ELEN E6884/COMS 86884 Speech Recognition Lecture 11

Michael Picheny, Ellen Eide, Stanley F. Chen
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{picheny,eeide,stanchen}@us.ibm.com

17 November 2005

Administrivia

- Lab 3 handed back today
- Lab 4 due Sunday midnight
- next week is Thanksgiving
- select paper(s) for final presentation by Monday after Thanksgiving
  - E-mail me your selection
  - presentation guidelines on web site
- aiming for field trip to IBM between Tuesday (12/13) – Thursday (12/15)
  - please E-mail me your preference by tomorrow

Feedback from Last Week

What probability models are used in MAP, MMIE, etc.? How is the estimation of HMM parameters affected?

- same model form used: HMM w/ GMM output distributions
  - what's changed: how GMM parameters are estimated
  - HMM transition probabilities don't really matter
- (please ask questions during class!)

The Story So Far

The Fundamental Equation of Speech Recognition

$$\text{class}(\mathbf{x}) = \arg\max_{\omega} \; P(\omega|\mathbf{x}) = \arg\max_{\omega} \; P(\omega)P(\mathbf{x}|\omega)$$

- $P(\omega)$ — probability distribution over word sequences ω
  - language model
  - models frequency of each word sequence ω
- $P(\mathbf{x}|\omega)$ — probability of acoustic feature vectors $\mathbf{x}$ given word sequence ω
  - acoustic model

The Story So Far

Language model $P(\omega = w_1 \cdots w_l)$

- helps us disambiguate acoustically ambiguous utterances
  - *e.g.*, THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD
  - can make mediocre acoustic models perform acceptably
- estimates relative frequency of word sequences $\omega = w_1 \cdots w_l$
  - . . . *in the given domain*!
  - *i.e.*, assign high probs to likely word sequences
  - *i.e.*, assign low probs to unlikely word sequences

The Story So Far

How about those n-gram models?

- for very restricted, small-vocabulary domains
  - write a (Backus-Naur form/CFG) grammar, convert to FSA
  - problem: no one ever follows the script
    - how to handle out-of-grammar utterances?
  - use n-grams even for restricted domains?
- large vocabulary, unrestricted domain ASR
  - n-grams all the way

The Story So Far

N-gram models

$$
\begin{aligned}
& P(\omega = w_1 \cdots w_l) \\
&= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_l|w_1\cdots w_{l-1}) \\
&= \prod_{i=1}^{l} P(w_i|w_1\cdots w_{i-1})
\end{aligned}
$$

- Markov assumption: identity of next word depends only on last $n-1$ words, say n=3

$$P(w_i|w_1\cdots w_{i-1}) \approx P(w_i|w_{i-2}w_{i-1})$$

The Story So Far

N-gram models

- maximum likelihood estimation

$$P_{\text{MLE}}(w_i|w_{i-2}w_{i-1}) = \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\sum_{w_i}\text{count}(w_{i-2}w_{i-1}w_i)}$$
$$= \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\text{count}(w_{i-2}w_{i-1})}$$

- smoothing
  - better estimation in sparse data situations

Spam, Spam, Spam, Spam, Spam, and Spam

- n-gram models are robust
  - assign reasonable nonzero probs to all word sequences
  - handle unrestricted domains
- n-gram models are easy to build
  - can train on plain unannotated text
  - just count, normalize, and smooth
- n-gram models are scalable
  - fast and easy to build models on billions of words of text
  - can use larger n with more data
- n-gram models are great!
  - . . . or are they?

How Do N-Grams Suck?

Let me count the ways

- not great at modeling short-distance dependencies
- do not generalize well
  - training data contains sentence

    LET'S EAT STEAK ON TUESDAY
  - test data contains sentence

    LET'S EAT SIRLOIN ON THURSDAY
  - point: occurrence of STEAK ON TUESDAY does not affect estimate of $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$
- collecting more data can't fix this
  - (Brown *et al.*, 1992) 350MW training $\Rightarrow$ 15% trigrams unseen

How Do N-Grams Suck?

Let me count the ways

- not great at modeling medium-distance dependencies
  - within a sentence
- Fabio example

  FABIO, WHO WAS NEXT IN LINE, ASKED IF THE TELLER SPOKE ...
  - trigram model: $P(\text{ASKED} \mid \text{IN LINE})$

How Do N-Grams Suck?

Modeling medium-distance dependencies

- random generation of sentences with model $P(\omega = w_1 \cdots w_l)$
  - roll a K-sided die where . . .
  - each side s_ω corresponds to a word sequence ω . . .
  - and probability of landing on side s_ω is $P(\omega)$
  - $\Rightarrow$ reveals what word sequences model thinks is likely

How Do N-Grams Suck?

trigram model trained on 20M words of WSJ (lab 4)

AND WITH WHOM IT MATTERS AND IN THE SHORT -HYPHEN TERM

AT THE UNIVERSITY OF MICHIGAN IN A GENERALLY QUIET SESSION

THE STUDIO EXECUTIVES LAW

REVIEW WILL FOCUS ON INTERNATIONAL UNION OF THE STOCK MARKET

HOW FEDERAL LEGISLATION

"DOUBLE-QUOTE SPENDING

THE LOS ANGELES

THE TRADE PUBLICATION

SOME FORTY %PERCENT OF CASES ALLEGING GREEN PREPARING FORMS

NORTH AMERICAN FREE TRADE AGREEMENT (LEFT-PAREN NAFTA)RIGHT-PAREN ,COMMA WOULD MAKE STOCKS

A MORGAN STANLEY CAPITAL INTERNATIONAL PERSPECTIVE ,COMMA GENEVA

"DOUBLE-QUOTE THEY WILL STANDARD ENFORCEMENT

THE NEW YORK MISSILE FILINGS OF BUYERS

How Do N-Grams Suck?

Modeling medium-distance dependencies

- real sentences tend to “make sense” and be coherent
  - don’t end/start abruptly
  - have matching quotes
  - are about a single subject
  - some are even grammatical
- n-gram models don’t seem to know this
  - . . . and it’s hard to see how they could

How Do N-Grams Suck?

Let me count the ways

- not great at modeling long-distance dependencies
  - across multiple sentences
- see previous examples
- in real life, consecutive sentences tend to be on the same topic
  - referring to same entities, *e.g.*, Clinton
  - in a similar style, *e.g.*, formal *vs.* conversational
- n-gram models generate each sentence independently
  - identity of last sentence doesn't affect probabilities of future sentences
  - (suggests need to generalize definition of LM)

Shortcomings of N-Gram Models

Recap

- not great at modeling short-distance dependencies
- not great at modeling medium-distance dependencies
- not great at modeling long-distance dependencies
- basically, n-gram models are just a dumb idea
  - they are an insult to language modeling researchers
  - are great for me to poop on
  - n-gram models, . . . you're fired!

