中国科学院自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

Language Model Adaptation: An Overview

Institute of Automation, Chines Academy of Sciences, CASIA
National Laboratory of Pattern Recognition, NLPR
Intelligent Interaction Group
Ye Bai
March 31, 2018

Outline

- Review of Language Models
- Adaptation on n-gram based models
- Adaptation on neural network based models
- Summary

Review of Language Models

- Given a finite set Σ, what is the probability of a string of length T:

$$P(w_1, \dots, w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_T|w_1 \dots w_{T-1}),$$

where $\forall w_t \in \Sigma$.

Data sparsity

- Because exponential explosion problem, data sparsity always exists in language modeling.

- Simplification based on Markov property.

Markov property in LMs

- In n-gram and neural network language models (NNLMs), language modeling is simplified as a (finite state) Markov chain.

$$
\begin{aligned}
&P(w_1, \ldots, w_T) \\
&= P(w_1)P(w_2|w_1)\ldots P(w_t|w_{t-n+1}\ldots w_{t-1})\ldots P(w_T|w_{T-n+1}\ldots w_{T-1}),
\end{aligned}
$$

Markov property in LMs

- In recurrent neural network language models (NNLMs), language modeling is simplified as a Markov process.

$$P(w_T|w_1, ..., w_T) = f(w_T, h_T)$$

Back-off ngram language models

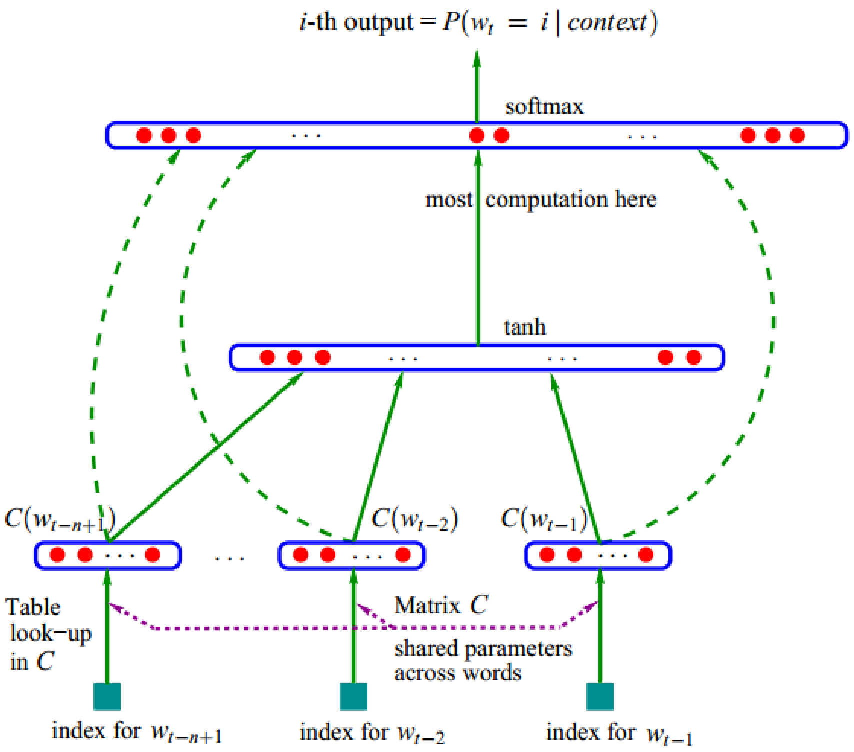
- Commonly used in ASR decoders.

$$P(w|h) = \begin{cases} \widehat{P(w|h)} & \text{,if } count(hw) > 0 \\ \alpha(h)\widehat{P(w|h')} & \text{,otherwise} \end{cases}$$

