Language Models

Professor Marie Roch

for details on N-gram models, see chapter 3 in draft:
Jurafsky, D., and Martin, J. H. (2023). *Speech and Language Processing (Pearson Prentice Hall, Upper Saddle River, NJ)*

SAN DIEGO STATE UNIVERSITY

Acoustic models

- Typically produce
  - Words (for very small vocabulary tasks)
  - Phonemes (much more common)

- Need to assemble these into word sequences

Lexical baseforms

- Describes the transcription of a word into subword units.
- Issues
  - pronunciations due to dialects, e.g. “tomato”
  - coarticulation
    - across words, “you” /y uw/ versus “did you ...” /jh uh/
  - common contractions

Pronouncing Dictionaries

- Carnegie-Mellon Pronouncing dictionary: http://www.speech.cs.cmu.edu/cgi-bin/cmudict
- Over 100,000 entries
- 39 phonemes
- Transcription examples:

DOLPHIN	D AA1 L F AH0 N
TOMATO	T AH0 M EY1 T OW2
TOMATO(2)	T AH0 M AA1 T OW2
YOU'VE	Y UW1 V

primary stress (1)

unstressed (0)

secondary stress (2)

What do we want to solve?

- Find words *W* that max. observations *O*

$$\widehat{W} = \arg\max_{w \in L} \underbrace{\mathrm{P}(O|W)}_{acoustic} \underbrace{\mathrm{P}(W)}_{language}$$

- How can we find *W* in a reasonable manner?

Probability imbalance

- Acoustic observations assumed independent
  - Clearly false
  - Underestimate of P(O|W)
- Language model scale factor (weight)

$$\hat{W} = \arg\max_{w \in L} \mathrm{P}(O \mid W)\,\mathrm{P}^{LMSF}(W)$$

$$\text{typical } LMSF \in [5,15]$$

Probability and sentence length

- Each time we add a word to W,
  P(W) decreases

- Large vocabulary language models tend to have lower probabilities, so the penalty for adding words becomes even greater.

- We can consider this to be a penalty for inserting words.

Insertion Penalty and Recognition Bias

- Search becomes biased:
  - Larger penalty → preference for shorter sentences with longer words
  - Smaller penalty → preference for longer sentences with shorter words

Word insertion penalty

- To avoid bias towards large or small words we use a tunable *word insertion penalty* parameter.

$$\hat{W} = \arg\max_{w \in L} \mathrm{P}(O \mid W) \cdot \underbrace{\mathrm{P}^{LMSF}(W) \cdot WIP^{N}}_{P(LM)} \qquad 0 < WIP \leq 1$$

- strong penalty → prefers longer sentences
- weak penalty → prefers shorter sentences

Decoding

- Decoders are used to determine the optimal word sequence.
- Combines the acoustic models with search that considers the language model

Narrowing search with a language model

- Don’t move or I’ll …
- Get ‘er …
- What will she think of …
- This enables …

Applications

- Speech recognition
- Handwriting recognition
- Spelling correction
- Augmentative communication

and more…

Constituencies

- Groupings of words
  - I didn't see you *behind the bush*.
  - She *ate quickly* as she was *late for the meeting*.
- Movement within the sentence:

  As she was late for the meeting, she ate quickly

  As she was late for, she ate quickly the meeting.
- Constituencies aid in prediction.

Strategies for construction

- Formal grammar
  - Requires intimate knowledge of the language
  - Usually context free and cannot be represented by a regular language
  - We will not be covering this in detail

N-gram models

- Suppose we wish to compute the probability the sentence: She sells seashells down by the seashore.
- We can think of this as a sequence of words:

$$\underbrace{\text{She}}_{w_1}\ \underbrace{\text{sells}}_{w_2}\ \underbrace{\text{seashells}}_{w_3}\ \underbrace{\text{down}}_{w_4}\ \underbrace{\text{by}}_{w_5}\ \underbrace{\text{the}}_{w_6}\ \underbrace{\text{seashore}}_{w_7}$$

$$P(w_1^7) = P(w_1, w_2, w_3, w_4, w_5, w_6, w_7)$$

Estimating word probability

- Suppose we wish to compute the probability w_2 (*sells* in the previous example).

