# 基于跨维随机场的统计语言模型

（申请清华大学博士学位论文）

培 养 单 位：电 子 工 程 系
学　　科：信 息 与 通 信 工 程
研　究　生：王　　斌
指 导 教 师：欧　智　坚　副　研　究　员

二〇一八年六月

# Statistical Language Models Based on Trans-dimensional Random Fields

Dissertation Submitted to

**Tsinghua University**

in partial fulfillment of the requirement

for the degree of

**Doctor of Philosophy**

in

**Information and Communication Engineering**

by

**Wang Bin**

Dissertation Supervisor : Associate Professor Ou Zhijian

**June, 2018**

# 关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：(1) 已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；(2) 为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；(3) 根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

**（保密的论文在解密后应遵守此规定）**

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

# 摘 要

统计语言模型旨在使用统计的方法建模自然语言句子的联合概率，它为各种人工智能任务提供自然语言约束，如自动语音识别和机器翻译。目前主流的语言建模方法是条件式模型，即将词序列的联合概率表示成条件概率的连乘。鉴于条件式模型在语言建模中存在的问题，本文提出了一种新颖的联合概率模型——跨维随机场（Trans-dimensional Random Field，TRF）模型，模型将变长序列看做变维空间的向量，通过定义一族随机场模型实现了对变长序列的联合产生式建模。相比于条件式语言模型，TRF 的优势主要体现在如下三点：首先，TRF 作为无向图模型，从理论上避免了条件式模型存在的标签偏置（Label Bias）的问题；其次，TRF 可以灵活的支持各种复杂的特征，包括离散特征和神经网络特征；最后，TRF 由于避免了局部归一化，模型的推理效率相比于神经网络语言模型显著提高。

TRF 的挑战源自模型的训练。由于需要计算模型的归一化系数和模型分布下的期望，模型的对数似然和它的梯度无法精确求解。本文针对 TRF 模型的训练进行了深入的研究，在随机近似的理论框架下提出了增强随机近似算法（Augmented Stochastic Approximation，AugSA），通过产生满足模型分布的跨维样本来联合估计模型的参数和归一化系数；在此基础上，为了使用复杂神经网络特征，本文将联合随机近似（Joint Stochastic Approximation，JSA）的思想引入到 AugSA 算法中，提出了 JSA+AugSA 算法；为了进一步提高训练算法的收敛性，本文探究了将噪声对比估计（Noise-contrastive Estimation，NCE）用于 TRF 训练，结合一些改进技术，提出了 JSA+NCE 算法，大幅度降低了模型的训练时间，同时进一步提升了模型性能。

本文成功将 TRF 应用于语言建模中，并在语音识别任务中对模型性能进行了评估。最终实验表明，TRF 模型在语音识别中的性能显著超越经典的 *n*-gram 语言模型，相对错误率下降最高达 16%；相比于神经网络语言模型，本文提出的使用双向 LSTM 特征的 LSTM-TRF 语言模型仅仅使用 4% 的参数即可获得与 LSTM 语言模型相近的结果，同时推理效率比 LSTM 语言模型提高了约 114 倍。本文首次将随机场方法的应用从定维情形（如图像）拓展到跨维情形（如语言模型），为变长序列的产生式建模在条件概率模型之外开辟了一条新的路径。

**关键词：**语言模型，随机场，随机近似，噪声对比估计

# Abstract

Statistical language models (LMs), which estimate the joint probabilities of natural sentences, form a crucial component in many artificial intelligence applications, such as automatic speech recognition and machine translation. The dominant approach for language modeling is conditional probability models, where the joint probability of a word sequence is factored into a product of local conditionals. Alternatively, this thesis proposes a novel approach, called trans-dimensional random field (TRF), which treats sequences of varying lengths as vectors of different dimensions. By explicitly mixing a collection of random fields, TRFs directly fit the joint probabilities of natural sentences of varying lengths. Compared with the conditional approach, TRFs have the following three advantages. First, as undirected graphical models, TRFs can theoretically overcome the label bias problem inherent in the conditional approach. Second, TRFs can flexibly support any kinds of complex features, including the discrete features and the neural network features. Third, by avoiding local normalization, TRFs significantly improve the inference efficiency over the neural network based language models.

Training TRFs is challenging. Calculating the log-likelihood and its gradients are generally intractable, because these involve calculating the normalization constants and the expectation with respect to the model distribution. This thesis thoroughly studies the model estimation of TRFs. In the framework of stochastic approximation (SA), we develop an effective training algorithm, called the augmented stochastic approximation (AugSA), which jointly estimates the model parameters and the normalization constants by generating the observations of different dimensions from the model distribution. Then in order to train TRFs with neural network features, the idea of joint stochastic approximation (JSA) is introduced to the basic AugSA method and we propose the JSA+AugSA method. To further improve the convergence of the training algorithms, noise-contrastive estimation (NCE) is used to train TRFs. With some modifications, we propose the JSA+NCE method, which significantly reduces the training time and further improves the model performance.

In this thesis, the TRF approach is successfully applied in language modeling and evaluated in terms of speech recognition word error rate (WER). Particularly, TRF LMs outperform the traditional ngram LMs, with as much as 16% relative WER reduction. Compared with the neural network approach, after incorporating the bidirectional LSTM

features into TRFs, the LSTM-TRF LM performs close to LSTM LMs with only 4% parameters, and are about 114x faster than LSTM LMs in computing sentence probabilities. In this thesis, it is the first time to extend the random field approach from the fix-dimensional setting (e.g. images) to the trans-dimensional setting (e.g. natural languages). This work provides an alternative approach for generative modeling of varying-length sequences beyond the dominant conditional probability models.

**Supporting Publications**

1. Bin Wang, Zhijian Ou, Jian Li, Akinori Kawamura. Joint-Character-POC N-Gram Language Modeling For Chinese Speech Recognition. International Symposium on Chinese Spoken Language Processing (ISCSLP), 2014.
2. Bin Wang, Zhijian Ou, Zhiqiang Tan. Trans-dimensional Random Fields for Language Modeling. Annual meeting of the Association for Computational Linguistics (ACL), Long Paper, 2015.
3. Bin Wang, Zhijian Ou, Zhiqiang Tan. Learning Trans-Dimensional Random Fields with Applications to Language Modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2017, 40(4): 876-890.
4. Bin Wang, Zhijian Ou. Language Modeling with Neural Trans-dimensional Random Fields. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2017.
5. Bin Wang, Zhijian Ou. Learning Neural Trans-dimensional Random Field Language Models with Noise-contrastive Estimation. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.
6. Bin Wang, Zhijian Ou. Improved training of neural trans-dimensional random field language models with dynamic noise-contrastive estimation. IEEE Workshop on Spoken Language Technology (SLT), 2018.

**Key words:** language model; random field; stochastic approximation; noise-contrastive estimation

# 目 录

**第 1 章 引言** ..... 1
1.1 研究的目的及意义 ..... 1
1.2 论文贡献和创新点 ..... 4
1.3 论文章节安排 ..... 4
**第 2 章 统计语言模型概述** ..... 6
2.1 语言模型评估指标 ..... 7
2.1.1 混淆度 ..... 7
2.1.2 错误率 ..... 8
2.2 *n*-gram 语言模型 ..... 10
2.2.1 Skipping 技术 ..... 11
2.2.2 Cache 技术 ..... 12
2.2.3 class-based 语言模型 ..... 13
2.3 最大熵模型 ..... 14
2.4 神经网络语言模型 ..... 15
2.5 整句最大熵语言模型 ..... 17
2.6 使用句法信息进行自然语言建模 ..... 18
2.7 本章小结 ..... 20
**第 3 章 跨维随机场语言模型** ..... 21
3.1 引言 ..... 21
3.2 模型定义 ..... 24
3.3 模型训练 ..... 25
3.3.1 最大似然估计 ..... 25
3.3.2 归一化系数的估计 ..... 26
3.3.3 随机近似理论 ..... 27
3.3.4 跨维混合采样 ..... 31
3.4 模型分析 ..... 34
3.4.1 TRF 与条件概率模型 ..... 34
3.4.2 TRF 与 WSME ..... 37
3.5 算法优化技术 ..... 40
3.5.1 学习速率 ..... 40

3.5.2 缩放矩阵的选择 ..... 40
3.5.3 维度分布 ..... 43
3.5.4 引入类别信息加速采样 ..... 44
3.5.5 超高维样本的处理 ..... 45
3.5.6 并行采样 ..... 46
3.5.7 参数正则和终止判决 ..... 46
3.6 仿真实验：词形建模 ..... 49
3.7 语言模型实验 ..... 51
3.7.1 模型插值 ..... 52
3.7.2 PTB 数据集上语言建模实验 ..... 52
3.7.3 中等规模语料语言建模实验 ..... 58
3.8 本章小结 ..... 60
**第 4 章 神经跨维随机场语言模型** ..... 62
4.1 引言 ..... 62
4.2 神经跨维随机场 ..... 64
4.2.1 CNN-TRF ..... 65
4.2.2 RNN-TRF ..... 68
4.2.3 Mix-TRF ..... 70
4.3 JSA+AugSA 算法 ..... 71
4.3.1 算法描述 ..... 71
4.3.2 使用辅助分布的跨维混合采样 ..... 74
4.3.3 PTB 数据集上语言模型实验 ..... 75
4.4 JSA+NCE 算法 ..... 78
4.4.1 噪声对比估计（NCE） ..... 78
4.4.2 JSA+NCE 算法 ..... 80
4.4.3 算法仿真：短词建模实验 ..... 82
4.4.4 PTB 数据集上语言模型实验 ..... 83
4.4.5 HKUST 中文语音识别实验 ..... 85
4.4.6 CHiME-4 语音识别实验 ..... 87
4.5 本章小结 ..... 89
**第 5 章 总结和展望** ..... 91
5.1 总结 ..... 91
5.2 展望 ..... 93

参考文献 ..... 95
致 谢 ..... 100
声 明 ..... 101
个人简历、在学期间发表的学术论文与研究成果 ..... 102

# 主要符号对照表

| | |
|---|---|
| Adam | 自适应随机梯度下降法 |
| AugSA | 增强随机近似（Augmented Stochastic Approximation） |
| CER | 字错误率（Character Error Rate） |
| CNN | 卷积神经网络（Convolutional Neural Network） |
| CRF | 条件随机场（Conditional Random Field） |
| FNN | 前馈神经网络（Forward Neural Network） |
| JSA | 联合随机近似（Joint Stochastic Approximation） |
| LM | 语言模型（Language Model） |
| LSTM | 长短时记忆（Long Short-term Memory） |
| MCMC | 马氏链蒙特卡洛（Markov Chain Monte Carlo） |
| MH | Metropolis-Hastings 采样方法 |
| MTMIS | 基于多次尝试的 Metropolis 独立采样（Multiple-trial Metropolis Independence Sampling） |
| NCE | 噪声对比估计（Noise-contrastive Estimation） |
| PPL | 混淆度（Perplexity） |
| RJ-MCMC | Reversible-jump Markov Chain Monte Carlo |
| RNN | 递归神经网络（Recurrent Neural Network） |
| SA | 随机近似（Stochastic Approximation） |
| SGD | 随机梯度下降法（Stochastic Gradient Descent） |
| TRF | 跨维随机场（Trans-dimensional Random Field） |
| WER | 词错误率（Word Error Rate） |
| WSME | 整句最大熵（Whole-Sentence Maximum Entropy） |
| discrete-TRF | 使用离散特征的跨维随机场 |
| neural-TRF | 使用神经网络特征的跨维随机场 |

# 第 1 章　引言

## 1.1　研究的目的及意义

语言和认知是随机现象[1]，这个认识已得到广泛肯定和践行。因此，为自然语言建立统计模型，是人工智能研究的一项重要基础问题。所谓统计语言模型，旨在用统计的方法对自然语言的概率分布进行建模，它为诸如自动语音识别、机器翻译、问答系统等各种人工智能任务提供语言学约束，是诸多与自然语言相关的人工智能系统的重要组成部分。在语言模型研究中，一般将自然语句表示成词序列的形式，即 $x_1, x_2, \ldots, x_l$，其中 $x_i$ 表示第 $i$ 个位置的词，$l$ 表示句子的长度，从而对词序列的联合概率 $p(x_1, \ldots, x_l)$ 进行建模。因此，语言模型本质上是一个（变长）序列建模问题。

对序列建模，目前的绝大多数做法是利用马氏性，建模成一个条件式模型，将序列的联合概率表示成条件概率的连乘，

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

其中 $\psi(\cdot)$ 表示将历史序列缩减映射成等价类，从而能有效的平滑估计条件概率。不同条件式语言模型的区别体现在对于等价类映射函数的选择和对于条件概率的估计方式上。例如，经典的 $n$gram 语言模型使用 $n$ 阶马氏链假设，仅考虑前 $n-1$ 个词对当前词的影响，即 $\psi(x_1, \ldots, x_{i-1}) = x_{i-n+1}, \ldots, x_{i-1}$，同时使用平滑算法[2] 来进行条件概率的参数估计；最大熵语言模型[3] 在 $n$ 阶马氏链假设的基础上，将 $n$ 阶条件概率定义成一个对数线性模型，从而通过最大似然准则来进行参数估计。近年来逐渐兴起的各种基于神经网络的语言模型亦是条件式语言模型，通过利用神经网络（例如，前馈神经网络语言模型[4] 或递归神经网络语言模型[5]），将历史信息编码为连续空间的向量，并在连续空间进行条件概率的参数估计。

目前，递归神经网络语言模型及其各种变形（如 LSTM 语言模型[6]，基于 Highway 的 LSTM 语言模型[7]，双向 LSTM 语言模型[8] 等）取得了优异的性能，无论是从模型混淆度还是语音识别错误率上，都显著超越传统的 $n$gram 语言模型。这一方面是由于递归神经网络可以编码更长的历史信息，另一方面得益于神经网络的平滑能力。然而，当前语言模型依然有很大的提升空间。举例来说，在本文的语音识别实验中（表4.6），在第一遍识别生成的 1000 候选句子中与正确句子最匹配的

句子的词错误率仅为 0.93%（称为 Oracle error rate），而即使是利用目前最好的神经网络语言模型——LSTM 语言模型对候选句子进行重新打分（Rescoring）得到的错误率依然高达 7.36%，其间的差距大概就是目前语言模型可以提升的性能空间。尽管取得了一定成功，可以发现目前的条件式序列模型在自然语言建模中存在以下三方面的缺陷，制约了语言模型性能的进一步提高。

**首先，条件概率模型是一个局部归一化模型，局部归一化模型存在典型的“标签偏置（Label Bias）”问题**。标签偏置是指条件模型对当前词的预测仅仅与相同历史条件下的概率做比较，而不与其他历史条件下的概率做比较[9]，换句话讲，由于模型在每个位置处的权重是被归一化的，因此即使历史中存在错误，模型依然可能提供一个较高的条件概率。标签偏置问题广泛存在与各类局部归一化模型（或称之为有向图模型）中，为了解决这一问题人们进行了大量的研究。最初人们提出了条件随机场（Conditional Random Field，CRF)[9]，使用一个无向图描述观测序列和状态序列间的约束以及状态序列内部的约束，从而解决了最大熵马尔科夫模型（Maximum-entropy Markov Model，MEMM）中存在的标签偏置问题；在基于神经网络的模型中，人们研究使用全局归一化的方式来避免传统的 Seq2seq 模型中存在的标签偏置问题[10]；此外，Sam Wiseman 等人提出了一个基于 Beam Search 的损失函数来直接对模型得分进行优化[11]，从而达到避免标签偏置的效果。

**其次，条件概率模型限制了对于复杂特征，尤其是句子级别的特征的使用**。自然语言是以成分为基本的组成单元[12]，语言中的词可以组成短语，短语可以组成子句，自然语言是基于这些成分之间的约束来拓展的。如何在语言模型中引入语言成分之间的约束关系，一直是语言模型研究的重要方向。最初人们研究对词进行归纳分类，将类别约束加入到语言模型中，或者考虑到有约束的语言成分之间可能相距较远，从而引入 Skipping 技术来建模长跨度的约束关系[13]；更进一步，人们提出了结构语言模型[14]，通过对所有的历史词组成的序列做局部成分句法分析获得局部的句法结构，来预测当前词；或者将局部的成分句法特征和经典的 *n*gram 特征相结合，使用最大熵模型进行分布拟合[15]。在最近的研究中，人们使用递归神经网络来对序列化的句法树进行产生式建模，从而可以建模句子和句法树的联合概率[16]。但是在条件概率的框架下，模型使用复杂的特征，尤其是句子级别的特征存在很大的局限性。

**最后，条件概率模型的局部归一化操作影响模型的推理效率，这点在神经网络语言模型中尤为突出**。在神经网络语言模型中，局部归一化需要计算 Softmax，由于语言建模任务中词典规模很大，Softmax 的计算严重影响模型的训练和推理效率，成为神经网络语言模型的计算瓶颈，因此很多研究致力于提高神经网络尤其

是递归神经网络的训练和推理效率。这些研究大致可以分为两类，第一类是通过引入一些假设来减少归一化的计算量。如 LightRNN[17] 将字典中的词分布到一个表中，通过分别预测表的行号和列号来预测当前词，从而减少了归一化的状态空间；SVD-Softmax[18] 则是通过使用奇异值分解来近似计算 Softmax；更为常见的做法是对词进行聚类，通过先对类别进行归一化，再对属于当前类别的词进行归一化的方式来构造一个层级 Softmax 结构[19,20]。这类做法的缺陷是需要改变模型的定义，因此往往会影响模型的性能。另一类做法是将归一化系数当做参数，然后使用一些特定的算法来同时估计参数和归一化系数。而这种做法的问题是条件模型的归一化系数过多，将每个归一化系数都储存下来是不可能的，因此实际中往往将所有的归一化系数都固定为一个经验值，然后使用诸如噪声对比估计（NCE）[21] 或自归一化算法[22] 来进行参数估计。然而这种做法往往过于依赖经验结论，随着数据集或训练任务的改变可能会存在很大的偏差。

针对上述问题，本文提出了跨维随机场（Trans-dimensional Random Field，TRF）模型[23,24]。这里的“维度”在语言模型任务中表示句子的长度，“跨维”意味着对变长句子进行建模。不同于条件式模型对条件概率进行建模，TRF 模型直接建模整句话的联合概率，即

$$p(x_1, \ldots, x_l) = \frac{\pi_l}{Z_l(\theta)} e^{\phi(x_1,\ldots,x_1;\theta)} \tag{1-1}$$

其中，$\pi_l$ 是模型的先验长度概率，$\phi(x_1, \ldots, x_1; \theta)$ 称之为随机场的势函数，通过编码整句话中的特征来返回一个实数，$\theta$ 为待估计参数，$Z_l(\theta)$ 是长度 $l$ 的归一化系数，即 $Z_l(\theta) = \sum_{(x1,\ldots,x_l)} e^{\phi(x_1,\ldots,x_l;\theta)}$。**TRF 模型可以很好的解决上述提到的条件式模型的三个问题**。首先，TRF 模型是一个全局归一化模型，由于模型避免了局部归一化，因此可以从理论上避免标签偏置的问题；其次，TRF 是整句语言模型，这使其可以灵活的支持各种特征，乃至句子级别的特征；最后，由于 TRF 是全局归一化模型，避免了繁琐的局部归一化操作，推理效率相比条件式模型显著提高。

然而，TRF 模型的训练存在很大的挑战。由于需要计算模型的归一化系数和模型分布下的期望，模型的对数似然和它的梯度往往无法精确求解。本文对 TRF 模型的训练方法进行了深入的研究，提出了一系列的模型训练方法，包括增强随机近似（AugSA）训练算法、JSA+AugSA 训练算法和 JSA+NCE 训练算法，并且在多个语音识别任务中验证了 TRF 语言模型和训练算法的有效性。

该研究首次将随机场的应用从定维情形扩展到跨维情形，并提出了行之有效的模型训练方法，这为语言模型乃至一般的序列建模问题开辟了一条新的思路。本文的相关工作先后发表于 ACL 2015 长文（Annual Meeting of the Association for

Computational Linguistics, Long Paper）、PAMI 2017（IEEE Transactions on Pattern Analysis and Machine Intelligence）、ASRU 2017（IEEE Automatic Speech Recognition and Understanding Workshop）、ICASSP 2018（IEEE International Conference on Acoustics, Speech and Signal Processing）顶级国际期刊和国际会议中。

## 1.2　论文贡献和创新点

本文的贡献和创新点概括如下：

(1) 提出了跨维随机场模型，模型避免了条件式模型存在的标签偏置的问题，可以灵活的支持离散特征和神经网络特征，并且避免了局部归一化进而使得模型的推理效率显著提高。

(2) 提出了有效的跨维随机场模型训练方法，包括：
   - 增强随机近似（AugSA）算法。依托于随机近似的基本理论，通过基于 MCMC 采样产生满足模型分布的样本来计算模型分布下的期望，来联合估计参数和归一化系数；
   - JSA+AugSA 算法。将联合随机近似（JSA）[25] 的思想和 AugSA 算法相结合，定义一个动态调整的辅助分布来作为采样的提议分布，从而有效提高了 AugSA 算法的采样和训练效率；
   - JSA+NCE 算法。将联合随机近似（JSA）的思想与噪声对比估计（NCE）相结合，首先使用了一个动态调整的噪声分布来产生样本，其次对噪声分布和噪声与数据分布的插值分布做鉴别，既提高了算法的稳定性和收敛效率，又避免了因数据样本的不充分性而造成的过拟合。

(3) 成功将跨维随机场模型应用于语言建模任务中，通过语音识别错误率对语言模型性能进行评估，取得如下实验结果：
   - TRF 语言模型在语音识别中的性能显著超越经典的 *n*gram 语言模型，PTB 英文实验中相对错误率下降达 16%；
   - TRF 语言模型仅仅使用 4% 的参数即可获得与 LSTM 语言模型相近的识别性能，且推理效率比 LSTM 提高约 114 倍；
   - TRF 语言模型拥有良好的语言无关性，在 HKUST 普通话数据集上依然可以保持性能的优势。

## 1.3　论文章节安排

论文的章节安排如下。第2章对语言模型的研究现状进行概述，包括模型的评估指标和一些经典的语言建模方法。第3章介绍了跨维随机场（TRF）模型，提出

了增强随机近似（AugSA）训练算法和跨维混合采样算法，本章将包含离散特征的 TRF 模型应用于语言建模任务中，进行了一系列的实验展示了 TRF 语言模型的性能，并与已有的基线模型进行了对比。第4章在 TRF 基本理论框架的基础上进行了更深层的研究，将神经网络特征引入到 TRF 模型中，从而提出了神经跨维随机场（neural-TRF）模型，同时提出了两种新的模型训练算法 JSA+AugSA 算法和 JSA+NCE 算法。第5章对本文的工作进行总结并对未来工作进行展望。

# 第 2 章　统计语言模型概述

统计语言模型是很多自然语言相关的人工智能系统的重要组成部分，它旨在使用统计的方法来拟合自然语言句子的联合概率，力求对自然语言的正确度和合理度给予一个数值化的衡量，从而为各种机器学习任务提供自然语言约束，如自动语音识别、手写文字识别、机器翻译、信息监测等等。为了能够方便的定义句子的联合概率，一般将自然语言的句子切分词序列的形式，即 $x_1, x_2, \cdots, x_l$，其中 $x_i$ 表示为第 $i$ 个位置的词，$l$ 表示序列的长度。值得一提的是，虽然这种切分对于英语这些自动分词的语言显得很直接，但是对于汉语这种以字为单位的语言就需要额外引入分词操作来完成对语料数据的分词，虽然实际中也可以直接将汉语句子看成是字序列来处理，但是经验结论显示，将汉语分词后建模词模型无论是从识别系统的效率还是从识别性能上都要比直接使用字模型要好。为了符号统一，下文统一使用词序列这种表示方法。语言模型的任务即为拟合给定词序列的联合概率 $p(x_1, x_2, \cdots, x_l)$，因此语言建模本质上是一个变长序列的建模问题。

目前主流的语言建模方法为条件概率模型，即通过考虑序列的马氏行而将序列的联合概率表示成条件概率的连乘，进而估计每个条件概率，不同语言建模方法的一个主要区别在于如何拟合条件概率。目前主流的条件概率的模型包括：$n$ 元文法模型（$n$-gram 模型）、最大熵语言模型、神经网络语言模型等。具体而言，$n$-gram 语言模型引入 $n$ 阶马氏链假设，即假设当前词仅仅依赖于前 $n-1$ 个词，通过估计 $n$ 阶转移概率 $p(x_i|x_{i-n+1}, \cdots, x_{i-1})$ 来建模自然语言序列；最大熵语言模型在 $n$ 阶马氏链假设的基础上，定义了一个指数模型来建模条件概率，通过最大化模型在训练集上的似然，将参数估计转换为一个凸优化问题，进而可以通过梯度相关的优化算法来进行参数求解；神经网络的语言模型则是依托于神经网络强大的非线性拟合能力，通过使用词向量将离散的词映射到连续向量空间，进而构建如前馈神经网络、递归神经网络或卷积神经网络来对条件概率进行建模。

与条件概率模型相对的是联合概率模型，即直接估计词序列的联合概率 $p(x_1, x_2, \cdots, x_l)$，这样做的初衷是为了能够自然的使用句子级别的特征，而不受条件概率建模的限制。一个典型的联合概率模型是整句最大熵（WSME）语言模型，它将序列的联合概率表示成指数模型形式，然后在整个变长序列空间进行归一化。本文提出的跨维随机场（TRF）语言模型也属于联合概率模型，与整句最大熵模型不同的是，TRF 定义了一个指数混合模型来对变长序列进行建模，即为每一个长度定义了一个指数模型，而不同长度对应的指数模型共享特征和参数，这样做保

证了模型的长度分布的合理性，使得模型的参数估计和在模型分布下采样均变得更加高效。

本章首先介绍统计语言模型的评估指标——混淆度和错误率，对比了两者的优劣，之后依次介绍一些典型的语言建模的方法，包括条件概率语言模型和联合概率模型，最后，本章对将句法信息应用于语言建模方面的研究进行介绍。

## 2.1　语言模型评估指标

统计语言模型有两种评估指标——混淆度和错误率，两种评估指标各有利弊。

### 2.1.1　混淆度

测试集上的混淆度 (Perplexity, PPL) 的定义如下：

$$\mathrm{PPL} = \exp\left\{\frac{-\log P}{W}\right\} \tag{2-1}$$

其中，$\log P$ 表示测试集上的对数似然值，通过对每句话的对数概率求和获得，$W$ 表示测试集的总词数。混淆度越小，说明模型对语言建模越准确。

从混淆度的定义上可以看到，对于给定的测试集（即 $W$ 固定），混淆度与对数似然 $\log P$ 成反比，这说明混淆度越小，似然值越大，模型越好，即所谓的最大似然准侧。最大似然是统计上行常用的一种估计方法，直观来讲即使模型在正样本上的概率尽可能的大，在负样本上的概率尽可能的小。对于语言建模这一特定任务，获得正样本（即符合语法的句子）很容易，比如我们可以很方便的通过网络爬虫获得大量文本，而负样本（即不符合语法的句子）却很难获得，因此实际中往往只需要然模型在语料上的概率尽可能的大。事实上，统计语言模型的一般就是通过最大化训练集上的似然来进行参数估计的，而之所以使用混淆度（PPL）而不是直接使用似然值来进行语言模型的评估，主要有如下两个原因：

(1) 混淆度有比较直观的意义。混淆度的提出最初是基于 n-gram 语言模型，即条件概率模型，在这种情况下，混淆度的定义可以改写为

$$\mathrm{PPL} = \exp\left\{\frac{1}{W}\sum_{i=1}^{W} -\log p(x_i|x_{-i})\right\} \tag{2-2}$$

其中 $x_i$ 表示第 $i$ 给位置的词，$x_{-i}$ 表示从第 1 个位置到第 $i-1$ 个位置的所有历史词，$p(x_i|x_{-i})$ 即为建模的条件概率（为了方便起见，这里假设数据集中的句子首尾相连拼接在一起，一个数据集即为长度为 $W$ 的一句话）。在这种

定义下，混淆度有一个比较直观的意义，即为给定历史的条件下，平均看来每个位置可能的词的数目。如果这个值越小，例如模型的混淆度等于 1，则说明在历史确定的情况下，对当前词的预测有且仅有一种可能，这说明模型的预测能力非常的准。

(2) 混淆度将似然值投影到一个便于比较的取值范围。例如，目前主流用于语言模型的混淆度取值大概在 100 左右，汉语会稍高一些，大概在 300 左右，在这个范围内进行模型的对比会比较直观。

混淆度主要的优点就是评估简单，只需要提供一个测试语料即可以进行计算，但使用混淆度评估语言模型也存在如下缺点：

(1) 对条件概率模型而言，混淆度的评估是基于准确的历史词，而在语音识别任务中，由声学模型产生的候选可能存在错误，在这种带错误的情况下的模型性能便无从评估。

(2) 混淆度需要模型满足归一化，这在条件概率模型的情况下很容易满足，而对于整句语言模型，由于归一化系数无法精确求解，因此混淆度无法计算，既是使用估计的归一化系数也会造成混淆度的巨大偏差。

(3) 混淆度在整句语言模型的定义存在混淆，这主要是由开始符和结束符所造成的。在实际语言建模中，为了能够标定句子的开始和结尾，通常会在每句话的开头和结尾分别添加一个开始符和结束符，并把它们当做普通的词来处理。对于条件概率模型，计算混淆度的时候会考虑对结束符的条件概率，忽略对开始符的条件概率，因此总词数 $W$ 为语料的实际词数加上结束符数目（即句子的数目），而对于整句语言模型，由于概率是对一句完整的句子进行计算的，开始符和结束符的作用都包含在内，因此总词数 $W$ 应该如何计算就存在混淆，从而导致整句语言模型与条件概率模型无法使用混淆度进行有效的对比。这种情况下便需要引入下一个评价指标——错误率。

### 2.1.2　错误率

错误率是指将语言模型应用于语音识别系统中，识别系统最终的错误率，根据句子的切分规则不同，又分为词错误率（Word Error Rate, WER）和字错误率（Character Error Rate, CER）。WER 的定义如下：

$$\mathrm{WER} = \frac{I_w + D_w + S_w}{N_w} \tag{2-3}$$

其中 $I_w$、$D_w$、$S_w$ 分别表示以词为单位的插入错误数、删除错误数和替换错误数，$N_w$ 表示测试集的总词数。为了能够计算错误数目 $I_w$、$D_w$ 和 $S_w$，需要使用动态规

划算法来逐句对识别结果和标准答案进行比对，获得最小的总错误数目（即两个序列的最小距离）。CER 的定义与 WER 类似:

$$\mathrm{CER} = \frac{I_c + D_c + S_c}{N_c} \tag{2-4}$$

其中 $I_c$、$D_c$、$S_c$ 分别表示以字为单位的插入错误数、删除错误数和替换错误数，$N_c$ 表示测试集的总字数目。总的来讲，对于英语这种自动分词的语言，一般使用 WER 来评估语言模型，而对于汉语这种词定义模糊语言，虽然使用的语言模型一般是词模型（先对语料分词，再训练词语言模型），但是最后计算错误率的时候往往使用 CER，这主要有一下几点原因：

