

A Scalable Distributed Syntactic, Semantic and Lexical Language Model

Shaojun Wang

`shaojun.wang@wright.edu`

`http://www.cs.wright.edu/~swang`

Kno.e.sis Center

Department of Computer Science and Engineering

Wright State University

Joint work with Ming Tan, Wenli Zhou, Lei Zheng

What is natural language?

- Natural language is perhaps one of the most intriguing stochastic processes
 - Natural language encodes messages via complex, hierarchically organized sequences.
 - The local lexical structure of the sequence conveys surface information, while the syntactic structure, encoding long range dependencies, carries deeper semantic information.

Language modeling: A density estimation problem

- Accurately calculate the probability of naturally occurring word sequences in human natural language.
- One of the core technologies of machine translation and speech recognition systems
- Three types of language models
 1. ***n*-gram** (Markov 1902, Shannon 1948, Jelinek et al. 1970s)
Encodes local word interactions
 - The ***workhorse*** of state-of-the-art machine translation and speech recognition systems
 - **Ignores the rich syntactic and semantic structures that constrain natural languages**
 2. **Syntactic model** (Chelba and Jelinek 2000, Charniak 2001, Roark 2001)
Exploits sentence level syntactic structure
 3. **Topic model** (Bellegarda 2000, Gildea and Hofmann 1999, Rosenfeld 1996, Wallach 2006)
Exploits document level semantic content

Language modeling: A density estimation problem (continue)

- But
 - Each model targets some specific, distinct linguistic phenomena
 - Lack of a unified probabilistic framework to encode arbitrary aspects of natural language with tractable training algorithm

How to combine statistical models?

- Linear interpolation
 - Easy to use (good)
 - Makes suboptimal use of components; limited improvement (bad)
- Maximum entropy approach
 - Very popular in natural language processing community due to the work by Berger et al., 1996 and Della Pietra et al., 1997
 - Constrained convex optimization problem

$$\begin{aligned} & \max_{p(x)} H(p(x)) \\ \text{s.t. } & \sum_x p(x) f_i(x) = \sum_x \tilde{p}(x) f_i(x), \quad i = 1, \dots, N \end{aligned}$$

Maximum entropy = MLE for undirected MRFs

- **Dual problem:** Maximum likelihood estimation

$$p_{\lambda^*}^{\text{MLE}}(x) = \arg \max_{p_{\lambda}(x)} \sum \tilde{p}(x) \log p_{\lambda}(x)$$

where $p_{\lambda}(x) = \frac{1}{\Phi_{\lambda}} \exp\left(\sum_{i=1}^N \lambda_i f_i(x)\right)$: undirected MRF

$$\underline{p^{\text{ME}}(x) = p_{\lambda^*}^{\text{MLE}}(x)}$$

- Supervised learning:
Only model distributions over explicitly observed features, but for natural language there are hidden structures we do not observe directly (bad)
- Training is expensive:
Mainly in feature expectation and normalization;
If the statistical model is too complex, training becomes intractable, have to use Markov chain Monte Carlo (MCMC) sampling methods (bad)

How to combine statistical models?

- We propose a principled approach, **directed Markov random fields (MRFs)**
 - Encode word lexical information, sentence syntactic structure, and document semantic content with a tractable parameter estimation algorithm by exploiting certain **factorization property**
 - When applied to combine trigram, PCFG and PLSA (Wang et al. 2005)
 - \exists a generalized inside-outside algorithm with cubic time complexity
 - Achieved moderate perplexity reduction on 40 million tokens corpus
 - We now combine n -gram, structured language model (SLM) and PLSA
 - SLM is more **complex** and **powerful** than PCFG

Undirected MRFs vs directed MRFs?

- Undirected Markov random fields

$$p_{\lambda}(x) = \exp(\langle \lambda, f(x) \rangle - \log(\Phi_{\lambda}))$$

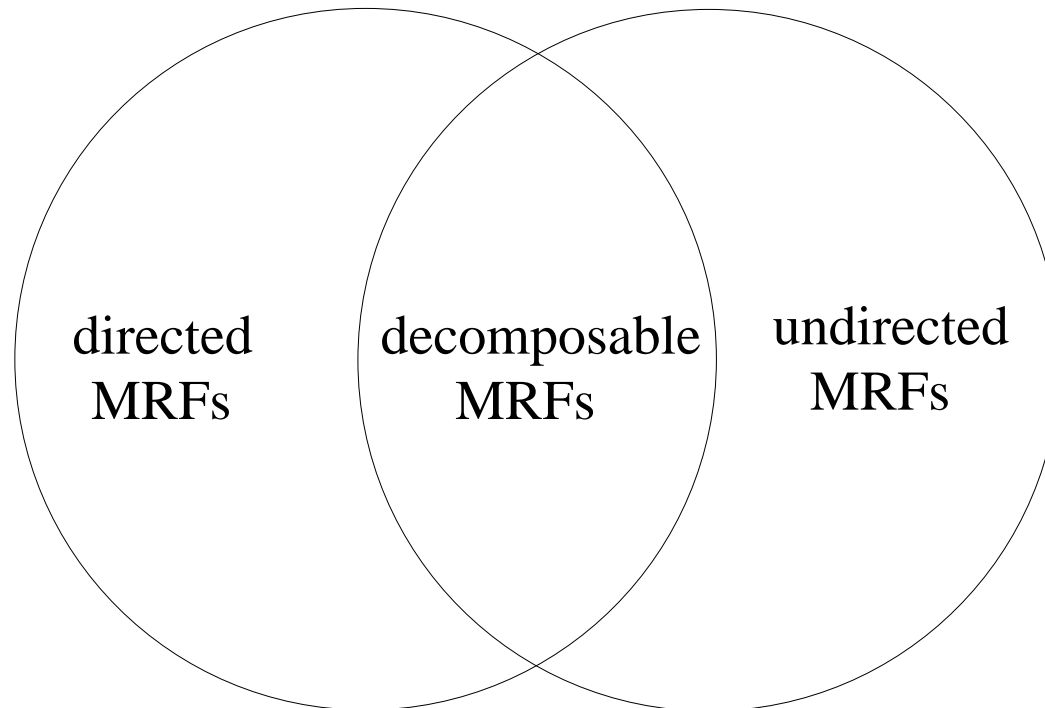
ONE global normalization factor: Φ_{λ}

- Directed Markov random fields

$$p_{\lambda}(x) = \prod_j \exp(\langle \lambda_j, f(x_j, \pi_j) \rangle - \log(\Phi_{\lambda_j}(\pi_j)))$$

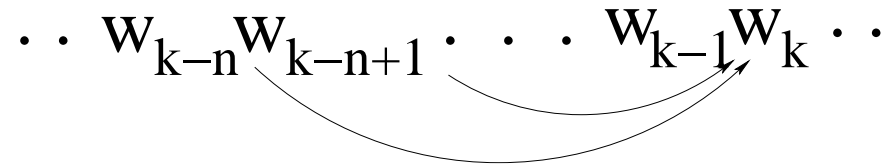
MANY local normalization factor: $\Phi_{\lambda_j}(\pi_j)$

Undirected MRFs vs directed MRFs?



- n -gram can be represented either by directed MRF or undirected MRF
 - directed MRF: relative frequency estimates with proper smoothing (good)
 - undirected MRF: feature expectation and normalization, plus optimization such as iterative scaling, coordinate descent, quasi-Newton etc. (bad)
- PCFG and SLM can be represented either by directed MRF or undirected MRF too