  We could estimate using a **relative frequency**

$$P(w_2) = \frac{\#\text{ times } w_2 \text{ occurs}}{\#\text{ of times all words occur}}$$

  but this ignores what we could have learned with the first word.

Conditional probability

Recall conditional probability

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

or in our problem:

$$P(w_2 \mid w_1) = \frac{P(w_2 \cap w_1)}{P(w_1)} = \frac{P(w_1 \cap w_2)}{P(w_1)}$$

$$= \frac{P(w_1, w_2)}{P(w_1)} \text{ defn} \cap \text{ for words}$$

Conditional probability

Next, consider $P(w_1, w_2)$

Since as $P(w_2 \mid w_1) = \dfrac{P(w_1, w_2)}{P(w_1)}$,

clearly $P(w_1, w_2) = P(w_2 \mid w_1) P(w_1)$

Chain rule

- Now let us consider:

$$P(w_1, w_2, w_3) = P(w_3 \mid w_1, w_2) \underbrace{P(w_1, w_2)}_{\text{we just did this part}}$$

$$= P(w_3 \mid w_1, w_2) P(w_2 \mid w_1) P(w_1)$$

- By applying conditional probability repeatedly, we derive the chain rule:

$$\begin{aligned} \mathrm{P}(W) &= \mathrm{P}(w_1 w_2 \ldots w_n) \\ &= \mathrm{P}(w_1)\,\mathrm{P}(w_2 \mid w_1)\,\mathrm{P}(w_3 \mid w_1 w_2) \ldots \mathrm{P}(w_n \mid w_1 w_2 \ldots w_{n-1}) \\ &= \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_1 w_2 \ldots w_{i-1}) \end{aligned}$$

Sparse problem space

- Suppose V distinct words.
- w_1^i has V^i possible sequences of words.
  Here, $w_1^i \triangleq w_1, w_2, w_3, \ldots, w_i$
- N_{Tokens} – The number of N-grams (including repetitions) occurring in a corpus
- Problem: In general, unique(*N*)<<valid tokens for the language.

  "The gently rolling hills were covered with bluebonnets"
  had no hits on Google at the time this slide was published.

Markov assumption

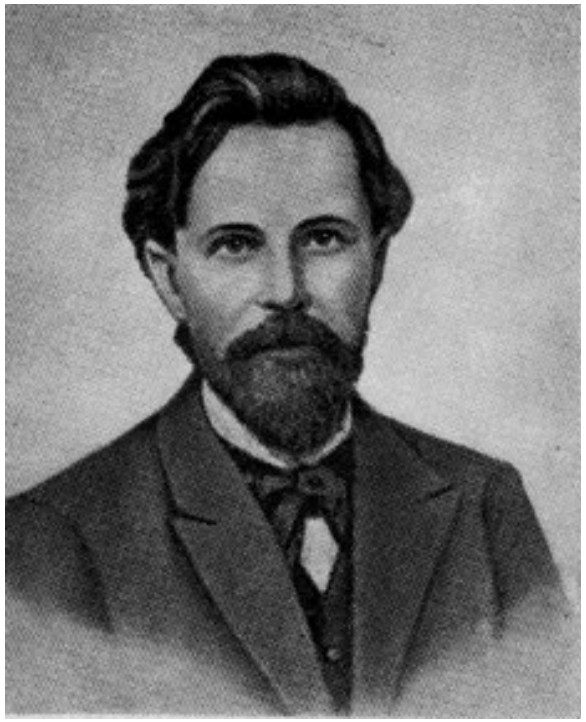
- A prediction is dependent on the current state but independent of previous conditions

- In our context:

$$\mathrm{P}(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-1}) \text{ by the Markov assumption}$$

which we at times relax to N-1 words:

$$\mathrm{P}(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-N+1}^{n-1})$$



Andrei Markov
1856-1922

Special N-grams

- Unigram
  - Only depends upon the word itself.
  - $P(w_i)$
- Bigram
  - $P(w_i|w_{i-1})$
- Trigram
  - $P(w_i|w_{i-1}, w_{i-2})$
- Quadrigram
  - $P(w_i|w_{i-1}, w_{i-2}, w_{i-3})$

Preparing a corpus

- Make case independent
- Remove punctuation and add start & end of sentence markers <s> </s>
- Other possibilities
  - part of speech tagging
  - lemmas: mapping of words with similar roots
    e.g., sing, sang, sung → sing
  - stemming: mapping of derived words to their root
    e.g., parted → part, ostriches → ostrich

An Example

<s> I am Sam </s>

<s> Sam I am </s>

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

Dr. Seuss, *Green Eggs and Ham*, 1960.