Outline

Advanced language modeling

- Unit I: techniques for restricted domains
  - aside: confidence
- Unit II: techniques for unrestricted domains
- Unit III: maximum entropy models
- Unit IV: other directions in language modeling
- Unit V: an apology to n-gram models

Language Modeling for Restricted Domains

Step 0: data collection

- we need relevant data to build statistical models
  - available online resources like newspapers . . .
  - not useful for applications like weather forecasts or driving directions or stock quotes or airline reservations

Data Collection for Specialized Applications

- ask co-workers to think up what they might say
- *Wizard of Oz* data collection
  - referring to the 1978 smash hit, *The Wiz*, starring Michael Jackson as the scarecrow
- get a simple system up fast
  - collect live data, improve your models, iterate
  - *e.g.*, MIT's Jupiter, 1-888-573-TALK

Language Modeling for Restricted Domains

Improving n-gram models: short-term dependencies

- main issue: data sparsity/lack of generalization

  I WANT TO FLY FROM BOSTON TO ALBUQUERQUE

  I WANT TO FLY FROM AUSTIN TO JUNEAU

- point: (handcrafted) grammars were kind of good at this

$$\text{[sentence]} \rightarrow \text{I WANT TO FLY FROM [city] TO [city]}$$

$$\text{[city]} \rightarrow \text{AUSTIN} \mid \text{BOSTON} \mid \text{JUNEAU} \mid \ldots$$

  - grammars are brittle

- can we combine robustness of n-gram models with generalization ability of grammars?

Combining N-Gram Models with Grammars

Approach 1: generate artificial data

- use grammars to generate artificial n-gram model training data
- *e.g.*, identify cities in real training data

  I WANT TO FLY FROM [BOSTON] TO [ALBUQUERQUE]

- replace city instances with other legal cities/places (according to city grammar)

  I WANT TO FLY FROM [NEW YORK] TO [PARIS, FRANCE]
  I WANT TO FLY FROM [RHODE ISLAND] TO [FRANCE]
  I WANT TO FLY FROM [REAGAN INTERNATIONAL] TO [JFK]
  ...

- also, for date and time expressions, airline names, etc.

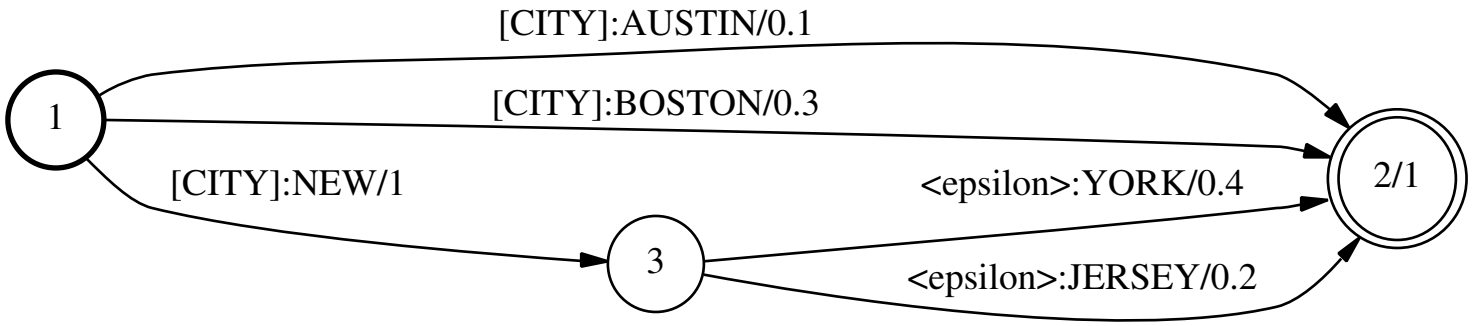
Combining N-Gram Models with Grammars

Approach 2: Embedded grammars (IBM terminology)

- instead of constructing n-gram models on *words*, build n-gram models on words and *constituents*
- *e.g.*, replace cities and dates in training set with special tokens

  I WANT TO FLY TO [CITY] ON [DATE]

- build n-gram model on new data, *e.g.*, $P(\text{[DATE]} \mid \text{[CITY] ON})$
- express grammars as weighted FST's



Combining N-Gram Models with Grammars

Embedded grammars (cont'd)

- possible implementation
  - regular n-gram: LM is acceptor A_{ngram}
  - w/ embedded grammars: LM is acceptor $A_{\text{ngram}} \circ T_{\text{grammar}}$
  - static expansion may be too large, *e.g.*, if large grammars
  - can do something similar to dynamic expansion of decoding graphs
    - on-the-fly composition
- embedded embedded grammars?
  - recursive transition networks (RTN's)

Embedded Grammars

Modeling short and medium-distance dependencies

- addresses sparse data issues in n-gram models
  - uses hand-crafted grammars to generalize
- can handle longer-distance dependencies since whole constituent treated as single token

  I WANT TO FLY TO WHITE PLAINS AIRPORT IN FIRST CLASS

  I WANT TO FLY TO [CITY] IN FIRST CLASS
- what about modeling whole-sentence grammaticality?
  - people don't speak grammatically
  - most apps just need to fill a few slots, *e.g.*, [FROM-CITY]

Language Modeling for Restricted Domains

Modeling dependencies across sentences

- many apps involve computer-human *dialogue*
  - you know what the computer said
  - you have a reasonable idea of what the human said before
  - you may have a pretty good idea what the human will say next
- directed dialogue
  - computer makes it clear what the human should say
  - *e.g.*, WHAT DAY DO YOU WANT TO FLY TO BOSTON?
- undirected or mixed initiative dialogue
  - user has option of saying arbitrary things at any point
  - *e.g.*, HOW MAY I HELP YOU?

Language Modeling for Restricted Domains

Modeling dependencies across sentences

- switching LM's based on context
- *e.g.*, directed dialogue
  - computer says: IS THIS FLIGHT OK?
    - activate [YES/NO] grammar
  - computer says: WHICH CITY DO YOU WANT TO FLY TO?
    - activate [CITY] grammar
- boost probabilities of entities mentioned before in dialogue?

There Are No Bad Systems, Only Bad Users

Aside: What to do when things go wrong?

- *e.g.*, ASR errors put user in dialogue state they can't get out of
- *e.g.*, you ask: IS THIS FLIGHT OK?
  - user responds: I WANT TO TALK TO AN OPERATOR
  - user responds: HELP, MY PANTS ARE ON FIRE!
- if activate specialized grammars/LM's for different situations
  - want to be able to detect out-of-grammar utterances
- can an ASR system detect when it's wrong?
  - even for in-grammar utterances?

Aside: Confidence and Rejection

- want to *reject* ASR hypotheses with low *confidence*
  - *e.g.*, say: I DID NOT UNDERSTAND; COULD YOU REPEAT?
- how to tell when you have low confidence?
  - hypotheses ω with low acoustic likelihoods $P(\mathbf{x}|\omega)$
  - cannot differentiate between low-quality channel or unusual speaker and true errors
- better: posterior probability

$$P(\omega|\mathbf{x}) = \frac{P(\mathbf{x}|\omega)P(\omega)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\omega)P(\omega)}{\sum_{\omega^*} P(\mathbf{x}|\omega^*)P(\omega^*)}$$

  - how much model prefers hypothesis ω over all others

Confidence and Rejection

Calculating posterior probabilities

$$P(\omega|\mathbf{x}) = \frac{P(\mathbf{x}|\omega)P(\omega)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|\omega)P(\omega)}{\sum_{\omega^*} P(\mathbf{x}|\omega^*)P(\omega^*)}$$

- to calculate a reasonably accurate posterior, need to sum over sufficiently rich set of competing hypotheses ω^*
  - generate lattice of most likely hypotheses, instead of 1-best
  - use Forward algorithm to compute denominator
  - to handle out-of-grammar utterances, create *garbage* models
    - simple acoustic model covering out-of-grammar utterances
- issue: language model weight or acoustic model weight?

