Lecture 10

Advanced Language Modeling

Bhuvana Ramabhadran, Michael Picheny, Stanley F. Chen

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{bhuvana,picheny,stanchen}@us.ibm.com

17 November 2009

Administrivia

- Lab 4 due Thursday, 11:59pm.
- Lab 3 handed back next week.
  - Answers: `/user1/faculty/stanchen/e6870/lab3_ans/`.
- Main feedback from last lecture.
  - Pace a little fast; derivations were "heavy".

Where Are We?

Review: Language Modeling

- The Fundamental Equation of Speech Recognition.

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

- $P(\omega = w_1 \cdots w_l)$ — models frequencies of word sequences $w_1 \cdots w_l$.
- Helps disambiguate acoustically ambiguous utterances.
  - *e.g.*, THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD

Review: Language Modeling

- Small vocabulary, restricted domains.
  - Write grammar; convert to finite-state acceptor.
  - Or possibly *n*-gram models.
- Large vocabulary, unrestricted domains.
  - *N*-gram models all the way.

Review: N-Gram Models

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

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

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

Review: N-Gram Models

- Maximum likelihood estimation

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

- Smoothing.
    - Better estimation in sparse data situations.

Spam, Spam, Spam, Spam, and Spam

- *N*-gram models are robust.
  - Assigns nonzero probs to all word sequences.
  - Handles unrestricted domains.
- *N*-gram models are easy to build.
  - Can train on plain unannotated text.
  - No iteration required over training corpus.
- *N*-gram models are scalable.
  - Can build models on billions of words of text, fast.
  - Can use larger n with more data.
- *N*-gram models are great!
  - Or are they?

The Dark Side of *N*-Gram Models

- In fact, *n*-gram models are deeply flawed.
- Let us count the ways.

What About Short-Distance Dependencies?

- Poor generalization.
  - Training data contains sentence:

    LET'S EAT STEAK ON TUESDAY
  - Test data contains sentence:

    LET'S EAT SIRLOIN ON THURSDAY
  - Occurrence of STEAK ON TUESDAY ...
  - Doesn't affect estimate of $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$
- Collecting more data won't fix this.
  - (Brown *et al.*, 1992) 350MW training $\Rightarrow$ 15% trigrams unseen.

Medium-Distance Dependencies?

- "Medium-distance" $\Leftrightarrow$ within sentence.
- Fabio example:

  FABIO, WHO WAS NEXT IN LINE, ASKED IF THE TELLER SPOKE . . .

  - Trigram model: $P(\text{ASKED} \mid \text{IN LINE})$

Medium-Distance Dependencies?

- Random generation of sentences with $P(\omega = w_1 \cdots w_l)$:
  - 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 $P(\omega)$
- Reveals what word sequences model thinks is likely.

Trigram Model, 20M Words of WSJ

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

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.
- Why can’t *n*-gram models model this stuff?

Long-Distance Dependencies?

- "Long-distance" $\Leftrightarrow$ between sentences.
- See previous examples.
- In real life, adjacent sentences tend to be on same topic.
  - Referring to same entities, *e.g.*, Clinton.
  - In a similar style, *e.g.*, formal *vs.* conversational.
- Why can't n-gram models model this stuff?
- $P(\omega = w_1 \cdots w_l)$ = frequency of $w_1 \cdots w_l$ as sentence?

Recap: Shortcomings of *N*-Gram Models

- 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!

Where Are We?

Where Are We?

2. Techniques for Restricted Domains
   - Embedded Grammars
   - Using Dialogue State
   - Confidence and Rejection

Improving Short-Distance Modeling

- 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 are good for this:

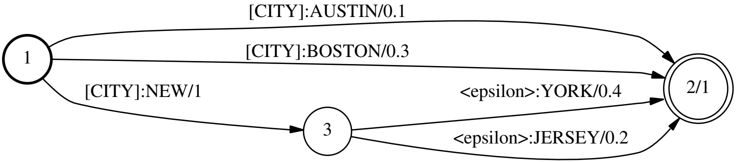
$$\begin{aligned} \text{[sentence]} &\rightarrow \text{I WANT TO FLY FROM [city] TO [city]} \\ \text{[city]} &\rightarrow \text{AUSTIN} \mid \text{BOSTON} \mid \text{JUNEAU} \mid \ldots \end{aligned}$$

- Can we combine robustness of *n*-gram models . . .
  - With generalization ability of grammars?

Combining N-Gram Models with Grammars

- Replace cities and dates, say, in training set with *class* tokens:

  I WANT TO FLY TO [CITY] ON [DATE]
- Build n-gram model on new data, *e.g.*, P([DATE] | [CITY] ON)
- Instead of n-gram model on *words* . . .
  - We have n-gram model over words and *classes*.
- To model probability of class expanding to particular token, use WFSM:



The Model

- Given word sequence $w_1 \cdots w_l$.
  - Substitute in classes to get class/word sequence $C = c_1 \cdots c_{l'}$.

