CMP 670

Statistical Natural Language Processing (Spring 2019)
Homework 1

EROL ÖZKAN
N18245609
erolozkan@outlook.com

1. BASIC MODEL

a. Data Cleaning

Beginning and end of sentence speech tags (<s> and </s>) are added to all sentences. An example is shown in Table 1.

Table 1: Data Cleaning Example

I am Sam	⟶	<s> I am Sam </s>

b. N-Gram Language Model – Simple Test (Runner_First.py)

3 simple sentences are given in Table 2. From these sentences Unigram, Bigram and Trigram models are trained. Some of the statistics are shown in Table 3.

Table 2: Example Training Corpus

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Table 3: Some Examples from trained N-gram model

ONEGRAM PROBABILITIES		
ONEGRAM	**COUNT**	**PROB**
<s>	3	0.15
I	3	0.15
am	2	0.1
Sam	2	0.1
</s>	3	0.15
do	1	0.05
not	1	0.05
like	1	0.05
green	1	0.05
eggs	1	0.05
and	1	0.05
ham	1	0.05

ONEGRAM ADD ONE		
ONEGRAM	**COUNT**	**PROB**
<s>	3	0.125
I	3	0.125
am	2	0.09
Sam	2	0.09
</s>	3	0.12
do	1	0.06
not	1	0.06
like	1	0.06
green	1	0.06
eggs	1	0.06
and	1	0.06
ham	1	0.06

BIGRAM PROBABILITIES		
ONEGRAM	**COUNT**	**PROB**
('<s>', 'I')	2	0.66
('I', 'am')	2	0.66
('am', 'Sam')	1	0.5
('Sam', '</s>')	1	0.5
('<s>', 'Sam')	1	0.33
('Sam', 'I')	1	0.5
('I', 'am')	2	0.66
('am', '</s>')	1	0.5
('<s>', 'I')	2	0.66
('I', 'do')	1	0.33
('do', 'not')	1	1
('not', 'like')	1	1
('like', 'green')	1	1
('green', 'eggs')	1	1
('eggs', 'and')	1	1
('and', 'ham')	1	1
('ham', '</s>')	1	1

TRIGRAM PROBABILITIES		
ONEGRAM	COUNT	PROB
('<s>', 'I', 'am')	1	0.5
('I', 'am', 'Sam')	1	0.5
('am', 'Sam', '</s>')	1	1
('<s>', 'Sam', 'I')	1	1
('Sam', 'I', 'am')	1	1
('I', 'am', '</s>')	1	0.5
('<s>', 'I', 'do')	1	0.5
('I', 'do', 'not')	1	1
('do', 'not', 'like')	1	1
('not', 'like', 'green')	1	1
('like', 'green', 'eggs')	1	1
('green', 'eggs', 'and')	1	1
('eggs', 'and', 'ham')	1	1
('and', 'ham', '</s>')	1	1

c. Laplace Smoothing (output/1gram-add-one.txt)

For unigram model Laplace Smoothing is implemented. Table 3 shows Laplace Smoothing results. Here K value is 1. So it is simply Add-one Smoothing. Formula is shown below.

$$P(w_s)_{add-one} = \frac{C(w_s) + 1}{N + V}$$

d. Good-Turing Smoothing (output/good_turing_smooting*.txt)

For Bigram and Trigram model Good-Turing Smoothing is implemented. For this purpose frequencies of frequencies are calculated using below formula.

$$N_c = \sum_x count(x) = c$$

Using this c values c* values are calculated with below formula. These new c* values are then used to replace the maximum likelihood scores. Note that there exists some N_{c+1} values which are zero. Therefore, some c^*values for these values are also zero. This is because of data sparsity problem. There are some methods to estimate this zero values.

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

The purpose of Good-Turing Smoothing is to estimate the frequency of zero count events. To do this bigram zero occurrence and trigram zero occurrence probabilities are calculated using below formula. In this equation, N_1 is the number of counts seen once and N is the total number of counts seen in the training corpus. These values are assigned to test dataset words that never occurred in the training corpus.

$$P(N - gramwithzero - count) = \frac{N_1}{N}$$

e. N-Gram Language Model – Real Dataset Test - Perplexity of Test Dataset

Perplexity of test data is calculated using unigram, bigram and trigram models. Results are shown in Table 4.

