Statistical Language Modeling for Speech Recognition

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:

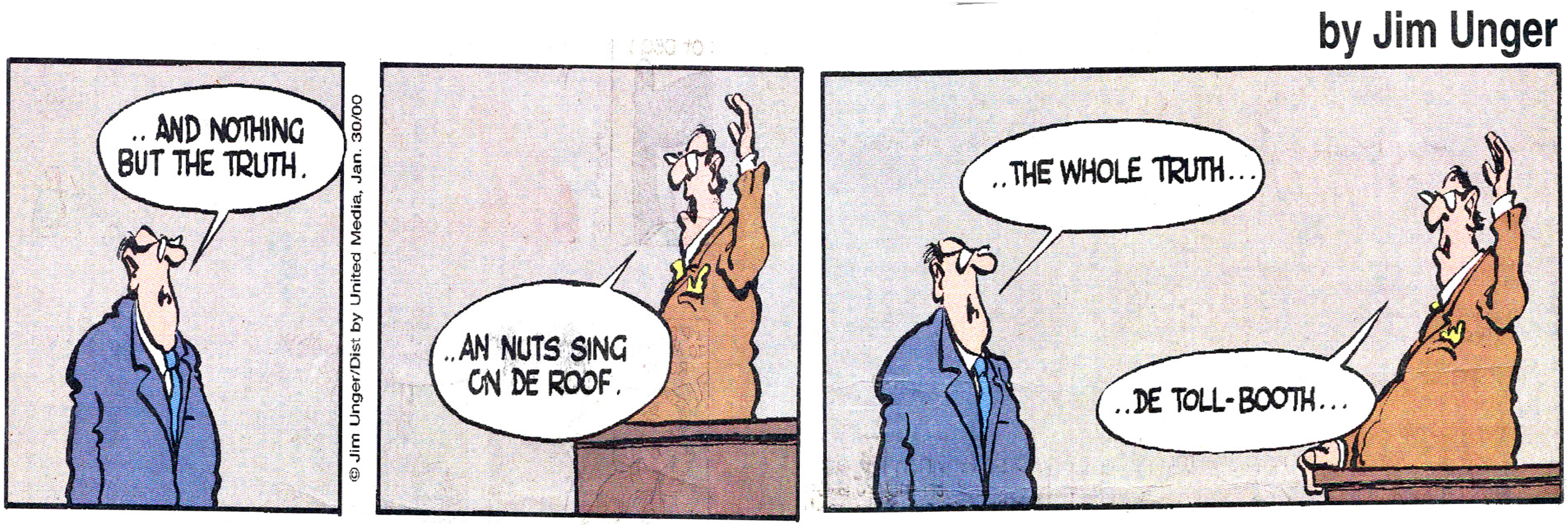
1. X. Huang et. al., Spoken Language Processing, Chapter 11
2. R. Rosenfeld, "Two Decades of Statistical Language Modeling: Where Do We Go from Here?," Proceedings of IEEE, August, 2000
3. Joshua Goodman's (Microsoft Research) Public Presentation Material
4. S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," IEEE ASSP, March 1987
5. R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," ICASSP 1995
6. C.X. Zhai, "Statistical Language Models for Information Retrieval (Synthesis Lectures Series on Human Language Technologies)," Morgan & Claypool Publishers, 2008

What is Language Modeling ?

- Language Modeling (LM) deals with the probability distribution of word sequences, e.g.:

$P(\text{“}\textit{hi}\text{”})=0.01$, $P(\text{“}\textit{and nothing but the truth}\text{”}) \approx 0.001$

$P(\text{“}\textit{and nuts sing on the roof}\text{”}) \approx 0$



From Joshua Goodman’s material

What is Language Modeling ? (cont.)

- For a word sequence $\boldsymbol{W}$, $P(\boldsymbol{W})$ can be decomposed into a product of conditional probabilities:

chain (multiplication) rule

$$P(\boldsymbol{W}) = P(w_1, w_2, \ldots, w_m)$$
$$= P(w_1) P(w_2|w_1) P(w_3|w_1, w_2) \ldots P(w_m|w_1, w_2, \ldots, w_{m-1})$$
$$= P(w_1) \prod_{i=2}^{m} P(w_i|w_1, w_2, \ldots, w_{i-1})$$

  – E.g.: *P*(“*and nothing but the truth*”) = *P*(“*and*”) ×*P*(“*nothing|and*”) × *P*(“*but|and nothing*”) × *P*(“*the|and nothing but*”) × *P*(“*truth|and nothing but the*”)

  – However, it’s impossible to estimate and store $P(w_i|\underbrace{w_1, w_2, \ldots, w_{i-1}}_{\text{History of } w_i})$ if i is large (data sparseness problem etc.)

What is LM Used for ?

- Statistical language modeling attempts to capture the regularities of natural languages
  - Improve the performance of various natural language applications by estimating the probability distribution of various linguistic units, such as words, sentences, and whole documents
  - The first significant model was proposed in 1980s

$$P(\boldsymbol{W}) = P(w_1, w_2, \ldots, w_m)?$$

What is LM Used for ? (cont.)

- Statistical language modeling is most prevailing in many application domains
  - Speech recognition
  - Spelling correction
  - Handwriting recognition
  - Optical character recognition (OCR)
  - Machine translation
  - Document classification and routing
  - Information retrieval

Current Status

- Ironically, the most successful statistical language modeling techniques use very little knowledge of what language is
  - The most prevailing *n*-gram language models take no advantage of the fact that what is being modeled is language

$$P(w_i | w_1, w_2, \ldots, w_{i-1}) \approx P(w_i | \underbrace{w_{i-n+1}, w_{i-n+2}, \ldots, w_{i-1}}_{\text{History of length } n\text{-1}})$$

  - It may be a sequence of arbitrary symbols, with no deep structure, intention, or though behind then
  - F. Jelinek said "*put language back into language modeling*"
    - "Closing remarks" presented at the 1995 Language Modeling Summer Workshop, Baltimore

LM in Speech Recognition

- For a given acoustic observation $X = x_1, x_2, \ldots, x_n$, the goal of speech recognition is to find out the corresponding word sequence $W = w_1, w_2, \ldots, w_m$ that has the maximum posterior probability $P(W|X)$

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X})$$

Bayes classification rule

$$= \arg\max_{\mathbf{W}} \frac{p(\mathbf{X}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{X})}$$

$$= \arg\max_{\mathbf{W}} p(\mathbf{X}|\mathbf{W})P(\mathbf{W})$$

$W = w_1, w_2, \ldots w_i, \ldots, w_m$

where $w_i \in \text{Voc}\{w_1, w_2, \ldots\ldots, w_V\}$

Acoustic Modeling

Language Modeling

Posterior Probability

Prior Probability

The Trigram Approximation

- The trigram modeling assumes that each word depends only on the previous two words (a window of three words total) → Second-order Markov modeling
  - "tri" means three, "gram" means writing
  - E.g.:

    P("*the*|… *whole truth and nothing but*") ≈ *P*("*the*|*nothing but*")

    P("*truth*|… *whole truth and nothing but the*") ≈ *P*("*truth*|*but the*")

  - Similar definition for bigram (a window of two words in total)

- How do we find probabilities?
  - Get real text, and start counting (empirically) !

