NLP Programming Tutorial 1 - Unigram Language Models

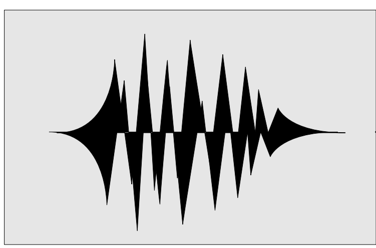
Graham Neubig
Nara Institute of Science and Technology (NAIST)

Language Model Basics

Why Language Models?

- We have an English speech recognition system, which answer is better?

Speech



W_1 = speech recognition system

W_2 = speech cognition system

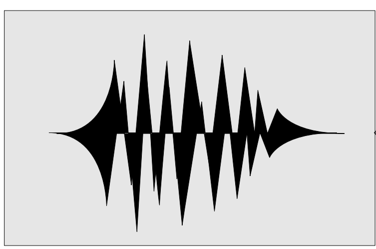
W_3 = speck podcast histamine

W_4 = スピーチ が 救出 ストン

Why Language Models?

- We have an English speech recognition system, which answer is better?

Speech



W_1 = speech recognition system

W_2 = speech cognition system

W_3 = speck podcast histamine

W_4 = スピーチ が 救出 ストン

- Language models tell us the answer!

Probabilistic Language Models

- Language models assign a probability to each sentence

W_1 = speech recognition system	$P(W_1) = 4.021 * 10^{-3}$
W_2 = speech cognition system	$P(W_2) = 8.932 * 10^{-4}$
W_3 = speck podcast histamine	$P(W_3) = 2.432 * 10^{-7}$
W_4 = スピーチ が 救出 ストン	$P(W_4) = 9.124 * 10^{-23}$

- We want $P(W_1) > P(W_2) > P(W_3) > P(W_4)$
  - (or $P(W_4) > P(W_1), P(W_2), P(W_3)$ for Japanese?)

Calculating Sentence Probabilities

- We want the probability of

$$W = \text{speech recognition system}$$

- Represent this mathematically as:

$$P(|W| = 3, w_1=\text{"speech"}, w_2=\text{"recognition"}, w_3=\text{"system"})$$

Calculating Sentence Probabilities

- We want the probability of

W = speech recognition system

- Represent this mathematically as (using chain rule):

$P(|W| = 3, w_1=\text{"speech"}, w_2=\text{"recognition"}, w_3=\text{"system"}) =$

$P(w_1=\text{"speech"} \mid w_0 = \text{"<s>"})$

$* P(w_2=\text{"recognition"} \mid w_0 = \text{"<s>"}, w_1=\text{"speech"})$

$* P(w_3=\text{"system"} \mid w_0 = \text{"<s>"}, w_1=\text{"speech"}, w_2=\text{"recognition"})$

$* P(w_4=\text{"</s>"} \mid w_0 = \text{"<s>"}, w_1=\text{"speech"}, w_2=\text{"recognition"}, w_3=\text{"system"})$

NOTE:
sentence start <s> and end </s> symbol

NOTE:
$P(w_0 = \text{<s>}) = 1$

Incremental Computation

- Previous equation can be written:

$$P(W)=\prod_{i=1}^{|W|+1} P(w_i|w_0 \ldots w_{i-1})$$

- How do we decide probability?

$$P(w_i|w_0 \ldots w_{i-1})$$

Maximum Likelihood Estimation

- Calculate word strings in corpus, take fraction

$$P(w_i | w_1 \ldots w_{i-1}) = \frac{c(w_1 \ldots w_i)}{c(w_1 \ldots w_{i-1})}$$

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

P(live | <s> i) = c(<s> i live)/c(<s> i) = 1 / 2 = 0.5

P(am | <s> i) = c(<s> i am)/c(<s> i) = 1 / 2 = 0.5

Problem With Full Estimation

- Weak when counts are low:

Training:

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

Test:

<s> i live in nara . </s>

↓

$P(\text{nara}|\text{<s> i live in}) = 0/1 = 0$

↓

$P(W=\text{<s> i live in nara . </s>}) = 0$

Unigram Model

- Do not use history:

$$P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i) = \frac{c(w_i)}{\sum_{\tilde{w}} c(\tilde{w})}$$

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

P(nara) = 1/20 = 0.05
P(i) = 2/20 = 0.1
P(</s>) = 3/20 = 0.15

P(W=i live in nara . </s>) =
0.1 * 0.05 * 0.1 * 0.05 * 0.15 * 0.15 = 5.625 * 10^{-7}

Be Careful of Integers!

- Divide two integers, you get an integer (rounded down)

```
first_int = 1
second_int = 2

print first_int/second_int
```

```
$ ./my-program.py
0
```

- Convert one integer to a float, and you will be OK

```
print float(first_int)/second_int
```

```
$ ./my-program.py
0.5
```

What about Unknown Words?!

