Foundations of Statistical Natural Language Processing:
A Case Study of Text Input System

Jianfeng Gao, Hisami Suzuki
Microsoft Research

Weihai, 8/23/2007
Who should be here?

- Interested in statistical Natural Language Processing
  - What is NLP? NLP = AI? What is the role of Pr in NLP?
- Want to develop a simple and useful NLP system by yourself
  - For fun, course project, mind exercise?
- Look for topics for your master/PhD thesis
  - A difficult topic: very hard to beat simple baseline
  - An easy topic: others cannot beat it either
- Start NLP/IME business and compete with MS
Outline

- Probability: a brief refresher
- Input Method Editor (IME): problems and solutions
- Modeling: capture language structure
- Training: learn model parameters from data
- Search: predict using model (won’t discuss in detail)
- Do It Yourself (DIY) tips
Probability: a brief refresher (1/2)

- Probability space: \( x \in X \)
  - \( P(x) \in [0, 1] \)
  - \( \sum_{x \in X} P(x) = 1 \)
  - Cannot say \( P(x) > P(y) \) if \( y \notin X \)

- Joint probability: \( P(x, y) \)
  - Probability that \( x \) and \( y \) are both true

- Conditional probability: \( P(y|x) \)
  - Probability that \( y \) is true when we already know \( x \) is true

- Independence: \( P(x, y) = P(x)P(y) \)
  - \( x \) and \( y \) are independent
Probability: a brief refresher (2/2)

- \( H \): assumptions on which the probabilities are based

- **Product rule** – from the definition of conditional probability
  \[
P(x, y | H) = P(x | y, H)P(y | H) = P(y | x, H)P(x | H)
\]

- **Sum rule** – a rewrite of the marginal probability definition
  \[
P(x | H) = \sum_y P(x, y | H) = \sum_y P(x | y, H)P(y | H)
\]

- **Bayes rule** – from the product rule
  \[
P(y | x, H) = P(x | y, H)P(y | H) / P(x | H)
\]
Input method editor (IME)

- Software to convert keystrokes (Pinyin) to text output

Gao and Suzuki, Weihan-2007
A Bayesian approach to IME

- Find the best output $W$ of a given input $A$ via

$$W = \arg\max_w P(W|A)$$

$$W = \arg\max_w \frac{P(A|W)P(W)}{P(A)}$$

$$W = \arg\max_w P(A|W)P(W)$$

- $P(A|W)$: typing (translation) model
  - Dealing with typing error, e.g., $zh \rightarrow z$

- $P(W)$: language model (LM), e.g., trigram model
Three fundamental research tasks

- **Modeling**: capture language structure/dependencies via the probabilistic model
  - \( Pr(W|A) = P_\theta(W|A) = P(W|A, \theta) \)
- **Training**: estimation of free parameters using training data
  - \( \theta = \arg\max_\theta P(W|A, \theta) \)
- **Search**: finding “best” conversion given the model
  - \( W = \arg\max_W P(W|A, \theta) \)
- **Additional important tasks**
  - Data/dict acquisition and processing (word segmentation)
  - Evaluation methodology
Development of IME: data

- Dictionary – mapping from Pinyin to Chinese words
- Training data, \((W)\) and \((W, A)\)
  - Chinese text – LM training
    - Obtained from Chinese web pages
  - Pinyin and Chinese text pairs – discriminative training
    - Check our website
- Data processing
  - Word segmentation
  - Training/dev/test split (cross-validation)
  - Gold standard
Development of IME: evaluation

- Perplexity – quality of LM
  - Geometric average inverse probability
  - Branching factor of a doc: predicting power of LM
  - Lower perplexities are better
  - Character perplexity for Chinese
    \[ pplx = 2^H \text{ where } H = \frac{1}{|W|} \log P(W) \]

- Character error rate (CER) – quality of IME
  - Test set \((A, W^*)\)
  - CER = edit distance between converted \(W\) and \(W^*\)
  - Correlation with perplexity
Development of IME: build it bit by bit

• Baseline
  • Straw-man versus state-of-the-art
  • IME: Trigram LM, MLE, Viterbi search
• Improve the baseline via
  • Better training data: dictionary (OOV), segmentation, balanced corpus etc.
  • Better modeling: capture richer linguistic information?
  • Better training: lead to better CER/perplexity?
  • Better search (decoding): less search error and faster