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

$$P(w_1 \cdots w_l) = \sum_C \prod_{i=1}^{l'+1} P(c_i | c_{i-2} c_{i-1}) \times P(\text{words}(c_i) \mid c_i)$$

- Sum over all possible ways to substitute in classes?
  - *e.g.*, treat MAY as verb or date?
  - Viterbi approximation.

Implementing Embedded Grammars

- Need final LM as WFSA.
  - Convert word/class *n*-gram model to WFSM.
  - Compose with transducer expanding each class ...
  - To its corresponding WFSM.
- Static or on-the-fly composition?
  - What if city grammar contains 100,000 cities?

Recap: Embedded Grammars

- Improves modeling of short-distance dependencies.
- Improves modeling of medium-distance dependencies, *e.g.*,

  I WANT TO FLY TO WHITE PLAINS AIRPORT IN FIRST CLASS

  I WANT TO FLY TO [CITY] IN FIRST CLASS
- More robust than grammars alone.

Where Are We?

2. Techniques for Restricted Domains
   - Embedded Grammars
   - Using Dialogue State
   - Confidence and Rejection

Modeling Dependencies Across Sentences

- Many apps involve computer-human *dialogue*.
  - We know what the computer said.
  - We have a good guess of what the human said before.
  - This gives us lots of hints . . .
  - About 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?

Modeling Dependencies Across Sentences

- Switch LM's based on context.
  - *e.g.*, IS THIS FLIGHT OK?
  - *e.g.*, WHICH CITY DO YOU WANT TO FLY TO?
  - If use "correct" LM, accuracy goes way up.
- Boost probabilities of entities mentioned before in dialogue?

There Are No Bad Systems, Only Bad Users

- What if the user doesn't obey the current grammar?
  - Or *any* grammar?
- *e.g.*, system asks: IS THIS FLIGHT OK?
  - User responds: I WANT TO TALK TO AN OPERATOR
  - User responds: HELP, MY PANTS ARE ON FIRE!
- More generally, what if we make ASR errors?
  - Whether utterances are in-grammar or not?
- To gracefully recover from errors ...
  - We need to know when errors are being made.

Where Are We?

2 Techniques for Restricted Domains

- Embedded Grammars
- Using Dialogue State
- Confidence and Rejection

Rejecting Hypotheses With Low Confidence

- *e.g.*, I DID NOT UNDERSTAND; COULD YOU REPEAT?
- How to tell when you have low confidence?
  - Low acoustic likelihood $P(\mathbf{x}|\omega)$?
- Better: posterior probability.
  - How much model prefers hypothesis ω over all others.

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

Computing Posterior Probabilities

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

- Need to sum over sufficiently rich set of hypotheses ω^*.
  - Generate lattice of most likely hypotheses.
  - Which algorithm to compute denominator?
- For out-of-grammar utterances, use *garbage* models.
  - Simple models that kind of cover *any* utterance.
- Issue: language model weight or acoustic model weight?

Recap: Confidence and Rejection

- Accurate rejection essential for usable dialogue systems.
- Posterior probabilities are more or less state-of-the-art.
- Can we use confidence to improve WER?
  - *e.g.*, other information sources, like back-end 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 database directly in LM?)

Where Are We?

Where Are We?

3 Techniques for Unrestricted Domains

- Short-Distance Dependencies: Word Classes
- Aside: Decoding With Advanced LM's
- Medium-Distance Dependencies: Grammars
- Long-Distance Dependencies: Adaptation
- Linear Interpolation Revisited

Class *N*-Gram Models

- Word *n*-gram models do not generalize well.

  LET'S EAT STEAK ON TUESDAY

  LET'S EAT SIRLOIN ON THURSDAY

  - Occurrence of STEAK ON TUESDAY ...
  - Doesn't affect estimate of $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$
- Embedded grammars: *n*-gram models on words and classes.
  - Counts shared among members of same class.

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

Class *N*-Gram Models

- Embedded grammars, typically.
  - Classes can contain phrases, *e.g.*, THIS AFTERNOON.
  - Not all words belong to classes.
  - Same word/phrase may belong to multiple classes.
  - Class grammars are manually constructed.
- Class *n*-gram models, typically.
  - Classes only contain single words.
  - All words are assigned to a class …
  - And only a single class.
  - Classes are induced automatically from data.

$$P(w_1 \cdots w_l) = \sum_{C} \prod_{i=1}^{l'+1} P(c_i | c_{i-2} c_{i-1}) \times P(w_i | c_i)$$

How To Assign Words To Classes?

- With vocab sizes of 50,000+, don't want to do this by hand.
- Basic idea: similar words tend to occur in similar contexts.
  - *e.g.*, beverage words occur to right of word DRINK
- Use one of the zillions of existing clustering algorithms?
  - *e.g.*, map each word to point in $\mathcal{R}^k$ . . .
  - Based on frequency of words in fixed positions to left and right.

