CS 333: Natural Language Processing

Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 2 due on Thursday
- Caroline has help hours today
- Ashley has help hours tomorrow
- My help hours: Thursday 4-5

Language Modeling

Last week: we made up the probabilities:

p(cat | The girl put out a bowl of milk for her)

p(hat | The girl put out a bowl of milk for her)

Language Modeling

Last week: we made up the probabilities:

p(cat | The girl put out a bowl of milk for her)

p(hat | The girl put out a bowl of milk for her)

This week: we will estimate the probabilities based on corpus frequency data.

Language Modeling

Last week: we made up the probabilities:

p(cat | The girl put out a bowl of milk for her)

p(hat | The girl put out a bowl of milk for her)

This week: we will estimate the probabilities based on corpus frequency data.

We will be able to use the same model to generate text.

Language Modeling

Language Modeling

Today's goal: **assign a probability to a sentence**

This is useful in many tasks:

- Machine Translation:
  - P(high winds tonite) > P(large winds tonite)
- Spelling Correction
  - The office is about fifteen minuets from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)

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:

$$P(w_5 \mid w_1,w_2,w_3,w_4)$$

A model that computes either of these:

$$P(W) \quad \text{or} \quad P(w_n \mid w_1,w_2 \ldots w_{n-1})$$

is called a **language model**.

Language Modeling

- Challenge: compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: rely on the Chain Rule of Probability

Chain Rule

- Recall the definition of conditional probabilities

$\mathbf{p(B \mid A)}$ = Rewriting: $\mathbf{P(A,B)} = P(A)P(B|A)$

$= \frac{P(A,B)}{P(A)}$

Chain Rule in General:

$P(x_1, x_2 \dots x_n) = P(x_1)P(x_2|x_1)\dots P(x_n|x_1 \dots x_{n-1})$

Chain Rule

- Recall the definition of conditional probabilities

  $\mathbf{p(B \mid A) = P(A,B)/P(A)}$

  Rewriting: $\mathbf{P(A,B) = P(A)P(B \mid A)}$

- The Chain Rule in General

Chain Rule

The Chain Rule applied to compute the joint probability of words in sentence:

$$P(w_1, w_2 \ldots w_n) = \prod_n P(w_n \mid w_1 w_2 \ldots w_{n-1})$$

P("its water is so transparent") =

p(its) · p(water | its) · p(is | its water) ·
p(so | its water is) · p(transparent | its water is so)

Estimating Probabilities

Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

Estimating Probabilities

Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

No! Too many possible sentences!

We'll never see enough data for estimating these

Markov Assumption

Simplifying assumption:

$$P(\text{the} \mid \sout{\text{its water is so transparent}}\ \text{that}) \approx P(\text{the} \mid \text{that})$$

First Order Markov Assumption

$$p(w_n \mid w_1 w_2 \cdots w_{n-1}) \approx p(w_n \mid w_{n-1})$$

Markov Assumption

Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

First Order Markov Assumption: $P(w_n \mid w_1 w_2 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1})$

Markov Assumption

Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

Or maybe:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

(2nd Order Markov Assumption)

Markov Assumption

Approximate each component in the product:

$$P(w_1, w_2 \ldots w_n) = \prod_n P(w_n \mid w_1 w_2 \ldots w_{n-1})$$

$$P(w_1, w_2 \ldots w_n) \approx \prod_n P(w_n \mid w_{n-k} \ldots w_{n-1})$$

where k is our window size or our context length

Simplest Case: Unigram Model

$$P(w_1, w_2 \ldots w_n) \approx \prod_n P(w_n)$$

Randomly sampled words from our vocab

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:

$$P(w_1, w_2 \ldots w_n) \approx \prod_n P(w_n \mid w_{n-1})$$

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

Estimating N-Gram Probabilities

How Do We Estimate Probabilities?

- Answer: based on observing frequencies in a training corpus!

$$P(w_1, w_2 \ldots w_n) \approx \prod_n P(w_n \mid w_{n-1})$$

Estimating Bigram Probabilities

The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

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

Another Example

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

- From 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw Bigram Probabilities

- Normalize by unigrams:

- Result:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram Estimates of Sentence Probabilities

P(<s> I want english food </s>) =

$$P(I \mid \langle s \rangle) \times P(want \mid I) \times$$

$$P(english \mid want) \times P(food \mid english)$$

$$\times P(\langle /s \rangle \mid food)$$

$$= 0.000031$$

Practicalities

When programming, we will handle probabilities in **log space** to avoid numerical underflow.

(This will be true for the rest of the class!)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

log
e

Google N-Gram Release (2006)

AUG
3

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.

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google N-Gram Release (2006)

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://ngrams.googlelabs.com/

Generating Text

- 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