P("*the* | *nothing but*") ≈*C*["*nothing but the*"]/*C*["*nothing but*"]

count

Probability may be *0*

Maximum Likelihood Estimate (ML/MLE) for LM

- Given a a training corpus T and the language model Λ

$$\text{Corpus} \quad T = w_{1-th}\, w_{2-th} \ldots w_{k-th} \ldots\ldots w_{L-th}$$

$$\text{Vocabulary} \quad W = \{w_1, w_2, \ldots, w_V\}$$

N-grams with same history are collected together

$$p(T|\Lambda) \cong \prod_{w_{k-th}} p\left(w_{k-th} \,\middle|\, \text{history of } w_{k-th}\right)$$

$$= \prod_h \prod_{w_i} \lambda_{hw_i}^{N_{hw_i}} \qquad \forall h \in T, \quad \sum_{w_j} \lambda_{hw_j} = 1$$

- – Essentially, the distribution of the sample counts N_{hw_i} with the same history h referred as a multinominal (polynominal) distribution

$$\forall h \in T, \quad P\left(N_{hw_1}, \ldots, N_{hw_V}\right) = \frac{N_h!}{\prod_{w_i} N_{hw_i}!} \prod_{w_i} \lambda_{hw_i}^{N_{hw_i}}, \qquad \sum_{w_i} N_{hw_i} = N_h \text{ and } \sum_{w_j} \lambda_{hw_j} = 1$$

$$\text{where } p(w_i|h) = \lambda_{hw_i}, \; N_{hw_i} = C[hw_i], \; N_h = \sum_{w_i} C[hw_i] = C[h] \text{ in corpus } T$$

…陳水扁 總統 訪問 美國 紐約 … 陳水扁 總統 在 巴拿馬 表示 …

P(總統|陳水扁)=?

Maximum Likelihood Estimate (ML/MLE) for LM (cont.)

- Take logarithm of $p(T|\Lambda)$, we have

$$\Phi(\Lambda) = \log p(T|\Lambda) = \sum_h \sum_{w_i} N_{hw_i} \log \lambda_{hw_i}$$

- For any pair (h, w_j), try to maximize $\Phi(\Lambda)$ and subject to $\sum_{w_j} \lambda_{hw_j} = 1, \forall h$

$$\frac{d \log x}{dx} = \frac{1}{x}$$

$$\Rightarrow \overline{\Phi}(\Lambda) = \Phi(\Lambda) + \sum_h l_h \left(\sum_{w_j} \lambda_{hw_j} - 1 \right)$$

$$\frac{\partial \overline{\Phi}(\Lambda)}{\partial \lambda_{hw_i}} = \frac{\partial \left[\sum_h \sum_{w_i} N_{hw_i} \log \lambda_{hw_i} + \sum_h l_h \left(\sum_{w_j} \lambda_{hw_j} - 1 \right) \right]}{\partial \lambda_{hw_i}}$$

$$\Rightarrow \frac{N_{hw_i}}{\lambda_{hw_i}} + l_h = 0 \Rightarrow \frac{N_{hw_1}}{\lambda_{hw_1}} = \frac{N_{hw_2}}{\lambda_{hw_2}} = \ldots\ldots = \frac{N_{hw_V}}{\lambda_{hw_V}} = -l_h$$

$$\frac{1}{2} = \frac{2}{4} = \frac{3}{6} = \frac{1+2+3}{2+4+6}$$

$$\Rightarrow \frac{\sum_{w_s} N_{hw_s}}{\sum_{w_j} \lambda_{hw_j}} = -l_h \Rightarrow l_h = -\sum_{w_s} N_{hw_s} = -N_h$$

$$\therefore \hat{\lambda}_{hw_i} = \frac{N_{hw_i}}{N_h} = \frac{C[hw_i]}{C[h]}$$

Lagrange Multiplier: http://www.slimy.com/~steuard/teaching/tutorials/Lagrange.html

Main Issues for LM

- Evaluation
  - How can you tell a good language model from a bad one
  - Run a speech recognizer or adopt other statistical measurements
- Smoothing
  - Deal with data sparseness of real training data
  - Various approaches have been proposed
- Caching
  - If you say something, you are likely to say it again later
  - Adjust word frequencies observed in the current conversation
- Clustering
  - Group words with similar properties (similar semantic or grammatical) into the same class
  - Another efficient way to handle the data sparseness problem

Evaluation

- Two most common metrics for evaluation a language model
  - Word Recognition Error Rate (WER)
  - Perplexity (PP)

- Word Recognition Error Rate
  - Requires the participation of a speech recognition system (slow!)
  - Need to deal with the combination of acoustic probabilities and language model probabilities (penalizing or weighting between them)

Evaluation (cont.)

- Perplexity
  - Perplexity is geometric average inverse language model probability (measure language model difficulty, not acoustic difficulty/confusability)

$$PP\left(\boldsymbol{W} = w_1, w_2, \ldots, w_m\right) = \sqrt[m]{\frac{1}{P(w_1)} \cdot \prod_{i=2}^{m} \frac{1}{P(w_i \mid w_1, w_2, \ldots, w_{i-1})}}$$

  - Can be roughly interpreted as the geometric mean of the branching factor of the text when presented to the language model

w_{i-2}, w_{i-1}

  - For trigram modeling:

$$PP\left(\boldsymbol{W} = w_1, w_2, \ldots, w_m\right) = \sqrt[m]{\frac{1}{P(w_1)} \cdot \frac{1}{P(w_2 \mid w_1)} \cdot \prod_{i=3}^{m} \frac{1}{P(w_i \mid w_{i-2}, w_{i-1})}}$$

Evaluation (cont.)

- More about Perplexity
  - Perplexity is an indication of the complexity of the language if we have an accurate estimate of $P(\boldsymbol{W})$
  - A language with higher perplexity means that the number of words branching from a previous word is larger on average
  - A langue model with perplexity *L* has roughly the same difficulty as another language model in which every word can be followed by *L* different words with equal probabilities

  - Examples:
    - Ask a speech recognizer to recognize digits: “0, 1, 2, 3, 4, 5, 6, 7, 8, 9” – easy – perplexity $\approx$10

    - Ask a speech recognizer to recognize names at a large institute (10,000 persons) – hard – perplexity $\approx$ 10,000

Evaluation (cont.)

- More about Perplexity (Cont.)
  - Training-set perplexity: measures how the language model fits the training data
  - Test-set perplexity: evaluates the generalization capability of the language model
    - When we say perplexity, we mean “test-set perplexity”

Evaluation (cont.)

- Is a language model with lower perplexity is better?
  - The true (optimal) model for data has the lowest possible perplexity
  - The lower the perplexity, the closer we are to the true model
  - Typically, perplexity correlates well with speech recognition word error rate
    - Correlates better when both models are trained on same data
    - Doesn’t correlate well when training data changes
  - The 20,000-word continuous speech recognition for Wall Street Journal (WSJ) task has a perplexity about 128 ~ 176 (trigram)
  - The 2,000-word conversational Air Travel Information System (ATIS) task has a perplexity less than 20

Evaluation (cont.)

