Announcements, Assignments, and Reminders

- Visit the URL links that were covered in previous lectures.
Second Guest Lecture, Tuesday, June 4th

Advanced Language Models
Dr. Geoff Zweig
Microsoft Research

Tuesday, June 4th, 2013, 1:30-3:20pm, Thomson Hall, 235

This lecture will present the state-of-the-art in language modeling and describe several cutting edge research problems. The presentation is divided into the two themes of discrete and continuous-space word representations. Using traditional N-gram language models as a jumping-off point, we first describe discrete maximum entropy language models and their extensions into whole-sentence and class-based models. We then turn to continuous space language models, and cover feed-forward and recurrent neural network language models. A remarkable property of the continuous space models is that the learned word representations can be shown to capture syntactic and semantic regularities in language, and we conclude by discussing some recent work in this area and open research questions.

Third Guest Lecture, Thursday, June 6th

Deep Learning for ASR
Dr. Li Deng
Microsoft Research

Thursday, June 6th, 2013, 1:30-3:20pm, Thomson Hall, 235

Semantic information embedded in the speech signal manifests itself in a dynamic process rooted in the deep linguistic hierarchy as an intrinsic part of the human cognitive system. Modeling both the dynamic process and the deep structure for advancing speech technology has been an active pursuit for over more than 20 years, but it is only within past three years or so that technological breakthrough has been created by a methodology commonly referred to as "deep learning". Deep neural nets are recently being used to supersede the Gaussian mixture model component in HMM-based speech recognition, and has produced dramatic error rate reduction in both phone recognition and large vocabulary speech recognition of industry scale while keeping the HMM component intact. On the other hand, the (constrained) dynamic Bayesian networks have been developed for many years to improve the dynamic models of speech aimed to overcome the IID assumption as a key weakness of the HMM, with a set of techniques commonly known as hidden dynamic/trajectory models or articulatory-like segmental representations. A history of these two largely separate lines of research will be critically reviewed and analyzed in the context of modeling the deep and dynamic linguistic hierarchy for advancing speech recognition technology. Both the traditional and new types of deep neural networks and their learning methods will be described in this lecture, demonstrating their tremendous impact on ASR performance created since 2010. Future directions will be discussed, focusing on how the dynamic properties of speech can be more naturally embedded into the deep learning framework than the current deep neural network and HMM hybrid approach.
ICASSP 2013 — this week

- One of the main research conferences on speech recognition algorithms, ICASSP, is happening right now (this week).
- Visit the web page http://www.icassp2013.com/ to get an idea on some of the new things that are being proposed.
- Next week, Dr. Deng will be reporting back to us on some of the recent developments.
- Why am I not there?
  1: I need to teach EE516! 😊
  2: One of the main research conferences on machine learning algorithms, NIPS (http://nips.cc/Conferences/2013/), has a paper submission deadline this Friday!!

Cumulative Outstanding Reading

- Read chapters 1 and 2 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read chapters 3 and 4 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 6 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read HMM sections in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 9 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 11 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
On Final Project

- Will be held Monday, June 10th, 2013
- time/place: TBD
- Project should ideally be on some aspect of the material we have learnt, some aspect of speech processing or recognition. Possible good projects include:
  - A modern advanced paper summary, of papers that we are not going to cover in this class.
  - A new idea of your own, new algorithms and/or theoretical results.
  - Implement a speech recognition system in HTK or some other system.
  - new speech coding, speech application, or
  - application of ideas from speech recognition to other types of data (but must explain in speech terminology).

The ideal project should be research-oriented
- Ideal project would lead to a conference and/or journal paper.
- Fine to combine it with your own research.
- Deadline every Monday, 5:00pm up until day of final project 6/10.
Final Project - Toolkits

- HTK - HMM toolkit (Cambridge, UK)
- CMU-Sphinx http://cmusphinx.sourceforge.net/ - HMM-based Speech recognition toolkit, CMU
- GMTK - general DBN toolkit, originally for speech but useful in general.
- Matlab - good for small problems and for speech processing ideas (but doesn’t scale to larger systems and/or data).

