Language Model

Introduction to N-grams

Probabilistic Language Model

- Goal: assign a probability to a sentence
- Application:
  - Machine Translation

    P(high winds tonight) > P(large winds tonight)
  - Spelling Correction

    P(about 15 minutes from) > P(about 15 minuets from)
  - Speech Recognition

    P(I saw a van) > P(eyes awe of an)

How to Compute Language Modeling

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

$$p(w_t | w_1, ..., w_{t-1})$$

Chain Rule of Probability

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_n|w_1, ..., w_{n-1})$$

Example

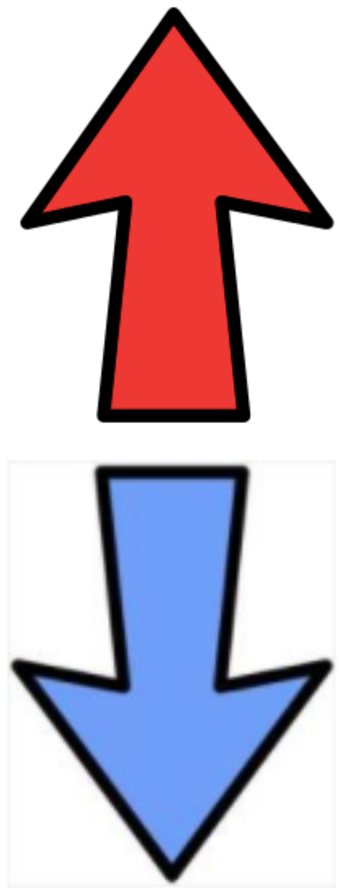
$$\begin{aligned} &p(\text{its water is so transparent}) \\ &\quad = p(\text{its}) \times p(\text{water}|\text{its}) \times p(\text{is}|\text{its water}) \times p(\text{so}|\text{its water is}) \\ &\qquad \times p(\text{transparent}|\text{its water is so}) \end{aligned}$$

The probability of a sentence can be obtained via the Chain Rule of Probability

Evaluation of Language Model

real
frequently observed

higher probability



ungrammatical
rarely observed

lower probability

Evaluation via Perplexity

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

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

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

Markov Assumption

- Too many possible combinations of $w_1, ..., w_t$
- Impossible to infer $p(w_t|w_1, ..., w_{t-1})$
- Approximation:
  - Unigram

$$p(w_t|w_1, \cdots, w_{t-1}) \approx P_{W_2|W_1}(w_t|w_{t-1})$$

  - Bigram

$$p(w_t|w_1, \cdots, w_{t-1}) \approx P_{W_3|W_1,W_2}(w_t|w_{t-1}, w_{t-2})$$

  - Higher order approximation…

Parameter Estimation

- In a bigram language model, parameters are

$$P_{W_2|W_1}(w_2|w_1), \qquad \forall\ w_1, w_2 \in \text{vocabulary}$$

- Straight forward: the ML estimator

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

$\text{count}(\cdot,\cdot)$ is the number of cooccurrence of a word pair.
$\text{count}(\cdot)$ 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:

$$P_{W_2|W_1}(\mathrm{I}|\langle\mathrm{s}\rangle) = \frac{2}{3} \qquad P_{W_2|W_1}(\mathrm{Sam}|\langle\mathrm{s}\rangle) = \frac{1}{3} \qquad P_{W_2|W_1}(\mathrm{am}|\mathrm{I}) = \frac{2}{3}$$

$$P_{W_2|W_1}(\langle/\mathrm{s}\rangle|\mathrm{Sam}) = \frac{1}{2} \qquad P_{W_2|W_1}(\mathrm{Sam}|\mathrm{am}) = \frac{1}{2} \qquad P_{W_2|W_1}(\mathrm{do}|\mathrm{I}) = \frac{1}{3}$$

……

Online Resource

Google

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.