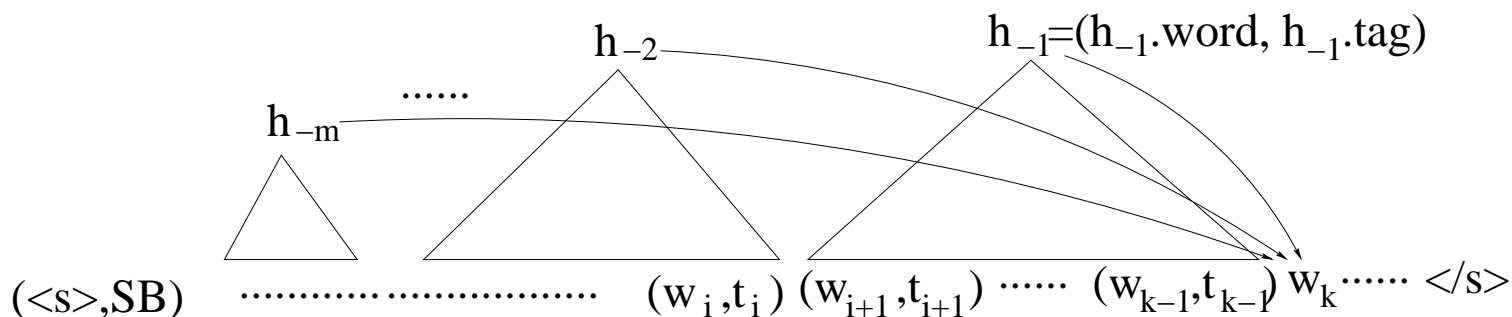
n -gram



- WORD-PREDICTOR: predicts each word on the basis of previous $n-1$ words

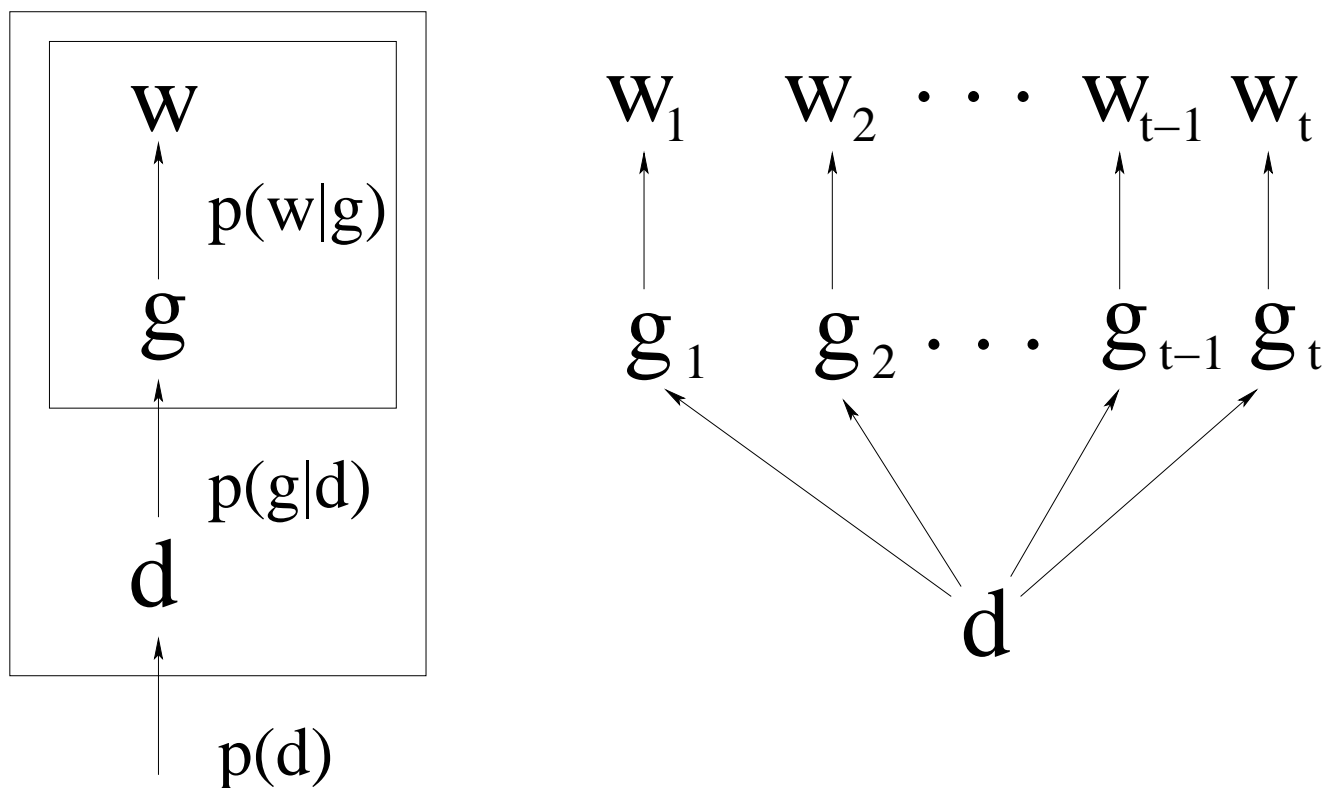
Structured language model (SLM)

- The SLM (Chelba and Jelinek 2000) uses syntactic information beyond the regular n -gram models to capture sentence level long range dependencies.
- m -th order SLM has three operators
 - The **WORD-PREDICTOR** generates the next word w_k with probability $p(w_k|h_{-m}^{-1})$ based on $h_{-m}^{-1} = h_{-1}, \dots, h_{-m}$, the m most recent exposed headwords in the word-parse k -prefix



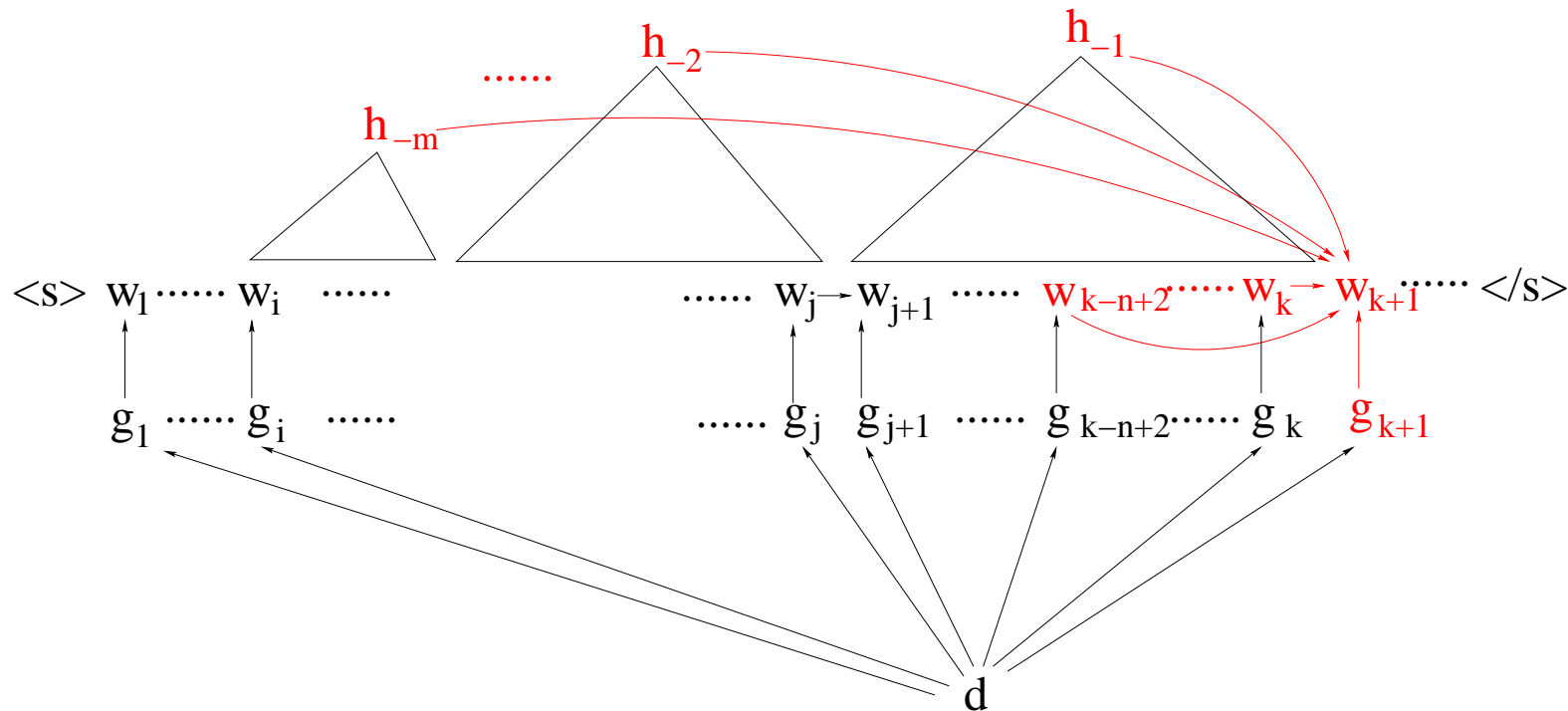
- The **TAGGER** attaches tag t_k to the most recently generated word w_k with probability $p(t_k|w_k, h_{-1}.tag, \dots, h_{-m}.tag)$
- The **CONSTRUCTOR** builds the partial parse T_k from T_{k-1} , w_k , and t_k in a series of moves ending with NULL, where a parse move a is made with probability $p(a|h_{-m}^{-1})$; $a \in \{(\text{unary, NTlabel}), (\text{adjoin-left, NTlabel}), (\text{adjoin-right, NTlabel}), \text{null}\}$

Probabilistic latent semantic analysis (PLSA)



- Choose a document d with probability $p(d)$;
- SEMANTIZER: select a semantic class g with probability $p(g|d)$
- WORD-PREDICTOR: predict a word w with probability $p(w|g)$.

Combining n -gram, m -order SLM and PLSA under directed MRF paradigm



- TAGGER and CONSTRUCTOR in SLM and SEMANTIZER in PLSA remain unchanged
- WORD-PREDICTOR generates the next word, w_{k+1} , based on most recent m exposed headwords h_{-m}^{-1} in the word-parse k -prefix, n -gram history

$w_{k-n+1}^k = w_{k-n+1}, \dots, w_k$ and its semantic content g_{k+1}

Likelihood of fully labeled sentence

$$P_p(W, T, G|d) = \prod_{g \in \mathcal{G}} \left(p(g|d)^{\#(g, W, G, d)} \prod_{h_{-1}, \dots, h_{-m} \in \mathcal{H}} \left(\prod_{w, w_{-1}, \dots, w_{-n+1} \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g)^{\#(w_{-n+1}^{-1} w h_{-m}^{-1} g, W^l, T^l, G^l, d)} \right. \right. \\ \left. \left. \prod_{t \in \mathcal{O}} p(t|w h_{-m}^{-1} \text{tag})^{\#(t, w h_{-m}^{-1} \text{tag}, W^l, T^l, d)} \prod_{a \in \mathcal{A}} p(a|h_{-m}^{-1})^{\#(a, h_{-m}^{-1}, W^l, T^l, d)} \right) \right)$$

with local normalizations

$$\sum_{w \in \mathcal{V}} p(w|w_{-n+1}^{-1} h_{-m}^{-1} g) = 1$$

$$\sum_{t \in \mathcal{O}} p(t|w h_{-m}^{-1} \text{tag}) = 1$$

$$\sum_{a \in \mathcal{A}} p(a|h_{-m}^{-1}) = 1$$

$$\sum_{g \in \mathcal{G}} p(g|d) = 1$$

Efficient training algorithm?