Confidence and Rejection

Recap

- accurate rejection essential for usable dialogue systems
- posterior probabilities are more or less state-of-the-art
- if you think you're wrong, can you use this information to somehow improve WER?
  - *e.g.*, if have other information sources, like back end parser/database

    I WANT TO FLY FROM FORT WORTH TO BOSTON (0.4)
    I WANT TO FLY FROM FORT WORTH TO AUSTIN (0.3)
    I WENT TO FLY FROM FORT WORTH TO AUSTIN (0.3)
  - encode this info in LM?

Where Are We?

Advanced language modeling

- Unit I: techniques for restricted domains
  - aside: confidence
- Unit II: techniques for unrestricted domains
- Unit III: maximum entropy models
- Unit IV: other directions in language modeling
- Unit V: an apology to n-gram models

Language Modeling for Unrestricted Domains

Overview

- short-distance dependencies
  - class n-gram models
- medium-distance dependencies
  - grammar-based language models
- long-distance dependencies
  - cache and trigger models
  - topic language models
- linear interpolation revisited

Short-Distance Dependencies

Class n-gram models

- word n-gram models do not generalize well

  LET'S EAT STEAK ON TUESDAY

  LET'S EAT SIRLOIN ON THURSDAY

  - point: occurrence of STEAK ON TUESDAY does not affect estimate of $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$

- in embedded grammars, some words/phrases are members of grammars (*e.g.*, the city name grammar)
  - n-gram models on words and constituents rather than just words
  - counts shared among members of a grammar

    LET'S EAT [FOOD] ON [DAY-OF-WEEK]

Class N-Gram Models

- embedded grammars
  - grammars are manually constructed
  - can contain phrases as well as words, *e.g.*, THIS AFTERNOON
- class n-gram model
  - let's say we have way of assigning single words to classes . . .
  - in context-independent manner (hard classing)
  - *e.g.*, class bigram model

$$P(w_i|w_{i-1}) = P(w_i \mid \text{class}(w_i)) \times P(\text{class}(w_i) \mid \text{class}(w_{i-1}))$$

  - class expansion prob $\Leftrightarrow$ grammar expansion prob
  - class n-gram prob $\Leftrightarrow$ constituent n-gram prob

Class N-Gram Models

How can we assign words to classes?

- with vocab sizes of 50,000+, don't want to do this by hand
- maybe we can do this using statistical methods?
  - similar words tend to occur in similar contexts
  - *e.g.*, beverage words occur to right of word DRINK
- possible algorithm (Schutze, 1992)
  - for each word, collect count for each word occurring one position to left; for each word occurring one position to right; etc.
  - dimensionality reduction: latent semantic analysis (LSA), single value decomposition (SVD)
  - cluster, *e.g.*, with k-means clustering

Class N-Gram Models

How can we assign words to classes? (Brown *et al.*, 1992)

- maximum likelihood!
  - fix number of classes, *e.g.*, 1000
  - find assignment of words to classes that maximizes likelihood of the training data . . .
  - with respect to class bigram model

$$P(w_i|w_{i-1}) = P(w_i \mid \text{class}(w_i)) \times P(\text{class}(w_i) \mid \text{class}(w_{i-1}))$$

- naturally groups words occurring in similar contexts
- directly optimizes an objective function we care about

Class N-Gram Models

How can we assign words to classes? (Brown *et al.*, 1992)

- basic algorithm
  - come up with initial assignment of words to classes
  - repeatedly consider reassigning each word to each other class
    - do move if helps likelihood
  - stop when no more moves help

Example Word Classes

900M words of training data, various sources

THE TONIGHT'S SARAJEVO'S JUPITER'S PLATO'S CHILDHOOD'S
GRAVITY'S EVOLUTION'S

OF

AS BODES AUGURS BODED AUGURED

HAVE HAVEN'T WHO'VE

DOLLARS BARRELS BUSHELS DOLLARS' KILOLITERS

MR. MS. MRS. MESSRS. MRS

HIS SADDAM'S MOZART'S CHRIST'S LENIN'S NAPOLEON'S JESUS'
ARISTOTLE'S DUMMY'S APARTHEID'S FEMINISM'S

ROSE FELL DROPPED GAINED JUMPED CLIMBED SLIPPED TOTALED
EASED PLUNGED SOARED SURGED TOTALING AVERAGED RALLIED
TUMBLED SLID SANK SLUMPED REBOUNDED PLUMMETED TOTALLED
DIPPED FIRMED RETREATED TOTALLING LEAPED SHRANK
SKIDDED ROCKETED SAGGED LEAPT ZOOMED SPURTED NOSEDIVED

Class N-Gram Model Performance

- *e.g.*, class trigram model

$$P(w_i|w_{i-2}w_{i-1}) = P(w_i \mid C(w_i)) \times P(C(w_i) \mid C(w_{i-2})C(w_{i-1}))$$

  - still compute classes using class bigram model
- outperforms word n-gram models with small training sets
- on larger training sets, word n-gram models win ($<$1% absolute WER)
- can we combine the two?

Combining Multiple Models

- *e.g.*, in smoothing, combining a higher-order n-gram model with a lower-order one

$$P_{\text{interp}}(w_i|w_{i-1}) = \lambda_{w_{i-1}} P_{\text{MLE}}(w_i|w_{i-1}) + (1 - \lambda_{w_{i-1}}) P_{\text{interp}}(w_i)$$

- linear interpolation
  - fast
  - combined model probabilities sum to 1 correctly
  - easy to train λ to maximize likelihood of data (EM algorithm)
  - effective

Combining Word and Class N-Gram Models

- linear interpolation — a hammer for combining models

$$P_{\text{combine}}(w_i|w_{i-2}w_{i-1}) = \lambda \times P_{\text{word}}(w_i|w_{i-2}w_{i-1}) + (1-\lambda) \times P_{\text{class}}(w_i|w_{i-2}w_{i-1})$$

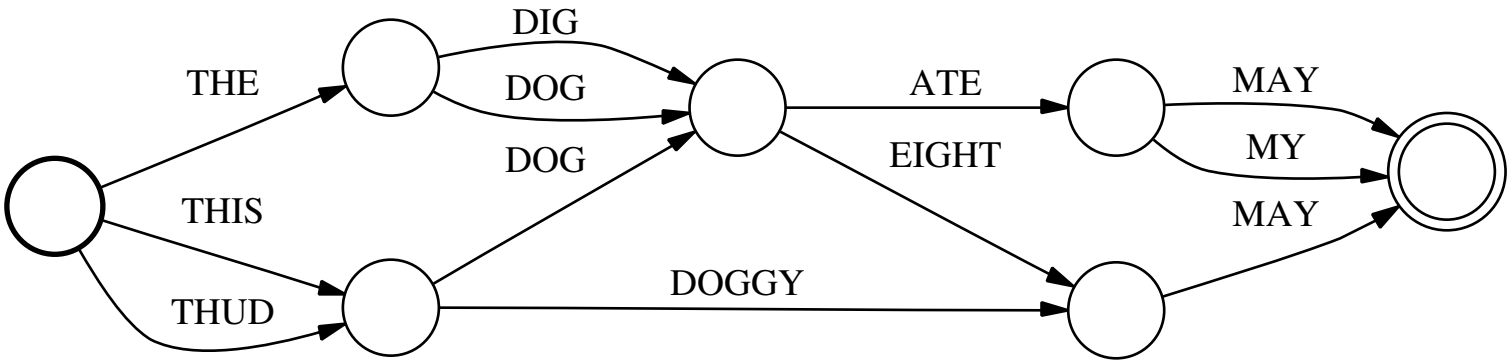
- small gain over either model alone ($<$1% absolute WER)
- state-of-the-art single-domain language model for large training sets (4-grams)
  - . . . in the research community
- conceivably, λ can be history-dependent

