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

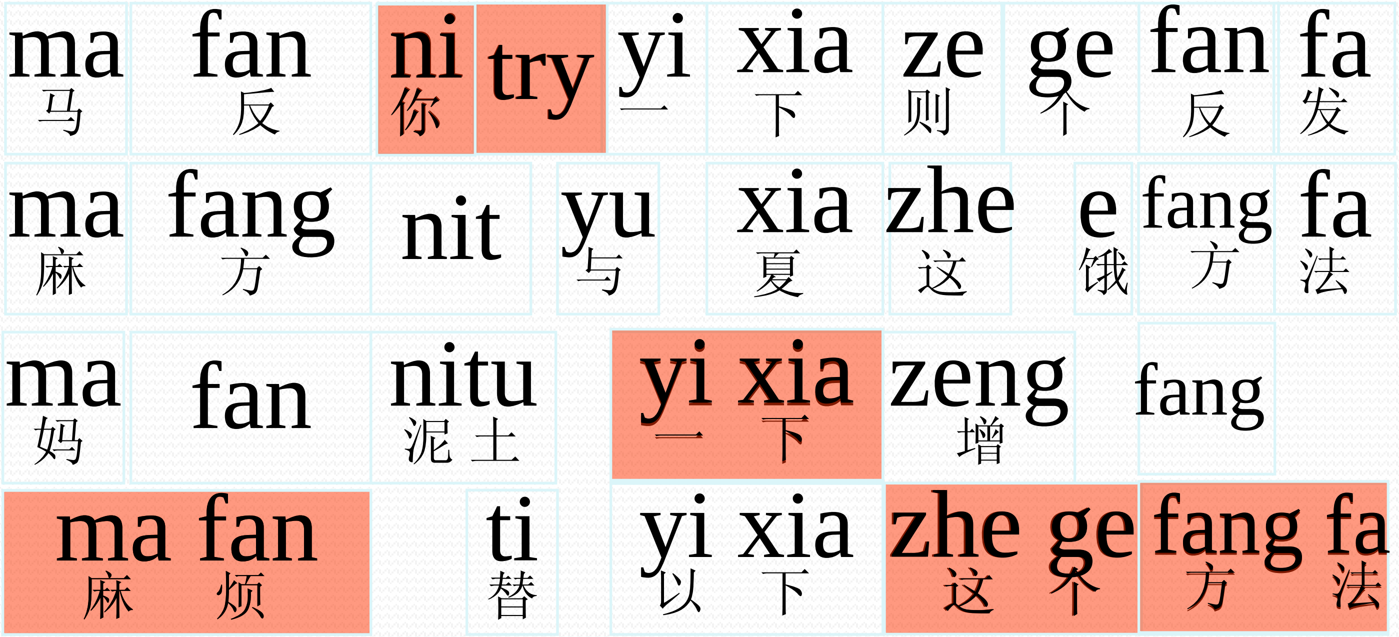
Probability: a brief refresher (2/2)

- H: assumptions on which the probabilities are based
- Product rule –from the def 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 def
  - $P(x|H) = \sum_y P(x, y|H) = \sum_y P(x|y, H)P(y|H)$
- 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

mafangnitryyixoazegefanfa



ma 马 | fan 反 | ni 你 | try | yi 一 | xia 下 | ze 则 | ge 个 | fan 反 | fa 发

ma 麻 | fang 方 | nit | yu 与 | xia 夏 | zhe 这 | e 饿 | fang 方 | fa 法

ma 妈 | fan | nitu 泥土 | yi xia 一下 | zeng 增 | fang

ma fan 麻烦 | ti 替 | yi xia 以下 | zhe ge 这个 | fang fa 方法

A Bayesian approach to IME

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

$$W = \text{argmax}_w P(W|A)$$

$$W = \text{argmax}_w \frac{P(A|W)P(W)}{P(A)}$$

$$W = \text{argmax}_w P(A|W)P(W)$$

- $P(A|W)$: typing (translation) model
  - Dealing with typing error, e.g., zh → 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
  - $\theta = \text{argmax}_{\theta} P(W|A, \theta)$
- Search: finding “best” conversion given the model
  - $W = \text{argmax}_{W} P(W|A, \theta)$
- 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

$$pplx = 2^{H} \quad where\ H = \frac{1}{|W|} \log P(W)$$

- 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

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)

$$
\begin{aligned}
P(\mathit{the\ dog\ of\ our\ neighbor\ barks}) = {} & P(\mathit{the} \mid {<}s{>}) \\
& \times P(\mathit{dog} \mid {<}s{>}, \mathit{the}) \\
& \times P(\mathit{of} \mid {<}s{>}, \mathit{the, dog}) \\
& \ldots \\
& \times P(\mathit{barks} \mid {<}s{>}, \mathit{the, dog, of, our, neighbor}) \\
& \times P({<}/s{>} \mid {<}s{>}, \mathit{the, dog, or, our, neighbor, barks})
\end{aligned}
$$

$$
\begin{aligned}
P(w_1, w_2 \ldots w_n) = {} & P(w_1 \mid {<}s{>}) \\
& \times P(w_2 \mid {<}s{>}\, w_1) \\
& \times P(w_3 \mid {<}s{>}\, w_1 w_2) \\
& \ldots \\
& \times P(w_n \mid {<}s{>}\, w_1 w_2 \ldots w_{n-1}) \\
& \times P({<}/s{>} \mid {<}s{>}\, w_1 w_2 \ldots w_n)
\end{aligned}
$$