Training Corpus

… denied the allegations
… denied the reports
… denied the claims
… denied the request

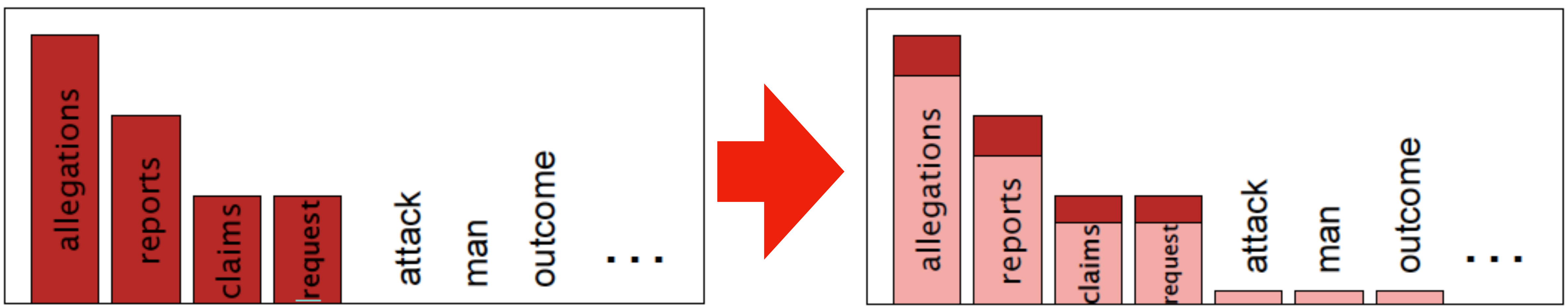
$P(\text{offer}|\text{denied the}) = 0$

Real Applications

… denied the offer

Smoothing the "Zero"s

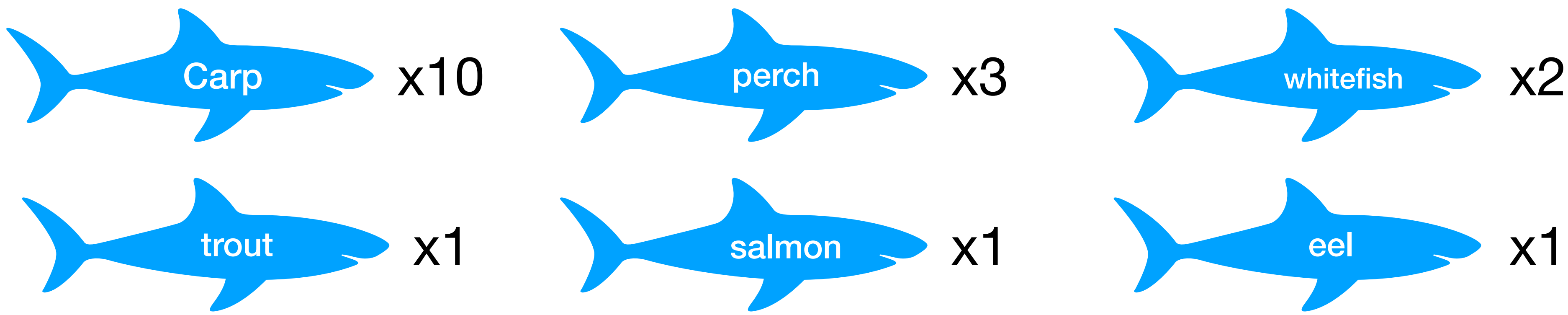
When we have sparse statistics, steal probability mass to generalize better.



Reference: Chen, Stanley F., and Joshua Goodman. "An empirical study of smoothing techniques for language modeling."

Good Turing Smoothing

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



- 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?

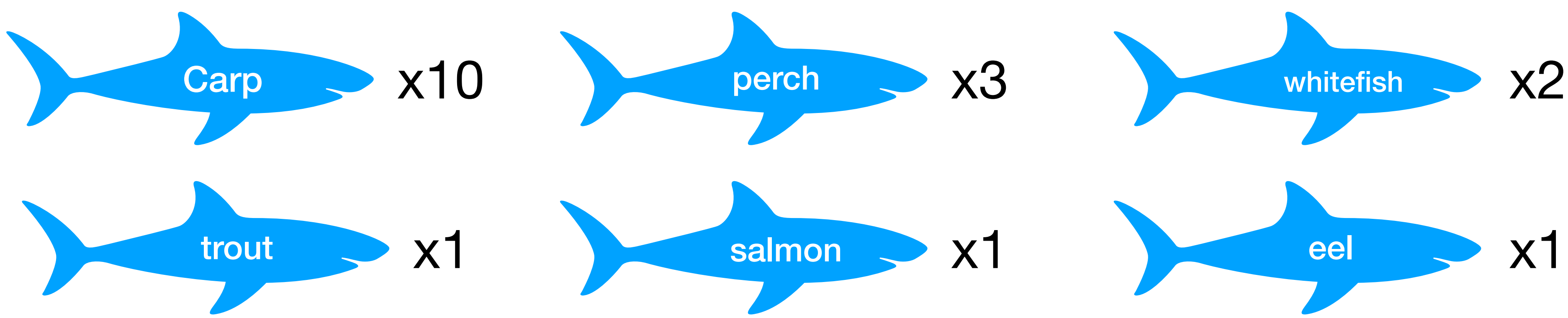
Leave-One-Validation

- 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?
    - # of fishes occur once / 18 = 3/18
- The fraction of held-out fishes are seen k times in training?
    - (# 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