Practical Considerations with Class N-Gram Models

- not well-suited to one-pass decoding
  - difficult to build static decoding graph
    - trick with backoff arcs and only storing n-grams with nonzero counts doesn't really work
  - difficult to implement LM lookahead efficiently
    - may not achieve full gain, which is small to begin with
- smaller than word n-gram models
  - n-gram model over vocab of $\sim$1000 rather than $\sim$50000
  - few additional parameters: $P(w_i \mid \text{class}(w_i))$
- easy to add new words to vocabulary
  - only need to initialize $P(w_{\text{new}} \mid \text{class}(w_{\text{new}}))$

Aside: Lattice Rescoring

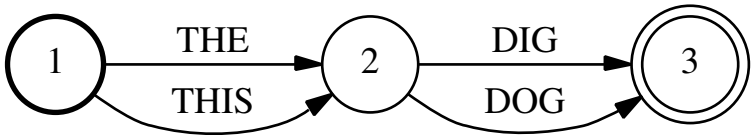
- two-pass decoding
  - generate lattices with, say, word bigram model
  - want to rescore lattices with class 4-gram model



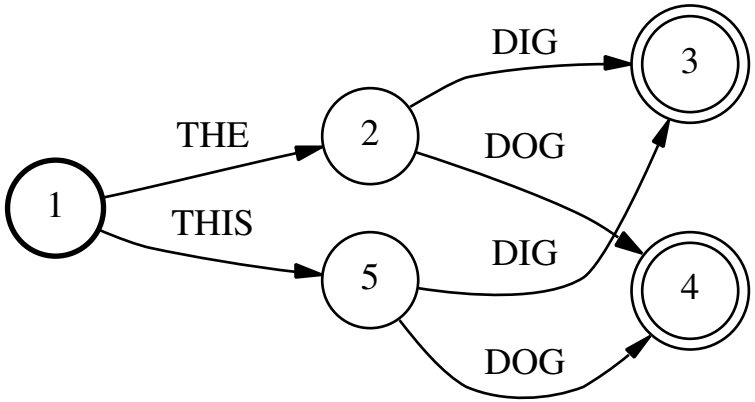
- N-best list rescoring?
  - keep acoustic scores from first pass
  - for each hypothesis, compute new LM score, add in
  - there you go
- lattice may contain exponential number of paths

Lattice Rescoring

- can we just put new LM scores directly on lattice arcs?



  - not without possibly expanding the lattice
  - *e.g.*, bigram model expansion



Lattice Rescoring

- is there an easy way of doing this?
  - $\Rightarrow$ compose lattice with WFSA encoding LM!
  - keep acoustic scores from first pass in lattice
  - composition adds in new LM scores, expanding lattice if needed
  - use DP to find highest scoring path in rescored lattice
- expressing class n-gram model as an WFSA
  - just like for word n-gram model, but use class n-gram probs

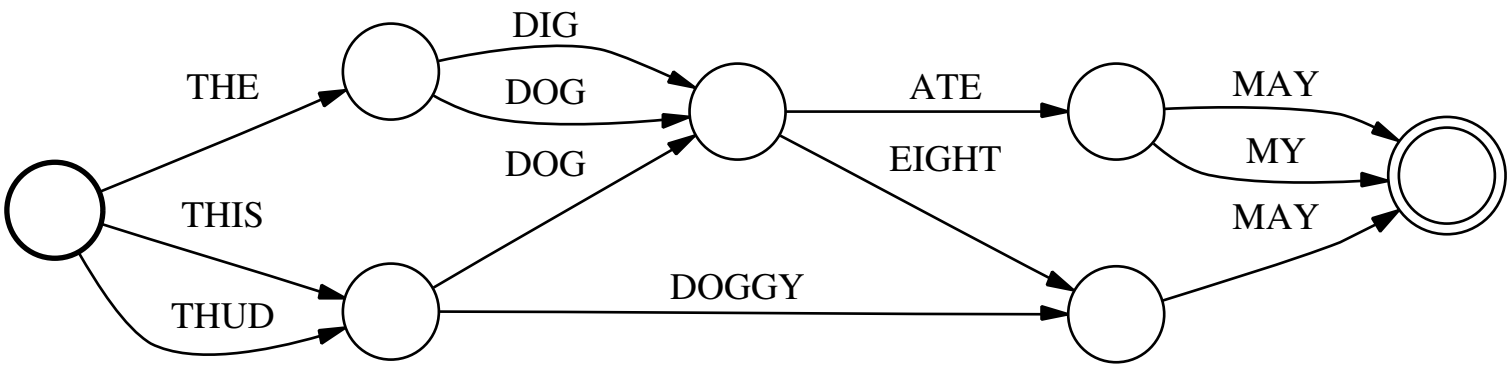
$$P(w_i|w_{i-2}w_{i-1}) = P(w_i \mid C(w_i)) \times P(C(w_i) \mid C(w_{i-2})C(w_{i-1}))$$

- what if WFSA corresponding to LM is too big?
  - dynamic on-the-fly expansion of relevant parts can be done

Aside: Acoustic Lattice Rescoring

How do we rescore lattices with new acoustic models rather than language models?

- have lattice A_{LM} containing LM scores from first pass
- pretend this is the full language model FSA, do regular decoding
  - *i.e.*, expand lattice to underlying HMM via FSM composition; do Viterbi



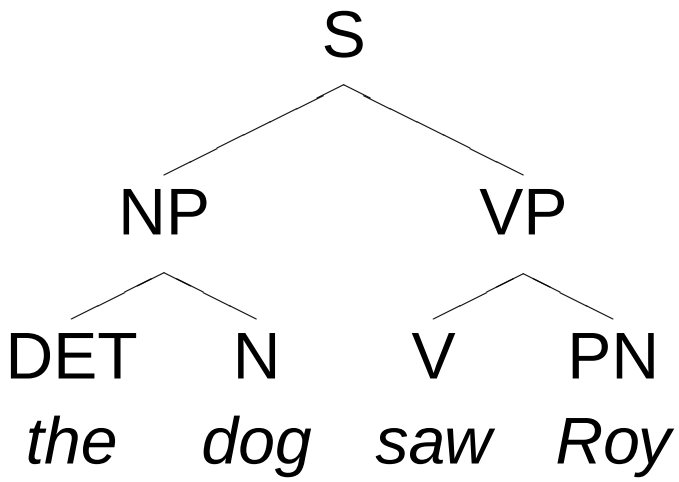
Where Are We?

