Language Modeling

- Goal: estimate $P(\omega = w_1 \cdots w_l)$.
    - Frequency of word sequence $w_1 \cdots w_l$.
- Helps disambiguate acoustically ambiguous utterances.

THIS IS OUR ROOM FOR A FOUR HOUR PERIOD .
THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD

$$\omega^* = \operatorname*{arg\,max}_{\omega} P(\omega|\mathbf{x}) = \operatorname*{arg\,max}_{\omega} P(\omega)P_\omega(\mathbf{x})$$

- Analogy: multiple-choice test.
    - LM restricts choices given to acoustic model.
    - The fewer choices, the better you do.

Review: Language Modeling

- Decompose probability of sequence ...
  - Into product of conditional probabilities.
- *e.g.*, trigram model ⇒ Markov order 2 ⇒ ...
  - Remember last 2 words.

$$P(w_1 \cdots w_L) = \prod_{i=1}^{L+1} P(w_i | w_{i-2} w_{i-1})$$

$$P(\text{I LIKE TO BIKE}) = P(\text{I} | \triangleright \triangleright) \times P(\text{LIKE} | \triangleright \text{ I}) \times P(\text{TO} | \text{I LIKE}) \times P(\text{BIKE} | \text{LIKE TO}) \times P(\triangleleft | \text{TO BIKE})$$

Review: *N*-Gram Models

- Maximum likelihood estimation.

$$P_{\text{MLE}}(\textsc{to}|\textsc{i like}) = \frac{c(\textsc{i like to})}{c(\textsc{i like})}$$

- Smoothing.
  - Helps when data is sparse, *e.g.*, for low counts.

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 $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$.
- More data won't fix this, *e.g.*, (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 ∞-sided die where ...
  - Each side labeled with word sequence ω ...
  - And probability of landing on that side 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.
  - About single subject.
  - May even be grammatical.
- Why can’t *n*-gram models model this stuff?

Long-Distance Dependencies?

- "Long-distance" $\Leftrightarrow$ between sentences.
- See data generated from trigram model.
- 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, dumb idea.
    - Insult to language modeling researchers.
    - Great for me to poop on.
    - *N*-gram models, . . . you're fired!

Part I

Language Modeling, Pre-2005-ish

Where Are We?

Improving Short-Distance Modeling

- Word n-gram models do not generalize well.
  - Occurrence of STEAK ON TUESDAY ...
  - Doesn't affect $P(\text{THURSDAY} \mid \text{SIRLOIN ON})$.
- Idea: word n-gram ⇒ class n-grams!?

$$P_{\text{MLE}}([\text{DAY}] \mid [\text{FOOD}]\ [\text{PREP}]) = \frac{c([\text{FOOD}]\ [\text{PREP}]\ [\text{DAY}])}{c([\text{FOOD}]\ [\text{PREP}])}$$

- Any instance of class trigram increases ...
  - Probs of all other instances of class trigram.
  - ⇒ Generalization!

Getting From Class to Word Probabilities

- What we have:

$$P([\text{DAY}] \mid [\text{FOOD}]\ [\text{PREP}]) \Leftrightarrow P(c_i | c_{i-2} c_{i-1})$$

- What we want:

$$P(\text{THURSDAY} \mid \text{SIRLOIN ON}) \Leftrightarrow P(w_i | w_{i-2} w_{i-1})$$

- Predict current word given (hidden) class.
  - Simplification: each word belongs to single class.

$$P(w_i | w_{i-2} w_{i-1}) = \sum_{c_i} P(c_i | c_{i-2} c_{i-1}) \times P(w_i | c_i)$$

$$P(\text{THURSDAY} \mid \text{SIRLOIN ON}) = \\ P([\text{DAY}] \mid [\text{FOOD}]\ [\text{PREP}]) \times P(\text{THURSDAY} \mid [\text{DAY}])$$

How To Assign Words To Classes?

- For generalization to work sensibly . . .
  - Group “related” words in same class.

ROSE FELL DROPPED GAINED JUMPED CLIMBED SLIPPED
HEYDAY MINE’S STILL MACHINE NEWEST HORRIFIC BEECH

- With vocab sizes of 50,000+, can’t do this manually.
  - $\Rightarrow$ Unsupervised clustering.

Use Existing Clustering Algorithms on $\mathcal{R}^k$?

- Basic idea: similar words tend to occur in similar contexts.
  - *e.g.*, beverages occur to right of word DRINK.
- Characterize each word by distribution of words …
  - That occur to left and right.
  - *e.g.*, $w_i \Rightarrow (P_L(w_{i-1}|w_i), P_R(w_{i+1}|w_i))$.

A Better Way (Brown *et al.*, 1992)

- Goal: find word classes such that . . .
  - Class trigram model gives good performance.
- Idea: find word classes such that . . .
  - Class bigram model gives good performance.

$$P(w_i | w_{i-1}) = P(c_i | c_{i-1}) \times P(w_i | c_i)$$

- Optimize likelihood of training data (MLE).
  - Fix number of classes, *e.g.*, 1000.
- Directly optimizes objective function we care about.

How To Do Search?

