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?

\[ W_1 = \text{speech recognition system} \]
\[ W_2 = \text{speech cognition system} \]
\[ W_3 = \text{speck podcast histamine} \]
\[ W_4 = \text{スピーチが救出ストン} \]
Why Language Models?

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

Speech

$W_1 = \text{speech recognition system}$

$W_2 = \text{speech cognition system}$

$W_3 = \text{speck podcast histamine}$

$W_4 = \text{スピーチ が 救出 ストン}$

• Language models tell us the answer!
Probabilistic Language Models

- Language models assign a probability to each sentence

\[ W_1 = \text{speech recognition system} \]
\[ W_2 = \text{speech cognition system} \]
\[ W_3 = \text{speck podcast histamine} \]
\[ W_4 = \text{スピーチ が 救出 ストン} \]

\[ P(W_1) = 4.021 \times 10^{-3} \]
\[ P(W_2) = 8.932 \times 10^{-4} \]
\[ P(W_3) = 2.432 \times 10^{-7} \]
\[ P(W_4) = 9.124 \times 10^{-23} \]

- 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

\[ W = \text{speech recognition system} \]

• Represent this mathematically as (using chain rule):

\[
P(|W| = 3, w_1 = "speech", w_2 = "recognition", w_3 = "system") =
\]

\[
P(w_1 = "speech" | w_0 = "<s>") 
* P(w_2 = "recognition" | w_0 = "<s>", w_1 = "speech") 
* P(w_3 = "system" | w_0 = "<s>", w_1 = "speech", w_2 = "recognition") 
* P(w_4 = "/s>" | w_0 = "<s>", w_1 = "speech", w_2 = "recognition", w_3 = "system")
\]

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

NOTE:
P(w_0 = <s>) = 1
Incremental Computation

• Previous equation can be written:

\[ P(W) = \prod_{i=1}^{\left|W\right|+1} P(w_i | w_0 \ldots w_{i-1}) \]

• How do we decide probability?

\[ P(w_i | w_0 \ldots w_{i-1}) \]
Maximum Likelihood Estimation

- Calculate word strings in corpus, take fraction

\[ P(w_i|w_1…w_{i-1}) = \frac{c(w_1…w_i)}{c(w_1…w_{i-1})} \]

\[ i \text{ live in osaka . } <\text{s}> \]
\[ i \text{ am a graduate student . } <\text{s}> \]
\[ \text{my school is in nara . } <\text{s}> \]

\[ P(\text{live} | <\text{s}> \text{ i}) = \frac{c(<\text{s}> \text{ i live})}{c(<\text{s}> \text{ i})} = \frac{1}{2} = 0.5 \]
\[ P(\text{am} | <\text{s}> \text{ i}) = \frac{c(<\text{s}> \text{ i am})}{c(<\text{s}> \text{ i})} = \frac{1}{2} = 0.5 \]
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>

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

  \[ P(W=<s> \ i \ live \ in \ nara . \ </s>) = 0 \]
Unigram Model

- Do not use history:

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

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

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

\[
P(\text{nara}) = 1/20 = 0.05
\]
\[
P(\text{i}) = 2/20 = 0.1
\]
\[
P(</s>) = 3/20 = 0.15
\]
Be Careful of Integers!

- Divide two integers, you get an integer (rounded down)
  ```python
  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.* \(</s>\) \(P(\text{nara}) = 1/20 = 0.05\)
  - *i am a graduate student.* \(</s>\) \(P(i) = 2/20 = 0.1\)
  - *my school is in nara.* \(</s>\) \(P(\text{kyoto}) = 0/20 = 0\)

- **Often, unknown words are ignored** (ASR)

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

\[
P(w_i) = \lambda_1 P_{ML}(w_i) + \left(1 - \lambda_1\right) \frac{1}{N}
\]
Unknown Word Example

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

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

- $P(\text{nara}) = 0.95 \times 0.05 + 0.05 \times \frac{1}{10^6} = 0.04750005$
- $P(i) = 0.95 \times 0.10 + 0.05 \times \frac{1}{10^6} = 0.09500005$
- $P(\text{kyoto}) = 0.95 \times 0.00 + 0.05 \times \frac{1}{10^6} = 0.00000005$
Evaluating Language Models
Experimental Setup

- Use training and test sets

**Training Data**
- I live in Osaka
- I am a graduate student
- My school is in Nara
  ...

**Testing Data**
- I live in Nara
- I am a student
- I have lots of homework
  ...

**Model Accuracy**
- Likelihood
- Log Likelihood
- Entropy
- Perplexity
## Likelihood

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

\[
P(W_{test} | M) = \prod_{w \in W_{test}} P(w | M)
\]

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Likelihood Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>i live in nara</td>
<td>$2.52 \times 10^{-21}$</td>
</tr>
<tr>
<td>i am a student</td>
<td>$3.48 \times 10^{-19}$</td>
</tr>
<tr>
<td>my classes are hard</td>
<td>$2.15 \times 10^{-34}$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$1.89 \times 10^{-73}$</strong></td>
</tr>
</tbody>
</table>
Log Likelihood

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

\[
\log P(W_{test} | M) = \sum_{w \in W_{test}} \log P(w | M)
\]

\[
\begin{align*}
\log P(w=“i live in nara”|M) &= -20.58 \\
\log P(w=“i am a student”|M) &= -18.45 \\
\log P(w=“my classes are hard”|M) &= -33.67 \\
&= -72.60
\end{align*}
\]
Calculating Logs

- Python's math package has a function for logs

```python
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**

\[
H(W_{test} \mid M) = \frac{1}{|W_{test}|} \sum_{w \in W_{test}} - \log_2 P(w \mid M)
\]

- \(i\) live in nara
  \[\log_2 P(w=\text{“i live in nara”} \mid M) = 68.43\]
- \(i\) am a student
  \[\log_2 P(w=\text{“i am a student”} \mid M) = 61.32\]
- my classes are hard
  \[\log_2 P(w=\text{“my classes are hard”} \mid M) = 111.84\]

- \# of words = \(\frac{68.43 + 61.32 + 111.84}{12} = 20.13\)

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

\[ \lambda_1 = 0.95, \quad \lambda_{\text{unk}} = 1 - \lambda_1, \quad V = 1000000, \quad W = 0, \quad H = 0 \]

**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_{\text{unk}} / V`  
if `probabilities[w]` exists  
set `P += \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!