- Inference and parameter estimation problems seem to be plausibly intractable
 - The added complexity due to n -gram and PLSA models in the composite n -gram/ m -SLM/PLSA language model
 - Violation of tree structure in the topology of the underlying random field model

Generalized inside-outside algorithm

- The likelihood of a training corpus \mathcal{D} , a collection of documents, is

$$\mathcal{L}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \left(\prod_l \left(\sum_{G^l} \left(\sum_{T^l} P_p(W^l, T^l, G^l | d) \right) \right) \right)$$

- **Good news:**
 - Following Jelinek's ingenious definition of the inside and outside probabilities for SLM (Jelinek 2004)
 - \exists an exact EM training algorithm, **a generalized inside-outside algorithm** (Wang et al. 2006)
- **Bad news:**
 - Computational complexity $O(L^6)$, L : sentence length,
 - Exact EM is not practical for a large scale corpus even with the use of pruning on charts (Jelinek 2004)

N -best approximate EM algorithm

The linear time N -best list approximate EM involves two steps:

1. N -best list search: For each sentence W in document d , find N -best parse trees,

$$\mathcal{T}_N^l = \arg \max_{\mathcal{T}'^l_N} \left\{ \sum_{G^l} \sum_{T^l \in \mathcal{T}'^l_N} P_p(W^l, T^l, G^l | d), \|\mathcal{T}'^l_N\| = N \right\}$$

where \mathcal{T}_N is a collection of \mathcal{T}_N^l for sentences over entire corpus \mathcal{D} .

2. EM update: Perform one iteration (or several iterations) of EM algorithm to estimate model parameters that maximizes N -best-list likelihood of the training corpus \mathcal{D} .

$$\tilde{\mathcal{L}}(\mathcal{D}, p, \mathcal{T}_N) = \prod_{d \in \mathcal{D}} \left(\prod_l \left(\sum_{G^l} \left(\sum_{T^l \in \mathcal{T}_N^l} P_p(W^l, T^l, G^l | d) \right) \right) \right)$$

Iterate steps (1) and (2) until the convergence of the N -best-list likelihood.

Its convergence properties can be proven by Zangwill's global convergence theorem (Zangwill 1969)

N -best approximate EM algorithm (continue)

1. N -best list search: **A synchronous, multi-stack search strategy**
 - A set of stacks storing partial parses of the most likely ones for a given prefix W_k , the less probable parses are purged
 - The hypotheses are ranked according to $\log(\sum_{G_k} P(W_k, T_k, G_k | d))$
 - Each stack contains partial parses constructed by the same number of constructor operations
 - The width of the pruning is controlled by:
 - maximum number of stack entries
 - log-probability threshold

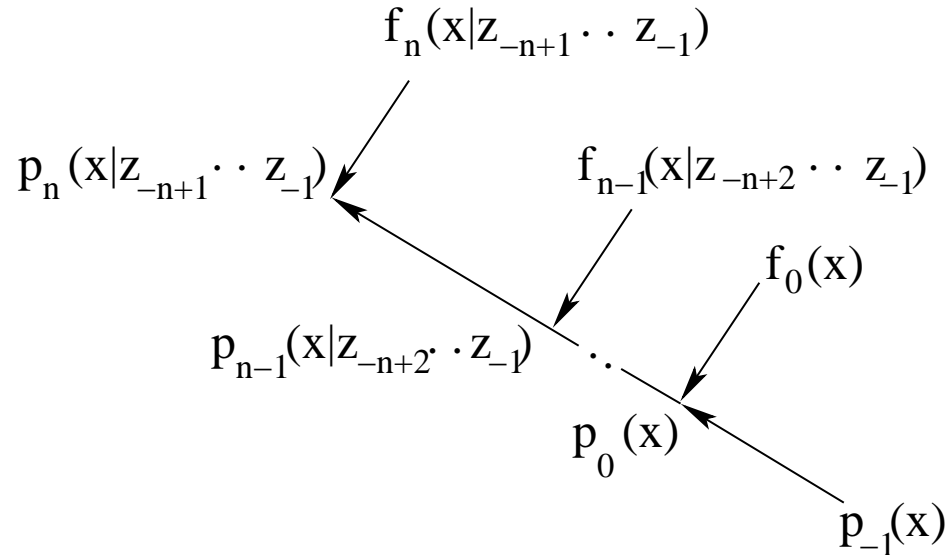
⇒ **Greedy best first search!**

N -best approximate EM algorithm (continue)

2. EM update:

- **E-step:** Compute expected counts
 - For the WORD-PREDICTOR and the SEMANTIZER, use forward-backward recursive formulas that are similar to those in hidden Markov models to compute the expected counts.
 - For the TAGGER and the CONSTRUCTOR, the expected count of $twh_{-m}^{-1}.tag$ and ah_{-m}^{-1} over parse T^l of sentence W^l in document d is the real count appeared in parse tree T^l times the posterior distribution $P_p(T^l | W^l, d)$.
- **M-step:** Assuming that the count ranges and the corresponding interpolation values for each order are kept fixed to their initial values, the only parameters to be re-estimated are the maximal order counts for each model component.
 - The interpolation scheme obtain a smooth probability estimate for each model component.

Linear interpolation for a Markov chain



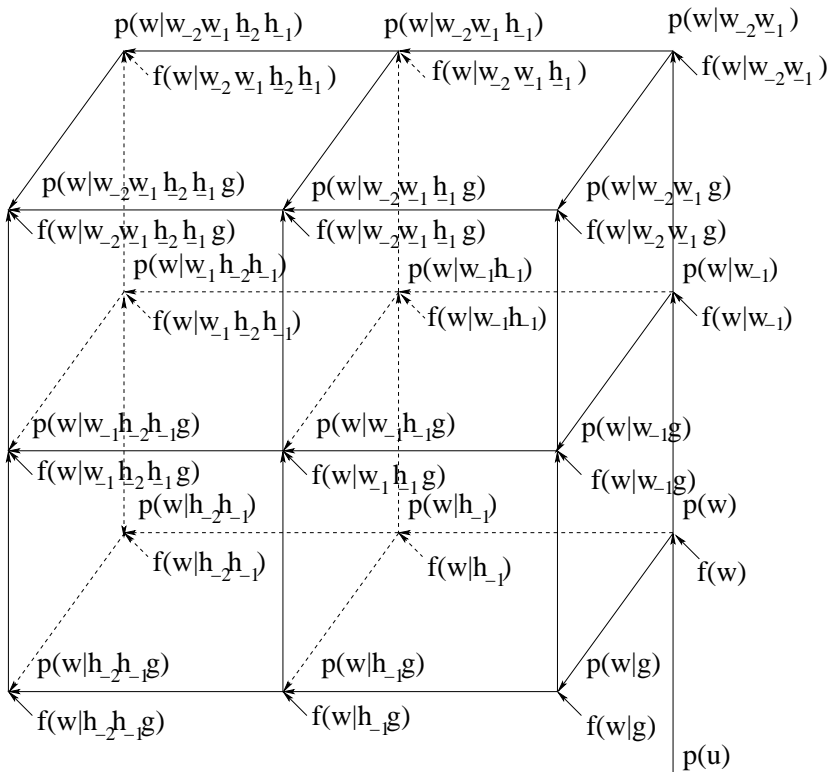
- Recursive mixing scheme of relative frequency estimates at different orders

$$\begin{aligned}
 p_n(x|z_{-n+1} \dots z_{-1}) &= \lambda(z_{-n+1} \dots z_{-1}) \cdot p_{n-1}(x|z_{-n+2} \dots z_{-1}) \\
 &\quad + (1 - \lambda(z_{-n+1} \dots z_{-1})) \cdot f_n(x|z_{-n+1} \dots z_{-1}) \\
 p_{-1}(x) &= \text{uniform}(x)
 \end{aligned}$$

- For TAGGER or CONSTRUCTOR, conditional context is a Markov chain
- For WORD-PREDICTOR, conditional context $w_{-n+1}^{-1} h_{-m}^{-1} g$ is **not** a Markov chain.

⇒ generalize Jelinek and Mercer's (1981) original recursive mixing scheme

Linear interpolation lattice for WORD-PREDICTOR of trigram/2-SLM/PLSA



The lattice is formed by 3 Markov chains, $w_{-2}w_{-1}$, $h_{-2}h_{-1}$ and g . Each vertex is visited in a bottom up, back to front, right to left order.

u : vocabulary

$p(u)$: uniform distribution of u

- Each vertex is linearly interpolated with lower vertice and its frequency, such as,

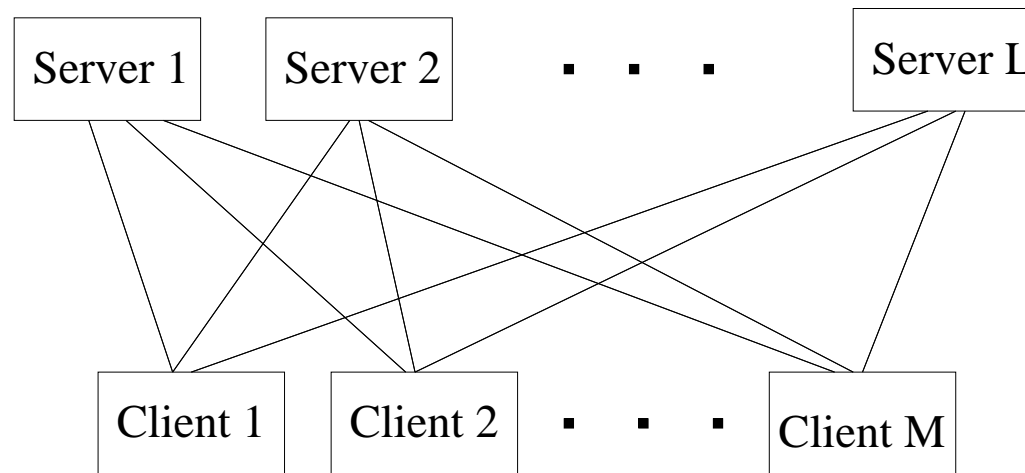
$$\begin{aligned}
 p(w|w_{-2}w_{-1}h_{-2}h_{-1}g) &= \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-1}h_{-2}h_{-1}g) \\
 &+ \lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-1}g) \\
 &+ \lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g) \cdot p(w|w_{-2}w_{-1}h_{-2}h_{-1}) \\
 &+ (1 - \lambda_w(w_{-2}w_{-1}h_{-2}h_{-1}g) - \lambda_h(w_{-2}w_{-1}h_{-2}h_{-1}g) \\
 &\quad - \lambda_g(w_{-2}w_{-1}h_{-2}h_{-1}g)) \cdot f(w|w_{-2}w_{-1}h_{-2}h_{-1}g)
 \end{aligned}$$