- The perplexity of bigram with different vocabulary size

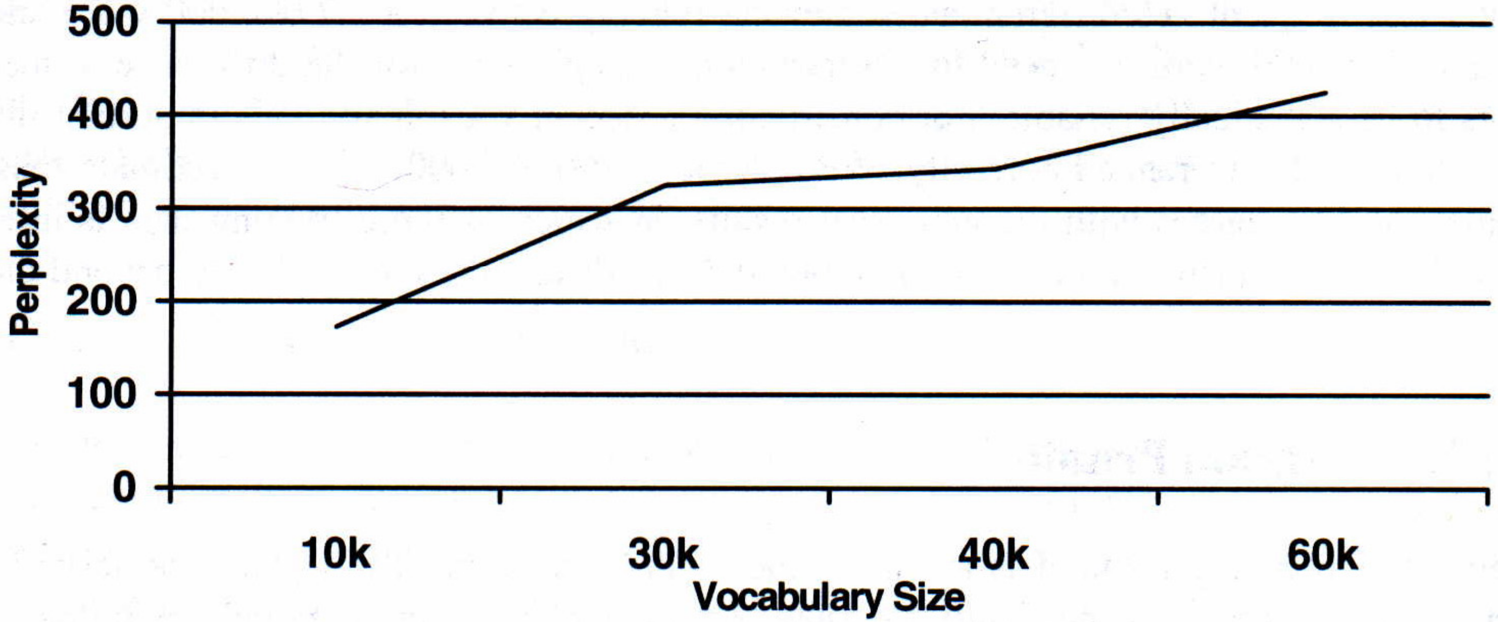


Figure 11.6 The perplexity of bigram with different vocabulary sizes. The training set consists of 500 million words derived from various sources, including newspapers and email. The test set comes from the whole Microsoft Encarta, an encyclopedia that has a wide coverage of different topics.

Evaluation (cont.)

- A rough rule of thumb (by Rosenfeld)
  - Reduction of 5% in perplexity is usually not practically significant
  - A 10% ~ 20% reduction is noteworthy, and usually translates into some improvement in application performance
  - A perplexity improvement of 30% or more over a good baseline is quite significant

Vocabulary	Perplexity	WER
zero |one |two |three |four |five |six |seven |eight |nine	10	5
John |tom |sam |bon |ron | |susan |sharon |carol |laura |sarah	10	7
bit |bite |boot |bait |bat |bet |beat |boat |burt |bart	10	9

Tasks of recognizing 10 isolated-words using IBM ViaVoice

Perplexity cannot always reflect the difficulty of a speech recognition task

Smoothing

- Maximum likelihood (ML) estimate of language models has been shown previously, e.g.:
  – Trigam probabilities

$$P_{ML}(z \mid xy) = \frac{C[xyz]}{\sum_w C[xyw]} = \frac{C[xyz]}{\underset{\text{count}}{\underbrace{C}}[xy]}$$

  – Bigram probabilities

$$P_{ML}(y \mid x) = \frac{C[xy]}{\sum_w C[xw]} = \frac{C[xy]}{C[x]}$$

Smoothing (cont.)

- Data Sparseness
  - Many actually possible events (word successions) in the test set may not be well observed in the training set/data
    - E.g. bigram modeling

$P(read|Mulan)=0 \Rightarrow P(Mulan\ read\ a\ book)=0$

$\Rightarrow P(W)=0 \qquad \Rightarrow P(X|W)P(W)=0$

  - Whenever a string $\boldsymbol{W}$ such that $P(\boldsymbol{W})=0$ occurs during speech recognition task, an error will be made

Smoothing (cont.)

- Operations of smoothing
  - Assign all strings (or events/word successions) a nonzero probability if they never occur in the training data
  - Tend to make distributions flatter by adjusting lower probabilities upward and high probabilities downward

Smoothing: Simple Models

- Add-one smoothing
  - For example, pretend each trigram occurs once more than it actually does

$$P_{smooth}(z \mid xy) \approx \frac{C[xyz]+1}{\sum_{w}(C[xyw]+1)} = \frac{C[xyz]+1}{C[xy]+V}$$

V : number of total vocabulary words

- Add delta smoothing

$$P_{smooth}(z \mid xy) \approx \frac{C[xyz]+\delta}{C[xy]+V\delta}$$

Should the word "unicorn" receive the same probability mass as the word "animal" if they are both unseen in the training data ?

Works badly. DO NOT DO THESE TWO (Joshua Goodman said)

Smoothing: Back-Off Models

- The general form for *n*-gram back-off

$$
\begin{aligned}
&P_{smooth}(w_i \mid w_{i-n+1},\ldots,w_{i-1}) \\
&= \begin{cases} \hat{P}(w_i \mid w_{i-n+1},\ldots,w_{i-1}) & \text{if } C[w_{i-n+1},\ldots,w_{i-1},w_i] > 0 \\ \alpha(w_{i-n+1},\ldots,w_{i-1}) \cdot P_{smooth}(w_i \mid w_{i-n+2},\ldots,w_{i-1}) & \text{if } C[w_{i-n+1},\ldots,w_{i-1},w_i] = 0 \end{cases}
\end{aligned}
$$

  – $\alpha(w_{i-n+1},\ldots,w_{i-1})$: normalizing/scaling factor chosen to make the conditional probability sum to 1

    - I.e., $\sum_{w_i} P_{smooth}(w_i \mid w_{i-n+1},\ldots,w_{i-1}) = 1$

For example, $$\alpha(w_{i-n+1},\ldots,w_{i-1}) = \frac{1 - \sum_{w_i, C[w_{i-n+1},\ldots,w_{i-1},w_i]>0} \overbrace{\hat{P}(w_i \mid w_{i-n+1},\ldots,w_{i-1})}^{n\text{-gram}}}{\sum_{w_j, C[w_{i-n+1},\ldots,w_{i-1},w_i]=0} \underbrace{P_{smooth}(w_j \mid w_{i-n+2},\ldots,w_{i-1})}_{\text{smoothed } (n\text{-}1)\text{-gram}}}$$

Smoothing: Interpolated Models

- The general form for Interpolated n-gram back-off

$$P_{smooth}(w_i \mid w_{i-n+1},\ldots,w_{i-1})$$
$$= \lambda(w_{i-n+1},\ldots,w_{i-1})P_{ML}(w_i \mid w_{i-n+1},\ldots,w_{i-1}) + (1-\lambda(w_{i-n+1},\ldots,w_{i-1}))P_{smooth}(w_i \mid w_{i-n+2},\ldots,w_{i-1})$$

$$P_{ML}(w_i \mid w_{i-n+1},\ldots,w_{i-1}) = \frac{C[w_{i-n+1},\ldots,w_{i-1},w_i]}{C[w_{i-n+1},\ldots,w_{i-1}]}$$

count

- The key difference between backoff and interpolated models
  - For *n*-grams with nonzero counts, interpolated models use information from lower-order distributions while back-off models do not
  - Moreover, in interpolated models, *n*-grams with the same counts can have different probability estimates

Clustering

- Class-based language Models
  - Define classes for words that exhibit similar semantic or grammatical behavior

