Language Modeling: State of the Art

Zhijian Ou[1] Bin Wang[2]

[1]Speech Processing and Machine Intelligence (SPMI) Lab,
Tsinghua University, Beijing, China.

[2]Liulishuo Information Technology Co., Ltd.
Shanghai, China.





Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

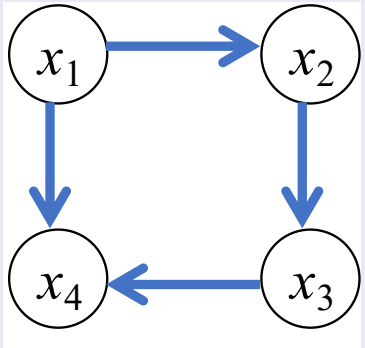
Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Introduction

Directed Graphical Models / Bayesian Networks (BNs)

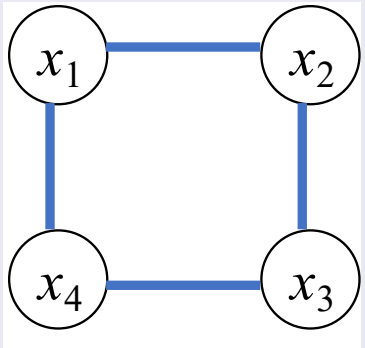
- Self-normalized
- e.g. Hidden Markov Models (HMMs), Neural network (NN) based classifiers, Variational AutoEncoders (VAEs), Generative Adversarial Networks (GANs), auto-regressive models (e.g. RNNs/LSTMs)



$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_1, x_3)$$

Undirected Graphical Models / Random Fields (RFs) / Energy-based models

- Involves the normalizing constant (the partition function) Z
- e.g. Conditional Random Fields (CRFs)



$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z}\Phi(x_1, x_2)\Phi(x_2, x_3)\Phi(x_3, x_4)\Phi(x_1, x_4)$$

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Introduction

Language model

"Language modeling is the art of determining the probability of a sequence of words."

– Joshua T. Goodman in "A Bit of Progress in Language Modeling", 2001

Key points

1. Language model concentrates on text, not speech or image.
2. Words are commonly treated as the basic units, and the rule of organizing words into sentences is to be modeled.
3. To determinate the probability, statistical approach is introduced, called statistical language modeling.

Statistical language model

Formulation

Calculates the probability of a sentence or a sequence of words:

- Input: a word sequence $x_1, \ldots, x_l$
- output: the joint probability $p(x_1, \ldots, x_l)$

Example

- Modeling the grammar:

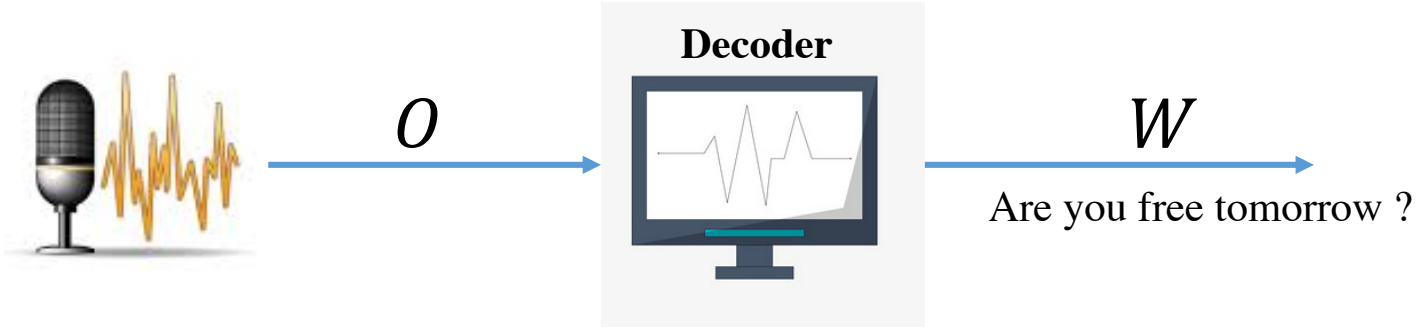
$$p(\text{I am a student}) > p(\text{a I student am})$$

- Modeling the empirical phrases:

$$p(\text{New York}) > p(\text{Old York})$$

Why we need language models

Language models introduce the linguistic constrains to many artificial intelligence applications, such as automatic speech recognition and machine translation.



Bayes Rules

Given the input observation (speech) O, determine the optimal word sequence W:

$$\arg\max_W P(W|O) = \arg\max_W \frac{P(O|W)P(W)}{P(O)}$$

$P(O|W)$ is the acoustic likelihood, and $P(W)$ is the language model.

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Language model evaluation

Perplexity (PPL)

The perplexity on a test set is defined as follows:

$$PPL = \exp\{\frac{-\log P}{N_W}\}$$

where $\log P$ is the log-likelihood on the test set and N_W is the total number of words in the test set. A better LM has a lower PPL.

Example

PPLs on Penn Treebank dataset.

"Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014."

models	PPL
5gram	141
LSTM	78.4

Language model evaluation

Word error rate (WER)

The definition of WER is as follow:

$$WER = \frac{I_W + D_W + S_W}{N_W}$$

where I_W, D_W and S_W denote the number of insertion, deletion and substitution errors respectively, and N_W is the total number of words.

Example

recognized sentence:	i	am	want		buy	a gift for my	friend
true sentence:	i		want	to	buy	a gift for my	friends

$I_W = 1$ ("am"), $D_W = 1$ ("to"), $S_W = 1$("friends $\rightarrow$ friend")

$N_W = 9$

$WER = 3/9$

Language model evaluation

PPL

- PPL calculation is simple.
- Intuitively, PPL is equal to the average number of possible words at current position, given the history words.
- Perplexity is often measured under true history, which is certainly not true in speech recognition systems.

WER

- WER is often the final metric we want to optimize.
- WER calculation does not require the LM to be normalized; Unnormalized LMs, such as whole-sentence LMs, can be used.
- Results are often noisy, especially for small test sets.
- Evaluation needs a full recognition system.

Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Directed graphical models

The directed graphical approach represents the joint probability of a word sequence $p(x_1, \ldots, x_l)$ as the product of local conditional probabilities.

$$p(x_1, \ldots, x_l) = \prod_{i=1}^{l} p(x_i | x_1, \ldots, x_{i-1})$$

To avoid degenerate representation of the conditionals, the history $x_1, \ldots, x_{i-1}$ is reduced to equivalence classes through a mapping $\phi(\cdot)$ with the assumption:

$$p(x_i | x_1, \ldots, x_{i-1}) \approx p(x_i | \phi(x_1, \ldots, x_{i-1}))$$

Language modeling in this directed approach consists of finding

1. suitable mappings $\phi(\cdot)$
2. effective methods to estimate $p(x_i | \phi(x_1, \ldots, x_{i-1}))$

Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

n-gram language models

1. Assume the current word x_i only depends on the previous $n-1$ words

$$\phi(x_1, \ldots, x_{i-1}) = x_{i-n+1}, \ldots, x_{i-1}$$

2. Estimate the conditional probabilities by smoothing the conditional frequence.

$$p(x_i|x_{i-n+1}, \ldots, x_{i-1}) = Smooth\left\{\frac{count(x_{i-n+1}, \ldots, x_{i-1}, x_i)}{count(x_{i-n+1}, \ldots, x_{i-1})}\right\}$$

where

- $count(\cdot)$ is the count of the $n$gram observed in the training text.
- $Smooth$ denotes some smoothing method.

Smoothing

- Avoid over-fitting to improve language model's performance on test set.
- Avoid zero probabilities (when the count of some $n$gram is 0) in evaluating sentences consisting of errors.

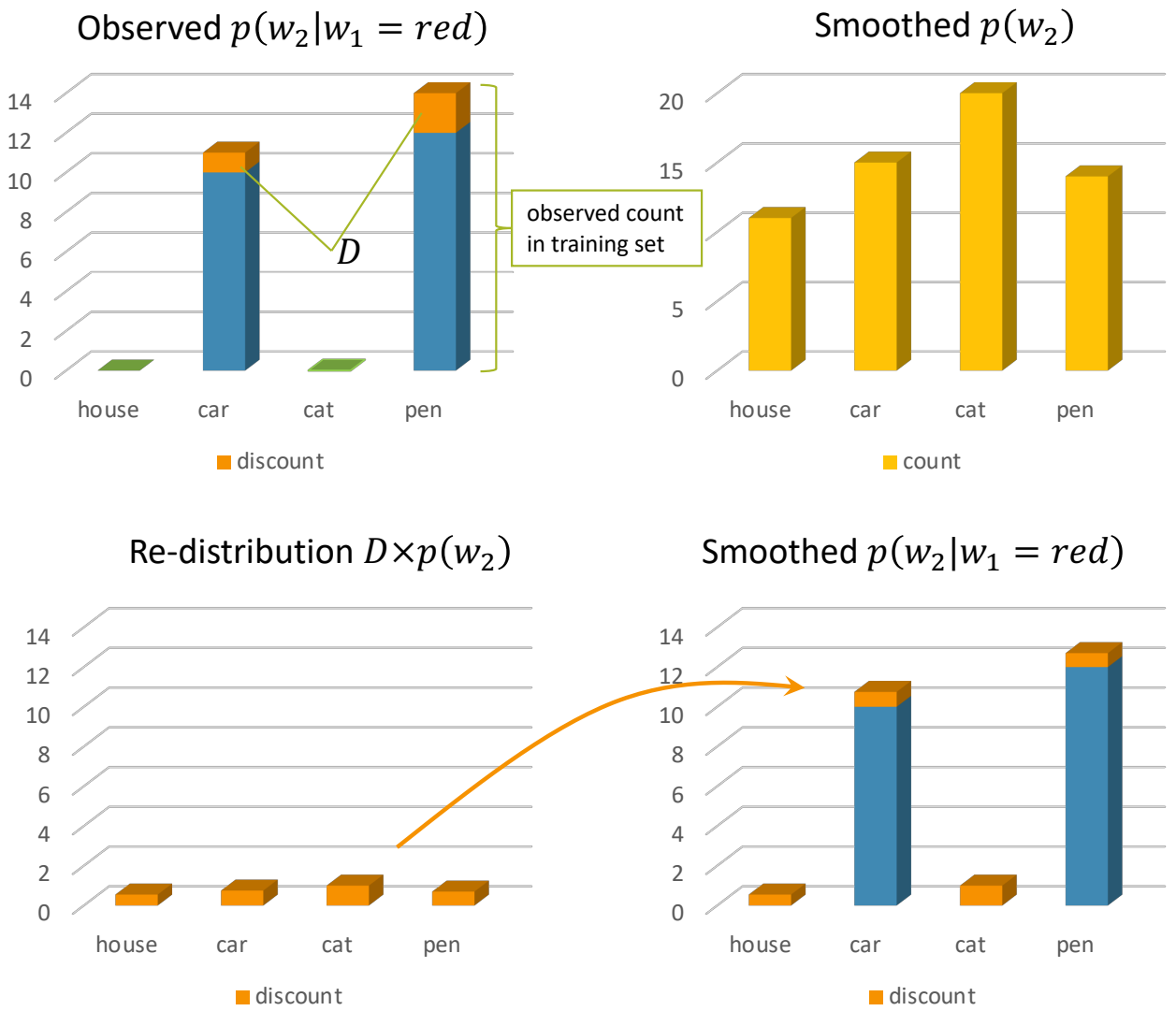
Modified Kneser-Ney smoothing (most popular)

$$p_{MKN}(x_i|h_i) = \frac{\max\{count(h_i, x_i) - D(count(h_i, x_i)), 0\}}{count(h_i)} + \gamma(h_i) P_{MKN}(x_i|h_i^{n-2})$$

where $h_i = x_{i-n+1}, \ldots, x_{i-1}$ and $h_i^{n-2} = x_{i-n+2}, \ldots, x_{i-1}$. $D(\cdot)$ is an empirical discount function.

