

Accelerated Natural Language Processing

Report of Assignment 1

Student number: s1836694 s1802612

1. Task 1

Figure 1.1 shows the code of the preprocessing function. The function utilizes regular expression, which made dataset only contain characters in English alphabet, space, '0', '#' and '.' character. The first step is to lowercase all the English alphabet. Then, all digits are substituted with 0. After that, we use '##' to mark the start of a sequence and use '#' to mark the end. Finally, characters with accent and umlauts and other punctuation marks are removed.

```
#Q1 - preprocessing
def preprocess_line(line):

    #Lower Case
    line = line.lower()

    #all digits to 0
    line = re.sub(r'[0-9]', '0', line)

    #What else we did
    #add '##' at the beginning of every paragraph
    line = re.sub(r'^', '##', line)
    #add '#' at the end of every paragraph
    line = re.sub(r'\n', '#\n', line)
    ''' The reason for marking the start and the end of each sentence:
    - Marking the beginning of each sentence makes the 3-gram model
    always have a starting state '##' as two-character history
    (when a new sequence starts or when the last sentence is over).
    - The introduction of the ending symbol '#' makes each sentence finite
    and allows the model to split each sentence.
    ...

    #remove characters with accent and umlauts and other punctuation marks
    line = re.sub(r'^a-z0\s#.', '', line)

    return line
```

Figure 1-1 The preprocessing function

2. Task 2

The estimation method of generating the 'model-br.en' might be Maximum-likelihood estimation (MLE). E.q. 2.1 shows how to use MLE to calculate the probability in the trigram model.

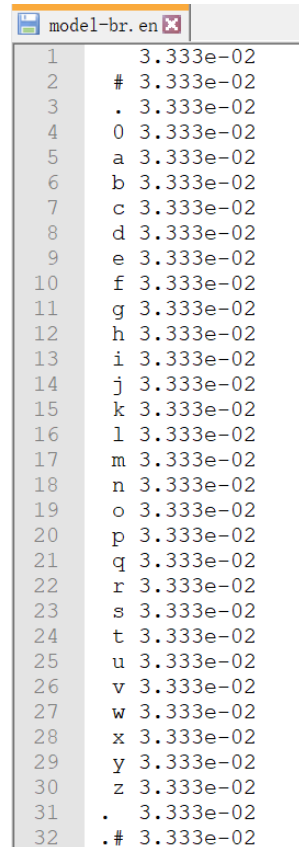
$$p(x_i | x_{i-2}, x_{i-1}) = \frac{c(x_{i-2}x_{i-1}x_i)}{c(x_{i-2}x_{i-1})} \quad (2.1)$$

When going through the model file, it is proved that the sum of the probabilities of all trigrams with the same two-character history equals to 1.

$$\sum_i p(x_i | x_{n-1}, x_n) = \frac{\sum_i c(x_{n-1}x_nx_i)}{c(x_{n-1}x_n)} = 1 \quad (2.2)$$

Figure 2-1 lists the first 30 lines of 'model-br.en', and all the trigrams have the same previous two characters (two 'space' characters). Eq.2.3 illustrates the procedure of adding up all the probabilities and the result does equal to one.

$$3.333 * 10^{-2} * 30 \approx 1 \quad (2.3)$$



1	3.333e-02
2	# 3.333e-02
3	. 3.333e-02
4	0 3.333e-02
5	a 3.333e-02
6	b 3.333e-02
7	c 3.333e-02
8	d 3.333e-02
9	e 3.333e-02
10	f 3.333e-02
11	g 3.333e-02
12	h 3.333e-02
13	i 3.333e-02
14	j 3.333e-02
15	k 3.333e-02
16	l 3.333e-02
17	m 3.333e-02
18	n 3.333e-02
19	o 3.333e-02
20	p 3.333e-02
21	q 3.333e-02
22	r 3.333e-02
23	s 3.333e-02
24	t 3.333e-02
25	u 3.333e-02
26	v 3.333e-02
27	w 3.333e-02
28	x 3.333e-02
29	y 3.333e-02
30	z 3.333e-02
31	. 3.333e-02
32	.# 3.333e-02

Figure 2-1 Line 1 to 30 of 'model-br.en'