  - Katz
  - Kneser-Ney
  - …

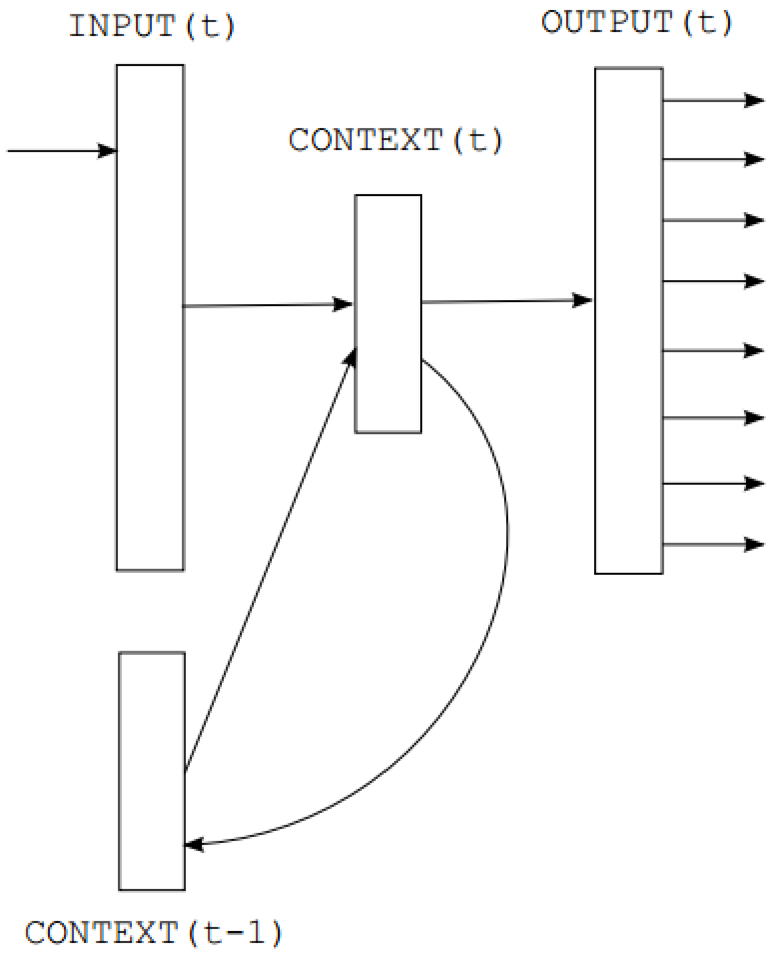
Neural network language models

- Use a continues function to approximate a discrete probability distribution.
- Do not need smoothing.



Recurrent neural network language models

- Using a hidden state, say, output of hidden layer, to record history information of a sentence.



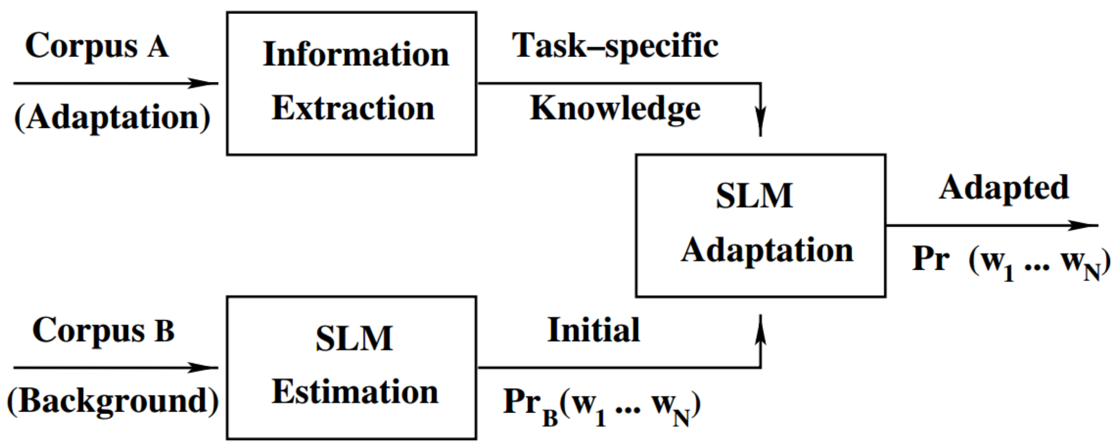
Perplexity

$$LP = -\frac{1}{T}\log_2 \hat{P}r(W_1^T)$$

$$PP = 2^{LP}.$$

Language Model Adaptation

- For a specific domain, it's difficult to collect text data, e.g., finance, navigation.
- Using another domain data to adapt models.