Table 4: Perplexity Scores of Test Data

UNIGRAM PERPLEXITY	726.57	add-one smoothing used
BIGRAM PERPLEXITY	17.52	$\log_2(0)$ is assumed as zero. Some c^* values are zero.
TRIGRAM PERPLEXITY	1.97	$\log_2(0)$ is assumed as zero. Some c^* values are zero.

f. N-Gram Language Model – Generate Sentences, Calculate Perplexity

Foreach unigram, bigram and trigram model, 5 sentences are generated. Word are selected randomly from the corpus. That's why, for example, the probability of p("</s>"), p(".") or p("</s>" \ ".") are high. Perplexity of these sentences are calculated. Generated sentences and their perplexity score are given in Table 5. $\log_2(0)$ is assumed as zero not as -infinite.

Table 5: Randomly Genereated Sentences and Their Perplexity Score

<s> flatus banking 214,938 transmittable computations Sewer Dequindre concordant lucks brothels Billions accommodated veal abyss embroiled world-oriented blacking hypocrites dynamical confusion </s>	UNIGRAM PERPLEXITY 185054.64 UNIGRAM PERPLEXITY (Add-One Smoothing) 125766.33
<s> impressionistic zipper Westchester countries athletics Ideal yardstick Pm all-purpose obsession Women resourcefully True chafe every vibrated debated skin-perceptiveness Sea animosity </s>	
<s> qualitative Grovers dazzler strutted drip- phrasing uniforms France-Germany politicking hawk-faced Owing preponderantly forts cross-legged Curzon Bassi Harding Maeterlinck rove bracket </s>	
<s> Revolutionibus armament earnestly grinding recitative undertaken customarily hurley ambidextrous Division's 1500 oblong Continental Masters self-deceiving persiflage terrorists unwarrantable Comique churning </s>	
<s> widens reserve edgy unobtrusive 135 validly gingerly 20% felt joking 1966 Magi swift remnant Potemkin predispositions ceremony steaming Keeshond sceneries </s>	

<s> Speaking generally the ward-personnel and regard to Muller's would shoot at Fox reported Wagner and surprised . </s>	UNIGRAM PERPLEXITY 2866.33 UNIGRAM PERPLEXITY (Add-One Smoothing) 2720.43 BIGRAM PERPLEXITY 73.51
<s> set targets have cooling-heating units would curse : Forensic Pathology Seminar . </s>	
<s> Mayor approving) bodies such reference height . </s>	
<s> Jobs for Warwickshire in batting for irradiation of shame ? ? </s>	
<s> Cousin Joshua R. F. Gregorio that Hino decided he snarled . </s>	

<s> Reception into the action be unilateral or multilateral ? ? </s>	UNIGRAM PERPLEXITY 1271.61 UNIGRAM PERPLEXITY (Add-One Smoothing) 1258.56 BIGRAM PERPLEXITY 65.02 TRIGRAM PERPLEXITY 4.16
<s> Marty's heart skipped a piece up the rear gate in the work progresses the frame and moving parts become a composer </s>	
<s> Garth brought one in a curve C means a square with its fellows . </s>	
<s> Oso growled . </s>	
<s> Poetry for a final desperate plea from the Stalag commander . </s>	

- Unigram model selects random words without using any context information for the words. That's why, generated sentences have no meanings and the grammar is the worst.
- Bigram model selects the first word as "<s>", then randomly generates next words using bigram model (p(x\<s>)). It is better than the unigram model and sentences are grammatically more correct. Perplexity is also way lower.
- Trigram model selects first word as "<s>". It selects second word using bigram model, then generates next words using trigram model. Sentences are better than both unigram and bigram model considering both meaning and the grammar.

g. Error Analysis

- There are unigram, bigram and trigram word units that occur in test dataset, but does not occur in training dataset. The probability of these word unit's is 0. $\log_2(0)$=-infinite without any smoothing. In order to calculate Perplexity, I considered these values as zero! Only add-one smoothing has non-zero values. There were zero values in Good-Turing Smoothing also.
- As going from unigram to trigram, the probability of test dataset word unit does not occur in training dataset increases. That's why in these cases, some interpolation or back-off strategies are necessary.

h. Language Model Evaluation

Entropy: $\mathrm{H(X)} = \sum_x \mathrm{P(x)} log_2 \mathrm{P(x)}$ a measure of uncertainty/disorder

Cross-entropy: $H_m(w_1..w_n) = \frac{-1}{n} log_2 P_m(w_1..w_n)$ Model should have low uncertainty (entropy) about which word comes next. (Lower cross-entropy ⇒ model is better at predicting next word.)