- How likely is that next species is trout? **1/18**
- Assume there are new species, how likely is it that next species is new? **3/18**
- Now how likely is that next species is trout?

1/18 * (2/3) = 1/27

Good Turing Smoothing

- N_i is the number of words that occur i times
- $N = \sum_i iN_i$ 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 Smoothing

Good Turing

NIPS 2017

Absolute Discounting Smoothing

Steal probability mass to unseen samples

discounted bigram

Interpolation weight

$$P^{(\mathrm{AD})}_{W_2|W_1}(w_2|w_1) = \frac{\mathrm{count}(w_2, w_1) - d}{\mathrm{count}(w_1)} + \lambda(w_1) p_W(w_2)$$

unigram distribution

Absolute Discounting

glasses | sun

$$P^{(\mathrm{AD})}_{W_2|W_1}(w_2|w_1) = \frac{\mathrm{count}(w_2, w_1) - d}{\mathrm{count}(w_1)} + \lambda(w_1) p_W(w_2)$$

P(glasses) << P(Fransisco)

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

- Principle of probability

$$\sum_{w_1} P^{(\mathrm{AD})}_{W_2|W_1}(w_2|w_1) P_W(w_1) \neq P_W(w_2)$$

- Choice of continuation distribution!

Kneser-Ney Smoothing

$$P^{(\mathrm{KN})}_{W_2|W_1}(w_2|w_1) = \frac{\mathrm{count}(w_2, w_1) - d}{\mathrm{count}(w_1)} + \lambda(w_1) P_{\mathrm{cont}}(w_2)$$

The normalized discount; the probability mass we've discounted

$$\lambda(w) = \frac{d}{c(w)} |\{w' : \mathrm{count}(w, w') > 0\}|$$

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

The number of word types that can follow w

P(glasses) > P(Fransisco)

Marginal constraint gives an only solution to the interpolated distribution.