Smoothing

Figure: An example of smoothing the bigram probability $p(w_2|w_1 = \text{red})$



Advantage and disadvantage

Advantages

1. Model training and inference (calculating the probability of a sentence) are simple and efficient.
2. n-gram LMs can be easily represented by WFSTs and be incorporated into the current WFST-based speech recognition systems.

Drawbacks

1. n-gram LMs can not model complex linguistic constraints.
2. The heuristic smoothing methods are not good enough to estimate the conditional probabilities. As a result, the PPLs of n-gram LMs on test set are significantly larger than the PPLs on training set.

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Neural network language models

- Directed graphical models

$$
\begin{aligned}
p(x_1, \ldots, x_l) &= \prod_{i=1}^{l} p(x_i | x_1, \ldots, x_{i-1}) \\
&\approx \prod_{i=1}^{l} p(x_i | \phi(x_1, \ldots, x_{i-1}))
\end{aligned}
$$

- The conditional probability is defined and estimated by neural networks. Based on the neural network structure used in LMs, there are several classes:
  1. Feedforward neural network LMs
  2. Recurrent neural network LMs
  3. Convolutional neural network LMs

Feedforward neural network (FNN) LMs

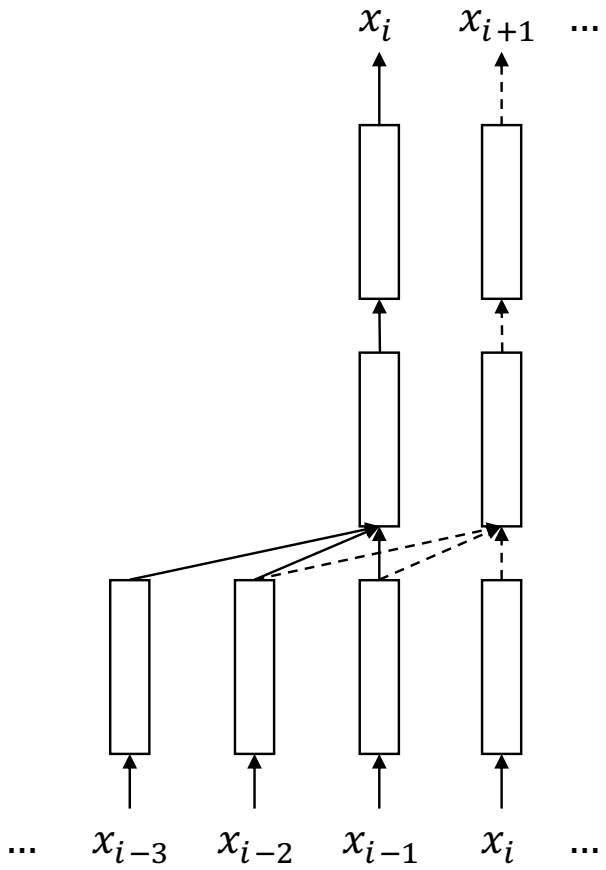


Figure: An example of FNN LMs

- Assume the current word x_i only depends on the previous $n-1$ words, i.e.

$$\phi(x_1, \ldots, x_{i-1}) = x_{i-n+1}, \ldots, x_{i-1}$$

- Define a neural network to estimate the conditional probability.

Compared with n-gram LMs, FNN LMs estimate the n-order conditional probability by defining a simple neural network.

- Word embedding helps to cluster similar words
- Achieve better smoothing

FNN LMs do not achieve significant improvement over n-gram LMs.

Recurrent neural network (RNN) LMs

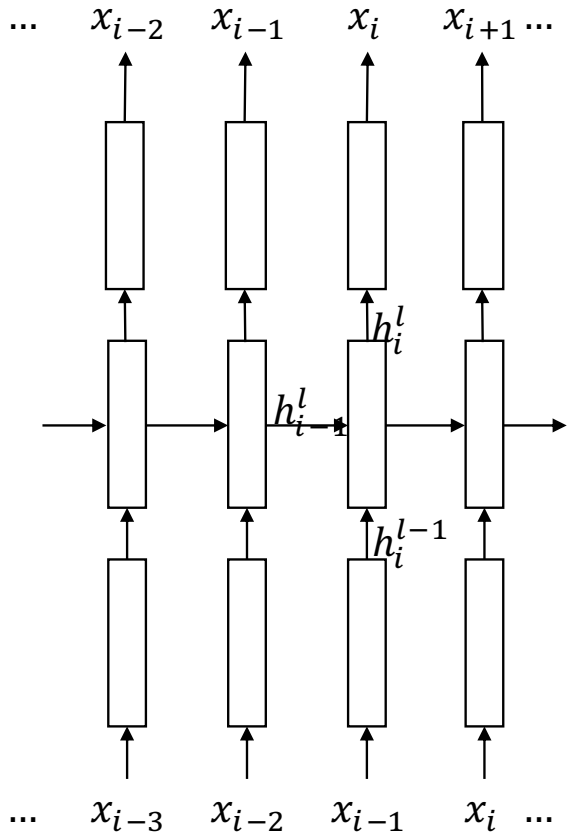


Figure: An example of RNN LMs

- All the history words are encoded into a continuous state vector, i.e. the hidden state of RNN.
- Define a recurrent neural network to estimate the conditional probability.

- Word embedding helps to cluster similar words
- RNN LMs can model the long-range word dependence.

LSTM LMs

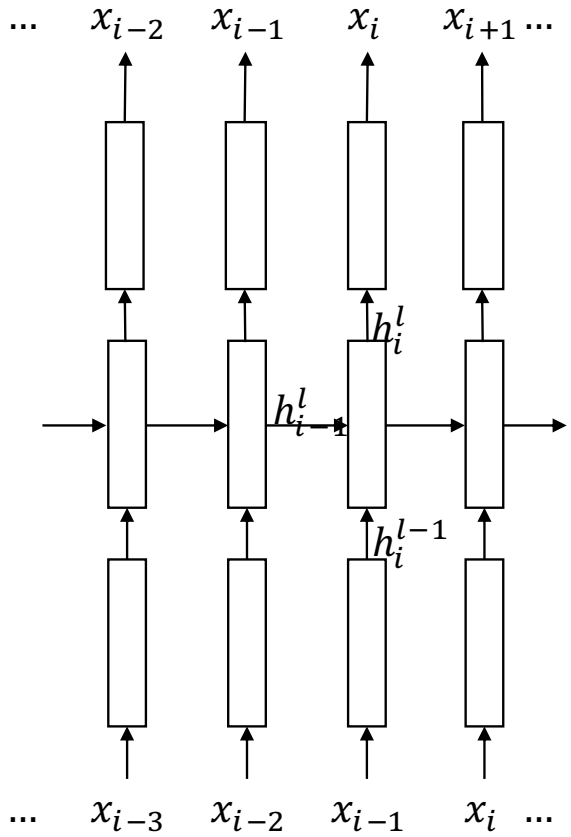


Figure: An example of RNN LMs

Introduce the long-short term memory (LSTM) units into RNN LMs

classical RNN LM

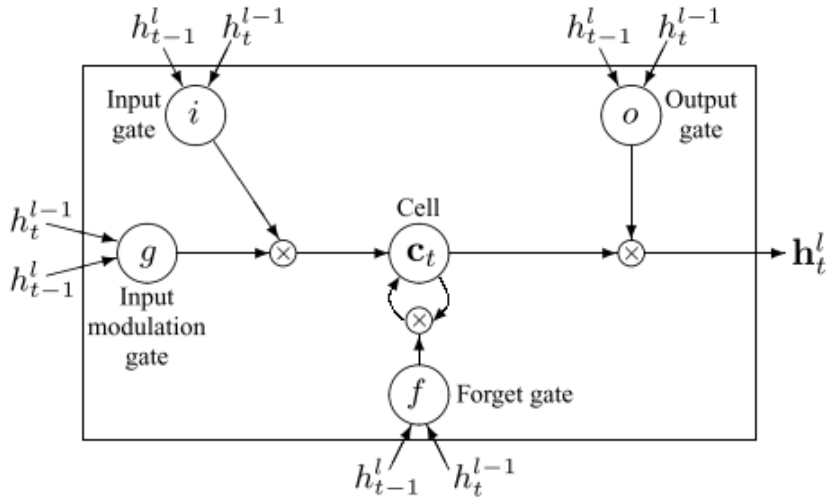
$$h_i^{l-1}, h_{i-1}^l \rightarrow h_i^l$$

LSTM LM

$$h_i^{l-1}, h_{i-1}^l, c_{i-1}^l \rightarrow h_i^l, c_i^l$$

LSTM LMs

Figure: A graphical representation of LSTM memory cells.



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ \tanh \end{pmatrix} T_{4n,2n} \begin{pmatrix} h_i^{l-1} \\ h_{i-1}^l \end{pmatrix}$$

$$c_i^l = f \odot c_{i-1}^l + i \odot g$$

$$h_i^l = o \odot \tanh(c_i^l)$$

W. Zaremba, I. Sutskever, O. Vinyals, "Recurrent neural network regularization." arXiv preprint arXiv:1409.2329, 2014.

Empirical evaluation of LSTM LMs

Table: PPLs on Penn Treebank dataset.