The Usual Way (Brown *et al.*, 1992)

- Maximum likelihood!
  - Fix number of classes, *e.g.*, 1000.
  - Choose class assignments to maximize training likelihood …
  - With respect to class bigram model:

$$P(w_1 \cdots w_l) = \prod_{i=1}^{l'+1} P(c_i | c_{i-1}) \times P(w_i | c_i)$$

- Naturally groups words occurring in similar contexts.
- Directly optimizes objective function we care about?
  - Optimize classes for class *bigram* model …
  - Regardless of order of final class *n*-gram model.

How To Do Search?

- Come up with initial assignment of words to classes.
- Consider reassigning each word to each other class.
  - Do move if helps likelihood.
- Stop when no more moves help.

Example Classes, 900MW Training Data

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 TUMBLED SLID SANK SLUMPED REBOUNDED PLUMMETED DIPPED FIRMED RETREATED TOTALLING LEAPED SHRANK SKIDDED ROCKETED SAGGED LEAPT ZOOMED SPURTED RALLIED TOTALLED NOSEDIVED

Class *N*-Gram Model Performance

- On small training sets, better than word *n*-gram models.
- On large training sets, worse than word *n*-gram models.
- Can we combine the two?

Combining Multiple Models

- *e.g.*, interpolated smoothing.

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

- Linear interpolation: A “hammer” for combining models.
  - Fast.
  - Combined model probabilities sum to 1 correctly.
  - Easy to train λ to maximize likelihood of data. (How?)
  - Effective.

Combining Word and Class N-Gram Models

- Gain over either model alone.
  - Conceivably, λ can be history-dependent.

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

	training set (sents.)			
	1k	10k	100k	900k
word n-gram	34.5%	30.4%	25.7%	22.3%
class n-gram	34.8%	30.1%	26.3%	23.9%
interpolated	34.0%	29.0%	24.7%	21.6%

Practical Considerations

- 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 $P(w_{\text{new}} \mid \text{class}(w_{\text{new}}))$.
- How to decode with class *n*-gram models?

Where Are We?

3 Techniques for Unrestricted Domains

- Short-Distance Dependencies: Word Classes
- Aside: Decoding With Advanced LM's
- Medium-Distance Dependencies: Grammars
- Long-Distance Dependencies: Adaptation
- Linear Interpolation Revisited

Decoding With Class *N*-Gram Models

- (First-pass) decoding.
  - Need LM expressed as WFSA.
  - Take class *n*-gram WFSA . . .
  - Compose with transducer rewriting each class as all members.
  - 50000 words/1000 classes = 50 words/class.
- Currently, only word *n*-gram models and grammars.

Lattice Rescoring

- Can be implemented as weighted composition.
  - Again, want LM as WFSA, but need not be *static*.
- On-the-fly composition.
  - Generate states/arcs of machine on demand.
- More generally, on-the-fly *expansion*.
  - Generate states/arcs of machine on demand ...
  - Regardless of how we're doing the expansion.
- Example: word or class n-gram models.
  - *e.g.*, one state for each $(n-1)$-gram history $w_{i-2}w_{i-1}$.
  - Outgoing arc for each w_i with prob $P(w_i | w_{i-2}w_{i-1})$.
  - Avoid backoff approximation from WFSA conversion.

N-Best List Rescoring

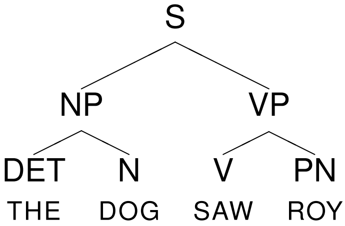
- For each hypothesis $w_1 \ldots w_l$ in N-best list ...
  - Compute $P_{\text{LM}}(w_1 \ldots w_l)$.
- If you can't do this fast, your LM ain't real practical.

Where Are We?

3 Techniques for Unrestricted Domains
- Short-Distance Dependencies: Word Classes
- Aside: Decoding With Advanced LM's
- Medium-Distance Dependencies: Grammars
- Long-Distance Dependencies: Adaptation
- Linear Interpolation Revisited

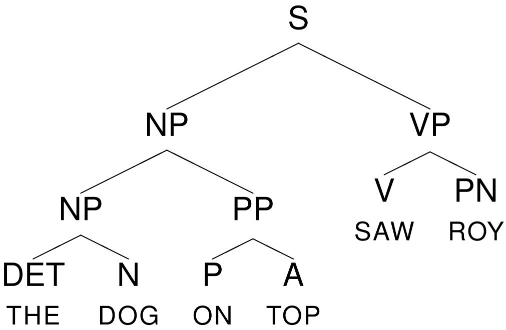
Modeling Medium-Distance Dependencies

- *N*-gram models predict identity of next word . . .
  - Based on identities of words in fixed positions in past.
  - *e.g.*, the two words immediately to left.
- Important words for prediction may occur elsewhere.
  - Important word for predicting SAW is DOG.