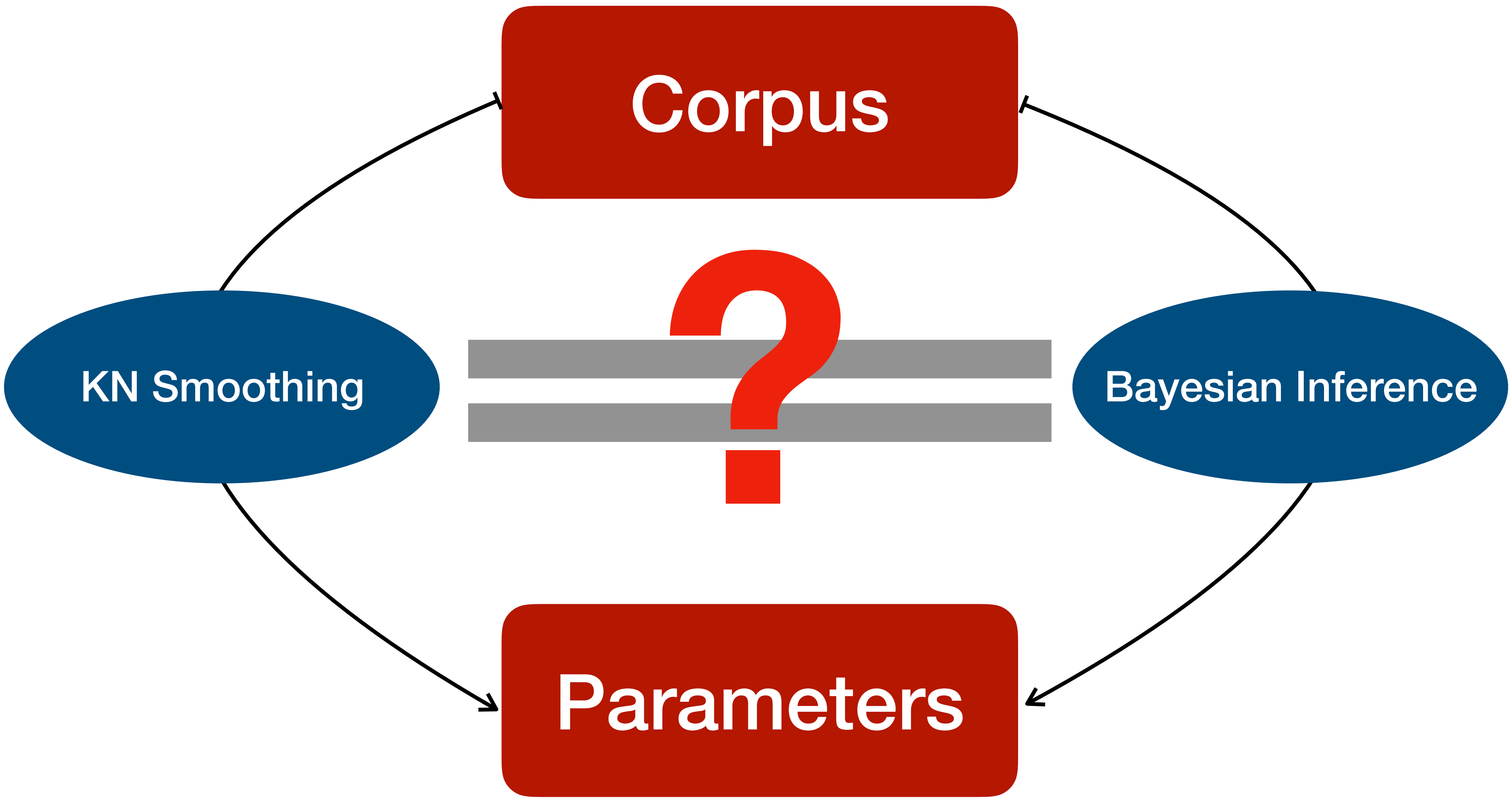
Trigram and More

- Recursive formulation of KN smoothing

$$P^{(\mathrm{KN})}(w_n|w_1, w_2, ..., w_{n-1}) = \frac{\max\{\mathrm{count}^{(KN)}(w_1, w_2, ..., w_n) - d, 0\}}{\mathrm{count}^{(KN)}(w_1, w_2, ..., w_{n-1})} + \lambda(w_1, ..., w_{n-1}) P^{(\mathrm{KN})}(w_n|w_2, ..., w_{n-1})$$

$$\mathrm{count}^{(KN)}(\cdot) = \begin{cases} \#\text{ of occurrence of } \cdot & \text{for the highest order} \\ \#\text{ of unique word types for } \cdot & \text{for lower order} \end{cases}$$