Models	Test set
5gram	141.2
RNN LM [1]	124.7
LSTM LM (large, with dropout) [2]	78.4 (44.5% over 5gram)

Table: Our WER results

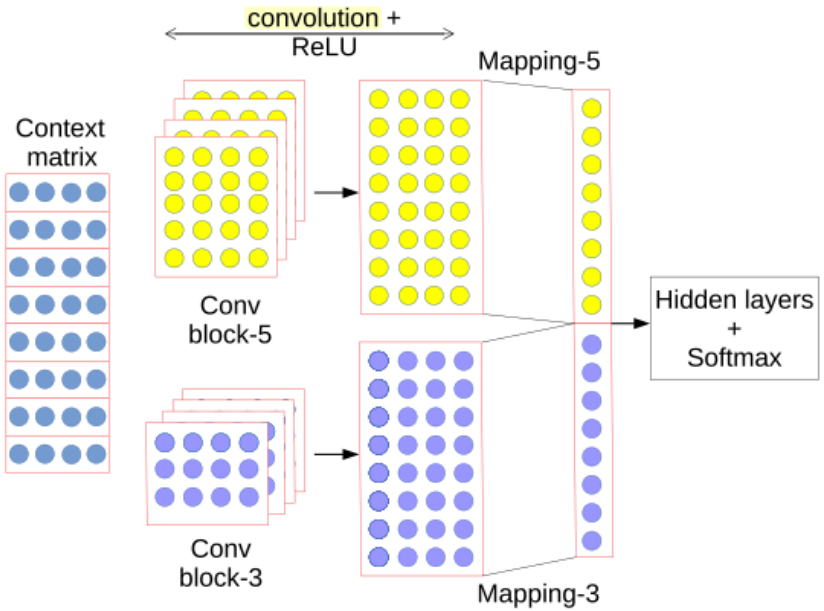
Models	WERs
5gram	8.78
RNN LM [1]	7.90
LSTM LM (large, with dropout) [2]	7.36 (16.2% over 5gram)

[1] T. Mikolov, "Statistical Language Models Based on Neural Networks." 2012.

[2] W. Zaremba, I. Sutskever, O. Vinyals, "Recurrent neural network regularization." arXiv preprint arXiv:1409.2329, 2014.

Convolutional neural network (CNN) LMs

To achieve speed-up compared to RNN LMs, since the computation of CNN can be parallelized over time.



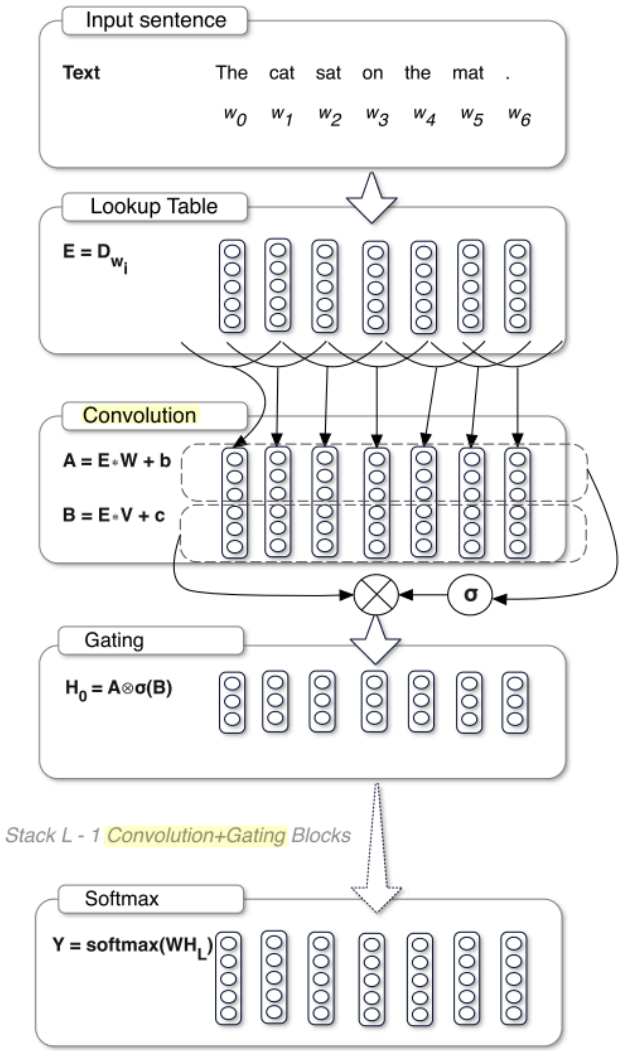
1. $\phi(x_1, \ldots, x_{i-1}) = x_{i-n+1}, \ldots, x_{i-1}$
2. **Conv block**-n: kernal size of n
3. **Mapping**-n: full connected layer

Table: PPL on Penn Treebank

Models	Test PPL
LSTM	78.4
CNN	91

N. Pham, G. Kruszewski, G. Boleda, "Convolutional Neural Network Language Models" Conference on Empirical Methods in Natural Language Processing, 2016.

Gated convolutional neural network (G-CNN) LMs



- Deep CNN structure. (13∼22 CNN layers)
- Gating mechanism is introduced.

$$H_0 = \sigma(E * W + b) \otimes (E * V + c)$$

gate values — input values

- σ is sigmoid function
- $*$ is the convolution operation
- $\otimes$ is the element-wise product

Table: PPLs on Google Billion Word dataset

Models	Test PPL
LSTM	39.8
GCNN-13	38.1

Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, (Facebook AI Research) "Language Modeling with Gated Convolutional Networks." 2016

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Computation bottleneck of neural network LMs

Advantages of NN LMs

1. Word embedding in continuous vector space creates a notion of similarity between words and achieves a level of generalization that is hard with discrete features.
2. NN LMs has the ability to learn non-linear interactions between underlying features.
3. LSTM LMs can model longer contexts by using memory cells.

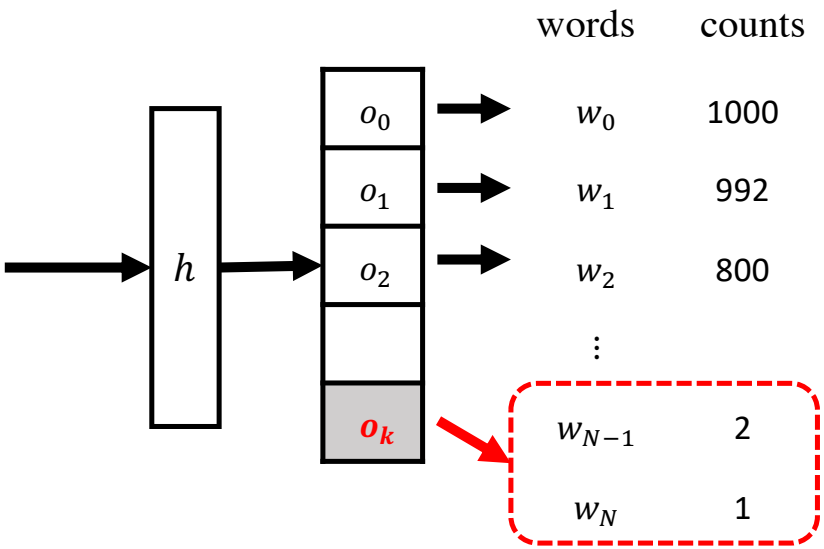
One drawback of NN LMs

The computation of Softmax at the final output layers is expensive, which limits the application of NN LMs on very large vocabulary tasks.

Example

Penn Treebank dataset: a vocabulary of **10 K** words
Google Billion Word dataset: a vocabulary of over **800 K** words

Approximate Softmax - Shortlist



- Sort all the words in vocabulary according to the word count, and map the less frequent words to a single unit.
- Words in the single unit are assumed to be distributed based on the unigram p_{uni}.

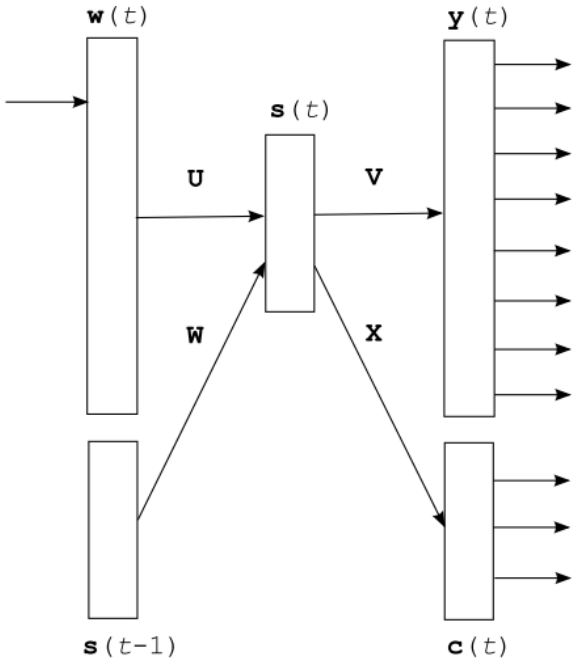
$$p(w_i|h) = \begin{cases} \text{softmax}(o_i), & \text{if } i < k \\ \text{softmax}(o_k)p_{uni}(w_i), & \text{otherwise} \end{cases}$$

computational cost :

$$|V| \to k$$

where $|V|$ is the vocabulary size, and k is the size of the shortlist.

Approximate Softmax - Class-based Softmax



1. Cluster all the words in the vocabulary into classes.
2. NN LMs first predict the class, and then given class predict the words.

Factorize the probability:

$$p(x|h) = p(c|h) \times p(x|h,c)$$
$$= \mathbf{c}(t) \times \mathbf{y}(t)$$

computational cost :

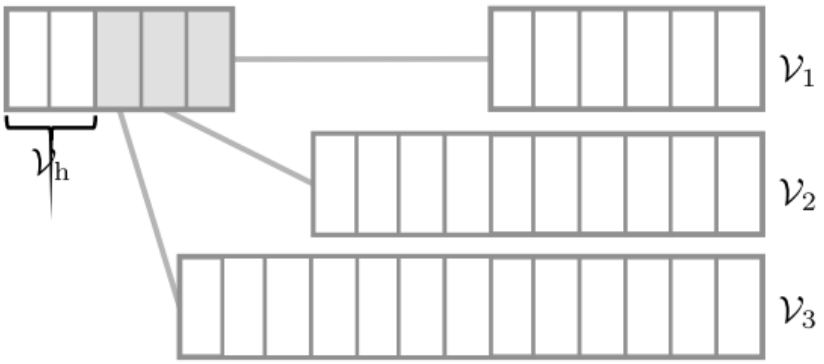
$$|V| \rightarrow |C| + |V|/|C|$$

where $|V|$ is the vocabulary size and $|C|$ is the class number.

T. Mikolov, "Statistical Language Models Based on Neural Networks." 2012.

Approximate Softmax - Adaptive Softmax

Combining the ideas of the shotlist and the class-based softmax.



Hierarchical model

1. The first level includes both the most frequent words and the clusters.
2. The outputs in the second level are associated with rare words.