Modeling Medium-Distance Dependencies

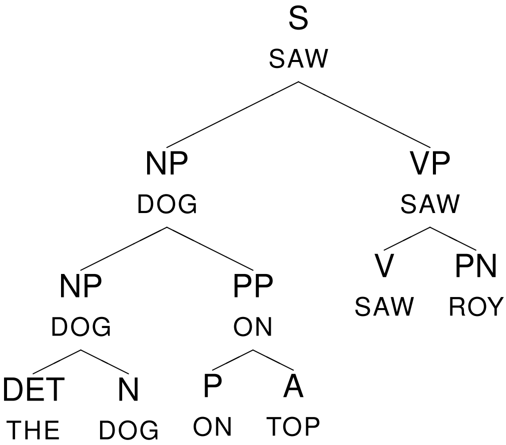
- Important words for prediction may occur elsewhere.
  - Important word for predicting SAW is DOG.



- Instead of condition on a fixed number of words back . . .
  - Condition on words in fixed positions in parse tree!?

Using Grammatical Structure

- Each constituent has a *headword*.
  - Condition on preceding *exposed* headwords?



Using Grammatical Structure

- Predict next word based on preceding *exposed* headwords.

$$
\begin{array}{llcccl}
P(& \text{THE} & | & \triangleright & \triangleright &) \\
P(& \text{DOG} & | & \triangleright & \text{THE} &) \\
P(& \text{ON} & | & \triangleright & \text{DOG} &) \\
P(& \text{TOP} & | & \text{DOG} & \text{ON} &) \\
P(& \text{SAW} & | & \triangleright & \text{DOG} &) \\
P(& \text{ROY} & | & \text{DOG} & \text{SAW} &)
\end{array}
$$

  - Picks most relevant preceding words, regardless of position.
- *Structured language model* (Chelba and Jelinek, 2000).

Hey, Where Do Parse Trees Come From?

- Come up with grammar rules:

$$\begin{array}{ccl} \text{S} & \rightarrow & \text{NP VP} \\ \text{NP} & \rightarrow & \text{DET N} \mid \text{PN} \mid \text{NP PP} \\ \text{N} & \rightarrow & \text{DOG} \mid \text{CAT} \end{array}$$

  - 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*.
  - *e.g.*, Penn Treebank (Switchboard, WSJ text).

Structured Language Modeling

- Decoding: another hidden variable to worry about.
  - *N*-gram models: 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.
  - On-the-fly expansion of equivalent WFSA.

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%

Recap: Structured Language Modeling

- 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?

3 Techniques for Unrestricted Domains

- Short-Distance Dependencies: Word Classes
- Aside: Decoding With Advanced LM's
- Medium-Distance Dependencies: Grammars
- Long-Distance Dependencies: Adaptation
- 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 and phrases in previous sentences ...
  - Are more likely to occur in future sentences.
  - *e.g.*, GENISCO, GENISCO'S, FRIEDMAN, SHARES.
- Language model *adaptation*.
  - Adapt language model to current style or topic.
  - Similar in spirit to acoustic adaptation.
- Distribution over single sentences $P(\omega = w_1 \cdots w_l)$ ...
  - $\Rightarrow$ Distribution over sentence *sequences* $P(\vec{\omega} = \omega_1 \cdots \omega_L)$.

Cache Language Models

- How to boost probabilities of recently-occurring words?
- Idea: build language model on recent words.
  - *e.g.*, last k=500 words in current document.
- How to combine with primary language model?
  - Linear interpolation.

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

- $\Rightarrow$ *Cache language models* (Kuhn and De Mori, 1990).

Beyond Cache Language Models

- What's the problem?
  - Does seeing THE boost the probability of THE?
  - Does seeing MATSUI boost the probability of YANKEES?
- Can we induce which words *trigger* which other words?
  - Let's say your training corpus is subdivided into articles.
  - How might one find trigger pairs?

Trigger Language Models

- How to combine with primary language model?
  - Linear interpolation?
  - Give a word a unigram count every time triggered?

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

- Another way: maximum entropy models (Lau *et al.*, 1993).

Beyond Trigger Language Models

- Some groups of words are mutual triggers.
  - *e.g.*, IMMUNE, LIVER, TISSUE, TRANSPLANTS, etc.
  - Corresponding to a *topic*, *e.g.*, medicine.
  - Difficult to discover all pairwise relations: sparse data.
- May not want to trigger words based on a single word event.
  - Some words are ambiguous.
  - *e.g.*, LIVER $\Rightarrow$ TRANSPLANTS or CHICKEN?
- $\Rightarrow$ Topic language models.

Topic Language Models

- 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.
- Decoding.
  - Try to guess current topic (*e.g.*, from past utterances).
  - Use appropriate topic-specific language model(s).