(1) 首先，汉语的测试音频的标注一般都没有进行分词，所以无法直接评估 WER，虽然可以使用同样的分词工具对音频标注进行分词，但分词结果毕竟会引入额外的误差；

(2) 其次，识别结果可能会把词拆成单字识别出来，这样虽然词序列存在错误，但是依然可以获得正确的字序列，因此汉语的 CER 一般要低于 WER；

(3) 最后，汉语不同于英语，不会在词之间添加空格，最后的识别结果依然会拼接成字序列输出，因此直接计算 CER 比较符合汉语使用习惯。

相比于混淆度，使用错误率进行评估有明显的优势。首先，错误率是语言模型最终的优化目标，直接反应语言模型在识别系统中的性能；其次，错误率的评估不要求语言模型满足归一化，对于无法精确归一化的语言模型，错误率是唯一合理的评价指标。当然，使用错误率的进行评估也存在一些缺陷：

(1) 错误率的计算比混淆度复杂。计算错误率需要一整套完整的识别系统，包括一个性能足够好的声学模型，测试音频和标注文本，译码器等等，而混淆度仅仅需要文本语料即可；

(2) 错误率的结果与识别任务、识别策略和超参数的设置有很大关系。例如，1-pass 译码的错误率一般比多候选重新打分的识别结果要低，但是很多语言模型，如递归神经网络语言模型、整句语言模型无法使用 1-pass 译码；又如，识别中声学模型的得分与语言模型的得分需要进行加权平均来计算最终的得分，这个加权系数会随着模型的改变而变化，一般需要在额外的开发集上调到一个比较好的值。

## 2.2　$n$-gram 语言模型

定义 $x_1, x_2, \cdots, x_l$ 表示长度为 $l$ 的词序列，其中 $x_i$ 表示为第 $i$ 个位置的词，统计语言模型的任务为建模序列的联合概率，即 $p(x_1, x_2, \cdots, x_l)$。由于词序列的长度是可变的，为了能够对变长序列进行建模，主流的方法是将序列的联合分布表示成条件概率连乘，即：

$$p(x_1, \cdots, x_l) = \prod_{i=1}^{l} p(x_i | x_1, \cdots, x_{i-1}) \tag{2-5}$$

然后通过估计条件概率从而计算出整个序列的联合概率，本文称这种方法为条件概率模型。由于估计出所有可能的条件概率是不可行的，因此 $n$-gram 引入 $n$ 阶马氏链假设，即仅仅考虑前 $n-1$ 个词对当前词的影响：

$$p(x_1, \cdots, x_l) \approx \prod_{i=1}^{l} p(x_i | x_{i-n+1}, \cdots, x_{i-1}) \tag{2-6}$$

因此，$n$-gram 语言模型的目标就是去估计这一系列条件概率。

$n$-gram 语言模型训练是基于最大似然准则，在最大似然准则的情况下，条件概率可以通过计算训练集上的频率来获得，即：

$$p(x_i | x_{i-n+1:i-1}) = \frac{\mathrm{c}(x_{i-n+1:i})}{\mathrm{c}(x_{i-n+1:i-1})} \tag{2-7}$$

其中 $x_{i-n+1:i-1} = x_{i-n+1}, \cdots, x_{i-1}$ 表示从第 $i-n+1$ 到第 $i-1$ 个位置的词序列，$\mathrm{c}(x_{i-n+1:i-1})$ 表示词组 $x_{i-n+1:i-1}$ 在训练集中出现的次数。但是这种简单的处理存在一个严重的问题，即模型会严重过拟合在训练集上，例如，如果某个序列 $x_{i-n+1:i-1} = H$、$x_i = W$ 并没有出现在训练集中，则使用上述公式计算的条件概率 $p(W|H) = 0$。零概率在语言模型中是致命的，首先，由于训练语料通常并不充分，往往很难涵盖所有可能的词搭配，因此零概率的赋予过于严格；其次，在识别任务中，产生的句子往往会存在错误，如果某个条件概率的为零的话，会造成整句话的概率为零，语言模型就无法对包含错误的句子进行评估。为了避免模型严重过拟合在训练集上，就需要使用平滑算法[2] 来对条件概率进行平滑。因此，对 $n$-gram 语言模型来说，平滑算法的选择就是至关重要。

目前最常用的平滑方法是 Modified Keneser-Ney（MKN）平滑，其做法如下：

$$p_{\mathrm{MKN}}(x_i|x_{i-n+1:i-1}) = \frac{\max\{c(x_{i-n+1:i}) - D(c(x_{i-n+1:i})), 0\}}{c(x_{i-n+1:i-1})} + \gamma(x_{i-n+1:i-1}) p_{\mathrm{MKN}}(x_i|x_{i-n+2:i-1}) \tag{2-8}$$

公式中可以看出，$n$ 阶条件概率 $p_{\mathrm{MKN}}(x_i|x_{i-n+1:i-1})$ 是通过一个插值来进行计算的，这个插值包括两部分，第一部分是一个折扣的频率，$D(c(x_{i-n+1:i})$ 是一个折扣常数，通常会随 $x_{i-n+1:i}$ 出现次数的变化而不同。第二部分 $p_{\mathrm{MKN}}(x_i|x_{i-n+2:i-1})$ 是一个 $n-1$ 阶的平滑后的概率，系数 $\gamma$ 是为了确保条件概率的归一化。

这种插值模型有两个主要的优点，首先，MKN 平滑算法避免了 0 概率。如果某个 $n$ 元搭配在训练集中没有出现，则其概率会通过 $n-1$ 阶的概率来计算，以此类推，最终会会退到 unigram 的概率，而 unigram 的概率通常使用一个简单的折扣平滑算法来确保概率都大于 0；其次，插值模型有效的减少了参数的规模。对于一个 $n$-gram 语言模型，模型不需要记录每一个 $n$-gram 的概率，仅仅需要记录在训练集中出现的 $n$-gram 的概率，而对于未出现的 $n$ 阶条件概率，可以通过 $n-1$ 阶条件概率来计算获得，因此模型需要保存的参数数目等于训练集中出现的 1 到 $n$ 元词组的数目加上归一化常数（又称为回退系数）$\gamma(x_{i-n+1:i-1})$ 的数目，这显著降低了模型参数的数目，使得训练较高阶的 $n$-gram 语言模型成为可能。

$n$-gram 语言模型最大的优势就是训练简单快捷，这是由于模型的参数估计仅仅需要统计每个 $n$ 元词组在训练集中出现的数目，然后在做一些简单的平滑运算；其次，$n$-gram 语言模型可以方便的表示成有限状态转换器（WFST）[26] 并嵌入到基于 WFST 的语音识别译码中。因此，$n$-gram 依然是目前使用最为广泛的语言模型。然而，$n$-gram 语言模型的缺点就是模型定义过于简单，这种简单一方面体现在模型仅仅考虑相邻的 $n$ 个词的依赖关系，没有考虑更长跨度的约束，另一方面是因为模型没有使用词信息之外的语言学信息，使得模型无法建模更深层的语言学知识。此外，简单的使用平滑算法也无法对模型进行充分的平滑，使得模型在训练集上的似然值依然远远高于测试集上的似然值。

为了能够应对 $n$-gram 模型存在的一些问题，在 $n$-gram 模型的基础上，针对一些特定的方面，人提出了一些提升的技术，下面依次进行简单的介绍。

### 2.2.1　Skipping 技术

Skipping 技术的提出主要是考虑到自然语言中的这样一个事实，即存在约束关系的词在句中的位置不一定相邻，如句子的谓语一般依赖于句子的主语，而

主谓语之间可能会插入额外的修饰词。为了能够建模这种跨越式的词约束关系，人们提出了 Skipping 技术。以 5 阶模型为例，模型不仅仅建模 5 阶条件概率 $p(x_i|x_{i-4},x_{i-3},x_{i-2},x_{i-1})$，还可以考虑当前词与前 4 个词中的任意 3 个词之间的依赖关系，建模条件概率 $p(x_i|x_{i-4},x_{i-2},x_{i-1})$、$p(x_i|x_{i-4},x_{i-3},x_{i-1})$、$p(x_i|x_{i-4},x_{i-3},x_{i-2})$。同理，还可以考虑当前词与前 4 个词中的任意 2 个词之间的依赖关系，即 $p(x_i|x_{i-4},x_{i-3})$、$p(x_i|x_{i-4},x_{i-2})$、$p(x_i|x_{i-4},x_{i-1})$。这些低阶的条件概率一般需要与传统的 5 阶概率做插值来获得较好的效果。

Skipping 技术的想法虽然简单，但是在实际任务中的效果却不尽如人意，这主要是因为在引入较多的低阶概率之后，模型的参数估计变得更加的复杂，适用于传统 *n*-gram 模型的平滑算法无法在复杂的情况下发挥效果。Skipping 的想法对语言建模很有意义，即词不仅仅依赖于与其相邻的词，还可能依赖于更远距离的词，这种想法促使人们去研究如何使用更长跨度的信息来建模语言模型，因此 Skipping 策略常常被应用于基于特征的语言模型中，如后面提到的最大熵模型，以及本文提到的跨维随机场模型中。

### 2.2.2　Cache 技术

Cache 技术是基于语言在使用时的一个简单的规律：如果一个词被使用，那么在之后不远处，这个词往往会被再次使用，尤其当这个词并不常见，如一些人名、地名、专有名词等。考虑到这个特性，人们研究建立一个临时的模型，仅仅使用最近出现的一些词序列，并且将这个临时的模型与传统的 *n*-gram 模型组合使用，已达到提高性能的效果。

Cache 模型从提高模型混淆度上来看是成功的，由于动态估计的临时模型可以很好的包含语言内容的信息，但是 Cache 模型在识别任务上却表现不佳。主要是因为临时模型过分依赖于一个正确的结果，而识别的结果往往包含一些错误的词，这会使得临时模型估计的偏差很大，从而影响对后续词的预测。

Cache 模型的思路值得考虑，它首先维护一个基础的模型，即 *n*-gram 模型，由于 *n*-gram 模型存在不足，因此建立额外的辅助模型，通过组合两个模型，如线性插值，来获得一个新的模型，新模型便同时具备了 *n*-gram 模型和额外模型的特点。这个思路在语言模型研究中影响很大，它的依据是如果两个模型包含不相关的信息，那么组合后的结果往往会好于两个模型单独的结果。之后的很多的模型研究往往会借助这一特点，一个新提出的模型往往会尝试和 *n*-gram 模型进行组合，以求达到更好的效果。

### 2.2.3　class-based 语言模型

*n*-gram 模型这种单纯的去建模词序列的做法并不能很好的表达语言，举个简单的例子，假设有如下一句话：

“欢迎”“参观”“清华大学”

句中我们已经完成了对词的切分。这句话总共包含三个词，最后一个词“清华大学”表明地点，如果将表地点的词替换换掉其他的地点名字，可以很容的造出更多的类似的句子，如：

“欢迎”“参观”“北京大学”
“欢迎”“参观”“上海”
“欢迎”“参观”“故宫”

这里姑且称语言的这种拓展性为一种“语法规则”，因此，这种规则可以被归纳为：

“欢迎”“参观”⟨地名⟩

其中⟨地名⟩是一个标签，表示此处可以放置任何表示地名的词。如果语言模型能很好的建模建模这种抽象的语法规则，那么及时面对没有在训练集中出现的搭配，如：

“欢迎”“参观”“南京”
“欢迎”“参观”“交通大学”

模型也能够判断它们是符合语言的句子。基于这种思路，人们提出了基于类别的语言模型。首先，人们对词进行聚类，抽象出类别标签（class），如上文提到的⟨地名⟩就是类别标签，而词“北京大学”、“上海”等都属于⟨地名⟩这个标签，然后将类别标签引入到语言模型中，这就是 class-based 语言模型。

实际中，对词进行聚类有两种思路，一种称之为 hard-class，即每个词仅仅属于一个类别，这种情况下不需要训练语料有额外的类别标签，仅仅需在字典中记录每个词对应的类别即可，而词的类别一般也可以通过一些贪婪算法[27] 来自动获得，推理也比较简单，因为给定词序列之后类别序列就已经确定了；另一种分类方式称之为 soft-class，即同一个词根据上下文的不同可能会被归为若干类，词性标注（POS）就属于这 soft-class。这种情况下类别处理起来就比较繁琐，首先训练数据需要有额外的类别标签，其次，推理过程需要对给定词序列的所有可能的类别序列做积分来获得词序列的边缘概率，计算复杂度较大。

为了能够使用类别信息，经典的 class-based 语言模型[28] 基于 hard-class，即每个词被归为固定的一类，然后将条件概率表示成如下形式：

$$p(x_i|x_{i-n+1:i-1}) = p(x_i|c_i)p(c_i|c_{i-n+1:i-1}) \tag{2-9}$$

其中 $c_i$ 表示第 $i$ 个词 $x_i$ 的类别。这种模型的定义类似于隐马模型（HMM），首先通过类别之间的转移概率来预测下一个词的类别，然后根据当前类别预测当前词。这种模型定义虽然可以显著减少参数数目（由于类别的数目比词的数目要小的多），但是却存在明显的缺陷，首先，hard-class 的定义存在很大的混淆，仅仅将一个词归为一类显然无法满足自然语言建模的需求；其次，根据有向图的定义，在给定当前类别 $c_i$ 的情况下，当前词 $x_i$ 的取值与历史统计独立，这种假设显然也过于严格。因此，实际任务中往往需要将 class-based 模型与传统的 $n$-gram 模型做组合来获得模型性能的提升。

虽然简单的构建 class-based 模型并不能提高语言模型的性能，但是这种 class-based 的特征却很好的补充了 $n$-gram 模型的缺陷，同时可以有效的提高模型的平滑性能，因此类别特征尤其是 hard-class 特征往往被用于最大熵模型中，作为 $n$-gram 特征的良好的补充，如 Model M 语言模型[29]。本文也将类别信息应用于跨维随机场语言模型中，从而有效的提高了模型的性能。

## 2.3　最大熵模型

$n$-gram 语言模型使用平滑算法来进行参数估计的，平滑算法往往基于经验结论，从而限制了 $n$-gram 模型的拓展[30]。为了寻求一种更有效的参数估计方法，人们提出了最大熵语言模型（Maximum Entropy，ME）[31][3]，$n$ 阶最大熵模型的定义如下：

$$p_{ME}(x_i|h_i) = \frac{1}{Z(x_{h_i})} e^{\lambda^T f(x_{h_i}, x_i)} \tag{2-10}$$

其中 $h_i = x_{i-n+1}, \cdots, x_{i-1}$ 表示历史词序列，$f(\cdot)$ 是一个特征函数，通过作用于局部 $n$ 元词组 $h_i, x_i$ 上返回一个特征向量，$\lambda$ 是对应的参数向量，$Z(h_i)$ 是归一化系数：

$$Z(h_i) = \sum_{x_i} e^{\lambda^T f(h_i, x_i)} \tag{2-11}$$

这里对最大熵模型做如下几点说明：首先，最大熵模型的提出是为了使模型满

足在训练集上的特征期望等于模型分布下的特征期望的情况下熵最大，理论上，这一最大熵问题等价于最大化指数模型在训练集上是似然[32]。因此实际使用中，最大熵模型是通过对模型 (2-10) 进行最大似然估计来训练参数 $\lambda$ 的。

其次，最大熵模型与 $n$-gram 模型一样，依然基于 $n$ 阶马氏链假设来建模序列，但是相比于 $n$-gram 模型，最大熵模型可以灵活的支持各种局部特征，上文提到的 Skipping 特征、class 特征都可以通过定义对应的特征函数，很方便的嵌入到最大熵模型中，并使通过添加正则项来避免过拟合。

由于最大熵语言模型特征定义的灵活性，根据模型定义的不同从而衍生出很多具体的语言模型。如 Model M[29] 引入了词特征和类特征，同时仔细调节了正则项来获得很好的平滑效果;[15] 将局部句法成分当做特征引入最大熵模型中，力求是模型能够应用更复杂的句法成分信息；而对数线性模型（log-bilinear model）[33] 将词向量（Word Embedding）引入到最大熵模型中，通过考虑词与词之间的二阶依赖关系来对语言进行建模。

## 2.4　神经网络语言模型

近年来，神经网络在各种模式识别领域都取得了巨大的成功，这一方面得益于神经网络理论研究的进步，更归功于矩阵计算硬件（如 GPU）的高速发展，使得快速训练更复杂的神经网络模型成为可能。在语言模型领域，递归神经网络以及其变形形式已经超越传统的 $n$-gram 语言模型成为目前最好的语言模型，这不仅仅是由于神经网络在分布拟合方面和模型平滑方面的先天优势，更是因为递归神经网络可以通过连续空间向量来对所有的历史词进行编码，使得语言模型能够考虑更长跨度的信息，这对于语言建模尤为重要。

神经网络语言模型主要分为两种——前馈网神经网络和递归神经网络。前馈神经网络[4]（Forward Neural Network，FNN）是基于 $n$ 阶马氏链假设，使用神经网络来建模 $n$ 阶条件概率 $p(x_i|x_{i-n+1:i-1})$；而递归神经网络[5]（Recurrent Neural Network，RNN）是通过历史信息编码为一个连续空间向量，进而用来估计条件概率，RNN 的优势在于能够编码更长跨度的信息，因此在语言模型的性能超越了 FNN，成为了目前主流的语言建模方法。而为了能够减轻 RNN 训练过程中出现的梯度消失和梯度膨胀的问题，人们对标准的 RNN 进行了改进，提出 Long Short Term Memory (LSTM）模型[6]，通过引入一个记忆单元与若干逻辑门来控制数据的传递，使得 RNN 的训练更加的鲁棒，因此 LSTM 模型被广泛应用于包括语言模型在内的各种序列建模问题中。图2.1展示了前馈神经网络和递归神经网络的。

神经网络语言模型的参数估计是一个非凸优化问题，旨在最大化训练集上

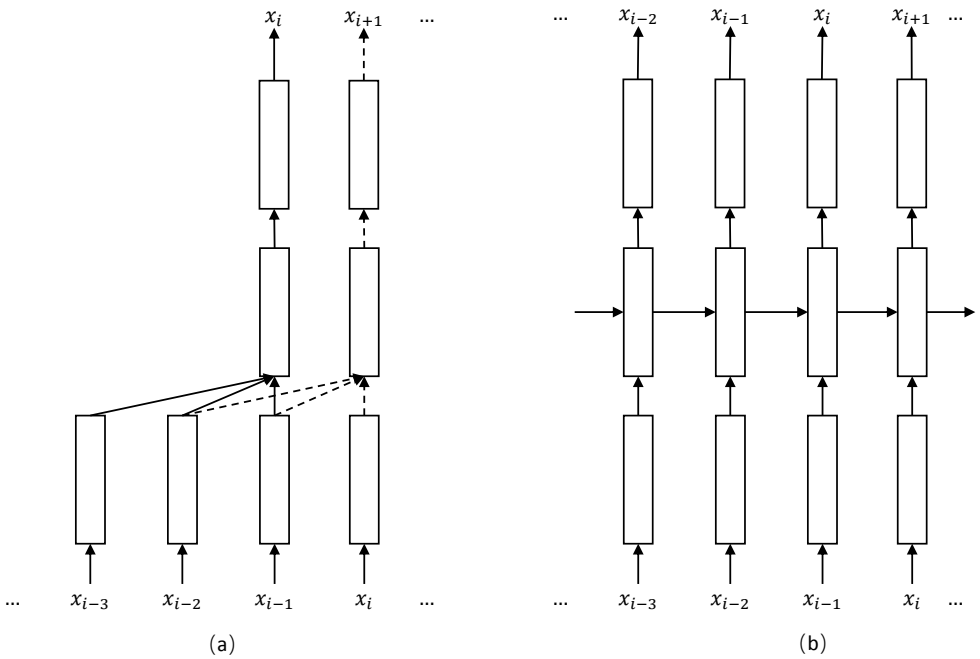



图 2.1　(a) 前馈神经网络语言模型和 (b) 递归神经网络语言模型

的似然，为了能够提高收敛性，人们提出的很多优化算法，如随机梯度下降法（Stochastic Gradient Descent，SGD）、Adam[34]、AdaGrad[35]、RMSProp[36] 以及一些基于二阶距的优化方法如 Hession-Free[37]、拟牛顿法（Quasi-Newton）[38,39]。神经网络的训练同样样面临着过拟合的问题，一般有两种解决方式，第一种方式称之为 Early Stop[19]，即模型参数训练过程实时监测开发集上的混淆度，如果混淆度的没有明显的下降甚至开始上升，则减小学习速率，当学习速率减小到一定程序时就停止训练；另一种方式是 dropout[40]，即在训练过程中随机将一些神经元节点的取值设为 0，这样做是为了每次参数更新的时候仅仅更新一个子网络，从而在训练过程中动态模拟参数平均，进而避免过拟合。

神经网络语言模型的优势主要体现在一下几点：

- 首先，神经网络拥有建模任意复杂分布的能力，这种能力需要在足够的矩阵运算能力的保障下才能得以体现，近年来 GPU 的飞速发展为神经网络提供了良好的数值计算平台；
- 其次，对于与语言模型这一特定任务，神经网络可以将离散的词投影到连续向量空间进行建模，这不仅仅可以减少参数的数目，同时可以对模型进行很好的平滑；
- 最后，递归神经网络，尤其是 LSTM 可以通过连续空间向量来编码更长的历史信息，这对于语言建模这这一任务尤为重要。目前的实验中，LSTM 的语言模型相比于 *n*-gram 语言模型混淆度的相对下降超过 20%，应用于语音识

别中的相对错误率的下降约 10%。

神经网络语言模型属于条件概率模型，即通过序列的马氏性来讲联合概率表示成条件概率的连乘 (2-6)，进而估计每个条件概率，除了条件概率模型的一些限制外，神经网络语言模型的最大问题在于训练和推理效率都很低，这是源自神经网络输出层 Softmax 函数的计算，使得模型需要在词表空间进行归一化，由于语言建模任务中词表很大，从 $10^4$ 到 $10^5$ 不等，使得 Softmax 的计算效率很低，从而严重影响神经网络语言模型的使用效率。为了提高计算效率，人们提出很多改进归一化计算的技术，大致可以分为两类：

- 第一类方式是通过一些近似策略减小 softmax 的计算代价。如 LightRNN[17] 将字典中的词分布到一个表中，通过分别预测表的行号和列号来预测当前词，从而减少了归一化的状态空间；SVD-Softmax[18] 则是通过使用奇异值分解来近似 Softmax 的计算；更为常见的做法是对词进行聚类，通过先对类别进行归一化，再对当前类别下的词进行归一化的方式来构造一个层级 Softmax 结构[19,20]。然而这种做法由于引入了额外的假设，往往会影响模型的性能。
- 另一类做法是将归一化系数当做参数，然后使用一些特定的算法来同时估计参数和归一化系数。而这种做法的问题是条件模型的归一化系数过多，将每个归一化系数都储存下来是不可能的，因此实际中往往将所有的归一化系数都固定为一个经验值，然后使用诸如噪声对比估计（NCE）[41–43] 或自归一化算法[22] 来进行参数估计。然而这种做法往往过于依赖经验结论，随着数据集或训练任务的改变可能会存在很大的偏差。

LSTM 语言模型有很多的变形形式，一些是基于模型本身的改进，如在 LSTM 中添加 Highway 层的 Highway LSTM 语言模型[7]，另一些是基于语言建模这一特定的任务做的改进，如将字信息引入到语言建模中的字模型[44] 或字词混合模型[45]，使用双向序列信息的双向神经网络模型[8,46] 等。值得一提的是，卷积神经网络（Convolutional Neural Network，CNN）也被用于语言建模中[47,48]，CNN 的建模思路与 FNN 类似，即建模一个固定阶数的条件概率模型，为了能够达到和 LSTM 可比的效果，CNN 语言模型通常阶数会比较高，如 10 阶模型，这种情况下，CNN 相比于 LSTM 最大的优点就是训练快，这是因为 CNN 语言模型不同位置的输出可以并行计算，相比于需要递归计算的 LSTM 训练效率会大大提高。

## 2.5　整句最大熵语言模型

以上介绍的语言模型都属于条件概率模型，即将联合概率表示成条件概率的连乘（即 (2-6)），相对的，整句最大熵（Whole-Sentence Maximum Entropy，WSME）

语言模型[49] 的目标是直接建模序列的联合概率 $p(x_1,\cdots,x_l)$，并且使用一个全局的归一化系数 $Z$ 来对各个长度的序列整体进行归一化，即：

$$p(x_1,\cdots,x_l;\lambda)=\frac{1}{Z}e^{\phi(x_1,\cdots,x_l;\theta)} \tag{2-12}$$

其中 $\phi(x_1,\cdots,x_l;\theta)$ 表示指数模型的势函数，参数为 $\theta$。虽然最大熵模型拥有能够自然的嵌入句子级别特征的潜力，但是目前的整句最大熵模型的效果却不尽如人意[49–51]。失败主要来自于两方面的原因：

- 首先，使用全局归一化系数来对各个长度进行统一归一化的做法存在很大的问题。由于语言模型是一个序列建模问题，随着序列长度的增加，状态空间呈指数增长。鉴于语言模型中的特征往往是长度无关且位置无关的，因此如果仅仅使用一个全局归一化系数对所有长度进行归一化，那么模型的边缘长度的分布很有可能会随长度的增加而成指数上升。假设一种极端的情况，即势函数 $\phi(x_1,\cdots,x_l;\theta)$ 的值恒等于 0，则模型变成是一个均匀分布，这样模型的长度分布为 $p(l)=\sum_{x_1,\cdots,x_l}p(x_1,\cdots,x_l)=V^l/Z$，其中 $V$ 表示词典大小，即语言所包含的词的数目。这显然是不合理，对于序列建模问题，由于长尾效应的存在，当序列长度足够大时，长度的分布应该呈衰减趋势，而不应该呈指数增长的趋势。
- 其次，整句最大熵模型为了能够计算模型分布下的期望，采用了采样的方法来近似模型分布，采样方法包括 Gibbs 采样、独立 Metropolis-Hasting 采样和重要性采样等等，然而，对于语言建模任务，由于其状态空间极大，简单的使用这些采样方法是无法获得有效的样本的，因此整句最大熵模型的参数估计效果很差，这使其很难获得有效的结果

## 2.6　使用句法信息进行自然语言建模

自然语言是基于成分的，这种成分见的依赖约束关系可以通过句法树来体现，因此语言模型的研究一直希望能够将句法树信息应用进来。但在自然语言中使用句法树信息存在很多亟待解决的问题。首先，语言模型是一个产生式模型，为了将句法信息考虑进来，就需要构建一个词序列和句法树的联合概率模型，这意味着训练模型需要足够多有标注的完备数据，而包含句法树标注的数据极其有限；其次，为了能够使用语言模型，我们需要对给定词序列的所有可能的句法树进行积分，从而获得词序列的边缘概率，而对于一个树形结构做积分十分的困难；最后，句法树的获得需要观测到完整的句子，目前主流的语言模型都是条件概率模型，概

率的估计仅仅依赖于历史词序列，在这种情况下是无法获得一个完整的句法树的。

虽然句法树的使用存在很多困难，但是人们依然做了很多相关的研究。Ciprian Chelba 提出了是结构语言模型（Structured Language Model）[14]，其基本思路是对历史词序列进行局部句法分析（partial parse）获得句法信息，然后拟合给定历史词和其局部句法信息的条件下的当前词的概率。局部句法分析的一个最大是好处是可以有效的提取短语的中心词，而这些中心词往往对于预测当前词尤为重要。文中的结构语言模型并没有获得显著的提升，使用局部句法分析来提取中心词的想法为在条件概率的框架下引入句法信息提供以一个思路，随后 Jun Wu 将局部句法特征和 *n*-gram 特征一起嵌入到最大熵模型的框架中[52]，研究了一系列提高算法计算效率的方法[15]，并且在混淆度和语音识别错误率上都超越了传统的 *n*-gram 语言模型。

使用局部句法结构的方法一个关键的问题是需要获得序列的局部句法信息，虽然我们可以选择包含句法标注的数据作为训练，但是在测试时，尤其是在进行语音识别打分的时候，测试的句子往往没有句法结构标注，这种情况下我们需要以来一个额外的句法分析工具来对测试句子做句法分析，这使得模型的性能部分依赖于句法分析的准确率。避免这种情况的办法是对词序列和句法树进行联合产生式建模，因此 Chris Dyer 提出了递归神经网络语法模型（Recurrent neural network grammars，RNNG）[16]。模型的基本思路是首先对句法树基于句法分析操作（action）的顺讯进行序列化，然后将句法分析操作中的读入一个新词替换为产生一个新词，这样模型就从一个鉴别模型转变为一个产生式模型，最后借助递归神经网络的强大的序列建模能力来建模序列化后的句法树。RNNG 依然是一个条件概率模型，但是与传统的条件概率语言模型不同的是，RNNG 建模的不是词词序列，而是操作（action）序列，这个操作序列可以直接对应于一个完整的句法树，由于句法树的叶子节点即为词，因此 RNNG 实际建模了词序列和句法树的联合概率。

RNNG 为使用句法结构提供了另一思路，即将句法树基于一定顺序进行序列化，进而转换为序列建模问题。句法树的序列化表示已经被大量应用于句法分析算法中，具体的序列化顺序也根据具体问题而有所不同，如依赖关树中常用 shift-reduce 顺序[53]，成分树结构中用到的 left-corner 顺序[54]、Top-down 顺序[16] 等，甚至简单的使用树的深度优先遍历[55] 将树序列化。而 RNNG 这种直接建模词序列和句法树的联合概率的模型最大是问题就是推理困难，为了能够计算词序列的边缘概率，需要对所有可能的句法树进行求和，实际中可以使用重要性采样来近似积分，获得一个近似的边缘概率，或者也可以通过计算词序列和最优句法树的联合概率的方式来计算一个得分，这个得分可以直接被用于语音识别等识别任务中。

## 2.7　本章小结

本章首先介绍了语言模型的评估指标，对比了不同指标的优劣，然后对目前已有的语言模型进行了较完整的介绍，最后介绍了使用句法树进行语言建模的相关研究工作。本章对比了每种语言模型优缺点，为下文的研究提供了很好的参考。

# 第 3 章　跨维随机场语言模型

语言建模任务中每个词对应于一个非负整数，因此语言建模可以看做是一个变维向量（变长词序列）的建模的问题，句子的长度对应向量的维度。考虑对变维空间的样本进行建模这一基本问题，本章提出了跨维随机场模型（Trans-dimensional Random Field，TRF），定义了一个包含多个随机场的混合模型，每个随机场对应于一个特定的维度。在随机近似理论的框架下，本章提出了一个有效的训练算法——增强随机近似算法（Augmented Stochastic Approximation，AugSA），可以联合估计模型参数和每个随机场的归一化系数，同时引入了一些优化技术来提高算法的收敛性和计算效率，使得算法能够在较大数据集上训练跨维随机场模型。本章详细介绍了跨维随机场模型的技术细节，讨论了其相比于有向图模型的理论优势，通过多个实验详细评估了模型和训练算法的性能，在词形建模实验中，由于模型的对数似然和梯度都可以被精确求解，训练算法的收敛性可以被准确的验证，并且与已有算法进行了对比。在语言建模实验中，跨维随机场模型展示了它对丰富特征的支持能力，使其可以获得与有向图模型的良好互补性。同时由于避免了局部归一化的计算，其推理效率相比于神经网络模型有显著的提升。