Word n-gram model

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

model	**# parameters**
unigram $P(w_1)$	20,000
bigram $P(w_2\|w_1)$	400M
trigram $P(w_3\|w_1w_2)$	8 x 10^{12}
fourgram $P(w_4\|w_1w_2w_3)$	1.6 x 10^{17}

From Manning and Schütze 1999: 194

Word N-gram model

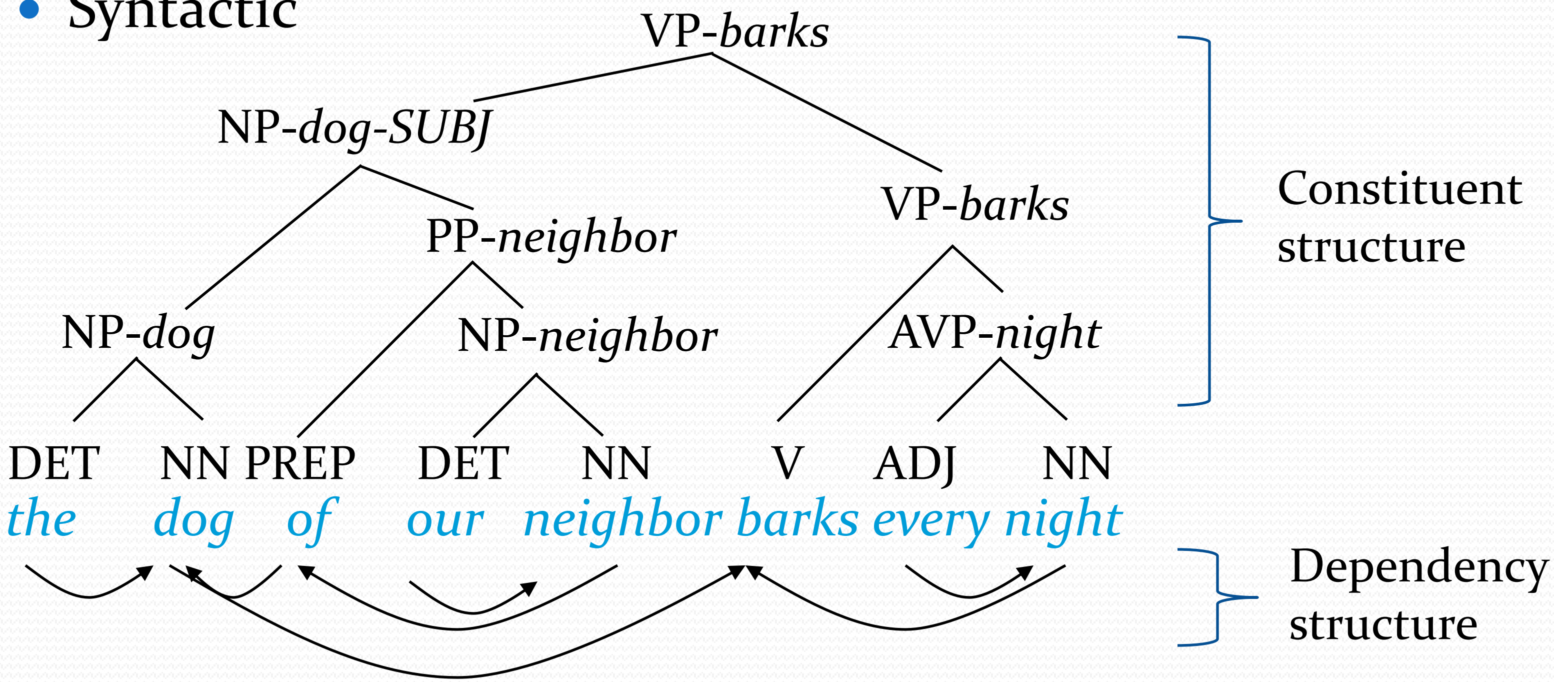
- Markov independence assumption
  - A word depends only on *N-1* preceding words
  - *N*=3 → word trigram model
- Reduce the number of *parameters* in the model
  - By forming *equivalence classes*
- Word trigram model

$$P(w_i \mid \text{<s>}\, w_1, w_2 \ldots w_{i-2}\, w_{i-1}) = P(w_i \mid w_{i-2}\, w_{i-1})$$

$$\begin{aligned} P(w_1, w_2 \ldots w_n) = {} & P(w_1 \mid \text{<s>}) \\ & \times P(w_2 \mid \text{<s>}\, w_1) \\ & \times P(w_3 \mid w_1 w_2) \\ & \ldots \\ & \times P(w_n \mid w_{n-2}\, w_{n-1}) \\ & \times P(\text{</s>} \mid w_{n-1}\, w_n) \end{aligned}$$

But language has structure!