Outline

- Review of Language Models
- Adaptation on n-gram based models
- Adaptation on neural network based models
- Summary

Adaptation on n-gram models

- Model Composition
  - Linear interpolation
  - MAP
  - Cache model

Linear interpolation

- Given l models, its convex combination

$$Pr(w;\lambda) = \sum_{i=1}^{l} \lambda_i Pr_i(w)$$

  denotes an adapted model.
- Use EM algorithm to estimate parameters

$$\lambda_i^{(n+1)} = \sum_{t=1}^{m} \frac{\lambda_i^{(n)} Pr_i(w_t)}{\sum_{j=1}^{l} \lambda_j^{(n)} Pr_j(w_t)}.$$

Linear interpolation

- Fill-up technique

$$\Pr(w_q|h_q) = \begin{cases} \Pr_A(w_q|h_q) & \text{if } C_A(h_q w_q) \geqslant T; \\ \beta \Pr_B(w_q|h_q) & \text{otherwise,} \end{cases}$$

Maximum a-posteriori

- Maximum is a generalization of linear interpolation.

- It introduces a priori to adjust estimated probability distribution.

Review n-gram

- Given a fixed history h, a sample set $S = hw_{h_1}, \ldots, hw_{h_m}$ can be seen as a realization of m i.i.d random variables, drawn according to the law represented by $Pr(w \mid h)$.
- N-gram model this distribution as multinomial distribution:

$$Pr(S;\theta) = \frac{m!}{\prod_{w \in V} c(w)!} \prod_{w \in V} \theta_w^{c(w)} \quad \text{where} \quad c(w) = \sum_{i=1}^{m} \delta(w_i = w)$$

  and its maximum likelihood estimator is the word count.
- **Overfitting.**

Maximum a-posteriori

- Bayesian formulation:

$$Pr(\theta \mid S) = \frac{Pr(S \mid \theta)\ Pr(\theta)}{Pr(S)}$$

- MAP criterion

$$\theta^{MAP} = arg \max_{\theta \in \Theta} Pr(\theta \mid S) = arg \max_{\theta \in \Theta} Pr(S \mid \theta) Pr(\theta).$$

- The conjugate prior of the multinomial distribution is Dirichlet distribution

Maximum a-posteriori

- The conjugate prior of the multinomial distribution is Dirichlet distribution. The resulting MAP estimate is

$$\theta^{MAP} = arg \max_{\theta \in \Theta} \prod_{w \in V} \theta_w^{c(w)+c'(w)} = [\frac{c(w)+c'(w)}{m+m'}]_{w \in V}$$

- For two samples:

$$Pr^{MAP}(w) = (\frac{m}{m+m'})f(w) + (\frac{m'}{m+m'})f'(w)$$

Cache models

- Motivation: a word which is observed in previous data may be more likely to appear in the future.
- Use last N words to train a LM to adapt.
- A special case widely used for within-domain adaptation.