## 3.1　引言

概率图模型为描述概率分布提供了一个统一的框架，并被广泛应用于构建各种人工智能系统。概率图模型分为两类，一类是有向图模型（或称之为贝叶斯模型），将联合概率分解为一系列局部条件概率的连乘。另一类是无向图模型（或称之为马尔科夫随机场），其联合概率被定义为正比于局部势函数的乘积。在一些特定的任务中，我们需要对一个固定维度空间的变量进行建模，如对固定大小的图像进行建模，我们称这种情形为定维情形，而在另一些任务中，待建模的变量拥有不同的维度，如语言模型对句子序列进行建模（维度对应句子的长度），我们称这种情形为跨维情形。本章研究的是跨维情形下的概率图建模问题。

目前，在跨维情形中，尤其是序列建模问题中，有向图模型（如隐马模型、动态贝叶斯网络等）依然是主流方法，而无向图模型却鲜用于序列问题中，这主要是因为训练无向图模型要比训练有向图模型更有挑战[56]. 一般来讲，即使是在定维的情形中，计算无向图模型的对数似然和梯度在理论上都是不可行的，这是由于需要计算模型的归一化系数和模型分布下的期望，而这两项的计算都需要在变量的状态空间内进行积分。虽然在一些简单的情况下，如低阶的随机场模型，或者

模型状态空间较小的情况，归一化系数和模型期望可以通过一些动态规划算法来精确求解或近似估计，目前被广泛使用的条件随机场（Conditional Random Field，CRF）模型就满足上述条件，这是由于 CRF 仅仅对给定观测序列条件下状态序列的分布进行建模，而状态序列空间往往比较小（如词性标签），且模型特征阶数比较低（如 2 阶），因此，目前关于随机场方法的大部分研究都是针对 CRF 模型。然而，CRF 模型是一个鉴别模型，仅仅能被用于鉴别任务，如自然语言的分词、词性标注或图像的分类任务，而不能被用于产生式任务，如语言模型。

本章旨在构建一个产生式的随机场模型来对跨维序列进行建模，模型可以支持更高的阶数以及更大的参数空间，使其能够更好的建模序列中的长跨度约束。这种无向图方法相比于有向图方法主要的优势体现在以下几点：

(1) 在概率图中，无向图使用无向边来连接状态，在一些特定的任务中，使用无向图来描述状态间的约束更加的自然，而有向图使用有向边来描述状态间约束的做法往往与事实不符；

(2) 无向图模型可以避免有向图建模中存在的“标签偏置（Label Bias）”的问题，这是由于无向图模型不进行局部归一化，这使得不同历史条件下的特征权重直接可比[9]；

(3) 无向图模型可以更加灵活的支持复杂特征，相比于有向图模型，建模能力更强；

(4) 无向图模型是全局归一化模型，由于避免了局部归一化，模型推理的效率要比需要频繁归一化的有向图模型（如神经网络模型）要高。

以上需求成为本文随机场方法研究的主要动机。

然而已有的研究证实[56]，简单的构建一个基于模板的固定维度的随机场模型并将其应用于跨维样本的做法是存在问题的。Stephen Della Pietra 等人提出了一个适用于跨维样本的随机场模型[32]，但是文中用于模型训练的改进迭代缩放（Improved Iterative Scaling，IIS）算法和 Gibbs 采样方法无法适用于较大状态空间的问题（如语言模型）中，文中仅仅将随机场模型应用于英文单词仿真任务上，并没有对模型性能进行数值化的衡量，因此文中提出的特征引入策略和 IIS 训练算法被主要用于后续 CRF 的研究中。另一个先验的工作是整句最大熵（Whole-Sentence Maximum Entropy，WSME）语言模型[49]（见第2.5节），这个工作的目的与本章的主要目的相同，即对自然语言序列进行联合产生式建模。尽管 WSME 语言模型拥有能够灵活嵌入各种句子级别特征的潜力，但是目前的整句最大熵模型的效果却不尽如人意[49–51]，这一方面是由于对不同长度序列进行统一归一化的方法并不能很好的描述变长序列，另一方面源自模型使用的估计算法不够有效，使得模型无

法很好的估计参数。

为了能够描述变维样本，本章提出了一个全新的概率模型——跨维随机场模型（Trans-dimensional Random Field，TRF）。TRF 是一个混合模型，包含多个随机场模型，每个随机场模型对应于一个不同的维度（长度），随机场模型之间共享参数。基于这样的模型定义，本章在随机近似理论的框架下提出了一个有效的参数训练算法——增强随机近似算法（Augmented Stochastic Approximation，AugSA），来联合估计模型的参数和每个随机场的归一化系数。AugSA 算法包含两个主要部分，首先通过本章提出的跨维混合采样算法来产生满足模型分布的样本，然后计算样本下的期望来同时更新参数和归一化系数。此外，本章还引入了一些优化策略来提高模型的收敛性，策略涵盖学习速率的配置、迭代收敛的判决、二阶距信息的使用、先验长度分布的取值、过长序列的处理以及如何使用类别信息和多 CPU 来加速训练。这些策略的引入使得 TRF 可以在一个较大规模的数据集上训练语言模型。

本章设计了多个实验对 TRF 模型、训练算法和采样算法进行评估，包括：

(1) 词形建模实验，即将英文单词当做字母序列，对字母序列进行建模。由于任务的状态空间很小（即 26 个英文字母），此实验可以精确的计算似然值和梯度来求解参数。因此这个实验可以准确的验证算法在不同配置下的收敛性，同时和已有的随机场优化算法进行对比。

(2) 第二个实验是语言模型实验，这也是随机场方法研究的初衷，即跨维随机场语言模型。本章在 Peen Treebank（PTB）英文数据集上验证了 TRF 模型和训练算法的有效性，实验结果表明，TRF 模型显著超越传统的 *n*gram 语言模型，相对语音识别错误率下降 10%，并且获得和 LSTM 语言模型相近的结果，但是 TRF 的推理效率是 LSTM 语言模型的 63 倍。此外，TRF 模型展现出与 LSTM 语言模型优良的互补性，两个模型的插值模型可以获得最低的语音识别错误率。

(3) 为了验证 TRF 语言模型随着训练数据增加是否能保持性能的优势，本章在 Google 1-billion 英文语料上进行语言模型的实验，通过依次增加训练集的规模来观察模型性能的变化。

以上实验全面验证了 TRF 模型在语言建模这一序列建模中的能力，以及 AugSA 训练算法和跨维混合采样算法的有效性。这些实验结果首次为随机场方法在语言建模中的应用提供了强有力的经验证明。

本章的组织结构如下。第3.2节给出了 TRF 模型的定义，第3.3节提出了增强随机近似算法和跨维混合采样算法，第3.4节对 TRF 模型进行简单的分析，对比了

TRF 与条件概率模型和整句最大熵模型，第3.5节介绍了一系列的算法优化技术，用以提高算法的效率，第3.6节通过一个仿真实验验证了 TRF 模型的性能和训练算法的有效性，第3.7节将 TRF 应用于语言建模任务中，通过语音识别词错误率对模型进行了评估，最后，在第3.8节中对本章进行总结。本章的主要内容先后发表于 ACL 2015（Annual Meeting of the Association for Computational Linguistics）[23] 和 PAMI 2017（IEEE Transactions on Pattern Analysis and Machine Intelligence）[24]。

## 3.2　模型定义

首先需要明确，在本章中使用的“样本”，对应于语言模型表示的是一句话，或一个词序列，这是因为在实际语言模型中，每个词会被赋予一个非负整数编号，称为词号，因此一句话（一个词序列）可以看作一个非负整数向量，既是下文用到的一个“样本”，而“样本”的“维度”即表示句子的长度。

假设需要对变维空间的样本进行随机场建模，每个样本空间的维度各不相同，例如不同大小的图像，或者不同长度的序列等等。定义 $x^j$ 表示 $j$ 维空间 $\mathcal{X}^j$ 内的一个样本，维度 $j$ 的取值从 1 到 $m$，各维度空间的并集就是这里需要建模的整个样本空间，即 $\mathcal{X} = \bigcup_{j=1}^{m} \mathcal{X}^j$。为了强调样本的维度，本章定义一个数对 $(j, x^j)$ 来表示混合空间 $\mathcal{X}$ 中的一个样本，尽管维度 $j$ 已经暗含在样本在 $x^j$ 中。

对于每一个维度 $j = 1, \cdots, m$，假设样本 $x^j$ 满足如下指数分布：

$$p_j(x^j; \lambda) = \frac{1}{Z_j(\lambda)} e^{\lambda^T f(x^j)}$$

其中 $f(x^j) = (f_1(x^j), f_2(x^j), \ldots, f_d(x^j))^T$ 是一个 $d$ 维特征向量，$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_d)^T$ 是对应的参数向量，$Z_j(\lambda)$ 是模型的归一化系数，即：

$$Z_j(\lambda) = \sum_{x^j} e^{\lambda^T f(x^j)}, \qquad j = 1, \ldots, m$$

为了保证模型的唯一性，本章假设特征 $f_i(x^j), i = 1, \ldots, d$ 的任意线性组合都不会是一个常数，同时假设维度 $j$ 存在一个先验分布 $\pi_j$ 满足 $\sum_{j=1}^{m} \pi_j = 1$，因此维度和样本的联合分布可以表示为：

$$p(j, x^j; \pi, \lambda) = \pi_j \, p_j(x^j; \lambda) = \frac{\pi_j}{Z_j(\lambda)} e^{\lambda^T f(x^j)} \tag{3-1}$$

其中 $\pi = (\pi_1, \ldots, \pi_m)^T$.

特征函数 $f_i(x^j), i = 1,\ldots,d$ 可以定义为任意以 $x^j$ 为输入的函数，在很多任务中，如语言模型，特征函数往往被定义为 $x^j$ 上的局部函数，且常常与位置和维度无关[32]。在本章的实验中，每个特征 $f_i(x^j)$ 被定义为如下形式：

$$f_i(x^j) = \sum_k f_i(x^j, k) \tag{3-2}$$

其中 $f_i(x^j, k)$ 是定义在序列 $x^j$ 的位置 $k$ 处的一个二值函数（返回 0 或者 1）。以 3 元文法为例，如果三个特定的词（如 “a yellow dog”）出现在词序列的第 $k$ 到 $k+2$ 的位置，则 $f_i(x^j, k)$ 返回 1，否则返回 0。不同维度的样本不同位置处的二值特征 $f_i(x^j, k)$ 共享同一个参数 $\lambda_i$，这称之为特征的位置独立性和维度独立性，因此，特征 $f_i(x^j)$ 返回一个非负整数，表示样本 $x^j$ 中某个特定的文法出现的次数。

## 3.3　模型训练

本节在随机近似的理论框架下提出了一个新颖的学习算法来联合估计模型的参数 $\lambda$ 和归一化系数 $Z_1(\lambda),\cdots,Z_m(\lambda)$。这里不直接估计 $Z_j(\lambda)$，而通过估计一个相对的对数比值：

$$\zeta_j^*(\lambda) = \log \frac{Z_j(\lambda)}{Z_1(\lambda)}, \quad j = 1,\ldots,m \tag{3-3}$$

其中 $Z_1(\lambda)$ 被作为一个基准值，这是因为它可以被精确的求解。下文设计了一种名为增强随机近似（AugSA）的算法来估计模型，同时提出了跨维混合采样算法来产生满足分布的样本。

### 3.3.1　最大似然估计

本文使用 $D$ 来表示训练数据集，使用 $D_j, j = 1,\cdots,m$ 表示训练集中维度为 $j$ 的样本组成的子集，满足 $D = \bigcup_{j=1}^m D_j$，使用 $n_j = |D_j|$ 表示集合 $D_j$ 中的样本数目，$n = \sum_{j=1}^m n_j$ 表示训练样本的总数目。下面对模型 (3-1) 的中变量 $(\pi, \lambda)$ 进行最大似然估计。

**推论 1**：　对于模型 (3-1)，变量 $\pi$ 的最大似然估计（MLE）为 $\tilde{\pi} = (\tilde{\pi}_1,\ldots,\tilde{\pi}_m)^T$，其中 $\tilde{\pi}_j = n_j/n$ 表示经验维度分布。$\lambda$ 的最大似然估计由下式确定：

$$\tilde{p}[f] - p_\lambda[f] = 0 \tag{3-4}$$

$\tilde{p}[f]$ 表示特征向量 $f$ 在经验分布下的期望，

$$\tilde{p}[f] = \frac{1}{n}\sum_{j=1}^{m}\sum_{x^j \in D_j} f(x^j) \tag{3-5}$$

$p_\lambda[f]$ 表示特征向量 $f$ 在模型分布 (3-1) 下的期望，

$$p_\lambda[f] = \sum_{j=1}^{m}\frac{n_j}{n}\sum_{x^j \in \mathcal{X}^j} p_j(x^j;\lambda) f(x^j) \tag{3-6}$$

**证明** 对于模型 (3-1)，$\pi_j$ 表示维度 $j$ 的概率，显然，其最大似然估计为训练集中的维度分布，即 $\tilde{\pi} = n_j/n$。固定 $\pi_j$，模型在训练集上的对数似然表示为：

$$L(\lambda) = \frac{1}{n}\sum_{j=1}^{m}\sum_{x^j \in D_j} \log p(j, x^j; \pi, \lambda),$$

计算 $L(\lambda)$ 对参数 $\lambda$ 梯度并使梯度等于 0，得到：

$$\frac{\partial L(\lambda)}{\partial \lambda} = \tilde{p}[f] - p_\lambda[f] = 0.$$

即目标函数 (3-4)。 □

### 3.3.2 归一化系数的估计

公式 (3-4) 给出了求解 $\lambda$ 的目标函数，下面推导求解归一化系数的目标函数。公式 (3-3) 中定义了一个相对的对数归一化常数 $\zeta_j^*(\lambda)$，这里引入自适应混合采样的理论[57] 来估计模型的归一化系数。首先，引入一个新的变量 $\zeta$ 将模型 (3-1) 重定义为如下形式：

$$p(j, x^j; \lambda, \zeta) = \frac{\pi_j}{Z_1(\lambda)} e^{\lambda^T f(x^j) - \zeta_j} \tag{3-7}$$

其中 $\zeta = (\zeta_1, \ldots, \zeta_m)^T$ 且满足 $\zeta_1 = 0$。值得一提的是，公式 (3-3) 中的 $\zeta_j^*(\lambda)$ 是真实的归一化系数，因此与参数 $\lambda$ 的取值有关，而 $\zeta_j$ 则表示 $\zeta_j^*(\lambda)$ 的估计值，是一个待估计变量。然后有如下推论：

**推论 2**：对于模型 (3-7)，给定长度分布 $\pi_j$ 和参数 $\lambda$ 的情况下，下面方程对变量 $\zeta_j$ 的根即为真实的归一化常数 $\zeta_j^*(\lambda)$：

$$\sum_{x^j} p(j, x^j; \lambda, \zeta) = \pi_j, \quad j = 1, \ldots, m \tag{3-8}$$

**证明** 将公式 (3-7) 代入 (3-8)，得到：

$$\begin{aligned}
\pi_j &= \sum_{x^j} p(j, x^j; \lambda, \zeta) \\
&= \sum_{x^j} \frac{\pi_j}{Z_1(\lambda)} e^{\lambda^T f(x^j) - \zeta_j} \\
&= \frac{\pi_j}{Z_1(\lambda)} e^{-\zeta_j} \sum_{x^j} e^{\lambda^T f(x^j)} \\
&= \pi_j \frac{Z_j(\lambda)}{Z_1(\lambda)} e^{-\zeta_j}
\end{aligned}$$

所以

$$\zeta_j = \log \frac{Z_j(\lambda)}{Z_1(\lambda)} = \zeta_j^*(\lambda) \qquad \square$$

### 3.3.3 随机近似理论

公式 (3-4) (3-8) 给出了 TRF 模型参数估计和归一化系数估计的目标函数，但是目标函数的求解都需要计算模型分布下的期望，这在实际建模任务（如语言模型）中是不可行的，因此本节引入随机近似（Stochastic Approximation，SA）的理论来求解。

1951 年，Herbert Robbins 和 Sutton Monro 提出了随机近似（Stochastic Approximation，SA）的方法[58] 用来求解一种特殊形式的方程的根，随后人们对这一方法从理论和应用层面进行了大量的研究[59,60]。SA 方法的优势是可以避免方程中期望项的计算，这恰恰是 TRF 训练面临的问题，因此本文在 SA 理论的框架下提出了一种增强的随机近似（AugSA）方法，来联合求解目标函数 (3-4) 和 (3-8)。

首先，这里简单介绍一下 SA 的基本结论。假设需要求解目标方程 $h(\theta) = 0$ 的根 $\theta^*$，$h(\theta)$ 有如下期望的形式：

$$h(\theta) = E_{Y \sim g(\cdot;\theta)}[H(Y; \theta)] \tag{3-9}$$

其中 $\theta \in R^d$ 是一个维度为 $d$ 参数向量，$Y$ 是带参概率分布 $g(\cdot;\theta)$ 的样本，$H(Y;\theta) \in R^d$ 是 $Y$ 的一个函数。给定初始参数 $\theta^{(0)}$ 和 $Y^{(0)}$，SA 使用如下迭代算法计算方程的根：

**随机近似（SA）算法：**

(1) 产生样本 $Y^{(t)} \sim K(Y^{(t-1)},\cdot)$，其中 $K(\cdot,\cdot)$ 是一个马尔科夫转移概率，保证马氏链的稳态分布为 $g(\cdot;\theta)$；

(2) 计算 $\theta^{(t)} = \theta^{(t-1)} + \gamma_t A H(Y^{(t)};\theta^{(t-1)})$，其中 $\gamma_t$ 表示学习速率，$A$ 是一个可逆的缩放矩阵。

其中 $t = 1,2,\ldots$ 表示迭代次数。

在每次迭代过程中，可以通过多次执行马尔科夫转移来产生多个样本 $Y$，然后通过计算样本上的均值来更新参数，可以得出如下的改进算法，这种改进可以有效的降低训练过程中的振荡，提高训练稳定性。

**基于多次转移的 SA 算法：**

(1) 令 $Y^{(t,0)} = Y^{(t-1,K)}$，对 $k$ 从 1 到 $k$，依次产生样本 $Y^{(t,k)} \sim K(Y^{(t,k-1)},\cdot)$，其中 $K(\cdot,\cdot)$ 是一个马尔科夫转移概率，保证马氏链的稳态分布为 $g(\cdot;\theta)$；

(2) 计算 $\theta^{(t)} = \theta^{(t-1)} + \gamma_t A\{\frac{1}{K}\sum_{Y\in B^{(t)}} H(Y;\theta^{(t-1)})\}$, 其中 $B^{(t)}$ 是样本集合，即 $B^{(t)} = \{Y^{(t,k)}|k = 1,\cdots,K\}$.

对于 SA 算法在不同条件下的收敛性，人们进行了大量的研究[59,60]，结论表明，在满足学习速率 $\gamma_t = \gamma_0/t^\beta$ 且 $\gamma_0 > 0, 0.5 < \beta \leq 1$ 的情况下，$\gamma_t^{-1/2}(\theta^{(t)} - \theta^*)$ 随 $t \to \infty$ 收敛到一个多元正太分布 $N(0,\Sigma)$。此外，在 $\beta = 1$ 的条件下，正太分布的方差 $\Sigma$ 在学习速率 $\gamma_t = 1/t$ 和缩放矩阵 $A = -\{\partial h(\theta^*)/\partial\theta\}^{-1}$ 的时候达到最小值，这意味着此条件下迭代算法的收敛最快。基于此结论，可以获得如下的最优 SA 迭代算法：

$$\theta^{(t)} = \theta^{(t-1)} - \frac{1}{t}\left\{\frac{\partial h}{\partial\theta}(\theta^*)\right\}^{-1} H(Y^{(t)};\theta^{(t-1)}), \tag{3-10}$$

然而，这种最优的更新公式 (3-10) 实际中往往不可求解，因为偏导 $\partial h(\theta^*)/\partial\theta$ 往往不可计算，因此实际中一般有两种方法来进行近似。一种方法是通过 SA 同时估计 $\theta^*$ 和 $\partial h(\theta^*)/\partial\theta$[61]；另一种方法是使用轨迹平均 $\bar{\theta}^{(t)} = t^{-1}\sum_{i=1}^{t}\theta^{(i)}$，同时减慢学习速率的收敛，如将学习速率设为 $\gamma_t = \gamma_0/t^\beta$ 且 $1/2 < \beta < 1$[62]。本文的实验使用第一种方法来加速 SA 算法的收敛性，第二种方法在本文实验中没有收到效果。

为了使用 SA 来训练 TRF 模型，首先将目标函数公式 (3-4)(3-8) 套入到 SA 的

框架 $h(\theta)=0$ 中表示成联立方程的形式，有：

$$\begin{cases} \theta = (\lambda, \zeta_{(1)}) \\ Y = (j, x^j) \\ g(\cdot;\theta) = p(j, x^j; \lambda, \zeta) \\ H(Y;\theta) = \begin{pmatrix} \tilde{p}[f] - \frac{n_j/n}{\pi_j} f(x^j) \\ \delta_{(1)}(j) - \pi_{(1)} \end{pmatrix} \end{cases} \tag{3-11}$$

其中 $\zeta_{(1)} = (\zeta_2, \ldots, \zeta_m)^T$，$\pi_{(1)} = (\pi_2, \ldots, \pi_m)^T$，$\delta_{(1)}(j) = (\delta_2(j), \ldots, \delta_m(j))^T$，$\delta_l(j) = 1(j=l)$ 为二值函数，在 $j=l$ 时取 1，其他情况取 0。从而有如下推论：

**推论** 3：　在公式 (3-11) 的定义下，$h(\theta)=0$ 的根 $\theta^* = (\lambda^*, \zeta^*_{(1)})$ 即为 TRF 模型 (3-7) 在最大似然准则下的最优参数和目标归一化系数。

**证明** 将 (3-11) 代入公式 (3-9) 中有：

$$\tilde{p}[f] - \sum_{j=1}^{m} \frac{n_j}{n} \sum_{x^j} p_j(x^j; \lambda, \zeta) f(x^j) = 0 \tag{3-12}$$

$$\sum_{x^j} p(j, x^j; \lambda, \zeta) = \pi_j, \quad j = 2, \ldots, m \tag{3-13}$$

其中

$$p_j(x^j; \lambda, \zeta) = \frac{1}{Z_1(\lambda)} e^{\lambda^T f(x^j) - \zeta_j}$$

于是有如下结论：

(1) 首先，公式 (3-13) 既是目标方程 (3-8) 在 $j=2,\ldots,m$ 时的形式，而对于 $j=1$，由于 $\zeta_1 = 0$，方程 (3-8) 显然成立。因此公式 (3-13) 的解 $\zeta^*_{(1)}$ 是准确的归一化常数；

(2) 然后，在上述条件下，公式 (3-12) 就等价于目标方程 (3-4)，从而保证 $\lambda^*$ 是最大似然估计。 □

下面考虑如何估计可逆缩放矩阵 $A$ 来提高算法的收敛效率。为了简单起见，本文假设 $A$ 是一个分块对角阵 $(A_\lambda, A_\zeta)$，分别对应参数 $\lambda$ 和归一化系数 $\zeta$，更进一步，假设 $A_\lambda^{-1}$ 是一个对角阵，并且使用经验方差来近似最优的缩放矩阵，即

$\sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$，其中 $\sigma_i, i = 1, \ldots, d$ 是特征 $f_i$ 的经验方差，

$$\sigma_i = \sum_{j=1}^{m} \frac{n_j}{n} \tilde{p}_j \left[ (f_i - \tilde{p}_j[f_i])^2 \right]$$

其中 $\tilde{p}_j[f_i] = \frac{1}{n_j} \sum_{x^j \in D_j} f_i(x^j)$。最终参数 $\lambda$ 在第 $t$ 次迭代的更新公式为：

$$\lambda^{(t)} = \lambda^{(t-1)} + \gamma_{\lambda,t} \sigma^{-1} \left\{ \tilde{p}[f] - \frac{1}{K} \sum_{(j,x^j) \in B^{(t)}} \frac{n_j/n}{\pi_j} f(x^j) \right\} \tag{3-14}$$

其中 $\gamma_{\lambda,t}$ 表示 $\lambda$ 的学习速率，第3.5.2节会对种做法的正确性进行更详细的分析。对于归一化系数 $\zeta$ 的缩放矩阵 $A_\zeta$，本文引入自适应混合采样[57] 方法，可以精确的计算最优的缩放矩阵来更新 $\zeta$。从而有如下推论。

**推论** 4：　固定 $\lambda$，将 (3-11) 重写为 $\zeta_{(1)}$ 的函数：

$$\begin{cases} \theta = \zeta_{(1)} \\ Y = j \\ g(\cdot; \zeta_{(1)}) = p(j, x^j; \lambda, \zeta) \\ H(Y; \zeta_{(1)}) = \delta_{(1)}(j) - \pi_{(1)} \end{cases}$$

并且 $h(\zeta_{(1)})_j = \sum_{x^j} p(j, x^j; \lambda, \zeta) - \pi_j, \quad j = 2, \ldots, m$，$\zeta^*_{(1)}$ 表示给定 $\lambda$ 的准确归一化系数，则最优的缩放矩阵 $A_\zeta = \{-\partial h(\zeta^*_{(1)}) / \partial \zeta_{(1)}\}^{-1}$ 满足

$$A_\zeta \, H(Y; \zeta_{(1)}) = \left( \frac{\delta_2(j)}{\pi_2}, \ldots, \frac{\delta_m(j)}{\pi_m} \right)^T - \frac{\delta_1(j)}{\pi_1} \tag{3-15}$$

**证明** 对于任意的 $j, k = 2, \ldots, m$，有

$$\frac{\partial}{\partial \zeta_k} \left\{ \sum_{x^j} p(j, x^j; \lambda, \zeta) - \pi_j \right\} = \begin{cases} p(j;\zeta) p(k;\zeta) & \text{if } k \neq j, \\ -p(j;\zeta) + p^2(j;\zeta) & \text{if } k = j. \end{cases}$$

其中 $p(j;\zeta) = \sum_{x^j} p(j, x^j; \lambda, \zeta)$。因此，$A_\zeta^{-1} = -\partial h(\zeta^*_{(1)}) / \partial \zeta_{(1)} = \Pi_{(1)} - \pi_{(1)} \pi_{(1)}^T$，且 $A_\zeta = \Pi_{(1)}^{-1} + \pi_1^{-1} 1 1^T$，其中 $\Pi_{(1)} = \mathrm{diag}(\pi_{(1)})$，$\pi_{(1)} = (\pi_2, \ldots, \pi_m)^T$，$1 = (1, \ldots, 1)^T$ 是 $m-1$ 维向量。直接计算可得 (3-15)。 □

根据推论4，在第 $t$ 次迭代时 $\zeta$ 的更新公式如下：

$$\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + \gamma_{\zeta,t} \left\{ \frac{\delta_1(B^{(t)})}{\pi_1}, \cdots, \frac{\delta_m(B^{(t)})}{\pi_m} \right\}^T, \tag{3-16}$$

$$\zeta^{(t)} = \zeta^{(t-\frac{1}{2})} - \zeta_1^{(t-\frac{1}{2})}, \tag{3-17}$$

其中 $\zeta_1^{(t)}$ 表示 $\zeta^{(t)}$ 的第一个元素，在公式 (3-17) 中被设为 0，$\gamma_{\zeta,t}$ 是 $\zeta$ 的学习速率，$\delta_l(B^{(t)})$ 表示维度 $l$ 在样本集 $B^{(t)}$ 中出现的频率：

$$\delta_l(B^{(t)}) = \frac{1}{K} \sum_{(j,x^j)\in B^{(t)}} 1(j = l), \quad l = 1, \ldots, m$$

完整的 SA 的算法整理在表3.1中。首先是初始化（Step 1），算法将参数 $\lambda$ 全部初始化为 0，由于此时的模型其实是均匀分布，因此可以准确的计算出归一化系数并用来初始化 $\zeta$；随后，在每次迭代过程中，算法先使用跨维混合采样（见第3.3.4节）来产生 $K$ 个样本（Step 3-8），进而计算样本上的期望来更新参数 $\lambda$ 和归一化常数 $\zeta$（Step 9-10）。

### 3.3.4　跨维混合采样

本节提出了跨维混合采样（Trans-dimensional mixture sampling）算法来生成满足分布 $p(j, x^j; \lambda, \zeta)$（公式 (3-7)）的样本，所谓的“跨维”要求产生的样本能在不同维度之间游走，“混合”表示分布 $p(j, x^j; \lambda, \zeta)$ 是一个混合分布，每个子分布定义在不同的子空间中，通过维度 $j$ 来标定。跨维混合采样是一种马氏链蒙特卡洛（Markov Chain Monte Carlo，MCMC）算法，算法通过构建一个转移策略（转移矩阵）来保证马氏链的稳态分布即为当前参数下的模型分布。如果固定 $(\lambda, \zeta)$，跨维混合采样算法本质上是 RJ-MCMC 采样算法[63] 在 TRF 模型上的一种特定的应用。对应于语言模型的具体任务，跨维混合采样可以用来产生不同长度的句子。

跨维混合采样算法包含两步——“局部跳转（local jump）”和“马氏转移（markov move）”，“局部跳转”使样本在不同维度之间跳转，而“马氏转移”使样本固定维度在子空间内部进行转移。首先，这里定义 $j^{(t-1)}$ 和 $x^{(t-1)}$ 表示 $t-1$ 时刻的维度和样本，为方便起见，使用 $(\lambda, \zeta)$ 表示 $t-1$ 时刻的参数和归一化系数，即 $(\lambda^{(t-1)}, \zeta^{(t-1)})$，采样算法描述如下。

**第一步：局部跳转**。此阶段使用 Metropolis-Hastings（MH）算法来产生新的维度。假设 $j^{(t-1)} = k$，首先产生一个新的维度 $j \sim \Gamma(k, \cdot)$，跳转概率 $\Gamma(k, j)$ 定义为 $k$

