Chapter 6

Language Model

Language Model Definition

- **Definition:** A computational mechanism that predicts the probability of a sequence of words.
- **Goal:** Estimate how likely a given text is or generate coherent text.
- **Goal:** Estimate the likelihood of the next word given previous words

Probabilistic Language Models

- The goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds) > P(**large** winds)
  - Spelling Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(You're right) >> P(your write)
  - + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

  $$P(S) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

- 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:

  $P(S)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

The Chain Rule: General

- The definition of conditional probabilities

  $P(A \mid B) = P(A, B) / P(B)$

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

- More 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: joint probability in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

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)

How to estimate these 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

Andrei Markov

- Simplifying assumption:

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

- Or maybe

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

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

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

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Docs = [“The cat sat”, “The dog barked”, “The cat meowed”]

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

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

Quiz

- **Docs = [**“The cat sat”, “The dog barked”, “The cat meowed”]

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

N-gram models

- **Problem:** If a word or n-gram is unseen in training, its probability is **zero** (e.g., "NLP" never follows "love").
- **Solutions:**
  - **Add-1 (Laplace) Smoothing:** Add 1 to all counts.

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

Quiz

If we estimate a bigram language model from the following corpus, **what is P(not|do)?**

<s> I am Sam </s>

<s> Sam I am </s>

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

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 (absolute measure)

- Out of 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 (relative measure)

- Normalize by unigrams:

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

- Result:

	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

$P(\langle s\rangle \text{ I want english food } \langle /s\rangle) =$

$P(I|\langle s\rangle)$

$\times \ P(\text{want}|I)$

$\times \ P(\text{english}|\text{want})$

$\times \ P(\text{food}|\text{english})$

$\times \ P(\langle /s\rangle|\text{food})$

$= \ .000031$

What kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

world

grammar

grammar (contingent zero)

grammar (structural zero)

Practical Issues

- We do everything in log space
  - Avoid underflow: multiplying extremely small numbers
  - Adding is faster than multiplying

$$p_1 \times p_2 \times p_3 \times p_4 \Rightarrow \log p_1 + \log p_2 + \log p_3 + \log p_4$$

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, machine translation 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 instead
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- How well can we predict the next word?

  I always order pizza with cheese and ____

  The 33rd President of the US was ____

  I saw a ____

  - Unigrams are terrible at this game. (Why?)

- A better model
  - 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(sentence)

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

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

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

Chain rule:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Example

- How hard is the task of recognizing digits ‘0,1,2,3,4,5,6,7,8,9’
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- Perplexity is weighted equivalent branching factor (number of possible children)

QUESTION 3

A traffic signal has three colors: green, yellow, and red, which appear with the following probabilities. Using a unigram model, **what is the perplexity of the sequence (green, yellow, red)?**

P(green) = 2/5

P(yellow) = 1/5

P(red) = 2/5

$$PP(green, yellow, red) = \left(\frac{2}{5} \times \frac{1}{5} \times \frac{2}{5}\right)^{-\frac{1}{3}}$$

Lower perplexity = better model

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109