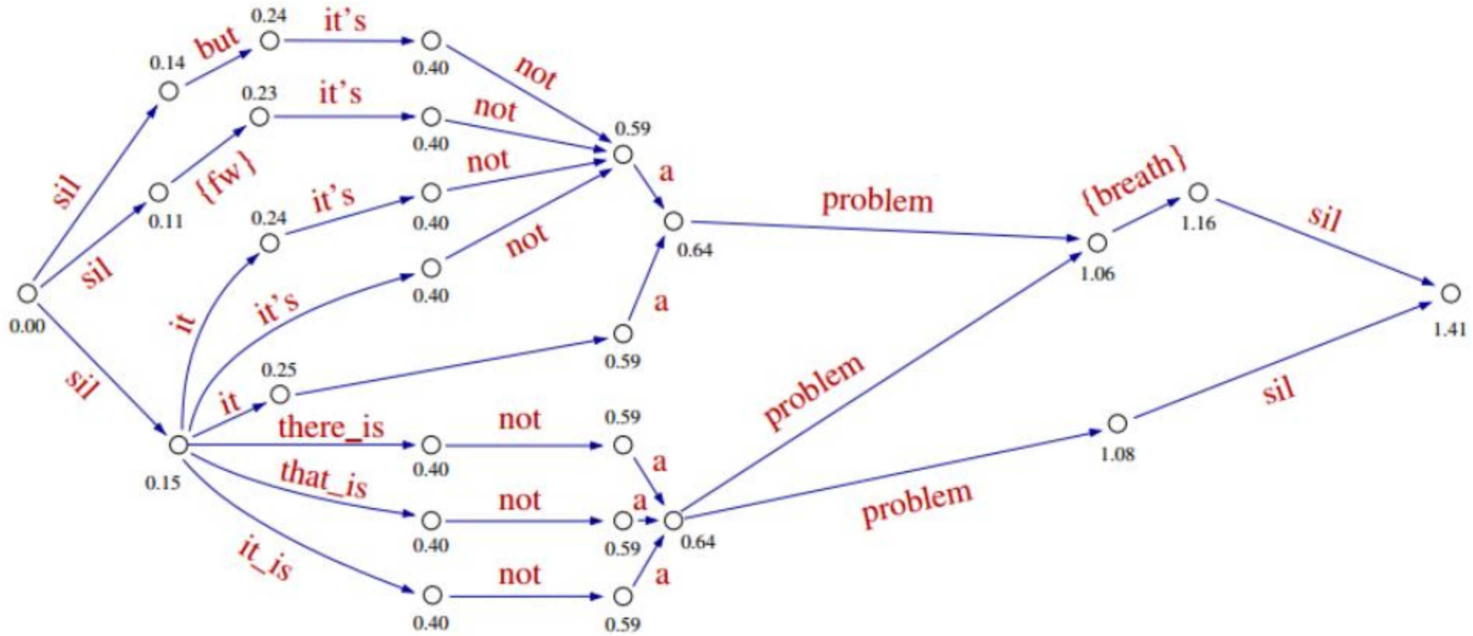
Learning N-gram Language Models from Uncertain Data

Learning N-gram Language Models from Uncertain Data

Vitaly Kuznetsov[1,2], *Hank Liao*[2], *Mehryar Mohri*[1,2], *Michael Riley*[2], *Brian Roark*[2]
[1]Courant Institute, New York University
[2]Google, Inc.
{vitalyk,hankliao,mohri,riley,roark}@google.com

- Uncertain data: semi-supervised, inaccurate labelled data, e.g., lattices generated from an existing ASR system.