表 3.1　增强随机近似 (AugSA) 算法

初始化 $\lambda^{(0)} = (0, \ldots, 0)^T$，$\zeta^{(0)} = \zeta^*(\lambda^{(0)}) - \zeta_1^*(\lambda^{(0)})$
**for** $t = 1, 2, \ldots, t_{max}$ **do**
　令 $B^{(t)} = \emptyset$
　令 $(j^{(t,0)}, x^{(t,0)}) = (j^{(t-1,K)}, x^{(t-1,K)})$
　**for** $k = 1 \rightarrow K$ **do**
　　令 $(j^{(t,k)}, x^{(t,k)}) = \text{sample}(j^{(t,k-1)}, x^{(t,k-1)})$　▷ 详见表3.2中的跨维混合采样算法
　　令 $B^{(t)} = B^{(t)} \cup \{(j^{(t,k)}, x^{(t,k)})\}$
　**end for**
　使用公式 (3-14) 跟新参数 $\lambda$。

$$\lambda^{(t)} = \lambda^{(t-1)} + \gamma_{\lambda,t}\sigma^{-1}\left\{\tilde{p}[f] - \frac{1}{K}\sum_{(j,x^j)\in B^{(t)}}\frac{n_j/n}{\pi_j}f(x^j)\right\}$$

　计算 $\delta_l(B^{(t)}) = \frac{1}{K}\sum_{(j,x^j)\in B^{(t)}} 1(j = l)$，使用公式 (3-16) 和 (3-17) 更新归一化系数 $\zeta$。

$$\begin{cases} \zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + \gamma_{\zeta,t}\left\{\dfrac{\delta_1(B^{(t)})}{\pi_1}, \cdots, \dfrac{\delta_m(B^{(t)})}{\pi_m}\right\}^T \\ \zeta^{(t)} = \zeta^{(t-\frac{1}{2})} - \zeta_1^{(t-\frac{1}{2})} \end{cases}$$

**end for**

的一个邻域内的均匀分布:

$$\Gamma(k,j) = \begin{cases} \dfrac{1}{3}, \text{ if } k \in [2, m-1], j \in [k-1, k+1] \\ \dfrac{1}{2}, \text{ if } k = 1, j \in [1,2] \text{ or } k = m, j \in [m-1, m] \\ 0, \text{ otherwise} \end{cases} \tag{3-18}$$

其中 $m$ 表示最大维度。公式 (3-18) 限制了维度 $j$ 和 $k$ 之间的差距不超过 1，如果 $j = k$，算法保留样本 $x^{(t-1)}$ 直接进入下一阶段，即 $j^{(t)} = k$；如果 $j = k+1$ 或者 $j = k-1$，两种情况要分开处理。

如果 $j = k+1$，首先基于一个提议分布产生一个新的元素 $u \sim g_{k+1}(\cdot|x^{(t-1)})$，然

表 3.2　跨维混合采样。算法 1 是普通版本（第3.3.4节），算法 2 是引入类别信息后的加速版本（第3.5.4节）。Unifrom[0, 1] 是一个函数随机返回一个满足 [0,1] 均匀分布的实数。

| 算法 1 | 算法 2 |
|---|---|
| **function** SAMPLE $(j^{(t-1)}, x^{(t-1)})$ | **function** FASTSAMPLE $(j^{(t-1)}, x^{(t-1)})$ |
| $k = j^{(t-1)}$ | $k = j^{(t-1)}$ |
| $(j^{(t)}, x^{(t)}) = (j^{(t-1)}, x^{(t-1)})$ | $(j^{(t)}, x^{(t)}) = (j^{(t-1)}, x^{(t-1)})$ |
| ***Step I:* 局部跳转** | ***Step I:* 局部跳转** |
| $j^{(t)} = k$ | $j^{(t)} = k$ |
| $x^{(t)} = x^{(t-1)}$ | $x^{(t)} = x^{(t-1)}$ |
| $j \sim \Gamma(k, \cdot)$ ▷ 公式 (3-18)) | $j \sim \Gamma(k, \cdot)$ ▷ 公式 (3-18)) |
| **if** $j = k + 1$ **then** | **if** $j = k + 1$ **then** |
| | $c_0 \sim Q_{k+1}(c)$ |
| $u \sim g_{k+1}(\cdot \mid x^{(t-1)})$ ▷ 见公式 (3-21) | $u \sim \breve{g}_{k+1}(\cdot \mid x^{(t-1)}, c_0)$ ▷ 见公式 (3-37) |
| $p =$ 公式 (3-19) | $p =$ 公式 (3-38) |
| **if** Unifrom$[0, 1]$ <= $p$ **then** | **if** Unifrom$[0, 1]$ <= $p$ **then** |
| $j^{(t)} = j$, $x^{(t)} = \{x^{(t-1)}, u\}$ | $j^{(t)} = j$, $x^{(t)} = \{x^{(t-1)}, u\}$ |
| **end if** | **end if** |
| **end if** | **end if** |
| **if** $j = k - 1$ **then** | **if** $j = k - 1$ **then** |
| $p =$ 公式 (3-20) | $p =$ 公式 (3-39) |
| **if** Unifrom$[0, 1]$ <= $p$ **then** | **if** Unifrom$[0, 1]$ <= $p$ **then** |
| $j^{(t)} = j$, $x^{(t)} = x_{1:j}^{(t-1)}$ | $j^{(t)} = j$, $x^{(t)} = x_{1:j}^{(t-1)}$ |
| **end if** | **end if** |
| **end if** | **end if** |
| ***Step II:* 马氏转移** | ***Step II:* 马氏转移** |
| **for** $i = 1 \rightarrow j^{(t)}$ **do** | **for** $i = 1 \rightarrow j^{(t)}$ **do** |
| | $c_0 \sim Q_i(c)$ |
| | **if** Unifrom$[0, 1]$ <= 公式 (3-36) **then** |
| | $c_i^{(t)} = c_0$ |
| | **end if** |
| $u \sim p(j^{(t)}, \{x_{1:i-1}^{(t)}, \bullet, x_{i+1:j^{(t)}}^{(t)}\}; \lambda, \zeta)$ ▷ 见公式 (3-7)) | $u \sim p(j^{(t)}, \{x_{1:i-1}^{(t)}, \bullet, x_{i+1:j^{(t)}}^{(t)}\}; \lambda, \zeta)$ 且 $u \in \mathcal{A}_{c_i^{(t)}}$ ▷ 见公式 (3-7) |
| $x_i^{(t)} = u$ | $x_i^{(t)} = u$ |
| **end for** | **end for** |
| **return** $(j^{(t)}, x^{(t)})$ | **return** $(j^{(t)}, x^{(t)})$ |
| **end function** | **end function** |

后以如下概率来接受 $j^{(t)} = j\,(= k+1)$ 和 $x^{(t)} = \{x^{(t-1)}, u\}$：

$$\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, \{x^{(t-1)}, u\}; \lambda, \zeta)}{p(k, x^{(t-1)}; \lambda, \zeta) g_{k+1}(u|x^{(t-1)})}\right\}, \tag{3-19}$$

其中 $\{x^{(t-1)}, u\}$ 表示一个 $k+1$ 维的向量，其前个 $k$ 元素是 $x^{(t-1)}$ 最后一个元素是 $u$。

如果 $j = k-1$，则以如下概率接受 $j^{(t)} = j\,(= k-1)$ 和 $x^{(t)} = x_{1:k-1}^{(t-1)}$：

$$\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, x_{1:j}^{(t-1)}; \lambda, \zeta) g_k(x_k^{(t-1)}|x_{1:j}^{(t-1)})}{p(k, x^{(t-1)}; \lambda, \zeta)}\right\}, \tag{3-20}$$

其中 $x_{1:j}^{(t-1)}$ 表示向量 $x^{(t-1)}$ 的前 $j$ 个元素组成的子向量，$x_k^{(t-1)}$ 表示向量 $x^{(t-1)}$ 的第 $k$ 个元素。

在公式 (3-19) 和公式 (3-20) 中，分布 $g_{k+1}(u|x^k)$ 可以被指定为任意定义在 $u$ 上的合法的概率密度函数，在本文的实验中使用如下定义：

$$g_{k+1}(u|x^k) = \frac{p(k+1, \{x^k, u\}; \lambda, \zeta)}{\sum_w p(k+1, \{x^k, w\}; \lambda, \zeta)}. \tag{3-21}$$

其中，$x^k$ 是一个维度为 $k$ 的向量，即在公式 (3-19) 中表示为 $x^k = x^{(t-1)}$，在公式 (3-20) 表示为 $x^{k-1} = x_{1:j}^{(t-1)}$。在本文的实验中，“局部跳转”的接受率大概在 30% 左右，从而证实了算法的有效性。

**第二步：马氏转移**。在“局部跳转”之后，目前的序列为：

$$x^{(t)} = \begin{cases} x^{(t-1)} & \text{if } j^{(t)} = k \\ \{x^{(t-1)}, u\} & \text{if } j^{(t)} = k+1 \\ x_{1:k-1}^{(t-1)} & \text{if } j^{(t)} = k-1 \end{cases} \tag{3-22}$$

然后算法使用 Gibbs 采样来对向量 $x^{(t)}$ 从第一个元素到最后一个元素依次进行采样。表3.2中的算法 1 列出了整个采样算法的伪代码。

## 3.4　模型分析

### 3.4.1　TRF 与条件概率模型

本节通过一个简单的例子来解释标签偏置问题在序列建模中是如何产生的，而 TRF 模型又是如何避免这一问题的。本节定义了下面这个简单的语言模型的任务。

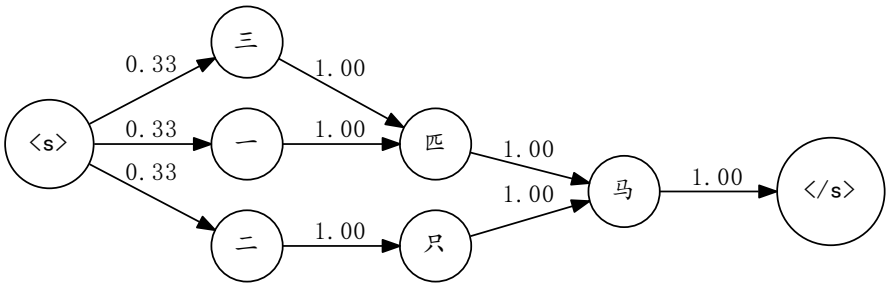



图 3.1　二阶条件概率模型的状态转移图

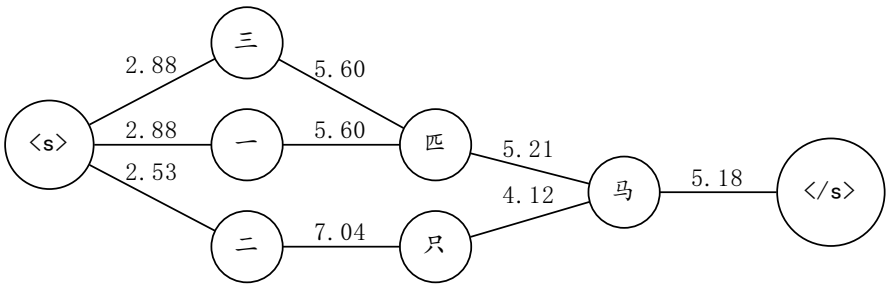



图 3.2　使用二阶特征的随机场模型的状态图

考虑一个仅仅包含 3 个汉字的句子建模问题，训练集仅仅包含如下三句话：

一 匹 马
二 只 马
三 匹 马

这个虚假的训练集模拟了这种情况，由于“匹”和“马”的搭配符合汉语语法规律，因此会大量的出现在训练集中（这里出现两次），而“只”和“马”的这种错误的搭配可能由于笔误或者数据噪声的原因，仅仅出现了一次。在这种情况下，训练集训练的语言模型用来衡量如下两句测试句子的概率：

二 匹 马
一 只 马

由于这两句测试句子并没有出现在训练集中，因此这个例子旨在验证不同模型对于训练集之外的句子的评估能力，即模型的平滑能力，这种能力在实际的语言建模任务中至关重要。下面分别对条件概率模型和随机场模型进行具体分析。

**条件概率模型**。这里使用一个二阶条件概率模型，基于最大似然估计，条件概率模型的状态转移图见图3.1，其中 $\langle s\rangle$ 表示句子的开始，$\langle /s\rangle$ 表示句子的结束，弧上的数值表示转移概率。

为了计算测试集中句子的概率，将联合概率表示成条件概率的连乘：

$$p(二\ 匹\ 马) = p(二|\langle s\rangle)p(匹|二)p(马|匹)p(\langle /s\rangle|马)$$
$$p(一\ 只\ 马) = p(一|\langle s\rangle)p(只|一)p(马|只)p(\langle /s\rangle|马)$$

将图3.1中存在的概率（即训练集中出现的搭配）代入到上式中，有：

$$p(二\ 匹\ 马) = 0.33p(匹|二)$$
$$p(一\ 只\ 马) = 0.33p(只|一)$$

而对于条件概率 $p(匹|二)$ 和 $p(只|一)$, 由于测试集中的搭配“二 匹”和“一 只”没有出现在训练集中，因此条件概率均为 0，从而导致测试集中两句话的概率均为 0。实际中为了避免这种情况，一般需要对条件概率进行平滑，因此，对于条件概率模型，测试集中两句话概率大小完全由平滑算法所决定。

事实上对于测试集上的两句话而言，我们有理由相信“二 匹 马”的概率大于“一 只 马”的概率，因为“匹 马”的搭配更符合语法，因此在训练集中出现的次数多于“只 马”。然而，条件概率模型存在“标签偏置”的问题，这使得不同历史条件下的概率无法进行比较，从而使得条件概率 $p(马|匹) = p(马|只) = 1$，这就无法建模搭配“匹 马”和“只 马”的正确性，也就无法对测试集中的两句话进行合理的评估。

**随机场模型**。这里定义一个简单的随机场模型来对长度为 3 的句子进行建模：

$$p(x_1, x_2, x_3) = \frac{1}{Z} e^{\lambda^T f(x_1, x_2, x_3)}$$

为了对比的公正，随机场模型仅包含与条件概率模型相同的特征，即训练集中出现的 bigram 特征，为了便于对比，这里将随机场表示成无向图的形式展示在图3.2中，其中每个无向边表示对应的 bigram 特征，边上的数值表示特征对应的参数在最大似然准则下的最优解。

对于随机场模型，最大似然估计要求特征的经验期望等于模型分布下的期望，由于训练集中“匹 马”出现的次数大于“只 马”，因此模型中特征“匹 马”的参数应该大于特征“只 马”的参数，图3.2中的结果也验证了这一点，“匹 马”对应

的参数为 5.21，大于“只 马”对应的参数 4.12。因此，随机场模型在对测试集的两句话进行评估时，可以通过后两个字的搭配很好的区分两句话的概率。此模型下测试集两句话的对数概率分别为：

$$\log p(\text{二 匹 马}) = 12.92 - \log Z = -7.06$$
$$\log p(\text{一 只 马}) = 12.19 - \log Z = -7.79$$

其中 $\log Z = 19.98$。这个简单的例子展示了条件概率模型的标签偏置问题在语言建模任务中的表现，展示了随机场模型这种无向图模型在处理标签偏置问题上的优越性，这为本文中跨维随机场模型的研究提供了佐证。

### 3.4.2　TRF 与 WSME

本节就整句最大熵（WSME）模型和跨维随机场（TRF）模型的关系和区别进行说明，WSME 模型的介绍见第2.5节。如果在 WSME 模型公式 (2-12) 的定义中引入维度特征，可以将 WSME 模型表示成如下形式：

$$p(j, x^j; \lambda, \nu) = \frac{1}{Z(\lambda, \nu)} e^{\nu^T \delta(j) + \lambda^T f(x^j)} \tag{3-23}$$

这里使用 TRF 模型的定义方式，将维度 $j$ 单独提取出来，将分布表示成 $j$ 和 $x^j$ 的联合概率。其中 $\delta(j) = (\delta_1(j), \cdots, \delta_m(j))^T$ 表示维度特征，$\delta_l(j) = 1(j = l)$，$f(x^j)$ 仅仅表示维度无关特征，如 ngram 特征，$\nu = (\nu_1, \ldots, \nu_m)^T$ 和 $\lambda$ 表示对应的参数向量，$Z(\lambda, \nu)$ 表示全局的归一化系数：

$$Z(\lambda, \nu) = \sum_{j=1}^{m} \sum_{x^j \in \mathcal{X}^j} e^{\nu^T \delta(j) + \lambda^T f(x^j)} = \sum_{j=1}^{m} e^{\nu_j} Z_j(\lambda) \tag{3-24}$$

然后可以证明，TRF 模型 (3-1) 和 WSME 模型 (3-23) 的最大似然估计是等价的，即有如下推论：

**推论 5**：　对于 TRF 模型 (3-1) 和 WSME 模型 (3-23)：

(i) 如果 $\hat{\lambda}$ 是参数 $\lambda$ 在模型 (3-1) 下的最大似然估计，那么 $(\check{\lambda}, \check{\nu})$ 是参数 $(\lambda, \nu)$ 在模型 (3-23) 下的最大似然估计，并且满足如下对应关系：

$$\check{\lambda} = \hat{\lambda} \tag{3-25}$$
$$e^{\check{\nu}_j} = c\tilde{\pi}_j e^{-\zeta_j^*(\check{\lambda})}, \quad j = 1, \ldots, m \tag{3-26}$$

其中 $c$ 是一个常数，$\tilde{\pi}_j$ 是经验维度分布。

(ii) 如果 $(\check{\lambda}, \check{\nu})$ 是参数 $(\lambda, \nu)$ 在模型 (3-23) 下的最大似然估计，那么公式 (3-26) 成立且 $\hat{\lambda} = \check{\lambda}$ 是参数 $\lambda$ 在模型 (3-1) 下的最大似然估计。

(iii) 在 (i) 或 (ii) 的条件下，模型 (3-23) 模型 (3-1) 等价，即 $p(j, x^j; \check{\lambda}, \check{\nu}) = p(j, x^j; \tilde{\pi}, \hat{\lambda})$。

**证明** 模型 (3-23) 在参数 $(\lambda, \nu)$ 下的似然函数为：

$$L(\lambda, \nu) = \frac{1}{n} \sum_{j=1}^{m} \sum_{x^j \in D_j} \log p(j, x^j; \lambda, \nu)$$

对 $(\lambda, \nu)$ 求导并另导数等于 0，有：

$$\frac{\partial L(\lambda, \nu)}{\partial \lambda} = \tilde{p}[f] - p_{\lambda,\nu}[f] = 0, \tag{3-27}$$

$$\frac{\partial L(\lambda, \nu)}{\partial \nu} = \tilde{p}[\delta] - p_{\lambda,\nu}[\delta] = 0, \tag{3-28}$$

其中 $\tilde{p}[\delta]$ 和 $\tilde{p}[f]$ 表示 $\delta$ 和 $f$ 在经验分布下的期望，而 $p_{\lambda,\nu}[\delta]$ 和 $p_{\lambda,\nu}[f]$ 表示其在模型分布 (3-23) 下的期望，定义类似于公式 (3-5) 和公式 (3-6)。直接计算可得

$$\begin{aligned}
\tilde{p}[\delta_j] &= \tilde{\pi}_j \\
p_{\lambda,\nu}[\delta_j] &= \sum_{x^j \in \mathcal{X}^j} p(j, x^j; \lambda, \nu) \\
&= \frac{e^{\nu_j}}{Z(\lambda, \nu)} \sum_{x^j \in \mathcal{X}^j} e^{\lambda^T f(x^j)} \\
&= \frac{e^{\nu_j}}{Z(\lambda, \nu)} Z_j(\lambda).
\end{aligned}$$

代入公式 (3-28) 得：

$$\tilde{\pi}_j = \frac{e^{\nu_j} Z_j(\lambda)}{Z(\lambda, \nu)}, \quad j = 1, \ldots, m, \tag{3-29}$$

在公式 (3-29) 的约束下，模型 (3-23) 的最大似然估计模型有如下形式：

$$p(j, x^j; \lambda, \nu) = \frac{\tilde{\pi}_j}{Z_j(\lambda)} e^{\lambda^T f(x^j)}. \tag{3-30}$$

将公式 (3-30) 代入直接计算有：

$$\begin{aligned} p_{\lambda,\nu}[f] &= \sum_{j=1}^{m} \sum_{x^j \in \mathcal{X}^j} p(j, x^j; \lambda, \nu) f(x^j) \\ &= \sum_{j=1}^{m} \tilde{\pi}_j p_{\lambda,j}[f] = p_\lambda[f]. \end{aligned}$$

因此公式 (3-27) 等价于 $\tilde{p}[f] - p_\lambda[f] = 0$，即公式 (3-4)。于是有：

(i) 如果 $\hat{\lambda}$ 是参数 $\lambda$ 在模型 (3-1) 下的最大似然估计，则 $\hat{\lambda}$ 满足公式 (3-4)，等价于公式 (3-27)，因此 $\hat{\lambda}$ 是模型 (3-23) 的最大似然估计；又因为 $\nu$ 的最大似然估计满足公式 (3-29) 形式，得证。

(ii) 如果 $(\check{\lambda}, \check{\nu})$ 是参数 $(\lambda, \nu)$ 在模型 (3-23) 下的最大似然估计，则 $\check{\lambda}$ 满足 (3-27)，等价于公式 (3-4)，因此 $\hat{\lambda} = \check{\lambda}$ 是 TRF 模型 (3-1) 的最大似然估计

(iii) 根据公式 (3-30)，可知。 □

综上所述 TRF 模型相比于 WSME 模型的一个重要的特点就是显示引入了维度的约束，虽然 WSME 的工作中[64] 也会引入序列长度作为特征，但是 TRF 这种以局部归一化的方式引入长度约束可以使模型更容易进行采样和训练，主要有一下几点原因：

- 首先可以看到，模型 (3-1) 是一个随机场混合模型，每个随机场定义在一个固定维度的子空间上，因此称之为跨维随机场（Trans-dimensional Random Field）。TRF 引入一个自由变量 $\pi_j$ 作为混合权重将不同维度空间上的随机场模型进行加权，使得模型的维度分布手动可控，从而保证了维度分布的合理性，基于最大似然估计，$\pi_j$ 可以设为或近似为经验的维度分布。
- 事实上，WSME（公式 (2-12)）也可以被看做是一个随机场混合模型，

$$p(j, x^j; \lambda) = \frac{Z_j(\lambda)}{Z(\lambda)} \cdot \frac{1}{Z_j(\lambda)} e^{\lambda^T f(x^j)}, \tag{3-31}$$

其中 $Z(\lambda) = \sum_x e^{\lambda^T f(x)} = \sum_{j=1}^{m} Z_j(\lambda)$。混合权重为 $Z_j(\lambda)/Z(\lambda)$，即与每个维度的归一化系数 $Z_j(\lambda)$ 成正比，但是在实际任务中，$Z_j(\lambda)$ 往往随维度 $j$ 呈指数增加，因此混合权往往成指数递增。由于混合权重本质上为模型的边缘维度概率（在语言模型中是即模型的长度概率），这使得模型在最大长度处的概率接近于 1，这显然是不合理。这种不合理性进而导致 WSME 模型的采样和参数训练都性能很差，因此现有的 WSME 的结果都并不理想，往往需要与传统模型如 *n*gram 语言模型做组合才能收到微弱的效果。

## 3.5　算法优化技术

增强随机近似（AugSA）算法依然存在着计算昂贵、收敛缓慢的问题，尤其是在参数 $\lambda$ 的维度很高的时候。本章提出了一些优化技术，从理论和应用两方面探究如何提高算法的收敛性以及降低计算代价。

### 3.5.1　学习速率

在第3.3.3节介绍 SA 收敛性时提到，当 $\gamma_t = O(t^{-1})$ 时可以保证 SA 算法的渐进收敛性，在实际应用中，为了能够增加算法初期的收敛速度，同时保证算法的收敛性能，本章中采取下如下这种两阶段法来设置学习速率，类似的方法也出现在前人的研究中[57,61]，即：

$$\gamma_{\lambda,t} = \begin{cases} (t_c + t^{\beta_\lambda})^{-1} & \text{if } t \le t_0 \\ (t_c + t - t_0 + t_0^{\beta_\lambda})^{-1} & \text{if } t > t_0, \end{cases} \tag{3-32}$$

$$\gamma_{\zeta,t} = \begin{cases} t^{-\beta_\zeta} & \text{if } t \le t_0 \\ (t - t_0 + t_0^{\beta_\zeta})^{-1} & \text{if } t > t_0, \end{cases} \tag{3-33}$$

其中 $0.5 < \beta_\lambda, \beta_\zeta < 1$，$t_0$ 表示预热次数，$t_c$ 是一个偏置常数。在第一阶段，即 $t \le t_0$，由于 $0.5 < \beta < 1$，这使得学习速率的下降速度比 $t^{-1}$ 慢，从而保证了学习速率有一个较大的取值，这迫使待估计的模型参数 $\lambda^{(t)}$ 和归一化常数 $\zeta^{(t)}$ 可以快速收敛到最优值附近；在第二阶段，即 $t > t_0$，学习速率以 $t^{-1}$ 衰减，保证了 SA 算法的渐进收敛性。偏置常数 $t_c$ 用来控制 $\lambda$ 的初始学习速率不至于过大。在本章的实验中，$t_0$ 的选择需要保证第一阶段模型在开发集上的似然值已趋于收敛。

### 3.5.2　缩放矩阵的选择

在公式 (3-14) 中引入了一个对角矩阵 $\sigma$ 来缩放 $\lambda$ 的更新方向，本节对这种做法的正确性进行讨论。

为了简单起见，这里仅仅认为公式 (3-11) 的 $\lambda$ 为未知变量，而 $\zeta = \zeta^*(\lambda)$ 为真实的归一化常数，于是有：

$$\begin{cases} \theta = \lambda \\ Y = (j, x^j) \\ g(\cdot;\theta) = p(j, x^j; \lambda, \zeta) \\ H(Y;\theta) = \tilde{p}[f] - \dfrac{n_j/n}{\pi_j} f(x^j) \end{cases}$$

且 $h(\theta) = h(\lambda) = \tilde{p}[f] - p_\lambda[f]$ 对应于公式 (3-4)。在第3.3.3节中提到，最优的缩放矩阵 $A$ 应该满足 $A^{-1} = -\partial h(\lambda^*)/\partial\lambda$，直接计算可得：

$$C(\lambda) = -\frac{\partial h(\lambda)}{\partial \lambda} = \sum_{j=1}^{m} \pi_j \{p_{\lambda,j}[ff^T] - p_{\lambda,j}[f]p_{\lambda,j}[f]^T\} \tag{3-34}$$

其中 $p_{\lambda,j}[ff^T] = \sum_{x^j \in \mathcal{X}^j} p_j(x^j;\lambda) f(x^j) f^T(x^j)$，$p_{\lambda,j}[f] = \sum_{x^j \in \mathcal{X}^j} p_j(x^j;\lambda) f(x^j)$。通常来讲，$C(\lambda^*)$ 是未知的因此无法被直接计算，这里提出两种做法来进行近似。

#### 3.5.2.1　在线估计方差

一种解决思路是在线估计 $C(\lambda^*)$[61]，这种思路被用在 AugSA 最初的算法设计中[23]，算法通过将 $C(\lambda^*)$ 的对角元素向量 $\hat{\sigma}$ 当做变量引入到 SA 算法中，在每次迭代中更新它的值，具体更新公式如下：

$$\hat{\sigma}^{(t-\frac{1}{2})} = \frac{1}{K} \sum_{(j,x^j) \in B^{(t)}} f(x^j)^2 - \sum_{j=1}^{m} \pi_j (\bar{p}_j[f])^2$$
$$\hat{\sigma}^{(t)} = \hat{\sigma}^{(t-1)} + \gamma_{\hat{\sigma}} (\hat{\sigma}^{(t-\frac{1}{2})} - \hat{\sigma}^{(t-1)})$$

其中 $t$ 表示迭代时刻，$\bar{p}_j[f]$ 表示 $t$ 时刻采样获得的维度为 $j$ 的样本中特征的期望，即

$$\bar{p}_j[f] = \frac{1}{|B_j^{(t)}|} \sum_{(j,x^j) \in B_j^{(t)}} f(x^j),$$

$B_j^{(t)}$ 表示集合 $B^{(t)}$ 中所有维度为 $j$ 的样本组成的子集，$|B_j^{(t)}|$ 是其集合大小（样本个数）。在线更新的 $\hat{\sigma}$ 会被用于公式 (3-14) 中缩放梯度（替代 $\sigma$ 的位置）。

#### 3.5.2.2　使用经验方差

在本文的实验中采用了另一种做法，即使用经验协方差阵来近似 $C(\lambda^*)$。经验协方差矩阵的定义如下：

$$\tilde{C} = \sum_{j=1}^{m} \frac{n_j}{n} \{\tilde{p}_j[ff^T] - \tilde{p}_j[f]\tilde{p}_j[f]^T\},$$

其中 $\tilde{p}_j[f^T f] = n_j^{-1} \sum_{x^j \in D_j} f(x^j) f(x^j)^T$，$\tilde{p}_j[f] = n_j^{-1} \sum_{x^j \in D_j} f(x^j)$，本质上就是将公式 (3-34) 中的模型分布 $p$ 全部替换为经验分布 $\tilde{p}$，维度分布 $\pi_j$ 设为经验维度分

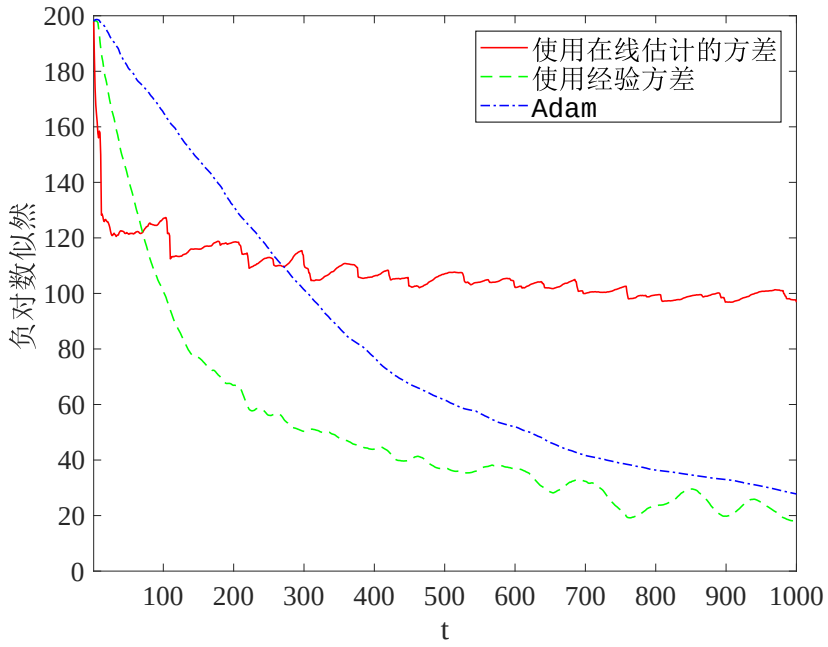



图 3.3　比较了三种不同的优化算法。红线表示使用在线估计的方差来缩放梯度(第3.5.2.1节)[23]，绿线表示使用经验方差来缩放梯度（第3.5.2.2节），蓝线表示 Adam 方法[34]。在 PTB 的训练集上进行训练模型，图中显示了不同算法的情况下，负对数似然值随迭代次数的收敛曲线。