Final Project - pending deadlines

Every Monday from now up until June 10th (our final presentations day). All should be submitted to our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/26924) Specific deadlines are as follows:

- May 27th: 11:45pm: project proposal update (1 page max). (have you all done it??)
- June 3rd: 11:45pm: project status update (1 page max).
- June 10th: 11:00am: final project report (4 pages max).

Note, all deadlines are at 11:45pm at night except for last one which is at 11:00am in the morning.
Office hours and/or email if you have any questions.
Discriminative training methods can produce better generative models (conditional likelihood, extended Baum Welch, minimum classification error (MCE)),

Modeling how words are pronounced is also important in ASR.

Word Pronunciations

What does Markov chain tell us? One thing is the pronunciation of a word. Each state might correspond to a phone.

“Pronunciation modeling” is an important sub-field within speech recognition.

Often words have only one pronunciation
Word Pronunciations

- The same word, however, can have many pronunciations.
- Example: The pronunciation Markov transition matrix for the word “and”

```
æ   n   d
```

- In general, each word has a set of associated pronunciations, and sometimes there can be many.
- How do we know how a word is pronounced?

```
∃ standard pronunciation dictionaries that one can use. E.g.,
PRONLEX http://www.ldc.upenn.edu/Catalog/readme_files/comlex_pron.readme.html or CMU dict
http://www.speech.cs.cmu.edu/cgi-bin/cmudict
Examples from pronlex (see above link for details):
bating .xb’et.IG
abba ’@b.x #NAME
abbenhaus ’@b.Inh+Ws #NAME
abbey ’@b.i #NAME
abbott ’@b.xt #NAME
abboud .xb’ud #NAME
abbreviated .xbr’iv.i+et.Id
abby ’@b.i
abdominal .@bd’am.In.xl
```

- Commercial ASR systems use their own pronunciation lexicon, and this is a critical part of the performance of such systems.
some American dialect distinguish the vowels in “sawed” and “sod”, while others do not; the ending “-ing” can be pronounced with a vowel more like “heed” or one more like “hid”, and with a final consonant like that of “sing” or like that of “sin”. This does not take account of considerable variation of actual quality in these sounds: thus some (New Yorkers) pronounce the vowel of “sawed” as a sequence of a vowel like that in “Sue” followed by one like that in “Bud”, while in less stigmatized dialects it is a single vowel (that may or may not be like that in “sod”).

Combining all these variants for the transcription of the word “dogging” we would get 12 pronunciations – three versions of the first vowel, two versions of the second vowel, and two versions of the final consonant. Then someone else comes along to tell us that some Chicagoans not only merge the vowels in “sawed” and “sod” but also move both of them towards the front of mouth, with a sound similar (in extreme cases) to the more standard pronunciation of “sad”. Now we have $4 \times 2 \times 2 = 16$ pronunciations for the simple word “dogging” – with a comparable 16 available for “logging” and “hogging” and so forth, and plenty of variants yet to catalogue.
**Word Pronunciation: units**

- Pronunciation lexica can be big (100k-500k).
- Many possible units may be used to specify a pronunciation
- most common: phonemes and their realizations, phones. Ex: using ARPAbet chocolate pudding → CaKxIlt pUdG
- phones can often be characterized acoustically (using formants, and their characteristic frequencies)
- syllables - longer (200ms) and more (about 3000 for English)
- individual articulatory gestures within the vocal tract (semi-synchronously), very low-level. Factored representation.

---

**Word Pronunciations and Discriminability**

- How many pronunciations should a given word model be given?
- If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.
- This is generatively accurate, but can cause confusion with other words.
- Too few pronunciations per word, poor model.
- Too many pronunciations per word, performance (in terms of classification error) drops due to confusability between words.
- Ideal point is somewhere in the middle. Where can only be determined empirically.
HMM: Pronunciation Modeling

- Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”.
- Words are pronounced in different many ways, depending on context.
- Two approaches:
  - introduce many possible pronunciations for each word irrespective of context. But this can increase confusability (words start blurring into each other)
  - Map to correct pronunciation dynamically based on context, and questions about context, only include a small number of pronunciations per context. In this case, need need mapping $T(BASEFORM) \rightarrow SURFACE\ FORM$ where surface form has details about variability.