Table: Evaluation on GPUs

	PPL	training time
Full softmax	144	83 min
Adaptive softmax	147	30 min

E. Grave, et al. "Efficient softmax approximation for GPUs." arXiv:1609.04309, 2016.

Sampled Softmax

Assuming most probabilities should be obtained by the more frequent words.

Standard softmax: summation over all the words in vocabulary

$$p(w_i|h_i) = \frac{e^{o(w_i,h_i)}}{\sum_{w' \in V} e^{o(w',h_i)}}$$

Sampled softmax: summation over a sampled set of words

$$p(w_i|h_i) = \frac{e^{o(w_i,h_i)}}{\sum_{w' \in S \cup \{w_i\}} e^{o(w,h_i)}}$$

where S denotes a sample set drawn from a distribution (e.g. a log-uniform distribution)

computational cost :

$$|V| \rightarrow |S| + 1$$

Rafal Josefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Wu Yonghui, "Exploring the limits of language modeling," arXiv preprint arXiv:1602.02410v2, 2016.

Self-normalization strategy

$$p(w_i|h_i) = \frac{e^{o(w_i,h_i)}}{\sum_{w' \in S \cup \{w_i\}} e^{o(w,h_i)}}$$

Standard method: Optimizing the log-likelihood:

$$\arg\max \sum_{i}^{T} \log p(w_i|h_i)$$

Self-normalization: adding a penalty to yield the normalization constant $Z(h_i) = \sum_{w' \in S \cup \{w_i\}} e^{o(w',h_i)}$ to be close to 1

$$\arg\max \sum_{i}^{T} \log p(w_i|h_i) + \alpha \log^2 \left(\sum_{w' \in S \cup \{w_i\}} e^{o(w',h_i)} \right)$$

Skipping softmax computation at inference time !!

S. Kumar, et al. (Google Inc.) "Lattice rescoring strategies for long short term memory language models in speech recognition." ASRU, 2017

Noise-contrastive estimation (NCE)

Noise-contrastive estimation (NCE) is a training method to skip softmax calculation.

- the model distribution $p_m(x;\theta)$ with parameter θ
- the data distribution $p_d(x)$ whose samples are the training set
- a fixed noise distribution $p_n(x)$

Assuming for each sample x from training data, ν noise samples are generated from $p_n(x)$, then the posterior probabilities of a sample x belonging to the data distribution $C = 1$ and the noise distribution $C = 0$ are given by

$$P(C = 1|x;\theta) = \frac{p_m(x;\theta)}{p_m(x;\theta) + \nu p_n(x)}$$

$$P(C = 0|x;\theta) = \frac{\nu p_n(x)}{p_m(x;\theta) + \nu p_n(x)}$$

M. U. Gutmann, A. Hyvarinen, "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics." Journal of Machine Learning Research 13, 2012

Noise-contrastive estimation (NCE)

NCE defines a binary clustering question:

$$P(C=1|x;\theta) = \frac{p_m(x;\theta)}{p_m(x;\theta)+\nu p_n(x)} \qquad \text{be data}$$

$$P(C=0|x;\theta) = \frac{\nu p_n(x)}{p_m(x;\theta)+\nu p_n(x)} \qquad \text{be noise}$$

To optimize the clustering question, NCE maximizes the following conditional log-likelihood.:

$$J(\theta) = \frac{1}{|D|}\sum_{x\in D}\log P(C=1|x;\theta) + \frac{\nu}{|S|}\sum_{x\in S}\log P(C=0|x;\theta)$$

- samples from training set D belong to the data distribution
- noise samples belong to the noise distribution

Conclusion:

$$p_m(x;\theta) \to p_d(x)$$

NCE for unnormalized models

Unnormalized model:

$$p_m(x;\theta) = \frac{1}{e^c} p_m^0(x;\theta)$$

where $p_m^0(x;\theta)$ is the unnormalized probability and c is the logarithmic normalization constant which is hard to be computed exactly.

NCE training

$$\log p_m(x;\theta,c) = \log p_m^0(x;\theta) - c$$

The normalization constant c can be treated as the normal parameters and updated together with ordinary parameters θ.

Conclusion:

$$p_m^0(x;\theta) \to p_d(x)$$

$$c \to \text{ true logarithmic normalization constant}$$

NCE in NN LMs

Fix the normalization constants $Z(h_i)$ to an empirical constant, such as 1 or e^9.

$$p(w_i|h_i) = \frac{e^{o(w_i,h_i)}}{Z(h_i)} \rightarrow \frac{1}{c} e^{o(w_i,h_i)}$$

Apply NCE to estimate the neural network parameters.

- For each word w_i, draw ν noise words based on the uniform/unigram distribution (p_n)
- Maximize the NCE loss to update the parameters and fix the normalization constants.

#output layer	train speed (w/s)	
	CE	NCE
20k	10.1k	19.7 k
25k	9.1k	19.7k
30k	8.0k	19.7 k

X. Chen, et al."Recurrent neural network language model training with noise contrastive estimation for speech recognition.", ICASSP, 2015

Improving NCE

Methods to further improve the training speed by reducing the sampling numbers in NCE.

B. Zoph, et al, "Simple, fast noise-contrastive estimation for large rnn vocabularies." in HLT-NAACL, 2016.

- Accelerate the NCE training by sharing the noise samples for each target word in the mini-batch to reduce the sample times.

Y. Oualil and D. Klakow, "A batch noise contrastive estimation approach for training large vocabulary language models," arXiv preprint, 2017.

- Skip the sampling process by taking the other target words in current mini-batch as the noise samples.

PPL on Google Billion Word dataset (vocabulary $\approx 800K$)

Models	Test PPL
LSTM (adaptive softmax)	48.0
LSTM (B-NCE)	48.1

Summary of Speed-up methods

methods	**speed-up training**	**speed-up inference**
full softmax	$O(\|V\|)$	$O(\|V\|)$
shortlist softmax	$O(k)$	$O(k)$
class-based softmax	$O(\|C\| + \|V\|/\|C\|)$	$O(\|C\| + \|V\|/\|C\|)$
adaptive softmax	$O(k)$	$O(k)$
sampled softmax	$O(\|S\|)$	$O(\|V\|)$
self-normalization strategy	$O(\|S\|)$	$O(1)$
NCE	$O(\|S\|)$	$O(1)$

Table: Comparison of different speed-up methods. $|V|$ is the vocabulary size. $|C|$ is the class number in class-based softmax. k is the shortlist size. $|S|$ is the sample number.

Contents




1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Character-based LMs

Large vocabulary task.

The naive way:

Mapping the less frequent and unknown words to a single $\langle \text{unk} \rangle$ token.

A better way:

- Generate character-level embedding by constructing a NN on the character sequence of each words
- Concatenate the word-level embedding (for most frequent words) and character-level embedding

Problem: solve the OOVs on the input side. The softmax at output layers still needs to be computed over all the words.

Character-only LMs:

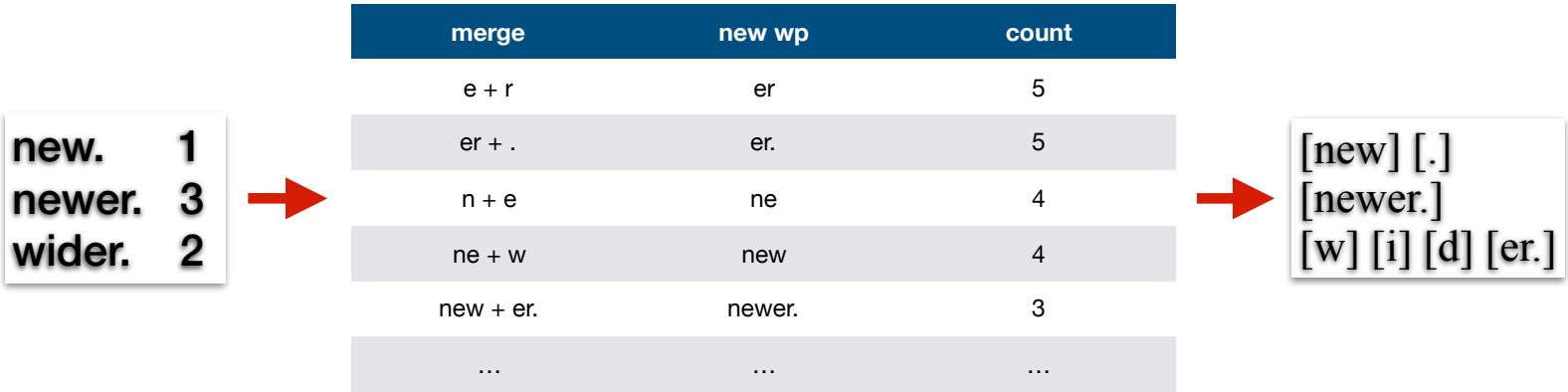
Build the pure character-only LMs by modeling character sequences.
Problem: bad performance !

Tradeoff between characters and words

Word-piece (subword / character ngram) based LMs:

- Splitting words into word-pieces can be performed automatically.
- The algorithm based on a simple data compression technique - Byte Pair Encoding (BPE), that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte.
- Cover the frequent words and avoid the OOV problems.
- Compatible with word-piece based end2end ASR systems.

Figure: A example of word-piece generation



WERs of word-piece based LSTM LMs

A. Zeyer, et al, "Improved training of end-to-end attention models for speech recognition." arXiv preprint arXiv:1805.03294, 2018.