布。这样的近似基于如下事实，如果拟合的分布 $p$ 足够接近经验分布 $\tilde{p}$，则 $\tilde{C}$ 就是 $C(\lambda^*)$ 的一个合理的估计。取 $\tilde{C}$ 的对角元素，就可以得到公式 (3-14)。

使用经验方差来缩放梯度的做法可以从另一个角度来理解。对于训练对数线性模型而言，基于梯度的优化算法可以通过对特征的均值和方差进行归一化来加速收敛[65]。事实上，更新公式 (3-14) 可以被看做如下形式：

$$\sigma^{1/2}\lambda^{(t)} = \sigma^{1/2}\lambda^{(t-1)} + \gamma_{\lambda,t}\left\{\tilde{p}[g] - \frac{1}{K}\sum_{(j,x^j)\in B^{(t)}}\frac{n_j/n}{\pi_j}g(x^j)\right\},$$

其中 $g(x^j) = \sigma^{-1/2}f(x^j)$ 表示被归一化的特征向量，其每个分量的方差都为 1，$\sigma^{1/2}\lambda$ 是其对应的参数向量。

相比于在 SA 迭代中在线估计 $C(\lambda^*)$，本文使用经验方差 $\sigma$ 的做法有如下好处：

(1) 首先，可以有效减少内存开销和计算开销，由于 $\sigma$ 可以被离线计算出来；

(2) 其次，可以使训练算法更加稳定。由于在线估计的 $\hat{\sigma}$ 方差很大，使用时往往需要引入一个阈值 $h$ 来限制其值不会过小，即使用 $\max\{\hat{\sigma}, h\}$，实际中阈值不能太小（如 $10^{-4}$），这使得大部分（约 80%）的方差被约束成为一个常数 $h$，因此算法退化为普通的梯度法。而对于经验方差 $\sigma$ 一般仅仅需要一个很

小的阈值（如 $10^{-15}$），这保证绝大多数的 $\lambda$ 都可以从方差的使用中受益。

图3.3中对比了两种做法的算法收敛性，可以看到，使用经验方差可以使模型的收敛更快。同时图中比较了一种自适应的优化算法——Adam 算法[34]，图中的结果显示，Adam 在实验中的收敛要差于使用经验方差的算法，却好于使用在线估计方差的算法。值得一提的是，在后文使用神经网络特征的 TRF 模型中，一般使用 Adam 来训练模型，这主要是因为 Adam 的实现简单，而对于神经网络这种复杂的势函数的情况，计算经验方差也变的十分困难。

### 3.5.3　维度分布

推论6给出了维度分布 $\pi_j$ 的最大似然估计，即 $\pi_j = n_j/n$。实际中不能直接使用经验维度分布 $n_j/n$，这主要有以下两个方面的原因：

(1) 经验概率有可能为 0，如果训练集中某个维度 $j$ 没有出现，则 $n_j/n = 0$，这使得跨维混合采样无法跳转到这个维度，也就无从估计其归一化系数 $\zeta_j$；

(2) 即使 $\pi_j$ 不为 0，若某个维度的概率 $\pi_j$ 过小，则会造成 $\zeta_j$ 估计震荡严重，这主要是因为更新公式 (3-16) 需要除以 $\pi_j$，从而严重影响算法的稳定性。

为了直观的解释上述问题，考虑如下情况，假设某个维度 $j$ 的概率为 $\pi_j = 10^{-5}$，每次迭代的产生 $K = 300$ 个不同维度的样本，如果维度 $j$ 在这批样本 $B^{(t)}$ 中至少出现一次，则 $\delta_j(B^{(t)}) \geq 1/300$，根据公式 (3-16)，$\zeta_j$ 的更新值至少为 $300^{-1}/10^{-5} = 1000/3$（忽略学习速率 $\gamma_{\zeta,t}$），这通常要远远大于 $\zeta_j$ 的真实值，尤其是在 $j$ 比较小的情况下。更进一步，如果 $j = 1$，由于更新公式 (3-17) 存在，所有的 $(\zeta_2, \ldots, \zeta_m)$ 都会受到影响。为了避免这种振荡，算法需要避免 $\pi_j$ 过小，尤其是避免 $\pi_1$ 过小。

本章使用如下方法来平滑 $\pi_j$：

$$\pi_j^0 = \frac{\max\{u_j, c\}}{\sum_{j=1}^{m} \max\{u_j, c\}}, \tag{3-35}$$

其中 $c$ 是一个常数（在本文的实验中设为 $10^{-5}$），

$$u_j = \begin{cases} n_{max}/n, & \text{if } j \leq j_{max} \\ n_j/n, & \text{if } j > j_{max} \end{cases}$$

$n_{max} = \max_j n_j$，$j_{max} = \arg\max_j n_j$。平滑算法的思路很简单，首先需要保证 $\pi_j$ 是一个随 $j$ 的非增函数，力求算法可以更有效的估计 $\zeta_1$，为了保证这一点，算法首先计算 $u_j$ 来提高 $j < j_{max}$ 处的概率；其次，为了避免 0 概率的出现，算法引入一

个常数 $c$ 来对调整后的概率再做一次平滑。在本文的实验中，这种平滑后的 $\pi_j^0$ 可以有效的提高 $\zeta_j$ 估计的准确度（见图3.4）。

### 3.5.4　引入类别信息加速采样

对于第3.3.4节中提的跨维混合采样算法，算法在第二步中使用 Gibbs 采样（表表3.2中的 Step 26）获得每个位置处的元素 $x_i^{(t)}$，这需要计算给定其他位置时第 $i$ 个位置处的条件概率。但是这个条件概率的计算十分复杂，这是因为条件概率的计算需要枚举第 $i$ 个位置处所有可能的取值，而取值集合 $\mathcal{A}$（如语言模型中的词表）可能会非常的大。因此，Gibbs 采样操作是跨维混合采样的计算瓶颈，限制了 TRF 模型在大规模数据上的使用。

本节介绍一种通过引入类别信息来降低采样计算开销的方法。首先，假设取值集合 $\mathcal{A}$ 中的每个取值都被分配到一个唯一的类别，这个类别集合为记为 $C$，总类别数目记为 $|C|$，那么，每个类别平均有 $|\mathcal{A}|/|C|$ 个值。这样，算法先采 $x_i^{(t)}$ 的类别，记为 $c_i^{(t)}$，然后固定类别采属于此类别的 $x_i^{(t)}$，这样平均而言计算开销就从 $|\mathcal{A}|$ 降到了 $|C| + |\mathcal{A}|/|C|$。这种使用类别来加速训练的思路常常被用在各种语言模型中，如最大熵模型[66] 和 RNN 语言模型[5]。

改进的采样算法的伪代码见表3.2中的算法 2，这里使用 $\mathcal{A}_c$ 来表示 $\mathcal{A}$ 中属于类别 $c$ 的子集。在马氏转移阶段（表3.2中 Step 21-28），对每个位置 $i$，首先通过一个提议分布 $Q_i(c)$ 来获得一个类别 $c_0$，然后使用如下概率来接受 $c_0$：

$$\min\left\{1, \frac{Q_i(c_i^{(t)})}{Q_i(c_0)} \frac{p_i(c_0)}{p_i(c_i^{(t)})}\right\} \tag{3-36}$$

其中

$$p_i(c) = \sum_{w \in \mathcal{A}_c} p(j^{(t)}, \{x_{1:i-1}^{(t)}, w, x_{i+1:j^{(t)}}^{(t)}\}; \lambda, \zeta)$$

概率 $Q_i(c)$ 和 $p_i(c)$ 实际上依赖于其它位置处的取值 $\{x_{1:i-1}^{(t)}, x_{i+1:j^{(t)}}^{(t)}\}$，但是为了简单起见省略了这些符号。随后算法从属于 $c_i^{(t)}$ 的所有可能的取值 $\mathcal{A}_{c_i^{(t)}}$ 中进行采样，获得 $x_i^{(t)}$。

同理，在局部跳转阶段，如果新产生的维度大于原始维度，即 $j = k + 1$（Step 7-14），同样可以使用类别信息来加速采样。首先，产生一个新的类别 $c_0 \sim Q_{k+1}(c)$，

然后根据类别 $c_0$ 基于如下分布来产生 $u$：

$$\breve{g}_{k+1}(u|x^k, c_0) = \frac{p(k+1, \{x^k, u\}; \lambda, \zeta)}{\sum_{w \in \mathcal{A}_{c_0}} p(k+1, \{x^k, w\}; \lambda, \zeta)} \tag{3-37}$$

其中 $x^k = x^{(t-1)}$。然后以如下概率来接受 $j^{(t)} = j$ 和 $x^{(t)} = \{x^{(t-1)}, u\}$：

$$\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, \{x^{(t-1)}, u\}; \lambda, \zeta)}{p(k, x^{(t-1)}; \lambda, \zeta) Q_{k+1}(c_0) \breve{g}_{k+1}(u|x^{(t-1)}, c_0)}\right\} \tag{3-38}$$

其实就是将公式 (3-19) 中的 $g_{k+1}(u|x^{(t-1)})$ 替换为类别 $c_0$ 概率和给定类别 $u$ 的概率的乘积，即 $Q_{k+1}(c_0)\breve{g}_{k+1}(u|x^{(t-1)}, c_0)$。

相应的 $j = k-1$ 的情况也需要作出改变，将接受概率 (3-20) 改为：

$$\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, x_{1:j}^{(t-1)}; \lambda, \zeta) Q_k(c) \breve{g}_k(x_k^{(t-1)}|x_{1:j}^{(t-1)}, c)}{p(k, x^{(t-1)}; \lambda, \zeta)}\right\}, \tag{3-39}$$

其中 $c = c_k^{(t-1)}$ 是第 $k$ 个位置处的类别。

实验中通过如下方式动态构建类别的提议分布 $Q_i(c)$，基本思路是通过选取 TRF 模型中和位置 $i$ 的类别有关的特征和参数来组成一个子模型，通过这个子模型来对类别采样。这里使用 $x^l$ 来表示当前时刻的序列（在算法 2 的 Step 8中表示的 $\{x^{(t-1)}, u\}$，Step 22中表示的 $x^{(t)}$）。首先，构建一个子模型 $p_l^c(x^l)$，模型中仅仅包含与 $x_i^l$ 的类别相关的特征和参数（参数使用当前时刻的估计值），然后定义如下分布：

$$Q_i(c) = p_l^c(\{x_{1:i-1}^l, c, x_{i+1:l}^l\})$$

由于 $p_l^c(x^l)$ 不依赖于 $x_i^l$ 而仅仅依赖于 $x_i^l$ 的类别，因此可以直接对类别采样。

### 3.5.5　超高维样本的处理

TRF 模型的训练中经常会遇到维度很大的样本（如长度很长的句子），尤其是在大规模的训练数据中，这种超高维样本的一个特点就是数量稀少，例如在本章使用的 Google 1-billion 语言模型数据集中[67]，句子的最大长度超过 1000，而长度超过 100 的句子仅仅占总句子数的 0.2%。为了能够减少混合模型中子模型的数目，本文将模型定义中的最大维度 $m$ 设为一个相对较小的值（如 100），然后在模型

(3-7) 的基础上引入一个特殊的子模型来对维度超过 $m$ 的样本进行建模：

$$p(j, x^j; \lambda, \zeta) = \frac{n_{m_+}/n}{Z_1(\lambda)} e^{\lambda^T f(x^j) - \zeta_{m_+}}, \quad j > m, \tag{3-40}$$

这时 $\zeta = (\zeta_1, \ldots, \zeta_m, \zeta_{m_+})^T, \zeta_1 = 0$ 表示模型待估计的归一化常数，$n_{m_+}$ 表示训练集中维度大于 $m$ 的样本数目。模型使用了一个归一化常数 $\zeta_{m+}$ 来对所有维度大于 $m$ 的样本进行建模，同时先验维度分布 $\pi_{m+} = n_{m_+}/n$ 同样需要经过平滑来保证归一化（第3.5.3节）。这时需要简单的调整 AugSA 算法来适应这种情况，首先变维混合采样（表3.2）的最大维度需要稍大于 $m$（本文的实验中设为 $m+2$），这样保证算法可以产生大于 $m$ 的样本来估计 $\zeta_{m_+}$；其次，$\zeta_{m_+}$ 可以和其他的 $\zeta_j$ 一样使用公式 (3-16) 和公式 (3-17) 进行更新，其中 $\delta_{m_+}(B^{(t)}) = K^{-1} \sum_{(j,x^j)\in B^{(t)}} 1(j > m)$ 表示样本中维度大于 $m$ 的样本频率。

### 3.5.6　并行采样

AugSA 算法（表3.1）中的采样部分可以很容的被修改为并行的方式。在每个迭代时刻 $t$，当前的参数 $\lambda$ 和归一化常数 $\zeta$ 为 $(\lambda^{(t-1)}, \zeta^{(t-1)})$，可以构建 $M$ 条独立的 MCMC 链，然后在 $M$ 个 CPU 上分别进行跨维混合采样，这样，为了获得一个包含 $K$ 个样本的样本集 $B^{(t)}$，每条链仅仅需要产生 $K/M$ 个样本，即进行 $K/M$ 次跨维混合采样，从而采样效率可以提高 $M$ 倍。由于 AugSA 算法的主要计算代价即为采样，因此整个算法的效率提高了 $M$ 倍

### 3.5.7　参数正则和终止判决

在实际应用中，常常需要引入正则项来缓解过拟合。假设 $L(\theta)$ 是模型在训练集上的似然函数，$\theta \in R^d$ 是待优化参数向量，相比于直接最大化似然函数 $L(\theta)$，即：

$$\theta^* = \arg\max_{\theta} L(\theta)$$

正则是指通过增加一个惩罚项 $P(\theta)$，然后最大化 $L(\theta) + P(\theta)$：

$$\theta^{**} = \arg\max_{\theta} L(\theta) + P(\theta)$$

其中 $P(\theta)$ 是一个 $\theta$ 的凸函数，且在 $\theta = 0$ 的时候取最大值，由于对数线性模型中 $\theta = 0$ 意味着均匀分布，因此 $\theta^{**}$ 即在似然最大和模型更均匀之间取折中，从而缓

解了模型过拟合在训练集上。常用的惩罚函数有 L1 正则项 $P(\theta) = -c||\theta||_1$ 和 L2 正则项 $P(\theta) = -c||\theta||_2$，其中 $c > 0$ 是一个正实数，$||\theta||_1 = \sum_{i=1}^{d} |\theta_i|$ 表示向量 $\theta$ 的一范数，$||\theta||_2 = \sqrt{\sum_{i=1}^{d} \theta_i^2}$ 表示向量 $\theta$ 的二范数。

令 $\mu \in R$ 表示 L2 正则常数，引入 L2 正则的参数 $\lambda$ 的更新公式 (3-14) 被修改为：

$$\lambda^{(t)} = \lambda^{(t-1)} + \gamma_{\lambda,t}(\sigma + \mu)^{-1} \left\{ -\mu\lambda^{(t-1)} + \tilde{p}[f] - \frac{1}{K} \sum_{(j,x^j)\in B^{(t)}} \frac{n_j/n}{\pi_j} f(x^j) \right\}$$

其中 $\sigma + \mu$ 表示对角矩阵 $\sigma$ 的每个对角元素都增加 $\mu$。

除了正则，另一个重要的问题是迭代终止条件的判决。一种选择是通过观测验证集上的似然值，当似然值没有显著变化时停止迭代，这种做法称为 “Early Stop”，常被用于 RNN 的训练[5]。但是对于 AugSA 算法来讲，对数似然的值取决于当前时刻估计的归一化系数 $\zeta_j$，训练过程中可能会存在振荡，因此直接观察开发集上的似然值并不稳定。因此本文使用另一种做法，即观察开发集和训练集上似然的比值，通过计算比值，可以消去归一化常数，从而避免了 $\zeta_j$ 的影响，这种方法被用于限制玻尔兹曼机（Restricted Boltzmann Machines，RBM）的训练[68]。

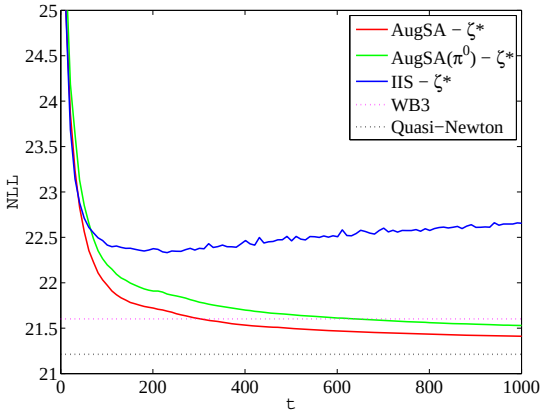
定义 TRF 模型 (3-7) 在给定的数据集 $B$ 上的对数似然值如下：

$$L_B(\lambda,\zeta) = \frac{1}{|B|} \sum_{(j,x^j)\in B} \log p(j,x^j;\lambda,\zeta) = \frac{1}{|B|} \sum_{(j,x^j)\in B} C(j,x^j;\lambda) - \sum_{j=1}^{m} \pi_j(B)\zeta_j$$

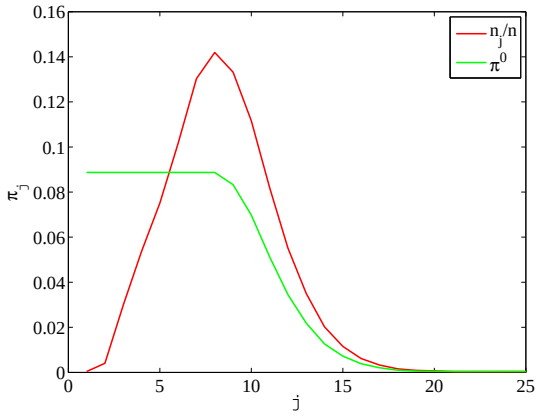
其中 $C(j,x^j;\lambda) = \lambda^T f(x^j) + \log \pi_j - \log Z_1(\lambda)$ 仅仅依赖于参数 $\lambda$ 而与 $\zeta$ 无关，$\pi_j(B)$ 表示数据集 $B$ 中的维度分布。则训练集 $B_r$ 和开发集 $B_d$ 上的对数似然值之差为（即似然值之比的对数）：

$$D_{B_r,B_d} = L_{B_r}(\lambda,\zeta) - L_{B_d}(\lambda,\zeta)$$

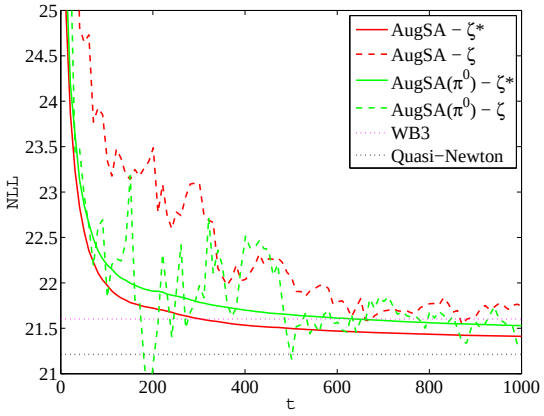
公式中可以发现，如果训练集 $B_r$ 和 $B_d$ 中维度分布相同，即 $\pi_j(B_r) = \pi_j(B_d)$，则 $D_{B_r,B_d}$ 仅仅依赖于 $\lambda$，而与 $\zeta$ 无关，由此，实验中只需要观测训练集和开发集上的对数似然值之差既可以观测模型的收敛情况，当差值没有明显变化时，则停止迭代。



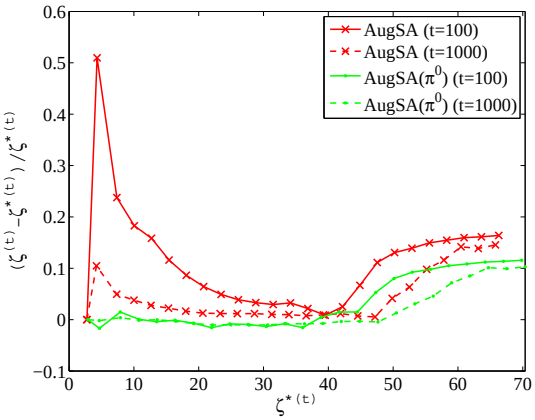

(a) 训练集上的 NLL




(b) 长度分布




(c) 训练集上的 NLL




(d) $(\zeta^{(t)} - \zeta^{*(t)})/\zeta^{*(t)}$ 随 $\zeta^{*(t)}$ 的变化

图 3.4　词形建模实验结果。(a)(c) 展示 TRF 模型在训练集上的负对数似然（NLL）随迭代次数的变化曲线，(b) 展示了经验长度分布 $n_j/n$ 和平滑后的长度分布 $\pi^0$（公式 (3-35)），(d) 展示了在迭代次数 $t = 100$ 和 1000 时，估计的归一化常数 $\zeta^{(t)}$ 与真实归一化常数 $\zeta^{*(t)}$ 的相对差距。

- “AugSA” 表示使用经验长度分布的 AugSA 算法，即 $\pi_j = n_j/n$；
- “AugSA($\pi^0$)” 表示使用平滑后的长度分布的的 AugSA 算法，即 $\pi_j = \pi^0_j$，详见第3.5.3节；
- “IIS” 表示提升迭代缩放算法（improved iterative scalling，IIS）；
- “WB3” 表示使用 Witten-Bell 平滑的 3-gram 模型；
- “Quasi-Newton” 表示使用精确梯度的拟牛顿算法；
- 短横杠之后的 “$\zeta^*$” 和 “$\zeta$” 分别表示 NLL 是使用当前参数时的准确的归一化系数和估计的归一化系数计算的。

## 3.6　仿真实验：词形建模

本节设计了一个仿真实验——词形建模实验，来确切的评估 TRF 模型和增强随机近似（AugSA）算法的性能。所谓词形建模，即将英文单词看做是字母序列，然后对字母序列进行建模，这种方法可以用来模拟单词的形态，进而自动产生满足与真实单词“相似”的字母序列[32]。在本节的实验中准确的评估了模型的似然值，对比了不同配置的 AugSA 算法与已有训练算法（如 IIS 算法）的收敛性。

实验的训练数据是从英文 English Gigword 数据集[①] 中提取的所有英文单词，通过排除重复的单词，生成一个英文单词列表，然后将列表按一定比例切分为训练集和测试集，最终训练集中包含 139 K 个不同的单词，测试集中包含 15 K 个英文的单词，单词的最大长度为 25（即 $m = 25$）。

为了构建 TRF 模型（公式 (3-1)），实验首先提取训练集中出现的所有 1 元、2 元、3 元字母对，来定义如下特征：

$$
\begin{aligned}
f_v(x^j) &= \sum_{i=1}^{j} f_{i,v}(x^j), \\
f_{vw}(x^j) &= \sum_{i=1}^{j-1} f_{i,vw}(x^j), \\
f_{vwy}(x^j) &= \sum_{i=1}^{j-2} f_{i,vwy}(x^j),
\end{aligned}
$$

其中 $x^j$ 表示一个包含 $j$ 个字母的单词 $(x_1^j, \ldots, x_j^j)$，$v$、$vw$ 和 $vwy$ 分别表示训练集中出现的 1 元、2 元、3 元字母对，即常说的 unigram、bigram 和 trigram，$f_{i,v}$、$f_{i,vw}$、$f_{i,vwy}$ 则表示位置 $i$ 处的 1 阶、2 阶、3 阶特征：

$$
\begin{aligned}
f_{i,v}(x^j) &= \begin{cases} 1 & \text{if } x_i^j = v \\ 0 & \text{otherwise,} \end{cases} \\
f_{i,vw}(x^j) &= \begin{cases} 1 & \text{if } x_i^j = v \text{ and } x_{i+1}^j = w \\ 0 & \text{otherwise,} \end{cases} \\
f_{i,vwy}(x^j) &= \begin{cases} 1 & \text{if } x_i^j = v \text{ and } x_{i+1}^j = w \text{ and } x_{i+2}^j = y \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

实验中将如下特征加入到 TRF 模型中：$f_v$、$f_{vw}$、$f_{vwy}$、$f_{i=0,v}$、$f_{i=end,v}$、$f_{i=0,vw}$ 和 $f_{i=end,vw}$，其中 $f_{i=0,\bullet}$ 表示字母对出现在单词的开头，而 $f_{i=end,\bullet}$ 表示字母对出现在

① https://catalog.ldc.upenn.edu/LDC2003T05

单词的结尾，引入这种开头、结尾的特征主要是为了使 TRF 模型与经典的 3-gram 模型[2] 包含相同的特征，最终的总特征数目为 $d = 11,929$。由于在此实验中没有使用类别信息，且模型简单也不需要对算法进行额外的加速（$\mathcal{A}$ 即为 26 个英文字母），因此实验使用普通版本的 AugSA 算法（表3.1，采样使用表3.2中的算法 1）来训练 TRF 模型。在每次迭代时，使用跨维混合采样（第3.3.4节）产生 $K = 100$ 个序列（即单词），学习速率使用公式 (3-32) 和公式 (3-33)，参数配置为 $t_c = 100$，$\beta_\lambda = 0.8$，$\beta_\zeta = 0.6$，$t_0 = 200$，最大迭代次数固定为 $t_{max} = 1000$。实验尝试将长度分布 $\pi_j$ 设为经验长度分布 $n_j/n$ 或者平滑后的长度分布 $\pi_j^0$（公式 (3-35)）来观察长度分布的选择对 $\zeta$ 估计的影响。第3.5.4节到第3.5.7节中提到的优化技术没有用在这个实验中。

实验对比了如下三种已有的方法：

(1) 首先是被广泛应用于语言建模中的 $n$-gram 模型[2]，实验将英文单词看做是字母序列，使用开源工具包 SRILM[①]训练了一个 3gram 模型，平滑算法使用 Witten-Bell 折扣平滑（记为 “WB3”）。这个 3gram 模型与 TRF 模型包含完全一样的特征，因此可以作为基线来衡量 TRF 模型在参数拟合方面的优势。

(2) 实验尝试使用提升迭代缩放（improved iterative scaling，IIS）算法[32] 来训练 TRF 模型，为了与 AugSA 算法进行对比，每次迭代同样产生 $K = 100$ 个随机长度的英文单词，而与 AugSA 不同的是，IIS 的采样并不使用第3.3.4节中提出的跨维混合采样算法，而是使用文献[32] 中的做法，首先根据经验长度 $n_j/n$ 产生一个长度 $j$，然后以上一次产生的长度为 $j$ 的序列为初值，使用 Gibbs 采样来产生一个新的长度为 $j$ 的序列。

(3) 由于在本实验中，期望 $p_\lambda[f]$ 可以被精确的计算出来，因此可以使用拟牛顿（Quasi-Newton）方法来求解方程 (3-4)（记为 Quasi-Newton）。这个方法提供了 TRF 模型最大似然估计的一个准确解。但是对于更加复杂的 TRF 模型，如语言模型，精确计算期望 $p_\lambda[f]$ 依然不可行，因此拟牛顿方法也无法使用。

为了对比不同方法的性能，图3.4(a)展示了不同模型收敛性能。对于 AugSA 和 IIS 这种迭代算法，图中画出了模型的负对数似然值（NLL）随迭代次数的变化曲线，为了移除归一化系数的影响，这里均使用准确的归一化系数来计算 NLL；对于 $n$gram 模型（WB3）和拟牛顿法训练的 TRF 模型（Quasi-Newton），图中使用横线标出了最终模型的 NLL，作为两个基准值。通过对比 WB3 和 Quasi-Newton 的结果可以看出，虽然 TRF 与 3gram 使用完全一致的特征，但是 TRF 的性能超越了 3gram 模型，这是由于 TRF 避免了标签偏置的问题，使得模型在有限特征的情况

① http://www.speech.sri.com/projects/srilm/

下可以进行更好的平滑（详见第3.4.1节）。需要注意的是，图3.4中展示的是训练集上的 NLL，而对于每种算法而言，测试集上的似然值表现与其在训练集上的表现高度一致，这主要是因为任务中训练集数据足够充分，且任务足够简单。实验的结果分析如下。

- 首先，IIS（蓝实线）在实验中表现不佳，NLL 在迭代次数超过一定值时开始发散，这是由于每次迭代产生的 $K = 100$ 个样本不足以很好的估计期望 $p_\lambda[f]$；相对的，AugSA 的算法（红实线和绿实线）表现却相当稳定，使用与 IIS 相同的采样数目，NLL 稳步收敛到精确解（Quasi-Newton 标定的横线），这表明即使使用很少的样本数目，AugSA 也能获得很好的收敛；
- 使用 AugSA 训练的 TRF 模型在使用相同特征的情况下，可以达到比 WB3 更低的 NLL，从而说明 TRF 在参数数拟合方面超越传统的 $n$-gram 模型，这是由于 TRF 避免了标签偏置的问题，使得模型在有限特征的情况下可以进行更好的平滑（详见第3.4.1节）；
- 其次，将长度分布从 $n_j/n$（红实线）修改为 $\pi_j^0$（绿实线）后（平滑前后长度分布见图3.4(b)），虽然轻微减缓了收敛速度，但是这一改动却使得模型对归一化系数的估计更加准确（如图3.4(d)）。

为了分析 AugSA 对归一化系数的估计情况，图3.4(c)对比了 AugSA 算法中使用准确归一化系数（实线）和使用估计的归一化系数（虚线）计算的 NLL，结果发现使用平滑后的 $\pi_j^0$ 可以显著提高模型对归一化系数估计的准确度，从而使得绿虚线可以更加靠近绿实线。事实上，使用经验长度分布 $n_j/n$ 高估了 NLL，如图3.4(c)中红虚线始终在红实线的上方。此外，图3.4(d)展示了在 $t = 100$ 和 $t = 1000$ 两个时刻，真实的归一化系数 $\zeta^{*(t)}$ 和估计的归一化系数 $\zeta^{(t)}$ 之间的相对差距，结果发现使用平滑后的长度分布 $\pi_j^0$，估计的 $\zeta^{(t)}$ 要更接近真实值 $\zeta^{*(t)}$，尤其是在归一化系数较小的时候。

## 3.7　语言模型实验

本节中将 TRF 模型用于实际的语言建模任务，并在模型混淆度（PPL）和语音识别词错误率（WER）上来评估 TRF 语言模型的性能。本节介绍了两个实验，首先，本节在 PTB 数据集上训练语言模型，并将其用于 WSJ'92 测试集上进行语音识别实验；随后，为了验证 TRF 语言模型是否能在更大规模的语料上进行训练，实验选择了 Google 1billion 数据集，通过逐渐增加训练集规模观察模型性能的变化。每个实验中，TRF 语言模型都与目前主流的语言建模方法（如 $n$-gram 模型、RNN 和 LSTM 语言模型）进行了对比，同时进行了模型插值实验来验证 TRF 语言模型