- Hill climbing.
- 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

OF

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

AS BODES AUGURS BODED AUGURED

HAVE HAVEN'T WHO'VE

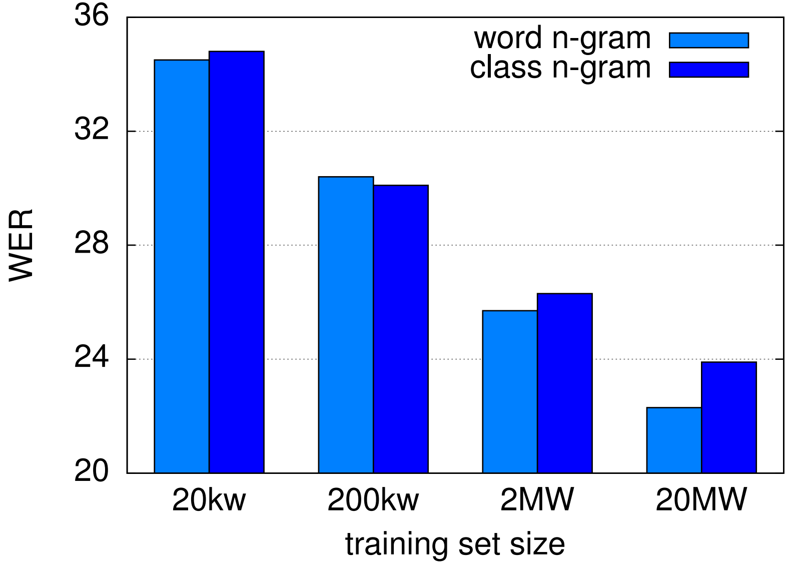
DOLLARS BARRELS BUSHELS DOLLARS' KILOLITERS

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 (WSJ)

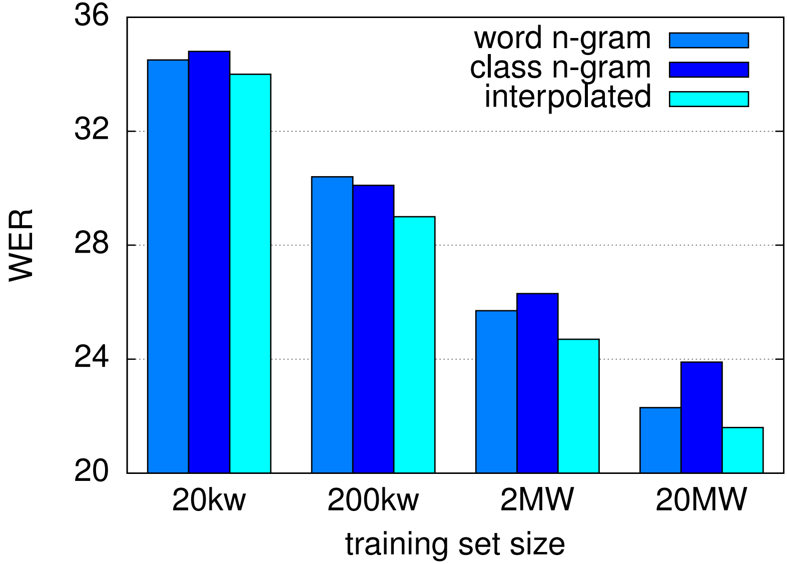


Combining Multiple Models

- On small training sets, class better than word models.
  - On large training sets, word better than class.
  - Can we combine the two?
- Linear interpolation: A “hammer” for combining models.
  - Combined model probabilities sum to 1 correctly.
  - Easy to train λ to maximize likelihood of data. (How?)
  - Fast and effective?

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

Combining Word and Class *N*-Gram Models



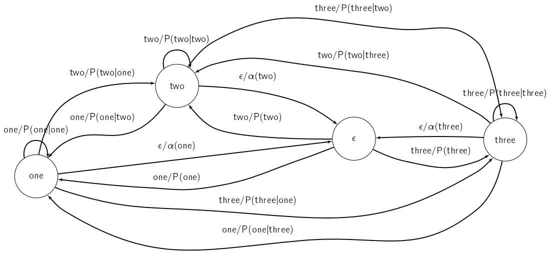
Discussion: Class *N*-Gram Models

- Smaller than word *n*-gram models.
  - *N*-gram model over vocab of $\sim$1000, not $\sim$50000.
  - Few additional parameters: $P(w_i \mid c_i)$.
  - Interpolation $\Rightarrow$ overall model larger.
- Easy to add new words to vocabulary.
  - Only need to initialize $P(w_{\text{new}} \mid c_{\text{new}})$.

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

Discussion

- Static decoding?
  - Start with class *n*-gram model as FSA.
  - Expand each class to *all* members (generalization).

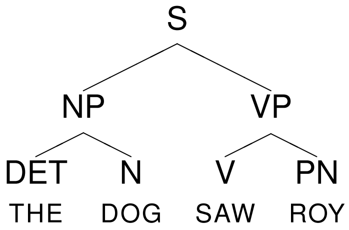


- Dynamic decoding or lattice rescoring only.

Where Are We?