Example: Seymore and Rosenfeld (1997)

- Assigning topics to documents.
  - One way: manual labels, *e.g.*, Broadcast News corpus.
  - Another way: automatic clustering.
  - Map each document to point in $\mathcal{R}^{|V|}$ ...
  - Based on frequency of each word in vocab.
- Guessing the current topic.
  - Select topic LM's that maximize likelihood of adaptation data.
  - Adapt on previous utterances or first-pass decoding.

Example: Seymore and Rosenfeld (1997)

- Topic LM's may be sparse.
  - Combine with general LM.
- How to combine selected topic LM's and general LM?
  - Linear interpolation!

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

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

So, Do Cache Models Work?

- Um, -cough-, kind of.
- Good PP gains (up to $\sim$20%).
- WER gains: little to none.
  - *e.g.*, (Iyer and Ostendorf, 1999; Goodman, 2001).

What About Trigger and Topic Models?

- Triggers.
  - Good PP gains (up to $\sim$30%)
  - WER gains: unclear; *e.g.*, (Rosenfeld, 1996).
- Topic models.
  - Good PP gains (up to $\sim$30%)
  - WER gains: up to 1% absolute.
  - *e.g.*, (Iyer and Ostendorf, 1999; Goodman, 2001).

Recap: Adaptive Language Modeling

- ASR errors can cause adaptation errors.
  - In lower WER domains, LM adaptation may help more.
- Large PP gains, but small WER gains.
  - What's the dillio?
- Increases system complexity for ASR.
  - *e.g.*, how to adapt LM scores with static decoding?
- Unclear whether worth the effort.
  - Not used in most products/live systems?
  - Not used in most research evaluation systems.

Recap: LM's for Unrestricted Domains

- Short-distance dependencies.
  - Interpolate class *n*-gram with word *n*-gram.
  - $<$1% absolute WER gain; pain to implement.
- Medium-distance dependencies.
  - Interpolate grammatical LM with word *n*-gram.
  - $<$1% absolute WER gain; pain to implement.
- Long-distance dependencies.
  - Interpolate adaptive LM with static *n*-gram.
  - $<$1% absolute WER gain; pain to implement.

IBM

Where Are We?

3. Techniques for Unrestricted Domains
   - Short-Distance Dependencies: Word Classes
   - Aside: Decoding With Advanced LM's
   - Medium-Distance Dependencies: Grammars
   - Long-Distance Dependencies: Adaptation
   - 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: 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.
  - $\Rightarrow$ $\sim$1% WER gain (WSJ *N*-best list rescoring).

What Up?

- Humans use short, medium, and long-distance info.
  - Short: BUY BEER, PURCHASE WINE.
  - Medium: complete, grammatical sentences.
  - Long: coherent sequences of sentences.
- Sources of info seem complementary.
- And yet, linear interpolation fails to yield cumulative gains.
  - Maybe, instead of a hammer, we need a Swiss army knife.

A Thought Experiment: Scenario 1

- Consider a unigram LM with triggers.
- Talking to two people, each on a different topic.
  - $P(\text{OPERA}|\text{NEW YORK}) = 0.01$
  - $P(\text{OPERA}|\text{DETROIT}) = 0.001$
  - $P(\text{OPERA}|\text{NEW YORK}, \text{DETROIT}) = ?$
- *e.g.*, hidden variables; mutually exclusive histories.

$$P(y|x_1, x_2) = \lambda_1 P(y|x_1) + \lambda_2 P(y|x_2)$$

A Thought Experiment: Scenario 2

- Talking to one person about two topics simultaneously.
  - $P(\text{LEWINSKY}|\text{CLINTON}) = 0.01$
  - $P(\text{LEWINSKY}|\text{POLITICS}) = 0.001$
  - $P(\text{LEWINSKY}|\text{CLINTON}, \text{POLITICS}) = ?$
- *e.g.*, dependent topics, one subsuming the other.

$$P(y|x_1, x_2) = P(y|x_1)$$

A Thought Experiment: Scenario 3

- Talking to one person about two topics simultaneously.
  - $P(\text{MATSUI}|\text{YANKEES}) = 0.01$
  - $P(\text{MATSUI}|\text{JAPAN}) = 0.001$
  - $P(\text{MATSUI}|\text{YANKEES}, \text{JAPAN}) = ?$
- *e.g.*, independent topics, partially overlapping.

$$
\begin{aligned}
P(y|x_1, x_2) &= \frac{P(x_1, x_2|y)P(y)}{P(x_1, x_2)} \\
&= \frac{P(x_1|y)P(x_2|y)}{P(x_1)P(x_2)} P(y) \\
&= P(y) \frac{P(y|x_1)}{P(y)} \frac{P(y|x_2)}{P(y)}
\end{aligned}
$$

Combining Information Sources

- Point: the correct way to combine multiple histories ...
  - Very much depends on their relationship!
- The old way.
  - Use linear interpolation ...
  - Because it's *easy*.
- A new way?
  - Can we actually use some principles!?