Ex: phonemic spelling of bottle:
- /b a a t a x l/ (in ARPABET form)
- In American English, /t/ is “flapped” $\Rightarrow [dx]$.
- Also /a x l/ $\Rightarrow [el]$
- Resulting surface form: [b a a d x e l]
- Goal: build probabilistic mapping from base (phoneme) to surface forms (phones).
- Might also have phoneme deletion, so need to include special “_” in output $y$ alphabet.
- Let $x = x_{1:m}$ be a string of $m$ phonemes (baseforms) and $y = y_{1:m}$ be a string of $m$ phones (surface forms).
- Might use a model such as:

$$p(y|x) = \prod_n p(y_n|x, y_{1:n-1}) \quad (16.1)$$
HMM: Pronunciation Modeling

- Need to make conditional independence assumptions to be tractable. Two typical models:
  1) current phone is dependent on window of phonemes and previous phone

\[
p(y|x) = \prod_{n} p(y_n|x_{n-r:n+r}, y_{n-1})
\]  

(16.2)

2) current phone is dependent only on window of phonemes

\[
p(y|x) = \prod_{n} p(y_n|x_{n-r:n+r})
\]

(16.3)

Outline of today

- Finish Pronunciation modeling.

- Start language modeling.
Good books (for today)

- our book (Huang, Acero, Hon, “Spoken Language Processing”)
- Jurafsky&Martin’s book on NLP (for the language modeling part).

HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance) — this gives training data pairs, but we do need local phoneme/phone distances, cost constraints, etc. for DP
- Learn \( p(y_k|x_{k-r:k+r}) \) from this aligned data
- Use this mapping to transform words as they are hypothesized in the ASR system
- Hence, we get non-dictionary pronunciations, even for words for which we have never seen surface forms
**HMM: Pronunciation Modeling**

How to learn these models:
- approximately 50 phonemes, 200 phones
- context of size \((2r + 1)\)
- implies table of size \(200 \times 50^{2r+1}\)

Solution, decision trees (DTs):
- pools together common contexts (e.g., matters not the particular phone, but rather that it is vowel)
- lower-dimensional representation (need less data)
- Generalizes better

DT input is a list of “features”, and a set of possible questions about these features, such as \(x \in C_1\), meaning is \(x\) a member of the set that has answer “yes” to the question. \(C_1\) could be, say, set of all vowels.

Leaf node is distribution over all phones.

Each leaf’s probability distribution counts that context.

How to build these trees?

Consider distribution over just \(y_k\) having entropy:

\[
H(Y_k) = -\sum_{y_k} p(y_k) \log p(y_k) \tag{16.4}
\]
HMM: Pronunciation Modeling

- Consider set (or question) $S$ splitting the data, and corresponding conditional entropies:
  \[
  H(Y_k|x \in S) = -\sum_{y_k} p(y_k|x \in S) \log p(y_k|x \in S) \quad (16.5)
  \]
  \[
  H(Y_k|x \notin S) = -\sum_{y_k} p(y_k|x \notin S) \log p(y_k|x \notin S) \quad (16.6)
  \]
- How likely is “$S$” to be true? “#()” is count function.
  \[
  p(x \in S) = \frac{\#(x \in S)}{\#(x \in S) + \#(x \notin S)} \quad (16.7)
  \]
- Average entropy after the split:
  \[
  H(Y_k|S) = H(Y_k|x \in S)p(x \in S) + H(Y_k|x \notin S)p(x \notin S) \quad (16.8)
  \]
- Entropy reduction is the “value” of the split $S$:
  \[
  H(Y_k) - H(Y_k|S) = I(Y_k; S) \quad (16.9)
  \]

HMM: Pronunciation Modeling

- Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.
- Typical heuristic is to use greedy algorithm (top down) to define tree:
  