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



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*

$$P(w|h) = \sum_s P(w,s|h) = \sum_s P(s|h)P(w|s,h)$$

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

- Define mapping function Φ
  - Φ maps word history into equivalence classes

$$P(w_i \mid w_1 \ldots w_{i-1}) = P(w|h) = P(w|\Phi(h))$$

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

$$
\begin{aligned}
P(w|h) &= \sum_s P(s|h) P(w|\Phi(s,h)) \\
&\approx \sum_s \frac{P(s|h)}{\sum_s P(s|h)} P(w|\Phi(s,h)) \quad \leftarrow \text{N-best approximation} \\
&\approx \max_s P(w|\Phi(s,h)), \text{ where } s = \arg\max_s P(s|h) \quad \leftarrow \text{Viterbi approximation}
\end{aligned}
$$

Defining Φ

- *s* is a chunk sequence
  - Φ(*s*) → two previous headword
  - Headword trigram model (Gao et al., 2002b)
- *s* is a dependency graph
  - Φ(*s*) → linked word to its left
  - Dependency LM (Gao and Suzuki, 2003)
- *s* is a word cluster sequence
  - Φ(*s*) → 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 Φ(s) as two previous headwords
- Example
  - [*The <u>dog</u>*] [*of our <u>neighbor</u>*] [*<u>barks</u>*] [*every <u>night</u>*]

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 Φ(s) as two previous headwords
- Example
  - [*The dog*] [*of our neighbor*] [*barks*] [*every night*]

    h_{i-2} h_{i-1} w_i

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)

$$P(w_i \,|\, \Phi(w_1 ... w_{i-1})) = P(H_i \,|\, \Phi(w_1 ... w_{i-1})) \times P(w_i \,|\, \Phi(w_1 ... w_{i-1}) H_i) + P(F_i \,|\, \Phi(w_1 ... w_{i-1})) \times P(w_i \,|\, \Phi(w_1 ... w_{i-1}) F_i)$$

- Incorporating assumptions using headword
  - Dependency between headwords (*dog~barks*)
  - Headword dependency is permutable (*barks~dogs*)

$$P(w_i \,|\, \Phi(w_1 ... w_{i-1}) H_i) = \boxed{\lambda_1 (\boxed{\lambda_2 P(w_i \,|\, h_{i-2} h_{i-1} H_i)} + \boxed{(1-\lambda_2) P(w_i \,|\, h_{i-1} h_{i-2} H_i))}} \boxed{+ (1-\lambda_1) P(w_i \,|\, w_{i-2} w_{i-1} H_i)}$$

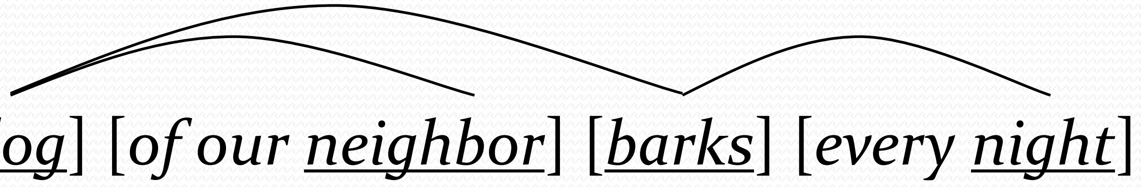
Gao and Suzuki, Weihan-2007

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)

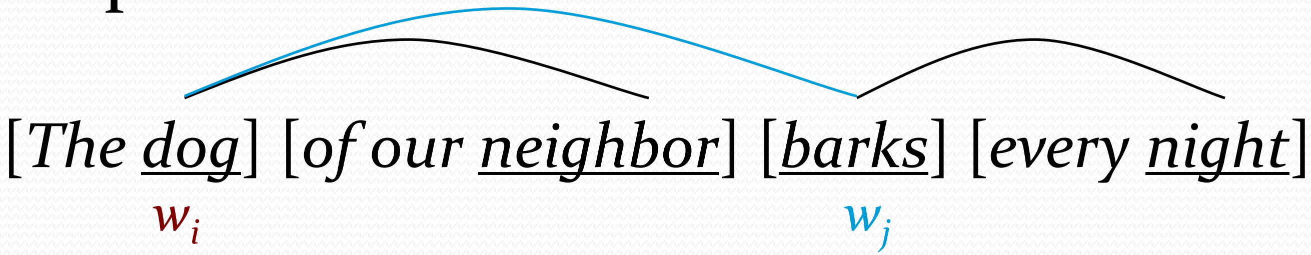
- *s* is a dependency graph among headwords
- Constraint on dependency structure *D*
  - Planar: no line crossing
  - Acyclic: contains no cycle
  - Define Φ(s) as the linked word on the left
- Example



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

Dependency language model (DLM)