Where Are We?

Where Are We?

4 Maximum Entropy Models
- Introduction
- Smoothing and *N*-Gram Models, Revisited
- Results

What Do We Really Know?

- Let's say we have some training data $\mathcal{D}$.
  - y_i = current word; x_i = preceding words.

$$\mathcal{D} = \{(x_1, y_1), ..., (x_D, y_D)\}$$

- We want to build a conditional LM: $P^*(y_i|x_i)$.
- Say the word MIX follows the word MEOW ...
  - 40 times in the training data.
- What does this imply about what we want for $P^*(y_i|x_i)$?

$$\text{count}(\text{MEOW}) \times P^*_{\text{avg}}(\text{MIX}|\text{MEOW}) \approx 40$$

$$P^*_{\text{avg}}(\text{MIX}|\text{MEOW}) = \frac{1}{\text{count}(\text{MEOW})} \sum_{d:\text{MEOW last word of } x_d} P^*(\text{MIX}|x_d)$$

Constraints

- Each *constraint* can be viewed as encoding a piece of info.

$$\text{count}(\text{MEOW}) \times P^*_{\text{avg}}(\text{MIX}|\text{MEOW}) = 40$$

- Can combine multiple sources of info ...
  - By just making lots of constraints.
- The point: We want *each* constraint to hold ...
  - Regardless of what *other* constraints we try to enforce.

Constraint-Based Modeling

- The old way.
  - Use linear interpolation . . .
  - Because it's *easy*.
- The new way.
  - Find single model that satisfies . . .
  - All of our constraints simultaneously.

There Can Be Only One

- There may be lots of models $P^*(y_i|x_i)$ ...
  - That satisfy a given set of constraints.
- Example: building a trigram model $P^*(y_i|x_i)$ ...
  - Given five bigram frequency constraints.
- Which model to pick?
  - The one with the *maximum entropy* (Jaynes, 1957).

Maximum Entropy

- Entropy $\Leftrightarrow$ uniformness $\Leftrightarrow$ least assumptions.
- Maximum entropy model given some constraints . . .
  - Models exactly what you know, and assumes nothing more.
- The entropy $H(P)$ of $P(\cdot)$ is

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

- For conditional distribution, maximize (given training $\mathcal{D}$):

$$H(P^*) = -\sum_{d} \sum_{y} P^*(y|x_d) \log P^*(y|x_d)$$

Finding the Maximum Entropy Model

- A general way of representing linear constraints.
- For each constraint, make a *feature* function $f_i(x, y)$ ...
  - That is 1 when the feature is active, 0 otherwise.
- Then, constraints have the form:

$$\sum_d \sum_y P^*(y|x_d) f_i(x_d, y) = \sum_d f_i(x_d, y_d)$$

$$\mathcal{D} = \{(x_1, y_1), ..., (x_D, y_D)\}$$

Expressing Linear Constraints

- What we had before.

$$\text{count}(\text{MEOW}) \times P^*_{\text{avg}}(\text{MIX}|\text{MEOW}) = 40$$

$$P^*_{\text{avg}}(\text{MIX}|\text{MEOW}) = \frac{1}{\text{count}(\text{MEOW})} \sum_{d:\text{MEOW} \in x_d} P^*(\text{MIX}|x_d)$$

- Rearranged.

$$f_i(x, y) = \begin{cases} 1 & \text{if } y = \text{MIX and MEOW last word of } x \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_d \sum_y P^*(y|x_d) f_i(x_d, y) = \sum_d f_i(x_d, y_d)$$

Finding the Maximum Entropy Model

- One feature for each constraint: $f_1(x, y), \ldots, f_F(x, y)$.
- One parameter for each feature: $\Lambda = \{\lambda_i, \ldots, \lambda_F\}$.
- The maximum entropy model has the form:

$$P_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^F \lambda_i f_i(x, y))}{Z_\Lambda(x)}$$

- $Z_\Lambda(x)$ = normalizer = $\sum_{y'} \exp(\sum_{i=1}^F \lambda_i f_i(x, y'))$.
- a.k.a. *exponential models*, *log-linear models*.

How to Find the λ_i's?

- Given a model of the form:

$$P_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^{F} \lambda_i f_i(x, y))}{Z_\Lambda(x)}$$

  - $\{\lambda_i\}$ satisfying constraints (derived from training) …
  - Are also the ML estimates of the $\{\lambda_i\}$!
- Also, training likelihood is convex function of $\{\lambda_i\}$!
- $\Rightarrow$ Can find the $\{\lambda_i\}$ using hill-climbing.
  - *e.g.*, iterative scaling; L-BFGS.

Recap: Maximum Entropy, Part I

- Elegant as all hell.
- Single global optimum when training parameters.
- Principled way to combine lots of information sources.
- But does it blend?