WEEKDAY = *Sunday, Monday, Tuesday, …*

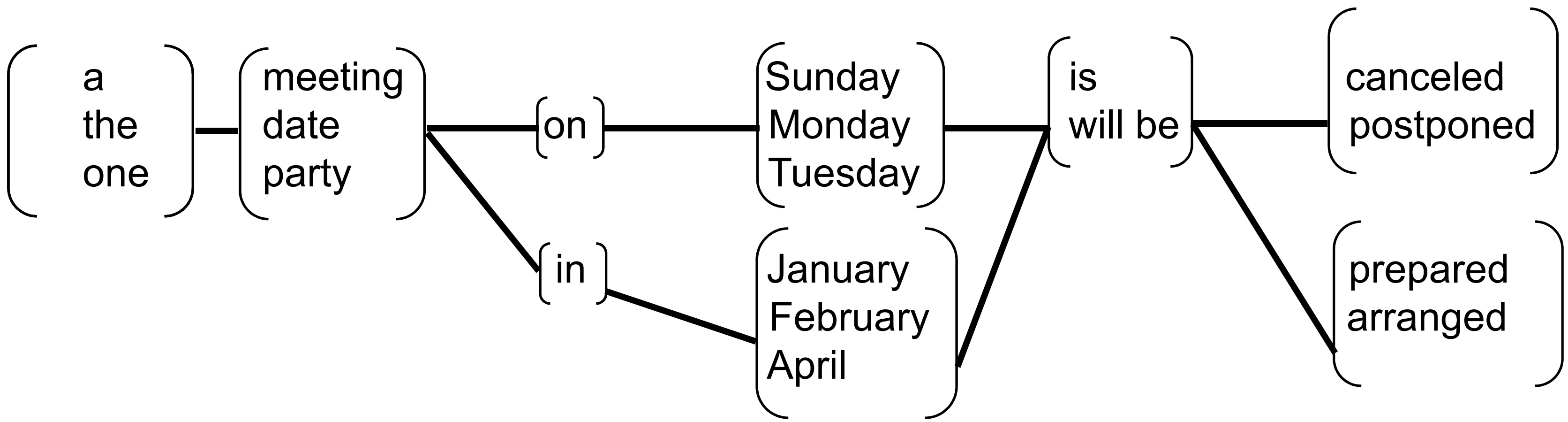
MONTH = *January, February, April, May, June, …*

EVENT=*meeting, class, party, …*

- E.g., $P(Tuesday|\ party\ on)$ is similar to $P(Monday|\ party\ on)$?

Clustering (cont.)

- A word may belong to more than one class and a class may contain more than one word (many-to-many mapping)



Clustering (cont.)

- The *n*-gram model can be computed based on the previous *n*-1 classes
  - If trigram approximation and unique mappings from words to word classes are used (deterministic class assignment)

$$P(w_i \mid w_{i-n+1} \ldots w_{i-1}) \approx P(w_i \mid w_{i-2}, w_{i-1})$$

$$P(w_i \mid w_{i-2}, w_{i-1}) \approx P(w_i \mid Class(w_i))P(Class(w_i) \mid Class(w_{i-2})Class(w_{i-1}))$$

$Class(w_i)$: the class which w_i belongs to

  - Empirically estimate the probabilities

$$P(w_i \mid Class(w_i)) = \frac{C[w_i]}{C[Class(w_i)]}$$

$$P(Class(w_i) \mid Class(w_{i-2})Class(w_{i-1})) = \frac{C[Class(w_{i-2})Class(w_{i-1})Class(w_i)]}{C[Class(w_{i-2})Class(w_{i-1})]}$$

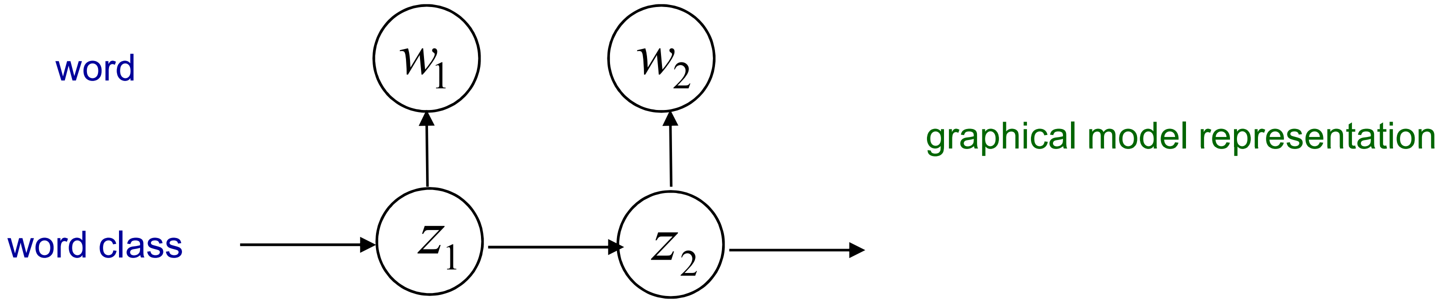
    - Further probability smoothing is also needed

Clustering (cont.)

- Clustering is another way to battle data sparseness problem (smoothing of the language model)
- For general-purpose large vocabulary dictation application, class-based *n*-grams have not significant improved recognition accuracy
  – Mainly used as a back-off model to complement the lower-order *n*-grams for better smoothing
- For limited (or narrow discourse) domain speech recognition, the class-based *n*-gram is very helpful
  – Because the class can efficiently encode semantic information for improved keyword-spotting and speech understanding accuracy
  – Good results are often achieved by manual clustering of semantic categories

Class-based Bigram Model Jelinek et al., 1992

- A kind of first-order hidden Markov model



  - Nondeterministic class assignment

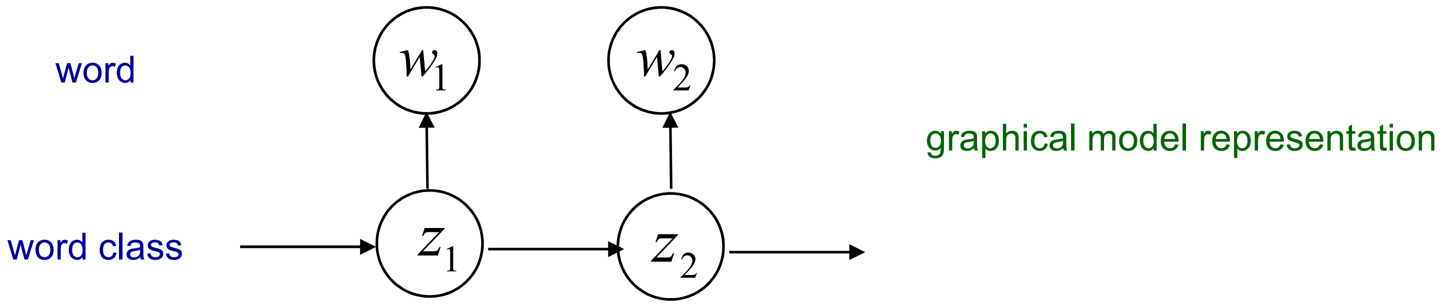
$$P(w_2 \mid w_1) = \sum_{z_1} \sum_{z_2} P(z_1 \mid w_1) \cdot P(z_2 \mid z_1) \cdot P(w_2 \mid z_2)$$