Unit II: Language modeling for unrestricted domains

- short-distance dependencies
  - class n-gram models
- medium-distance dependencies
  - grammar-based language models
- long-distance dependencies
  - cache and trigger models
  - topic language models
- linear interpolation revisited

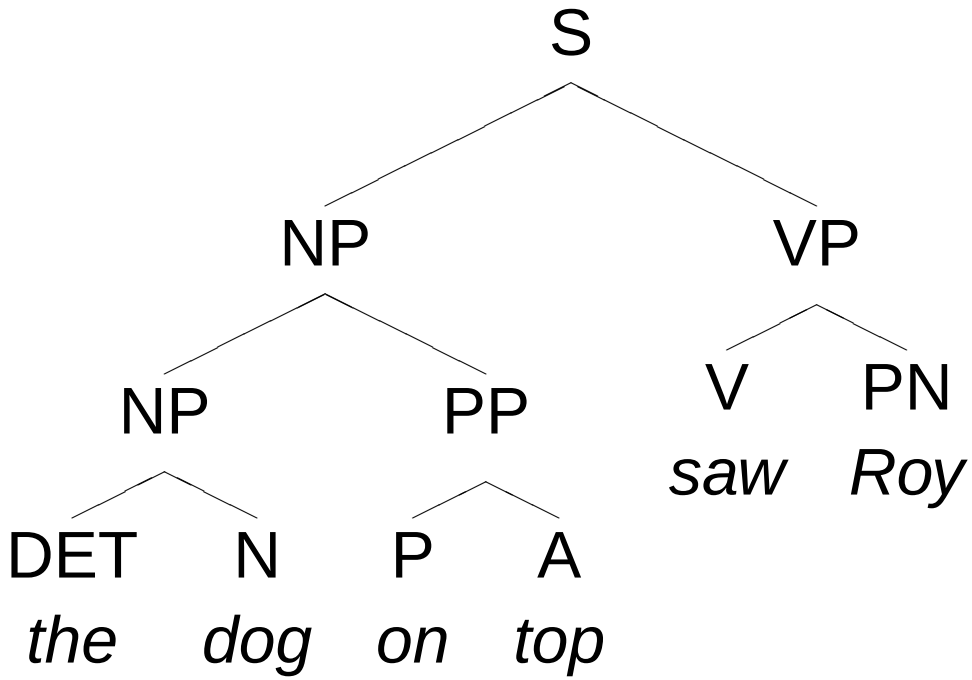
Modeling Medium-Distance Dependencies

- n-gram models predict identity of next word . . .
  - based on identities of words in fixed positions in past
  - *e.g.*, the word immediately to left, and word to left of that
- important words for prediction may occur in many positions
  - important word for predicting *saw* is *dog*



Modeling Medium-Distance Dependencies

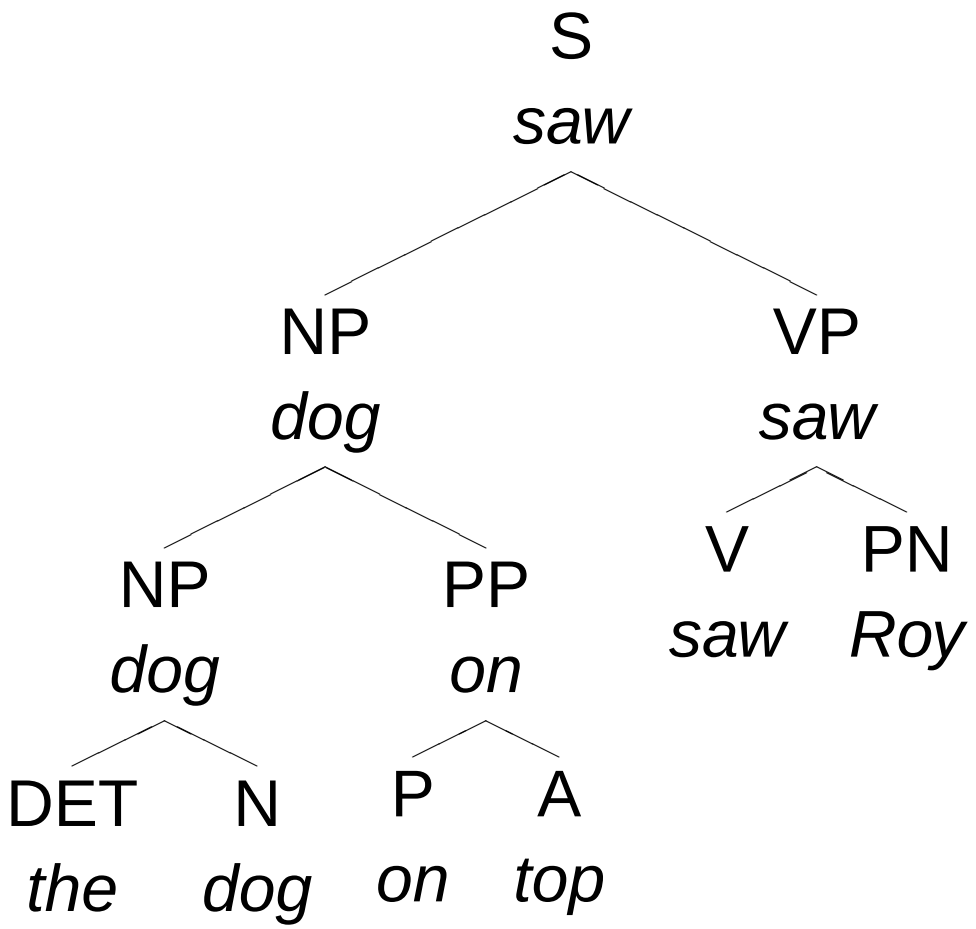
- important words for prediction may occur in many positions
  - important word for predicting *saw* is *dog*



- n-gram model predicts *saw* using words *on top*
- shouldn't condition on words a fixed number of words back?
  - should condition on words in fixed positions in parse tree!?

Modeling Medium-Distance Dependencies

- each constituent has a *headword*
  - predict next word based on preceding *exposed* headwords?



Modeling Medium-Distance Dependencies

- predict next word based on preceding *exposed* headwords

$$
\begin{aligned}
P(\ & \textit{the} \mid \triangleright\ \ \triangleright\ ) \\
P(\ & \textit{dog} \mid \triangleright\ \ \textit{the}\ ) \\
P(\ & \textit{on} \mid \triangleright\ \ \textit{dog}\ ) \\
P(\ & \textit{top} \mid \textit{dog}\ \ \textit{on}\ ) \\
P(\ & \textit{saw} \mid \triangleright\ \ \textit{dog}\ ) \\
P(\ & \textit{Roy} \mid \textit{dog}\ \ \textit{saw}\ )
\end{aligned}
$$

  - picks most relevant preceding words, regardless of position

- *structured language model* (Chelba and Jelinek, 2000)

Structured Language Modeling

Hey, where do those parse trees come from?

- come up with grammar rules

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow DET\ N \mid PN \mid NP\ PP \\
N &\rightarrow \textit{dog} \mid \textit{cat}
\end{aligned}
$$

  - these describe legal constituents/parse trees
- come up with probabilistic parameterization
  - way of assigning probabilities to parse trees
- can extract rules and train probabilities using a *treebank*
  - manually-parsed text, *e.g.*, Penn Treebank (Switchboard, WSJ text)

Structured Language Modeling

- decoding
  - n-grams: find most likely word sequence
  - structured LM: find most likely word sequence and parse tree
- not yet implemented in one-pass decoder
- evaluated via lattice rescoring
  - conceptually, can encode structured LM as WFSA . . .
  - with dynamic on-the-fly expansion of relevant parts

Structured Language Modeling

So, does it work?