表 3.3　模型插值方法。$s(x^j)$ 表示插值后的长度为 $j$ 的序列 $x^j$ 的语言模型得分，可以被用于语音识别；$p_1(x_i|h_i)$ 和 $p_2(x_i|h_i)$ 表示给定历史词序列 $h_i$ 当前词 $x_i$ 的条件概率；$p_1(x^j)$ 和 $p_2(x^j)$ 表示句子 $x^j$ 的联合概率；$0 < \alpha < 1$ 是插值系数，一般需要在开发集上调到最优。

| | |
|---|---|
| W | $s(x^j) = \log\{\prod_{i=1}^{j} [\alpha p_1(x_i\|h_i) + (1-\alpha)p_2(x_i\|h_i)]\}$ |
| S | $s(x^j) = \log\{\alpha p_1(x^j) + (1-\alpha)p_2(x^j)\}$ |
| Log | $s(x^j) = \alpha \log p_1(x^j) + (1-\alpha) \log p_2(x^j)$ |

与已有语言模型的互补性。本节首先对语言模型插值技术进行介绍。

### 3.7.1　模型插值

在语音识别任务中，通常使用模型插值的方式来对模型进行组合，从而获得比单独使用每个模型更好的效果。模型插值有如下三种做法（见表3.3）：

- 词概率插值（W）。这种方法主要针对条件概率模型，假设两个语言模型在给定历史词 $h_i$ 的情况下对当前词 $x_i$ 的条件概率分别为 $p_1(x_i|h_i)$ 和 $p_2(x_i|h_i)$，则插值后的条件概率模型为 $p(x_i|h_i) = \alpha p_1(x_i|h_i) + (1-\alpha)p_2(x_i|h_i)$；
- 句概率插值（S）。与词概率插值相对，句子概率插值通过计算两个模型对句子的联合概率的插值。假设两个模型对长为 $j$ 的序列 $x^j$ 联合概率分别为 $p_1(x^j)$ 和 $p_2(x^j)$，则插值后的句子联合概率为 $p(x^j) = \alpha p_1(x^j) + (1-\alpha)p_2(x^j)$；
- 对数线性插值（Log）。由于识别中仅仅需要每个句子的得分，而模型得分一般使用每句话的对数概率，因此可以对对数概率做插值来获得一个新的语言模型得分，即对数线性插值。假设两个模型在长为 $j$ 的序列 $x^j$ 上的联合概率分别为 $p_1(x^j)$ 和 $p_2(x^j)$，则插值后的这句话的语言模型得分为 $s(x^j) = \alpha \log p_1(x^j) + (1-\alpha) \log p_2(x^j)$。这种插值方法与前两种方法有所不同，首先，前两种插值方法是对概率做插值，插值后的概率可以保证归一化，而对数线性插值获得的是一个不满足归一化的“得分”；其次，前两种方法插值后的概率需要计算对数来用于语音识别，而对数线性插值的结果直接就是语言模型的“得分”，因此可以直接用于语音识别。

下文的实验中选择对数线性插值来进行模型组合，一方面是因为这种插值的实现简单，可以兼容条件概率模型和整句概率模型；另一方面实验发现 TRF 模型与神经网路语言模型的对数线性插值好于其他插值方法。

### 3.7.2　PTB 数据集上语言建模实验

本节将 TRF 模型用于语言建模任务中，实验对比了不同语言建模方法的混淆度（PPL）和其在语音识别中的词错误率（WER）。

表 3.4　TRF 语言模型中定义的特征

| 代号 | 包含特征 |
|---|---|
| w | $(w_{-3}w_{-2}w_{-1}w_0)$, $(w_{-2}w_{-1}w_0)$, $(w_{-1}w_0)$, $(w_0)$ |
| c | $(c_{-3}c_{-2}c_{-1}c_0)$, $(c_{-2}c_{-1}c_0)$, $(c_{-1}c_0)$, $(c_0)$ |
| ws | $(w_{-3}w_0)$, $(w_{-3}w_{-2}w_0)$, $(w_{-3}w_{-1}w_0)$, $(w_{-2}w_0)$ |
| cs | $(c_{-3}c_0)$, $(c_{-3}c_{-2}c_0)$, $(c_{-3}c_{-1}c_0)$, $(c_{-2}c_0)$ |
| wsh | $(w_{-4}w_0)$, $(w_{-5}w_0)$ |
| csh | $(c_{-4}c_0)$, $(c_{-5}c_0)$ |
| cpw | $(c_{-3}c_{-2}c_{-1}w_0)$, $(c_{-2}c_{-1}w_0)$, $(c_{-1}w_0)$ |
| tied | $(c_{-9:-6}, c_0)$, $(w_{-9:-6}, w_0)$ |

实验中，训练集和开发集来自于 Penn Treebank 数据集中的 WSJ 部分（记为“PTB 数据集”），这批数据常被用于语言模型的评估[5]，原始数据被预先分为 24 部分，实验选取 0-20 部分为训练集，21-22 部分为开发集，23-24 部分为测试集（本实验中没有用到），最终训练集、开发集和测试集分别包含 930K、74K、82K 词。实验使用一个 10K 大小的词典，其中包含一个表示集外词的辅助符号“⟨UNK⟩”。实验对比了多个基线模型，包括：

- $n$-gram 语言模型，平滑算法使用 Modified Kneser-Ney 平滑[2]（记为“KN$n$”），使用开源工具包 SRILM[①]训练获得；
- 递归神经网络语言模型（RNN LM）[5]，包含 1 个隐层，隐层节点数为 250，为了保证模型性能，实验中使用标准的 Softmax 输出层，没有使用任何加速 Softmax 计算的方法。模型使用开源工具 RNNLM toolkit[②]训练获得；
- LSTM 语言模型。模型包含 2 个隐层，每个隐层包含 250 个节点，因为模型规模比较小并不容易过拟合，因此没有使用 dropout，使用文献[40] 中提供的开源工具来训练模型。

对于 TRF 模型，实验考虑引入更丰富的特征来进行语言建模，见表3.4。整体而言，实验考虑同时使用词信息和类信息。首先通过一种自动聚类算法[27] 来将每个词分配到一个类别中，总类别数目为 200（这里尝试了不同的类别，从 100 到 200 到 500，发现 200 类效果最好）。表3.4中列出了 TRF 使用的所有特征，$w_i$ 和 $c_i$（i=0,-1,...,-5）分别表示在每个相对位置 $i$ 处的词和类别，例如 $w_0$、$c_0$ 表示当前位置的词和相应的类别，而 $w_{-1}$、$c_{-1}$ 表示当前位置的前一个位置处的词和类别。最终 TRF 特征的定义如下：

- “w” / “c” 表示词/类的 $n$ 元文法特征，即 $n$gram 特征，这些特征与传统的

① http://www.speech.sri.com/projects/srilm/

② http://rnnlm.org/

*n*gram 中用到的特征相对应，例如 $(w_{-1}w_0)$ 表示词的二元文法（bigram）特征，而 $(c_{-3}c_{-2}c_{-1}c_0)$ 表示类别的四元文法（4gram）特征；

- “ws” / “cs” 表示跳跃式的 *n*gram 特征，这些特征与第2.2.1节中介绍的特征类似，即跳过一个或多个位置来考虑词/类的约束关系，如 $(w_{-3}w_0)$ 就考虑相距为 3 的两个词的约束关系；
- “wsh” / “csh” 表示高阶的跳跃式的 *n*gram 特征，这里定义了距离为 4 和 5 的 bigram 约束；
- “cpw” 表示词和类的混合特征，之前定义的特征中，单个特征仅仅包含词和类信息中的一种，而所谓混合特征顾名思义，表示单个特征中即用到词信息也用到类别信息，如 $(c_{-1}w_0)$ 即考虑了当前位置处的词和前一个位置处的类别；
- “tied” 表示一类共享参数的特征[69]，例如 $(w_{-9:-6}, w_0)$ 表示距离为 6 到 9 的跳跃式词的 bigram 共享同一个参数，同理 $(c_{-9:-6}, c_0)$ 表示距离为 6 到 9 的跳跃式类别的 bigram 共享同一个参数，这种做法的主要目的就是减少参数规模，一方面使模型更容易训练，一方面也可以防止过拟合。

表3.4中的所有特征都定义成公式 (3-2) 中所示的位置独立和长度独立的形式，并且只有出现在训练集中的特征才被加入到模型中。

本实验使用加速版的 AugSA 算法（表3.1，采样使用表3.2中的算法 2）来训练 TRF 模型。由于此数据集中的句子长度较小，最大长度为 82，因此不需要第3.5.5节中的技术，除此之外，第3.5节中的其他的所有优化技术都被用在此实验中，具体的配置参数如下：

- 模型的最大长度为 $m = 82$，即数据集中的最大长度；
- 每次迭代，产生 $K = 300$ 个不同长度的句子；
- 学习速率 $\gamma_\lambda$ 和 $\gamma_\zeta$ 使用公式 (3-32) 和公式 (3-33) 中的定义，配置参数为 $t_c = 3000$、$\beta_\lambda = 0.8$、$\beta_\zeta = 0.6$ 、$t_0 = 2000$；
- 长度概率使用平滑后的 $\pi_j^0$（公式 (3-35)）；
- 使用 L2 正则来防止过拟合，正则常数 $\mu = 4 \times 10^{-5}$（公式 (3.5.7)），同时第3.5.7节中的终止判决方法被用于判断迭代终止条件；
- 使用 8 个 CPU 来进行并行采样，见第3.5.6节。

实验尝试了使用 IIS 方法来估计 TRF，但是正如词形建模实验的结果所示，训练效果很差，且病态的发散，因此并没有列在结果中。

为了能够评估语言模型在语音识别中的性能，即词错误率（WER），实验首先在 PTB 训练集上训练语言模型，然后使用重新打分的形式来计算识别 WER。测试

表 3.5　WSJ'92 测试集上的词错误率（WER）、负对数似然值（NLL）和混淆度（PPL）。“10 TRFs”表示进了 10 词独立的 AugSA 训练获得 10 个参数不同的 TRF 模型，然后计算了 10 个模型的“均值 ± 标准差”，同时表中展示了其中一个 TRF 模型的结果。“Sep Std/Sm AIS”表示标准的 AIS/平滑后的 AIS。“参数”表示特征（参数）数目，单位是百万（million）。

| Model | WER (%) | NLL | PPL | 参数 (M) |
|---|---|---|---|---|
| KN4 (cutoff=0000) | 8.71 | 98.0 | 295.4 | 1.6 |
| KN5 (cutoff=00000) | 8.78 | 97.9 | 294.4 | 2.3 |
| KN6 (cutoff=000000) | 8.61 | 97.9 | 293.7 | 3.0 |
| RNN[5] | 7.90 | 95.6 | 257.6 | 5.1 |
| LSTM[6] [40] | 7.87 | 98.6 | 306.3 | 6.0 |
| 10 TRFs：特征“w+c+ws+cs+wsh+csh+tied”，200 类别，进行了 10 次独立的 AugSA 训练 | | | | |
| AugSA | **7.90±0.07** | 95.3±0.5 | 252.7±7.3 | 7.7 |
| Sep Std AIS | 8.33±0.23 | 102.3±1.4 | 380.9±32.4 | 7.7 |
| Sep Sm AIS | 7.89±0.09 | 102.2±1.0 | 378.9±22.5 | 7.7 |
| Global AIS | 7.90±0.07 | 116.2±10.3 | 1066.4±889.5 | 7.7 |
| 10 个 TRF 中某一个模型的结果 | | | | |
| AugSA | 8.03 | 95.1 | 250.4 | 7.7 |
| Sep Std AIS | 8.41 | 103.7 | 412.4 | 7.7 |
| Sep Sm AIS | 8.02 | 103.3 | 402.9 | 7.7 |
| Global AIS | 8.03 | 122.5 | 1225.7 | 7.7 |
| 模型插值结果，使用对数线性插值，插值系数 $\alpha = 0.5$ | | | | |
| RNN+KN5 | 8.00 | | | |
| RNN+TRF | 7.71±0.06 | | | |
| LSTM+KN5 | 8.09 | | | |
| LSTM+TRF | **7.59±0.06** | | | |

表 3.6　表3.5中不同模型训练时间和推理时间对比，训练时间表示模型从开始训练到训练结束的总时间，推理时间表示平均而言模型对每句话的 1000 个候选重新打分的时间，即识别出每句话的时间。

| Model | 训练时间 | 推理时间 | 参数 (M) |
|---|---|---|---|
| KN5 (cutoff=00000) | 22 秒（1 CPU） | 0.06 秒（1 CPU） | 2.3 |
| RNN | 25 小时（1 CPU） | 40 秒（1 CPU） | 5.1 |
| LSTM | 1 小时（1 GPU） | 10 秒（1 GPU） | 6.0 |
| TRF(AugSA) | 20 小时（8 CPUs） | **0.16 秒**（1 CPU） | 7.7 |

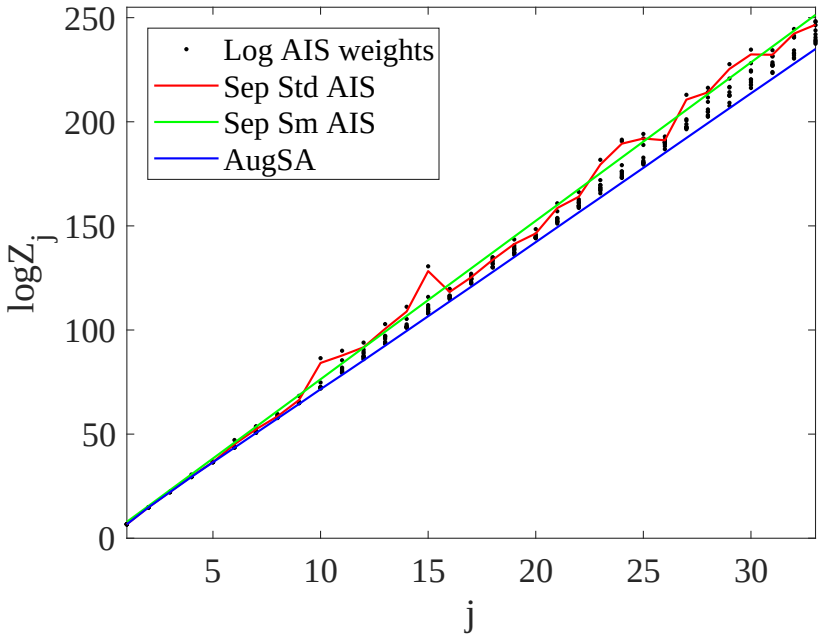



图 3.5　对一个 TRF，使用不同方法计算的对数归一化系数 $\log Z_j, j = 1,\ldots,33$。不同的归一化系数情况下模型的 WER、NLL、PPL 显示在表3.5中。点（Log AIS weight）表示 AIS 的 10 条链计算的对数权重。

集为 WSJ'92 测试音频（包含 330 句话），首先使用开源工具包 Kaldi[①]来对测试音频进行一遍识别，识别使用一个基于 DNN 的声学模型和一个 3gram 语言模型，生成了一个 1000-best 的多候选结果，多候选的神谕错误率（oracle WER）为 0.93%。

由于 AugSA 的训练有一定的随机性，且归一化常数 $\zeta_j$ 是估计值，因此实验中总共进行了 10 词独立的 AugSA 训练，计算了所获得的 10 个 TRF 模型的 PPL 和 WER 的均值和标准差。实验将 TRF 语言模型与基线模型（$n$-gram、RNN、LSTM）进行对比，结果显示在表3.5中。对实验结果做如下分析。

第一，丰富特征的使用，尤其是"tied"特征的使用展示了模型对于复杂特征的灵活支持，这是 TRF 模型相比于其他语言模型的最大优势，结果表明，通过引入丰富的特征，TRF 语言模型获得大约 7.90% 的 WER，超越了 KN5 语言模型，相对错误率下降高达 10%，获得与 RNN 语言模型和 LSTM 语言模型相近的性能。

第二，TRF 模型展现了和条件概率模型 RNN、LSTM 语言模型良好的互补性，实验使用了句子级别的对数线性插值（详见表3.3），即将两个模型计算的同一句话的对数概率做插值，插值系数分别使用 $0 < \alpha < 1$ 和 $1 - \alpha$。结果表明，TRF 与 LSTM 的插值（"LSTM+TRF"）可以获得最低的 WER，约 7.59%，比 KN5 模型相对下降 13.6%，比 LSTM 模型相对下降 3.2%。

① http://kaldi.sourceforge.net/

第三，模型的训练时间如表3.6，训练单个 TRF 模型、RNN 模型和 LSTM 模型的训练时间分别是 20 小时（使用 CPUs），25 小时（使用 CPUs）和 1 个小时（使用 GPU），而对于测试效率，TRF 模型计算句子的概率比 RNN、LSTM 模型要高效，这是由于 TRF 模型不需要繁琐是局部归一化，而归一化的计算往往是神经网络语言模型的计算瓶颈。具体而言，TRF、RNN 和 LSTM 对每句话的 1000 个候选重新打分的时间分别为：0.16 秒、40 秒和 10 秒，TRF 的推理效率大约是 RNN 的 250 倍、LSTM 的 63 倍。此外，在本实验中，由于 RNN 和 LSTM 的使用标准的 Softmax 层，为了保证模型性能没有引入任何加速技术，基于一些文献中的结果[20]，使用一些基于类别的加速策略可以使 LSTM 的计算提速大概 2 到 10 倍，这依然比 TRF 的速度慢。

第四，AIS[70] 方法同样可以被用来估计归一化系数，为了验证 AugSA 对归一化系数估计的好坏，实验使用 AIS 来对 10 个 TRF 模型的归一化系数重新估计，这里存在两种做法：第一种做法是基于模型定义 (3-1)，使用 AIS 分别对每个维度下的归一化系数进行估计；第二种做法是将模型看成全局归一化模型 (3-23)，固定 AugSA 估计的 $\zeta$，对全局归一化系数进行估计，这种情况下需要用来跨维混合采样来产生样本。两种方法的具体实现方式如下：

- 对于第一种方法，对每个维度的归一化系数 $Z_j$ 都需要进行一次 AIS 算法，每次 AIS 包含 10 条链，20K 个中间分布，基准分布为将所有参数 $\lambda$ 都设为 0。由于 1000 候选的最大长度为 33，为了节约时间，这里仅仅估计维度 $j$ 从 1 到 33 的归一化系数，整个计算过程耗时 20 小时（即使使用了 8 个 CPU 并行计算）。由于每个 $Z_j$ 是独立估计的，因此不同维度的估计的归一化系数存在很大的震荡，如图3.5中红线所示，这造成识别结果显著变差，为了能缓解这种震荡，对每个模型的归一化系数，实验对 AIS 估计的 $\log Z_j, j = 1,\ldots,33$ 进行了线性回归，获得一个平滑后的结果，平滑后的 $\log Z_j$ 是 $j$ 的线性函数，如图3.5中绿线所示，然后使用平滑后的归一化系数来计算错误率。为了能够区分平滑和非平滑的做法，这里使用 “Sep Sm AIS” 来表示平滑后的结果，使用 “Sep Std AIS” 表示平滑前的结果，WER 结果显示在表3.5中。结果显示平滑后的结果与 AugSA 的结果相近，从而从另一个侧面验证了 AugSA 算法对归一化系数估计的有效性。
- 对于第二种做法，这里称之为 “Global AIS”，每次 AIS 使用跨维混合采样来产生不同维度的样本，包含 10 条链 200K 的中间分布，基准分布为将所有的 $\lambda$ 设为 0，$\nu$ 根据 $\zeta^*(0)$ 赋值。计算每个 TRF 的全局归一化系数耗时 8 小时（使用 10CPU 进行链间并行）。引入全局归一化系数不影响识别 WER，但是

会影响 PPL 和 NLL。结果总结在表3.5中。

### 3.7.3　中等规模语料语言建模实验

本节主要验证 TRF 语言模型的可拓展性（Scalability），即随着训练语料的增加和特征数目的膨胀，TRF 语言模型是否依然保持性能的优势，AugSA 是否依然有效。由于 PTB 数据集规模较小，本节选择了一个较大规模的数据集 Google 1billion 英文数据集[①]进行实验。数据集中，训练集包含 99 个文本文件，每个文件包含大概 8M 词，预留集包含 50 个文件，每个文件包含大概 160K 词，实验选择预留集中的第一个和第二个文件（即 “news.en.heldout-00000-of-00050” 和 “news.en.heldout-00001-of-00050”）分别作为开发集和测试集，然后逐步增加训练集来观察模型性能的变化。

首先，使用训练集的一个文件（大约 8M 词）作为训练集来训练模型，实验提取出现频率最高的 20K 个词组成词表，训练集、开发集和测试集中，词表外的词都被统一映射成一个辅助符号 “⟨UNK⟩”，随后使用自动聚类算法[27] 来将词表分为 200 类。同时在当前训练集上训练各种语言模型用作对比，包括 $n$-gram、RNN、LSTM 和 TRF 模型。TRF 模型的的特征选择表3.4中的 “w+c+ws+cs+wsh+csh+tied”，然后使用 AugSA 算法来进行参数训练。随后，将训练数据扩大为月 16M 词（2 个训练集文件），依然提取一个 20K 的字典然后重新聚类成 200 类，然后在新的语料上训练各种语言模型，TRF 模型的特征选择不变。以此类推，继续降训练数据扩大为 32M 词（4 个训练集文件）然后重复上述操作。

在整个实验中，由于语料中句子的最大长度很长，超过 1000，且长度超过 100 的句子仅仅占总句子数的 0.2%，因为本实验使用了第3.5.5节中提出的技术来定义 TRF 模型，来减小训练代价。TRF 配置如下：

- 最大长度为 $m = 100$，为了能够估计 $\zeta_{m+}$，在每次 AugSA 迭代中，算法产生 $K = 300$ 个长度从 1 到 102 的样本，从而保证样本长度可以超过 $m = 100$；
- 学习速率使用公式 (3-32) 和公式 (3-33)，具体参数为 $t_c = 1000$、$\beta_\lambda = 0$、$\beta_\zeta = 0.6$、$t_0 = 2000$；
- 长度分布使用平滑后的长度分布 $\pi_j^0$（公式 (3-35)）；
- 迭代次数固定为 $t_{max} = 50{,}000$ 来保证模型的充分训练，L2 正则常数为 $10^{-5}$；
- 实验使用 12 个 CPU 核来并行运行训练算法。

实验在测试集上计算了不同模型的混淆度（PPL），同时将语言模型用于 WSJ’92 的测试集上进行语音识别实验（对每句话的 1000 个候选重新打分），PPL 和 WER

① https://github.com/ciprian-chelba/1-billion-word-language-modeling-benchmark

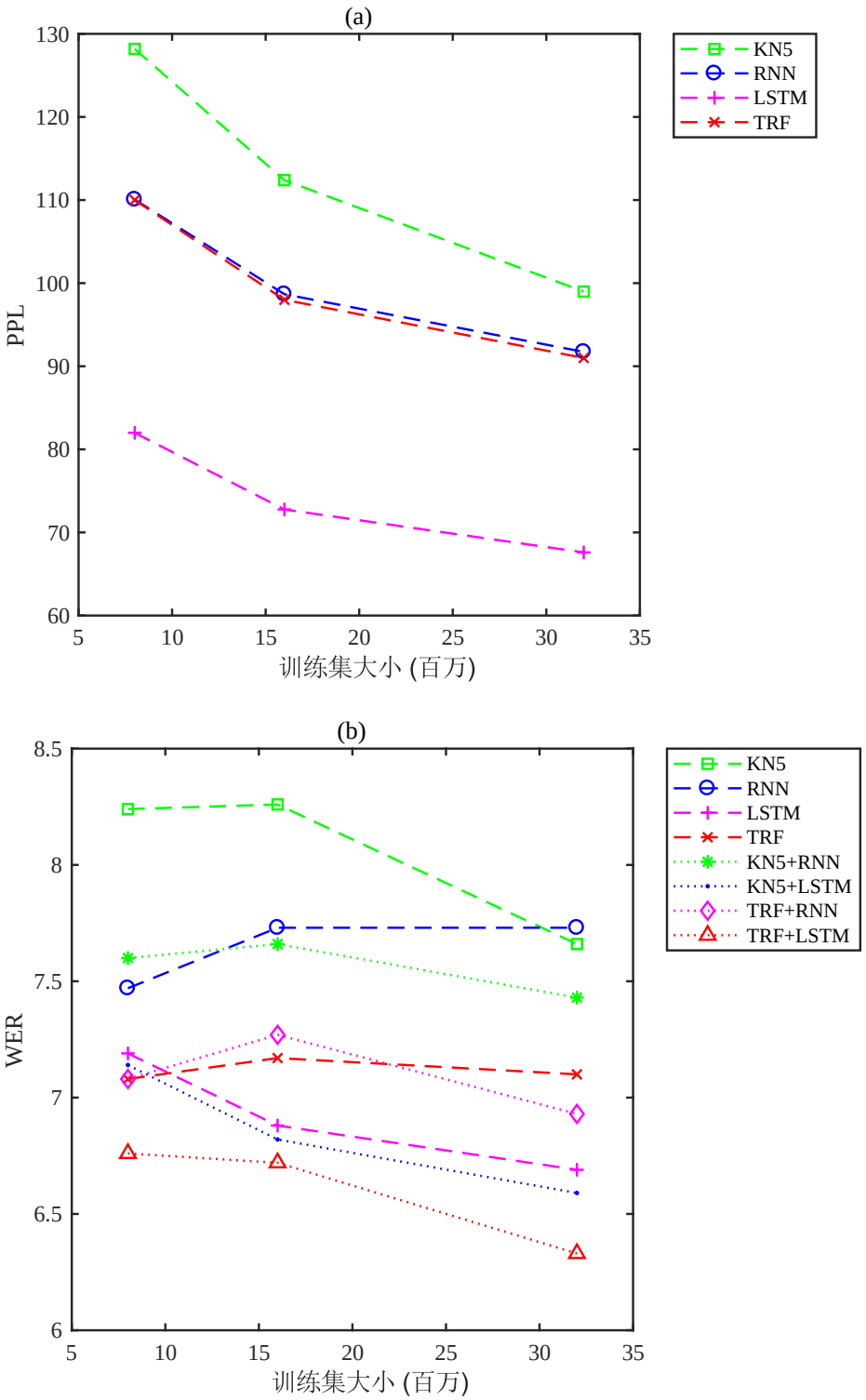



图 3.6　不同语言模型混淆度（PPL）和识别词错误率（WER）随训练语料大小的变化趋势。

表 3.7　不同模型的混淆度随训练语料大小（8 M、16 M、32 M）的变化。
KN5 的剪枝参数为 00002；RNN[5] 包好 1 个隐层 250 个隐层节点，且使用 200 类别来加速；LSTM[40] 使用了 2 个隐层 250 个隐层节点，无类别信息；TRF 模型的特征为“w+c+ws+cs+wsh+csh+tied”。“#feat”表示参数（特征）数目，单位是百万（million）.

| models | 8 M | | | 16 M | | | 32 M | | |
|---|---|---|---|---|---|---|---|---|---|
| | PPL | WER | #feat | PPL | WER | #feat | PPL | WER | #feat |
| 单个模型的结果 | | | | | | | | | |
| KN5 | 128 | 8.24 | 18.1 | 112 | 8.26 | 33.6 | 99 | 7.66 | 61.8 |
| RNN | 110 | 7.47 | 9.0 | 99 | 7.73 | 9.0 | 92 | 7.73 | 9.0 |
| LSTM | 82 | 7.19 | 9.9 | 73 | 6.88 | 9.9 | 68 | 6.69 | 9.9 |
| TRF | ~110 | 7.08 | 17.5 | ~98 | 7.17 | 29.2 | ~91 | 7.10 | 48.7 |
| 模型插值后的 WER，插值系数 $\alpha = 0.5$ | | | | | | | | | |
| RNN+KN5 | | 7.60 | | | 7.66 | | | 7.43 | |
| LSTM+KN5 | | 7.14 | | | 6.82 | | | 6.59 | |
| RNN+TRF | | 7.08 | | | 7.27 | | | 6.93 | |
| LSTM+TRF | | **6.76** | | | **6.72** | | | **6.33** | |

显示在表3.7和图3.6中。这些结果展现了 TRF 语言模型的可拓展性，在包含 32M 词的训练语料上依然保持模型性能，且 PPL 和 WER 都显著超越传统的 KN5 模型。TRF 模型与神经网络模型展现出优良的互补性，TRF 模型与 LSTM 模型的插值可以获得最低的 WRE，具体来讲，在 32M 训练数据的情况下，“LSTM+TRF”获得 6.33% 的 WER，比 KN5 相对下降 17.4%，比 LSTM 相对下降 5.4%，比插值模型“LSTM+KN5”依然可以获得 4.0% 的相对错误率下降。

## 3.8　本章小结

本章提出了跨维随机场（TRF）模型，对变长序列进行无向图建模，提出了有效的训练算法——增强随机近似（AugSA）算法和模型采样算法——跨维混合采样算法，提出了一系列的算法优化技术来提高算法的效率，使得可以在较大规模训练数据上的训练 TRF 语言模型，随后，将 TRF 模型应用于语言建模这一特定的任务中，验证了 TRF 模型及 AugSA 算法的有效性。

本章展现了 TRF 模型这种无向图模型相比于有向图模型的优势，解决了有向图的如下三个问题：

- 标签偏置。标签偏置是由于条件模型在每个位置处的权重是局部归一化的，因此不同历史条件下的权重无法直接进行比较，而 TRF 作为一种全局归一化模型，可以有效的避免标签偏置问题。第3.4.1节通过一个简单的例子解释了 TRF 在处理标签偏置问题的优越性，第3.6节中的仿真实验展示了在使用

同样特征的情况下，TRF 可以获得比有向图更好的性能。

- 复杂特征的使用。条件概率模型由于受限于链式假设，因此无法使用全局的特征，对复杂特征的支持也不理想，本章的实验中，TRF 使用了多种特征，特征涵盖词信息和类信息，特征阶数高达 10 阶，从而证实了 TRF 模型对复杂特征的灵活支持。
- 推理效率。条件概率模型，尤其是神经网络模型，由于局部归一化的存在，推理效率很低。相比而言，TRF 模型的由于避免了繁琐的局部归一化操作，使得推理效率大大提高，显著高于神经网络语言模型，本章的实验（表3.6）直观的验证了这一点，TRF 的模型的推理效率是 LSTM 模型的 63 倍。

# 第 4 章　神经跨维随机场语言模型

第3章介绍了跨维随机场（TRF）模型并将其用于语言建模中。TRF 的优势一方面体现在其对任意的复杂的特征的灵活支持，另一方面体现在其推理效率要比基于神经网络的 RNN 或 LSTM 语言模型高。本章提出了神经跨维随机场（Neural Trans-dimensional Random Field，neural-TRF），在 TRF 模型的基础上，使用神经网络来定义 TRF 模型的势函数。便于对比，这里将第3章介绍的使用离散特征的 TRF 称之为离散跨维随机场（记为 discrete-TRF）。neural-TRF 即可以使用神经网络的非线性建模能力，又可以避免局部归一化操作来提高推理效率，从而拥有 TRF 模型和神经网络模型两者的优势。然而，neural-TRF 的模型训练存在很大的挑战，首先，由于势函数不再是参数的线性函数，这使得模型的参数优化不再是一个凸优化问题；其次，由于神经网络定义的复杂性，模型的训练和采样效率都受到了巨大的冲击。为了能够应对这些问题，本章提出了两种训练算法——JSA+AugSA 算法和 JSA+NCE 算法，使得可以有效的训练 neural-TRF 语言模型，同时获得超越已有语言模型方法的效果。