  - Deterministic class assignment

$$P(w_2 \mid w_1) = P(z_2 \mid z_1) \cdot P(w_2 \mid z_2)$$

    - Needing estimation of class bigram and word unigram probabilities

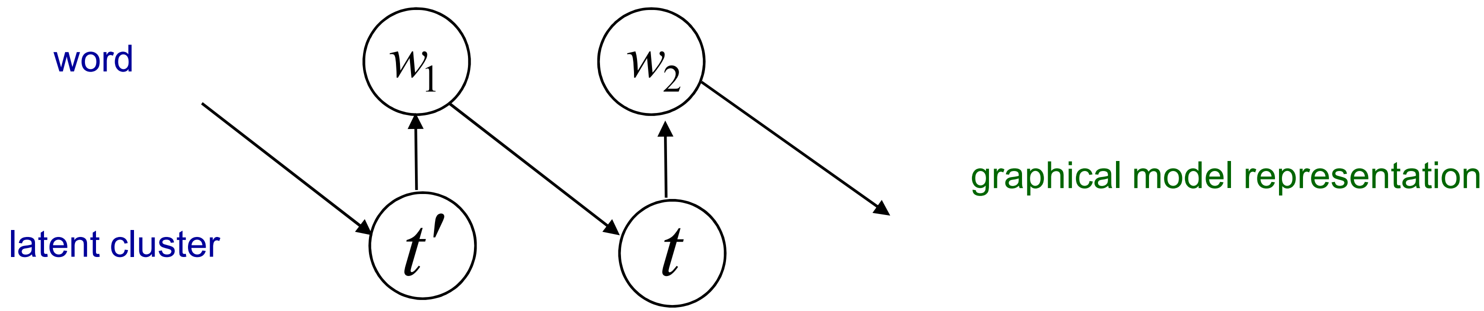
Class-based Bigram Model: Explanation



$$
\begin{aligned}
P(w_2 \mid w_1) &= \sum_{z_1} \sum_{z_2} P(w_2, z_1, z_2 | w_1) \\
&= \sum_{z_1} \sum_{z_2} P(z_1 | w_1) P(z_2 | z_1, w_1) P(w_2 | z_2, z_1, w_1) \\
&\approx \sum_{z_1} \sum_{z_2} P(z_1 | w_1) P(z_2 | z_1) P(w_2 | z_2)
\end{aligned}
$$

Aggregate Markov Model

- An alternative approach for class-based bigram LMs



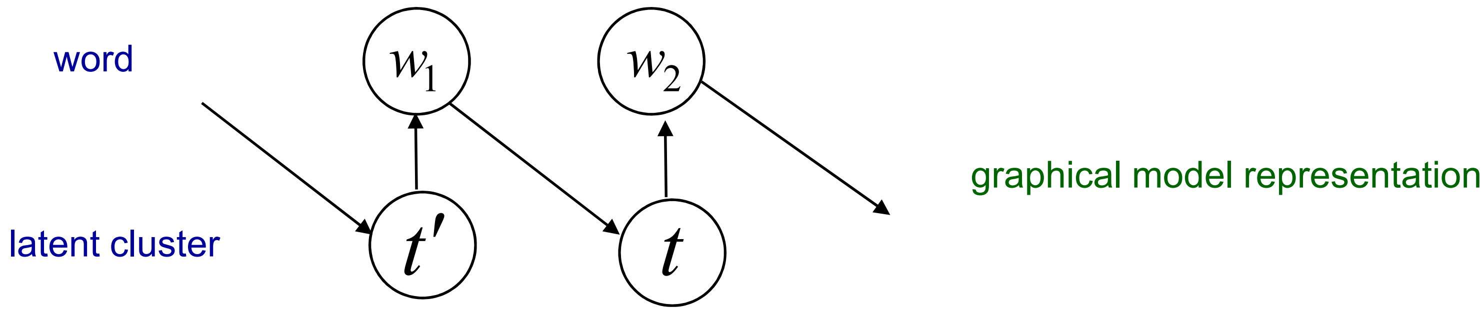
$$P(w_2 \mid w_1) = \sum_t P(w_2 \mid t) P(t \mid w_1)$$

  – Models trained by maximizing the log-likelihood of the training corpus

$$l = \sum_{w_1, w_2} n(w_1, w_2) \ln P(w_2 \mid w_1)$$

L. Saul and F. Pereira, "Aggregate and mixed-order Markov models for statistical language processing," in *Proc. EMNLP 1997*.

Aggregate Markov Model: Explanation



$$
\begin{aligned}
P(w_2 \mid w_1) &= \sum_{t'} \sum_{t} P(w_2, t', t \mid w_1) \\
&= \sum_{t'} \sum_{t} P(t' \mid w_1) P(t \mid t', w_1) P(w_2 \mid t, t', w_1) \\
&\approx \sum_{t'} \sum_{t} P(t' \mid w_1) P(t \mid w_1) P(w_2 \mid t) \\
&= \sum_{t} \sum_{t} P(w_2 \mid t) P(t \mid w_1)
\end{aligned}
$$

Caching (cont.)

- The basic idea of cashing is to accumulate *n*-grams dictated so far in the current document/conversation and use these to create dynamic *n*-grams model

- Trigram interpolated with unigram cache

$$P_{cache}(z \mid xy) \approx \lambda P_{smooth}(z \mid xy) + (1-\lambda) P_{cache}(z \mid history)$$

$$history: \text{document/conversation dictated so far}$$

$$P_{cache}(z \mid history) = \frac{C[z \in history]}{length[history]} = \frac{C[z \in history]}{\sum_w C[w \in history]}$$

- Trigram interpolated with bigram cache

$$P_{cache}(z \mid xy) \approx \lambda P_{smooth}(z \mid xy) + (1-\lambda) P_{cache}(z \mid y, history)$$

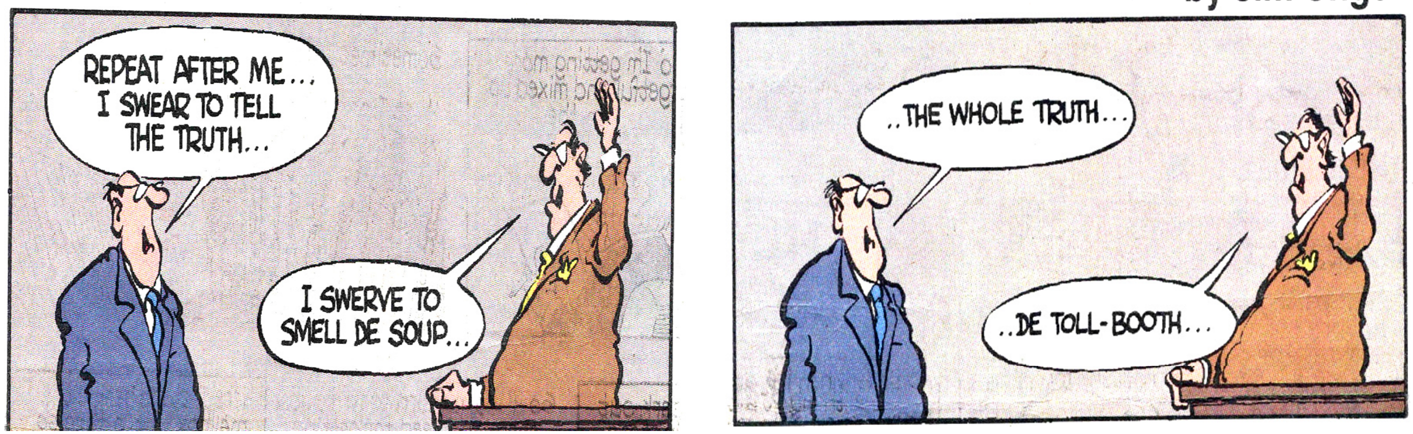
$$P_{cache}(z \mid y, history) = \frac{C[yz \in history]}{C[y \in history]}$$

Caching (cont.)