- Simple ML estimation doesn't work

i live in osaka . </s> i am a graduate student . </s> my school is in nara . </s>	P(nara) = 1/20 = 0.05 P(i) = 2/20 = 0.1 P(kyoto) = 0/20 = 0

- Often, unknown words are ignored (ASR)
- Better way to solve
  - Save some probability for unknown words ($\lambda_{unk} = 1-\lambda_1$)
  - Guess total vocabulary size (N), including unknowns

$$P(w_i) = \lambda_1 P_{ML}(w_i) + (1-\lambda_1)\frac{1}{N}$$

Unknown Word Example

- Total vocabulary size: $N=10^6$
- Unknown word probability: $\lambda_{unk}=0.05$ ($\lambda_1 = 0.95$)

$$P(w_i)=\lambda_1 P_{ML}(w_i)+(1-\lambda_1)\frac{1}{N}$$

P(nara) = 0.95*0.05 + 0.05*(1/10^6) = 0.04750005

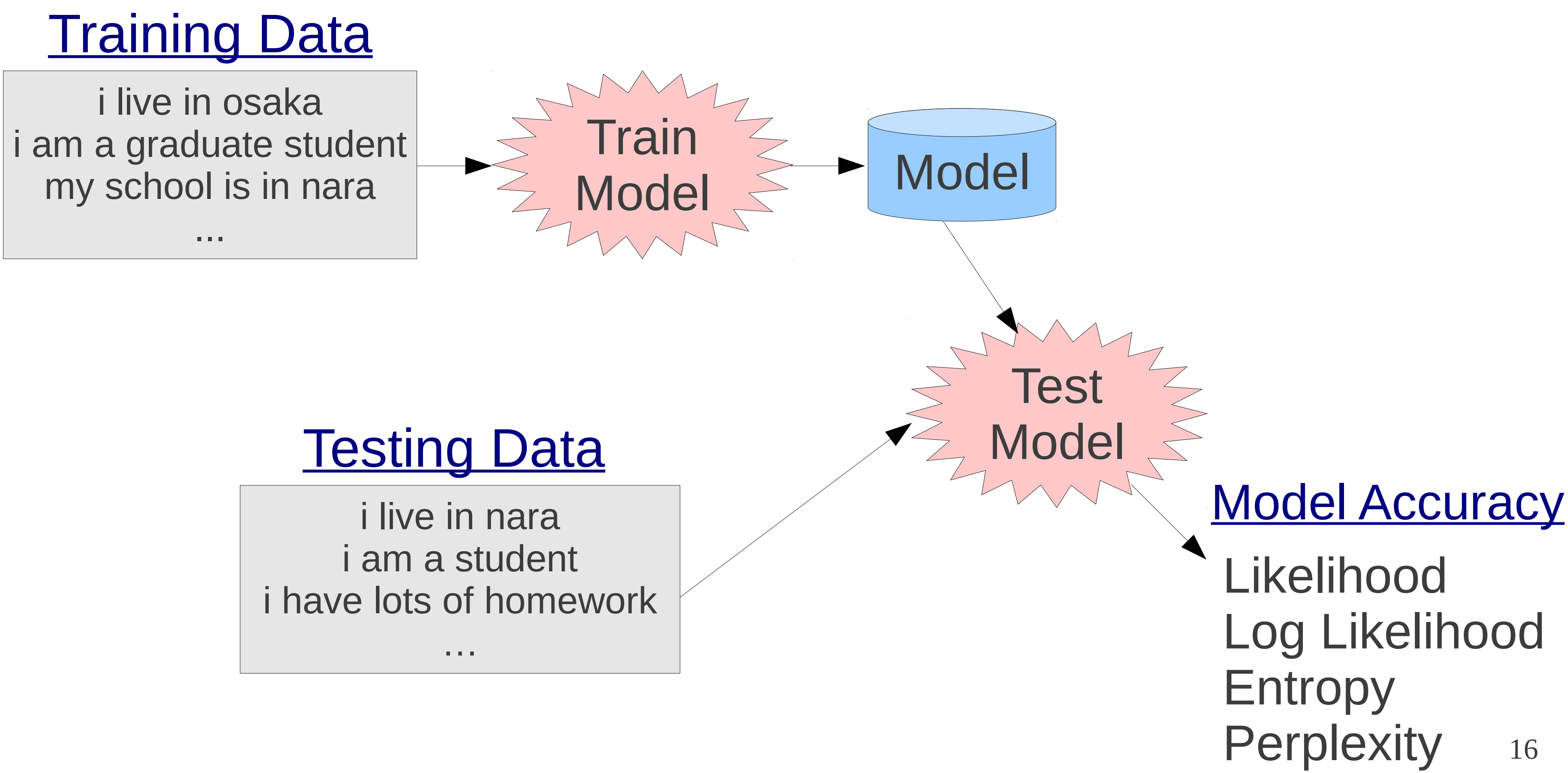
P(i) = 0.95*0.10 + 0.05*(1/10^6) = 0.09500005