## 4.1　引言

力求解决条件概率模型存在的诸多问题，第3章中提出了跨维随机场模型（TRF），并将其用于语言建模中。TRF 模型的主要优势体现在三点：第一，避免了条件概率中存在的标签偏置的问题，提高了模型的性能；第二，可以灵活的支持各种复杂的特征，前文的实验中使用了包括 *n*gram 特征在内的多种离散特征，使得模型的性能显著超越传统的 *n*-gram 模型，同时达到和 LSTM 相近的结果；第三，可以避免局部归一化操作，这使得模型在计算给定句子的概率时效率比神经网络模型要高。前文不仅在仿真实验中验证了模型和算法的正确性，并且在多个实际的语言建模和语音识别实验上验证了 TRF 的语言模型的性能。

然而，第3章中的工作依然存在很多的不足。首先，TRF 模型特征的设计不能体现其对复杂特征的支持。实际上，TRF 的定义中不仅仅支持线性的势函数，而且支持非线性势函数。在第3章中，将势函数定义为参数的线性函数，即 $\lambda^T f(x^j)$，并且在实验中使用了离散特征（如 *n*gram 特征及其变形形式），这大大限制了模型的性能。为了便于区分，本章将使用离散特征的 TRF 模型称为 “discrete-TRF”。这种限制可以通过与将 discrete-TRF 与目前性能最好的条件概率模型——LSTM 语言模型的比较中发现。第一，LSTM 语言模型将每个词表示成连续空间向量，这些

相似词的向量表示在连续空间具有相似性，这使得模型可以更好的进行模型平滑，而这点在离散特征中是很难体现的；第二，LSTM 使用神经网络来学习特征间的非线性约束，而离散 TRF 使用的是特征的间的线性约束；第三，LSTM 可以通过一个记忆单元来对很长的历史进行编码，这使得模型可以对很长跨度的约束进行建模，这在离散特征中是很难达到的。为了能够使用神经网络语言模型的上述优势，本章探究如何将 TRF 模型的基本理论和神经网络结合起来。

其次，TRF 模型的训练算法——AugSA 算法效率较低。AugSA 是基于采样的迭代算法，每次迭代基于 MCMC 获得满足 TRF 分布的样本来计算模型分布下的期望。为了能够在一个跨维空间内产生样本，第3章提出了跨维混合采样算法，跨维混合采样在语言模型任务中效率很低，为了能够提高采样效率，上一章通过对词进行聚类的方式来加速采样（见第3.5.4节），但是，这种加速策略存在很大的局限性：第一，聚类算法效率较低，第3章中的语言模型实验中，聚类算法往往需要几天的时间来完成；第二，这种基于类别的加速策略需要用到一个子模型来作为类别的提议分布，这要求在 TRF 模型中必须引入类别特征，这限制了 TRF 的势函数定义；第三，对于势函数为非线性函数的情况，AugSA 算法的收敛效率受到了严峻的挑战，因此本文迫切需要探究更有效的训练算法来应对更复杂的 TRF 模型。

基于上述分析，本章对 TRF 模型进行了一系列的拓展和研究，既包括模型定义上的改动，也包括训练算法的改进和新算法的提出，具体包括：

- 首先，本章提出了神经跨维随机场（neural trans-dimensional random field，neural-TRF）模型，与 discrete-TRF 的区别在于，neural-TRF 的势函数是通过一个神经网络定义的非线性函数，这使得模型可以提取连续空间特征，并对特征进行非线性建模。
- 其次，将联合随机近似（Joint Stochastic Approximation，JSA）[25] 的基本思想与 AugSA 的算法相结合，提出了 JSA+AugSA 训练算法。算法定义了一个辅助分布来作为采样过程的提议分布，每次迭代过程中，算法一方面更新模型参数和归一化系数，一方面最小化辅助分布和模型分布的 Kullback–Leibler 距离，使辅助分布逼近模型分布从从而来提高采样的性能。
- 最后，本章探究了一种新型的训练算法——噪声对比估计（Noise Contrastive Estimation，NCE）[21]。不同于 AugSA 算法直接最大化模型的似然值，NCE 通过对训练集中的训练样本和一个噪声分布下的噪声样本进行鉴别训练来训练模型参数和归一化系数，由于噪声分布与模型分布独立，因此从噪声分布中采样要比从 TRF 模型分布下采样要简单的多，同时可以有效的提高算法的收敛效率。本章对经典的 NCE 算法进行了一些改动，提出了 JSA+NCE 算

法，算法使用一个动态逼近数据分布的噪声分布来产生样本，并且对噪声分布和噪声与数据的差值分布进行鉴别，一方面提高了算法的收敛性，同时又避免了模型的过拟合。

本章在多个数据集上进行了实验，全面验证了模型和算法的有效性，数据集语言涵盖英语和普通话，从而检验了模型和算法的语言无关性。在 PTB 英文实验中的结果显示，使用 JSA+NCE 训练的 RNN-TRF 语言模型可以获得最低的识别错误率，显著超越经典的 $n$-gram 语言模型，相对错误率下降约 16%，且模型可以显著减少参数的使用，仅仅使用 4% 的参数即可获得与 LSTM 相近的结果，且推理效率是 LSTM 的 114 倍。

本章的结构组织如下。第4.2节介绍了 neural-TRF 的模型定义，列出了本章实验室中用到的三种神经网络结构；第4.3节介绍了 JSA+AugSA 算法，并通过一个语言模型实验验证了算法的有效性；第4.4节介绍了 JSA+NCE 算法，通过一个仿真实验和两个语言模型实验展示了算法准确性和有效性，相比于 AugSA 算法的可以获得显著的训练效率的提升；第4.5节对本章进行总结。本章的部分内容先后发表于 ASRU 2017（IEEE Automatic Speech Recognition and Understanding Workshop）[71] 和 ICASSP 2018（IEEE International Conference on Acoustics, Speech and Signal Processing）[72] 中。

## 4.2　神经跨维随机场

本章使用 $x^j = (x_1, \cdots, x_j)$ 表示一个长度为 $j$ 的句子，$j$ 从 1 到 m 取值。根据公式 (3-7) 中的定义，本章假设 $(j, x^j)$ 满足如下联合分布：

$$p(j, x^j; \theta, \zeta) = \pi_j p_j(x^j; \theta, \zeta) = \frac{\pi_j}{Z_1(\theta)} e^{\phi(x^j;\theta)-\zeta_j} \tag{4-1}$$

其中 $\theta$ 表示势函数的参数，$\zeta = (\zeta_1, \ldots, \zeta_m)^T$ 是归一化常数，且满足 $\zeta_1 = 0$，$\pi_j$ 是模型的先验长度分布，$Z_1(\theta)$ 表示长度为 1 的模型的归一化系数。$p_j(x^j; \theta, \zeta)$ 表示长度 $j$ 的子模型，即

$$p_j(x^j; \theta, \zeta) = \frac{1}{Z_1(\theta)} e^{\phi(x^j;\theta)-\zeta_j}$$

公式 (4-1) 中 $\phi(x^j; \theta)$ 表示模型的势函数，在离散 TRF 模型 (3-7) 中被定义为参数的线性函数，本章使用神经网络来定义模型的势函数，下面介绍三种实验中用到的神经网络结构。

### 4.2.1　CNN-TRF

使用卷积神经网络来定义势函数，如图4.1，本章称这种模型为 CNN-TRF，具体介绍每层如下：

- **词向量：**首先将句子中的每个词 $x_i \in V$ $(i = 1, \cdots, j)$ 映射成连续空间向量 $e_i \in R^{d_e}$，定义 $E \in R^{|V| \times d_e}$ 表示待估计的词向量矩阵，$|V|$ 表示词表大小，$d_e$ 表示词向量维度，由于每个词 $x_i$ 被表示为非负整数（称之为词号），其对应的词向量 $e_i$ 可以通过索引矩阵 $E$ 的对应列获得，即：

$$e_i = E[:, x_i], \quad i = 1, \cdots, j$$

其中 $E[:, x_i]$ 表示矩阵 $E$ 的第 $x_i$ 列。

- **投影层：**使用一个投影层对词向量进行降维，

$$y_i = ReLU\{W_p e_i + b_p\}, \quad i = 1, \cdots, j$$

其中 $y_i \in R^{d_p}$ 表示投影层第 $i$ 个位置的输出，维度 $d_p < d_e$，$W_p \in R^{d_p \times d_e}$ 和 $b_p \in R^{d_p}$ 是表示投影层的待估计参数，$ReLU$ 表示 Rectified Linear Units 激活函数，将输入向量或矩阵的每个分量与 0 比取最大，即：

$$ReLU\{x\} = \max(x, 0)$$

- **CNN-bank 模块：**投影层的输出被送入一个 CNN-bank 模块，模块包含多个宽度从 1 到 K 的一维卷积滤波器，用来模拟传统的 n 元文法特征，即 1gram、2gram、直到 $K$gram[44]。令 $F \in R^{d_p \times k}$ 表示一个宽度为 $k$（$k = 1, \cdots, K$）的卷积滤波器，令 $Y = [y_1, \cdots, y_j] \in R^{d_p \times j}$ 表示投影层的输出矩阵。首先需要对 $Y$ 的首尾对称补零，使其长度增加 $k-1$，结果记为 $Y' \in R^{d_p \times (j+k-1)}$，具体而言，如果 $k-1$ 为偶数，则在 $Y$ 首尾各增加 $(k-1)/2$ 个零向量，结果 $Y' = [\vec{0}, y_1, \cdots, y_j, \vec{0}]$，这里 $\vec{0} \in R^{d_p \times (k-1)/2}$ 为 0 矩阵；如果 $k-1$ 为奇数，则在 $Y$ 的首尾分别增加 $\lceil (k-1)/2 \rceil$ 和 $\lfloor (k-1)/2 \rfloor$ 个零向量，结果 $Y' = [\dot{0}, y_1, \cdots, y_j, \ddot{0}]$，这里 $\dot{0} \in R^{d_p \times \lceil (k-1)/2 \rceil}$、$\ddot{0} \in R^{d_p \times \lfloor (k-1)/2 \rfloor}$ 为 0 矩阵。这种补零方式保证了一维卷积的输入输出长度不变，常被称为 *half convolution*[①]。随后对 $Y'$ 和滤波器

① http://deeplearning.net/software/theano/library/tensor/nnet/conv.html

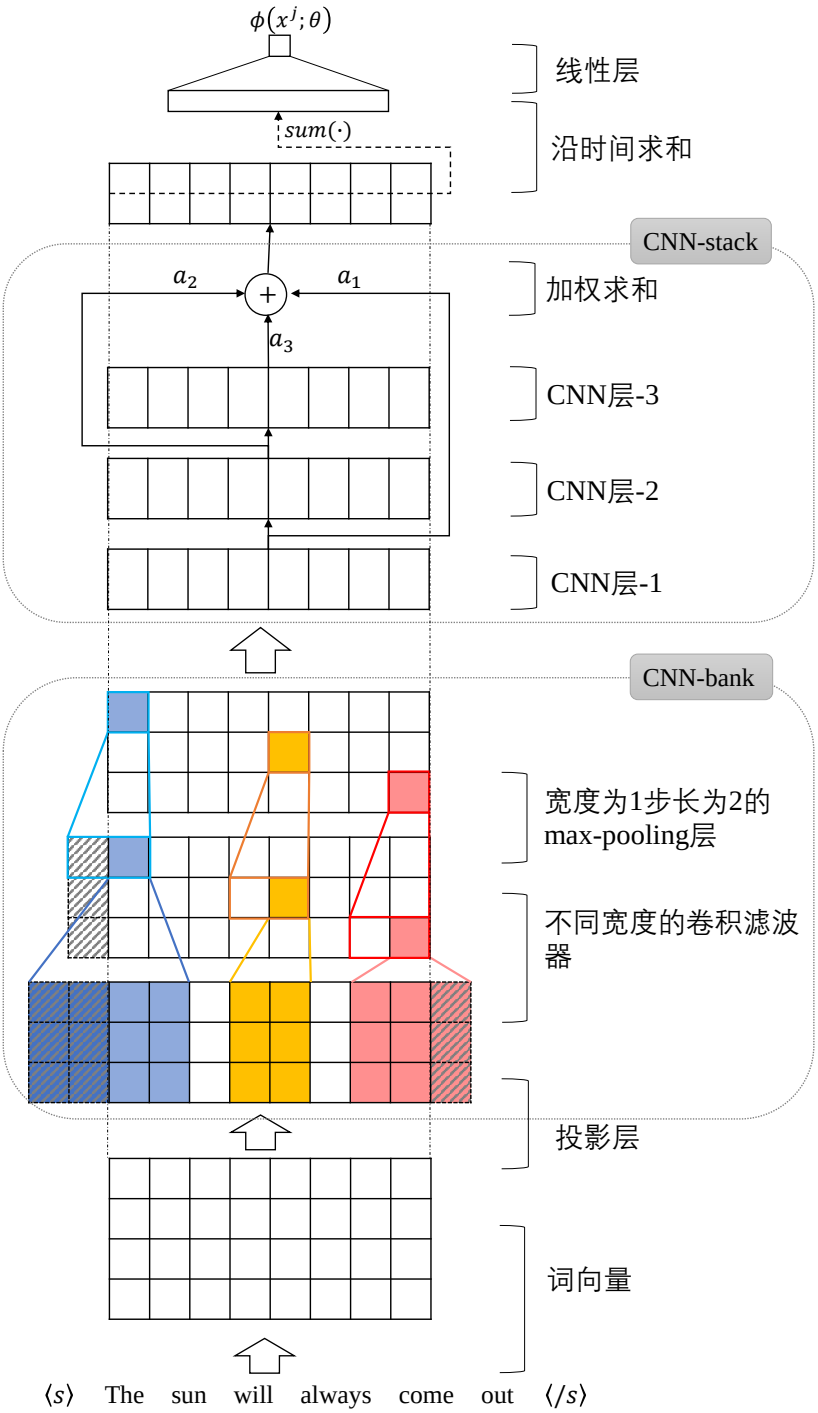



图 4.1　卷积神经网 TRF（CNN-TRF）的势函数定义。

$F$ 进行卷积操作，输出的特征向量 $f \in R^j$ 为：

$$f[i] = ReLU\{\langle Y'[:, i:i+k-1], F\rangle\}, \quad i = 1, \cdots, j$$

其中 $f[i] \in R$ 表示向量 $f$ 的第 $i$ 个元素，即滤波器 $F$ 在第 $i$ 个位置处的输出，$Y'[:, i:i+k-1]$ 表示矩阵 $Y'$ 的第 $i$ 到 $(i+k-1)$ 列，$\langle A, B\rangle$ 表示矩阵的 Forbenius 内积，即矩阵的对应元素相乘并求和：

$$\langle A, B\rangle = \sum_{i,j} A_{ij} B_{ij}$$

所有卷积滤波器的输出特征堆叠起来，后面接一个宽度为 2 步长为 1 的 max-pooling 层，max-pooling 同样使用 *half convolution* 来进行补零。假设对于每个滤波器宽度 $k$（$k = 1, \ldots, K$）对应 $w$ 个不同的卷积滤波器，则 CNN-bank 模块的最终输出为 $Y_b \in R^{wK \times j}$。

- **CNN-stack 模块：**之后链接一个 CNN-stack 模块，模块包含 $N$ 个级联的一维卷积层，每个卷积层的输出被加权求和，做法类似于文献中的 skipping 链接[73]。令 $Y_s^n \in R^{d_s \times j}$ 表示第 $n$ 个卷积层的输出，$n = 1, \cdots, N$，实验中令 $d_s < wK$ 来降低维度，并且每个卷积层都是用 *half convolution* 来保持长度，且卷积层仅仅包含一种宽度的卷积滤波器（如宽度为 3）。则 CNN-stack 模块的输出 $Y_s \in R^{d_s \times j}$，为

$$Y_s[:, i] = ReLU\left\{\sum_{n=1}^{N} a_n * Y_s^n[:, i]\right\}, i = 1, \cdots, j$$

其中 $a_n \in R^{d_s}$ 表示第 $n$ 个卷积层的权重，$*$ 表示逐元素乘。

- **沿时间求和：**最终输出矩阵延时间（长度）求和获得 neural-TRF 的势函数：

$$\phi(x^j; \theta) = \lambda^T \sum_{i=1}^{j} Y_s[:, i] + c$$

其中 $\lambda \in R^{d_s}, c \in R$ 表示参数。势函数的参数 $\theta$ 包含上述神经网络中定义的所有参数。

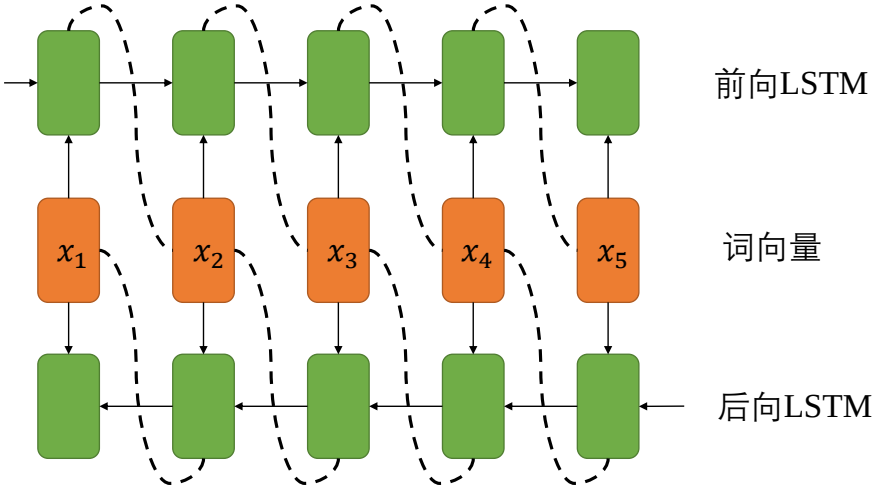



图 4.2　递归神经网 TRF（rnn-TRF）的势函数定义。

### 4.2.2　RNN-TRF

相比于卷积神经网络，递归神经网络在序列建模上更有优势，本节定义了一个基于双向 LSTM 的势函数结构，如图4.2所示，具体结构如下：

- **词向量：**和卷积神经网络一样，将句子中的每个词 $x_i \in V$ $(i = 1, \cdots, j)$ 映射成连续空间向量 $e_i \in R^{d_e}$（图4.2中橙色部分），

$$e_i = E[:, x_i], \quad i = 1, \ldots, j$$

其中 $E \in R^{|V| \times d_e}$ 表示待估计的映射矩阵，$|V|$ 表示词表大小，$E[:, x_i]$ 表示矩阵 $E$ 的第 $x_i$ 列。

- **双向 LSTM：**词向量被分别送入一个前向 LSTM 和后向 LSTM 网络（如图4.2中绿色部分），用来提取长跨度的双向的序列化特征：

$$h_f[:, i] = LSTM(h_f[:, i-1], e_i), \quad i = 1, \ldots, j$$
$$h_b[:, i] = LSTM(h_b[:, i+1], e_i), \quad i = j, \ldots, 1$$

其中 $h_f, h_b \in R^{j \times d_e}$ 分别表示前向 LSTM 和后向 LSTM 输出特征，$h_f[:, i]$ 和 $h_b[:, i]$ 分别表示其第 $i$ 列，即第 $i$ 个位置处的特征向量，这里使用 $LSTM(\cdot)$ 来指代 LSTM 的非线性操作（为了简单，此处省略了记忆向量），具体计算

公式如下：

$$
\begin{aligned}
LSTM &: h_{i-1}, e_i, c_{i-1} \rightarrow h_i, c_i \\
&\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T \begin{pmatrix} h_{i-1} \\ e_i \end{pmatrix} \\
&c_i = f * c_{i-1} + i * g \\
&h_i = o * \tanh(c_i)
\end{aligned}
$$

其中 $h_{i-1}$ 表示上一个时刻的隐层向量，在前向 LSTM 中指代 $h_f[:, i-1]$，在后向 LSTM 中指代 $h_b[:, i+1]$，$c_{i-1}$ 表示上一时刻的记忆向量，$e_i$ 表示当前时刻的词向量，$T \in R^{4d_e \times 2d_e}$ 是参数矩阵，且 $i, f, o, g \in R^{d_e}$，$*$ 表示向量逐元素乘。sigm 表示 sigmoid 函数：

$$
\text{sigm}(x) = \frac{1}{1 + e^{-x}}
$$

tanh 表示双曲正切函数：

$$
\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}
$$

- **向量预测：**为了建模由 LSTM 编码的历史信息与当前词之间的约束关系，引入如下势函数的定义，即将 LSTM 每个位置处的特征向量（隐层向量）与其所指方向的下一个词的词向量做点积（如图4.2虚线所示），然后将每个位置处的点积结果求和来获得势函数：

$$
\phi(x^j; \theta) = \sum_{i=1}^{j-1} h_f[:, i]^T e_{i+1} + \sum_{i=i+1}^{j} h_b[:, i]^T e_{i-1}
$$

这种做法与 LSTM 语言模型中共享输入输出映射矩阵的做法类似，但区别是这里并不需要计算 Softmax。势函数的参数 $\theta$ 则包含上述神经网络中的所有的参数。

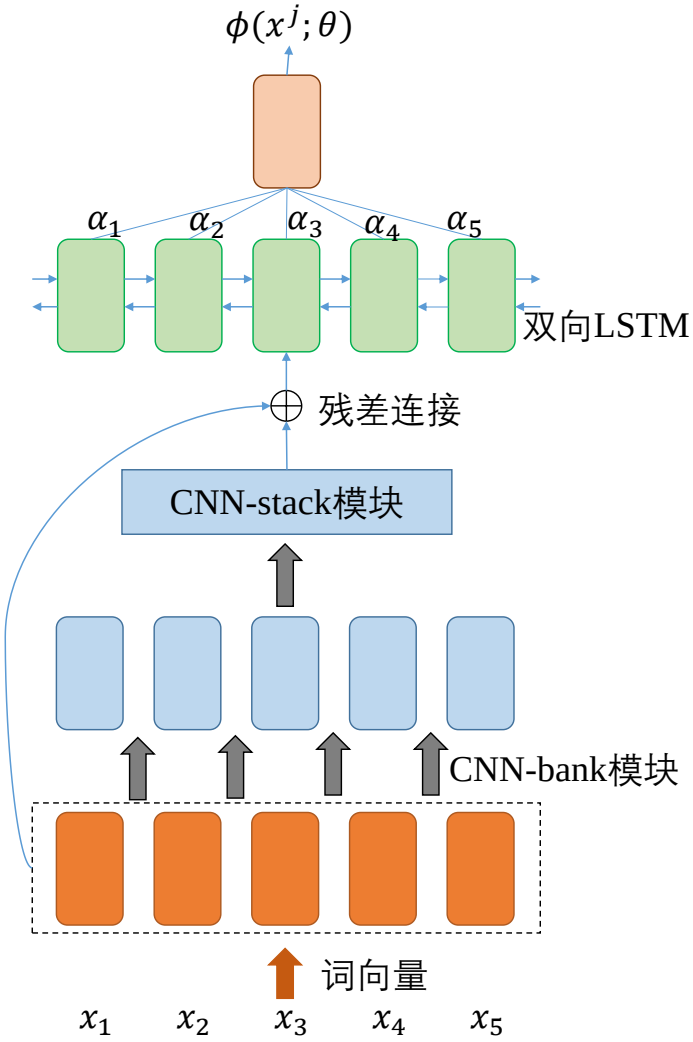



图 4.3　混合神经网 TRF（mix-TRF）的势函数定义

### 4.2.3　Mix-TRF

本节将卷积神经网络和递归神经网络结合起来定义了混合神经网络 TRF（Mix-TRF），势函数定义如图4.3，具体模块如下：

- **CNN：**网络底层是一个深层卷积神经网络，结构与 CNN-TRF（图4.1）中类似，首先将每个词映射为词向量，词向量依次被送入“CNN-bank”模块和“CNN-stack”模块，与 CNN-TRF 中不同的是，这里移除了“CNN-stack”层中的加权求和，而仅仅保留最后一个卷积层的输出。卷基层的输出与词向量求和，作为 CNN 模块的最终输出，这种连接方式称之为“残差连接”。
- **双向 LSTM：**卷积模块的输出被送入一个双向 LSTM 用来提取长跨度的双向的序列化特征，这里同样使用 $h_f, h_b \in R^{j \times d_e}$ 来表示前向 LSTM 和后向 LSTM 的输出，其计算与 RNN-TRF 相同。随后引入引入了注意机制（attention

mechanism）计算势函数：

$$\phi(x^j;\theta) = \lambda^T \sum_{i=1}^{j} \alpha_i h[:,i] + c$$
$$\alpha_i = \beta^T h[:,i], \quad i = 1,\cdots,j$$

其中 $h = [h_f; h_b] \in R^{2d\times j}$ 表示双向 LSTM 的隐层向量，$d$ 表示每个方向的 LSTM 的隐层维度，$h[:,i]$ 表示矩阵 $h$ 的第 $i$ 列，$\lambda,\beta \in R^{2d}$ 和 $c \in R$ 是待估计参数。同样，$\theta$ 包含神经网络中的所有参数。

## 4.3　JSA+AugSA 算法

本章在 AugSA 算法的基础上，引入联合随机近似（JSA）的思想，定义了一个辅助分布来作为 MCMC 采样的提议分布，同时在迭代过程中最小化辅助分布与模型分布的 KL 的距离来调整辅助分布。这样做有如下两点好处：

(1) 加速采样。使用辅助分布作为采样的提议分布进行 MH 采样，避免了在模型分布下直接进行 Gibbs 采样，加速了采样过程；

(2) 提高采样效率。由于辅助分布可以始终接近模型分布，有效提高了 MH 的采样的接受率。

### 4.3.1　算法描述

第3.3章给出了 TRF 模型训练的目标函数，即公式 (3-4) 和公式 (3-8)，对应于 neural-TRF 模型（公式 (4-1)），将目标函数重写为如下形式：

$$E_D\left[\frac{\partial\phi}{\partial\theta}\right] - \sum_{j=1}^{m}\frac{n_j}{n}E_{p_j}\left[\frac{\partial\phi}{\partial\theta}\right] = 0, \tag{4-2}$$

$$\sum_{x^j} p(j,x^j;\theta,\zeta) = \pi_j, \tag{4-3}$$

其中 $D$ 表示训练集，$n_j$ 表示训练集中长度为 $j$ 的句子数目，$n$ 表示训练集总句子数，即 $n = \sum_{j=1}^{m} n_j$，$E_D$ 表示训练集 $D$ 上的经验期望，而 $E_{p_j}$ 表示子模型分布 $p_j(x^j;\theta,\zeta)$ 下的期望，即：

$$E_D\left[\frac{\partial\phi}{\partial\theta}\right] = \frac{1}{n}\sum_{j=1}^{m}\sum_{x^j\in D_j}\frac{\partial\phi(x^j;\theta)}{\partial\theta}$$

$$E_{p_j}\left[\frac{\partial \phi}{\partial \theta}\right] = \sum_{x^j} p_j(x^j;\theta,\zeta)\frac{\partial \phi(x^j;\theta)}{\partial \theta}$$

第3.3.3节提出了 AugSA 算法来求解上述目标函数。

为了能够将 AugSA 方法应用于 neural-TRF 模型，本章借鉴了联合随机近似（Joint Stochastic Approximation，JSA）[25] 的思想，定义了一个带参的辅助分布 $q(j,x^j;\mu)$，来作为 AugSA 中的 MCMC 采样步骤中的提议分布（算法3.1中的第6步），然后使用 MH 算法来进行采样。为了能够提高 MH 算法的接受概率，需要辅助分布 $q(j,x^j;\mu)$ 尽量接近模型分布 $p(j,x^j;\theta,\zeta)$，因此算法通过优化辅助分布 $q(j,x^j;\mu)$ 和模型分布 $p(j,x^j;\theta,\zeta)$ 的 KL 距离来使得辅助分布 $q(j,x^j;\mu)$ 逼近模型分布，因此 JSA+AugSA 算法通过求解如下三个联立方程来优化模型：

$$\begin{cases} E_D\left[\dfrac{\partial \phi}{\partial \theta}\right] - \displaystyle\sum_{j=1}^{m} \frac{n_j}{n} E_{p_j}\left[\frac{\partial \phi}{\partial \theta}\right] = 0 \\ \displaystyle\sum_{x^j} p(j,x^j;\theta,\zeta) = \pi_j \\ \dfrac{\partial}{\partial \mu} KL(p(j,x^j;\theta,\zeta)||q(j,x^j;\mu)) = 0 \end{cases} \tag{4-4}$$

其中前两个方程就是 neural-TRF 的参数 $\theta$ 和归一化系数 $\zeta$ 的目标函数（公式 (4-2) 和公式 (4-3)）。在每次迭代中，辅助分布 $q(j,x^j;\mu)$ 的参数 $\mu$ 和模型参数 $\theta$、归一化系数 $\zeta$ 被一同更新，同时 $q(j,x^j;\mu)$ 被作为提议分布用到下文介绍的使用辅助分布的跨维混合采样中（见第4.3.2节）。在本章中，辅助分布 $q(j,x^j;\mu)$ 使用一个 LSTM 递归神经网络语言模型。

此外，本章对算法进行了一些额外的改进来提高算法的效率和收敛性，使其更适用于 neural-TRF。第一个改进称之为训练集 minibatch 技术，即每次迭代中从训练集中随机选择一个 minibatch 的句子（如 100 或 1000 个句子）用来计算经验期望，这是因为不同于 discrete-TRT，neural-TRF 的目标方程公式 (4-2) 中势函数 $\phi$ 对参数 $\theta$ 的导数与参数 $\theta$ 有关，因此经验期望项不再是一个常数，而是随 $\theta$ 的更新而变化。其次，本章使用 Adam[34] 优化算法来更新参数 $\theta$，节省了用于估计经验方差的计算开销。

JSA+AugSA 被整理在表4.1中，具体做法如下：在第 $t$ 次迭代，算法首先从训练集 $D$ 中随机选择 $K_D$ 个句子，组成训练数据集 $D^{(t)}$，然后使用跨维混合采样

表 4.1　JSA+AugSA 算法

**Require:** 训练集 $D$

随机初始化 $\theta^{(0)}$ and $\mu^{(0)}$

初始化归一化系数 $\zeta^{(0)} = (0, \log|\mathcal{V}|, 2\log|\mathcal{V}|, \cdots, (m-1)\log|\mathcal{V}|)^T$，其中 $|\mathcal{V}|$ 表示词表大小。