Distributed architecture

Challenges when handling large scale corpus

- Data can't be stored in a single machine
- Parameters can't be stored in a single machine

Strategy: adopt server-client paradigm



- Clients store partitioned data and perform “Map” function: compute expected counts
- Servers store parameters (counts) for “Reduce” function
- Hash $w_{k-n+1}^k h_{-m}^{-1} g_k$ by the last word w_k and topic g_k

Follow-up EM algorithm to improve word prediction power

Use a large amount of the partial parse trees that are generated during the synchronous, multi-stack search strategy to re-estimate WORD-PREDICTOR to improve its predictive power

- The *language model* probability assignment for the word at position $k+1$ in the input sentence can be written as:

$$P_p(w_{k+1}|W_k, d) = \sum_{h_{-m}^{-1} \in T_k; T_k \in Z_k, g_{k+1}} p(w_{k+1}|w_{k-n+2}^k h_{-m}^{-1} g_{k+1}) p(g_{k+1}|d) p(T_k|W_k, d)$$

where

$$p(T_k|W_k, d) = \frac{\sum_{G_k} p(W_k, T_k, G_k|d)}{\sum_{T_k \in Z_k} \sum_{G_k} p(W_k, T_k, G_k|d)}$$

to ensure a proper probability normalization over word strings W_k

- Z_k is the set of all parses present in our stacks at the current stage k
- G_k is the semantic node string up to k

Follow-up EM algorithm to improve word prediction power (continue)

- The likelihood of a training corpus \mathcal{D} that uses partial parse trees generated during the process of the synchronous, multi-stack search strategy can be written as

$$\tilde{\mathcal{L}}(\mathcal{D}, p) = \prod_{d \in \mathcal{D}} \prod_l \left(\prod_k P_p(w_{k+1}^{(l)} | W_k^l, d) \right) \quad (1)$$

- Estimate $p(w_{k+1} | w_{k-n+2}^k h_{-m}^{-1} g_{k+1})$ by maximizing Eqn. (1) using EM
 - **E-step:** gather expected joint counts $C(w_{k+1} w_{k-n+2}^k h_{-m}^{-1} g, d)$ of the WORD-PREDICTOR model with each count weighted by $p(w_{k+1} | w_{k-n+2}^k h_{-m}^{-1} g_{k+1}) p(g_{k+1} | d) p(T_k | W_k, d)$ normalized over $h_{-1}, \dots, h_{-m} \in T_k \in Z_k$ and g_{k+1}
 - **M-step:** uses the same count smoothing technique as that described in the N-best list approximate EM
- Adopt a similar distributed architecture
- In fact, most improvements are from this algorithm

Use the model for testing

- A document of the test data is not contained in the original training corpus, the parameters $p(g|d)$ have to be re-estimated by maximizing the probability of word subsequence seen so far, i.e., a pseudo-document $\tilde{d}_k = (W_k, S)$, while holding the other parameters fixed.
- We use three methods:
 - One step online EM, $\gamma = \frac{1}{|\tilde{d}_k|+1}$
 - Online EM with fixed learning rate, $\gamma = 0.2$

$$p(g|\tilde{d}_k) = \gamma \frac{\sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}} p(w_k | w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1} | W_{k-1}, \tilde{d}_{k-1})}{\sum_{h_{-m}^{-1} \in T_{k-1}; T_{k-1} \in Z_{k-1}, g} p(w_k | w_{k-n+1}^{k-1} h_{-m}^{-1} g) p(g|\tilde{d}_{k-1}) P_p(T_{k-1} | W_{k-1}, \tilde{d}_{k-1})} + (1 - \gamma) p(g|\tilde{d}_{k-1})$$

- Batch EM
- Perplexity results are sensitive to these three methods and the initial values
- Online EM with fixed learning rate not only has the cheapest computational cost but also leads to highest perplexity reductions

Combining n -gram, m -SLM and PLSA through latent maximum entropy (LME)

$$\max_{P(W,T,G,D)} - \sum_{W,T,G,D} P(W, T, G, D) \log P(W, T, G, D)$$

subject to the following **nonlinear** constraints

$$\sum_{W,T,G,D} P(W, T, G, D) \underline{f}(w_{-n+1}^{-1} w h_{-m}^{-1} g) = \sum_{W,D} \tilde{P}(W, D) \sum_{T,G} P(T, G|W, D) \underline{f}(w_{-n+1}^{-1} w h_{-m}^{-1} g)$$

$$\sum_{W,T,G,D} P(W, T, G, D) \underline{f}(t w h_m^{-1} . \text{tag}) = \sum_{W,D} \tilde{P}(W, D) \sum_{T,G} P(T, G|W, D) \underline{f}(t w h_m^{-1} . \text{tag})$$

$$\sum_{W,T,G,D} P(W, T, G, D) \underline{f}(a h_{-m}^{-1}) = \sum_{W,D} \tilde{P}(W, D) \sum_{T,G} P(T, G|W, D) \underline{f}(a h_{-m}^{-1})$$

$$\sum_{W,T,G,D} P(W, T, G, D) \underline{f}(g d) = \sum_{W,D} \tilde{P}(W, D) \sum_{T,G} P(T, G|W, D) \underline{f}(g d)$$

where $\tilde{P}(W, D)$ denotes the empirical distribution of a sentence in a document over training corpus. This is a **non-convex optimization** problem due to the nonlinear constraints and there is **no closed form solution**.

Combining n -gram, m -SLM and PLSA under undirected MRF paradigm

- The likelihood of fully labeled sentence

$$P_{\underline{\lambda}}(W, T, G, D) = \frac{1}{Z_{\underline{\lambda}}} e^{\langle \underline{\lambda}, \# \underline{f}(W, T, G, D) \rangle}$$

One global normalization factor $Z_{\underline{\lambda}} = \sum_{W, T, G, D} e^{\langle \underline{\lambda}, \# \underline{f}(W, T, G, D) \rangle}$ to ensure a proper distribution.

- Maximum likelihood estimation (MLE)

$$\max_{\underline{\lambda}} \sum_{W, D} \tilde{P}(W, D) \sum_{T, G} \log P_{\underline{\lambda}}(W, T, G, D)$$

- Stationary points of MLE \equiv feasible solutions of LME (good)
- Computing $Z_{\underline{\lambda}}$ is intractable (bad)
- Computing right hand side feature expectations is tractable (good)
- Computing left hand side feature expectations is intractable (bad)

Related work

The most relevant work is by Khudanpur and Wu (2000), where they

- used SLM and a word clustering model to extract relevant grammatical and semantic features
- then integrated these features with n -grams by maximum conditional entropy approach
- thus they used undirected *conditional* MRFs

Comments:

- Our composite language model is *a generative model*, all features play important roles in EM iterations to allow maximal order events for WORD-PREDICTOR to appear;

vs.

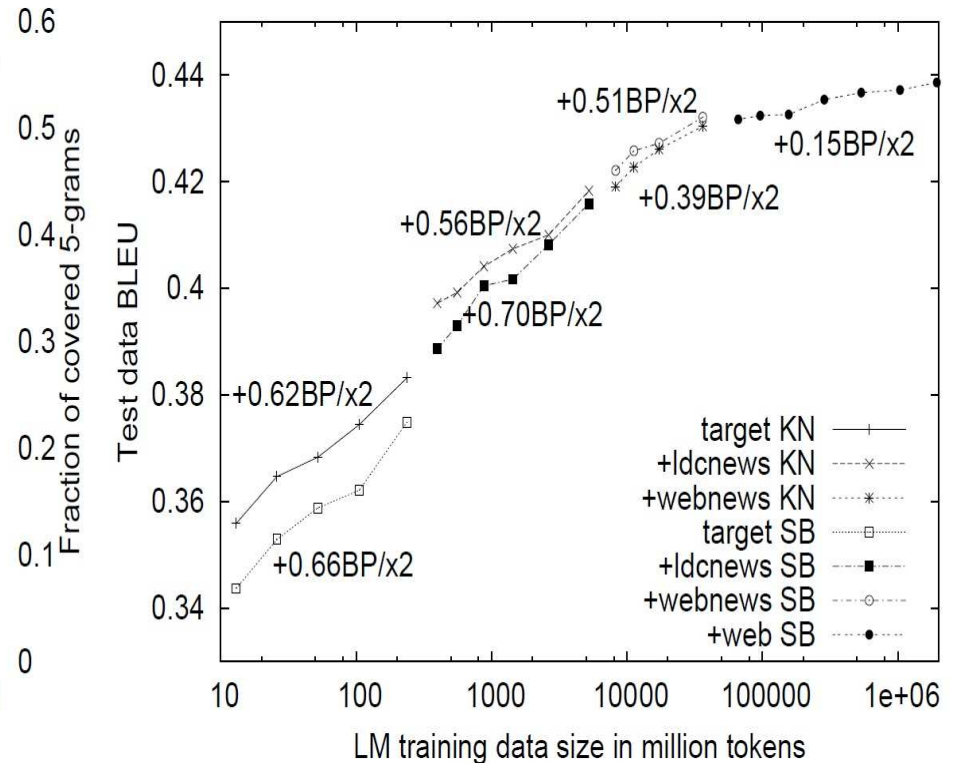
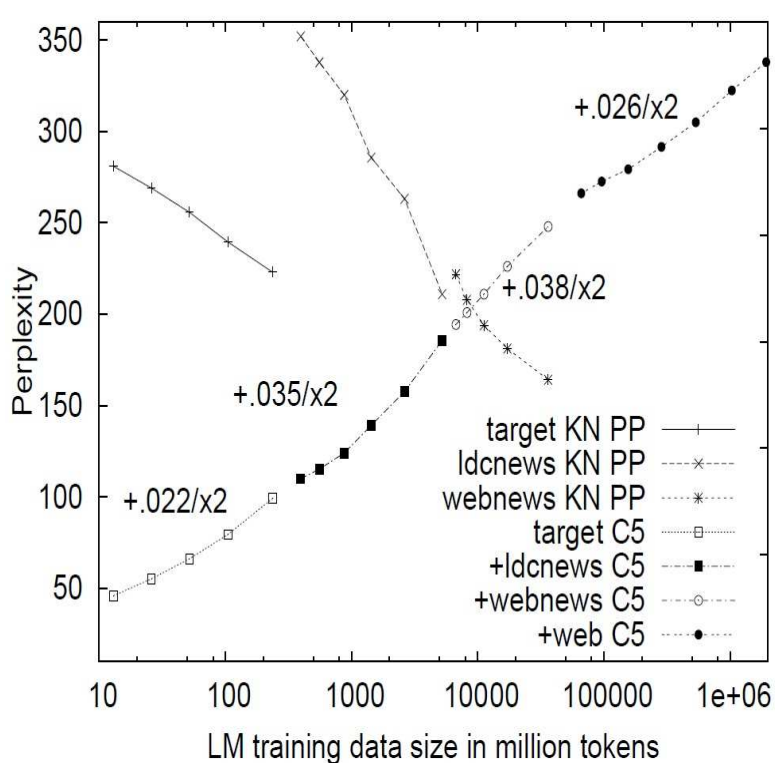
in Khudanpur and Wu (2000), the counts for all events are fixed after feature extraction from SLM and word clustering, which hinders the predictive power of WORD-PREDICTOR