Bayesian Interpretation of KN Smoothing



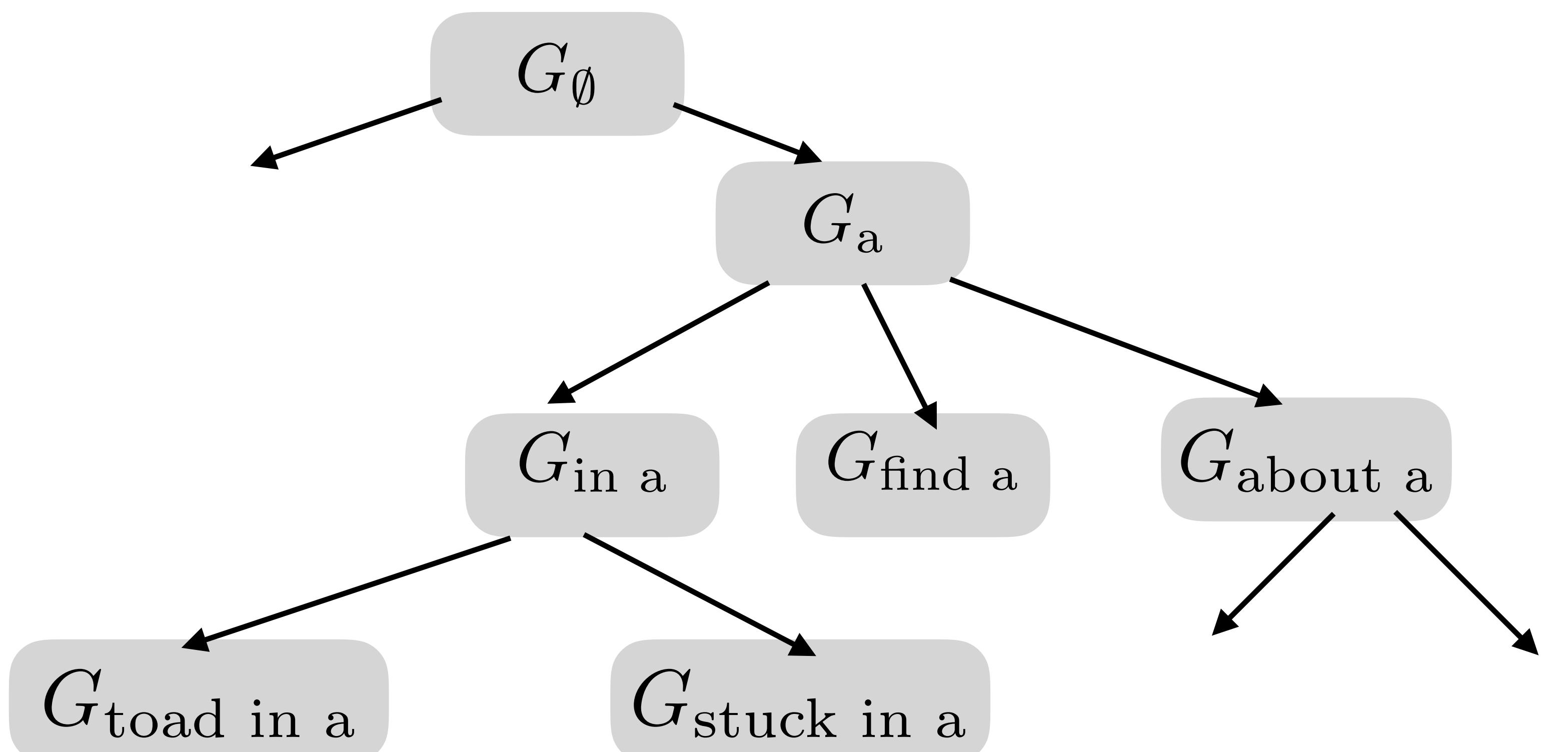
https://www.stats.ox.ac.uk/~teh/research/compling/hpylm.pdf

Smoothing via Context Tree

- Parameterized the conditional distribution of Markov model

$$P(w|u = (w_1, ..., w_{n-1})) = G_u(w)$$

- G_u is a probability vector associated with context u



- Smoothing equals the dependency between G_u and $G_{\mathrm{parent}(u)}$

Chinese Restaurant Process

- Chinese Restaurant Process $\mathrm{CRP}(d, \theta, G_0)$ is a distribution over distributions over a probability space
- CRP is defined over draws from $G_1 = \mathrm{CPR}(d, \theta, G_0)$
- 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

base distribution $P(w|w_2, ..., w_n)$ $\xrightarrow{\textbf{CRP}}$ **Output distribution** $P(w|w_1, w_2, ..., w_n)$