Gao and Suzuki, Weihan-2007
Modeling

- Goal: how to incorporate language structure into a probabilistic model
- Task: next word prediction
  - Fill in the blank: “The dog of our neighbor ___”
- Starting point: word n-gram model
  - Very simple, yet surprisingly effective
  - Words are generated from left-to-right
  - Assumes no other structure than words themselves
Word N-gram model

- Word based model
  - Using chain rule on its *history* (=preceding words)

\[
P(\text{the dog of our neighbor barks}) = P(\text{the} \mid <s>) \\
\times P(\text{dog} \mid <s>, \text{the}) \\
\times P(\text{of} \mid <s>, \text{the, dog}) \\
\vdots \\
\times P(\text{barks} \mid <s>, \text{the, dog, of, our, neighbor}) \\
\times P(</s> \mid <s>, \text{the, dog, or, our, neighbor, barks})
\]

\[
P(w_1, w_2 \ldots w_n) = P(w_1 \mid <s>) \\
\times P(w_2 \mid <s> w_1) \\
\times P(w_3 \mid <s> w_1 w_2) \\
\vdots \\
\times P(w_n \mid <s> w_1 w_2 \ldots w_{n-1}) \\
\times P(</s> \mid <s> w_1 w_2 \ldots w_n)
\]
Word n-gram model

- How do we get probability estimates?
  - Get text and count! \( P(\text{the}|<s>) \approx C(<s> \text{ the})/C(<s>) \)
- Problem of using the whole history
  - Rare events: unreliable probability estimates
  - Assuming a vocabulary of 20,000 words,

<table>
<thead>
<tr>
<th>Model</th>
<th># Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram</td>
<td>( P(w_1) )</td>
</tr>
<tr>
<td>bigram</td>
<td>( P(w_2</td>
</tr>
<tr>
<td>trigram</td>
<td>( P(w_3</td>
</tr>
<tr>
<td>fourgram</td>
<td>( P(w_4</td>
</tr>
</tbody>
</table>

From Manning and Schütze 1999: 194
Word N-gram model

- Markov independence assumption
  - A word depends only on $N-1$ preceding words
  - $N=3 \rightarrow$ word trigram model
- Reduce the number of *parameters* in the model
  - By forming *equivalence classes*
- Word trigram model
  
  $P(w_i \mid <s> w_i, w_2 \ldots w_{i-2} w_{i-1}) = P(w_i \mid w_{i-2} w_{i-1})$

  $P(w_1, w_2 \ldots w_n) = P(w_1 \mid <s>)$

  $\times P(w_2 \mid <s> w_1)$

  $\times P(w_3 \mid w_1 w_2)$

  $\ldots$

  $\times P(w_n \mid w_{n-2} w_{n-1})$

  $\times P(<s> \mid w_{n-1} w_n)$
But language has structure!

- Other ways to form equivalence classes
  - Morphological
    - Stemming: bark~barked~barks~barking
  - Syntactic

Gao and Suzuki, Weihan-2007
But language has structure!

- Other ways to form equivalence classes
  - Semantic
    - Cluster semantically related words: dog~husky~poodle

- Challenge
  - How to incorporate linguistic structure in a probabilistic model effectively
Modeling: basic idea

- Introduce language structure $s$ as hidden variable
  - Assignment of $s$ must be predicted given $h$
    \[
P(w \mid h) = \sum_s P(w, s \mid h) = \sum_s P(s \mid h)P(w \mid s, h)
    = \sum_s P(s \mid h)P(w \mid \Phi(s, h))
    \]

- Define mapping function $\Phi$
  - $\Phi$ maps word history into equivalence classes
    \[
P(w_i \mid w_1...w_{i-1}) = P(w \mid h) = P(w \mid \Phi(h))
    \]
    Word trigram if $\Phi(h) = (w_{i-2}w_{i-1})$
Finding all possible assignment of $s$

- Detect $s$ via parsing: an independent NLP problem
  - POS tagging, dependency graph, word clusters...
  - Traditional NLP tasks: tools available
  - Finding all possible assignment of $s$ is often not realistic