Where Are We?

4 Maximum Entropy Models
- Introduction
- Smoothing and N-Gram Models, Revisited
- Results

Constraint-Based Modeling

- Kneser-Ney smoothing.

$$P_{\text{tri}}^{\text{KN}}(w) = \begin{cases} \frac{c_{\text{tri}}(w) - D}{c_{\text{tri}}(\bullet)} & \text{if } c_{\text{tri}}(w) > 0 \\ (1 - \lambda) \times P_{\text{bi}}^{\text{KN}}(w) & \text{if } c_{\text{tri}}(w) = 0 \end{cases}$$

- $P_{\text{bi}}^{\text{KN}}(w)$ chosen such that …
    - Unigram and bigram marginals of training data are met exactly.

Kneser-Ney Smoothing

- Bigram probabilities $P^{\text{KN}}_{\text{bi}}(w)$:
  - *Not* proportional to how often bigram occurs.
  - Proportional to how many word types that bigram follows.

$$N_{1+}(\bullet w_{i-1} w_i) \equiv |\{w_{i-2} : c(w_{i-2} w_{i-1} w_i) > 0\}|$$

$$P^{\text{KN}}_{\text{bi}}(w_i) = \frac{N_{1+}(\bullet w_{i-1} w_i)}{\sum_{w_i} N_{1+}(\bullet w_{i-1} w_i)}$$

What is a (Conventional) N-Gram Model?

- You gots parameters like:

$$P_{\text{BIG}},\ P_{\text{BIG}|\text{LIKE}},\ P_{\text{BIG}|\text{I LIKE}}$$

- You compute probs like so (for interpolated smoothing):

$$\begin{aligned}
P(\text{BIG}|\text{I LIKE}) &= \mu_{\text{I LIKE}} \times P_{\text{BIG}|\text{I LIKE}} + (1 - \mu_{\text{I LIKE}}) \times P(\text{BIG}|\text{LIKE} \\
P(\text{BIG}|\text{LIKE}) &= \mu_{\text{LIKE}} \times P_{\text{BIG}|\text{LIKE}} + (1 - \mu_{\text{LIKE}}) \times P(\text{BIG}) \\
P(\text{BIG}) &= \mu_{\epsilon} \times P_{\text{BIG}} + (1 - \mu_{\epsilon}) \times P_{\text{unif}}(\text{BIG})
\end{aligned}$$

What is an Exponential N-Gram Model?

- You gots parameters like:

$$\lambda_{\text{BIG}},\ \lambda_{\text{LIKE BIG}},\ \lambda_{\text{I LIKE BIG}}$$

- You compute probs like so:

$$P_\Lambda(\text{BIG}|\text{I LIKE}) = \frac{\exp(\lambda_{\text{BIG}} + \lambda_{\text{LIKE BIG}} + \lambda_{\text{I LIKE BIG}})}{Z_\Lambda(\text{I LIKE})}$$

- Just a different way of parameterizing n-gram models.
  - Can express same set of models.
  - Can convert between parameterizations exactly.

Smoothing for Exponential Models

- The smaller $|\lambda_i|$ is, the smaller its effect ...
  - And the smoother the model.
- Smoothing: pick λ_i's to optimize:

$$\text{obj fn} = \log \text{PP}_{\text{train}} + \frac{1}{(\text{\# train wds})}(\text{penalty for large } |\lambda_i|)$$

- ℓ_2^2 regularization (*e.g.*, Lau, 1994; Chen *et al.*, 2000).
  - Performs similarly to Kneser-Ney smoothing.

$$(\text{penalty}) = \sum_{i=1}^{F} \frac{\lambda_i^2}{2\sigma^2}$$

Recap: Maximum Entropy, Part II

- Constraint-based modeling has shown its worth in smoothing.
- Can express smoothed *n*-gram models using maximum entropy . . .
  - Only simpler.
  - Still single global optimum when training parameters.
- But does it blend?

Where Are We?

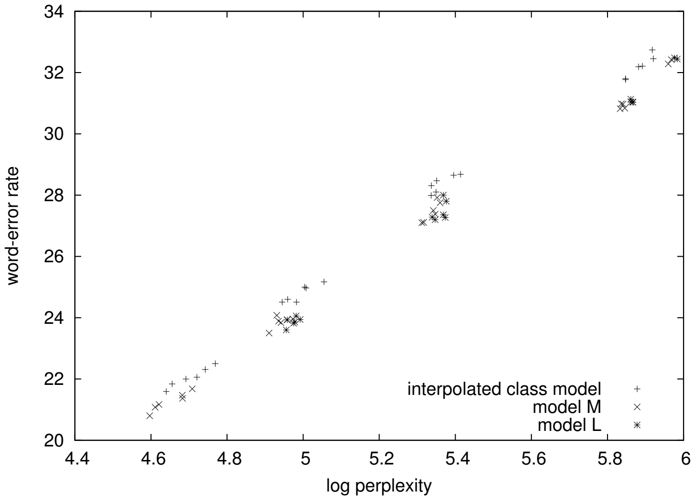
How Well Does It Work? Rosenfeld (1996)

- 38M words of WSJ training data.
- Trained maximum entropy model with . . .
  - *N*-gram, skip *n*-gram, and trigger features.
  - Interpolated with regular word *n*-gram and cache.
- 39% reduction in PP, 2% absolute reduction in WER for lattice rescoring.
  - Baseline: (pruned) Katz-smoothed(?) trigram model.
- Contrast: Goodman (2001), -50% PP, -0.9% WER.