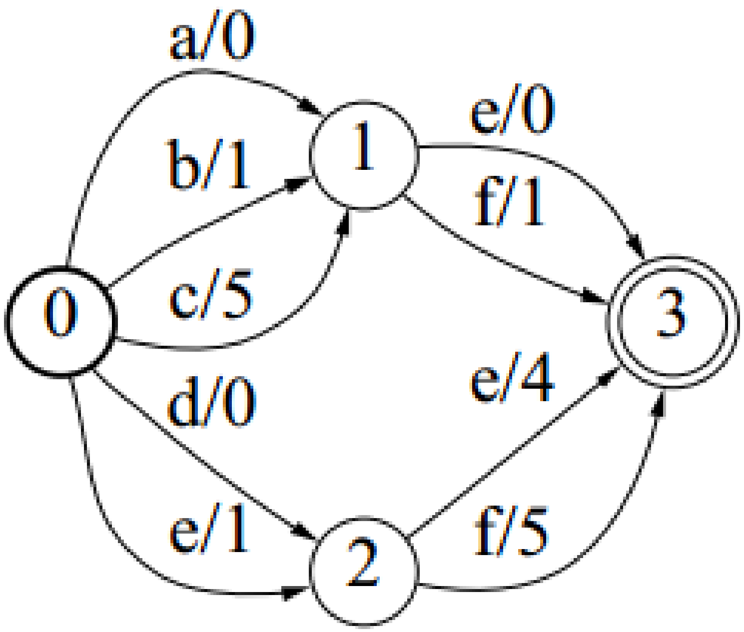
Lattice



- A lattice is a weighted finite state transducer. Input/output labels on edges are words, and weight on edges are scores.

Stochastic WFSTs

- Proposition:
  A WFST is stochastic, if and only if it is deterministic and the sum of the weights of the transitions leaving any state is 1.
- A WFST can be viewed as a probability distribution over all strings Σ*.



Learning N-gram Language Models from Uncertain Data

- Leverage the output of a speech recognition system to adapt an existing language model trained on a source domain to a target domain for which no human transcription is available.

- No exclusive text data. Use transcripts generated from an ASR system.

Learning N-gram Language Models from Uncertain Data

- Fractional count language models：Histogram

$$\widehat{\Pr}[\mathbf{w}] = \mathop{\mathbb{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{w})] = \sum \Pr_{S\sim\mathcal{L}}[S]\widehat{p}_S(\mathbf{w})$$

$$\mathop{\mathbb{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{w})] = \sum_{k=0}^{\infty} q_{\mathcal{L}}(k,\mathbf{w}) f(k),$$

$$q_{\mathcal{L}}(k,\mathbf{w}) = \sum_S \Pr_{S\sim\mathcal{L}}[c_S(\mathbf{w}) = k]$$

- Katz back-off model

$$\widehat{\Pr}(w|\mathbf{h}) = \begin{cases} \frac{1}{\lambda_{\mathbf{h},\mathcal{L}}}\left[\lambda_{\mathbf{h}w,\mathcal{L}} + \sum_{k=1}^{K} q_{\mathcal{L}}(k,w)k\left(\bar{d}_k - 1\right)\right] & \\ \quad +q(0,\mathbf{h}w)\beta_h(\mathcal{L})\widehat{\Pr}(w|\mathbf{h}') & \text{if } \mathbf{h}w \in \mathcal{L}, \\ \beta_{\mathbf{h}}(\mathcal{L})\widehat{\Pr}(w|\mathbf{h}'), & \text{otherwise,} \end{cases}$$

Learning N-gram Language Models from Uncertain Data

- Computing the Histograms
- Assume that the sample consists of two lattices T and U such that $\Pr_{S\sim\mathcal{L}}[S] = \Pr_{T\sim\mathcal{T}}[T] \Pr_{U\sim\mathcal{U}}[U]$.
- Then it can be decomposed as follows:

$$
\begin{aligned}
& q_{\mathcal{L}}(k, \mathbf{w}) \\
& = \sum_{S} \Pr_{S\sim\mathcal{L}}[c_S(\mathbf{w}) = k] \\
& = \sum_{T} \sum_{U} \sum_{j=0}^{k} \Pr_{T\sim\mathcal{T}}[c_T(\mathbf{w}) = j] \Pr_{U\sim\mathcal{U}}[c_U(\mathbf{w}) = k - j] \\
& = \sum_{j=0}^{k} \left(\sum_{T} \Pr_{T\sim\mathcal{T}}[c_T(\mathbf{w}) = j] \right) \left(\sum_{U} \Pr_{U\sim\mathcal{U}}[c_U(\mathbf{w}) = k - j] \right) \\
& = \sum_{j=0}^{k} q_{\mathcal{T}}(j, \mathbf{w}) q_{\mathcal{U}}(k - j, \mathbf{w}). \qquad (9)
\end{aligned}
$$