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

$$P\left(w_n \mid w_{n-N+1}^{n-1}\right) = \frac{C\left(w_{n-N+1}^{n-1} w_n\right)}{C\left(w_{w-N+1}^{n-1}\right)}$$

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*

Bigram Counts from 9,222 sentences

"i want"

w_i

	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

w_{i-1}

Bigram Probabilities

Unigram counts

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

$$P(\text{i want}) = \frac{C(\text{i want})}{C(\text{i})} = \frac{827}{2533} \approx 0.33$$

w_i

w_{i-1}	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|<s>)P(want|I)P(english|want)P(food|english)P(</s>|food)

=.000 031

Shakespeare:

N=884,647 tokens, V=29,066

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? •Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
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.

How will this work on *Huckleberry Finn*?

The need for n-gram smoothing

- Data for estimation is sparse.
- On a sample text with several million words
  - 50% of trigrams only occurred once
  - 80% of trigrams occurred less than 5 times
- Example: When pigs fly

$$\mathrm{P}(fly \mid when, pigs) = \frac{C(when, pigs, fly)}{C(when, pigs)}$$

$$= \frac{0}{C(when, pigs)} \qquad \text{if "when pigs fly" unseen}$$

Smoothing strategies

- Suppose P(fly | when, pigs) = 0
- Backoff strategies do the following
  - When estimating P(Z | X, Y) where C(XYZ)>0,
  - don't assign all of the probability, save some of it for the cases we haven't seen. This is called discounting and is based on Good-Turing counts

Smoothing strategies

- For things that have C(X, Y, Z) = 0, use P(Z|Y), but scale it by the amount of leftover probability
- To handle C(Y, Z) = 0, this process can be computed recursively.

Is our model any good? Perplexity

- Measure of ability of language model to predict next word
- Related to cross entropy of language, *H(L),* perplexity is $2^{H(L)}$

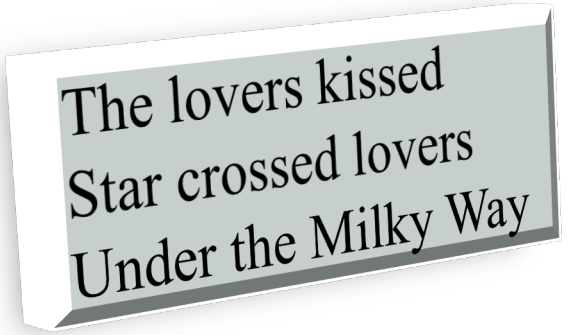
$$
\begin{aligned}
H(L) &= \lim_{n\to\infty} \frac{1}{n} H(w_1, w_2, \ldots, w_n) \\
&= -\lim_{n\to\infty} \frac{1}{n} \sum_{W\in L} P(w_1, w_2, \ldots, w_n) \log\left(P(w_1, w_2, \ldots, w_n)\right)
\end{aligned}
$$

- Lower perplexity indicates better modeling (theoretically)

Counting the number of times things occur

- c - # times a word occurs (e.g. c for xylophone is typically small)
- N_c - # of different words that occur c times