- The training algorithm in Khudanpur and Wu (2000) is computationally expensive, mainly in feature expectation and normalization;

vs.

ours is quite simple which is just expected relative frequency estimates with proper smoothing

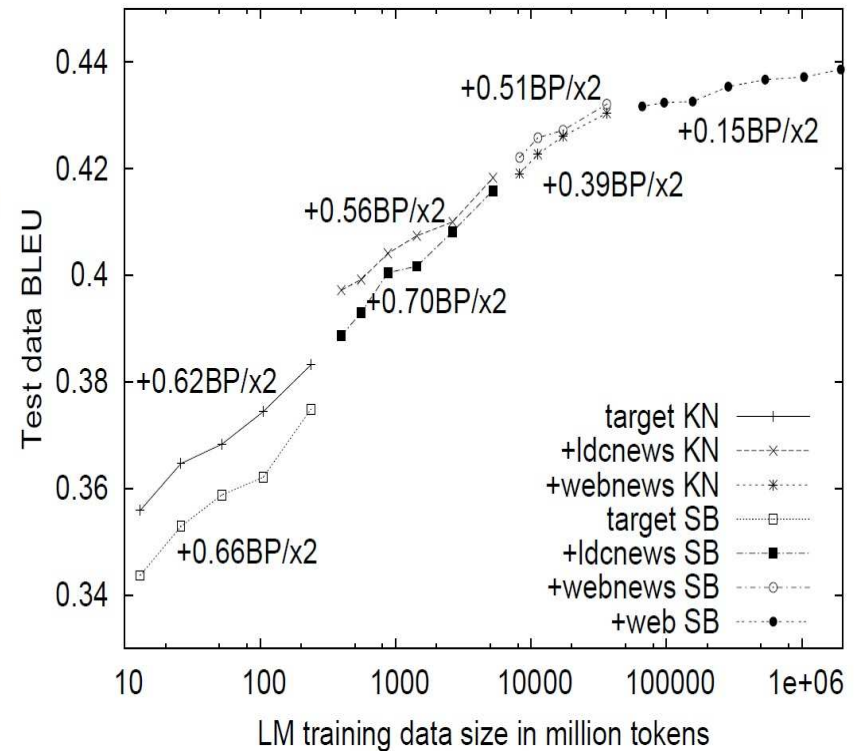
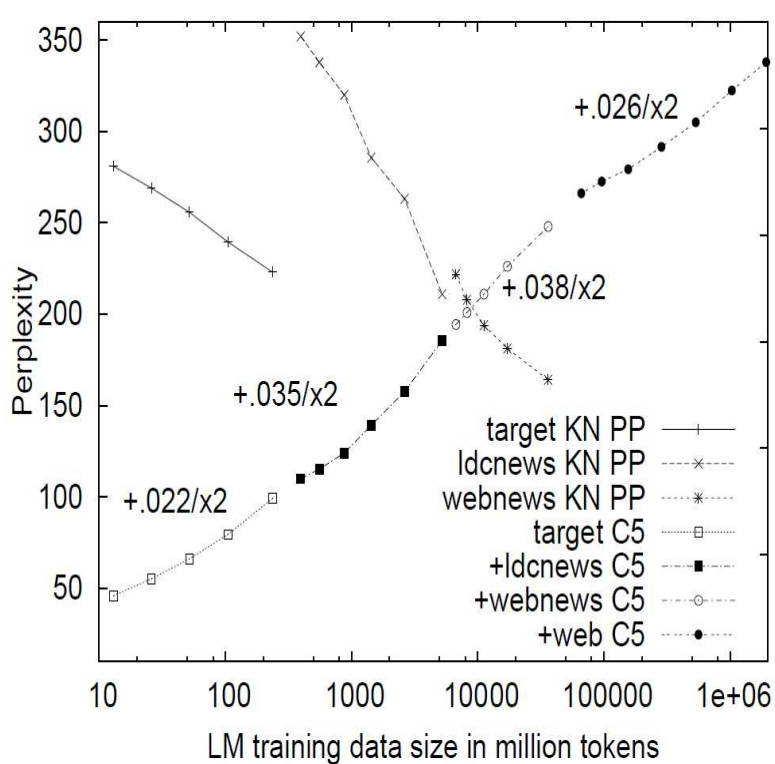
Google's famous results



- Use simple n -gram language model, the more the data, the better the result
 - Double data size \implies **0.5% \uparrow** BLEU score
 - Overall 4.5% BLEU score improvement from billion tokens to trillion tokens