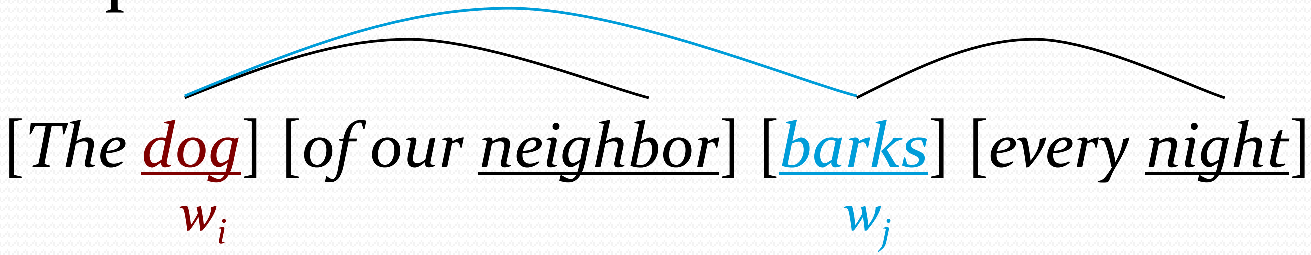
- *s* is a dependency graph among headwords
- Constraint on dependency structure *D*
  - Planar: no line crossing
  - Acyclic: contains no cycle
  - Define Φ(s) as the linked word on the left
- Example



  - [*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 Φ(s) as the linked word on the left
- Example



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

- Advantage
  - Capture *long-distance* dependency

Dependency parsing

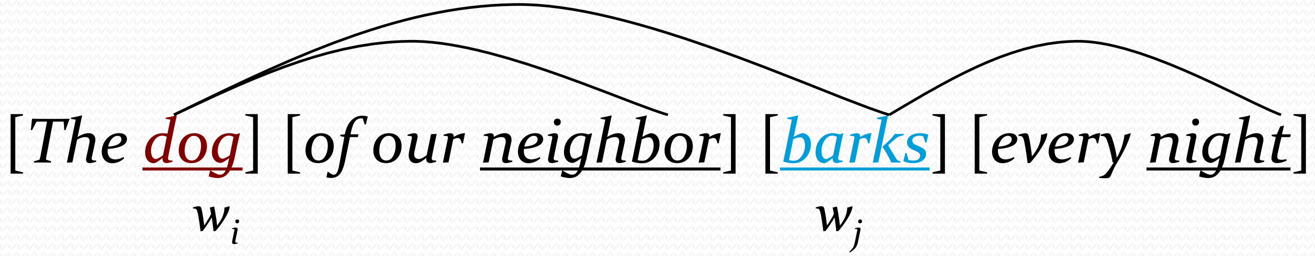
- The most probably dependency *D* is generated by

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

- 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)

$$P(w_j \mid \Phi(W_{j-1}, D_{j-1})) = \begin{cases} \boxed{\begin{array}{l} \lambda_1 \boxed{(P(w_j \mid w_i, R))} \\ \quad + (1-\lambda_1) P(w_j \mid w_{j-2}, w_{j-1}) \end{array}} & w_j\text{: headword} \\ \boxed{P(w_j \mid w_{j-2}, w_{j-1})} & w_j\text{: function word} \end{cases}$$



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

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 Φ(s) as two previous word clusters

- Example
  - *The poodle barks every night*
    - Estimate of *P* (*barks* | *poodle*) may be inaccurate
    - Estimate of *P* (*barks* | *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

$$
\begin{aligned}
&\prod_{i=1}^{N} P(W_i \mid w_{i-1}) \times P(w_i \mid W_i) \\
&= \prod_{i=1}^{N} \frac{P(w_{i-1}W_i)}{P(w_{i-1})} \times \frac{P(W_i w_i)}{P(W_i)} \\
&= \prod_{i=1}^{N} \frac{P(W_i w_i)}{P(w_{i-1})} \times \frac{P(w_{i-1}W_i)}{P(W_i)} \\
&= \prod_{i=1}^{N} \frac{P(w_i)}{P(w_{i-1})} \times P(w_{i-1} \mid W_i)
\end{aligned}
$$

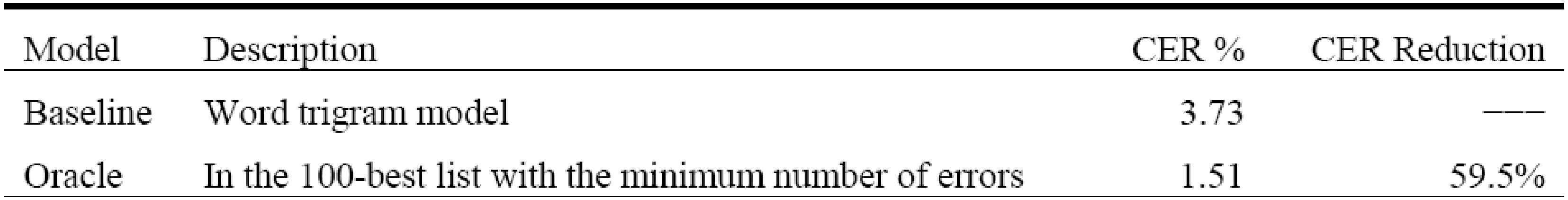
  - Sufficient to maximize $\prod_{i=1}^{N} P(w_{i-1} \mid W_i)$

