Language Model

Introduction to N-grams
Probabilistic Language Model

- **Goal**: assign a probability to a sentence

- **Application**:
  - Machine Translation
    \[ P(\text{high winds tonight}) > P(\text{large winds tonight}) \]
  - Spelling Correction
    \[ P(\text{about 15 minutes from}) > P(\text{about 15 minuets from}) \]
  - Speech Recognition
    \[ P(\text{I saw a van}) > P(\text{eyes awe of an}) \]
How to Compute Language Modeling

- For a given sentence $s = (w_1, \ldots, w_n)$
- Words $w_t$ are discrete
- Sequence length $n$ is random
- **Goal:** probability of an upcoming word

$$p(w_t|w_1, \ldots, w_{t-1})$$
**Chain Rule of Probability**

\[ p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_n|w_1, \ldots, w_{n-1}) \]

**Example**

\[ p(\text{its water is so transparent}) = p(\text{its}) \times p(\text{water}|\text{its}) \times p(\text{is}|\text{its water}) \times p(\text{so}|\text{its water is}) \times p(\text{transparent}|\text{its water is so}) \]

**The probability of a sentence can be obtained via the Chain Rule of Probability**
Evaluation of Language Model

real
frequently observed

ungrammatical
rarely observed

higher probability

lower probability
Evaluation via Perplexity

$$PP(s) = (p((w_1, \ldots, w_n)))^{\frac{1}{n}}$$

$$= \left( \prod_{t=1}^{n} \frac{1}{p(w_t|w_1, \ldots, w_{t-1})} \right)^{\frac{1}{n}}$$

The best language model is the one that best predicts an unseen sentence
Markov Assumption

- Too many possible combinations of \( w_1, \ldots, w_t \)
- Impossible to infer \( p(w_t|w_1, \ldots, w_{t-1}) \)
- Approximation:
  - Unigram
    \[
p(w_t|w_1, \ldots, w_{t-1}) \approx P_{W_2|W_1}(w_t|w_{t-1})
    \]
  - Bigram
    \[
p(w_t|w_1, \ldots, w_{t-1}) \approx P_{W_3|W_1,W_2}(w_t|w_{t-1}, w_{t-2})
    \]
  - Higher order approximation…
Parameter Estimation

• In a bigram language model, parameters are

\[ P_{W_2|W_1}(w_2|w_1), \quad \forall w_1, w_2 \in \text{vocabulary} \]

• Straight forward: the ML estimator

\[ P_{W_2|W_1}(w_2|w_1) = \frac{\text{count}(w_2, w_1)}{\text{count}(w_1)} \]

\text{count}(\cdot, \cdot) \text{ is the number of cooccurrence of a word pair.}
\text{count}(\cdot) \text{ is the number occurrence of a word.}
An Example

Training corpus:

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

Induced parameters:

\[
\begin{align*}
P_{W_2|W_1}(I|\langle s\rangle) &= \frac{2}{3} \\
P_{W_2|W_1}(Sam|\langle s\rangle) &= \frac{1}{3} \\
P_{W_2|W_1}(am|I) &= \frac{2}{3} \\

P_{W_2|W_1}(\langle s\rangle|Sam) &= \frac{1}{2} \\
P_{W_2|W_1}(Sam|am) &= \frac{1}{2} \\
P_{W_2|W_1}(do|I) &= \frac{1}{3}
\end{align*}
\]
Online Resource

All Our N-gram are Belong to You
Thursday, August 03, 2006

File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens: 1,024,908,267,229
Number of sentences: 95,119,665,584
Number of unigrams: 13,588,391
Number of bigrams: 314,843,401
Number of trigrams: 977,069,902
Number of fourgrams: 1,313,818,354
Number of fivegrams: 1,176,470,663

https://research.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html
Shakespeare as Corpus

- Corpus has 884,647 tokens
- Vocabulary size $V=29,066$
- Shakespeare produced 300,000 bigrams out of $V^2 = 844$ millions possible bigrams
- 99.96% of the bigram tables are zero (why?)
Practical Issues

- Sparsity
- Things that don’t occur in the training corpus, but occur in real life.

<table>
<thead>
<tr>
<th>Training Corpus</th>
<th>Real Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>... denied the allegations</td>
<td>... denied the offer</td>
</tr>
<tr>
<td>... denied the reports</td>
<td></td>
</tr>
<tr>
<td>... denied the claims</td>
<td></td>
</tr>
<tr>
<td>... denied the request</td>
<td></td>
</tr>
</tbody>
</table>

\[ P(\text{offer}|\text{denied the}) = 0 \]
Smoothing the “Zero”s

When we have sparse statistics, steal probability mass to generalize better.

Good Turing Smoothing

• Consider a scenario, one is fishing and caught 18 fishes

  - Carp x10
  - perch x3
  - whitefish x2
  - trout x1
  - salmon x1
  - eel x1

• How likely is that next species is trout?
• Assume there are new species, how likely is it that next species is new?
• Now how likely is that next species is trout?
**Leave-One-Validation**

- Take each of one of the fish out in turn
- 18 training sets of size 17, held-out of size 1
- The fraction of held-out fishes are unseen in the training?
  - \( \frac{\text{# of fishes occur once}}{18} = \frac{3}{18} \)
- The fraction of held-out fishes are seen \( k \) times in training?
  - \( \frac{(\text{# of fishes occur (k+1) times})^{(k+1)}}{18} \)

Use things-we-saw-(k+1)-times to estimate things-we-saw-k-times
Good Turing Smoothing