Table: WERs on Librispeecch dataset. (Seq2seq models)

model	WERs on test clean
4gram	4.82
word-piece LSTM	**3.82** (∼21% relative reduction)

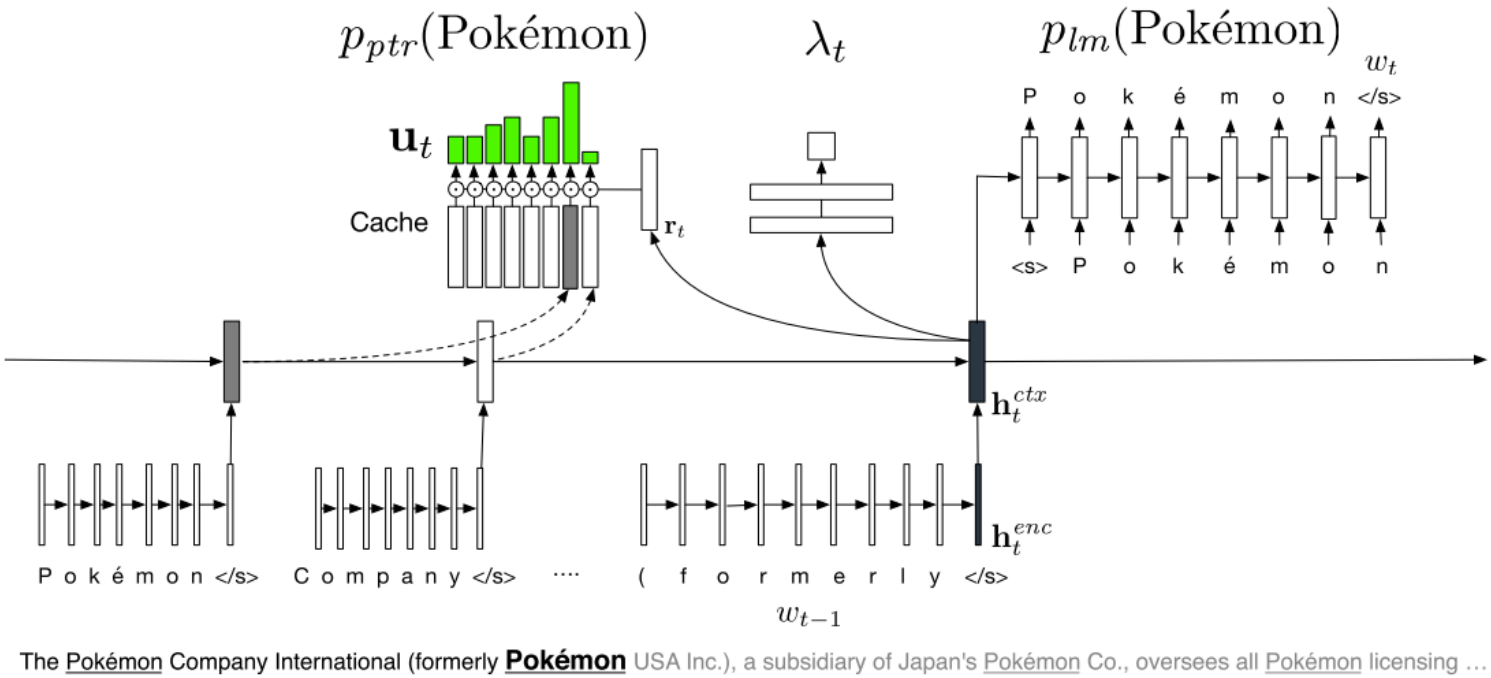
S. Toshniwal, et al. "A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition." arXiv preprint arXiv:1807.10857, 2018.

Table: WERs of LAS models

model	Switchboard	CallHome	Eval2000
LAS	17.1	27.9	22.6
+ word-piece LSTM	**15.6**	**26.6**	**21.1**

Create and Reuse words in Open-vocabulary LMs

$$p(\text{Pokémon}) = \lambda_t p_{lm}(\text{Pokémon}) + (1 - \lambda_t) p_{ptr}(\text{Pokémon})$$



- A hierarchical character-level LM to generate word character by character
- A continuous cache component to save the generated words
- Interpolate the two probabilities

K. Kawakami, C. Dyer, and P. Blunsom. "Learning to create and reuse words in open-vocabulary neural language modeling." arXiv preprint arXiv:1704.06986, 2017.

Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Undirected graphical models (UGM)

The undirected graphical approach directly models the joint probability of a sentence (sequence of words) $p(x^l)$, by assuming the follow exponential family model:

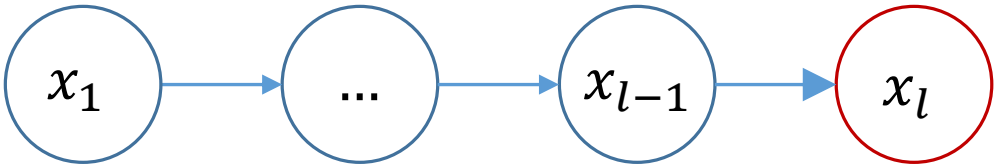
$$p(x^l;\theta) = p^0(x^l)\frac{1}{Z}e^{\phi(x^l;\theta)}$$

where $x^l = (x_1, \ldots, x_l)$, θ is the parameters and $p^0(x^l)$ is a prior probability. $\phi(\cdot)$ is called potential function or unnormalized log-probabilities. Z is the corresponding normalization constant.

Calculate an unnormalized log-probabilities based on features of a whole sentence and then normalize it on the space of sentences.

DGM and UGM

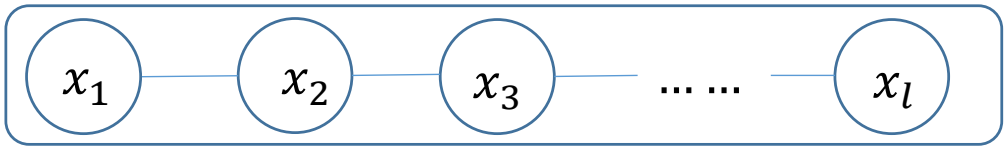
Directed graphical models (DGM):



$$p(x_1, \ldots, x_l) = \prod_{i=1}^{l} p(x_i | x_1, \ldots, x_{i-1})$$

- conditional probability
- local normalized

Undirected graphical models (UGM):



$$\phi(x_1, \ldots, x_l)$$

$$p(x_1, \ldots, x_l) = \frac{1}{Z_l} e^{\phi(x_1, \ldots, x_l)}$$

- joint probability
- global normalized

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Whole-sentence maximum entropy (WSME) LMs





Denote by $s = (x_1, x_2, \ldots)$ a sentence (a sequence of words), the joint probability $p(s)$ is defined as:

$$p(s) = \frac{1}{Z} \cdot p_0(s) \cdot \exp\left(\sum_i \lambda_i f_i(s)\right)$$

- $f_i(\cdot)$ is the feature function and λ_i is the corresponding parameter.
- $p_0(s)$ is a prior probability.
- Z is the global normalization constant.

Advantages

Support global sentence-level features.

Ronald Rosenfeld. "A whole sentence maximum entropy language model." 1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings, 1997.

Training WSME LMs

Given the training set D, the maximum likelihood estimation of WSME can be achieve by solving:

$$\sum_s p(s) f_i(s) = \frac{1}{|D|} \sum_{s \in D} f_i(s)$$

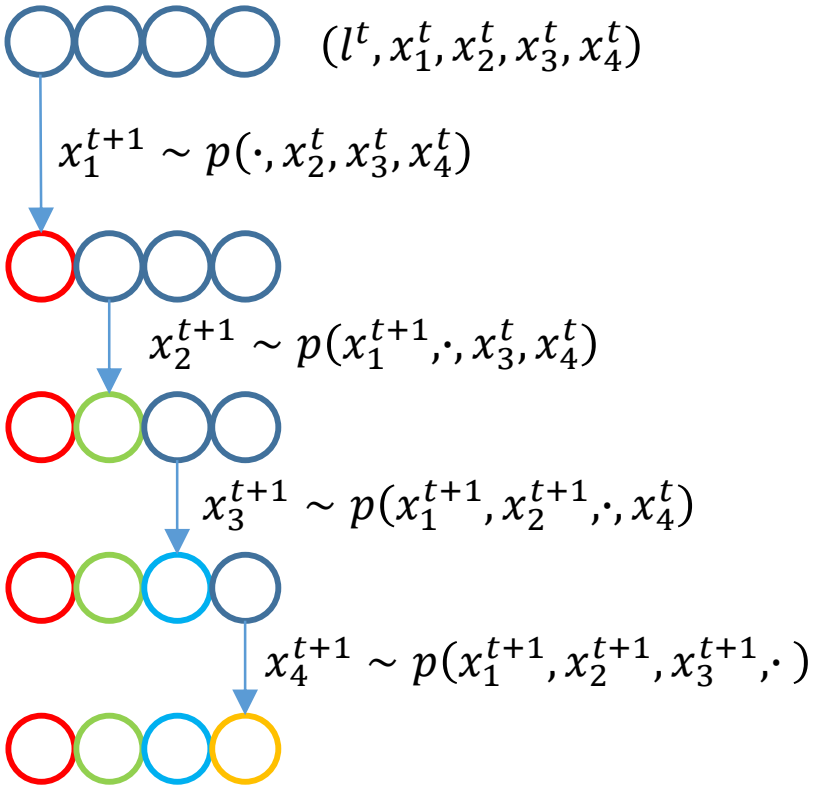
1. $\frac{1}{|D|} \sum_{s \in D} f_i(s)$ is the feature expectation on the training set.
2. $\sum_s p(s) f_i(s)$ is the feature expectation under model distribution $p(s)$.

The following methods are used to approximate $\sum_s p(s) f_i(s)$:

- Gibbs sampling
- Metropolis sampling
- Importance sampling

Gibbs sampling

1. Draw sentence $s \sim p(s)$, and get $S = \{s_1, s_2, \ldots, s_k\}$
2. Calculate $\sum_s p(s) f_i(s) \approx \frac{1}{k} \sum_{s \in S} f_i(s)$



Gibbs sampling:

- Choose a word position (by cycling through all word positions in order)
- Draw a new word to replace the old words (based on the conditional probability)

Transitions between lengths:

- $< /s >$ can be replaced by an ordinary word.
- words in the last position can be replaced by $< /s >$.

Metropolis sampling

1. Draw sentence $s \sim p(s)$, and get $S = \{s_1, s_2, \ldots, s_k\}$
2. Calculate $\sum_s p(s) f_i(s) \approx \frac{1}{k} \sum_{s \in S} f_i(s)$

Metropolis sampling

When drawing the word at position i, a new word v is proposed from the proposal $q(\cdot)$ to replace the original word w. Then the new sentence is accepted with probability:

$$\min \left\{ 1, \frac{p(s_v) \cdot q(w)}{p(s_w) \cdot q(v)} \right\}$$

- s_v is the new sentence where the i-th word is v.
- s_w is the original sentence where the i-th word is w.
- $q(\cdot)$ can be set as an unigram distribution.

Importance sampling (Best)

$p(s) = p^0(s)/Z$ where $p^0(s) = p_0(s)\exp\left(\sum_i \lambda_i f_i(s)\right)$ in WSME LMs.

Importance sampling

$$\sum_s p(s) f_i(s) = \frac{\sum_s p^0(s) f_i(s)}{Z} = \frac{\sum_s p^0(s) f_i(s)}{\sum_s p^0(s)}$$

$$= \frac{\sum_s q(s) \frac{p^0(s)}{q(s)} f_i(s)}{\sum_s q(s) \frac{p^0(s)}{q(s)}}$$

$$\approx \frac{\sum_{s\sim q(s)} \frac{p^0(s)}{q(s)} f_i(s)}{\sum_{s\sim q(s)} \frac{p^0(s)}{q(s)}}$$

- Generate sample sentences from a simple distribution $q(s)$;
- Approximate the expectation $\sum_s p(s) f_i(s)$ using the above equation.

PPL of WSME LMs

WSME:

$$p(s) = \frac{1}{Z} \cdot p_0(s) \cdot \exp\left(\sum_i \lambda_i f_i(s)\right)$$

Using importance sample to estimate the normalization constant:

$$Z = \sum_s p_0(s) \exp\left(\sum_i \lambda_i f_i(s)\right) = E_{p_0}\left[\exp\left(\sum_i \lambda_i f_i(s)\right)\right]$$

- p_0 is commonly defined as an n-gram LMs.
- Generating samples from p_0 is sample.
- Approximate Z by generating samples from p_0.

PPLs computed by this method are noisy

Features in WSME LMs

1. Sentence length
2. n-gram features

$$f_\alpha(s) = \#\text{ of times } n\text{-gram } \alpha \text{ occurs in } s$$

3. Syntactic Features

I/NNP think/VBP it/PRPA might/AUX be/VB correct/JJ

4. Stochastic Context-Free Grammar [1] and Dependency Grammar [2]

---

[1] F. Amaya and J. M. Benedi. "Improvement of a whole sentence maximum entropy language model using grammatical features." ACL, 2001.

[2] T. Ruokolainen, T. Alumäe and M. Dobrinkat. "Using Dependency Grammar Features in Whole Sentence Maximum Entropy Language Model for Speech Recognition." Baltic HLT. 2010.

WSME results

PPLs

As the normalization constant Z cannot be calculated exactly, the PPLs of WSME LMs are noisy. And their performance results ever reported are not satisfactory.

WERs

WSME LMs can be used in n-best rescoring.

Table: Rosenfeld R, Chen S F, Zhu X. "Whole-sentence exponential language models: a vehicle for linguistic-statistical integration." , 2001.

Models	WERs
baseline trigram	36.53
WSME	36.38

Table: T. Ruokolainen, et al. "Using Dependency Grammar Features in Whole Sentence Maximum Entropy Language Model for Speech Recognition." , 2010.

Models	WERs
baseline trigram	29.6
WSME	30.6
trigram + WSME	**27.9**

Contents





1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Trans-dimensional random field (TRF) LMs

Assume the sentence x^j and its length j are distributed from an exponential family model:

$$p(j, x^j; \theta) = \pi_j \cdot \frac{1}{Z_j(\theta)} e^{\phi(x^j;\theta)}$$

where

- x^j is a word sequence with length j $(j = 1, 2, \ldots, m)$.
- π_j is the prior length probability, with $\sum_{j=1}^{m} \pi_j = 1$.
- $\phi(\cdot; \theta)$ is called potential function with parameters θ.
- $Z_j(\theta) = \sum_{x^j} e^{\phi(x^j;\theta)}$ is the normalization constant of length j.

Differences between TRF LMs and WSME LMs




WSME: $$p(s) = \frac{1}{Z} \cdot \exp\left(\sum_i \lambda_i f_i(s)\right)$$

TRF: $$p(j, x^j; \theta) = \pi_j \cdot \frac{1}{Z_j(\theta)} e^{\phi(x^j;\theta)}$$

Formulation

WSME: using a single global normalization constant Z for all lengths.
TRF: using a different normalization constant Z_l for each length.

Features

WSME: using discrete features
TRF: using discrete features and neural network features

Training methods

TRF: propose several effective training methods.

Advantages - flexible in using features

Discrete features

"welcome to Taiwan island"

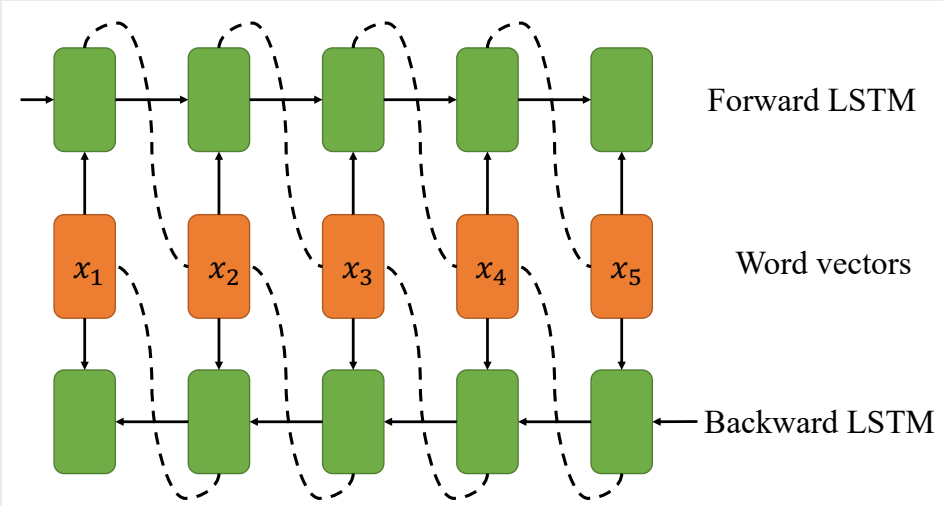
$f_{\text{welcome to}}(x^j) = 1$

$f_{\text{to xx island}}(x^j) = 1$

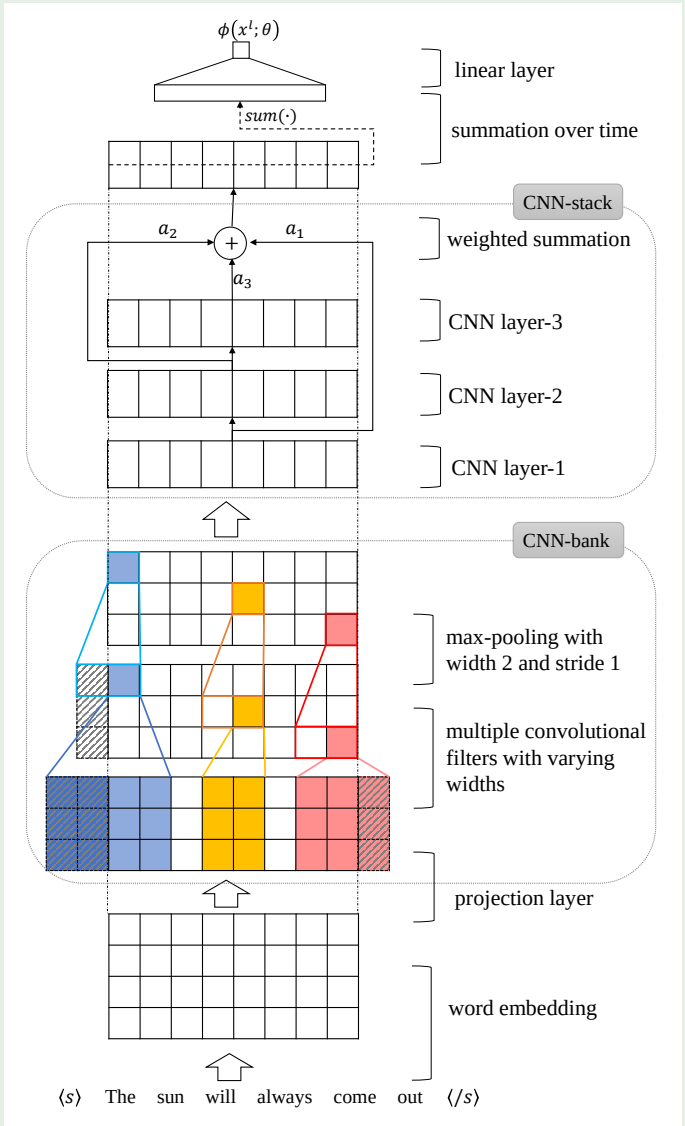
$f_{\text{Taiwan island}}(x^j) = 0$

. . . . . .

Bidirectional LSTM features



CNN features



Advantages - Avoid label bias

- **"Label bias" problem**

Training data

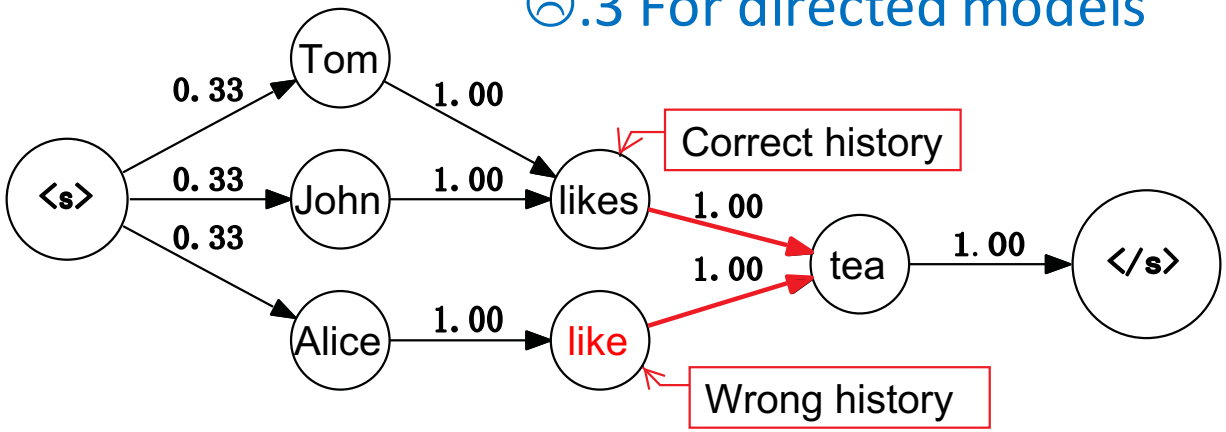
Tom likes tea
John likes tea
Alice like tea

⇒

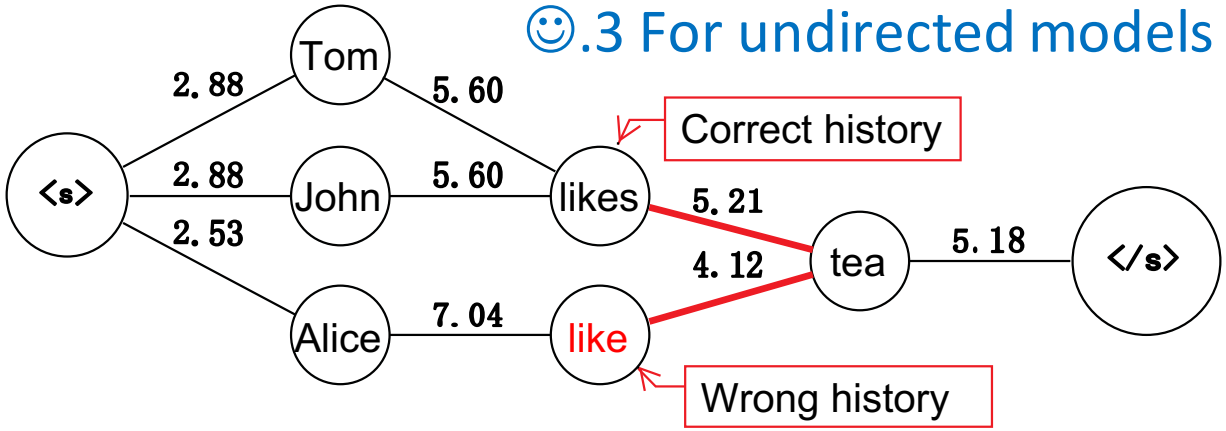
Testing data

Alice likes tea
Tom like tea

☹.3 For directed models



☺.3 For undirected models



Advantages - Effective inference

TRF: $$p(j, x^j; \theta) = \pi_j \cdot \frac{1}{Z_j(\theta)} e^{\phi(x^j;\theta)}$$

2. Effective inference

- NN LMs: Softmax calculation (local normalization) is very slow.

$$p(w) = \frac{o(w)}{\sum_{w \in V} o(w)}, \quad |V| \approx 10^4 \sim 10^6$$

- TRF LMs: The global normalization constants Z_l can be calculated or estimated beforehand. Inference is very fast.