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)

$$\frac{\text{\#chars wrongly converted}}{\text{\#chars in the target sentence}}$$

Results on Japanese IME (Gao and Suzuki, 2004)



Model	Description	CER %	CER Reduction
Baseline	Word trigram model	3.73	–––
Oracle	In the 100-best list with the minimum number of errors	1.51	59.5%

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
  - $P(\text{n} \mid \text{u}, \text{N})$, 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

- θ: model; X: data
- $\theta = \mathrm{argmax} P(\theta|X) = \mathrm{argmax} P(X|\theta)\mathrm{P}(\theta)/\mathrm{P}(X)$
  - Assume a uniform prior $P(\theta) = Const$
  - $P(X)$ is independent of θ, and is dropped
- $\theta = \mathrm{argmax}\ P(\theta|X) \approx \mathrm{argmax}\ P(X|\theta)$
  - Where $P(X|\theta)$ is the likelihood of parameter
- Key difference between MLE and Bayesian Estimation
  - MLE assume that θ is fixed but unknown,
  - Bayesian estimation assumes that θ itself is a random variable with a prior distribution $P(\theta)$.

MLE for trigram LM

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

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

- 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
  - $\text{Count}(w_1 w_2 w_3) = 0$
  - $P_{ML}(w_3|w_1 w_2) = \text{Count}(w_1 w_2 w_3)/\text{Count}(w_1 w_2) = 0$
  - $P(\text{W}) = P_{ML}(w_1)\, P_{ML}(w_2|w_1) \prod_i P_{ML}(w_i|w_{i-2}\, w_{i-1}) = 0$
  - $W = \text{argmax}_W P(A|W)P(W) =$… 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 λ's to vary – value of λ is a function of Count(.)

$$\begin{aligned} P(w_3|w_1,w_2) = {} & \lambda_1\big(\mathrm{C}(w_1,w_2,w_3)\big)P_{ML}(w_3|w_1,w_2) \\ & +\lambda_2(\mathrm{C}(w_2,w_3))P_{ML}(w_3|w_2) \\ & +\lambda_3(\mathrm{C}(w_3))P_{ML}(w_3) \end{aligned}$$

where $\lambda_1\big(\mathrm{C}(w_1,w_2,w_3)\big) + \lambda_2(\mathrm{C}(w_2,w_3)) + \lambda_3(\mathrm{C}(w_3)) = 1$

How to estimate λ's

- Split data into training, dev, test
- Optimize λ's on dev data (i.e., pick the best value of λ)

$$\boldsymbol{\lambda} = \mathrm{argmax}_{\boldsymbol{\lambda}} \sum_{(w_1,w_2,w_3) in\ dev\ data} \log P(w_3|w_1 w_2)$$

- Can use EM (expectation maximization) algorithm to find the λ'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

$$P(w_3|w_1,w_2) = \begin{cases} \dfrac{C(w_1,w_2,w_3) - D}{C(w_1,w_2)}, if\ C(w_1,w_2,w_3) > 0 \\ \alpha(w_1,w_2)P(w_3|w_2), if\ C(w_1,w_2,w_3) = 0 \end{cases}$$

- $D \in (0,1)$ is a discount constant – absolute discount
- α 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

$$\mathbf{D} = \operatorname{argmax}_{\mathbf{D}} \sum_{(w_1,w_2,w_3) in\ dev\ data} \log P(w_3|w_1 w_2)$$

- 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 λ = 0.4
- Refer to (Brant et al., 2007)

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

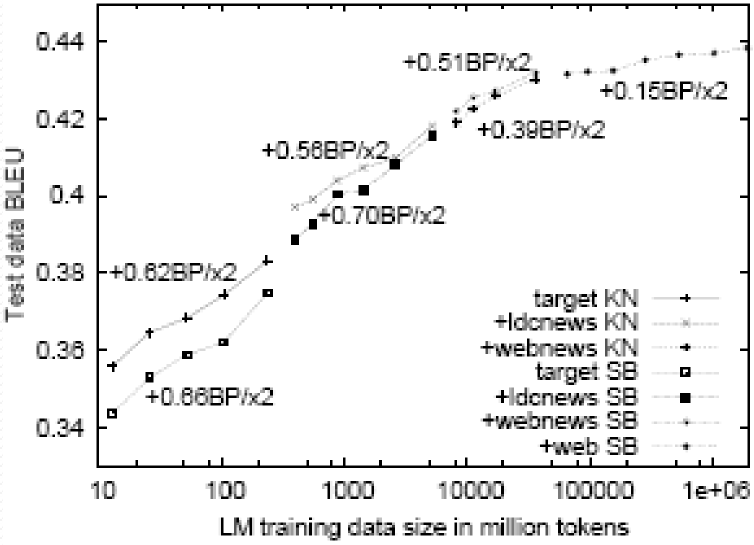


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

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

Discriminative training

- MLE – maximizing P(X|θ)
- Discriminative training – maximizing P(θ|X)