$$G_u(w) \sim \mathrm{CRP}(d_{|u|}, \theta_{|u|}, G_{\mathrm{parent}(u)}(w))$$

Neural Network in Language Model

- N-gram models can be thought as classifications

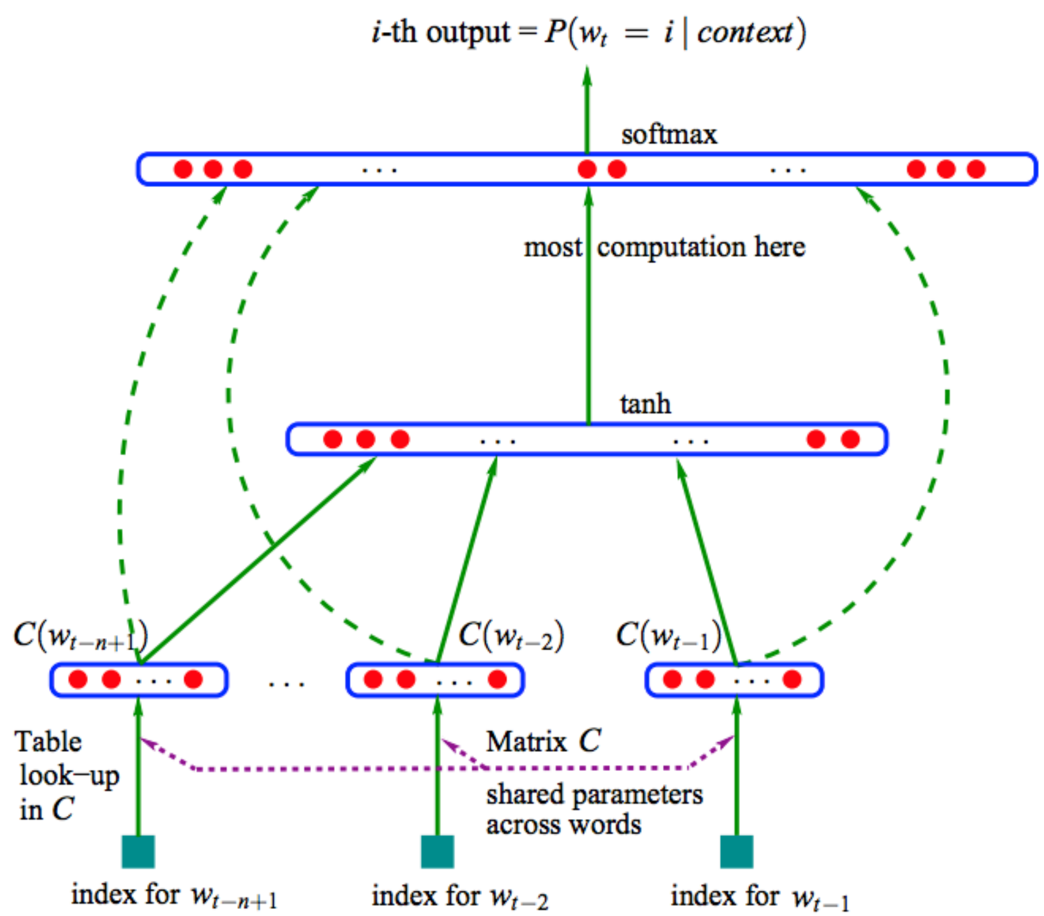
history predict **current word**

$$(w_1, ..., w_{n-1}) \xrightarrow{\text{predict}} w_n$$

- 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)



output word

softmax as probability distribution

context of N words

Reference: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

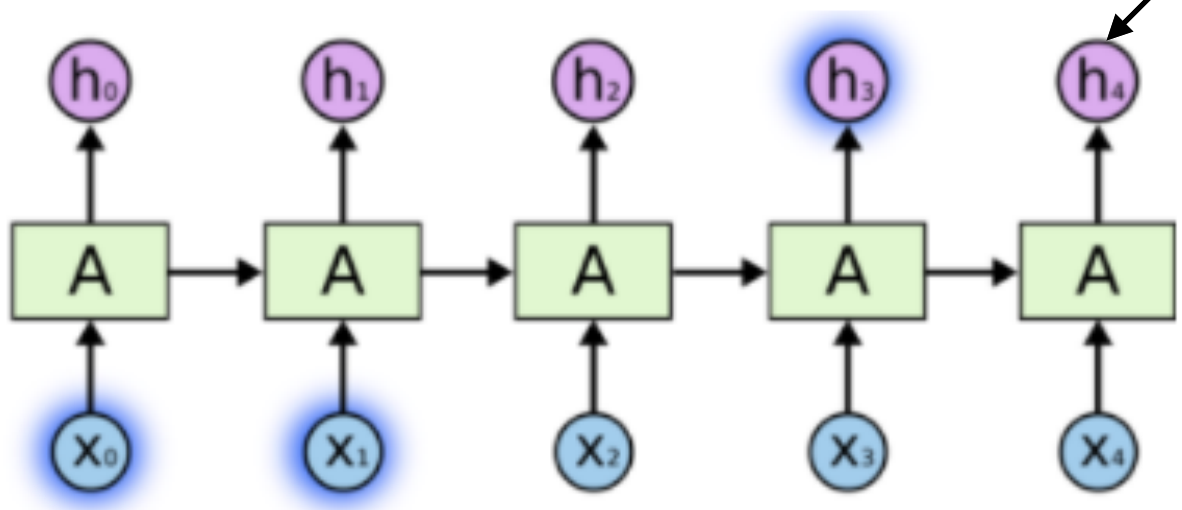
Curse of N-gram Models

- Language has **long-distance** dependencies

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

- Modeling via **RNN**

Use this to predict the next word



Reference: http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf