Language Modeling

Introduction to N-grams

Many Slides are adapted from slides by Dan Jurafsky
Probabilistic Language Models

• Today’s goal: assign a probability to a sentence
  • Machine Translation:
    • $P(\text{high winds tonite}) > P(\text{large winds tonite})$
  • Spell Correction
    • The office is about fifteen minuets from my house
      • $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
  • Speech Recognition
    • $P(\text{I saw a van}) >> P(\text{eyes awe of an})$
  • + Summarization, question-answering, etc., etc.!!
Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:
  \[ P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n) \]
- Related task: probability of an upcoming word:
  \[ P(w_5 | w_1, w_2, w_3, w_4) \]
- A model that computes either of these:
  \[ P(W) \quad \text{or} \quad P(w_n | w_1, w_2 \ldots w_{n-1}) \]
  is called a language model.
- Better: the grammar \quad But language model or LM is standard
How to compute $P(W)$

• How to compute this joint probability:

  • $P(\text{its, water, is, so, transparent, that})$

• Intuition: let’s rely on the Chain Rule of Probability
Reminder: The Chain Rule

• With 4 variables:
  \[ P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C) \]

• The Chain Rule in General
  \[ P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1}) \]
The Chain Rule applied to compute joint probability of words in sentence

\[
P(w_1 w_2 \ldots w_n) = \prod_i P(w_i | w_1 w_2 \ldots w_{i-1})
\]

\[
P(\text{“its water is so transparent”}) =
\]
\[
P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})
\]
\[
\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})
\]
How to estimate these probabilities

• Could we just count and divide?

\[
P(\text{its water is so transparent that}) = \frac{\text{Count}(\text{its water is so transparent that})}{\text{Count}(\text{its water is so transparent that})}
\]

• No! Too many possible sentences!

• We’ll never see enough data for estimating these
Markov Assumption

\[
P(w_1w_2\ldots w_n) \approx \prod_i P(w_i | w_{i-k}\ldots w_{i-1})
\]

• In other words, we approximate each component in the product

\[
P(w_i | w_1w_2\ldots w_{i-1}) \approx P(w_i | w_{i-k}\ldots w_{i-1})
\]
Markov Assumption

- Simplifying assumption:

\[ P(\text{the | its water is so transparent that}) \approx P(\text{the | that}) \]

- Or maybe

\[ P(\text{the | its water is so transparent that}) \approx P(\text{the | transparent that}) \]
Simplest case: Unigram model

\[ P(w_1w_2\ldots w_n) \approx \prod_{i} P(w_i) \]

Some automatically generated sentences from a unigram model

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
Bigram model

Condition on the previous word:

\[ P(w_i | w_1 w_2 \ldots w_{i-1}) \approx P(w_i | w_{i-1}) \]

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached this, would, be, a, record, november
N-gram models

• We can extend to trigrams, 4-grams, 5-grams
• In general this is an insufficient model of language
  • because language has long-distance dependencies:
    “The computer which I had just put into the machine room on the fifth floor crashed.”
• But we can often get away with N-gram models

Still, Most words depend on their previous few words
Introduction to N-grams

Language Modeling

Introduction to N-grams
Language Modeling

Estimating N-gram Probabilities
Estimating bigram probabilities

- The Maximum Likelihood Estimate

\[ P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} \]

\[ P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \]
An example

\[ P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \]

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

\[
P(I \mid <s>) = \frac{2}{3} = 0.67 \quad P(Sam \mid <s>) = \frac{1}{3} = 0.33 \quad P(am \mid I) = \frac{2}{3} = 0.67
\]
\[
P(<s> \mid Sam) = \frac{1}{2} = 0.5 \quad P(Sam \mid am) = \frac{1}{2} = 0.5 \quad P(do \mid I) = \frac{1}{3} = 0.33
\]
More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i’m looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day
## Raw bigram counts

- Out of 9222 sentences

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Raw bigram probabilities

- Normalize by unigrams:

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

- Result:

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0</td>
<td>0.0054</td>
<td>0.0079</td>
</tr>
<tr>
<td>to</td>
<td>0.00083</td>
<td>0</td>
<td>0.0017</td>
<td>0.28</td>
<td>0.00083</td>
<td>0.021</td>
<td>0.0025</td>
<td>0.0011</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>0.0027</td>
<td>0</td>
<td>0</td>
<td>0.0027</td>
<td>0.52</td>
<td>0.056</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0063</td>
<td>0</td>
<td>0.014</td>
<td>0</td>
<td>0.00092</td>
<td>0.0037</td>
<td>0.0063</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0.014</td>
<td>0</td>
<td>0.014</td>
<td>0</td>
<td>0</td>
<td>0.0029</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0059</td>
<td>0</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>0.0036</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bigram estimates of sentence probabilities

\[
P(<s> I \text{ want english food } </s>) = \\
P(I|<s>) \\
\times P(\text{want}|I) \\
\times P(\text{english}|\text{want}) \\
\times P(\text{food}|\text{english}) \\
\times P(</s>|\text{food}) \\
= .000031
\]
What kinds of knowledge?