$$P(\theta|\mathrm{X}) = \frac{P(\mathrm{X}|\theta)P(\theta)}{P(\mathrm{X})} = \frac{P(\mathrm{X}|\theta)P(\theta)}{\sum_{\theta'} P(\mathrm{X}|\theta')P(\theta')}$$

assume a uniform prior $P(\theta) = C$

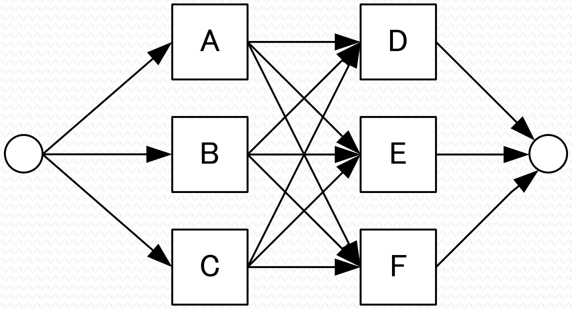
$$\begin{aligned}\mathrm{argmax} P(\theta|\mathrm{X}) &= \mathrm{argmax} \frac{P(\mathrm{X}|\theta)}{P(\mathrm{X}|\theta) + \sum_{\theta' \neq \theta} P(\mathrm{X}|\theta')} \\ &= \mathrm{argmax} \frac{1}{1 + \frac{\sum_{\theta' \neq \theta} P(\mathrm{X}|\theta')}{P(\mathrm{X}|\theta)}} \\ &= \mathrm{argmax} \frac{P(\mathrm{X}|\theta)}{\sum_{\theta' \neq \theta} P(\mathrm{X}|\theta')}\end{aligned}$$

- 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☹)



$P(A | <s>) = 0.2$
$P(B | <s>) = 0.15$
$P(C | <s>) = 0.1$

$P(D|A) = 0.2$
$P(E|A) = 0.15$
$P(F|A) = 0.01$

$P(D|B) = 0.08$
$P(E|B) = 0.1$
$P(F|B) = 0.05$

$P(D|C) = 0.1$
$P(E|C) = 0.1$
$P(F|C) = 0.15$

$P(</s> |D) = 0.2$
$P(</s> |E) = 0.1$
$P(</s> |F) = 0.1$

Rank	W	-logP(W)
1	<s>, A, D, </s>	2.1
2	<s>, A, E, </s>	2.5
3	<s>, B, D, </s>	2.6
4	<s>, C, D, </s>	2.7
5	<s>, B, E, </s>	2.8
6	<s>, C, F, </s>	2.8
7	<s>, C, E, </s>	3.0
8	<s>, B, F, </s>	3.1
9	<s>, A, F, </s>	3.7

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
  - Gao et al., (2002b, 2002c), Gao and Suzuki (2003, 2004)
- Better training method: 2005-present
  - directly minimize error rate
  - Gao et al., (2006, 2007)
- YOU CAN DO BETTER THAN US!

Better training data: Chinese IME results

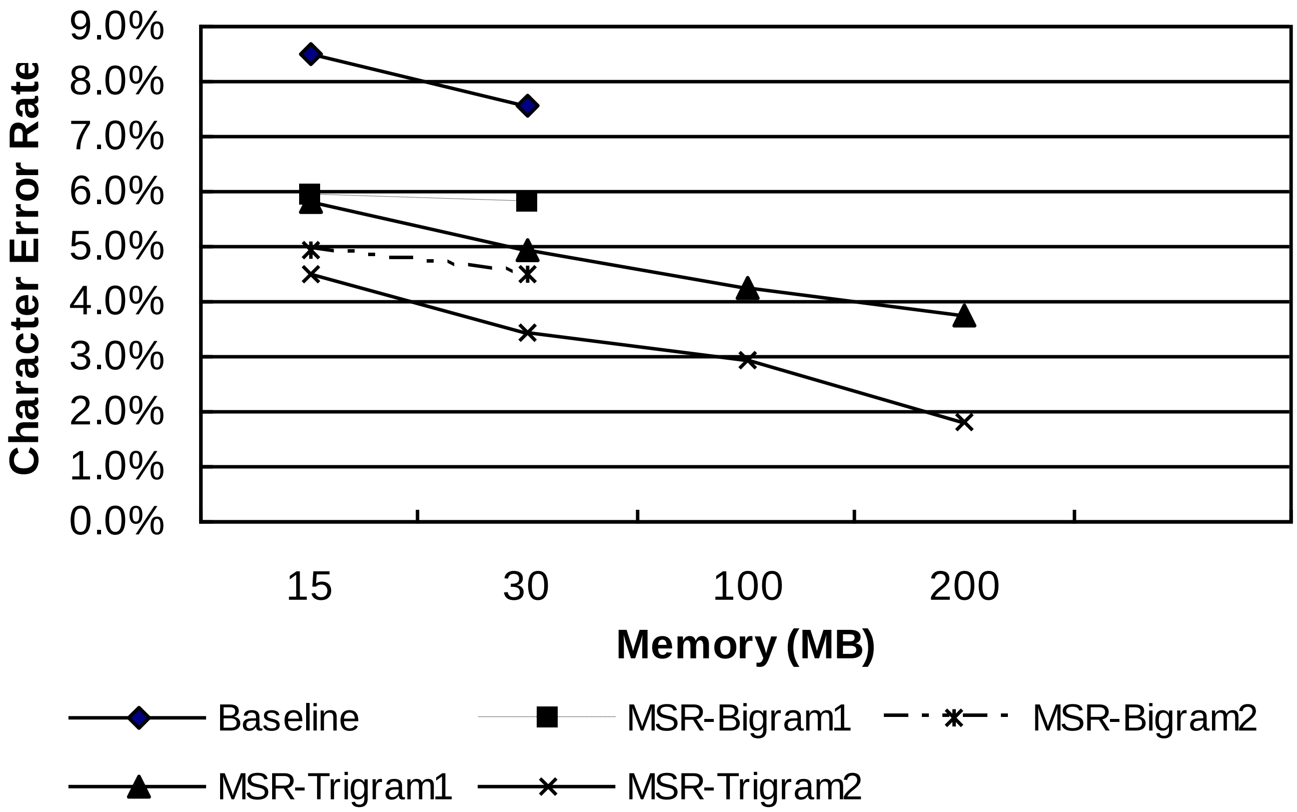
(Gao et al., 2002a)