3. Task 3

3.1 Estimation method and explanation

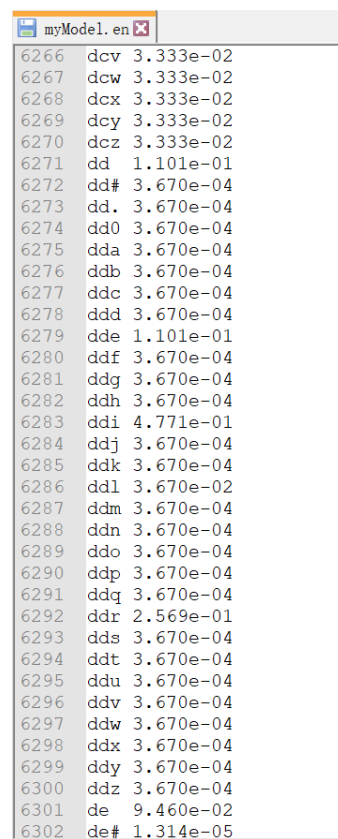
We built the trigram model which estimates the probabilities by Maximum-Likelihood Estimation. Markov assumption was utilized to simplify the model. It is assumed that the probability of a character only depends on the previous two characters in the trigram language model (Jurafsky and Martin, 2009, p.194). As there is no need to consider all the history for the current character, this assumption makes calculating the probability of a whole sequence much easier. Equation 3.1 shows how to compute the probability of the current character given two-history characters. $c(x_{i-2}x_{i-1}x_i)$ refers to the number of the trigram

' $x_{i-2}x_{i-1}x_i$ ' in the training text and $c(x_{i-2}x_{i-1})$ is the number of two-character history ' $x_{i-2}x_{i-1}$ '. The operation of division figures out the relative frequency (RF) of the trigram which is the estimation of the desired probability.

$$p(x_i|x_{i-2}, x_{i-1}) = \frac{c(x_{i-2}x_{i-1}x_i)}{c(x_{i-2}x_{i-1})} \quad (3.1)$$

3.2 The excerpt of the language model for English and the expectation

Figure 3-1 shows the excerpt of the trained model (myModel.en). As illustrated, all the estimated probabilities with the same previous two-character history ('dd').



Line	Character	Probability
6266	dcv	3.333e-02
6267	dcw	3.333e-02
6268	dcx	3.333e-02
6269	dcy	3.333e-02
6270	dcz	3.333e-02
6271	dd	1.101e-01
6272	dd#	3.670e-04
6273	dd.	3.670e-04
6274	dd0	3.670e-04
6275	dda	3.670e-04
6276	ddb	3.670e-04
6277	ddc	3.670e-04
6278	ddd	3.670e-04
6279	dde	1.101e-01
6280	ddf	3.670e-04
6281	ddg	3.670e-04
6282	ddh	3.670e-04
6283	ddi	4.771e-01
6284	ddj	3.670e-04
6285	ddk	3.670e-04
6286	ddl	3.670e-02
6287	ddm	3.670e-04
6288	ddn	3.670e-04
6289	ddo	3.670e-04
6290	ddp	3.670e-04
6291	ddq	3.670e-04
6292	ddr	2.569e-01
6293	dds	3.670e-04
6294	ddt	3.670e-04
6295	ddu	3.670e-04
6296	ddv	3.670e-04
6297	ddw	3.670e-04
6298	ddx	3.670e-04
6299	ddy	3.670e-04
6300	ddz	3.670e-04
6301	de	9.460e-02
6302	de#	1.314e-05

Figure 3-1 Line 6271 to 6300 of 'myModel.en'

It is expected that the sum of all the probabilities listed in Figure 3-1 should equal to 1 (see Equation 2.2 in Task 2). As shown in Equation 3.2, we add the probabilities from Lines 6271 to 6300 and the result is approximately equal to one, which matches our expectation.

$$(1.01 * 2 + 4.771 + 2.569) * 10^{-1} + 3.67 * 10^{-2} + 3.67 * 10^{-4} * 25 = 1.000075 \approx 1 \quad (3.2)$$

3.3 Zero probability and solution

As the training dataset is small, the model trained on the dataset cannot cover all the possible trigrams. This will cause problems. When the model is applied to the test dataset, it is very likely to encounter an error that the trigram in test set does not occur in the model. Thus, we use the add-k smoothing method to deal with the unseen characters in training dataset.