Example



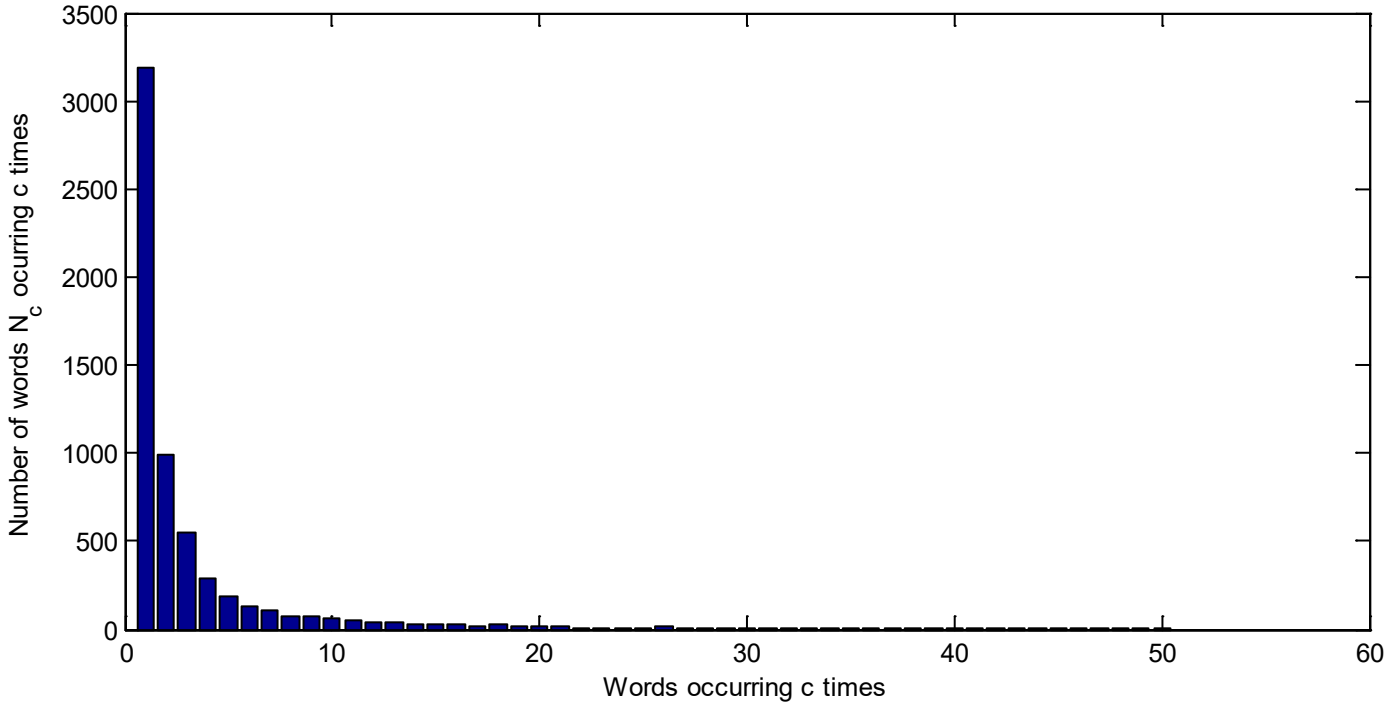
$$c \text{ is } \begin{cases} 1 & x \in \{crossed, kissed, milky, star, under, way\} \\ 2 & x \in \{lovers, the\} \end{cases}$$

$$N_1 = 6$$

$$N_2 = 2$$

Counts from the Switchboard Corpus

- N_C counts typically exhibit exponential decay



Turing counts

- Intuition: Words seen very few times probably have their probability underestimated.
- Goal: Assign some probability to unseen events

Image credit: www.jamboree.freedom-in-education.co.uk/w%27s%20craft%20corner/robin_hood.gif

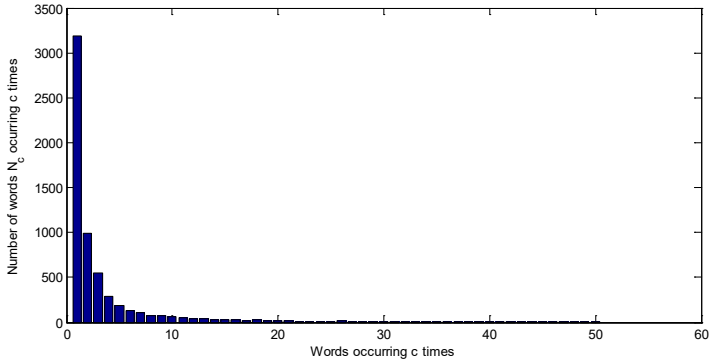
$$P_{GT}(unseen) = \frac{N_1}{N_{tokens}}$$