	Baseline	MSR-Bigram1	MSR-Bigram2	MSR-Trigram1	MSR-Trigram2
Training Set	IME	Total	Total	Total	Total
Lexicon & Segmentation Optimization	NO	YES	YES	YES	YES
Training Set Filtering	NO	YES (seed set: Total)	YES (seed set: Total)	YES (seed set: Total)	YES (seed set: Total)
Training Set Domain Adaptation	NO	NO	YES (seed set: IME training set)	NO	YES (seed set: IME training set)
Pruning Method	Count Cutoff	Predict Cluster + Stolcke	Predict Cluster + Stolcke	Stolcke	Stolcke

Table 10: Summary of techniques used in system evaluation

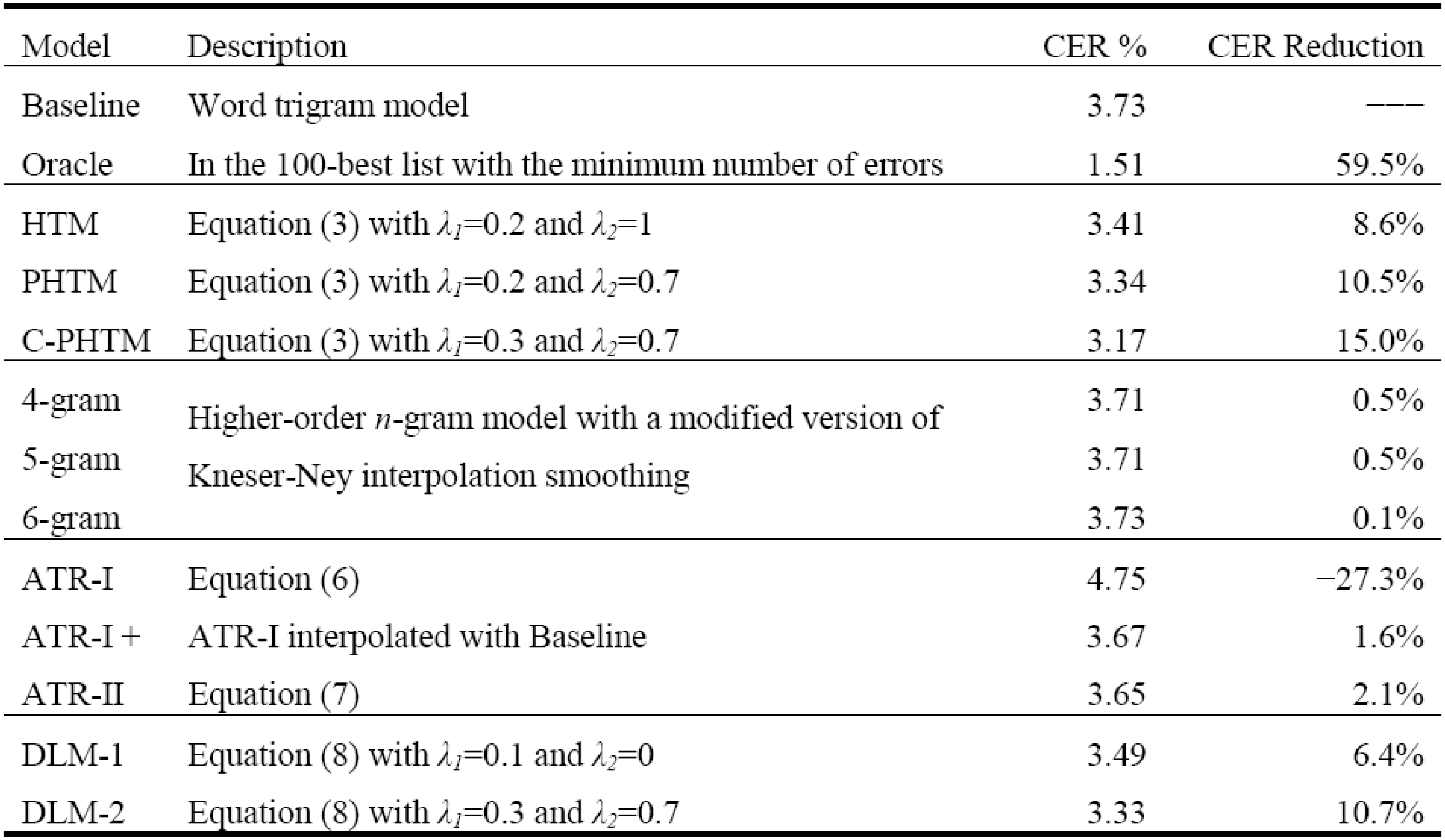
Better training data: Chinese IME results

