Language Modeling (Part 2)

LING83800: METHODS IN COMPUTATIONAL LINGUISTICS II

April 1, 2024

Spencer Caplan

Generalization and overfitting

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 peshwari
peshwari naan
naan </s>

I want to eat peshwari naan

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Shakespeare as a 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)

- 4-grams are worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

The wall street journal is no Shakespeare (sorry WSJ)

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	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
3 gram	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

P(“offer” | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

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)!

Bigram illustration

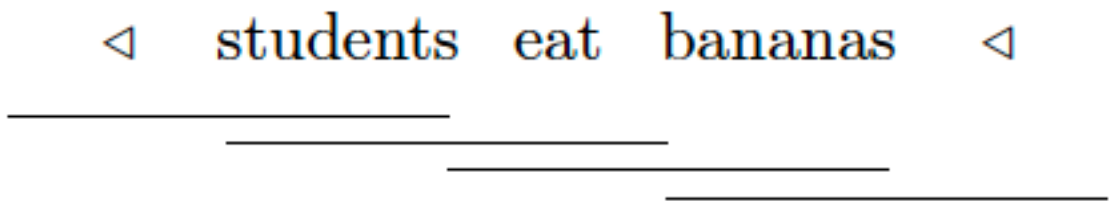


Figure 1.1: The four bigrams extracted by a bigram language from the sentence '*students eat bananas*', padded with '◁' symbols at its beginning and end.

Bigram formalism

In more detail, a *bigram language model* is defined as follows. If $\boldsymbol{W} = (W_1, \ldots, W_n)$ then

$$\mathrm{P}(\boldsymbol{W}) \quad = \quad \prod_{i=1}^{n+1} \mathrm{P}(W_i \mid W_{i-1}) \qquad (1.9)$$

where

$$\mathrm{P}(W_i{=}w' \mid W_{i-1}{=}w) \quad = \quad \Theta_{w,w'} \text{ for all } i \text{ in } 1, 2, \ldots, n+1$$

Bigram formalism

Example 1.9: A bigram model assigns the following probability to the string '*students eat bananas*'.

$$\mathrm{P}(\text{'}students\ eat\ bananas\text{'}) = \Theta_{\triangleleft,\text{'}students\text{'}}\ \Theta_{\text{'}students\text{'},\text{'}eat\text{'}}\ \Theta_{\text{'}eat\text{'},\text{'}bananas\text{'}}\ \Theta_{\text{'}bananas\text{'},\triangleleft}$$

Bigram formalism

Example 1.9: A bigram model assigns the following probability to the string '*students eat bananas*'.

$$\mathrm{P}(\text{'}students\ eat\ bananas\text{'}) = \Theta_{\triangleleft,\text{'}students\text{'}}\ \Theta_{\text{'}students\text{'},\text{'}eat\text{'}}\ \Theta_{\text{'}eat\text{'},\text{'}bananas\text{'}}\ \Theta_{\text{'}bananas\text{'},\triangleleft}$$

$$\widehat{\Theta}_{w,w'} = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,\circ}(\boldsymbol{d})}$$

Bigram smoothing

- We could smooth bigram Thetas like we did for unigrams
  - Unigram smoothing parameter: α
  - Bigram smoothing parameter: β

- However, aren't we much more likely to see a high-frequency word(w') following w than a low frequency one?
  - So we set a single constant β to represent the likelihood of *any* unseen bigram, but scale the estimate in our definition of theta to be proportional to the unigram count

$$\widetilde{\Theta}_{w,w'} = \frac{n_{w,w'}(\boldsymbol{d}) + \beta\,\widetilde{\theta}_{w'}}{n_{w,\circ}(\boldsymbol{d}) + \beta}$$

Add-alpha (beta) is a blunt tool

So depending on the application, there are better (more complicated) techniques

Sometimes it helps to use **less** context

Condition on less context for contexts you haven't learned much about

Backoff:

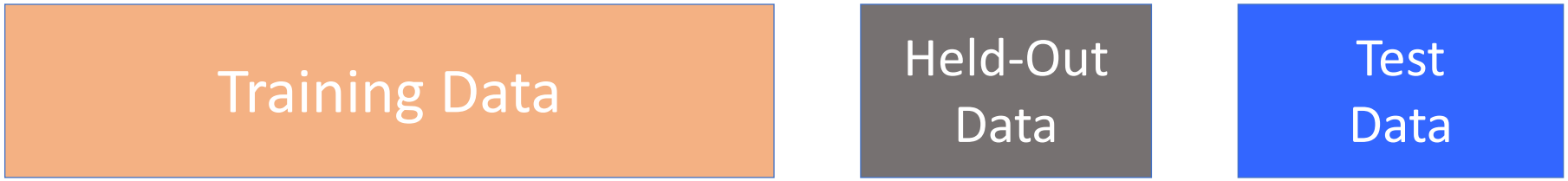
use trigram if you have good evidence,
otherwise bigram, otherwise unigram

Interpolation:

mix unigram, bigram, trigram according to some parameter λ

How to set the smoothing parameters?

- Use a **held-out** corpus



- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

Kneser-Ney Smoothing

Kneser-Ney Intuition

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading*_____?
    Francisco / glasses
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"

- The unigram is useful exactly when we haven't seen this bigram!
- Instead of P(w): "How likely is w"
- $P_{continuation}$(w): "How likely is w to appear as a novel continuation?
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

Kneser-Ney Definition

$$\widetilde{\Theta}_{w,w\prime} = \frac{n_{w,w\prime}(d) + \beta k_{w\prime}}{n_{w,0}(d) + \beta}$$

Still need to handle unseen unigrams!

$$k_{w\prime} = \frac{\text{number of word types that precede w'}}{\text{number of words types that precede any word}}$$

$$\widetilde{\kappa}_w = \frac{k_w(\boldsymbol{d}) + \alpha}{k_\circ(\boldsymbol{d}) + \alpha|\mathcal{W}|}$$

Probabilistic Language Models

- 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:

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

- Better: **the grammar** But **language model** or **LM** is standard in Engineering

(And sometimes that's exactly what we want)

- What does finite-state mean?
- Are the N-gram models we've talked about finite-state machines?
- If so, about how many states did we have in them?