- N-best and Viterbi approximation

$$P(w \mid h) = \sum_s P(s \mid h)P(w \mid \Phi(s, h))$$

$$\approx \sum_s \frac{P(s \mid h)}{\sum_s P(s \mid h)} P(w \mid \Phi(s, h)) \quad \leftarrow \text{N-best approximation}$$

$$\approx \max_s P(w \mid \Phi(s, h)), \text{ where } s = \arg \max_s P(s \mid h) \quad \leftarrow \text{Viterbi approximation}$$
Defining $\Phi$

- $s$ is a chunk sequence
  - $\Phi(s) \rightarrow$ two previous headword
  - Headword trigram model (Gao et al., 2002b)
- $s$ is a dependency graph
  - $\Phi(s) \rightarrow$ linked word to its left
  - Dependency LM (Gao and Suzuki, 2003)
- $s$ is a word cluster sequence
  - $\Phi(s) \rightarrow$ two previous word clusters
  - Cluster LM (Gao et al., 2002c)
Headword trigram model (HTM)

- $s$ is a chunk sequence
- Chunk (Abney, 1991)
  - Base phrase, typically contains one content word (headword) plus any number of function words.
  - Flat, non-hierarchical and span the word sequence
  - Closely related to the notion of bunsetsu in Japanese
  - Define $\Phi(s)$ as two previous headwords
- Example
  - [$The \text{ dog}$] [of our $\text{neighbor}$] [$barks$] [$every \text{ night}$]
Headword trigram model (HTM)

- $s$ is a chunk sequence
- Chunk (Abney, 1991)
  - Base phrase, typically contains one content word (headword) plus any number of function words.
  - Flat, non-hierarchical and span the word sequence
  - Closely related to the notion of bunsetsu in Japanese
  - Define $\Phi(s)$ as two previous headwords
- Example
  - $[\text{The } \underline{dog} \text{ of our } \underline{neighbor}] [\underline{barks}] [\text{every } \underline{night}]$
Headword trigram model (HTM)

- Using headword $H$ and function word $F$
  - 2-step model: generate class first, then generate words given the class (chain rule)
    $$P(w_i \mid \Phi(w_1...w_{i-1})) = P(H_i \mid \Phi(w_1...w_{i-1})) \times \left[P(w_i \mid \Phi(w_1...w_{i-1})H_i) + P(F_i \mid \Phi(w_1...w_{i-1})) \times P(w_i \mid \Phi(w_1...w_{i-1})F_i)\right]$$

- Incorporating assumptions using headword
  - Dependency between headwords ($dog \sim barks$)
  - Headword dependency is permutable ($barks \sim dogs$)

$$P(w_i \mid \Phi(w_1...w_{i-1})H_i) = \lambda_1 \left[\lambda_2 P(w_i \mid h_{i-2}h_{i-1}H_i) + (1 - \lambda_2)P(w_i \mid h_{i-1}h_{i-2}H_i))\right] + (1 - \lambda_1)P(w_i \mid w_{i-2}w_{i-1}H_i)$$
Detecting Headwords

• Assumed a one-to-one mapping between POS and word category (H/F)
• Generated a mapping table from POS-tagged text
  • Chose the more frequent category in case of ambiguity
• Accuracy of H/F detection: 98.5%
  • This is good enough
Dependency language model (DLM)

- $s$ is a dependency graph among headwords
- Constraint on dependency structure $D$
  - Planar: no line crossing
  - Acyclic: contains no cycle
  - Define $\Phi(s)$ as the linked word on the left
- Example

  - [The dog] [of our neighbor] [barks] [every night]
Dependency language model (DLM)

- $s$ is a dependency graph among headwords
- Constraint on dependency structure $D$
  - Planar: no line crossing
  - Acyclic: contains no cycle
  - Define $\Phi(s)$ as the linked word on the left
- Example
  - $[\text{The dog}]$ $[\text{of our neighbor}]$ $[\text{barks}]$ $[\text{every night}]$
  - $w_i$ $w_j$
Dependency language model (DLM)

- $s$ is a dependency graph among headwords.
- Constraint on dependency structure $D$:
  - Planar: no line crossing
  - Acyclic: contains no cycle
  - Define $\Phi(s)$ as the linked word on the left
- Example

- Advantage
  - Capture long-distance dependency
The most probably dependency $D$ is generated by

$$D^* = \arg \max_{D} P(D|W) = \arg \max_{D} \prod_{d \in D} P(d|W)$$

Parsing algorithm (approximation algorithm)