Perplexity: $2^{\text{cross-entropy}}$ Lower perplexity is better for a language model

2. LINEAR INTERPOLATION

Simple interpolation formula is given below.

$$\begin{aligned}\hat{P}(w_n|w_{n-1}w_{n-2}) = & \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ & +\lambda_2 P(w_n|w_{n-1}) \\ & +\lambda_3 P(w_n)\end{aligned}$$

We are trying to find λs to maximize the probability of held-out data. This formulate is shown below.

$$\log P(w_1...w_n \mid M(\lambda_1...\lambda_k)) = \sum_i \log P_{M(\lambda_1...\lambda_k)}(w_i \mid w_{i-1})$$

One of the simplest technique is to build an interpolated language model using brute-force approach. Note that sum of the λs equals to one. Perplexity scores using different lambda values are shown in Table 6.

Table 6: Validation Dataset Perplexity Scores Using Different Lambda Values

Lambda Set	Validation Dataset Perp. Score
[0.5, 0.3, 0.2]	177.43
[0.4, 0.4, 0.2]	179.89
[0.6, 0.2, 0.2]	182.95
[0.4, 0.3, 0.3]	184.64
[0.3, 0.4, 0.3]	192.29
[0.4, 0.2, 0.4]	197.73
[0.3, 0.3, 0.4]	200.46
[0.8, 0.1, 0.1]	211
[0.2, 0.6, 0.2]	212.82
[0.2, 0.4, 0.4]	218.81
[0.2, 0.2, 0.6]	252.05
[0.1, 0.8, 0.1]	278.99
[0.1, 0.4, 0.5]	281.43
[0.1, 0.3, 0.6]	299.59
[0.1, 0.2, 0.7]	330.99
[0.1, 0.1, 0.8]	395.77
[0.05, 0.15, 0.8]	471.68
[0.05, 0.05, 0.9]	636.17

Test dataset best perplexity score using lambda set **[Unigram=0.5, Bigram=0.3, Trigram=0.2]** is 177.43.

3. DISCOUNTING(output/*discounted*.txt)

Below table shows the discounted probabilities with β=0.5 and saved probability mass.

BIGRAM DISCOUNTED AND SAVED PROBABILITY MASS

BIGRAM	COUNT	DISCOUNTED
('<s>', 'I')	2	0.5
('I', 'am')	2	0.5
('am', 'Sam')	1	0.25
('Sam', '</s>')	1	0.25
('<s>', 'Sam')	1	0.166666667
('Sam', 'I')	1	0.25
('I', 'am')	2	0.5
('am', '</s>')	1	0.25
('<s>', 'I')	2	0.5
('I', 'do')	1	0.166666667
('do', 'not')	1	0.5
('not', 'like')	1	0.5
('like', 'green')	1	0.5
('green', 'eggs')	1	0.5
('eggs', 'and')	1	0.5
('and', 'ham')	1	0.5
('ham', '</s>')	1	0.5

UNIGRAM	SUM	1-SUM
I	0.6666667	0.333333333
not	0.5	0.5
Sam	0.5	0.5
like	0.5	0.5
</s>	0	1
am	0.5	0.5
ham	0.5	0.5
do	0.5	0.5
and	0.5	0.5
green	0.5	0.5
eggs	0.5	0.5
<s>	0.6666667	0.333333333

TRIGRAM DISCOUNTED AND SAVED PROBABILITY MASS

TRIGRAM	COUNT	DISCOUNTED
('<s>', 'I', 'am')	1	0.25
('I', 'am', 'Sam')	1	0.25
('am', 'Sam', '</s>')	1	0.5
('<s>', 'Sam', 'I')	1	0.5
('Sam', 'I', 'am')	1	0.5
('I', 'am', '</s>')	1	0.25
('<s>', 'I', 'do')	1	0.25
('I', 'do', 'not')	1	0.5
('do', 'not', 'like')	1	0.5
('not', 'like', 'green')	1	0.5
('like', 'green', 'eggs')	1	0.5
('green', 'eggs', 'and')	1	0.5
('eggs', 'and', 'ham')	1	0.5
('and', 'ham', '</s>')	1	0.5

BIGRAM	SUM	1-SUM
('Sam', 'I')	0.5	0.5
('Sam', '</s>')	0	1
('like', 'green')	0.5	0.5
('<s>', 'I')	0.5	0.5
('and', 'ham')	0.5	0.5
('eggs', 'and')	0.5	0.5
('do', 'not')	0.5	0.5
('I', 'do')	0.5	0.5
('am', '</s>')	0	1
('green', 'eggs')	0.5	0.5
('I', 'am')	0.5	0.5
('not', 'like')	0.5	0.5
('<s>', 'Sam')	0.5	0.5
('am', 'Sam')	0.5	0.5
('ham', '</s>')	0	1