  **A)** Start with single node top of tree
  
  **B)** Then, for each leaf node:
  
  1. compute $I(Y_k; S_i)$ for all splits $I()$ for each node
  2. If largest $I(Y_k; S_i)$ is large enough, split data down two branches of tree, or if $I()$ is small, stop.
  3. go back to step 1

- But how to choose the sets $S_i$?
- Input vector $x = x_{k-r:k+r}$ need to ask questions like: “is $x \in S$?” for $S \in 2^X$.
- but $2^{|X|}$ elements in power set, or $2^{(2r+1)L}$ where $L$ is number of phonemes
HMM: Pronunciation Modeling

- Solution: prior-cluster the phonemes into list of attributes.
- E.g., phone ≡ (consonant manner, consonant place, vowel manner, vowel place)
- Each attribute takes on only small (e.g., 12) number of values (which includes n/a)
- Ask question only of form (CM, CP, VM, VP) = (4, 3, 1, 2), so only $12^4$ sets per phoneme (still many, so more prior-cluster may be necessary depending on availability of computational resources and training data).
- There are many other advanced pronunciation modeling techniques, data issues, incorporating rate of speech information, and so on. This is still active area of research.

Language Modeling

- Recall goal equation:

$$ W^* \in \arg \max_w p(x|w)p(w) $$

where $p(x|w)$ is an HMM and $p(w)$ is a probability model over word strings.
- we've discussed $p(x|w)$ at length, but what about $p(w)$?
- if $w$ is a single word, no problem (counting) but if $w$ is all possible sentences of any length, this won't work.
- Say $V$ is set of words in vocabulary, $|V|$ is vocabulary size (which can be large, 100k-500k)
- Many approaches, overall goal is joint distribution over variable length set of words. Before doing that, how to measure language difficulty?
“Perplexity” of a Language

- A measure of the “difficulty” of a language as reflected by a language model (the worse the language model, the higher the perplexity).
- Ideally, language is generated by a stochastic process and perplexity measures how predictable a language is when generated from that process (so both language and language model are based on the same process).
- In practice, perplexity is always measure of a corpus of data with respect to a language model (so it depends on both).
- Approach: Train a language model on one portion of text, and with that trained model, compute the perplexity on a separate text corpus (should not be the same corpus).
- Perplexity also measures language model’s ability to generalize.

Perplexity

- Let $Q()$ be a language model (obtained in some way), hence $Q()$ can measure the probability of a length $N$ text string $Q(w_{1:N})$.
- $Q()$ is valid distribution, normalizes to 1.

$$\sum_{w_{1:N}} Q(w_{1:N}) = 1 \quad (16.11)$$

- Perplexity of $w_{1:N}$ relative to $Q()$ can be defined as:

$$\text{ppl}_{Q}(w_{1:N}) \triangleq [Q(w_{1:N})]^{-1/N} \quad (16.12)$$

- So, perplexity is inverse geometric mean of the probabilities (we’ll see why below)
Perplexity and Entropy

- Given distribution $p$, entropy of $p$: $H(p) = - \sum_i p_i \log p_i$.
- Given two distribution $p, q$, cross-entropy: $H_q(p) = - \sum_i p_i \log q_i \geq H(p)$.
- For stationary ergodic sources $w_1, w_2, \ldots$ with $w_i \sim Q$:
  \[
  H(W) = \lim_{N \to \infty} -\frac{1}{N} \log Q(w_1:N), \quad \text{where } w_1:N \sim Q(w_1:N)
  \] (16.13)
- For finite $N$, we have an estimate of the entropy:
  \[
  H_N(W) = -\frac{1}{N} \log Q(w_1:N)
  \] (16.14)
- Perplexity is $2^H$, where $H$ is entropy rate of word stochastic process.
  \[
  2^{H_N(W)} = 2^{-\frac{1}{N} \log Q(w_1:N)} = 2^{\log [Q(w_1:N)]^{-\frac{1}{N}}} = [Q(w_1:N)]^{-\frac{1}{N}}
  \]
- If $H$ is large, language is considered difficult.
- ppl is as difficult as a language with $|V| = \text{ppl}$, and with uniform distribution over those words.