- Operates L to R
- Link $w_j$ to each of its previous words $w_i$, and push the generated dependency $d_{ij}$ into a stack
- Violation of syntactic constraints (planar and acyclic): resolved by removing the dependency with the lowest probability in conflict
- Efficient: $O(n^2)$
  - Traditional parser is $O(n^5)$
  - Modified version of Yuret (1998)
Dependency language model (DLM)

\[ P(w_j | \Phi(W_{j-1}, D_{j-1})) = \]

\[ \lambda_1(P(w_j | w_i, R)) \]

\[ + (1 - \lambda_1)P(w_j | w_{j-2}, w_{j-1}) \]

\[ P(w_j | w_{j-2}, w_{j-1}) \]

\( w_j \): headword

\( w_j \): function word

[The **dog**] [of our **neighbor**] [**barks**] [every **night**]

Gao and Suzuki, Weihan-2007
Cluster language model (CLM)

- $s$ is a set of word clusters
- Goal: group similar words
  - Syntactic similarity: POS
  - Semantic similarity
    - WEEKDAY \{Monday, Tuesday, Wednesday...\}
    - DOG \{poodle, husky, lab, dog ... \}
- Define $\Phi(s)$ as two previous word clusters
- Example
  - *The poodle barks every night*
    - Estimate of $P(barks \mid poodle)$ may be inaccurate
    - Estimate of $P(barks \mid DOG)$ may be more reliable
CLM: forms

- Predicted and conditional words in $P(w_3 | w_1 w_2)$
  - $w_3$: predicted word
  - $w_1$ and $w_2$: conditional words

- Three basic cluster trigram models
  - Predictive cluster model
    $$P(w_i | w_{i-2} w_{i-1}) \approx P(W_i | w_{i-2} w_{i-1}) \times P(w_i | w_{i-2} w_{i-1} W_i)$$
  - Conditional cluster model
    $$P(w_i | w_{i-2} w_{i-1}) \approx P(w_i | W_{i-2} W_{i-1})$$
  - Combined cluster model
    $$P(w_i | w_{i-2} w_{i-1}) \approx P(W_i | W_{i-2} W_{i-1}) \times P(w_i | W_{i-2} W_{i-1} W_i)$$
Finding word clusters (Goodman, 2001)

- Objective function: maximize probability
  - In the case of predictive clustering, maximize
    \[
    \prod_{i=1}^{N} P(W_i \mid w_{i-1}) \times P(w_i \mid W_i)
    \]
    
    \[
    = \prod_{i=1}^{N} \frac{P(w_{i-1}W_i)}{P(w_{i-1})} \times \frac{P(W_iw_i)}{P(W_i)}
    \]
    
    \[
    = \prod_{i=1}^{N} \frac{P(W_iw_i)}{P(w_{i-1})} \times \frac{P(w_{i-1}W_i)}{P(W_i)}
    \]
    
    \[
    = \prod_{i=1}^{N} \frac{P(w_i)}{P(w_{i-1})} \times P(w_{i-1} \mid W_i)
    \]
    
    - Sufficient to maximize \( \prod_{i=1}^{N} P(w_{i-1} \mid W_i) \)
Data for Evaluation

- **Task**: Japanese IME
  - Baseline: word trigram model
  - N-best re-scoring task (N=100)
- **Corpus**: Newspaper (word-segmented)
  - Training: Nikkei (36 million words)
  - Test: Yomiuri (100,000 words)
- **Metric**: Character Error Rate (CER)
  \[
  \text{CER} = \frac{\#\text{chars wrongly converted}}{\#\text{chars in the target sentence}}
  \]
## Results on Japanese IME (Gao and Suzuki, 2004)

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>CER %</th>
<th>CER Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Word trigram model</td>
<td>3.73</td>
<td>——</td>
</tr>
<tr>
<td>Oracle</td>
<td>In the 100-best list with the minimum number of errors</td>
<td>1.51</td>
<td>59.5%</td>
</tr>
</tbody>
</table>
Modeling: summary

- Motivation
  - Incorporate linguistic structure in a probabilistic model
  - Word trigram model cannot capture long-distance dependency
- Three types of structures
  - Chunks, dependency, clusters
  - Substantial improvement over trigram model
- Challenge
  - Model simplicity vs. capturing structure
  - Modeling vs. training data size
Training: parameter estimation