Model M (Chen, 2008)

- Combine class and word *n*-gram features . . .
  - In single maximum entropy model.
- Compared to word trigram: -28% PP; -1.9% WER.
- Without interpolation with word *n*-gram model.

Perplexity *vs.* WER



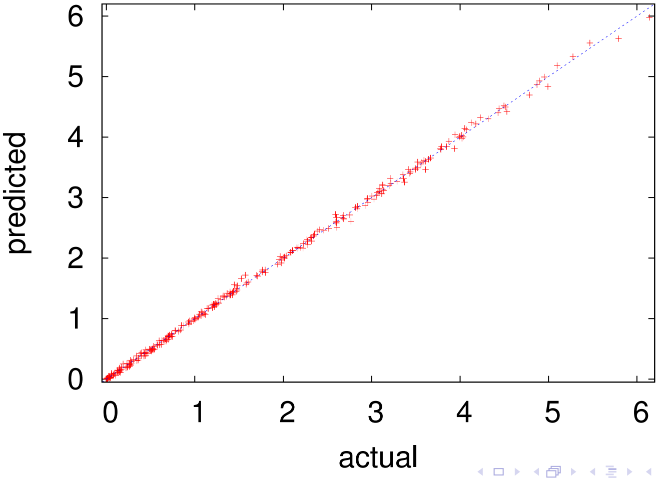
Performance Prediction (Chen, 2008)

- Given training set and test set from same distribution.
- Desire: want to optimize performance on *test* set.
- Reality: only have access to *training* set.

  (test perf) = (training perf) + (overfitting penalty)
- Can we estimate the *overfitting penalty*?

Yes

$$\log \mathrm{PP}_{\mathrm{test}} - \log \mathrm{PP}_{\mathrm{train}} \approx \frac{0.938}{(\#\ \text{train wds})} \sum_{i=1}^{F} |\lambda_i|$$



A Tool for Good

- Holds for many different types of data.
  - Different domains; languages; token types; vocabulary sizes; training set sizes; *n*-gram order.
- Holds for many different types of exponential models.
  - Word *n*-gram models; class-based *n*-gram models; minimum discrimination information models.
- Explains lots of diverse aspects of language modeling.

What's the Catch?

- Rosenfeld (1996): 200 computer-days to train.
- Slow training *vs.* regular *n*-gram model.
  - For each word, update $O(|V|)$ counts.

$$\sum_d \sum_y P^*(y|x_d) f_i(x_d, y) = \sum_d f_i(x_d, y_d)$$

  - Tens of passes through training data.
- Slow evaluation.
  - We have to evaluate $Z_\Lambda(x)$. Or do we?

$$\begin{aligned} P_\Lambda(y|x) &= \frac{\exp(\sum_{i=1}^F \lambda_i f_i(x, y))}{Z_\Lambda(x)} \\ Z_\Lambda(x) &= \sum_{y'} \exp(\sum_{i=1}^F \lambda_i f_i(x, y')) \end{aligned}$$

Recap: Maximum Entropy

- Some of the best WER results in LM literature.
  - Gain of 2%+ absolute WER over trigram (instead of <1%).
- Can surpass linear interpolation in WER in many contexts.
  - *Log*-linear interpolation.
  - Each is appropriate in different situations. (When?)
  - Together, powerful tool set for model combination.
- Performance prediction explains existing models …
  - And helps design new ones!
- Training is still too painful for most people.

Where Are We?

Other Directions in Language Modeling

Neural network LM's.
Super ARV LM.
LSA-based LM's.
Variable-length *n*-grams; skip *n*-grams.
Concatenating words to use in classing.
Context-dependent word classing.
Word classing at multiple granularities.
Alternate parameterizations of class *n*-grams.
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?

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 Is Used In Real Deployed 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 word *n*-gram models . . .
  - Built from different corpora.
  - Neural net LM's; Model M (+0.5% WER?)
- 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.
- Need to effectively combine 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 11: Discriminative training; ROVER; consensus.
- Week 12: Applications of ASR.
  - Speech-to-speech translation.
  - Spoken document retrieval.
- Week 13: Final presentations.

Course Feedback

1. Was this lecture mostly clear or unclear? What was the muddiest topic?
2. Other feedback (pace, content, atmosphere)?