Perplexity and Language

- For English written text, it is estimated that $\text{ppl} \approx 68$, or about $\log_2(68) = 6.09 \text{bits/word}$ (Rosenfield’96)
- If any substring in $w_{1:N}$ gets zero probability, then $\text{ppl} = \infty$ (if this happens in practice, we say it is an infinitely complex language relative to the LM)
- When ppl is computed on real data, is really upper bound on language complexity
Perplexity of a Language Models on a Corpus

- Perplexity (in practice) is really more like cross entropy.
  \[ H_q(p) = - \sum_i p_i \log q_i \geq H(p). \]
- We train language model, say \( P(w_{1:N}) \), on some data
  \[ D_{\text{train}} = \{w_{1:N_i}^{(i)}\}_{i=1}^{M_{\text{train}}} \]
drawn from distribution \( Q_{\text{train}}(w_{1:N_i}^{(i)}) \),
meaning \( w_{1:N_i} \sim Q() \) (e.g., maximum likelihood training).
- We evaluate \( P \) on data \( D_{\text{test}} = \{v_{1:N_i}^{(i)}\}_{i=1}^{M_{\text{test}}} \)
drawn from distribution \( Q_{\text{test}}(v_{1:N_i}^{(i)}) \), meaning \( v_{1:N_i} \sim Q_{\text{test}}() \), by computing:
  \[
  \log \text{ppl}_{M_{\text{test}}} = \frac{1}{M_{\text{test}}} \log \left( \prod_{i=1}^{M_{\text{test}}} P(w_{1:N}) \right) \tag{16.15}
  \]
- As \( M_{\text{test}} \to \infty \), \( \log \text{ppl}_{M_{\text{test}}} \to H_P(Q_{\text{test}}) \)
- If \( P \) is a consistent estimator of \( Q_{\text{train}} \), then as \( M_{\text{train}}, M_{\text{test}} \to \infty \),
  \( \log \text{ppl}_{M_{\text{test}}} \to H_{Q_{\text{train}}}(Q_{\text{test}}) \) and this \( = H(Q) \) only if train/test distributions are identical.

Perplexity and Language Models

- Perplexity is used to judge language models.
- E.g., if \( \text{ppl}(Q_A) < \text{ppl}(Q_B) \), where \( Q_A \) and \( Q_B \) are language models, then \( Q_A \) is better (perhaps).
- Lower perplexity means less choice, lower complexity, better predictability on a (held-out) corpus of text.
- Better predictability means error in:
  \[
  W^* \in \arg\max_w p(x|w)p(w) \tag{16.16}
  \]
is lower.
- \( \text{ppl} \) is notoriously difficult to compute properly (beware of judging a language model using \( \text{ppl} \))
- However, lower \( \text{ppl} \) doesn’t mean ASR accuracy necessarily better since acoustic confusability has a very large effect, in \( p(x|w)p(w) \).
Language Modeling

- Many possibilities for \( Q(w) \)
- Chain rule gives us:
  \[
  Q(w_{1:N}) = \prod_{i=1}^{N} Q(w_i | w_{1:i-1}) = \prod_{i=1}^{N} Q(w_i | h_i) \quad \text{where} \quad h_i \triangleq w_{1:i-1}
  \]  
  \[\text{(16.19)}\]
- Factor of the form \( Q(w_i | h_i) \) would require \(|V|^N\) size tables, huge since \(|V|\) could be 100k or more.
- How hard is it to find a 3-word string never before seen by search engine? What about a 4-word string, 5-word, etc?
- Hence, most strings don’t exist in training set (so table would be terribly sparse)
- test set string not seen in training set might encounter zero probabilities (\(\infty\) ppl)

Perplexity and Language Model Quality Assessment in ASR

- \( \text{ppl}(Q_A) < \text{ppl}(Q_B) \), where \( Q_A \) and \( Q_B \) are language models, why might not:
  \[
  W^* \in \arg\max_w p(x|w)Q_A(w)
  \]  
  result in lower error than
  \[
  W^* \in \arg\max_w p(x|w)Q_B(w)
  \]  
  \[\text{(16.17)}\]
  \[\text{(16.18)}\]