- Bayesian estimation paradigm
- Maximum likelihood estimation (MLE)
- Smoothing in N-gram language models
- Discriminative training (overview)
The Bayesian paradigm

- \( P(\text{model}|\text{data}) = P(\text{data}|\text{model}) \times P(\text{model}) / P(\text{data}) \)
  - \( P(\text{model}|\text{data}) \) – Posterior
  - \( P(\text{data}|\text{model}) \) – Likelihood
  - \( P(\text{model}) \) – Prior
  - \( P(\text{data}) \) – Marginal

- Likelihood versus probability
  - \( P(n \mid u, N) \), for fixed \( u \), \( P \) defines a probability over \( n \); for fixed \( n \), \( P \) defines the likelihood of \( u \).

- Never say “the likelihood of the data”
- Always say “the likelihood of the parameters given the data”
Maximum likelihood estimation

- $\theta$: model; $X$: data
- $\theta = \arg\max P(\theta|X) = \arg\max P(X|\theta)P(\theta)/P(X)$
  - Assume a uniform prior $P(\theta) = \text{Const}$
  - $P(X)$ is independent of $\theta$, and is dropped
- $\theta = \arg\max P(\theta|X) \approx \arg\max P(X|\theta)$
  - Where $P(X|\theta)$ is the likelihood of parameter

Key difference between MLE and Bayesian Estimation
- MLE assume that $\theta$ is fixed but unknown,
- Bayesian estimation assumes that $\theta$ itself is a random variable with a prior distribution $P(\theta)$. 
MLE for trigram LM

- \( P_{ML}(w_3|w_1w_2) = \frac{\text{Count}(w_1w_2w_3)}{\text{Count}(w_1w_2)} \)
- \( P_{ML}(w_2|w_1) = \frac{\text{Count}(w_1w_2)}{\text{Count}(w_1)} \)
- \( P_{ML}(w) = \frac{\text{Count}(w)}{N} \)
- It is easy – let us get real Chinese text and start counting

\[
P_{ML}(\text{barked}|\text{the}, \text{dog}) = \frac{\text{Count}(\text{the}, \text{dog}, \text{barked})}{\text{Count}(\text{the}, \text{dog})}
\]

- But why this is the MLE solution?
The derivation of MLE for N-gram

• Homework – an interview question of MSR 😊
• Hints
  • This is a constrained optimization problem
  • Use log likelihood as objective function
  • Assume a multinomial distribution of LM
  • Introduce Lagrange multiplier for the constraints
  • $\sum_{x \in X} P(x) = 1$, and $P(x) \geq 0$
Sparse data problems

- Say our vocabulary size is $|V|$
- There are $|V|^3$ parameters in the trigram LM
  - $|V| = 20,000 \Rightarrow 20,000^3 = 8 \times 10^{12}$ parameters
- Most trigrams have a zero count even in a large text corpus
  - $\text{Count}(w_1 w_2 w_3) = 0$
  - $P_{ML}(w_3|w_1 w_2) = \text{Count}(w_1 w_2 w_3)/\text{Count}(w_1 w_2) = 0$
  - $P(W) = P_{ML}(w_1) P_{ML}(w_2|w_1) \prod_i P_{ML}(w_i|w_{i-2} w_{i-1}) = 0$
  - $W = \arg\max_W P(A|W)P(W) = \ldots$ oops
Smoothing: adding one

- Add one smoothing (from Bayesian paradigm)
- But works very badly – do not use this

\[ P(w_3|w_2, w_1) = \frac{\text{Count}(w_1, w_2, w_3) + 1}{\text{Count}(w_1, w_2) + |V|} \]

- Add delta smoothing
- Still very bad – do not use this

\[ P(w_3|w_2, w_1) = \frac{\text{Count}(w_1, w_2, w_3) + \delta}{\text{Count}(w_1, w_2) + |V|\delta} \]
Smoothing: linear interpolation

- Linearly interpolate trigram, bigram and unigram prob

\[ P(w_3|w_1, w_2) = \lambda_1 P_{ML}(w_3|w_1, w_2) + \lambda_2 P_{ML}(w_3|w_2) + \lambda_3 P_{ML}(w_3) \]

where \( \lambda_1 + \lambda_2 + \lambda_3 = 1 \)