- um, -cough-, kind of
- issue: training is expensive
  - SLM trained on 20M words of WSJ text
  - trigram model trained on 40M words of WSJ text
- lattice rescoring
  - SLM: 14.5% WER
  - trigram: 13.7% WER
- well, can we get gains of both?
  - SLM may ignore preceding two words even when useful
  - linear interpolation!? $\Rightarrow$ 12.9%

Structured Language Modeling

Lessons

- grammatical language models not yet ready for prime time
  - need manually-parsed data to bootstrap parser
  - training is expensive; difficult to train on industrial-strength training sets
  - decoding is expensive and difficult to implement
  - a lot of work for little gain; easier to achieve gain with other methods
- if you have an exotic LM and need publishable results . . .
  - interpolate it with a trigram model

Where Are We?

Unit II: Language modeling for unrestricted domains

- short-distance dependencies
  - class n-gram models
- medium-distance dependencies
  - grammar-based language models
- long-distance dependencies
  - cache and trigger models
  - topic language models
- linear interpolation revisited

Modeling Long-Distance Dependencies

A group including Phillip C. Friedman , a Gardena , California , investor , raised its stake in Genisco Technology Corporation to seven . five % of the common shares outstanding .

Neither officials of Compton , California - based Genisco , an electronics manufacturer , nor Mr. Friedman could be reached for comment .

In a Securities and Exchange Commission filing , the group said it bought thirty two thousand common shares between August twenty fourth and last Tuesday at four dollars and twenty five cents to five dollars each .

The group might buy more shares , its filing said .

According to the filing , a request by Mr. Friedman to be put on Genisco's board was rejected by directors .

Mr. Friedman has requested that the board delay Genisco's decision to sell its headquarters and consolidate several divisions until the decision can be " much more thoroughly examined to determine if it is in the company's interests , " the filing said .

Modeling Long-Distance Dependencies

- observation: words in previous sentences are more likely to occur in future sentences
  - *e.g.*, GENISCO, GENISCO'S, FRIEDMAN, SHARES
  - much more likely than what n-gram model would predict
- current formulation of language models $P(\omega = w_1 \cdots w_l)$
  - probability distribution over single utterances $\omega = w_1 \cdots w_l$
  - implicitly assumes independence between utterances (*e.g.*, n-gram model)
  - should model consecutive utterances jointly $P(\vec{\omega} = \omega_1 \cdots \omega_L)$
- language model *adaptation*
  - similar in spirit to acoustic adaptation

Cache and Trigger Language Models

- how to boost probabilities of recently-occurring words?
- idea: combine static n-gram model with n-gram model built on recent data
  - *e.g.*, build bigram model on last k=500 words in current "document", *i.e.*, boost recent bigrams as well as unigrams
  - combine using linear interpolation

$$P_{\text{cache}}(w_i|w_{i-2}w_{i-1}, w_{i-500}^{i-1}) =$$
$$\lambda \times P_{\text{static}}(w_i|w_{i-2}w_{i-1}) + (1-\lambda) \times P_{w_{i-500}^{i-1}}(w_i|w_{i-1})$$

  - *cache language model* (Kuhn and De Mori, 1990)

Cache and Trigger Language Models

- can we improve on cache language models?
  - seeing the word THE doesn't boost the probability of THE in the future
  - seeing the word GENISCO boosts the probability of GENISCO'S in the future; MATSUI boosts YANKEES
- try to automatically induce which words *trigger* which other words
  - given a collection of training documents
  - count how often each pair of words co-occurs in a document
  - find pairs of words that co-occur much more frequently . . .
    - than would be expected if they were unrelated

Trigger Language Models

- combining triggers and a static language model
  - can we do the same thing we did for cache LM's?

$$P_{\text{cache}}(w_i|w_{i-2}w_{i-1}, w_{i-500}^{i-1}) =$$
$$\lambda \times P_{\text{static}}(w_i|w_{i-2}w_{i-1}) + (1-\lambda) \times P_{w_{i-500}^{i-1}}(w_i|w_{i-1})$$

  - when see word, give count to all triggered words instead (unigrams only)
- use maximum entropy models (Lau *et al.*, 1993)

Topic Language Models

- observations: there are groups of words that are all mutual triggers
  - *e.g.*, IMMUNE, LIVER, TISSUE, TRANSPLANTS, etc.
  - corresponding to a *topic*, *e.g.*, medicine
  - may not find all mutual triggering relationships because of sparse data
  - triggering based on single occurrence of single word
    - may be better to accumulate evidence from occurrences of many words
    - disambiguate words with many "senses"
    - *e.g.*, LIVER $\rightarrow$ TRANSPLANTS or CHICKEN?
- $\Rightarrow$ topic language models

Topic Language Models

Basic idea

- assign a topic (or topics) to each document in training corpus
  - *e.g.*, politics, medicine, Monica Lewinsky, cooking, etc.
- for each topic, build a topic-specific language model
  - *e.g.*, train n-gram model only on documents labeled with that topic
- when decoding
  - try to guess the current topic (*e.g.*, from past utterances)
  - use appropriate topic-specific language model(s)

Topic Language Models

Details (*e.g.*, Seymore and Rosenfeld, 1997)

- assigning topics to documents
  - manual labels, *e.g.*, keywords in Broadcast News corpus
  - automatic clustering
    - map each document to point in $\mathcal{R}^{|V|}$; frequency of each word in vocab in document
- guessing the current topic
  - find topic LM's that assign highest likelihood to adaptation data
  - adapt on previous utterances of document, or even whole document

Topic Language Models

Details

- topic LM's may be sparse
  - combine with general LM
  - linear interpolation!

$$P_{\text{topic}}(w_i|w_{i-2}w_{i-1}) =$$

$$\lambda_0 P_{\text{general}}(w_i|w_{i-2}w_{i-1}) + \sum_{t=1}^{T} \lambda_t P_t(w_i|w_{i-2}w_{i-1})$$

Adaptive Language Models/Modeling Long-Distance Dependencies

So, do they work?

- um, -cough-, kind of
- cache models
  - good PP gains ($\sim$20%)
  - small WER gains ($<$1% absolute) possible in low WER domains, *e.g.*, WSJ
  - issue: in ASR, cache only helps if get word correct the first time, in which case you would probably get later occurrences correct anyway

Adaptive Language Models/Modeling Long-Distance Dependencies

So, do they work?

- trigger models
  - good PP gains ($\sim$30%)
  - small WER gains ($<$1% absolute) possible
  - again, if make lots of ASR errors, triggers may hurt as much as they help
- topic models
  - ditto

Adaptive Language Models/Modeling Long-Distance Dependencies

Recap

- large PP gains, but small WER gains
  - in lower WER domains, LM adaptation may help more
- increases system complexity for ASR
  - *e.g.*, how to adapt LM scores if statically compiled decoding graph?
- basically, unclear whether worth the effort
  - not used in most products/live systems?
  - not used in most research evaluation systems

Language Modeling for Unrestricted Domains

Recap

- short-distance dependencies
  - linearly interpolate class n-gram model with word n-gram model
  - $<$1% absolute WER gain; pain to implement
- medium-distance dependencies
  - linearly interpolate grammatical LM with word n-gram model
  - $<$1% absolute WER gain; pain to implement
- long-distance dependencies
  - linearly interpolate adaptive LM with static n-gram model
  - $<$1% absolute WER gain; pain to implement

Where Are We?