1) Decisions might be dominated by the acoustic model \( p(x|w) \) and this can swamp any language model scores (hypothesis ranks don’t change).
2) Even if scores of \( Q_A(w) \) and \( Q_B(w) \) are different, hypothesis ranks might be the same \( \arg\max_w Q_A(w) = \arg\max_w Q_B(w). \)
3) In real ASR decoding systems, we perform approximate inference. Even if \( Q_A \) is better, we might not experience it since anything allowing us to see this benefit might get pruned (or approximated) out.
Language Modeling

• Tri-gram (3-gram) Language model:
  \[ Q(w_i|h_i) = Q(w_i|w_{i-1}, w_{i-2}) \]

• Other models (bi-gram, \( p(w_t|w_{t-1}) \))

• Why so successful?
  – Information about \( w_i \) falls of rapidly with distance to word:
    - N-gram is good tradeoff between estimation quality and information fall-off (some corpora 4-gram is better)
Language Modeling

- Trigram ML solution:
  \[ Q(w_t|w_{t-1}, w_{t-2}) = \frac{C(w_t, w_{t-1}, w_{t-2})}{C(w_{t-1}, w_{t-2})} \]

- But \( C(w_t, w_{t-1}, w_{t-2}) \) still sparse, many tri-grams won’t exist in reasonable size training data, underestimated (or zero) probabilities

- Even given sufficient training, \(|V|^3 \) table size is large (memory issues)

- There are many solutions (we survey some of them here, others see EE517 next quarter)

- All solutions are some form of “smoothing” (replace zeros with non-zeros in distribution)
Language Modeling

• What if use ML to determine $\lambda$ values?

$$(\lambda_1, \lambda_2, \lambda_3)^* = \arg\max_{\lambda_1, \lambda_2, \lambda_3} \sum_t \log Q(w_t|w_{t-1}, w_{t-2})$$

• If same training data used for $\lambda$ and LM training, optimal solution always (why?):

$$\lambda_1 = \lambda_2 = 0, \quad \lambda_3 = 1$$

• This gets worse since ideally we would like the $\lambda$ values to depend on the words (or at least a function of the words)
Language Modeling

- word-dependent lambda
  \[ p(w_t|h_t) = \lambda(h_t)f(w_t|h_t) + (1 - \lambda(h_t))f(w_t) \]

- But now there are too many \( \lambda \)'s. We use \( C(h_t) \), depending on count (in held out set), choose different mixture weights.
  \[ C() \in \{0, 1, \ldots, M\} \]

- Simplify: \( C() \in \{R_1, R_2, \ldots, R_N\} \in \mathbb{R} \)
  - \( N \ll M \)
  - \( R_1 = [0, \max(R_i)] \)
  - \( R_i = [\min(R_i), \max(R_i)] \)

- Counts are binned into ranges for simplicity
Adaptive Language Modeling

- Idea: LM $p(w_t|h_t)$ should not stay fixed but should change depending on recent events.
- Ideal case: $p(w_t|h_t) = 1$ if $w_t$ is correct word.
- Simple approach: cache-based LMs:
  $$ p(w_t|h_t) = \sum_i p_o(w_t|i) p(i|\phi(w_{t-1}), \phi(w_{t-2})) $$
  $$ \phi(w) = \text{word class of word } w $$
  $$ p_o(w_t|i) = \lambda_c p_{\text{cache}}(w_t|i) + (1 - \lambda_c) p_{\text{class}}(w_t|i) $$
  $$ p_{\text{class}}(w_t|i) = \frac{C(w_t, i)}{C(i)} $$
  $$ p_{\text{cache}}(w_t|i) = \text{something that gives high probability for words recently seen in } i\text{th cache} $$
Adaptive Language Modeling

- Ex: \( \text{ith cache consists of set of words } S_i \)
- \( \text{Let } k_t \text{ be a } K \text{-length history at time } t \)
- But other strategies exist as well.