- $P(\text{english} \mid \text{want}) = .0011$
- $P(\text{chinese} \mid \text{want}) = .0065$
- $P(\text{to} \mid \text{want}) = .66$
- $P(\text{eat} \mid \text{to}) = .28$
- $P(\text{food} \mid \text{to}) = 0$
- $P(\text{want} \mid \text{spend}) = 0$
- $P (i \mid <s>) = .25$
Practical Issues

• We do everything in log space
  • Avoid underflow
  • (also adding is faster than multiplying)

\[
\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4
\]
Language Modeling Toolkits

• SRILM
  • http://www.speech.sri.com/projects/srilm/
All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, ...

That’s why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.
Language Modeling

Estimating N-gram Probabilities
Language Modeling

Evaluation and Perplexity
Evaluation: How good is our model?

• Does our language model prefer good sentences to bad ones?
  • Assign higher probability to “real” or “frequently observed” sentences
    • Than “ungrammatical” or “rarely observed” sentences?

• We train parameters of our model on a training set.

• We test the model’s performance on data we haven’t seen.
  • A test set is an unseen dataset that is different from our training set, totally unused.
  • An evaluation metric tells us how well our model does on the test set.
Extrinsic evaluation of N-gram models

• Best evaluation for comparing models A and B
  • Put each model in a task
    • spelling corrector, speech recognizer, MT system
  • Run the task, get an accuracy for A and for B
    • How many misspelled words corrected properly
    • How many words translated correctly
  • Compare accuracy for A and B
Difficulty of extrinsic (in-vivo) evaluation of N-gram models

• Extrinsic evaluation
  • Time-consuming; can take days or weeks
• So
  • Sometimes use intrinsic evaluation: perplexity
Intuition of Perplexity

• The Shannon Game:
  • How well can we predict the next word?
    
    I always order pizza with cheese and ____
    
    The 33\textsuperscript{rd} President of the US was ____
    
    I saw a ____
  • Unigrams are terrible at this game. (Why?)

• A better model of a text
  • is one which assigns a higher probability to the word that actually occurs

{mushrooms 0.1
pepperoni 0.1
anchovies 0.01
....
fried rice 0.0001
....
and 1e-100}
Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest $P(\text{sentence})$

Perplexity is the inverse probability of the test set, normalized by the number of words:

$PP(W) = P(w_1 w_2 ... w_N)^{1/N}$

$= \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$

Chain rule:

For bigrams:

$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 ... w_{i-1})}}$

Minimizing perplexity is the same as maximizing probability
Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

<table>
<thead>
<tr>
<th>N-gram Order</th>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perplexity</td>
<td>962</td>
<td>170</td>
<td>109</td>
</tr>
</tbody>
</table>
Language Modeling

Evaluation and Perplexity
Language Modeling

Generalization and zeros
The Shannon Visualization Method

- Choose a random bigram \(<s>, w\) according to its probability
- Now choose a random bigram \((w, x)\) according to its probability
- And so on until we choose \(</s>\)
- Then string the words together

\(<s>\)

I

I want

want to

to eat

eat Chinese

Chinese food

food

</s>

I want to eat Chinese food
Approximating Shakespeare

Unigram
To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram
What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram
Sweet prince, Falstaff shall die. Harry of Monmouth’s grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.