Unit II: Language modeling for unrestricted domains

- short-distance dependencies
  - class n-gram models
- medium-distance dependencies
  - grammar-based language models
- long-distance dependencies
  - cache and trigger models
  - topic language models
- linear interpolation revisited

Linear Interpolation Revisited

- if short, medium, and long-distance modeling all achieve $\sim$1% WER gain . . .
  - what happens if we combine them all in one system . . .
  - using our hammer for combining models, linear interpolation?
- "A Bit of Progress in Language Modeling" (Goodman, 2001)
  - combined higher order n-grams, skip n-grams, class n-grams, cache models, and sentence mixtures
  - achieved 50% reduction in PP over baseline trigram (or 1 bit of entropy)
  - $\Rightarrow$ $\sim$1% WER gain (WSJ N-best list rescoring)

Linear Interpolation Revisited

What up?

- intuitively, it's clear that humans use short, medium, and long distance information in modeling language
  - short: BUY BEER, PURCHASE WINE
  - medium: complete, grammatical sentences
  - long: coherent sequences of sentences
- should get gains from modeling each type of dependency
- and yet, linear interpolation failed to yield cumulative gains
  - maybe, instead of a hammer, we need a screwdriver

Linear Interpolation Revisited

Case study

- say, unigram cache model

$$P_{\text{cache}}(w_i|w_{i-2}w_{i-1}, w_{i-500}^{i-1}) =$$
$$0.9 \times P_{\text{static}}(w_i|w_{i-2}w_{i-1}) + 0.1 \times P_{w_{i-500}^{i-1}}(w_i)$$

- compute $P_{\text{cache}}(\text{FRIEDMAN} \mid \text{KENTUCKY FRIED})$
  - where $P_{w_{i-500}^{i-1}}(\text{FRIEDMAN}) = 0.1$
  - $\Rightarrow P_{\text{cache}}(\text{FRIEDMAN} \mid \text{KENTUCKY FRIED}) \approx 0.1 \times 0.1 = 0.01$
- observation: $P_{\text{cache}}(\text{FRIEDMAN} \mid w_{i-2}w_{i-1}, w_{i-500}^{i-1}) \geq 0.01$ for *any* history

Linear Interpolation Revisited

- linear interpolation is like an OR
  - if *either* term being interpolated is high, the final prob is relatively high
- doesn't seem like correct behavior in this case
  - maybe linear interpolation is keeping us from getting full potential gain from each information source
- is there a way of combining things that acts like an AND?
  - want $P_{\text{cache}}(\text{FRIEDMAN} \mid \cdots)$ to be high only in contexts where word FRIEDMAN is plausible
  - *i.e.*, only if *both* terms being combined is high should the final prob be high

Where Are We?

Advanced language modeling

- Unit I: techniques for restricted domains
  - aside: confidence
- Unit II: techniques for unrestricted domains
- Unit III: maximum entropy models
- Unit IV: other directions in language modeling
- Unit V: an apology to n-gram models

Maximum Entropy Modeling

A new perspective on choosing models

- old way
  - manually select model form/parameterization (*e.g.*, Gaussian)
  - select parameters of model to maximize, say, likelihood of training data
- new way (Jaynes, 1957)
  - choose set of *constraints* the model should satisfy
  - choose model that satisfies these constraints . . .
  - over all possible model forms . . .
  - such that the model selected has the maximal entropy

Maximum Entropy Modeling

Remind me what that entropy thing is again?

- for a model or probability distribution $P(\mathbf{x})$ ...
- the entropy $H(P)$ (in bits) of $P(\cdot)$ is

$$H(P) = -\sum_{\mathbf{x}} P(\mathbf{x}) \log_2 P(\mathbf{x})$$

(where $0 \log_2 0 \equiv 0$)

Entropy

- deterministic distribution has zero bits of entropy

$$\begin{aligned} P(\mathbf{x}) &= \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_0 \\ 0 & \text{otherwise} \end{cases} \\ H(P) &= -\sum_{\mathbf{x}} P(\mathbf{x}) \log_2 P(\mathbf{x}) = -1 \log_2 1 = 0 \end{aligned}$$

- uniform distribution over N elements has $\log_2 N$ bits of entropy

$$\begin{aligned} P(\mathbf{x}) &= \frac{1}{N} \\ H(P) &= -\sum_{\mathbf{x}} P(\mathbf{x}) \log_2 P(\mathbf{x}) = -\sum_{\mathbf{x}} \frac{1}{N} \log_2 \frac{1}{N} \\ &= N \times \frac{1}{N} \log_2 N = \log_2 N \end{aligned}$$

Entropy

- with no constraints
  - deterministic distribution is minimum entropy model
  - uniform distribution is maximum entropy model
- information theoretic interpretation
  - average number of bits needed to code sample from $P(\mathbf{x})$
- entropy $\Leftrightarrow$ uniformness $\Leftrightarrow$ least assumptions
- maximum entropy model given some constraints . . .
  - models exactly what you know, and assumes nothing more

Maximum Entropy Modeling

Example: an (unfair) six-sided die

- before we roll it, what distribution would we guess?
  - uniform distribution
  - because, intuitively, this assumes the least
  - . . . as it is the maximum entropy distribution

What Are These Constraint Things?

- rolled the die 20 times, and don't remember everything, but . . .
  - there were a lot of odd outcomes, namely 14, and . . .
  - the value 5 came up seven times

$$f_1(x) = \begin{cases} 1 & \text{if } x \in \{1, 3, 5\} \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_x P(x) f_1(x) = \frac{14}{20} = 0.7$$

$$f_2(x) = \begin{cases} 1 & \text{if } x \in \{5\} \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_x P(x) f_2(x) = \frac{7}{20} = 0.35$$

What Are These Constraint Things?

- $f_i(x)$ are called *features*
  - may specify any subset of possible x values
- choose distribution $P(x)$ such that for each feature $f_i(x)$ . . .
  - expected frequency of $f_i(x)$ being active . . .
  - matches actual frequency of $f_i(x)$ in the training data, *e.g.*,

$$\sum_x P(x) f_1(x) = \frac{14}{20} = 0.7$$

  - of all such $P(x)$, select the one with maximal entropy

What Are These Constraint Things?

- rolled the die 20 times, and don't remember everything, but . . .
  - there were a lot of odd outcomes, namely 14, and . . .
  - the value 5 came up seven times
- the maximum entropy distribution

$$P(x) = (0.175, 0.1, 0.175, 0.1, 0.35, 0.1)$$

  - how can we compute this in general?