we will see later why this makes sense

Turing counts

- Reestimate the other counts

$$c^* = (c+1)\frac{E[N_{c+1}]}{E[N_c]}$$



- Turing suggested approximating the expectations by the observed counts

$$c^* = (c+1)\frac{N_{c+1}}{N_c}, \text{ e.g. } 4^* = (4+1)\frac{N_5}{N_4}$$

Estimating the missing mass

Missing mass

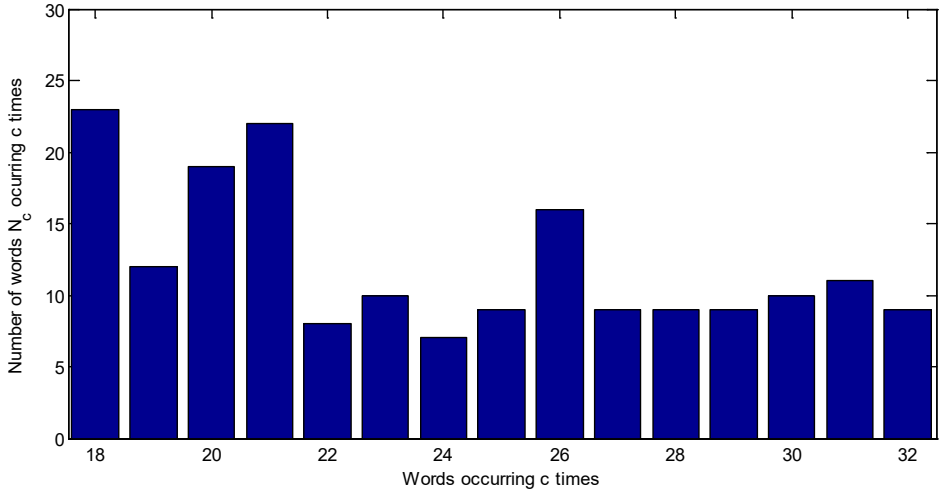
$$\sum_{w|count(w)=0} P(w) = \sum_{w|count(w)=0} \frac{0^*(w)}{N_{tokens}}$$

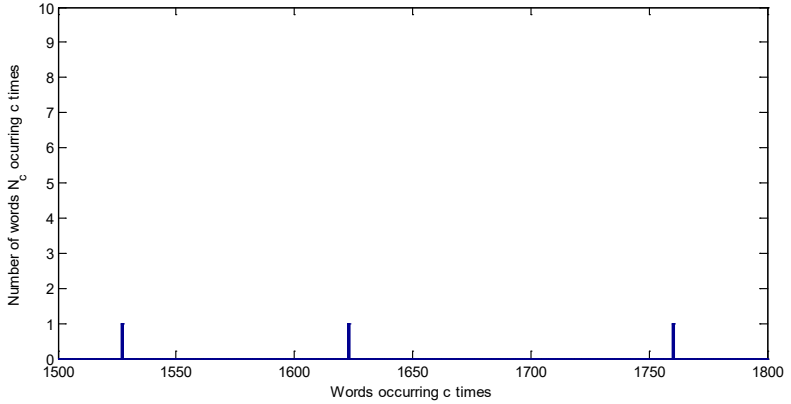
$$= \sum_{w|count(w)=0} \frac{(0+1)\frac{N_1}{N_0}}{N_{tokens}} = \sum_{w|count(w)=0} \frac{N_1}{N_0 N_{tokens}}$$

$$= N_0 \frac{N_1}{N_0 N_{tokens}} = \frac{N_1}{N_{tokens}} \text{ as } count(w) = 0 \text{ occurs } N_0 \text{ times}$$

Good-Turing counts

- Unfortunately, the counts can be noisy and can contain gaps





- Good suggested smoothing them

Linear Good-Turing estimates

- Church & Gale/Gale proposed:
  - Smooth counts to distribute weight over gaps
  - Perform a linear fit in log-log space and use the fit in place of counts
  - Details in: Gale, W. (**1994)** Good-Turing Smoothing Without Tears. *J. Quant. Linguistics, 2, 24 pp.*

Good-Turing estimates

- In practice only need approximations for poorly observed observations with low frequency (small c)
- Common to use unadjusted counts for $c \geq 5$.