Quadrigram
King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, ’tis a noble Lepidus.
Shakespeare as corpus

• N=884,647 tokens, V=29,066

• Shakespeare produced 300,000 bigram types out of $V^2 = 844$ million possible bigrams.
  • So 99.96% of the possible bigrams were never seen (have zero entries in the table)

• Quadrigrams worse: What's coming out looks like Shakespeare because it *is* Shakespeare
The wall street journal is not shakespeare (no offense)

Unigram
Months the my and issue of year foreign new exchange’s september were recession ex-
change new endorsed a acquire to six executives

Bigram
Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor
would seem to complete the major central planners one point five percent of U. S. E. has
already old M. X. corporation of living on information such as more frequently fishing to
keep her

Trigram
They also point to ninety nine point six billion dollars from two hundred four oh six three
percent of the rates of interest stores as Mexico and Brazil on market conditions
The perils of overfitting

• N-grams only work well for word prediction if the test corpus looks like the training corpus
  • In real life, it often doesn’t
• We need to train robust models that generalize!
  • One kind of generalization: Zeros!
    • Things that don’t ever occur in the training set
      • But occur in the test set
Zeros

• Training set:
  ... denied the allegations
  ... denied the reports
  ... denied the claims
  ... denied the request

• Test set
  ... denied the offer
  ... denied the loan

\[ P(“offer” \mid \text{denied the}) = 0 \]
Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can’t divide by 0)!
Language Modeling

Generalization and zeros
Language Modeling

Smoothing: Add-one (Laplace) smoothing
Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

MLE estimate:

\[
P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}
\]

Add-1 estimate:

\[
P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V + 1}
\]
<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>6</td>
<td>828</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>1</td>
<td>609</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>687</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>eat</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>17</td>
<td>3</td>
<td>43</td>
<td>1</td>
</tr>
<tr>
<td>chinese</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>83</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>lunch</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>spend</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Laplace-smoothed bigrams

\[ P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V} + 1 \]

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0.0015</td>
<td>0.21</td>
<td>0.00025</td>
<td>0.0025</td>
<td>0.00025</td>
<td>0.00025</td>
<td>0.00025</td>
<td>0.00075</td>
</tr>
<tr>
<td>want</td>
<td>0.0013</td>
<td>0.00042</td>
<td>0.26</td>
<td>0.00084</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0025</td>
<td>0.00084</td>
</tr>
<tr>
<td>to</td>
<td>0.00078</td>
<td>0.00026</td>
<td>0.0013</td>
<td>0.18</td>
<td>0.00078</td>
<td>0.00026</td>
<td>0.0018</td>
<td>0.055</td>
</tr>
<tr>
<td>eat</td>
<td>0.00046</td>
<td>0.00046</td>
<td>0.0014</td>
<td>0.00046</td>
<td>0.0078</td>
<td>0.0014</td>
<td>0.02</td>
<td>0.00046</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0012</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.00062</td>
<td>0.052</td>
<td>0.0012</td>
<td>0.00062</td>
</tr>
<tr>
<td>food</td>
<td>0.0063</td>
<td>0.00039</td>
<td>0.0063</td>
<td>0.00039</td>
<td>0.00079</td>
<td>0.002</td>
<td>0.00039</td>
<td>0.00039</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0017</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.00056</td>
<td>0.0011</td>
<td>0.00056</td>
<td>0.00056</td>
</tr>
<tr>
<td>spend</td>
<td>0.0012</td>
<td>0.00058</td>
<td>0.0012</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
<td>0.00058</td>
</tr>
</tbody>
</table>
Reconstituted counts

\[
c^* (w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V + 1}
\]

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>3.8</td>
<td>527</td>
<td>0.64</td>
<td>6.4</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>1.9</td>
</tr>
<tr>
<td>want</td>
<td>1.2</td>
<td>0.39</td>
<td>238</td>
<td>0.78</td>
<td>2.7</td>
<td>2.7</td>
<td>2.3</td>
<td>0.78</td>
</tr>
<tr>
<td>to</td>
<td>1.9</td>
<td>0.63</td>
<td>3.1</td>
<td>430</td>
<td>1.9</td>
<td>0.63</td>
<td>4.4</td>
<td>133</td>
</tr>
<tr>
<td>eat</td>
<td>0.34</td>
<td>0.34</td>
<td>1</td>
<td>0.34</td>
<td>5.8</td>
<td>1</td>
<td>15</td>
<td>0.34</td>
</tr>
<tr>
<td>chinese</td>
<td>0.2</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>8.2</td>
<td>0.2</td>
<td>0.098</td>
</tr>
<tr>
<td>food</td>
<td>6.9</td>
<td>0.43</td>
<td>6.9</td>
<td>0.43</td>
<td>0.86</td>
<td>2.2</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>lunch</td>
<td>0.57</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.38</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>spend</td>
<td>0.32</td>
<td>0.16</td>
<td>0.32</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
</tr>
</tbody>
</table>
## Compare with raw bigram counts

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>3.8</td>
<td>527</td>
<td>0.64</td>
<td>6.4</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>1.9</td>
</tr>
<tr>
<td>want</td>
<td>1.2</td>
<td>0.39</td>
<td>238</td>
<td>0.78</td>
<td>2.7</td>
<td>2.7</td>
<td>2.3</td>
<td>0.78</td>
</tr>
<tr>
<td>to</td>
<td>1.9</td>
<td>0.63</td>
<td>3.1</td>
<td>430</td>
<td>1.9</td>
<td>0.63</td>
<td>4.4</td>
<td>133</td>
</tr>
<tr>
<td>eat</td>
<td>0.34</td>
<td>0.34</td>
<td>1</td>
<td>0.34</td>
<td>5.8</td>
<td>1</td>
<td>15</td>
<td>0.34</td>
</tr>
<tr>
<td>chinese</td>
<td>0.2</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>8.2</td>
<td>0.2</td>
<td>0.098</td>
</tr>
<tr>
<td>food</td>
<td>6.9</td>
<td>0.43</td>
<td>6.9</td>
<td>0.43</td>
<td>0.86</td>
<td>2.2</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>lunch</td>
<td>0.57</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.38</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>spend</td>
<td>0.32</td>
<td>0.16</td>
<td>0.32</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Add-1 estimation is a blunt instrument

- So add-1 isn’t used for N-grams:
  - We’ll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn’t so huge.
Language Modeling

Smoothing: Add-one (Laplace) smoothing
Language Modeling

Interpolation, Backoff
Backoff and Interpolation

• Sometimes it helps to use less context
  • Condition on less context for contexts you haven’t learned much about

• Backoff:
  • use trigram if you have good evidence,
  • otherwise bigram, otherwise unigram

• Interpolation:
  • mix unigram, bigram, trigram

• Interpolation works better
Linear Interpolation

- Simple interpolation

\[
\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\
+ \lambda_2 P(w_n|w_{n-1}) \\
+ \lambda_3 P(w_n)
\]

\[\sum_{i} \lambda_i = 1\]

- Lambdas conditional on context:

\[
\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 (w_{n-1}^{n-1}) P(w_n|w_{n-2}w_{n-1}) \\
+ \lambda_2 (w_{n-2}^{n-1}) P(w_n|w_{n-1}) \\
+ \lambda_3 (w_{n-2}^{n-1}) P(w_n)
\]
N-gram Smoothing Summary

• Add-1 smoothing:
  • OK for text categorization, not for language modeling
• Backoff and Interpolation work better
• The most commonly used method:
  • Extended Interpolated Kneser-Ney
Language Modeling

Interpolation, Backoff
Language Modeling

Advanced: Kneser-Ney Smoothing
Advanced smoothing algorithms

• Intuition used by many smoothing algorithms
  • Good-Turing
  • Kneser-Ney

• Use the count of things we’ve seen
  • to help estimate the count of things we’ve never seen
Kneser-Ney Smoothing I (smart backoff)

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can’t see without my reading*?
  - “Francisco” is more common than “glasses”
  - ... but “Francisco” always follows “San”
- Instead of $P(w)$: “How likely is $w$”
- $P_{\text{continuation}}(w)$: “How likely is $w$ to appear as a novel continuation?”
  - For each word, count the number of unique bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

\[
P_{\text{CONTINUATION}}(w) \propto |\{ w_{i-1} : c(w_{i-1}, w) > 0 \}|
\]
Kneser-Ney Smoothing II

- How many times does $w$ appear as a novel continuation:

$$P_{\text{CONTINUATION}}(w) \propto \left\{|w_{i-1} : c(w_{i-1}, w) > 0|\right\}$$

- Normalized by the total number of word bigram types

$$P_{\text{CONTINUATION}}(w) = \frac{\left\{|w_{i-1} : c(w_{i-1}, w) > 0|\right\}}{\left\{|(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0|\right\}}$$
Kneser-Ney Smoothing III

\[ P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{CONTINUATION}(w_i) \]

\( \lambda \) is a normalizing constant; the probability mass we’ve discounted

\[ \lambda(w_{i-1}) = \frac{d}{c(w_{i-1})}\left|\{w : c(w_{i-1}, w) > 0\}\right| \]

The number of word types that can follow \( w_{i-1} \)

= # of word types we discounted

= # of times we applied normalized discount
Language Modeling

Advanced:
Kneser-Ney Smoothing