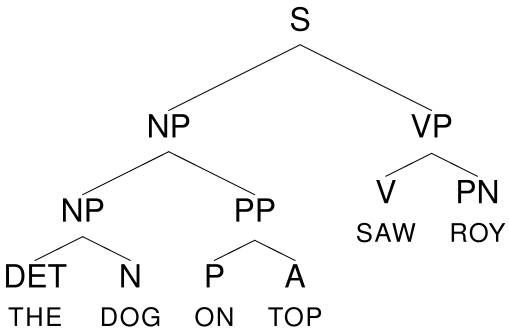
Modeling Medium-Distance Dependencies

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



Modeling Medium-Distance Dependencies

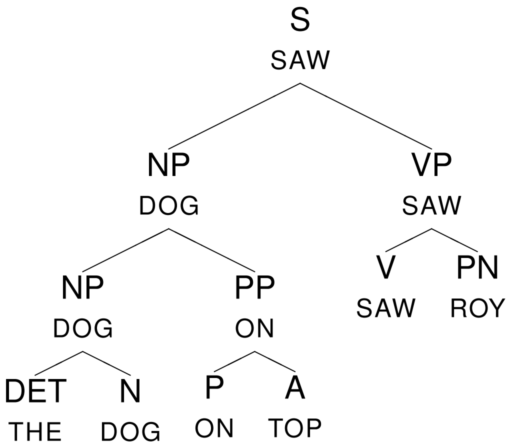
- Important words for prediction may occur elsewhere.
  - Important word for predicting SAW is DOG.



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

Using Grammatical Structure

- Each constituent has *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 . . .
  - That describe legal constituents/parse trees.

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

- Come up with probabilistic parametrization.
  - Way of assigning probabilities to parse trees.

$$P_{\text{MLE}}(\text{S} \rightarrow \text{NP VP}) = \frac{c(\text{S} \rightarrow \text{NP VP})}{c(\text{S})}$$

- Can extract rules and train probabilities using *treebank*.
  - *e.g.*, Penn Treebank (Switchboard, WSJ text).

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.
- Can we get gains of both?
  - 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; hard to scale.
  - “Decoding” is expensive; difficult to implement.
  - Easier to achieve gain with other methods.
- If have exotic LM and need publishable results . . .
  - Interpolate with trigram model (“ROVER effect”).

Where Are We?

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 $P(\omega = w_1 \cdots w_l)$ …
  - $\Rightarrow$ 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.

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

- *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?

HENSON	MUPPETS
TELESCOPE	ASTRONOMERS
CLOTS	DISSOLVER
NODES	LYMPH
SPINKS	HEAVYWEIGHT
DYSTROPHY	MUSCULAR
FEEDLOTS	FEEDLOT
SCHWEPPES	MOTT'S

Trigger Language Models

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

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

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

Topic Language Models

- Assign topic(s) to each document in training corpus.
  - *e.g.*, politics, medicine, Monica Lewinsky, cooking, etc.
- For each topic, build topic-specific language model.
  - *e.g.*, train *n*-gram model only on documents . . .
  - Labeled with 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 current topic.
  - Select topic LM's maximizing likelihood of test data.
  - Adapt on previous utterances or first-pass decoding.

Example: Seymore and Rosenfeld (1997)

- Training (transcript); topics: conspiracy; JFK assassination.

  THEY WERE RIDING THROUGH DALLAS WITH THE KENNEDYS WHEN THE FAMOUS SHOTS WERE FIRED

  HE WAS GRAVELY WOUNDED

  HEAR WHAT GOVERNOR AND MRS. JOHN CONNALLY THINK OF THE CONSPIRACY MOVIE J. F. K. . . .

- Test (decoded); topics: ???

  THE MURDER OF J. F. K. WAS IT A CONSPIRACY

  SHOULD SECRET GOVERNMENT FILES BE OPENED TO THE PUBLIC

  CAN THE TRAGIC MYSTERY EVER BE SATISFACTORILY RESOLVED . . .

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

- 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.
- PP $\neq$ WER.

References

- P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, R.L. Mercer, ”Class-based n-gram models of natural language”, Computational Linguistics, vol. 18, no. 4, pp. 467–479, 1992.
- C. Chelba and F. Jelinek, “Structured language modeling”, Computer Speech and Language, vol. 14, pp. 283–332, 2000.
- J.T. Goodman, “A Bit of Progress in Language Modeling”, Microsoft Research technical report MSR-TR-2001-72, 2001.
- R. Iyer and M. Ostendorf, “Modeling Long Distance Dependence in Language: Topic Mixtures vs. Dynamic Cache Models”, IEEE Transactions on Speech and Audio Processing, vol. 7, no. 1, pp. 30–39, 1999.

References (cont'd)

- R. Kuhn and R. De Mori, "A Cache-Based Natural Language Model for Speech Reproduction", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 6, pp. 570–583, 1990.
- R. Lau, R. Rosenfeld, S. Roukos, "Trigger-based Language Models: A Maximum Entropy Approach", ICASSP, vol. 2, pp. 45–48, 1993.
- R. Rosenfeld, "A Maximum Entropy Approach to Adaptive Statistical Language Modeling", Computer Speech and Language, vol. 10, pp. 187–228, 1996.
- K. Seymore and R. Rosenfeld, "Using Story Topics for Language Model Adaptation", Eurospeech, 1997.

Part II

Language Modeling, Post-2005-ish

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.
- Yet, linear interpolation fails to yield cumulative gains.
  - Maybe instead of hammer, need screwdriver?

How Should a Good LM Act?

- Say we have 1M sentences of training data $\mathcal{D}$.

  FEDERAL HOME LOAN MORTGAGE CORPORATION –DASH ONE .POINT FIVE BILLION DOLLARS OF REALESTATE MORTGAGE -HYPHEN INVESTMENT CONDUIT SECURITIES OFFERED BY MERRILL LYNCH &AMPERSAND COMPANY .PERIOD

  NONCOMPETITIVE TENDERS MUST BE RECEIVED BY NOON EASTERN TIME THURSDAY AT THE TREASURY OR AT FEDERAL RESERVE BANKS OR BRANCHES .PERIOD …

- Build LM $P(\omega)$ on this data.
- Generate 1M sentences of text $\mathcal{D}'$ according to $P(\omega)$.
- If word THE occurs $c(\text{THE})$ times in $\mathcal{D}$ …
  - How many times should occur in $\mathcal{D}'$ for “good” LM?
  - What about for bigram OF THE? Or any other n-gram?

Are Interpolated LM's "Good"?

- Build two models completely independently.
- Linearly interpolate, *e.g.*,

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

- Any guarantees resulting model is "good"?
- Example: tuned word/class *n*-gram model, 1M sentences.
  - $c_{\mathcal{D}}(\text{IN HONG KONG}) = 564$.
  - $c_{\mathcal{D}'}(\text{IN HONG KONG}) = 458$.

Is There Another Way?

- Can we combine multiple information sources . . .
  - *e.g.*, short, medium, and long-distance info . . .
- Such that resulting language model . . .
  - Is guaranteed to be “good”?

Where Are We?

Constraint-Based Modeling

- Come up with set of constraints on final LM.
  - Find LM that satisfies all the constraints.
- *e.g.*, want $P(\omega)$ such that when generate 1M sentences ...
  - IN HONG KONG occurs 564 times on average.
- Let $f_{\text{IN HONG KONG}}(\omega)$ be number of times ...
  - IN HONG KONG occurs in ω.

$$10^6 \times E_{P(\omega)}[f_{\text{IN HONG KONG}}(\omega)] = 564$$
$$10^6 \times \sum_{\omega} P(\omega) f_{\text{IN HONG KONG}}(\omega) = 564$$

More Generally

- Denote training data $\mathcal{D}$ as $(\omega_1, \ldots, \omega_D)$.
- Count of IN HONG KONG in training data:

$$\sum_{d=1}^{D} f_{\text{IN HONG KONG}}(\omega_d) = 564$$

- Then, our constraint becomes:

$$\sum_{d=1}^{D} \sum_{\omega} P(\omega) f_{\text{IN HONG KONG}}(\omega) = \sum_{d=1}^{D} f_{\text{IN HONG KONG}}(\omega_d)$$

Constraints and Feature Functions

- Each feature function $f_\alpha(\omega)$ determines a constraint:

$$\sum_{d=1}^{D} \sum_{\omega} P(\omega) f_\alpha(\omega) = \sum_{d=1}^{D} f_\alpha(\omega_d)$$

- What can feature functions $f_\alpha(\omega)$ look like?
  - How many times IN HONG KONG occurs in ω?
  - How many times [FOOD] [PREP] [DAY] occurs in ω?
  - Return 1 if ω is grammatical, 0 otherwise.
  - Return 1 if both HENSON and MUPPETS occur in ω.
  - Or anything else that can be computed!

Constraint-Based Modeling

- Given set of feature functions/constraints:

$$\sum_{d=1}^{D}\sum_{\omega} P(\omega) f_1(\omega) = \sum_{d=1}^{D} f_1(\omega_d)$$

$$\sum_{d=1}^{D}\sum_{\omega} P(\omega) f_2(\omega) = \sum_{d=1}^{D} f_2(\omega_d)$$

$$\dots\dots\dots\dots\dots\dots\dots$$

- How to find model $P(\omega)$ satisfying constraints?
- Problem: in general, many $P(\omega)$ possible.
    - Which one to pick?

Maximum Entropy Principle (Jaynes, 1957)

- The entropy $H(P)$ of $P(\omega)$ is

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

- Entropy $\Leftrightarrow$ uniformness $\Leftrightarrow$ least assumptions.
- Of models satisfying constraints . . .
  - Pick one with highest entropy!
  - Capture constraints; assume nothing more!

Can We Find the Maximum Entropy Model?

- Given features $f_1(x, y), \ldots, f_F(x, y)$.
- ME model satisfying associated constraints has form:

$$P_\Lambda(\omega) = \frac{1}{Z_\Lambda} \exp(\sum_{i=1}^{F} \lambda_i f_i(\omega))$$

- One parameter per feature: $\Lambda = \{\lambda_i, \ldots, \lambda_F\}$.
  - If $f_i(\omega) \neq 0$, multiply prob by $e^{\lambda_i f_i(\omega)}$.
- Z_Λ = normalizer = $\sum_\omega \exp(\sum_{i=1}^F \lambda_i f_i(\omega))$.
- a.k.a. *exponential model*, *log-linear model*.

How to Find the λ_i's?

$$P_\Lambda(\omega) = \frac{1}{Z_\Lambda} \exp(\sum_{i=1}^{F} \lambda_i f_i(\omega))$$

- $\{\lambda_i\}$'s satisfying constraints ...
  - Are ML estimates of $\{\lambda_i\}$!
- Training set likelihood is convex function of $\{\lambda_i\}$!
  - Can find $\{\lambda_i\}$ using hill-climbing.
  - *e.g.*, iterative scaling; L-BFGS.

Conditional Modeling (Berger *et al.*, 1996)

- Joint formulation: ω = sentence.

$$\mathcal{D} = (\omega_1, \ldots, \omega_D) \qquad \sum_{d=1}^{D} \sum_{\omega} P(\omega) f_i(\omega) = \sum_{d=1}^{D} f_i(\omega_d)$$

$$P_\Lambda(\omega) = \frac{1}{Z_\Lambda} \exp(\sum_{i=1}^{F} \lambda_i f_i(\omega))$$

- Conditional formulation: h = history; w = predicted word.

$$\mathcal{D} = ((h_1, w_1), \ldots, (h_D, w_D)) \quad \sum_{d=1}^{D} \sum_{w} P(w|h_d) f_i(h_d, w) = \sum_{d=1}^{D} f_i(h_d, w_d)$$

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

Recap: Maximum Entropy Modeling

- Elegant as all hell.
- Principled way to combine lots of information sources.
  - Design choice: which constraints to enforce?
  - Can use arbitrary feature functions!
- Single global optimum when training parameters.
  - Given features, “straightforward” to compute LM.
- But does it blend?

Where Are We?

Maximum Entropy *N*-gram Models?

- Can maximum entropy modeling . . .
  - Help build a better word *n*-gram model?
- Conditional formulation: one constraint per seen *n*-gram.

$$f_{\text{I LIKE BIG}}(h, w) = \begin{cases} 1 & \text{if } hw \text{ ends in I LIKE BIG} \\ 0 & \text{otherwise} \end{cases}$$

- Problem: MLE model is same as before!!!
  - Assigns zero-ish probs to unseen *n*-grams.
  - Maximum entropy-ness doesn't prevent overfitting.

Smoothing for Exponential Models

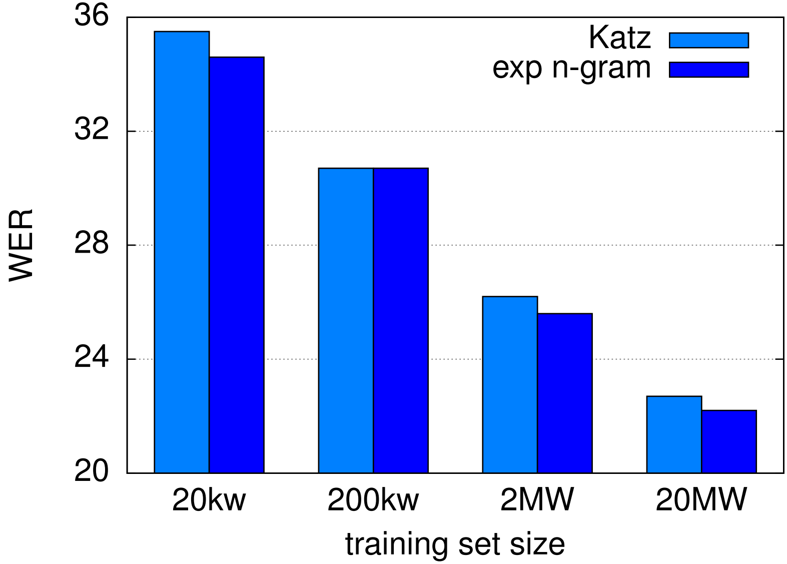
- Point: don't want to match training counts *exactly*!
- Can implement "fuzzy" constraints via penalty term:

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

- The smaller $|\lambda_i|$ is, the smaller its effect ...
  - And the smoother the model.
- *e.g.*, ℓ_2^2 regularization (*e.g.*, Chen and Rosenfeld, 2000).

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

Smoothing for Exponential Models (WSJ 4g)



Yay!?

- Smoothed exponential *n*-gram models perform well.
- Why don't people use them?
  - Conventional *n*-gram: count and normalize.
  - Exponential *n*-gram: run 100 iterations of training.
- Is there way to do constraint-based modeling ...
  - Within conventional *n*-gram framework?

Kneser-Ney Smoothing (1995)

- Back-off smoothing.

$$P_{\text{KN}}(w_i|w_{i-1}) = \begin{cases} P_{\text{primary}}(w_i|w_{i-1}) & \text{if } c(w_{i-1}w_i) > 0 \\ \alpha_{w_{i-1}} P_{\text{KN}}(w_i) & \text{otherwise} \end{cases}$$

- $P_{\text{KN}}(w_i)$ chosen such that ...
  - Unigram constraints met exactly.

Kneser-Ney Smoothing

- Unigram probabilities $P_{\mathrm{KN}}(w_i)$ ...
- *Not* proportional to how often unigram occurs.

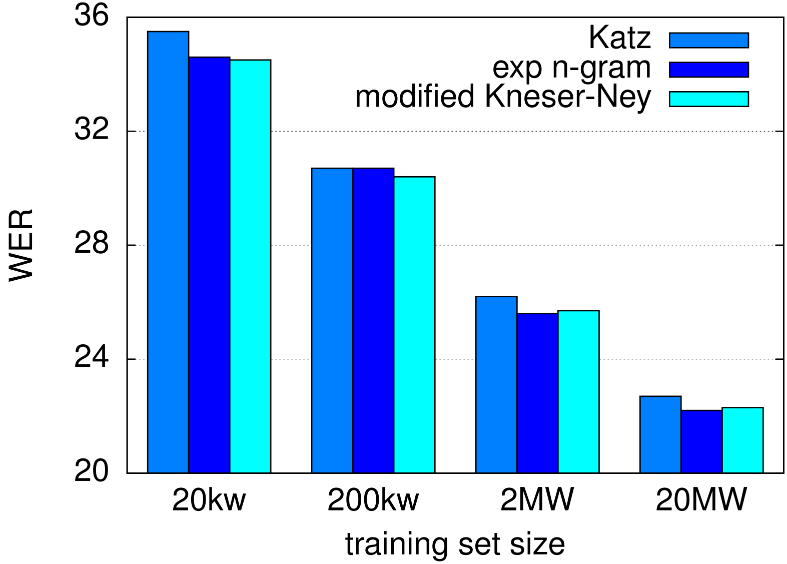
$$P_{\mathrm{KN}}(w_i) \neq \frac{c(w_i)}{\sum_{w_i} c(w_i)}$$

- Proportional to how many word types unigram follows!

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

$$P_{\mathrm{KN}}(w_i) = \frac{N_{1+}(\bullet w_i)}{\sum_{w_i} N_{1+}(\bullet w_i)}$$

Kneser-Ney Smoothing



Recap: *N*-Gram Models and Smoothing

- Best *n*-gram smoothing methods are all constraint-based.
- Can express smoothed *n*-gram models as . . .
    - Exponential models with simple ℓ_2^2 smoothing.
- "Modified" interpolated Kneser-Ney smoothing[†] . . .
    - Yields similar model, but much faster training.
    - Standard in literature for last 10+ years.
- Available in SRI LM toolkit.

  `http://www.speech.sri.com/projects/srilm/`

[†] (Chen and Goodman, 1998).

Where Are We?

What About Other Features?

- Exponential models make slightly better *n*-gram models.
  - Snore.
- Can we just toss in tons of cool features . . .
  - And get fabulous results?

Maybe!? (Rosenfeld, 1996)

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

What's the Catch?

- 200 computer-days to train.
- Really slow training.
  - For each word, update $O(|V|)$ counts.

$$\sum_{d=1}^{D} \sum_{w} P(w|h_d) f_i(h_d, w) = \sum_{d=1}^{D} f_i(h_d, w_d)$$

  - Tens of passes through training data.
- Really slow evaluation: evaluating $Z_\Lambda(x)$.

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

Newer Developments

- Fast training: optimizations for simple feature sets.
  - *e.g.*, train word *n*-gram model on 1GW in few hours.
- Fast evaluation: unnormalized models.
  - Not much slower than regular word *n*-gram.

$$P_\Lambda(w|h) = \exp(\sum_{i=1}^{F} \lambda_i f_i(h, w))$$

- Performance prediction.
  - How to intelligently select feature types.

Performance Prediction (Chen, 2008)

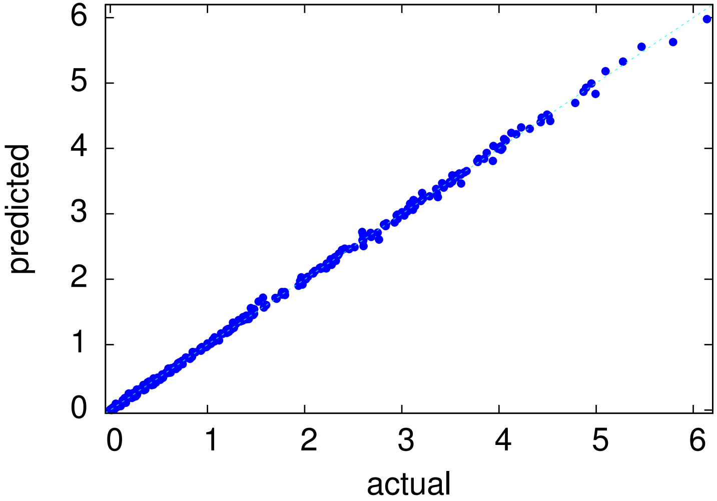
- Given training set and test set from same distribution.
- Desire: want to optimize performance on *test* set.
- Reality: only have access to *training* set.

$$\text{(test perf)} = \text{(training perf)} + \text{(overfitting penalty)}$$

- Can we estimate *overfitting penalty*?

Yes

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



A Tool for Good

- Holds for many different types of data.
  - Different domains; languages; token types; . . .
  - Vocab sizes; training set sizes; *n*-gram orders.
- 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.
- Can choose features types . . .
  - To *intentionally* shrink $\sum_{i=1}^{F} |\lambda_i|$.

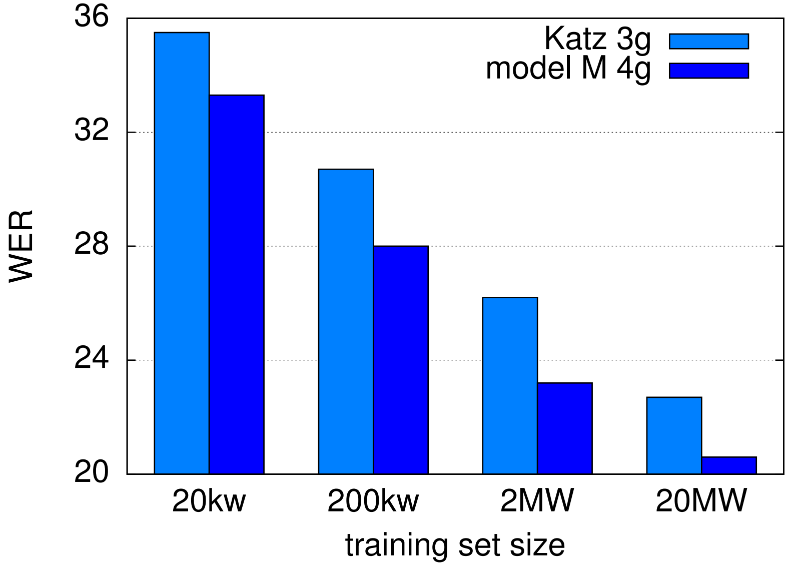
Model M (Chen, 2008; Chen and Chu, 2010)

- Old-timey class-based model (Brown, 1992).
  - Class prediction features: $c_{i-2}c_{i-1}c_i$.
  - Word prediction features: c_iw_i.

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

- Start from word *n*-gram model; convert to class model . . .
  - And choose feature types to reduce overfitting.
  - Class prediction features: $c_{i-2}c_{i-1}c_i$, $w_{i-2}w_{i-1}c_i$.
  - Word prediction features: $w_{i-2}w_{i-1}c_iw_i$.
- Without interpolation with word *n*-gram model.

Model M (WSJ)



36
32
28
24
20
WER
Katz 3g
model M 4g
20kw
200kw
2MW
20MW
training set size

Recap: Maximum Entropy

- Some of best WER results in LM literature.
    - Gain of up to 3% absolute WER over trigram (not <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 can still be very painful.
    - Depends very much on types of features.

Where Are We?

Introduction

- Ways to combine information sources.
  - Linear interpolation.
  - Exponential/log-linear models.
  - Anything else?
- Recently, good results with *neural networks*.

What Is A Neural Network?

- Represents function from input vector to output vector.

$$f : \mathcal{R}^{N_1} \Rightarrow \mathcal{R}^{N_L}$$

- Function is represented using sequence of L layers.
    - First layer is *input* layer; last is *output* layer.
    - Intermediate layers are *hidden* layers.
    - Each layer is vector $\mathbf{x}^l = (x_1^l, \ldots, x_{N_l}^l) \in \mathcal{R}^{N_l}$.
- Values in $(l+1)$th layer are function of values in lth layer.
    - $g(\cdot)$ is linear/non-linear increasing func, *e.g.*, sigmoid.

$$x_i^{l+1} = g(\sum_{j=1}^{N_l} w_{ij}^l x_j^l)$$

- Select parameters of model, *i.e.*, weights w_{ij}^l, ...
    - To optimize objective function of choice ...
    - Using gradient descent-ish algorithm like backprop.

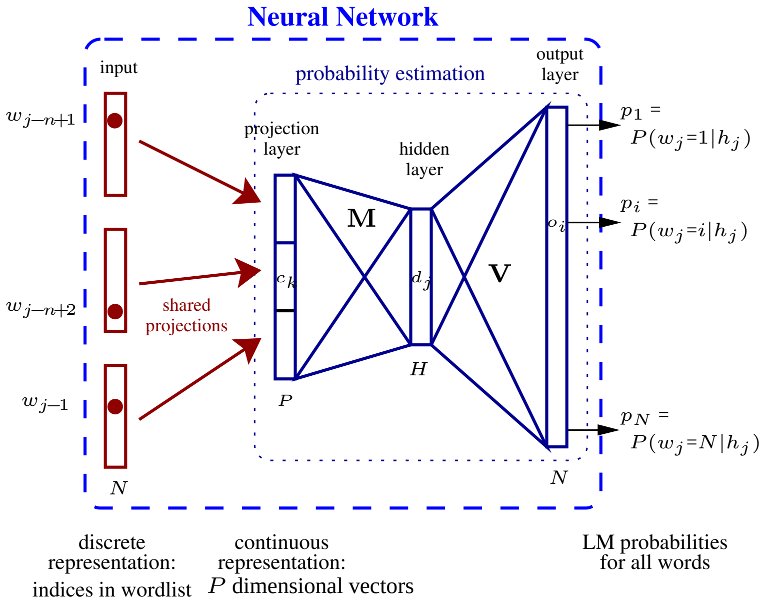
Designing a Neural Network LM

- What we have: something that learns $f : \mathcal{R}^{N_1} \Rightarrow \mathcal{R}^{N_L}$.
  - What we want: something that models $P(w_i | w_{i-2} w_{i-1})$.
  - How to map from words to continuous space?
- Binary coding.
  - To code one of V words, use vector of length V.
  - For vth word, 1 in position v, 0 everywhere else.
  - Input layer: $(n-1) \times V$ units; output layer: V units.
- Outputting a *probability*: *softmax* function.

$$p_i = \frac{e^{x_i^L}}{\sum_{j=1}^{N_L} e^{x_j^L}}$$

- Objective function: training set likelihood + regularization.

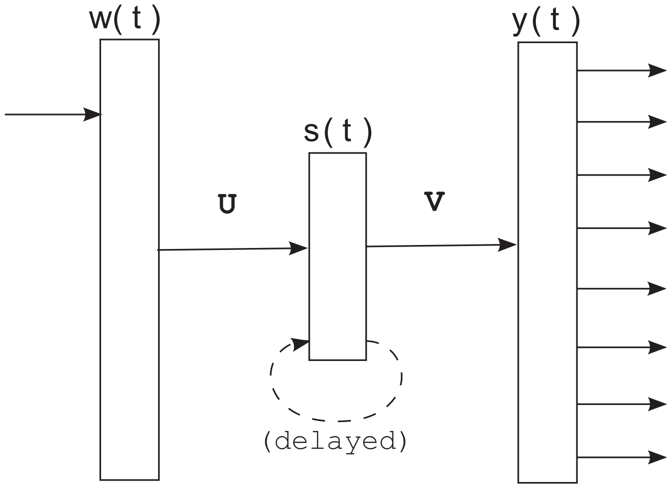
Example (Schwenk and Gauvain, 2005)



The Hidden Layers

- 1st hidden layer: projection layer.
  - Project each word in history from sparse Vd vector …
  - Down to $\sim$100d using shared linear projection.
- 2nd hidden layer: “real” hidden layer.
  - Nonlinear function of first layer, *e.g.*, tanh().

Recurrent NN LM's (Mikolov *et al.*, 2010)



Results (Mikolov *et al.*, 2011)

Model	Dev WER[%]	Eval WER[%]
Baseline - KN5	12.2	17.2
Discriminative LM [14]	11.5	16.9
Joint LM [7]	-	16.7
Static RNN	10.5	14.9
Static RNN + KN	10.2	14.6
Adapted RNN	9.8	14.5
Adapted RNN + KN	9.8	14.5
All RNN	**9.7**	**14.4**

Discussion

- Some of best WER results in LM literature.
  - Gain of up to 3% absolute WER over trigram (not <1%).
- Interpolation with word *n*-gram optional.
- Can integrate arbitrary features, *e.g.*, syntactic features.
  - Easy to condition on longer histories.
- Training is slow.
  - Optimizations: class-based modeling; reduced vocab.
- Evaluation is slow: matrix multiplies.
- Hard to analyze; tuning essential?
- Related to exponential models.
- Publicly-available toolkit:

  `http://www.fit.vutbr.cz/~imikolov/rnnlm/`

Where Are We?

Other Directions in Language Modeling

Discriminative training for 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 parametrizations 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.

What Is Used In Real Deployed Systems?

- Technology.
  - Mostly *n*-gram models; grammars.
  - Grammar switching based on dialogue state.
- Users cannot distinguish WER differences a few percent.
  - Good UI design is WAY, WAY, WAY more important . . .
  - Than small differences in ASR performance.
- Research developments in language modeling.
  - Not worth extra effort and complexity.
  - The more the data, the less the gain!
  - Difficult to implement in one-pass decoding paradigm.†

† Model M supported in IBM's Attila dynamic decoder.

Large-Vocabulary Research Systems

- *e.g.*, government evals: 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 extra effort and complexity.
- LM gains pale in comparison to acoustic modeling gains.

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.

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.
  - Faster computers? Data sets also growing.
- Need to effectively combine many sources of information.
  - Short, medium, and long distance.
  - Log-linear models, NN's promising, but slow to train.
- Evidence that LM's will help more when WER's are lower.
  - Human rescoring of *N*-best lists (Brill *et al.*, 1998).

References

- A.L. Berger, S.A. Della Pietra, V.J. Della Pietra, "A maximum entropy approach to natural language processing", Computational Linguistics, vol. 22, no. 1, pp. 39–71, 1996.
- E. Brill, R. Florian, J.C. Henderson, L. Mangu, "Beyond N-Grams: Can Linguistic Sophistication Improve Language Modeling?", ACL, pp. 186–190, 1998.
- S.F. Chen, "Performance Prediction for Exponential Language Models", IBM Research Division technical report RC 24671, 2008.
- S.F. Chen and S.M. Chu, "Enhanced Word Classing for Model M", Interspeech, 2010.
- S.F. Chen and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling", Harvard University technical report TR-10-98, 1998.

References

- S.F. Chen and R. Rosenfeld, “A Survey of Smoothing Techniques for Maximum Entropy Models”, IEEE Transactions on Speech and Audio Processing, vol. 8, no. 1, pp. 37–50, 2000.
- E.T. Jaynes, “Information Theory and Statistical Mechanics”, Physics Reviews, vol. 106, pp. 620–630, 1957.
- R. Kneser and H. Ney, “Improved Backing-off for M-Gram Language Modeling”, ICASSP, vol. 1, pp. 181–184, 1995.
- H. Schwenk and J.L. Gauvain, “Training Neural Network Language Models on Very Large Corpora”, HLT/EMNLP, pp. 201–208, 2005.
- T. Mikolov, “Statistical Language Models based on Neural Networks”, Ph.D. thesis, Brno University of Technology, 2012.