram: han hade butik inne i staden </s>

```
=====================================================
wi      wi+1      Ci,i+1      C(i)      P(wi+1|wi)
=====================================================
<s>      han      5059      59033      0.08569783002727288
han      hade      2044      21589      0.09467784519894391
hade      butik      1      13198      7.576905591756326e-05
butik      inne      1      2      0.5
inne      i      214      419      0.5107398568019093
i      staden      29      16501      0.0017574692442882249
staden      </s>      53      196      0.27040816326530615
=====================================================
Prob. unigrams:     7.46083358123965e-11
Geometric mean prob.: 0.03574829524595414
Entropy rate:     4.8059817450543125
Perplexity:     27.973361893758454
```

Figure 12: Bigram: han hade butik inne i staden </s>