The reason why add-k smoothing method is applied in the task is that add-1 smoothing method dramatically reduces the RF of occurred trigrams in the training set. The value of k is set to 0.01 in order to not change them too much. Actually, we also tried different values of k and found that there is a trade-off. As the value of k decreases, the difference between the perplexity of 'test' file using different language models increases (which is desired), but the perplexity for the same language model also increases (which is not desired).

4. Task 4

4.1 Outputs

(a) The output for the model trained on the English training data:

"we poin amend forculd i a port to us the be somy requidection proution same derammety the mis nons acto of adher rif unden ces vourcifieser ards forts on eurs st the sheinely.

wevand i a whe for.

gen to be matte plembet cale region ou we untow out whiplez tor sestany conatere arliat andaystruch fo"

(b) The output for the model in model-br.en:

"ont wandy bunchis.

what it wanna.

on man is.

whave blo.

thatch.

than a ple backing.

dole maree the to this ing.

ye.

whos right done daddy.

mys uppy sh thel do.

did wall i kiss ing do you kay.

da.

commy hos se way.

se.

you put one.
that se his.
the don thoes thats what themb hicks wha”

4.2 Explanation of the code

The function takes two arguments: the language model in Dictionary type and the number of characters that will be generated. The keys of Dictionary representing language model are the two-character history and the values are the distribution of the third character. The distribution is also in a dictionary type, with the keys representing the third character and the values representing the corresponding conditional probability. Function “generate_random_sequence” returns a list of N characters.

There are two main steps in the process. The function regards the previously generated 2 characters as the key to find the corresponding distribution; then it generates one character (sample) from the distribution. Furthermore, to avoid the early stopping of generation, the function sets the indexing key to ‘##’ (starting symbol) whenever the previous character is ‘#’ (ending symbol) excluding the situation of two continuous ‘#’ (which becomes starting symbols again).

4.3 Differences between two generated sequences and possible reasons

For the sake of simplicity, Sequence (a) refers to the one generated from the model on English training data and Sequence (b) refers to the one generated according to ‘model-br.en’. It is obvious that Sequence (a) has less sentences but with length, while Sequence (b) is made up by shorter sentences.

The most probable reason is that ‘model-br.en’ has higher probability of generating the ending symbol - ‘#’. The two model files almost equally have an extremely high probability of generating ‘#’ given two-character history. However, the likelihood of generating ‘.’ given two-character history in ‘model-br.en’ is much higher than that of another model. The higher probability of the occurrence of ‘.’ leads to more ‘#’ (ending symbol). Essentially, the differences between two models is derived from their training data, which means that the training data of ‘model-br.en’ should contain many short sentences and full stop symbols.

5. Task 5

5.1 The perplexity of the test document and the prediction of classification

Table 5.1 shows the perplexities of the test document using different language models. To make a prediction on which language the test file is written in, every model is run on the test file and calculate the perplexity respectively. The language model which gives the lowest perplexity is the more probable language of the test file. As shown in the table, the model trained on 'training.en' reaches the lowest perplexity, which means that the test file is more likely to be written by English.

Table 5-1 The perplexity of the test file under different language models

	model-br.en	myModel.en	myModel.de	myModel.es
Perplexity	22.09446	9.69801	42.32203	42.18186

Using Bayes' Rule:

$$P(\text{Language Class} \mid \text{Sequence}) = \frac{P(LC, Seq)}{P(Seq)} = \frac{P(Seq \mid LC) * P(LM)}{P(Seq)} \quad (5.1)$$

As the sequence (or test text) is already given (prior probability) and the probabilities of different language classes are the same (1/3), the conditional probability of language class given sequence (or test text) is only determined by $P(Seq \mid LC)$. $P(Seq \mid LC)$ is the inverse of the perplexity (see Eq. 5.2), so if the conditional probability of the word sequence is higher, the perplexity will be lower (Jurafsky and Martin, 2009, p.14). Therefore, the language class with the minimal perplexity on the test set has the maximal probability.

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-2}, w_{i-1})}} \quad (5.2)$$

5.2 Whether the perplexity under English model is enough to determine

The perplexity only under English model is not enough to determine whether the test file is written in English, because we don't have a benchmark to judge whether the perplexity is small enough to regard it as English. It is necessary to calculate the perplexity under different language models. According to the comparison between these calculated perplexities, which language is more probable can be determined.

References

Jurafsky, D. and Martin, J. (2009) *Speech and Language Processing, Speech and Language Processing*. Prentice Hall. doi: 10.1007/s00134-010-1760-5.