- Good-Turing is rarely used by itself, but typically used as part of something else.

Backoff

Only rely on lower-order N-grams when needed.

- Katz backoff
- Kneser-Ney

- Relies on *discounted probability* P^*
  - Reduce probability estimates
  - Give reduction to others

Katz backoff

$$\mathrm{P}_{katz}(z \mid x, y) = \begin{cases} \mathrm{P}^*(z \mid x, y) & \text{if } C(x, y, z) > 0 \\ \alpha(x, y) P_{katz}(z \mid y) & \text{elif } C(x, y) > 0 \\ \mathrm{P}_{katz}(z \mid y) & \text{otherwise} \end{cases}$$

$$\mathrm{P}_{katz}(z \mid y) = \begin{cases} \mathrm{P}^*(z \mid y) & C(y, z) > 0 \\ \alpha(y)\,\mathrm{P}^*(z) & \text{otherwise} \end{cases}$$

Notes:
- J&M printing 1 & 2 have an error in the third line of the upper formula
- Huang et al. present a more general form where the discount can be applied for counts greater than 0

Discounted probability (Katz)

As $\sum_{z_i \in V} P\,(z_i|x,y) = 1$, we need to discount the probability for any given z:

$$P^*(z|x,y) = \frac{c^*(x,y,z)}{c(x,y)}$$

On average:

$$\frac{c^*(x,y,z)}{c(x,y)} < \frac{c(x,y,z)}{c(x,y)}$$

so the sum is likely to be < 1

How much is left over?

$$\sum_{w_n : C(w_{n-N+1}^{n}) > 0} \mathrm{P}^*(w_n \mid w_{n-N+1}^{n-1}) \rightarrow \text{sum discounted P}$$

What's left over?

$$\beta(w_{n-N+1}^{n-1}) = 1 - \sum_{w_n : C(w_{n-N+1}^{n}) > 0} \mathrm{P}^*(w_n \mid w_{n-N+1}^{n-1})$$

Concrete example: trigram

- Trigrams seen in training: ***with you X***
  - with you i
  - with you there
- Backoff: you *word*
  - left over probability: $\beta(w_{n-N+1}^{n-1})P(word|\text{you})$
  - No need to use backoff bigrams for things that were observed: P(i|you) and P(there|you).

Concrete example: trigram

- Subtract out the P for observed trigrams and scale up the probability

$$\alpha(w_{n-N+1}^{n-1}) = \alpha(\underbrace{w_{n-2}, w_{n-1}}_{\text{depends on context trigram case}})$$

$$= \frac{\beta(w_{n-N+1}^{n-1})}{\left(1-(P(i \mid you)+P(there \mid you)\right)}$$

and we compute $\alpha(w_{n-2}, w_{n-1})P(word \mid you)$

Backoff weighting
(formal presentation)

$$\alpha(w_{n-N+1}^{n-1}) = \frac{\text{left over P}}{\text{Sum of Katz N-1 gram P's that we will use}}$$

$$= \frac{\beta(w_{n-N+1}^{n-1})}{\sum_{w_n : C(w_{n-N+1}^{n})=0} P_{katz}(w_n \mid w_{n-N+2}^{n-1})}$$

$$= \frac{1 - \sum_{w_n : C(w_{n-N+1}^{n})>0} \mathrm{P}^*(w_n \mid w_{n-N+1}^{n-1})}{1 - \sum_{w_n : C(w_{n-N+1}^{n})>0} \mathrm{P}^*(w_n \mid w_{n-N+2}^{n-1})}$$

Neural language models

- Advantages
  - As the net learns a representation, similarities can be captured
    Example: Consider food
    - Possible to learn common things about foods
    - Yet the individual items can still be considered distinct

    There are approaches to capture commonality in N-gram models (e.g. Knesser-Ney), but they lose the ability to distinguish the words

Neural language models

  - Word embeddings can learn low dimensional representations of words that can capture semantic information
- Disadvantages
  - Requires very large training data

Transformers are becoming competitive with traditional language models. See Irie et al. (2019) for an example: DOI:10.21437/Interspeech.2019-2225) and the discussion in the J&M chapter.