- Consider a scenario, one is fishing and caught 18 fishes:
  - Carp: x10
  - perch: x3
  - whitefish: x2
  - trout: x1
  - salmon: x1
  - eel: x1

- How likely is that next species is trout? \( \frac{1}{18} \)
- Assume there are new species, how likely is it that next species is new? \( \frac{3}{18} \)
- Now how likely is that next species is trout? \( \frac{1}{18} \times \frac{2}{3} = \frac{1}{27} \)
Good Turing Smoothing

- $N_i$ is the number of words that occur $i$ times
- $N = \sum_i i N_i$ is the number of samples
- A word (with occurrence $i$) should occur with probability

$$\frac{(i + 1)N_{i+1}}{N_i N}$$
Good Turing

NIPS 2017
Absolute Discounting Smoothing

Steal probability mass to unseen samples

\[ P_{W_2|W_1}^{(AD)}(w_2|w_1) = \frac{\text{count}(w_2, w_1) - d}{\text{count}(w_1)} + \lambda(w_1)p_W(w_2) \]
Absolute Discounting

\[ P_{W_2|W_1}^{(AD)}(w_2|w_1) = \frac{\text{count}(w_2, w_1) - d}{\text{count}(w_1)} + \lambda(w_1)p_W(w_2) \]

• Some word (e.g. Fransisco) always occurs with other words (e.g. San), but this contributes to the unigram distribution.

• Principle of probability

\[ \sum_{w_1} P_{W_2|W_1}^{(AD)}(w_2|w_1)p_W(w_1) \neq p_W(w_2) \]

• Choice of continuation distribution!
Kneser-Ney Smoothing

\[ P_{W_2|W_1}^{(KN)}(w_2|w_1) = \frac{\text{count}(w_2, w_1) - d}{\text{count}(w_1)} + \lambda(w_1)P_{\text{cont}}(w_2) \]

The normalized discount; the probability mass we’ve discounted

\[ \lambda(w) = \frac{d}{c(w)} |\{w' : \text{count}(w, w') > 0\}| \]

\[ P_{\text{cont}}(w) = \frac{|\{w' : \text{count}(w, w') > 0\}|}{\sum_{w' \in \text{vocabulary}} |\{w' : \text{count}(w, w') > 0\}|} \]

Marginal constraint gives an only solution to the interpolated distribution.
Trigram and More

• Recursive formulation of KN smoothing

\[ P^{(KN)}(w_n|w_1, w_2, \ldots, w_{n-1}) = \frac{\max\{\text{count}^{(KN)}(w_1, w_2, \ldots, w_n) - d, 0\}}{\text{count}^{(KN)}(w_1, w_2, \ldots, w_{n-1})} \]

\[ + \lambda(w_1, \ldots, w_{n-1})P^{(KN)}(w_n|w_2, \ldots, w_{n-1}) \]

\[ \text{count}^{(KN)}(\cdot) = \begin{cases} 
\# \text{ of occurrence of } \cdot & \text{for the highest order} \\
\# \text{ of unique word types for } \cdot & \text{for lower order} 
\end{cases} \]
Bayesian Interpretation of KN Smoothing

https://www.stats.ox.ac.uk/~teh/research/compling/hpylum.pdf
Smoothing via Context Tree

• Parameterized the conditional distribution of Markov model

\[ P(w|u = (w_1, \ldots, w_{n-1})) = G_u(w) \]

• \( G_u \) is a probability vector associated with context \( u \)

- Smoothing equals the dependency between \( G_u \) and \( G_{\text{parent}(u)} \)
Chinese Restaurant Process

- Chinese Restaurant Process $\text{CRP}(d, \theta, G_0)$ is a distribution over distributions over a probability space.
- CRP is defined over draws from $G_1 = \text{CRP}(d, \theta, G_0)$.
- Sample space is all (unbounded) tables in a restaurant.
- A sequence of customers visit this restaurant, and randomly pick a table to sit.
  - The first customer sits at the first table.
  - The $i$-th customer chooses his seat after observing the seating arrangement.
  - Samples from CRP is equivalent to the seating arrangements of infinite customers.
Sample Generation in CRP

- Let $x_i$ be the table the i-th customer sits
- Let $c_k$ be the number of customer sitting at table $k$
- Let $t.$ be the number of occupied tables
- W.p. $\frac{\theta + dt.}{\theta + (i - 1)}$ this customer sits from $G_0$; otherwise, he chooses his seat based on current seating arrangement.

$$x_i | x_1, \ldots, x_{i-1}, \text{seating arrangement} \sim \sum_{k=1}^{t.} \frac{c_k - d}{\theta + (i - 1)} \delta_{\text{table}_k} + \frac{\theta + dt.}{\theta + (i - 1)} G_0$$

- Interpolated weight
- Absolute discounting
- Continuation probability
CRP in Language Model

- CRP is defined over draws from
  - N-gram distribution is built on (N-1)-gram distribution

$$G_u(w) \sim \text{CRP}(d_u, \theta_u, G_{\text{parent}(u)}(w))$$
Neural Network in Language Model

- N-gram models can be thought as classifications

\[
\begin{align*}
\text{history} & \quad (w_1, \ldots, w_{n-1}) \\
\rightarrow & \quad \text{predict} \\
\rightarrow & \quad \text{current word} \\
& \quad w_n
\end{align*}
\]

- Discriminative model via neural networks
FNN in Language Model

- N-gram models are inherently classification problem:
  - Given the context, predict the next word (one of V classes)

Curse of N-gram Models

- Language has **long-distance** dependencies

  “The computer which I had just put into the machine room on the fifth floor crashed.”

- Modeling via **RNN**