Training TRF LMs

Challenges

- Maximum likelihood principle:

$$E_{p_d}\left[\frac{\partial \phi}{\partial \theta}\right] - E_{p(j,x^j;\theta,\zeta)}\left[\frac{\partial \phi}{\partial \theta}\right] = 0$$

Expectation on training set

Expectation under model distribution (Infeasible)

- Computing the normalization constants:

$$Z_j(\theta) = \sum_{x^j} e^{\phi(x^j;\theta)}, \quad j = 1, 2, \ldots$$

Training TRF LMs

Challenges

- Maximum likelihood principle:

$$E_{p_d}\left[\frac{\partial \phi}{\partial \theta}\right] - E_{p(j,x^j;\theta,\zeta)}\left[\frac{\partial \phi}{\partial \theta}\right] = 0$$

Expectation on training set

Expectation under model distribution (Infeasible)

- Computing the normalization constants:

$$Z_j(\theta) = \sum_{x^j} e^{\phi(x^j;\theta)}, \quad j = 1, 2, \ldots$$

Breakthrough !

1. **Augmented stochastic approximation (AugSA)**: generating sample sentences from the model distribution to jointly estimate the parameters and normalization constants.
2. **Dynamic noise-contrastive estimation (DNCE)**: model training by discriminating between noise sentences and training sentences.

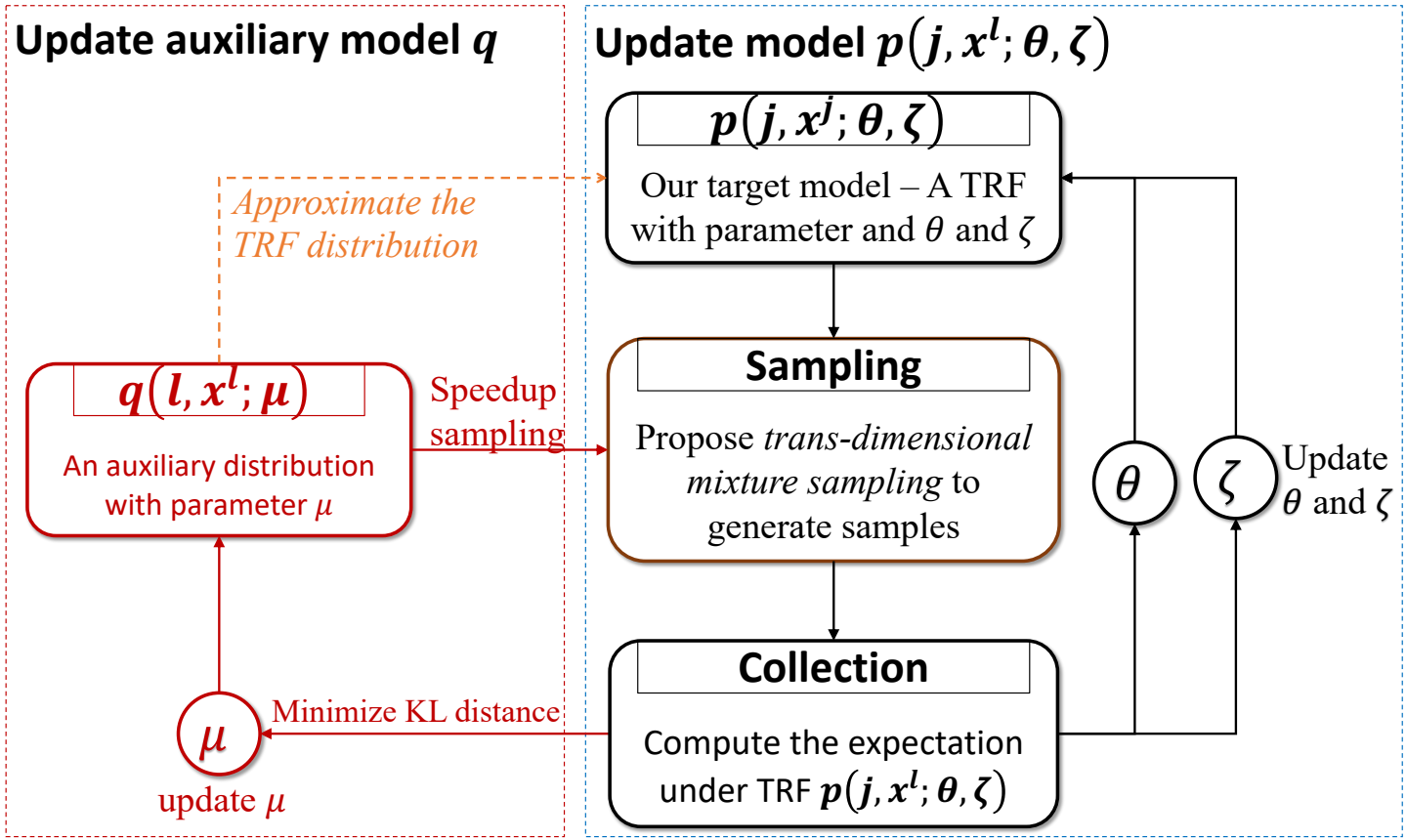
1. Augmented stochastic approximation (AugSA)

Define a distribution q:

$$\frac{\partial}{\partial \mu} KL(q(l, x^l; \mu)||p) = 0$$

$$E_{p_d}\left[\frac{\partial \phi}{\partial \theta}\right] - E_{p(j,x^j;\theta,\zeta)}\left[\frac{\partial \phi}{\partial \theta}\right] = 0$$

$$\sum_{x^l} p(l, x^l; \theta, \zeta) = \pi_l$$



2. Dynamic noise-contrastive estimation (DNCE)

Given model distribution $p_m(x)$, data distribution $p_d(x)$, and noise distribution $p_n(x)$ The probabilities of a sample x belonging to model distribution (C=0) and noise distribution (C=1) are

$$P(C=0|x) = \frac{p_m(x)}{p_m(x) + \nu p_n(x)}$$
$$P(C=1|x) = 1 - P(C=0|x)$$

The objectives:

$$\begin{cases} \min_{p_n} KL(p_d||p_n) & \Rightarrow p_n \to p_d \\ \max\limits_{p_m} \sum\limits_{x\sim\frac{p_d+p_n}{2}} \log P(C=0|x) + \sum\limits_{x\sim p_n} \log P(C=1|x) & \Rightarrow p_m \to \frac{p_n+p_d}{2} \to p_d \end{cases}$$

Advantages

- make p_n converge to p_d to achieve a reliable model estimation with a small sample number ν.
- alleviate the overfitting problem caused by the sparseness of the training set.

WERs on WSJ'92 **English** datasets

Table: Training set is Penn Treebank training set, Test set is WSJ'92 test set

Model	PPL	WER(%)	#Param(M)	Training time	Inference time
KN5	141.2	8.78	2.3	22s	0.06s
LSTM-2×200	113.9	7.96	4.6	1.7h	6.36s
LSTM-2×650	84.1	7.66	19.8	7.5h	6.36s
LSTM-2×1500	78.7	7.36	66.0	1 day	9.09s
RNN-TRF (DNCE)	∼66	**7.40**	2.6	1 day	**0.08s**

- RNN-TRF outperforms KN5, with relative WERs reduction 15.7%.
- RNN-TRF achieves close WERs to LSTM-2×1500 with only 4% parameters
- RNN-TRF is 114x faster than LSTM-2×1500 when computing the probabilities.

Bin Wang and Zhijian Ou, "Improved training of neural trans-dimensional random field language models with dynamic noise-contrastive estimation", Spoken Language Technology (SLT), 2018.

WERs on HKUST **Chinese** dataset

Table: Training set includes about 2.4 M Chinese characters. The vocabulary contains about 4 K Chinese characters

Model	WERs(%)	#Param(M)	Training time	Inference time
KN5	28.48	3.5	8s	0.004s
LSTM-2x200	27.60	2.2	0.5h	0.048s
RNN-TRF	27.72	1.4	1 day	0.009s
LSTM+KN5	27.36			
KN5+RNN-TRF	27.30			
LSTM+RNN-TRF	26.91			
model interpolation	26.87			

- RNN-TRF outperforms KN5, with relative WERs reduction 2.7%.
- RNN-TRF achieves close WER to LSTM with only 63% parameters
- RNN-TRF is 5x faster than LSTM when computing the probabilities.
- Interpolating RNN-TRF and LSTM and KN5 can achieve the lowest WERs.

These demonstrate the language independence of TRF LMs !

WERs on Google one-billion word dataset

TRF LMs on large vocabulary dataset. ($|V| \approx 800K$)

Model	WER (%)	#param (M)	Training time	Inference time
KN5	6.13	133	2.5 hours	0.491 second
LSTM-2×1024 (adaptive softmax)	5.55	191	6 days	0.909 second
neural TRF	5.47	114	14 days	0.017 second
KN5+LSTM	5.38			
KN5+RNN-TRF	5.51			
LSTM+RNN-TRF	5.25			
KN5+LSTM+RNN-TRF	5.06			

- RNN-TRF achieves slightly lower WER than LSTM with less parameters.
- RNN-TRF is 54x faster than LSTM when computing the probabilities.
- Interpolating RNN-TRF and LSTM and KN5 can achieve the lowest WERs.

These reveal the advantages of TRF LMs in large-vocabulary tasks !