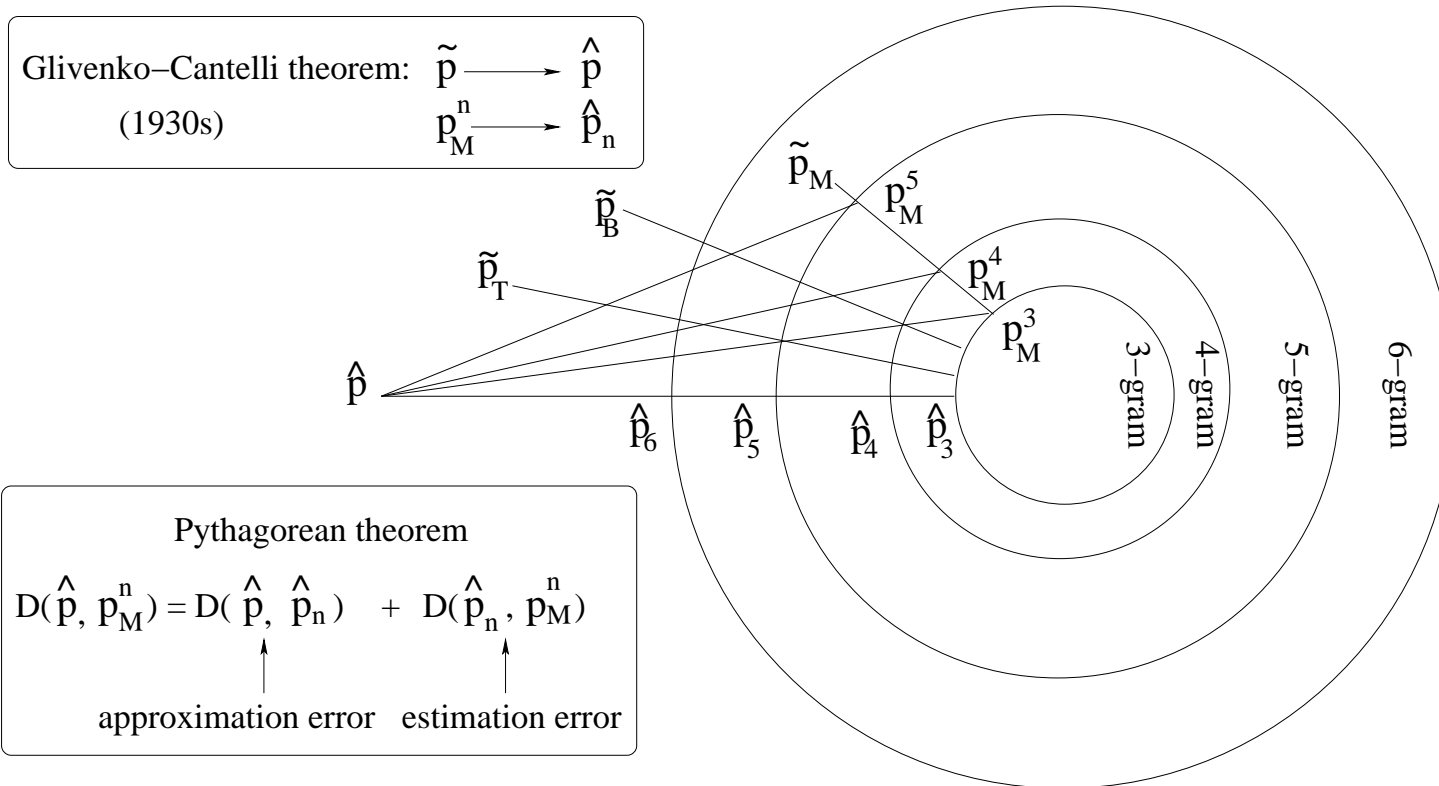
• **Why?**

Google's famous results



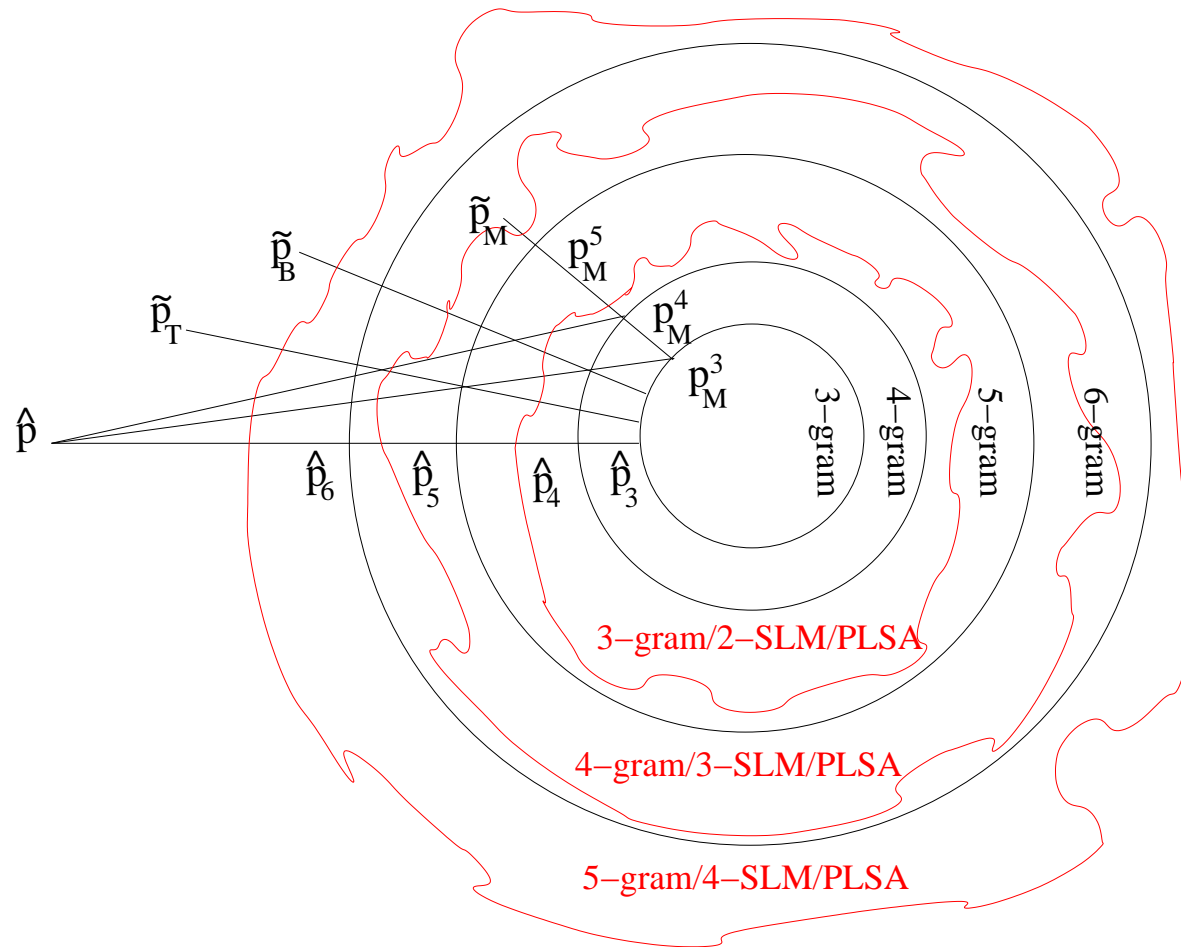
- Use simple n -gram language model, the more the data, the better the result
- **Why?** Excellent n -gram hit ratio on unseen test data

Language modeling: a data rich and feature rich density estimation problem



- \hat{p} : true (but unknown) distribution; $\hat{p}_n, n = 3, 4, 5, 6$: information projection of \hat{p} to n -gram
- \tilde{p} : empirical distribution, in particular, \tilde{p}_M : empirical distribution for million words corpus, \tilde{p}_B : empirical distribution for billion words corpus, \tilde{p}_T : empirical distribution for trillion words corpus;
- $p_M^n, n = 3, 4, 5$: information projection of \tilde{p}_M to n -gram

Language modeling: a data rich and feature rich density estimation problem



- Pythagorean theorem breaks down, nevertheless there is always a trade-off between approximation error and estimation error
- The supply of more data needs to be matched by demand on the model side

Experiments set-up

- Three training corpora taken from the LDC English gigaword corpus.
 1. 44 million tokens
 2. 230 million tokens
 3. 1.3 billion tokens
- One independent test corpus
 1. 320k tokens
- Two check corpora to determine linear interpolation coefficients:
 1. 1.7 million tokens for training corpus 1
 2. 13.7 million tokens for training corpora 2 and 3
- The vocabulary sizes are:
 - word (also WORD-PREDICTOR operation) vocabulary: 60 k, open - all words outside the vocabulary are mapped to the <unk> token, chosen from the most frequently occurred words in 44 millions tokens corpus
 - POS tag (also TAGGER operation) vocabulary: 40, closed
 - non-terminal tag vocabulary: 52, closed
 - CONSTRUCTOR operation vocabulary: 107, closed

Data sets and model initialization

	1.3 billion tokens training corpus
afp	19940512.0003 ~ 19961015.0568
afw	19941111.0001 ~ 19960414.0652
nyt	19940701.0001 ~ 19950131.0483
nyt	19950401.0001 ~ 20040909.0063
xin	19970901.0001 ~ 20041125.0119
	230 million tokens training corpus
afp	19940622.0336 ~ 19961031.0797
apw	19941111.0001 ~ 19960419.0765
nyt	19940701.0001 ~ 19941130.0405
	44 million tokens training corpus
afp	19940601.0001 ~ 19950721.0137
	13.7 million tokens check corpus
nyt	19950201.0001 ~ 19950331.0494
	1.7 million tokens check corpus
afp	19940512.0003 ~ 19940531.0197
	354 k tokens test corpus
cna	20041101.0006 ~ 20041217.0009

- Each model component of WORD-PREDICTOR, TAGGER, and CONSTRUCTOR is initialized from a set of parsed sentences
- We use the "openNLP" software, a maxent parser trained by Upenn treebank, to parse all sentences in 44 and 230 million tokens corpora, and a portion of 1.3 billion tokens corpus, then use them to initialize model parameters.

Baseline n -grams

- Perplexity results using linear interpolation and Kneser-Ney (KN) smoothing

44 M	LINEAR	KN	230 M	LINEAR	KN	1.3 B	LINEAR
$n=3$	262	244	$n=3$	217	195	$n=3$	161
$n=4$	258	235	$n=4$	200	183	$n=4$	141
$n=5$	260	235	$n=5$	201	183	$n=5$	138

- Statistics about the number of types of n -grams

	$n=3$	$n=4$	$n=5$
44 M	14,302,355	23,833,023	29,068,173
230 M	51,115,539	94,617,433	120,978,281
1.3 B	224,767,319	481,645,099	660,599,586

m-SLMs

- SLMs' perplexity results using linear interpolation

44 M	LINEAR	230 M	LINEAR	1.3 B	LINEAR
$m=2$	279				
		$m=3$	190		
				$m=4$	137

- Counts of the types in the predictor of the *m*-SLMs.

	$m=2$	$m=3$	$m=4$
44 M	189,002,525	269,685,833	318,174,025
230 M	267,507,672	1,154,020,346	1,417,977,184
1.3 B	946,683,807	1,342,323,444	1,849,882,215

For 230 million and 1.3 billion tokens corpora, fractional expected counts that are less than a threshold are pruned to significantly reduce the number of *m*-SLM predictor's types by 70%.

n -gram/PLSA

- Fix the size of topics in PLSA to be 200
- Perplexity (ppl) results and time consumption when different numbers of most likely topics are kept for each document in PLSA

CORPUS	n	# OF TOPICS	PPL	TIME (HOURS)	# OF SERVERS	# OF CLIENTS	# OF TYPES OF $w_{-n+1}^{-1}wg$
44M	3	5	196	0.5	40	100	120.1M
	3	10	194	1.0	40	100	218.6M
	3	20	190	2.7	80	100	537.8M
	3	50	189	6.3	80	100	1.123B
	3	100	189	11.2	80	100	1.616B
	3	200	188	19.3	80	100	2.280B
230M	4	5	146	25.6	280	100	0.681B
1.3B	5	2	111	26.5	400	100	1.790B
	5	5	102	75.0	400	100	4.391B

- Unpruned 5 topics in general account for 70% probability in $p(g|d)$
- Prune 200 topics to 5 in the rest of experiments

Perplexity results for various LMs

language model	44 M n=3, m=2	230 M n=4, m=3	1.3 B n=5, m=4
baseline n-gram (linear)	262	200	138
n-gram (KN)	244 6.9%	183 8.5%	--- ---
m-SLM	279 -6.5%	190 5.0%	137 0.0%
PLSA	825 -214.9%	812 -306.0%	773 -460.0%
n-gram+m-SLM	247 5.7%	184 8.0%	129 6.5%
n-gram+PLSA	235 10.3%	179 10.5%	128 7.2%
n-gram+m-SLM+PLSA	222 15.3%	175 12.5%	123 10.9%
n-gram/m-SLM	243 7.3%	171 14.5%	(125) 9.4%
n-gram/PLSA	196 25.2%	146 27.0%	102 26.1%
m-SLM/PLSA	198 24.4%	140 30.0%	(103) 25.4%
n-gram/PLSA+m-SLM/PLSA	183 30.2%	140 30.0%	(93) 32.6%
n-gram/m-SLM+m-SLM/PLSA	183 30.2%	139 30.5%	(94) 31.9%
n-gram/m-SLM+n-gram/PLSA	184 29.8%	137 31.5%	(91) 34.1%
n-gram/m-SLM+n-gram/PLSA +m-SLM/PLSA	180 31.3%	130 35.0%	--- ---
n-gram/m-SLM/PLSA	176 32.8%	--- ---	--- ---

+ denotes linear combination, / denotes composite model, topics are pruned from 200 to 5

Model size is a big issue!

- Counts of the types in predictor of 5-gram/PLSA, 5-gram/2-SLM (or 2-gram/4-SLM), and 4-SLM/PLSA when trained on 1.3b corpus.