(Gao et al., 2002a)



Better modeling: Japanese IME results

(Gao and Suzuki, 2004)



Model	Description	CER %	CER Reduction
Baseline	Word trigram model	3.73	---
Oracle	In the 100-best list with the minimum number of errors	1.51	59.5%
HTM	Equation (3) with λ_1=0.2 and λ_2=1	3.41	8.6%
PHTM	Equation (3) with λ_1=0.2 and λ_2=0.7	3.34	10.5%
C-PHTM	Equation (3) with λ_1=0.3 and λ_2=0.7	3.17	15.0%
4-gram	Higher-order *n*-gram model with a modified version of Kneser-Ney interpolation smoothing	3.71	0.5%
5-gram		3.71	0.5%
6-gram		3.73	0.1%
ATR-I	Equation (6)	4.75	−27.3%
ATR-I +	ATR-I interpolated with Baseline	3.67	1.6%
ATR-II	Equation (7)	3.65	2.1%
DLM-1	Equation (8) with λ_1=0.1 and λ_2=0	3.49	6.4%
DLM-2	Equation (8) with λ_1=0.3 and λ_2=0.7	3.33	10.7%

Better training: Japanese IME results
(Gao et al., 2007)

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

References

- Abney, Steven. 1991. Parsing by chunks. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-based parsing.* Kluwer Academic Publishers, Dordrecht.
- Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP*.
- Bellegarda, J. R. 2004. Latent semantic language modeling for speech recognition. *Mathematical foundations of speech and language processing*. Johnson, M et al. (Eds.), Springer, pp.73-103.
- Chelba, Ciprian and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4): 283-332.
- Chen, S. F., and Goodman, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359-394, October.
- Gao, Jianfeng, Galen Andrew, Mark Johnson and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *ACL.*
- Gao, Jianfeng, Hisami Suzuki and Wei Yuan. 2006. An empirical study on language model adaptation. *ACM Trans on Asian Language Information Processing*, 5(3): 207-227.
- Gao, Jianfeng and Hisami Suzuki. 2004. Capturing long distance dependency for language modeling: an empirical study. In *IJCNLP.*
- Gao, Jianfeng and Hisami Suzuki. 2003. Unsupervised learning of dependency structure for language modeling. In *ACL.*
- Gao, Jianfeng, Joshua Goodman, Mingjing Li and Kai-Fu Lee. 2002a. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, 1-1: 3-33.
- Gao, J., H. Suzuki and Y. Wen. 2002b. Exploiting headword dependency and predictive clustering for language modeling. In *EMNLP*.
- Gao, J, J. Goodman, G. Cao, H. Li. 2002c. Exploring asymmetric clustering for statistical language modeling. *ACL.*
- Gao, J. Goodman, J. and Miao, J. 2001. The use of clustering techniques for language model – application to Asian language. *Computational Linguistics and Chinese Language Processing*. Vol. 6, No. 1, pp 27-60.
- Goodman, J. 2001. A bit of progress in language modeling. In *Computer Speech and Language,* October 2001, pp 403-434.
- Manning, C., and H. Chutze. 1999. *Foundations of statistical natural language processing*. MIT Press. Cambridge.
- Huang, X., A. Acero, and H-W Hon. 2001. *Spoken language processing.* Prentice Hall PTR.
- Roark, Brian, Murat Saraclar and Michael Collins. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *ICASSP*.
- Rosenfeld, Ronald. 1994. Adaptive statistical language modeling: a maximum entropy approach. Ph.D. thesis, Carnegie Mellon University.
- Teh, Yee Whye. 2006. A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes. In *ACL*.
- Yuret, D. 1998. Discovery of linguistic relations using lexical attraction. Ph.D. thesis, MIT.

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.