- Real Life of Caching
  - Someone says “I swear to tell the truth”
  - System hears “I swerve to smell the soup”

  Cache remembers!
  - Someone says “The whole truth”, and, with cache, system hears “The toll booth.” – errors are locked in
- Caching works well when users corrects as they go, poorly or even hurts without correction

by Jim Unger



From Joshua Goodman’s material

Known Weakness in Current LM

- Brittleness Across Domain
  - Current language models are extremely sensitive to changes in the style or topic of the text on which they are trained
    - E.g., conversations vs. news broadcasts, fictions vs. politics
  - Language model adaptation
    - In-domain or contemporary text corpora/speech transcripts
    - Static or dynamic adaptation
    - Local contextual (*n*-gram) or global semantic/topical information
- False Independence Assumption
  - In order to remain trainable, the *n*-gram modeling assumes the probability of next word in a sentence depends only on the identity of last *n*-1 words
    - *n*-1-order Markov modeling

LM Integrated into Speech Recognition

- Theoretically,

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})P(\boldsymbol{X}|\boldsymbol{W})$$

- Practically, language model is a better predictor while acoustic probabilities aren't "real" probabilities
  - Penalize insertions

$$\hat{\boldsymbol{W}} = \arg\max_{\boldsymbol{W}} P(\boldsymbol{W})^{\alpha} P(\boldsymbol{X}|\boldsymbol{W}) \cdot \beta^{length(\boldsymbol{W})}$$

, where α, β can be empirically decided

- E.g., $\alpha = 8$

Good-Turing Estimate

- First published by Good (1953) while Turing is acknowledged

Use the notation *m*-grams instead of *n*-grams here

- A smoothing technique to deal with infrequent *m*-grams (*m*-gram smoothing), but it usually needs to be used together with other back-off schemes to achieve good performance

- How many words were seen once? Estimate for how many are unseen. All other estimates are adjusted (downward) to give probabilities for unseen

Good-Turing Estimate (cont.)

- For any *m*-gram, $\boldsymbol{a} = w_1^m$,that occurs *r* times ($r = c[w_1^m]$), we pretend it occurs *r** times ($r^* = c^*[w_1^m]$),

$$r^* = (r+1)\frac{n_{r+1}}{n_r},$$

A new frequency count

where n_r is the number of m-grams that occurs exactly r times in the training data

Not a conditional probability !

  – The probability estimate for a *m*-gram, $\boldsymbol{a} = w_1^m$, with *r* counts

$$P_{GT}(\boldsymbol{a}) = \frac{r^*}{N}, \text{ where } N \text{ is the size (total word counts) of the training data}$$

- The size (word counts) of the training data remains the same

$$\text{Let } \sum_{r=1}^{\infty} r \cdot n_r = N$$

$$\tilde{N} = \sum_{r=0}^{\infty} r^* \cdot n_r = \sum_{r=0}^{\infty} (r+1) \cdot n_{r+1} = \sum_{r'=1}^{\infty} r' \cdot n_{r'} = N \quad (\text{set } r' = r+1)$$

Good-Turing Estimate (cont.)

- It follows from above that the total probability estimate used for the set of *m*-grams that actually occur in the sample is

$$\sum_{w_1^m,\, c[w_1^m] > 0} P_{GT}\left(w_1^m\right) = 1 - \frac{n_1}{N}$$

- The probability of observing some previously unseen *m*-grams is

$$\sum_{w_1^m,\, c[w_1^m] = 0} P_{GT}\left(w_1^m\right) = \frac{n_1}{N}$$

  - Which is just a fraction of the singletons (*m*-grams occurring only once) in the text sample

Good-Turing Estimate: Example

- Imagine you are fishing. You have caught 10 Carp (鯉魚), 3 Cod (鱈魚), 2 tuna(鮪魚), 1 trout(鱒魚), 1 salmon(鮭魚), 1 eel(鰻魚)
- How likely is it that next species is new?
  - $p_0=n_1/N=3/18=$ 1/6
- How likely is eel? 1^*
  - $n_1=3$, $n_2=1$
  - $1^*=(1+1)\times 1/3=2/3$
  - $P(eel)=1^*/N=(2/3)/18=$ 1/27
- How likely is tuna? 2^*
  - $n_2=1$, $n_3=1$
  - $2^*=(2+1)\times 1/1=3$
  - $P(tuna)=2^*/N=3/18=$ 1/6
- But how likely is Cod? 3^*
  - Need a smoothing for n_4 in advance

Good-Turing Estimate (cont.)

- The Good-Turing estimate may yield some problems when $n_{r+1}=0$
  - An alternative strategy is to apply Good-Turing to the *m*-grams (events) seen at most *k* times, where *k* is a parameter chosen so that $n_{r+1} \neq 0$, $r=1,\ldots,k$

Good-Turing Estimate (cont.)

- For Good-Turing estimate, it may happen that an *m*-gram (event) occurring *k* times takes on a higher probability than an event occurring *k*+1 times
  - The choice of *k* may be selected in an attempt to overcome such a drawback

$$\hat{P}_{GT}(a_k) = \frac{k+1}{N} \cdot \frac{n_{k+1}}{n_k}$$

$$\hat{P}_{GT}(a_{k+1}) = \frac{k+2}{N} \cdot \frac{n_{k+2}}{n_{k+1}}$$

  - Experimentally, *k* ranging from 4 to 8 will not allow the about condition to be true (for $r \le k$)

$$\hat{P}_{GT}(a_k) < \hat{P}_{GT}(a_{k+1})$$
$$\Rightarrow (k+1) \cdot n_{k+1}^2 - n_k \cdot n_{k+2}(k+2) < 0$$

Katz Back-off Smoothing

1987

- Extend the intuition of the Good-Turing estimate by adding the combination of higher-order language models with lower-order ones
  - E.g., bigrams and unigram language models

- Larger counts are taken to be reliable, so they are not discounted
  - E.g., for frequency counts $r > k$

- Lower counts are discounted, with total reduced counts assigned to unseen events, based on the Good-Turning estimate
  - E.g., for frequency counts $r \leq k$

Katz Back-off Smoothing (cont.)

- Take the bigram (m-gram, m=2) counts for example:

$$C^*[w_{i-1}w_i] = \begin{cases} r & \text{if } r > k \\ d_r r & \text{if } k \ge r > 0 \\ \beta(w_{i-1}) P_{Katz}(w_i) & \text{if } r = 0 \end{cases}$$

1. $r = C[w_{i-1}w_i]$

2. $d_r \approx \frac{r^*}{r}$: discount constant, satisfying to the following two equations

Note: d_r should be calculated for different m-gram counts and different m-gram histories, e.g., w_{i-1} here

$$d_r = \mu \frac{r^*}{r} \quad \text{and} \quad \sum_{r=1}^{k} n_r (1 - d_r) r = n_1$$

3. $$\beta(w_{i-1}) = \frac{\sum_{w_i} C[w_{i-1}w_i] - \sum_{w_i : C[w_{i-1}w_i] > 0} C^*[w_{i-1}w_i]}{\sum_{w_i : C[w_{i-1}w_i] = 0} \boxed{P_{Katz}(w_i)}}$$

Assume lower level LM probability has been defined

Katz Back-off Smoothing (cont.)