Experiments

Google Preferred Lineup	Tokens ×1000	Word Error Rate (WER) Baseline	Adapted 1-best	Adapted Lattice	Δ
Video Games (dev set)	23.8	41.2	40.3	39.5	1.6
Anime & Teen Animation	4.3	29.9	30.3	29.7	0.2
Beauty & Fashion	37.3	29.6	29.1	28.0	1.6
Cars, Trucks & Racing	5.7	21.6	22.1	21.6	0.0
Comedy	9.3	55.3	54.9	54.9	0.4
Entertainment & Pop Culture	27.8	39.3	39.4	38.9	0.4
Food & Recipes	11.4	42.6	43.0	41.7	0.9
News	12.3	27.4	27.3	26.7	0.7
Parenting & Children Interest	11.7	38.0	38.4	37.1	0.9
Science & Education	15.9	22.0	22.4	21.5	0.5
Sports	6.7	47.3	47.9	47.3	0.0
Technology	23.7	23.1	23.1	22.3	0.8
Workouts, Weightlifting & Wellness	13.2	31.0	30.5	29.1	1.8
All test lineups	179.2	28.8	28.8	28.0	0.8

Outline

- Review of Language Models
- Adaptation on n-gram based models
- Adaptation on neural network based models
- Summary

Adaptation on neural network based models

- Most work is about fine-tuning: pre-train a model on background data, and then fine-tune on specific domain

- A comparative study

Approaches for Neural-Network Language Model Adaptation

Min Ma[1,*], *Michael Nirschl*[2], *Fadi Biadsy*[2], *Shankar Kumar*[2]

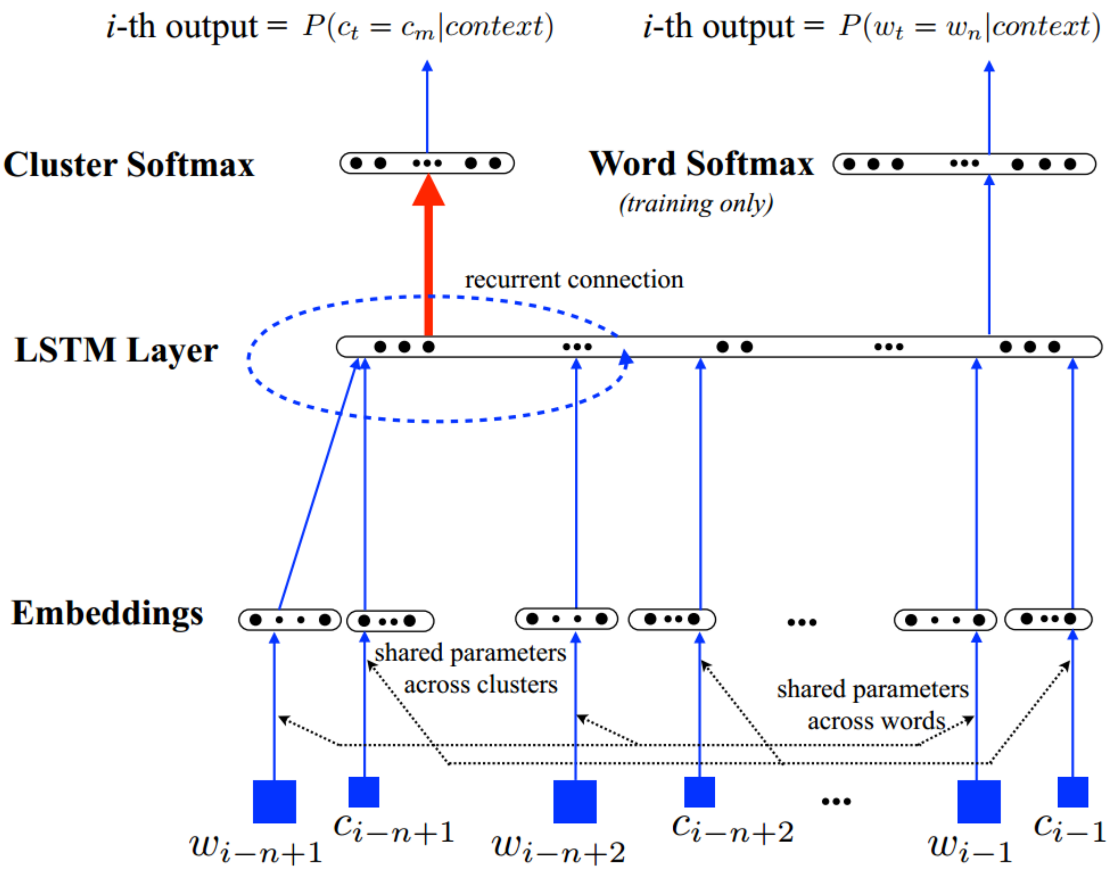
[1]Graduate Center, The City University of New York, NY, USA
[2]Google Inc, New York, NY, USA
mma@gradcenter.cuny.edu, {mnirschl, biadsy, shankarkumar}@google.com