Backoff Language Modeling

- Idea: Form \( p(w_t|h_t) \) from lower-order distributions when count falls below some threshold (but make sure to keep everything normalized)

\[
p_{\text{BO}}(w_t|w_{t-1}, w_{t-2}) =
\begin{cases}
  d_c(w_t, w_{t-1}, w_{t-2}) \frac{c(w_t, w_{t-1}, w_{t-2})}{c(w_t, w_{t-1})} & \text{if } c(w_t, w_{t-1}, w_{t-2}) > \tau \\
  \alpha(w_{t-1}, w_{t-2}) p_{\text{BO}}(w_t|w_{t-1}) & \text{else}
\end{cases}
\]

- Discount coefficient
- Backoff weight (BOW)
• Discount coefficient:
  – if \( d_{c(w)} = 1 \) for all \( w \), then we get ML solution since no mass gets distributed to lower-order models
  – \( d() \) determines amount of mass to “steal” away from higher-order ML solution and give to BO lower-order model.
  – There are *many* different forms of \( d() \). Much research in LM has been on finding appropriate form (see Chen paper).
  – “smoothing algorithms” is equivalent to finding appropriate form of \( d() \). This equation it “smooths” out ML solution so that no zeros are in distribution.
  – BO equation doesn’t require full table even though it does not give 0 probability to anything, much less memory.
Backoff Language Modeling

• Good/Turing Smoothing

\[ r^* = (r + 1)^{\frac{n_{r+1}}{n_r}} \]

• So if something has occurred 0 times, we will pretend it has occurred prop. to ratio of meta counts.

• We do this up to a limit, to get discount coeff:

\[ d_r = \begin{cases} 1 & r \geq k \quad \text{gtmax, upper GT discounting cutoff} \\ \frac{r^*/r - \frac{(k+1)n_{k+1}}{n_k}}{1 - \frac{(k+1)n_{k+1}}{n_1}} & \approx r^*/r \end{cases} \]

where \( r = C(w_t, w_{t-1}, w_{t-2}) \)
Backoff Language Modeling

• Normalization:

\[
\sum_{w_t} p_{\text{BO}}(w_t|w_{t-1}, w_{t-2}) = 1 \quad \forall w_{t-1}, w_{t-2}
\]

• The backoff weight (BOW, or \(\alpha\)) ensures that we have a valid normalized distribution.

\[
\alpha(w_{t-2}, w_{t-1}) = \frac{1 - \sum_{w:c>\tau} d_c p_{\text{ML}}}{\sum_{w:c\leq\tau} p_{\text{BO}}(w_t|w_{t-1})} = \frac{1 - \sum_{w:c>\tau} d_c p_{\text{ML}}}{1 - \sum_{w:c>\tau} p_{\text{BO}}(w_t|w_{t-1})}
\]

• This is crucial to get right, otherwise perplexity will be wrong.
Backoff Language Modeling

• Recursion:  
\[ p_{bo}(w_t|w_{t-1}, w_{t-2}) = \begin{cases} 
\frac{c(w_t, w_{t-1}, w_{t-2})}{\alpha(w_{t-1})} & \text{if } c(w_t, w_{t-1}, w_{t-2}) > \tau \\
\alpha(w_{t-1}, w_{t-2}) p_{bo}(w_t|w_{t-1}) & \text{else}
\end{cases} \]

• Normalization: 
– Make sure \( p_{bo} \) sums to 1.
\[ \sum_{w_t} p_{bo}(w_t|w_{t-1}, w_{t-2}) = 1 \quad \forall w_{t-1}, w_{t-2} \]

\[ \alpha_{w_{t-2}, w_{t-1}} = \frac{1 - \sum_{w:c \geq \tau} d_c p_{ML}}{\sum_{w:c \leq \tau} p_{bo} p(w_t|w_{t-1})} \]

\[ = \frac{1 - \sum_{w:c \geq \tau} d_c p_{ML}}{1 - \sum_{w:c \geq \tau} p_{bo} p(w_t|w_{t-1})} \]