- Derivation of the discount constant: $$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$$

Two constraints are imposed

$$\begin{cases} \sum_{r=1}^{k} n_r (1-d_r) r = n_1 \\ d_r = \mu \frac{r^*}{r} \end{cases}$$

Also, the following equation is known

$$\sum_{r=1}^{k} r n_r - \sum_{r=1}^{k} r^* n_r = n_1 - (k+1) n_{k+1}$$

$$\Rightarrow \sum_{r=1}^{k} (r n_r - r^* n_r) = n_1 - (k+1) n_{k+1}$$

$$\Rightarrow \frac{\sum_{r=1}^{k} n_r (r - r^*)}{n_1 - (k+1) n_{k+1}} = 1$$

Both sides multiplied by n_1

$$\Rightarrow \frac{\sum_{r=1}^{k} n_r (r - r^*) n_1}{n_1 - (k+1) n_{k+1}} = n_1$$

$$\Rightarrow \sum_{r=1}^{k} \frac{n_r (r - r^*) n_1}{n_1 - (k+1) n_{k+1}} = n_1 \quad (1)$$

$$\sum_{r=1}^{k} n_r [1 - d] r = n_1$$

$$\Rightarrow \sum_{r=1}^{k} n_r \left[1 - \mu \frac{r^*}{r}\right] r = n_1$$

$$\Rightarrow \sum_{r=1}^{k} n_r [r - \mu r^*] = n_1 \quad (2)$$

If equations (1) and (2) are related together, we have

$$\Rightarrow \frac{(r - r^*) n_1}{n_1 - (k+1) n_{k+1}} = r - u r^* \quad (3)$$

$$\Rightarrow \frac{(r - r^*) n_1}{r[n_1 - (k+1) n_{k+1}]} = 1 - u \frac{r^*}{r} = 1 - d_r$$

Both sides divided by r

$$\Rightarrow d_r = 1 - \frac{(r - r^*) n_1}{r[n_1 - (k+1) n_{k+1}]}$$

Katz Back-off Smoothing (cont.)

- Derivation of the discount constant d_r

$$\Rightarrow d_r = 1 - \frac{(r - r^*)n_1}{r\left[n_1 - (k+1)n_{k+1}\right]}$$

$$= \frac{r\left[n_1 - (k+1)n_{k+1}\right] - (r - r^*)n_1}{r\left[n_1 - (k+1)n_{k+1}\right]}$$

$$= \frac{r^* n_1 - r(k+1)n_{k+1}}{r\left[n_1 - (k+1)n_{k+1}\right]}$$

$$\boxed{\therefore \; d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}}$$

the $r \cdot n_1$ term in the nominator is eliminated

Both the nominator and denominator are divided by $r \cdot n_1$

Katz Back-off Smoothing (cont.)

- Take the conditional probabilities of bigrams (*m*-gram, *m*=2) for example:

$$P_{Katz}(w_i|w_{i-1}) = \begin{cases} C[w_{i-1}, w_i]/C[w_{i-1}] & \text{if } r > k \\ d_r\, C[w_{i-1}, w_i]/C[w_{i-1}] & \text{if } k \geq r > 0 \\ \alpha(w_{i-1}) P_{Katz}(w_i) & \text{if } r = 0 \end{cases}$$

1. discount constant

$$d_r = \frac{\dfrac{r^*}{r} - \dfrac{(k+1)\, n_{k+1}}{n_1}}{1 - \dfrac{(k+1)\, n_{k+1}}{n_1}}$$

2. normalizing constant

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : C[w_{i-1} w_i] > 0} P_{Katz}(w_i|w_{i-1})}{\sum_{w_i : C[w_{i-1} w_i] = 0} P_{Katz}(w_i)}$$

Katz Back-off Smoothing: Example

- A small vocabulary consists of only five words, i.e., $V = \{w_1, w_2, ..., w_5\}$. The frequency counts for word pairs started with w_1 are:

  $$C[w_1, w_2] = 3, C[w_1, w_3] = 2, \ C[w_1, w_4] = 1, C[w_1, w_1] = C[w_1, w_5] = 0$$

  , and the word frequency counts are:

  $$C[w_1] = 6, C[w_2] = 8, C[w_3] = 10, C[w_4] = 6, C[w_5] = 4$$

  Katz back-off smoothing with Good-Turing estimate is used here for word pairs with frequency counts equal to or less than two. Show the conditional probabilities of word bigrams started with w_1 , i.e.,

  $$P_{Katz}(w_1|w_1), P_{Katz}(w_2|w_1), ..., P_{Katz}(w_5|w_1) \ ?$$

Katz Back-off Smoothing: Example (cont.)

$r^* = (r+1)\dfrac{n_{r+1}}{n_r}$, where n_r is the number of n-grams that occurs exactly r times in the training data

$$d_r = \frac{\dfrac{r^*}{r} - \dfrac{(k+1)n_{k+1}}{n_1}}{1 - \dfrac{(k+1)n_{k+1}}{n_1}}$$

$$\therefore P_{Katz}(w_2|w_1) = P_{ML}(w_2|w_1) = \frac{3}{6} = \frac{1}{2}$$

$$1^* = (1+1)\cdot\frac{1}{1} = 2, \quad 2^* = (2+1)\cdot\frac{1}{1} = 3$$

$$d_2 = \frac{\dfrac{3}{2} - \dfrac{(2+1)\cdot 1}{1}}{1 - \dfrac{(2+1)\cdot 1}{1}} = \frac{\dfrac{3}{2} - 3}{-2} = \frac{3}{4}, \quad d_1 = \frac{\dfrac{2}{1} - \dfrac{(2+1)\cdot 1}{1}}{1 - \dfrac{(2+1)\cdot 1}{1}} = \frac{2-3}{1-3} = \frac{1}{2}$$

For $r = 2 \Rightarrow P_{Katz}(w_3|w_1) = d_2 \cdot P_{ML}(w_3|w_1) = \frac{3}{4}\cdot\frac{2}{6} = \frac{1}{4}$

For $r = 1 \Rightarrow P_{Katz}(w_4|w_1) = d_1 P_{ML}(w_4|w_1) = \frac{1}{2}\cdot\frac{1}{6} = \frac{1}{12}$

$$\alpha(w_1) = \frac{1 - \dfrac{1}{2} - \dfrac{1}{4} - \dfrac{1}{12}}{\dfrac{6}{34} + \dfrac{4}{34}} = \frac{34}{10}\cdot\frac{2}{12}$$

Noteice that $P_{Katz}(w) = P_{ML}(w)$ here

For $r = 0 \Rightarrow P_{Katz}(w_1|w_1) = \alpha(w_1)\cdot P_{ML}(w_1) = \frac{34}{10}\cdot\frac{2}{12}\cdot\frac{6}{34} = \frac{1}{10}$

$$P_{Katz}(w_5|w_1) = \alpha(w_1)\cdot P_{ML}(w_5) = \frac{34}{10}\cdot\frac{2}{12}\cdot\frac{4}{34} = \frac{1}{15}$$

And $P_{Katz}(w_1|w_1) + P_{Katz}(w_2|w_1) + ... + P_{Katz}(w_5|w_1) = 1$

Kneser-Ney Back-off Smoothing

1995

- Absolute discounting without the Good-Turning estimate
- The lower *n*-gram (back-off *n*-gram) is not proportional to the number of occurrences of a word but instead to the number of different words that it follows, e.g.:
  - In "San Francisco", "Francisco" only follows a single history, it should receive a low unigram probability

San Salvador ?

At Salvador
At Francisco

P(Salvador | At) ?
P(*Francisco* | *At*) ?

  - In "US dollars", "TW dollars" etc., "dollars" should receive a high unigram probability

C(US dollars)=200
C(HK dollars)=100
C(TW dollars)=25
.
.

Kneser-Ney Back-off Smoothing (cont.)