P(kyoto) = 0.95*0.00 + 0.05*(1/10^6) = 0.00000005

Evaluating Language Models

Experimental Setup

- Use training and test sets



Likelihood

- Likelihood is the probability of some observed data (the test set W_{test}), given the model M

$$P(W_{test}|M)=\prod_{\mathbf{w}\in W_{test}} P(\mathbf{w}|M)$$

i live in nara	P(w=”i live in nara”|M) =	$2.52*10^{-21}$
		x
i am a student	P(w=”i am a student”|M) =	$3.48*10^{-19}$
		x
my classes are hard	P(w=”my classes are hard”|M) =	$2.15*10^{-34}$
		=
		$1.89*10^{-73}$

Log Likelihood

- Likelihood uses very small numbers=underflow
- Taking the log resolves this problem

$$\log P(W_{test}|M) = \sum_{\boldsymbol{w} \in W_{test}} \log P(\boldsymbol{w}|M)$$

i live in nara	log P(w=”i live in nara”|M) =	-20.58
		+
i am a student	log P(w=”i am a student”|M) =	-18.45
		+
my classes are hard	log P(w=”my classes are hard”|M) =	-33.67
		=
		-72.60

Calculating Logs

- Python's math package has a function for logs

```
import math

print math.log(100)       # ln(100)
print math.log(100, 10) # log10(100)
```

```
$ ./my-program.py
4.60517018599
2.0
```

Entropy

- Entropy H is average negative $\log_2$ likelihood per word

$$H(W_{test}|M)=\frac{1}{|W_{test}|}\sum_{\mathbf{w}\in W_{test}}-\log_2 P(\mathbf{w}|M)$$

i live in nara
i am a student
my classes are hard

$\log_2$ P(w=”i live in nara”|M)= 68.43
+
$\log_2$ P(w=”i am a student”|M)= 61.32
+
$\log_2$ P(w=”my classes are hard”|M)= 111.84

/

# of words= 12

=

20.13

* note, we can also count </s> in # of words (in which case it is 15)

Entropy and Compression

- Entropy H is also the average number of bits needed to encode information (Shannon's information theory)

$$H=\frac{1}{|W_{test}|}\sum_{w\in W_{wtest}} -\log_2 P(w|M)$$

a bird a cat a dog a </s>

Encoding

a	→	1
bird	→	000
cat	→	001
dog	→	010
</s>	→	011

P(w= “a”) = 0.5 $-\log_2 0.5 = 1$

P(w= “bird”) = 0.125 $-\log_2 0.125 = 3$

P(w= “cat”) = 0.125 $-\log_2 0.125 = 3$

P(w= “dog”) = 0.125 $-\log_2 0.125 = 3$

P(w= “</s>”) = 0.125 $-\log_2 0.125 = 3$

1000100110101011

Perplexity

- Equal to two to the power of per-word entropy

$$PPL = 2^H$$

- (Mainly because it makes more impressive numbers)
- For uniform distributions, equal to the size of vocabulary

$$V = 5 \quad H = -\log_2 \frac{1}{5} \quad PPL = 2^H = 2^{-\log_2 \frac{1}{5}} = 2^{\log_2 5} = 5$$

Coverage

- The percentage of known words in the corpus

a bird a cat a dog a </s>

"dog" is an unknown word

Coverage: 7/8 *

* often omit the sentence-final symbol → 6/7

Exercise

Exercise

- Write two programs
  - train-unigram: Creates a unigram model
  - test-unigram: Reads a unigram model and calculates entropy and coverage for the test set
- Test them test/01-train-input.txt test/01-test-input.txt
- Train the model on data/wiki-en-train.word
- Calculate entropy and coverage on data/wiki-en-test.word
- Report your scores next week

train-unigram Pseudo-Code

create a **map** *counts*
create a **variable** *total_count* = 0

for each *line* **in** the *training_file*
  split *line* into an array of *words*
  append “</s>” to the end of *words*
  for each *word* **in** *words*
    add 1 to *counts*[*word*]
    add 1 to *total_count*

open the *model_file* for writing
for each word, count **in** counts
  probability = *counts*[*word*]/*total_count*
  print *word*, *probability* **to** *model_file*

test-unigram Pseudo-Code

$\lambda_1 = 0.95$, $\lambda_{unk} = 1-\lambda_1$, $V = 1000000$, $W = 0$, $H = 0$

Load Model

create a **map** *probabilities*
for each *line* **in** *model_file*
  split *line* into *w* and *P*
  set *probabilities*[*w*] = *P*

Test and Print

for each *line* **in** *test_file*
  split *line* into an array of *words*
  append “</s>” to the end of *words*
  for each *w* **in** *words*
    add 1 to *W*
    set $P = \lambda_{unk} / V$
    if *probabilities*[*w*] exists
      set $P \mathrel{+}= \lambda_1 *$ probabilities[*w*]
    else
      add 1 to *unk*
    add $-\log_2 P$ to *H*

print “entropy = ”+*H/W*
print “coverage = ” + (W-unk)/W

Thank You!