- Allow \( \lambda \)'s to vary – value of \( \lambda \) is a function of Count(.)

\[ P(w_3|w_1, w_2) = \lambda_1 (C(w_1, w_2, w_3)) P_{ML}(w_3|w_1, w_2) \]
\[ + \lambda_2 (C(w_2, w_3)) P_{ML}(w_3|w_2) \]
\[ + \lambda_3 (C(w_3)) P_{ML}(w_3) \]

where \( \lambda_1 (C(w_1, w_2, w_3)) + \lambda_2 (C(w_2, w_3)) + \lambda_3 (C(w_3)) = 1 \)
How to estimate $\lambda$’s

- Split data into training, dev, test
- Optimize $\lambda$’s on dev data (i.e., pick the best value of $\lambda$)

$$
\lambda = \arg\max_{\lambda} \sum_{(w_1, w_2, w_3) \text{in dev data}} \log P(w_3 | w_1 w_2)
$$

- Can use EM (expectation maximization) algorithm to find the $\lambda$’s
- Or use a generalized numerical optimization algorithm (e.g., Powell search)
  - The objective function is concave
Smoothing: backoff

- Backoff trigram to bigram, bigram to unigram

\[
P(w_3|w_1, w_2) = \begin{cases} 
\frac{C(w_1, w_2, w_3) - D}{C(w_1, w_2)}, & \text{if } C(w_1, w_2, w_3) > 0 \\
\alpha(w_1, w_2)P(w_3|w_2), & \text{if } C(w_1, w_2, w_3) = 0
\end{cases}
\]

- \(D \in (0,1)\) is a discount constant – absolute discount
- \(\alpha\) is calculated so probabilities sum to 1 (homework 🙂)

\[
1 = \sum_{(w_1, w_2)} P(w_3|w_1, w_2)
\]
Smoothing: improved backoff

- Allow $D$ to vary
  - Different $D$’s for different N-gram
  - Value of $D$’s as a function of Count(.)
  - Modified absolute discount
- Optimizing $D$’s on dev data using e.g., Powell search

$$D = \arg\max_D \sum_{(w_1,w_2,w_3) \text{ in dev data}} \log P(w_3|w_1w_2)$$

- Using word type probabilities rather than token probability for backoff models
  - Kneser-Ney smoothing
What is the best smoothing?

- It varies from task to task
  - Chen and Goodman (1999) gives a very thorough evaluation and descriptions of a number of methods

- My favorite smoothing methods
  - Modified absolute discount (Gao et al., 2001)
    - Simple to implement and use
    - Good performance across many tasks, e.g., IME, SMT, ASR
  - Interpolated Kneser-Ney
    - Recommended by Chen and Goodman (1999)
    - Best performance on our SMT system (trickier to use, though)
Google’s stupid smoothing 😊

- Simply set $D=0$, and $\lambda = 0.4$
- Refer to (Brant et al., 2007)

$$P(w_3|w_1, w_2) = \begin{cases} \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)}, & \text{if } C(w_1, w_2, w_3) > 0 \\ 0.4P(w_3|w_2), & \text{if } C(w_1, w_2, w_3) = 0 \end{cases}$$

![Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).](image)

- Do not do research until you run out of data (Eric Brill)
Discriminative training

- MLE – maximizing $P(X|\theta)$
- Discriminative training – maximizing $P(\theta|X)$

\[
P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} = \frac{P(X|\theta)P(\theta)}{\sum_{\theta'} P(X|\theta')P(\theta')} \quad \text{assume a uniform prior } P(\theta) = C
\]

\[
\text{argmax } P(\theta|X) = \text{argmax} \frac{P(X|\theta)}{P(X|\theta) + \sum_{\theta' \neq \theta} P(X|\theta')}
\]

\[
= \text{argmax} \frac{1}{1 + \frac{\sum_{\theta' \neq \theta} P(X|\theta')}}{P(X|\theta)}
\]

\[
= \text{argmax} \frac{P(X|\theta)}{\sum_{\theta' \neq \theta} P(X|\theta')}
\]

- E.g., Maximum Entropy (Rosenfeld, 1994), Perceptron (Roark et al., 2004)
Search: basic algorithms