COMPOSITE MODEL	TYPES OF	# OF TYPES	# OF SERVERS	# OF CLIENTS
5-GRAM/PLSA	$w_{-4}^{-1}wg$	4.39 B	400	100
5-GRAM/2-SLM OR 2-GRAM/4-SLM	$w_{-4}^{-1}wh_{-2}^{-1}$ OR $w_{-1}wh_{-4}^{-1}$	2.01 B	240	100
4-SLM/PLSA	$wh_{-4}^{-1}g$	4.88 B	400	100

Heavy pruning: purge fractional expected counts that are less than a threshold to reduce the number of predictor's types by 85%.

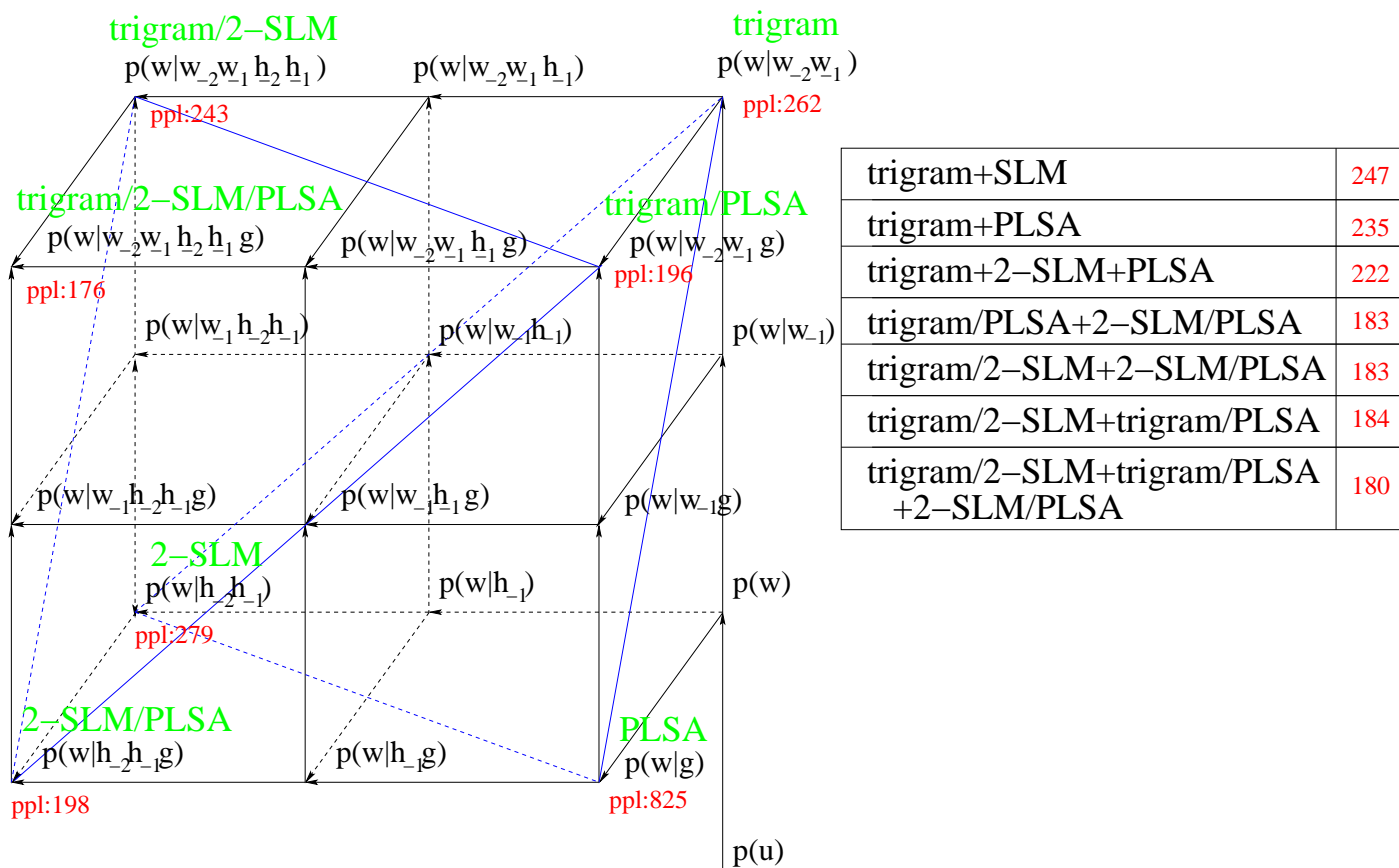
Using different testing methods

- Perplexity results

language model	44 M n=3, m=2	230 M n=4, m=3	1.3 B n=5, m=4
baseline n-gram (linear)	262	200	138
n-gram/PLSA ¹	202 22.9%	150 25.0%	107 22.5%
n-gram/m-SLM+n-gram/PLSA ¹	192 26.7 %	142 29.0%	(97) 29.1%
n-gram/PLSA ²	196 25.2%	146 27.0%	102 26.1%
n-gram/m-SLM+n-gram/PLSA ²	184 29.8%	137 31.5%	(91) 34.1%
n-gram/PLSA ³	201 23.3%	148 26.0%	104 24.6%
n-gram/m-SLM+n-gram/PLSA ³	189 27.9%	140 30.0%	(92) 33.3%

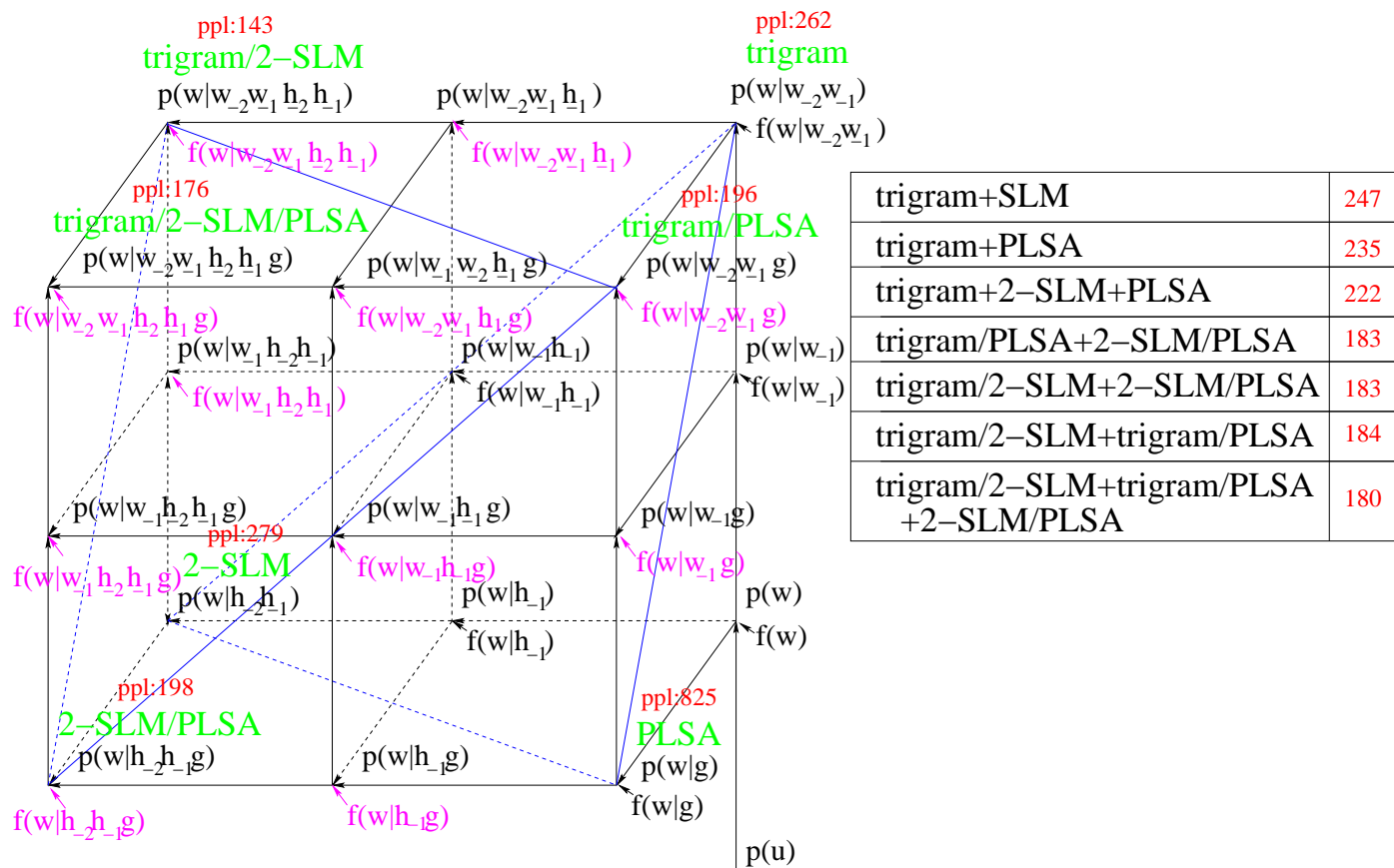
1. one step online EM
2. online EM with fixed learning rate $\gamma = 0.2$
3. batch EM

Why composite language model powerful?



- Composite model predicts word based on a hierarchically organized committee and the relative frequency estimate it extracts.
- Linear combination is restricted to a simplex.

Why composite language model powerful?



- Composite model encodes valuable relative frequency estimates explicitly or implicitly in training corpus
- The weights used in simple linear combination are context independent, thus more restricted
- The WORD-PREDICTOR of the composite model oversees all available information to make the most powerful prediction.