- Take the conditional probabilities of bigrams (m-gram, m=2) for example:

$$0 \le D \le 1$$

$$P_{KN}\left(w_i \middle| w_{i-1}\right) = \begin{cases} \dfrac{\max\left\{C\left[w_{i-1}, w_i\right] - D,\ 0\right\}}{C\left[w_{i-1}\right]} & \text{if } C\left[w_{i-1}, w_i\right] > 0 \\ \alpha\left(w_{i-1}\right) P_{KN}\left(w_i\right) & \text{otherwise} \end{cases}$$

1. $P_{KN}\left(w_i\right) = \boldsymbol{C}\left[\bullet\, w_i\right] / \sum_{w_j} \boldsymbol{C}\left[\bullet\, w_j\right],$

$$\boldsymbol{C}\left[\bullet\, w_i\right] \text{ is the unique words preceding } w_i$$

2. normalizing constant

$$\alpha\left(w_{i-1}\right) = \frac{1 - \sum_{w_i : C\left[w_{i-1} w_i\right] > 0} \dfrac{\max\left\{C\left[w_{i-1} w_i\right] - D,\ 0\right\}}{C\left[w_{i-1}\right]}}{\sum_{w_i : C\left[w_{i-1} w_i\right] = 0} P_{KN}\left(w_i\right)}$$

Kneser-Ney Back-off Smoothing: Example

- Given a text sequence as the following:

  SABCAABBCS (*S* is the sequence's start/end marks)

  Show the corresponding unigram conditional probabilities:

$$C[\bullet A] = 3 \qquad C[\bullet B] = 2$$

$$C[\bullet C] = 1 \qquad C[\bullet S] = 1$$

$$\Rightarrow P_{KN}(A) = \frac{3}{7}$$

$$P_{KN}(B) = \frac{2}{7}$$

$$P_{KN}(C) = \frac{1}{7}$$

$$P_{KN}(S) = \frac{1}{7}$$

Katz vs. Kneser-Ney Back-off Smoothing

- Example 1: Wall Street Journal (JSW), English
  - A vocabulary of 60,000 words and a corpus of 260 million words (read speech) from a newspaper such as Wall Street Journal

Table 11.2 *N*-gram perplexity and its corresponding speaker-independent speech recognition word error rate.

Models	Perplexity	Word Error Rate
Unigram Katz	1196.45	14.85%
Unigram Kneser-Ney	1199.59	14.86%
Bigram Katz	176.31	11.38%
Bigram Kneser-Ney	176.11	11.34%
Trigram Katz	95.19	9.69%
Trigram Kneser-Ney	91.47	9.60%

Katz vs. Kneser-Ney Back-off Smoothing (cont.)

- Example 2: Broadcast News Speech, Mandarin
  - A vocabulary of 72,000 words and a corpus of 170 million Chinese characters from Central News Agency (CNA)
  - Tested on Mandarin broadcast news speech collected in Taiwan, September 2002, about 3.7 hours

Models	**Perplexity**	**Character Error Rate (after tree-copy search, TC)**
Bigram Katz	959.56	**16.81**
Bigram Kneser-Ney	**942.34**	18.17
Tigram Katz	752.49	**14.62**
Tigram Kneser-Ney	**670.24**	14.90

  - The perplexities are high here, because the LM training materials are not speech transcripts but merely newswire texts

Interpolated Kneser-Ney Smoothing

- Always combine both the higher-order and the lower-order LM probability distributions
- Take the bigram (m-gram, m=2) conditional probabilities for example:

$$P_{IKN}(w_i \mid w_{i-1}) = \frac{\max\{C[w_{i-1}w_i] - D, 0\}}{C[w_{i-1}]} + \lambda(w_{i-1}) \frac{C[\bullet\, w_i]}{\sum_w C[\bullet\, w]}$$

  - Where
    - $C[\bullet\, w_i]$: the number of unique words that precede w_i
    - $\lambda(w_{i-1})$: a normalizing constant that makes the probabilities sum to 1

$$\lambda(w_{i-1}) = \frac{D}{C[w_{i-1}]} C[w_{i-1}\,\bullet]$$

$C[w_{i-1}\,\bullet]$: the number of unique words that follow the history w_{i-1}

Interpolated Kneser-Ney Smoothing (cont.)

- The exact formula for interpolated Kneser-Ney smoothed trigram conditional probabilities

$$P_{IKN}(w_i \mid w_{i-2}w_{i-1}) = \frac{\max\{C[w_{i-2}w_{i-1}w_i] - D_3, 0\}}{C[w_{i-2}w_{i-1}]} + \lambda(w_{i-2}w_{i-1})P_{IKN}(w_i \mid w_{i-1})$$

$$P_{IKN}(w_i \mid w_{i-1}) = \frac{\max\{C[\bullet\, w_{i-1}w_i] - D_2, 0\}}{\sum_w C[\bullet\, w_{i-1}w]} + \lambda(w_{i-1})P_{IKN}(w_i)$$

$$P_{IKN}(w_i) = \frac{\max\{C[\bullet\, w_i] - D_1, 0\}}{\sum_w C[\bullet\, w]} + \lambda\frac{1}{|V|}$$

For the *IKN* bigram and unigram, the number of unique words that precede a given history is considered, instead of the frequency counts.

Back-off vs. Interpolation

- When determining the probability of *n*-grams with nonzero counts, interpolated models use information from lower-order distributions while back-off models do not

- In both back-off and interpolated models, lower-order distributions are used in determining the probability of *n*-grams with zero counts

- It is easy to create a back-off version of an interpolated algorithm by modifying the normalizing constant

Witten-Bell Discounting

- A much better smoothing method that is only slightly more complex than add-one

- The count of "first time" *n*-grams is just for the number of *n*-gram **types** we have already seen in data
  - Probability of total unseen (zero-count) *n*-grams

$$\left[\sum_{i:c_i=0} p_i^*\right] = \frac{T}{N+T}$$

    - T : the types of *n*-grams we have already seen
    - T differs from V (V : total types of *n*-grams defined beforehand)

Witten-Bell Discounting (cont.)

- Probability mass is equally divided up to among all the zero-count *n*-grams

$$p_i^* = \frac{T}{Z(N+T)},$$

$$\text{where } \; Z = \sum_{i:\ C_i = 0} 1 \quad (\text{number of } n\text{-gram types with zero-counts})$$

- Discounted probability of the seen *n*-grams

$$p_i^* = \frac{c_i}{N+T} \quad \text{if } \; c_i > 0$$

$$(c_i : \text{the count of a seen } n\text{-gram } i)$$

Witten-Bell Discounting (cont.)

- Another formulation (in terms of frequency count)

$$c_i^* = \begin{cases} \dfrac{T}{Z}\dfrac{N}{N+T}, & \text{if } c_i = 0 \\ c_i \dfrac{N}{N+T}, & \text{if } c_i > 0 \end{cases}$$

- Example (of unigram modeling)
  - *V*={*A,B,C,D,E*}, |*V*|=5
  - *S*={*A,A,A,A,A,B,B,B,C,C*}, N=|S|=10
  - 5 for '*A*', 3 for '*B*', 2 for '*C*', 0 for '*D*','*E*', *T*=|{*A,B,C*}|=3, *Z*=2
  - P(*A*)=5/(10+3)=0.385
  - P(*B*)=3/(10+3)=0.23
  - P(*C*)=2/(10+3)=0.154
  - P(*D*)=P(E)=3/(10+3)*(1/2)=0.116

Witten-Bell Discounting (cont.)

- Extended to Bigram Modeling
  - Consider bigrams with the history word w_x
    - For zero-count bigrams (with w_x as the history)

$$\left[\sum_{i:C(w_x w_i)=0} p^*(w_i \mid w_x)\right] = \frac{T(w_x)}{C(w_x)+T(w_x)}$$

$$p^*(w_i \mid w_x) = \frac{T(w_x)}{Z(w_x)(C(w_x)+T(w_x))}$$

– $C(w_x)$: frequency count of word w_x in the corpus

– $T(w_x)$: types of nonzero-count bigrams (with w_x as the history)

– $Z(w_x)$: types of zero-count bigrams (with w_x as the history)

$$Z(w_x) = \sum_{i:C(w_x w_i)=0} 1$$

Witten-Bell Discounting (cont.)

- Extended to Bigram Modeling
  - For nonzero-count n-grams (with w_x as the history)

$$p^*(w_i \mid w_x) = \frac{C(w_x w_i)}{C(w_x) + T(w_x)}$$