- Search space: lattice
- Find 1-best conversion
  - Time-synchronous Viterbi decoder (left to right)
  - Efficiency – the use of beam
- Find N-best conversions
  - Time-asynchronous A* decoder (best-first search + heuristic function)
  - How to estimate future cost (heuristic function)
- 2-pass search
  - First pass: left-to-right search find the 1-best
  - Second pass: A* search using 1-best scores as future cost
- A good text book – (Huang et al., 2001)
Search: an example (homework 😞)

<table>
<thead>
<tr>
<th>Rank</th>
<th>W</th>
<th>-logP(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;s&gt;, A, D, &lt;/s&gt;</td>
<td>2.1</td>
</tr>
<tr>
<td>2</td>
<td>&lt;s&gt;, A, E, &lt;/s&gt;</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>&lt;s&gt;, B, D, &lt;/s&gt;</td>
<td>2.6</td>
</tr>
<tr>
<td>4</td>
<td>&lt;s&gt;, C, D, &lt;/s&gt;</td>
<td>2.7</td>
</tr>
<tr>
<td>5</td>
<td>&lt;s&gt;, B, E, &lt;/s&gt;</td>
<td>2.8</td>
</tr>
<tr>
<td>6</td>
<td>&lt;s&gt;, C, F, &lt;/s&gt;</td>
<td>2.8</td>
</tr>
<tr>
<td>7</td>
<td>&lt;s&gt;, C, E, &lt;/s&gt;</td>
<td>3.0</td>
</tr>
<tr>
<td>8</td>
<td>&lt;s&gt;, B, F, &lt;/s&gt;</td>
<td>3.1</td>
</tr>
<tr>
<td>9</td>
<td>&lt;s&gt;, A, F, &lt;/s&gt;</td>
<td>3.7</td>
</tr>
</tbody>
</table>

P(A) = 0.2, P(B) = 0.15, P(C) = 0.1, P(D|A) = 0.2, P(E|A) = 0.15, P(F|A) = 0.01, P(D|C) = 0.1, P(E|C) = 0.1, P(F|C) = 0.15, P(D|B) = 0.08, P(E|B) = 0.1, P(F|B) = 0.05
DIY: tools and data

- LM Toolkit
  - CMU SLM (probably out-of-date, still usable)
  - SRILM (most popular, implementation of KN smoothing)
  - MSR SLM (forthcoming, check our website)
- Training data
  - Crawl Chinese web pages
  - Discriminative training data, check our website
- Word segmentation
  - LDC word breaker
  - MSRSeg, check our website
- Visual Studio 2005
DIY: get your hands dirty

- Data preparation
  - Dictionary, pinyin-to-word mapping?
  - Training data acquisition and processing
- Baseline IME system
  - Train a trigram model using existing SLM toolkit
  - Code a Viterbi decoder
    - Access dictionary to generate lattice (define search space)
    - Access trigram probability to find the best word string given input:
      \[ W = \text{argmax } P(W|A) \approx \text{argmax } P(W) \]
- Evaluation
  - Quality of LM: perplexity
  - Quality of IME: CER
DIY: your research topics

- Better modeling
  - Latent semantic LM (Bellegarda, 2004)
  - Structured language model (Chelba and Jelinek, 2000)
- Better training
  - A Bayesian approach (Teh, 2006)
  - Discriminative training (Gao et al., 2007)
- Best IME system
  - Keep it as simple as possible
  - Excellent Engineering
  - Data, data, data!
What we did at MSR

- Better training data: 1999-2001
  - unified approach to Chinese SLM
  - Gao et al., (2002a)
- Better model form: 2002-2004
  - introduce language structure into SLM
- Better training method: 2005-present
  - directly minimize error rate

YOU CAN DO BETTER THAN US!
Better training data: Chinese IME results
(Gao et al., 2002a)

<table>
<thead>
<tr>
<th>Training Set</th>
<th>IME</th>
<th>Total</th>
<th>Total</th>
<th>Total</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexicon &amp; Segmentation Optimization</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Training Set Filtering</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Training Set Domain Adaptation</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Pruning Method</td>
<td>Count Cutoff</td>
<td>Predict Cluster + Stolcke</td>
<td>Predict Cluster + Stolcke</td>
<td>Stolcke</td>
<td>Stolcke</td>
</tr>
</tbody>
</table>

Table 10: Summary of techniques used in system evaluation
Better training data: Chinese IME results
(Gao et al., 2002a)

![Graph showing character error rate vs. memory (MB) for different models.]