Maximum Entropy Modeling

- as it turns out, maximum entropy models have the following form

$$P(x) = \prod_i \alpha_i^{f_i(x)}$$

  - (need to add constant feature $f_0(x) = 1$ for normalization)
  - α_i are parameters chosen such that the constraints are satisfied
  - to compute $P(x)$ for a given x ...
    - if $f_i(x)$ is active/nonzero, multiply in factor α_i

- also called *exponential models* or *log-linear models*

Maximum Entropy Modeling

$$f_1(x) = \begin{cases} 1 & \text{if } x \in \{1,3,5\} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x) = \begin{cases} 1 & \text{if } x \in \{5\} \\ 0 & \text{otherwise} \end{cases}$$

$$P(x) = \prod_i \alpha_i^{f_i(x)}$$

$$P(1) = P(3) = \alpha_0 \alpha_1$$

$$P(2) = P(4) = P(6) = \alpha_0$$

$$P(5) = \alpha_0 \alpha_1 \alpha_2$$

- $\alpha_0 = 0.1, \alpha_1 = 1.75, \alpha_2 = 2$

Maximum Entropy Modeling

- as it turns out, maximum entropy models are also maximum likelihood

$$P(x) = \prod_i \alpha_i^{f_i(x)}$$

  - the α_i that satisfy constraints derived from training data . . .
  - are the same α_i that maximize the likelihood of that training data
    - given a model of the above form

- likelihood of training data is convex function of α_i
  - *i.e.*, single local/global maximum in parameter space
  - easy to find optimal α_i (*e.g.*, iterative scaling)

A Rose By Any Other Name

- motivation for using exponential/log-linear models
  - maximum entropy
- maximum likelihood perspective is more useful
  - can use default distribution: $P(x) = P_0(x) \prod_i \alpha_i^{f_i(x)}$
  - to smooth, can use a prior over α_i and do MAP estimation
  - either case, no longer maximizing entropy
- however, still use name *maximum entropy* because sounds better

And Your Point Was?

Case study

- unigram cache model

$$P_{\text{cache}}(w_i|w_{i-2}w_{i-1}, w_{i-500}^{i-1}) =$$
$$0.9 \times P_{\text{static}}(w_i|w_{i-2}w_{i-1}) + 0.1 \times P_{w_{i-500}^{i-1}}(w_i)$$

- compute $P_{\text{cache}}(\text{FRIEDMAN} \mid \text{KENTUCKY FRIED})$
  - where $P_{w_{i-500}^{i-1}}(\text{FRIEDMAN}) = 0.1$
  - $\Rightarrow P_{\text{cache}}(\text{FRIEDMAN} \mid \text{KENTUCKY FRIED}) \approx 0.1 \times 0.1 = 0.01$
- observation: $P_{\text{cache}}(\text{FRIEDMAN} \mid w_{i-2}w_{i-1}, w_{i-500}^{i-1}) \geq 0.01$ for *any* history
  - linear interpolation acts like OR, we want AND

What About Maximum Entropy Models?

- combine through *multiplication* rather than *addition*

$$P_{\text{cache}}(w_i|w_{i-2}w_{i-1}, w_{i-500}^{i-1}) =$$
$$P_{\text{static}}(w_i|w_{i-2}w_{i-1}) \times \prod_i \alpha_i^{f_i(w_i, w_{i-500}^{i-1})}$$

$$f_1(w_i, w_{i-500}^{i-1}) = \begin{cases} 1 & \text{if } w_i = \text{FRIEDMAN, FRIEDMAN} \in w_{i-500}^{i-1} \\ 0 & \text{otherwise} \end{cases}$$

- where $\alpha_1 \approx 10$
  - the word FRIEDMAN is 10 times more likely than usual if you see the word FRIEDMAN in the last 500 words

Another Tool for Model Combination

Maximum entropy models (unlike linear interpolation)

- this gets the AND behavior we want
  - predict FRIEDMAN with high probability only if . . .
  - FRIEDMAN occurred recently AND . . .
  - the preceding two words are an OK left context for FRIEDMAN
- can combine in individual features rather than whole models
  - add in features to handle whatever model is lacking
- can combine disparate sources of information
  - features can ask arbitrary questions about past, *e.g.*,
    - $f_1(\cdot) = 1$ if . . . and $w_{i-1} =$ THE
    - . . . and last exposed headword is DOG
    - . . . and current topic is POLITICS

Well, How Well Does It Work?

(Rosenfeld, 1996)

- 40M words of WSJ training data
- trained maximum entropy model with . . .
  - n-gram, skip n-gram, and trigger features
- 30% reduction in PP, 2% absolute reduction in WER for lattice rescoring
  - over baseline trigram model
- training time: 200 computer-days

A Slow Boat To China

Why are maximum entropy models so lethargic?

- training updates
  - regular n-gram model: for each word, update $O(1)$ count
  - ME model: for each word, update $O(|V|)$ counts
- normalization — making probs sum to 1
  - same story
  - unnormalized models for fast decoding?

Model Combination Recap

Maximum entropy models and linear interpolation

- each is appropriate in different situations
- *e.g.*, when combining models trained on different domains (Switchboard, BN)
  - linear interpolation is more appropriate
  - a particular sentence is either Switchboard-ish, or news-ish, but not both
- together, they comprise a very powerful tool set for model combination
- maximum entropy models still too slow for prime time

Where Are We?

Advanced language modeling

- Unit I: techniques for restricted domains
  - aside: confidence
- Unit II: techniques for unrestricted domains
- Unit III: maximum entropy models
- Unit IV: other directions in language modeling
- Unit V: an apology to n-gram models

Other Directions in Language Modeling

- blah, blah, blah
  - neural network LM's
  - super ARV LM
  - LSA-based LM's
  - variable-length n-grams; skip n-grams
  - concatenating words together to form units for classing
  - context-dependent word classing
  - word classing at multiple granularities
  - alternate parameterizations of class n-gram probabilities
  - using part-of-speech tags
  - semantic structured LM
  - sentence-level mixtures
  - soft classing
  - hierarchical topic models
  - combining data/models from multiple domains
  - whole-sentence maximum entropy models

Where Are We?

Advanced language modeling

- Unit I: techniques for restricted domains
  - aside: confidence
- Unit II: techniques for unrestricted domains
- Unit III: maximum entropy models
- Unit IV: other directions in language modeling
- Unit V: an apology to n-gram models

An Apology to N-Gram Models

- I didn’t mean what I said about you
- you know I was kidding when I said you are great to poop on

What Do People Use In Real Life?

Deployed commercial systems

- technology
  - mostly n-gram models, grammars, embedded grammars
  - grammar switching based on dialogue state
- users cannot distinguish WER differences of a few percent
  - good user interface design is WAY, WAY, WAY, WAY more important than small differences in ASR performance
- research developments in language modeling
  - not worth the extra effort and complexity
  - difficult to implement in one-pass decoding paradigm

Large-Vocabulary Research Systems

- *e.g.*, government evaluations: Switchboard, Broadcast News
  - small differences in WER matter
  - interpolation of class and word n-gram models
  - interpolation of models built from different corpora
- recent advances
  - super ARV LM's (grammar-based class-based n-gram model)
  - neural net LM's
- modeling medium-to-long-distance dependencies
  - almost no gain in combination with other techniques?
  - not worth the extra effort and complexity
- LM gains pale in comparison to acoustic modeling gains

Where Do We Go From Here?

- n-gram models are just really easy to build
  - can train on billions and billions of words
  - smarter LM's tend to be orders of magnitude slower to train
  - faster computers? data sets also growing
- doing well involves combining many sources of information
  - short, medium, and long distance
  - log-linear models are promising, but slow to train and use
- evidence that LM's will help more when WER's are lower
  - human rescoring of N-best lists (Brill *et al.*, 1998)

The Road Ahead

- week 12: applications of ASR
  - audio-visual speech recognition
  - *Malach* project
- week 13: final presentations
- week 14: going to Disneyland