Data

- typed texts: 29 billion sentences from Google Search, Google Maps and crawled web documents.

- spoken texts: 2.6 million sentences from speech transcripts.

Adaptation on neural network based models



Three adaptation schemes

- Scheme I： fine-tune softmax layer
- Observation: transfer learning becomes easier and more effective with high-level abstract features.
- Fine-tuning all parameters may cause overfitting.

Table 1: *Cluster perplexity on development set of pre-trained model ("pre"), adapted model ("post") and relative changes.*

Model and Adaptation Strategy	pre	post	Δ PPL
MaxEnt Baseline LM	40	29	-27.5%
DNN LM using Scheme I	46	35	-23.9%
+ wordSF		35	-23.9%
+ wordSF, + DNN		43	-6.5%
LSTM LM using Scheme I	46	34	**-26.1%**
+ wordSF		35	-23.9%

Three adaptation schemes

- Scheme II：add an adaptation layer
- Add the adaptation layer between the hidden layer and softmax.
- This approach is conceptually similar to the linear hidden layer for acoustic model adaptation.

Table 2: *Cluster perplexity on development set of pre-trained model ("pre"), adapted model ("post") and relative changes.*

Model and Adaptation Strategy	pre	post	Δ PPL
DNN LM using Scheme II	46	34	-26.1%
+ wordSF		34	-26.1%
LSTM LM using Scheme II	46	34	-26.1%
+ wordSF		34	-26.1%

Three adaptation schemes

- Scheme III： add DNN adaptation layer in both pre-training and adaptation.

Model and Adaptation Strategy	pre	post	Δ PPL
LSTM LM using Scheme III	43	**30**	**-30.2%**
+ wordSF		31	-27.9%
+ wordSF, + LSTM		198	+360.5%

Experiments on ASR

Language Model	w_{NN}	Rel Δ (%)	WER(%)
non-adapted MaxEnt	0.0	+5.5	7.1
adapted MaxEnt baseline	0.0	0.0	**6.7**
non-adapted DNN (I)	0.5	+1.2	6.8
non-adapted DNN (I)	1.0	+6.0	7.1
adapted DNN (I)	0.5	0.0	6.7
adapted DNN (I)	1.0	+1.5	6.8
non-adapted LSTM (I)	0.5	+1.4	6.8
non-adapted LSTM (I)	1.0	+7.0	7.2
adapted LSTM (I)	0.5	0.0	6.7
adapted LSTM (I)	1.0	+3.2	6.9
non-adapted LSTM (III)	0.5	0.0	6.7
non-adapted LSTM (III)	1.0	+5.8	7.1
adapted LSTM (III)	0.5	**-2.3**	**6.6**
+wordSF	0.5	-2.0	6.6
adapted LSTM (III)	1.0	+1.7	6.8
+wordSF	1.0	0.0	6.7

Outline

- Review of Language Models
- Adaptation on n-gram based models
- Adaptation on neural network based models
- Summary

Summary

- N-gram adaptation
  – Interpolation
  – Uncertain data
- NNLM
  – Fine-tuning
  – Multitask leanring

Thank you!