An example of sentence probability

- Training corpus: 1.3 billion tokens
- Test document: <XIN_ENG_20041126_0168.story>
 - a. **5-gram**: perplexity = 97
 - b. **5-gram+PLSA**: perplexity = 93
 - c. **5-gram+4-SLM+PLSA**: perplexity = 83
 - d. **5-gram/PLSA**: perplexity = 71
 - e. **5-gram/PLSA+4-SLM/PLSA**: perplexity = 64

- The first four sentences

<S> *cpc initiates education campaign to strengthen members ' wavering convictions* </S> <S>
by zhao lei </S> <S> *beijing nov. 'nmbx xinhua the communist party of china cpc has decided to*
launch a mass internal educational campaign from january next year to prevent its members from
wavering in their convictions </S> <S> *the decision aiming to keep the nature of the party*
members intact was made at the meeting of the political bureau of the cpc central committee on this
oct. 'nmbx the cpc 's top power organ </S>

An example of sentence probability (continue)

the decision aiming to keep the nature of the

- a. -2.00317 -5.99654 -14.9793 -0.852055 -4.68269 -1.49193 -9.84554 -0.526566 -0.671103
- b. -2.05502 -6.08843 -13.2655 -0.950885 -4.78594 -1.56474 -9.81423 -0.6258 -0.761926
- c. -2.05416 -6.07556 -13.3486 -0.871798 -4.69523 -1.57311 -9.99731 -0.897362 -0.829652
- d. -1.72696 -5.65359 -14.2013 -0.99068 -5.43248 -1.65002 -7.6 -0.612751 -0.755122
- e. -1.80167 -5.73861 -14.5548 -0.893825 -5.05692 -1.60568 -7.92909 -0.751419 -0.755122

party members intact was made at the meeting of

- a. -6.52337 -5.93013 -14.992 -5.5802 -5.91863 -3.47798 -1.0155 -3.77026 -3.11882
- b. -6.48382 -6.00924 -13.8132 -5.57218 -5.98123 -3.56856 -1.1003 -3.87003 -3.14354
- c. -6.48696 -5.81026 -8.11845 -3.04638 -2.21191 -2.80501 -1.12155 -3.85156 -2.3551
- d. -3.46383 -5.03999 -15.242 -5.27819 -4.73655 -3.03394 -0.69443 -3.23709 -3.40986
- e. -3.80075 -5.16911 -8.52597 -3.38567 -2.54778 -2.74127 -0.790644 -3.36195 -2.64652

the political bureau of the cpc central committee

- a. -0.619712 -5.91994 -1.36559 -0.17816 -0.127888 -1.55966 -0.282506 -0.110539
- b. -0.710967 -5.96757 -1.47083 -0.278998 -0.313708 -1.66454 -0.387673 -0.215632
- c. -0.636643 -6.0839 -1.43513 -0.6519 -0.634246 -2.10113 -0.504145 -0.216812
- d. -0.475928 -4.13345 -0.527685 -0.226433 -0.204276 -1.55903 -0.379722 -0.147238
- e. -0.475442 -4.43649 -0.702968 -0.427385 -0.388118 -1.79781 -0.42272 -0.136813

on this oct. 'nmb the cpc 's top power

- a. -4.33953 -7.02792 -10.7495 -0.0380615 -3.87067 -9.93617 -3.54366 -4.19702 -7.6261
- b. -4.37441 -6.88172 -10.6397 -0.141938 -3.65821 -8.81816 -3.60823 -4.29886 -7.64586
- c. -3.57338 -6.86285 -10.9656 -0.131813 -3.8662 -8.85551 -3.42688 -4.28615 -7.82392
- d. -4.61674 -6.49064 -13.0595 -0.255452 -3.73302 -5.55244 -3.60481 -3.97708 -7.85289
- e. -3.85647 -6.61406 -12.5666 -0.178075 -3.92356 -5.90511 -3.46416 -4.03158 -7.91198

organ </s>

- a. -5.97561 -2.62716
- b. -6.08022 -2.67444

Why composite language model powerful?

- Statistics when n -gram is the same as SLM's WORD-PREDICTOR headword

CORPUS	$w_{-2}^{-1} = h_{-2}^{-1}$	$w_{-3}^{-1} = h_{-3}^{-1}$	$w_{-4}^{-1} = h_{-4}^{-1}$
44 M	57%	46%	38%
230 M	59%	46%	38%
1.3 B	55%	48%	43%

- n -gram alone is not viable to achieve similar affect even using web scale data
- The directed MRF paradigm effectively synergizes n -gram, m -SLM, and PLSA in a complementary, supplementary, and coherent way to form a powerful language model for word prediction of natural language.

BLEU score results for re-ranking N -best list in machine translation

- Same 1000-best list used by Zhang et al. 2006.
- Generated on 919 sentences from the MT03 Chinese-English evaluation set by Hiero (Chiang 2005). Its decoder uses a trigram with modified Kneser-Ney smoothing that trained on a 200 million words corpus
- Each translation has 11 features and language model is one of them.
- We substitute 5-gram/2-SLM+2-gram/4-SLM+5-gram/PLSA language model trained by 1.3 billion word corpus and use MERT (Och 2003) to optimize the BLEU score and re-rank the 1000-best list. 10-fold cross-validation BLEU score results are

System model	mean (%)	variance
Baseline	31.75	0.0431
5-gram	32.53	0.0484
(5-gram/SLM)	32.87	0.0502
5-gram/PLSA ¹	33.01	0.0412
(5-gram/SLM/PLSA ¹)	33.32	0.0431

Translations' "readability" results

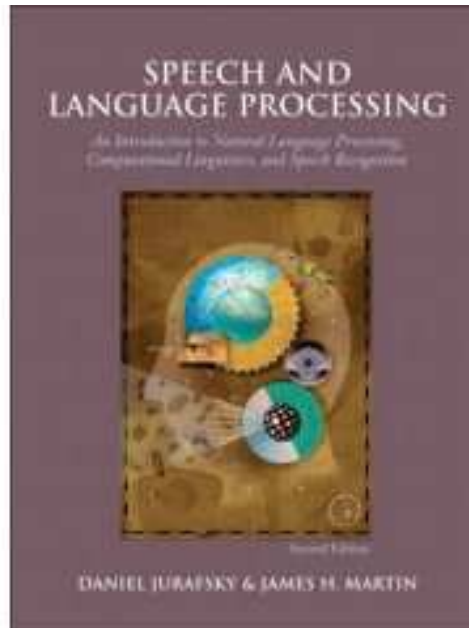
- Similar to the study conducted by Charniak et al. 2003
- The translations are sorted into 4 groups: good/bad syntax crossed with good/bad meaning by human judges

System model	P	S	G	W
Baseline	95	398	20	406
5-gram	122	406	24	367
(5-gram/SLM/PLSA ¹)	153	428	33	305

P: perfect S: only semantically correct G: only grammatically correct W: wrong

Is textbook right?

On page 482, “We said earlier that statistical parsers can take advantage of longer-distance information than n -grams, which suggests that they might do a better job at language modeling/word prediction. It turns out that **if we have a very large amount of training data, a 4-gram or 5-gram is nonetheless still the best way to do language modeling.**”



Jurafsky and Martin, *Speech and Language Processing*, 2nd Edition, Prentice Hall, 2008

About data

- “There is no data like more data” (Mercer at Arden House, 1985 from Jelinek 2004)
- “More data is more important than better algorithms” (Brill’s opinion from Jelinek 2004)
- But this doesn’t mean
 - Simple algorithms are better than sophisticated algorithms
- They should be compared at the same ground: using the same size of training data
- Training a composite model on trillion tokens corpus is feasible, affordable, and cheap in the era of cloud computing

Better MT evaluation metric?

- BLEU: n -gram based MT evaluation metric
- Closer agreement may be possible by incorporating syntactic structure and semantic information into the BLEU score evaluation
 - semantically similar words like “insure” and “ensure” in the example of BLEU paper (Papineni et al. 02) should be substituted in the formula
 - assign a weight to measure the goodness of syntactic structure

This modification will lead to a better metric and such information can be provided by our composite language models

Conclusion

- We have implemented a composite language model that
 - integrates n-gram, SLM and PLSA under the directed MRF paradigm
 - trained using corpora up to a billion tokens
 - stored on a supercomputer with up to 1000 processors
- The large scale distributed composite language model
 - gives significant perplexity reduction over n-grams
 - achieves significantly better translation quality measured by the BLEU score and “readability” of translations when applied to the task of reranking the N-best list in statistical machine translation
- As far as we know, this is **the first work**
 - to build a complex large scale distributed language model with a principled approach that
 - simultaneously exploits syntactic, semantic and lexical regularities
 - more powerful than n-grams trained on a very large corpus

Future work

- Integrate more advanced topic language models such as LDA, CTM, DTM (Blei et al 2003, 2006, 2007)
- Resort to hierarchical non-parametric Bayesian model (Teh 2006) for smoothing fractional counts due to latent variables in Kneser-Ney's sense in a principled manner
- Use Blue waters that has 300k cores to handle trillion tokens corpus
- Put it into
 - a phrased-based machine translation decoder (Koehn et al. 2003) that produces a lattice of alternative translations/transcriptions
 - a syntax-based decoder (Chiang 2007) that produces a forest of alternatives (such integration would, in the exact case, reside in an extremely difficult complexity class, probably PSPACE-complete)

We expect:

- 45% ~ 50% perplexity reduction
- 3% ~ 5% BLEU score improvement \implies 64 ~ 1024 times training data

Acknowledgements

- We would like to dedicate this work to the memory of Fred Jelinek, who passed away while we were finalizing this manuscript
- Work done jointly with Wright State University students Ming Tan, Wenli Zhou and Lei Zheng
- This research is supported by NSF under grant IIS:RI-small 0812483, a Google research award and AFOSR under grant FA9550-10-1-0335
- This work is supported in part by an allocation of computing time from the Ohio Supercomputer Center
- We would like to thank
 - Ciprian Chelba for providing the SLM code, answering many questions regarding SLM and consulting on various aspects of the work
 - Ying Zhang and Philip Resnik for providing the 1000-best list from Hiero for re-ranking in machine translation
 - Peng Xu for suggesting to look at conditional probability of a word given its document history to make the perplexity result much convincing.