Contents

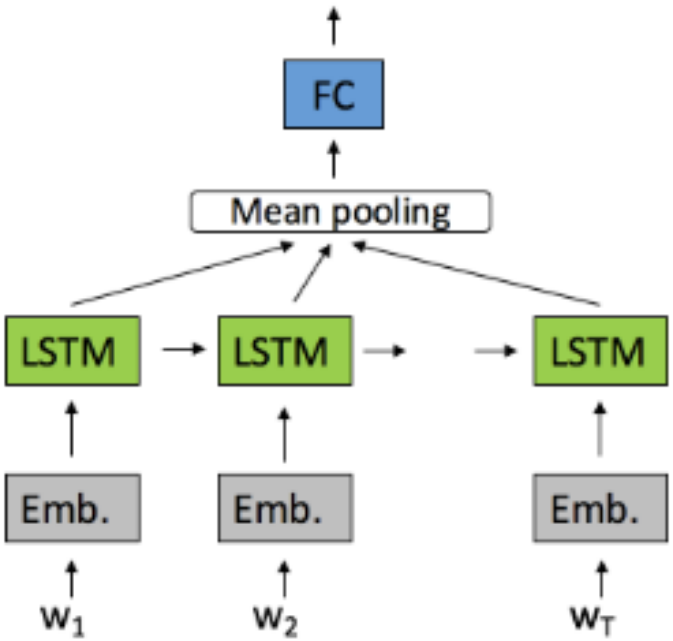




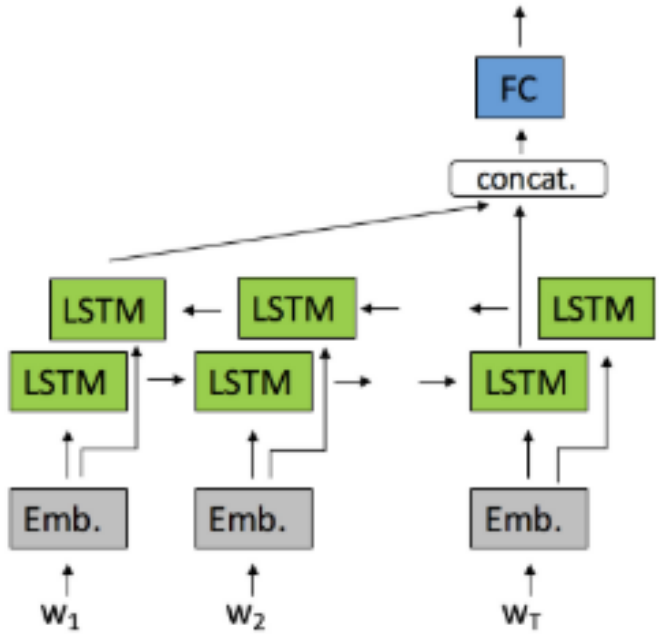
1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Whole sentence neural (WSN) LMs

1. Construct neural network on the whole sentence s to compute the joint probability $p(s)$.
2. Noise contrastive estimation with an elaborate noise generator is used to train WSN LMs.



(a) LSTM



(b) BiLSTM

Y. Huang, et al. "Whole sentence neural language models." ICASSP, 2018.

Whole sentence neural (WSN) LMs

Sampling in NCE

- A back-off n-gram LMs is used as the noise sampler.
- Noise samples are generated by introducing insertion, substitution or deletion errors to sentences selected from training set.
- Noise sample generation and log-likelihood evaluation from the noise distribution are not matched but empirically found to work well

Experiments

- 20 noise samples per data sentence are generated.
- The WSN LM is trained on a small corpus (24 M words) and interpolated with LSTM LMs.
- The performance of the WSN LMs alone is not reported and compared.

Whole sentence neural (WSN) LMs

Table: WERs on SWB and Conversational Interaction (CI) task

	SWB	CI
n-gram	6.9	8.5
+ word LSTM	6.5	8.5
+ whole sentence LM	6.3	8.3

- WSN LMs are complementary with n-gram LMs and LSTM LMs, and interpolating the WSN LMs with the LSTM and n-gram LMs can further reduce the WERs.
- The performance of the WSN LMs alone is not reported and compared.

Summary

Advantages of UGM based LMs

- Flexible in using features.
- Avoid label bias.
- Inference is fast, by avoiding local normalizations.

Challenges

Training and normalizing are challenging.

Models	Formulation	Training methods
WSME	$p(s) = \frac{1}{Z} \exp\left(\sum_i \lambda_i f_i(s)\right)$	Gibbs / Metropolis / Importance sampling to approximate the expectation
TRF	$p(j, x^j; \theta) = \frac{\pi_j}{Z_j(\theta)} e^{\phi(x^j;\theta)}$	AugSA, DNCE
WSN	$p(s) =$ output of neural network on whole sentences	NCE with an elaborate noise generator

Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

Current challenges and future directions



F. Jelinek, 1932 – 2010

- 1985: Every time I fire a linguist, the performance of the speech recognizer goes up.
- 1995: Put language back into language modeling.

Current challenges and future directions




- **Incorporate linguistic constrains, such as part-of-speech (POS) tags, syntactic trees.**

Challenges

- In training (with complete data), we need sentences with linguistic annotations. Data annotating is expensive.
  1. Training with incomplete data is needed.
- In inference (calculating sentence probabilities), the test sentences are usually not annotated.
  1. An extra automatic annotator (such as a parser) is needed.
  2. We need to calculate the marginal probability on the word sequence by marginalizing over the hidden variables.

Current challenges and future directions

- **Random field language modeling.**

Challenges

- Improve the training efficiency of random field LMs, especially on large training set.
- Investigate random field LMs with rich linguistic features.
- Investigate semi-supervised learning of random field LMs with limited annotated data and abundant unannotated data.
- Incorporate the random field LMs into the state-of-the-art ASR systems.

Current challenges and future directions

- **Large vocabulary / Dynamic vocabulary**

Challenges

- Reduce the computational cost and memory cost during LM training and inference in large vocabulary tasks.
- Handle the case where new words of new domains are dynamically added into the vocabulary.
- Investigate open-vocabulary LMs on language without word boundary, such as Chinese.

Contents

1. Introduction
   - language model
   - language model evaluation
2. The directed graphical model (DGM) based LMs
   - n-gram language models
   - Neural network language models
   - Speedup Softmax in NN LMs
   - Open-Vocabulary NN LMs
3. The undirected graphical model (UGM) based LMs
   - Whole-sentence maximum entropy (WSME) LMs
   - Trans-dimensional random field (TRF) LMs
   - Whole sentence neural (WSN) LMs
4. Current challenges and future directions
5. LM open-source toolkit

LM open-source toolkit

We provide open-source codes and hands-on exercises to help the learners to get familiar with the state-of-the-art techniques of language modeling.
toolkit: https://github.com/wbengine/TRF-NN-Tensorflow

- Written by Python 3.0+ based on Tensorflow
- Support different state-of-the-art language modeling methods
  - DGM: n-gram LMs, LSTM LMs
  - UGM: TRF LMs
- Experiments on different dataset of different size
  - Penn Treebank dataset (small)
  - Google billion word dataset (large)
  - HKUST mandarin dataset (small)
  - ......
- Evaluation by PPLs and WERs

LM open-source toolkit

There are several typical LM exercises.
Exercise 1: LMs on Penn Treebank

LMs:

- n-gram LMs with Modified Kneser-Ney smoothing
- LSTM LMs with different hidden layer sizes and dropout settings.
- TRF LMs with discrete features or neural network features, trained by AugSA or DNCE.

Evaluation

- PPLs on Penn Treebank test set
- Rescore n-best lists generated by recognizing the WSJ'92 test set to calculate the WERs.

LM open-source toolkit

Exercise 2: LMs on Google billion word dataset ($|V| \approx 800K$)

LMs:

- Distributed training of n-gram LMs.
- LSTM LMs with shortlist.
- LSTM LMs with adaptive Softmax
- LSTM LMs with B-NCE training
- TRF LMs with AugSA or DNCE training

Evaluation

- PPLs on the test set
- Rescore n-best lists generated by recognizing the WSJ'92 test set to calculate the WERs.

Once said in : J. Goodman, "A bit of progress in language modeling", Computer Speech & Language, 2001.

11.2 All hope abandon, ye who enter here

In this section,[10] we argue that meaningful, practical reductions in word error rate are hopeless. We point out that trigrams remain the de facto standard not because we don't know how to beat them, but because no improvements justify the cost. We claim with little evidence that entropy is a more meaningful measure of progress than perplexity, and that entropy improvements are small. We conclude that most language modeling research, including ours, by comparing to a straw man baseline and ignoring the cost of implementations, presents the illusion of progress without the substance. We go on to describe what, if any, language modeling research is worth the effort.

Begin long rambling cynical diatribe – no results or particularly novel ideas. Grad students thinking about research in language modeling *should* read this section

11.2.1 Practical word error rate reductions are hopeless

Most language modeling improvements require significantly more space than the trigram baseline compared to, and also typically require significantly more

Now we can beat n-gram significantly by RNN (Mikolov, 2010) and TRF (2015)

References I

[1] Stanley F. Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," Computer Speech & Language, vol. 13, pp. 359–394, 1999.

[2] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals,"Recurrent neural network regularization," arXiv:1409.2329, 2014.

[3] Rafal Jozefowicz, et al., "Exploring the limits of language modeling," arXiv:1602.02410, 2016.

[4] Gakuto Kurata, et al., "Empirical exploration of novel architectures and objectives for language models," INTERSPEECH, 2017.

[5] Edouard Grave, et al. "Efficient softmax approximation for gpus," arXiv:1609.04309, 2016.

[6] Shankar Kumar, et al., "Lattice rescoring strategies for long short term memory language models in speech recognition," ASRU, 2017.

References II

[7] Xie Chen, et al., "Recurrent neural network language model training with noise contrastive estimation for speech recognition," ICASSP, 2015.

[8] Kazuya Kawakami, Chris Dyer, and Phil Blunsom, "Learning to create and reuse words in open-vocabulary neural language modeling," arXiv:1704.06986, 2017.

[9] Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu, "Whole-sentence exponential language models: a vehicle for linguistic-statistical integration," Computer Speech & Language, 2001.

[10] Yinghui Huang, et al., "Whole sentence neural language models," ICASSP, 2018.

[11] Bin Wang, Zhijian Ou, Zhiqiang Tan. "Trans-dimensional Random Fields for Language Modeling". ACL, 2015.

References III

12. Bin Wang, Zhijian Ou, Zhiqiang Tan. "Learning Trans-Dimensional Random Fields with Applications to Language Modeling". IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2018, 40(4): 876-890.
13. Bin Wang, Zhijian Ou. "Language Modeling with Neural Trans-dimensional Random Fields". ASRU, 2017.
14. Bin Wang and Zhijian Ou, "Learning neural trans-dimensional random field language models with noise-contrastive estimation," ICASSP, 2018.
15. Bin Wang and Zhijian Ou, "Improved training of neural trans-dimensional random field language models with dynamic noise-contrastive estimation", Spoken Language Technology (SLT), 2018

Questions