- **Baseline**
- MSR-Bigram1
- MSR-Bigram2
- MSR-Trigram1
- MSR-Trigram2
Better modeling: Japanese IME results (Gao and Suzuki, 2004)

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>CER %</th>
<th>CER Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Word trigram model</td>
<td>3.73</td>
<td>——</td>
</tr>
<tr>
<td>Oracle</td>
<td>In the 100-best list with the minimum number of errors</td>
<td>1.51</td>
<td>59.5%</td>
</tr>
<tr>
<td>HTM</td>
<td>Equation (3) with ( \lambda_1=0.2 ) and ( \lambda_2=1 )</td>
<td>3.41</td>
<td>8.6%</td>
</tr>
<tr>
<td>PHTM</td>
<td>Equation (3) with ( \lambda_1=0.2 ) and ( \lambda_2=0.7 )</td>
<td>3.34</td>
<td>10.5%</td>
</tr>
<tr>
<td>C-PHTM</td>
<td>Equation (3) with ( \lambda_1=0.3 ) and ( \lambda_2=0.7 )</td>
<td>3.17</td>
<td>15.0%</td>
</tr>
<tr>
<td>4-gram</td>
<td>Higher-order ( n )-gram model with a modified version of</td>
<td>3.71</td>
<td>0.5%</td>
</tr>
<tr>
<td>5-gram</td>
<td>Kneser-Ney interpolation smoothing</td>
<td>3.71</td>
<td>0.5%</td>
</tr>
<tr>
<td>6-gram</td>
<td></td>
<td>3.73</td>
<td>0.1%</td>
</tr>
<tr>
<td>ATR-I</td>
<td>Equation (6)</td>
<td>4.75</td>
<td>-27.3%</td>
</tr>
<tr>
<td>ATR-I+</td>
<td>ATR-I interpolated with Baseline</td>
<td>3.67</td>
<td>1.6%</td>
</tr>
<tr>
<td>ATR-II</td>
<td>Equation (7)</td>
<td>3.65</td>
<td>2.1%</td>
</tr>
<tr>
<td>DLM-1</td>
<td>Equation (8) with ( \lambda_1=0.1 ) and ( \lambda_2=0 )</td>
<td>3.49</td>
<td>6.4%</td>
</tr>
<tr>
<td>DLM-2</td>
<td>Equation (8) with ( \lambda_1=0.3 ) and ( \lambda_2=0.7 )</td>
<td>3.33</td>
<td>10.7%</td>
</tr>
</tbody>
</table>
## Better training: Japanese IME results (Gao et al., 2007)

<table>
<thead>
<tr>
<th>Method</th>
<th>CER</th>
<th># features</th>
<th>time (min)</th>
<th># train iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (MAP)</td>
<td>7.98%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxEnt/L2</td>
<td>6.99%</td>
<td>295,337</td>
<td>27</td>
<td>665</td>
</tr>
<tr>
<td>MaxEnt/L1</td>
<td>7.01%</td>
<td>53,342</td>
<td>25</td>
<td>864</td>
</tr>
<tr>
<td>AvePerceptron</td>
<td>7.23%</td>
<td>167,591</td>
<td>6</td>
<td>56</td>
</tr>
<tr>
<td>Boosting</td>
<td>7.54%</td>
<td>32,994</td>
<td>175</td>
<td>71,000</td>
</tr>
<tr>
<td>BLasso</td>
<td>7.20%</td>
<td>33,126</td>
<td>238</td>
<td>250,000</td>
</tr>
</tbody>
</table>
References

- Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, Jeffrey Dean. 2007. Large language models in machine translation. In EMNLP.
- Gao, J. J. Goodman, G. Cao, H. Li. 2002c. Exploring asymmetric clustering for statistical language modeling. ACL.
- Roark, Brian, Murat Saraclar and Michael Collins. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In ICASSP.
- Teh, Yee Whye. 2006. A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes. In ACL.
Contact information

- Jianfeng Gao,
  http://research.microsoft.com/~jfgao/

- Hisami Suzuki,
  http://research.microsoft.com/~hisamis/

- The latest version of the slides and papers/tools can be found on our website.