**for** $t = 1, 2, \ldots, t_{max}$ **do**

　　从训练集 $D$ 中随机选择 $K_D$ 个句子，获得 $D^{(t)}$。

　　使用跨维混合采样（第4.3.2节）产生 $K_B$ 个句子，获得 $B^{(t)}$。

　　使用公式 (4-5) 更新参数 $\theta$。

　　使用公式 (4-6) 和公式 (4-7) 更新归一化系数 $\zeta$。

　　使用公式 (4-8) 更新辅助分布的参数 $\mu$。

**end for**

（第4.3.2节）产生 $K_B$ 个样本句子，获得样本集 $B^{(t)}$。最后，参数 $\theta$ 的更新公式如下：

$$\theta^{(t)} = \theta^{(t-1)} + \gamma_{\theta,t} Adam \left\{ \frac{1}{K_D} \sum_{(j,x^j)\in D^{(t)}} \frac{\partial \phi(x^j;\theta)}{\partial \theta} - \frac{1}{K_B} \sum_{(j,x^j)\in B^{(t)}} \frac{n_j/n}{\pi_j} \frac{\partial \phi(x^j;\theta)}{\partial \theta} \right\} \tag{4-5}$$

其中，$Adam$ 表示 Adam 优化算法，$\gamma_{\theta,t}$ 表示 $\theta$ 在第 $t$ 次迭代时的学习速率。这里 $\phi$ 对参数 $\theta$ 的导数可以使用反向传播算法来有效的计算。

归一化常数 $\zeta$ 的更新公式与 discrete-TRF 类似：

$$\zeta^{(t-\frac{1}{2})} = \zeta^{(t-1)} + \gamma_{\zeta,t} \left\{ \frac{\delta_1(B^{(t)})}{\pi_1^0}, \cdots, \frac{\delta_m(B^{(t)})}{\pi_m^0} \right\}^T, \tag{4-6}$$

$$\zeta^{(t)} = \zeta^{(t-\frac{1}{2})} - \zeta_1^{(t-\frac{1}{2})}, \tag{4-7}$$

其中 $\zeta_1^{(t)}$ 表示向量 $\zeta^{(t)}$ 的第一个元素，$\gamma_{\zeta,t}$ 是 $\zeta$ 在第 $t$ 次迭代时的学习速率，$\delta_j(B^{(t)})$ 正比于长度 $j$ 在 $B^{(t)}$ 中出现的次数，即：

$$\delta_j(B^{(t)}) = \frac{1}{K_B} \sum_{(l,x^l)\in B^{(t)}} 1(j=l).$$

辅助分布的参数 $\mu$ 的更新公式如下：

$$\mu^{(t)} = \mu^{(t-1)} + \gamma_{\mu,t} \sum_{(j,x^j)\in B^{(t)}} \frac{\partial}{\partial \mu} \log q(j, x^j; \mu), \tag{4-8}$$

其中 $\gamma_{\mu,t}$ 表示 $\mu$ 在第 $t$ 次迭代时的学习速率。

### 4.3.2　使用辅助分布的跨维混合采样

本节将第3.3.4节中提出的跨维混合采样算法进行拓展。跨维混合采样包含两步——局部跳转和马氏转移，局部跳转改变序列的长度，而马氏转移则改变序列每个位置处的取值。首先，辅助分布 $q(j, x^j; \mu)$ 被引入到算法中作为采样的提议分布；其次，算法使用基于多次尝试的 Metropolis 独立采样（MTMIS）[74] 来提高接受率。这里使用 $j^{(t-1)}$ 和 $x^{(t-1)}$ 表示前一个时刻的长度和序列，算法具体实现如下：

**第一步：局部跳转**。假设 $j^{(t-1)} = k$，首先产生一个新的长度 $j \sim \Gamma(k, \cdot)$，其中跳转概率 $\Gamma(k, \cdot)$ 定义为 $k$ 邻域内的均匀分布，即：

$$\Gamma(k, j) = \begin{cases} \frac{1}{\min(k+r, m) - \max(k-r, 1) + 1}, & \text{if } |k - j| \leq r \\ 0, & \text{otherwise} \end{cases}$$

其中 $m$ 表示最大长度，$r$ 表示为跳转距离，在第3.3.4节中设为 1，本章的实验中使用 3。

如果 $j = k$，序列保持不变，即 $j^{(t)} = j^{(t-1)}$、$x^{(t)} = x^{(t-1)}$ 然后直接进入下一步。

如果 $j > k$，则先使用提议分布在原始序列尾部产生一个长度为 $j - k$ 的子序列 $u^{j-k}$，即 $u^{j-k} \sim g(\cdot|x^{(t-1)})$，然后以如下概率接受 $j^{(t)} = j$ 和 $x^{(t)} = \{x^{(t-1)}, u^{j-k}\}$：

$$\min\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, \{x^{(t-1)}, u^{j-k}\}; \theta, \zeta)}{p(k, x^{(t-1)}; \theta, \zeta) g(u^{j-k}|x^{(t-1)})}\right\}, \tag{4-9}$$

其中 $\{x^{(t-1)}, u^{j-k}\}$ 表示一个长度为 $j$ 的序列，其前 $k$ 个词为 $x^{(t-1)}$，后 $j - k$ 个词为 $u^{j-k}$。

如果 $j < k$，以如下概率接受 $j^{(t)} = j$、$x^{(t)} = x_{1:j}^{(t-1)}$：

$$\min\left\{1, \frac{\Gamma(j,k)}{\Gamma(k,j)} \frac{p(j, x_{1:j}^{(t-1)}; \theta, \zeta) g(x_{j+1:k}^{(t-1)}|x_{1:j}^{(t-1)})}{p(k, x^{(t-1)}; \theta, \zeta)}\right\}, \tag{4-10}$$

其中 $x_{i:j}^{(t-1)}$ 表示序列 $x^{(t-1)}$ 的第 $i$ 到第 $j$ 个位置的词组成的子序列。

这里的提议分布 $g(\cdot|x^h)$ 是通过辅助分布 $q(j, x^j; \mu)$ 来计算的。由于本章中 $q(j, x^j; \mu)$ 被定义为一个 LSTM 递归神经网络，从 $g(\cdot|x^h)$ 中采样和计算概率 $g(\cdot|x^h)$ 都可以通过递归实现

**第二步：马氏转移**。此步骤固定序列的长度，使用分块 Gibbs 采样来对每个位置处的词进行采样，同时使用 MTMIS 来提高采样的接受率，提议分布同样使用 $g(\cdot|x^h)$。在本章的实验中，块的大小为 $s=5$，多次尝试数目为 $M=10$.

定义 $(j,x^j)$ 表示局部跳转之后获得的序列，对于位置 $i=1,s+1,2s+1,\ldots$，马氏转移做法如下：

- 产生 $M$ 个独立同分布的样本 $u^s_{(k)}\sim g(\cdot|x^j_{1:i-1})$，$k=1,\cdots,M$，计算

$$w(u^s_{(k)})=\frac{p(j,\{x^j_{1:i-1},u^s_{(k)},x^j_{i+s:j}\};\theta,\zeta)}{g(u^s_{(k)}|x^j_{1:i-1})}$$

  和 $W=\sum_{k=1}^M w(u^s_{(k)})$，其中 $u^s_{(k)}$ 表示一个长度为 $s$ 的序列，$\{x^j_{1:i-1},u^s_{(k)},x^j_{i+s:j}\}$ 表示三个序列 $x^j_{1:i-1}$、$u^s_{(k)}$ 和 $x^j_{i+s:j}$ 的串联。
- 从集合 $\{u^s_{(1)},\cdots,u^s_{(M)}\}$ 中基于概率 $w(u^s_{(k)})/W$ 进行采样，获得 $u^s$。
- 以如下概率接受 $x^{(t+1)}=\{x^j_{1:i-1},u^s,x^j_{i+s:j}\}$：

$$\min\left\{1,\frac{W}{W-w(u^s)+w(x^j_{i:i+s-1})}\right\}$$

同样，算法中的 $g(\cdot|x^h)$ 使用辅助分布 $q(j,x^j;\mu)$ 来计算。

### 4.3.3　PTB 数据集上语言模型实验

本节中，JSA+AugSA 算法被用于训练 CNN-TRF 语言模型，并在语音识别中与现有的基线语言模型做对比。语言模型的训练数据和语音识别的测试数据与第3.7.2节中相同：语言模型的训练数据来自于 Peen Treebank 数据集中的 WSJ 部分（记为“PTB 数据集”），原始数据被预先分为 24 部分，实验选取 0-20 部分为训练集，21-22 部分为开发集，23-24 部分为测试集，最终训练集、开发集和测试集分别包含 930K、74K、82K 词。实验使用一个 10K 大小的词典，其中包含一个辅助符号“⟨UNK⟩”表示集外词。语音识别的测试数据为 WSJ'92 测试音频（包含 330 句话），首先使用开源工具包 Kaldi[①]来对测试音频进行一遍识别，识别使用一个基于 DNN 的声学模型和一个 3gram 语言模型，生成了一个 1000-best 的多候选结果，多候选的神谕错误率（oracle WER）为 0.93%。

基线语言模型包括使用 Midified Kneser-Ney 平滑算法的 5gram 语言模型[2]（记为“KN5”），和三种不同配置的 LSTM 语言模型，分别包含 2 个隐层和 200/600/1500

① http://kaldi.sourceforge.net/

表 4.2　CNN-TRF 中势函数 CNN 的配置。“cnn-$k$-$n$”表示一个 1 维 CNN，宽度为 $k$，输出维度为 $n$。“$A \rightarrow B$”表示网络 $A$ 的输出被送入到网络 $B$。

| | |
|---|---|
| 词向量维度 | 256 |
| 投影层维度 | 128 |
| CNN-bank | cnn-$k$-128, 其中 $k$ 从 1 到 10 |
| max-pooling | 宽度为 2 步长为 1 |
| CNN-stack | cnn-3-128 → cnn-3-128 → cnn-3-128 |

表 4.3　不同语言模型的实验结果。“PPL”表示 PTB 测试集上的混淆度，“WER”表示 WSJ'92 测试集上的词错误率，“参数”表示模型的参数数目，单位是百万 (M)，“+”表示对数线性插值（见第3.7.1节），插值系数为 0.5，“训练时间”表示模型的训练时间，“推理时间”表示模型计算每句话的 1000 个候选的时间。

| Model | PPL | WER(%) | 参数 | 训练时间 | 推理时间 |
|---|---|---|---|---|---|
| KN5 | 141.2 | 8.78 | 2.3 | 22s (1 CPU) | 0.06s (1 CPU) |
| LSTM-2x200 | 113.9 | 7.96 | 4.6 | 1.7h (1 GPU) | 6.36s (1 GPU) |
| LSTM-2x650 | 84.1 | 7.66 | 19.8 | 7.5h (1 GPU) | 6.36s (1 GPU) |
| LSTM-2x1500 | 78.7 | 7.36 | 66.0 | 1 天 (1 GPU) | 9.09s (1 GPU) |
| discrete TRF[24] | ≥130 | 7.92 | 6.4 | 1 天 (8 CPUs) | 0.16s (1 CPU) |
| CNN-TRF | ≥37.4 | **7.60** | 4.0 | 3 天 (1 GPU) | **0.40s (1 GPU)** |
| KN5 + LSTM-2×1500 | - | 7.47 | | | |
| CNN-TRF + LSTM-2×1500 | - | **7.17** | | | |

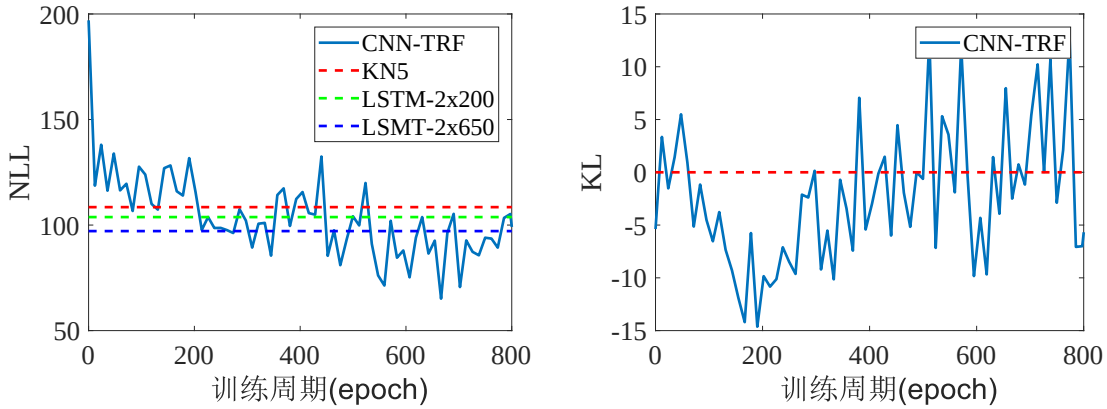
个隐层节点，LSTM 语言模型配置与文献[40] 中相同，并且使用文献提供的开源脚本进行训练。

CNN-TRF 语言模型的定义见第4.2.1节，详细的 CNN 的配置见表4.2，表4.1中的 JSA+AugSA 算法被用于训练 CNN-TRF 语言模型，具体参数如下：

- 在每次迭代中，算法从训练集中随机选择 $K_D = 1000$ 个不同长度的句子；
- 同时使用第4.3.2节中提到的跨维混合采样算法产生 $K_B = 100$ 个不同长度的句子作为样本；
- 辅助分布 $q(j, x^j; \mu)$ 定义为 LSTM 语言模型，包含 1 个隐层和 250 个隐层节点；
- (4-5)、(4-6) 和 (4-8) 中的学习速率分别设为：$\gamma_{\theta,t} = 1/(t + 10^4)$、$\gamma_{\zeta,t} = t^{-0.2}$、$\gamma_{\mu,t} = 1.0$，$t = 1, 2, \ldots$ 表示迭代次数；
- 长度分布 $\pi_j$ 选择平滑后的经验长度分布（公式 (3-35)）。

实验首先使用 word2vec 工具[①]来在 PTB 训练集上训练出词向量，然后初始化 CNN 的词向量矩阵，CNN 和 LSTM 其他参数均被随机初始化为-0.1 到 0.1 之间的

① https://code.google.com/archive/p/word2vec




(a)　(b)

图 4.4　(a) PTB 开发集上的负对数似然随训练周期的变化曲线；(b) 辅助分布 $q$ 和模型分布 $p$ 的 KL 距离随训练周期的变化曲线。

均匀分布，实验中观测模型在开发集上的对数似然值（使用宽度为 1000 的滑动平均进行平滑），直到似然值变化趋于平稳则停止迭代，最终迭代次数为 33,000（约 800 个训练周期）。开发集上的负对数似然值和辅助分布与模型分布的 KL 距离显示在图4.4中。

由于 CNN-TRF 的参数是被随机估计的，因此实验保存最后 10 个 epoch 的模型，并将其分别计算 PTB 测试集上的 PPL 和 1000 候选的语言模型得分，所得的 10 个 PPL 平均作为最终的 PPL，所得的 10 个语言模型得分平均作为最终的语言模型得分，并用其计算最终的 WER。这种做法与模型的对数线性插值类似，可以有效的降低算法随机性带来的误差。

最终的实验结果总结在表4.3中，实验结论如下。

- 首先，基于 discrete-TRF 的结果（第3.6节），AugSA 算法倾向去低估模型的混淆度，因此表4.3汇报的 CNN-TRF 的 PPL 仅仅是真实 PPL 的下界。
- 第二，CNN-TRF 获得 7.60% 的错误率，超越 discrete-TRF 中的最好结果（见表3.5），相对错误率下降 4.0%；相比于拥有同等参数规模的 “LSTM-2x200”，CNN-TRF 获得 4.5% 的相对错误率下降；相比于 “LSTM-2x650”，CNN-TRF 仅仅使用了 1/5 的参数即可获得与 LSTM 相近的结果；“LSTM-2x1500” 模型比 CNN-TRF 结果稍好，但是却使用了 CNN-TRF 模型 16.5 倍的参数。
- 第三，为了检验 CNN-TRF 与 LSTM 语言模型的互补性，实验对模型使用了对数线性插值（见第3.7.1节），插值系数固定为 0.5。结果表明 CNN-TRF 与 “LSTM-2x1500” 的插值模型可以获得最低的 WER，即 7.17%。
- 此外，CNN-TRF 模型的推理效率比 LSTM 语言模型显著提高，CNN-TRF 对

一句话的 1000 个候选重新打打分的平均时间为 0.4 秒[①]，这比 “LSTM-2x200” 和 “LSTM-2x650” 提高了 16 倍，比 “LSTM-2x1500” 提高了 23 倍。

## 4.4 JSA+NCE 算法

本节介绍了一种新型的模型训练算法——噪声对比估计（NCE），为了将 NCE 用于 neural-TRF 的训练中，本节对 NCE 算法进行了改进，使用了一个动态估计的噪声分布来产生样本，从而提出了 JSA+NCE 算法。相比于 JSA+AugSA 算法，JSA+NCE 算法有如下优势：

- 算法效率高。由于 NCE 仅仅需要从一个噪声分布中采样，相比于从 TRF 模型分布下采样，效率要高，从而有效提高训练速度；
- 算法收敛快。NCE 通过鉴别来训练模型，相比于 AugSA 算法，收敛更快。

### 4.4.1 噪声对比估计（NCE）

噪声对比估计（Noise Contrastive Estimation，NCE）[21] 是一种适用于未归一化模型估计的方法，由于 NCE 避免了模型的准确归一化，因此被成功应用于各种神经网络语言模型中来避免 Softmax 的计算[41–43,75]。NCE 的基本思想是定义一个鉴别任务，即定义一个非线性逻辑回归分类器来对数据集中的样本和噪声分布产生的样本进行分类，而模型的归一化系数可以被当做普通的参数被一同训练。

为了为了将 NCE 应用于 neural-TRF（公式 (4-1)）的参数估计，首先将公式 (4-1) 重写为如下形式，引入一个先验分布 $q(x^j)$：

$$p(j, x^j; \theta, \zeta) = \pi_j q(x^j) e^{\phi(x^j,;\theta)-\zeta_j} \tag{4-11}$$

这里使用 $\zeta_j$ 来表示对数归一化系数（而非前文中使用到的相对对数归一化系数），即 $\zeta_j = \log Z_j(\lambda)$，这样模型就可以移除全局归一化系数，即公式 (4-1) 中的 $Z_1(\theta)$。NCE 定义了如下的二值分类问题。假设对于每个训练集中的句子都对应于从噪声分布 $p_n(j, x^j)$ 中产生 $\nu$ 的个不同长度的噪声序列，则任意序列 $(j, x^j)$ 属于数据分布的概率 $P(C = 0|j, x^j; \theta, \zeta)$ 和属于噪声分布的概率 $P(C = 1|j, x^j; \theta, \zeta)$ 分别为：

$$P(C = 0|j, x^j; \theta, \zeta) = \frac{p(j, x^j; \theta, \zeta)}{p(j, x^j; \theta, \zeta) + \nu p_n(j, x^j)} \tag{4-12}$$

$$P(C = 1|j, x^j; \theta, \zeta) = 1 - P(C = 0|j, x^j; \theta, \zeta) \tag{4-13}$$

① 由于实验使用了 10 个 CNN-TRF 模型进行打分计算平均，因此 0.4 秒表示 10 个模型打分时间的总和。

给定数据集 $D$，NCE 最大化如下条件对数似然：

$$\begin{aligned} J_{\text{NCE}}(\theta,\zeta) = & \frac{1}{|D|}\sum_{(j,x^j)\in D}\log P(C=0|j,x^j;\theta,\zeta)+ \\ & \frac{\nu}{|B|}\sum_{(j,x^j)\in B}\log P(C=1|j,x^j;\theta,\zeta) \end{aligned} \tag{4-14}$$

其中 $B$ 表示噪声样本集，由噪声分布 $p_n(j,x^j)$ 产生，$|D|$ 和 $|B|$ 表示为集合 $D$ 和 $B$ 的样本数目，且满足 $\nu = |B|/|D|$。为了最大化目标函数 $J(\theta,\zeta)$，分别对 $\theta$ 和 $\zeta$ 求导可得：

$$\begin{aligned} \frac{\partial J_{\text{NCE}}(\theta,\zeta)}{\partial\theta} = & \sum_{(j,x^j)\in D} w_t(j,x^j;\theta,\zeta)\frac{\partial\phi(x^j;\theta)}{\partial\theta}+ \\ & \sum_{(j,x^j)\in B} w_n(j,x^j;\theta,\zeta)\frac{\partial\phi(x^j;\theta)}{\partial\theta} \\ \frac{\partial J_{\text{NCE}}(\theta,\zeta))}{\partial\zeta_l} = & -\sum_{(j,x^j)\in D} w_t(j,x^j;\theta,\zeta)1(l=j)- \\ & \sum_{(j,x^j)\in B} w_n(j,x^j;\theta,\zeta)1(l=j) \end{aligned} \qquad l=1,\ldots,m$$

其中

$$\begin{cases} w_t(j,x^j;\theta,\zeta) = \dfrac{1-P(C=0|j,x^j;\theta,\zeta)}{|D|} \\ w_n(j,x^j;\theta,\zeta) = -\dfrac{P(C=0|j,x^j;\theta,\zeta)}{|D|} \end{cases}$$

$1(l=j)$ 是一个二值函数，在 $l=j$ 时取 1，否则取 0；势函数 $\phi(x^l;\theta)$ 对参数 $\theta$ 的导数可以通过反向传播算法来计算。因此，任何基于梯度的算法都可以用来估计参数和归一化系数，如随机梯度下降法（Stochastic Gradient Descent，SGD）和 Adam[34]。

本文中，噪声分布被定义为如下形式：

$$p_n(j,x^j) = \pi_j p_n(x^j) \tag{4-15}$$

其中 $\pi_j, j=1,...,m$ 表示长度分布，$p_n(x^j)$ 表示一个条件概率模型，如 $n$gram 语言模型或神经网络语言模型，这样的定义使得噪声分布可以更加的贴近数据分布，从

而使得 NCE 可以在一个较小的采样数目 $\nu$（如 10）的情况下收获一个较好的效果。

### 4.4.2　JSA+NCE 算法

为了能够将 NCE 算法更好的应用于 neural-TRF 的训练，这里对算法进行一些改进，从而提出了 JSA+NCE 算法。首先，相比于 NCE 使用一个固定的噪声分布，JSA+NCE 使用一个动态的噪声分布。定义 $p_n(j,x^j;\mu)$ 为带参的噪声分布，参数为 $\mu$，则 JSA+NCE 通过最小化噪声分布与数据分布的 KL 距离来优化噪声分布，即求解如下目标方程：

$$\frac{\partial}{\partial \mu} KL(p_d(j,x^j)||p_n(j,x^j;\mu)) = 0 \tag{4-16}$$

然后，这个动态的噪声分布被用来计算分类概率 (4-12) 和 (4-13)（使用 $p_n(j,x^j;\mu)$ 替换 $p_n(j,x^j)$），用来最大化如下条件对数似然：

$$\begin{aligned} J(\theta,\zeta) = & \sum_{(j,x^j)\sim\frac{p_d+p_n}{2}} P(C=0|j,x^j;\theta,\zeta) + \\ & \nu \sum_{(j,x^j)\sim p_n} P(C=0|j,x^j;\theta,\zeta) \end{aligned} \tag{4-17}$$

相比于原始的 NCE 优化目标公式 (4-14) 对数据分布 $p_d$ 和噪声分布 $p_n$ 做鉴别，公式 (4-17) 选择对差值分布 $\frac{p_d+p_n}{2}$ 和噪声分布 $p_n$ 做鉴别，如下推论对算法的正确性给予了证明。

**推论** 6：　方程 (4-16) 的解记为 $\mu^*$，最大化公式 (4-17) 的最优点记为 $(\theta^*,\zeta^*)$，则如下等式成立：

$$p(j,x^j;\theta^*,\zeta^*) = p_n(j,x^j;\mu^*) = p_d(j,x^j)$$

**证明** 对于方程 (4-16)，显然有 $p_n(j,x^j;\mu^*) = p_d(j,x^j)$。对于方程 (4-17)，根据 NCE 的结论[21]，在 $\mu=\mu^*$ 固定的情况下，(4-17) 存在唯一的极值点，满足 $p(j,x^j;\theta^*,\zeta^*) = \frac{p_n(j,x^j;\mu^*)+p_d(j,x^j)}{2} = p_d(j,x^j)$ □

JSA+NCE 算法被整理在表4.4中，具体做法如下。在第 $t$ 次迭代中，首先从训练集 $D$ 中选择 $K_D$ 个句子，记为 $D^{(t)}$，用来更新噪声分布的的参数 $\mu$：

$$\mu^{(t)} = \mu^{(t-1)} + \gamma_{\mu,t} \sum_{(j,x^j)\in D^{(t)}} \frac{\partial}{\partial \mu} \log p_n(j,x^j;\mu), \tag{4-18}$$

表 4.4　JSA+NCE 算法

**Require:** 训练集 $D$

随机初始化 $\theta^{(0)}$ and $\mu^{(0)}$

初始化归一化系数 $\zeta^{(0)} = (\log|\mathcal{V}|, 2\log|\mathcal{V}|, \cdots, m\log|\mathcal{V}|)^T$，其中 $|\mathcal{V}|$ 表示词表大小。

**for** $t = 1, 2, \ldots, t_{max}$ **do**

　从训练集 $D$ 中随机选择 $K_D$ 个句子，获得 $D^{(t)}$

　使用公式 (4-18) 更新噪声分布的参数 $\mu$

　从噪声分布 $p_n(j, x^j; \mu)$ 中产生 $K_D$ 个句子，获得 $B_1^{(t)}$；再产生 $2\nu K_D$ 个句子，获得 $B_2^{(t)}$

　使用公式 (4-19) 更新模型参数 $\theta$

　使用公式 (4-20) 归一化常数 $\zeta$

**end for**

然后使用噪声分布 $p_n(j, x^j; \mu)$ 产生两个独立的样本集 $B_1^{(t)}$ 和 $B_2^{(t)}$，且样本集大小满足 $|B_1^{(t)}| = K_D$、$|B_2^{(t)}| = 2\nu K_D$，用来优化 $J(\theta, \zeta)$：

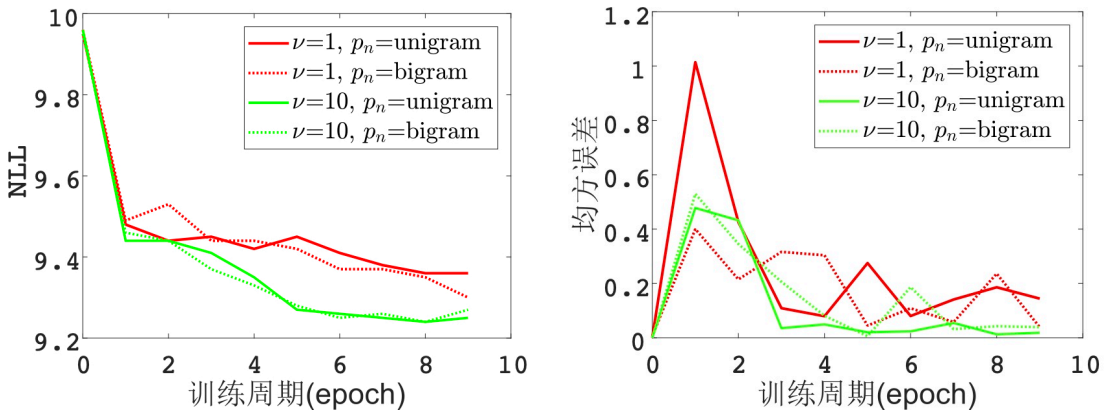
$$\theta^{(t)} = \theta^{(t-1)} + \gamma_{\theta,t} Adam\{\frac{\partial J(\theta,\zeta)}{\partial \theta}\} \tag{4-19}$$

$$\zeta_l^{(t)} = \zeta_l^{(t-1)} + \gamma_{\zeta,t} Adam\{\frac{\partial J(\theta,\zeta)}{\partial \zeta_l}\}, \quad l = 1, \ldots, m. \tag{4-20}$$

其中：

$$\begin{aligned}
\frac{\partial J(\theta,\zeta)}{\partial \theta} = & \sum_{(j,x^j)\in D^{(t)} \bigcup B_1^{(t)}} w_t(j, x^j; \theta, \zeta) \frac{\partial \phi(x^j;\theta)}{\partial \theta} + \\
& \sum_{(j,x^j)\in B_2^{(t)}} w_n(j, x^j; \theta, \zeta) \frac{\partial \phi(x^j;\theta)}{\partial \theta} \\
\frac{\partial J(\theta,\zeta))}{\partial \zeta_l} = & - \sum_{(j,x^j)\in D^{(t)} \bigcup B_1^{(t)}} w_t(j, x^j; \theta, \zeta) 1(l = j) - \\
& \sum_{(j,x^j)\in B_2^{(t)}} w_n(j, x^j; \theta, \zeta) 1(l = j) \qquad l = 1, \ldots, m
\end{aligned}$$

$$\begin{cases}
w_t(j, x^j; \theta, \zeta) = \dfrac{1 - P(C = 0|j, x^j; \theta, \zeta)}{2K_D} \\
w_n(j, x^j; \theta, \zeta) = -\dfrac{P(C = 0|j, x^j; \theta, \zeta)}{2K_D}
\end{cases}$$




(a)　(b)

图 4.5　短词建模实验结果。(a) 开发集上使用真实归一化系数计算的负对数似然（NLL）随训练周期数（epoch）的变化趋势。(b) NCE 估计的归一化系数 $\zeta$ 与真实的归一化系数 $\zeta^*$ 之间的均方误差（squared error）随训练周期（epoch）的变化趋势。实验比较了不同的采样数目 $\nu = 1, 10$ 和不同的噪声分布 $p_n(x^l) = unigram, bigram$ 对实验结果的影响。

*Adam* 表示 Adam[34] 优化算法。

JSA+NCE 的优势体现在如下几点。首先，噪声分布渐进收敛于数据分布，即保证了噪声分布可以对数据分布进行充分的近似，避免因分类问题过于简单而造成的模型估计的不充分，同时相比于使用固定的噪声分布，这种方式使得模型的收敛更加稳定；其次，通过对差值分布 $(p_n + p_d)/2$ 和噪声分布 $p_n$ 做鉴别，间接增加了数据样本的规模，从而避免了模型过拟合在经验样本上。

### 4.4.3　算法仿真：短词建模实验

本节中设计了一个仿真实验来验证 NCE 算法在 neural-TRF 模型估计中的有效性，由于实验简单，这里并没有使用第4.4.2节中提出的改进算法，而仅是使用基础的 NCE 算法（见第4.4.1节），即使用一个固定的 $n$-gram 模型作为噪声分布。这个实验称之为“短词建模”，与第3.6节中使用的词形建模实验类似，即将英文单词看做是字母序列，从而对字母序列进行建模。本次实验仅仅对长度不超过 3 的单词（短词）进行建模，因为在 $j \le 3$ 情况下，归一化系数 $Z_j(\lambda)$ 可以通过枚举的方式准确的计算出来，故而称本实验为“短词建模”。

训练集和开发集分别包含 5,143 和 69 个长度不超过 3 的英文单词，字典包含 26 个英文字母和两个辅助符号——用来标记序列开头的开始符和用来标记序列结尾的结束符，开始符和结束符分别被添加到每个序列的开头和结尾。模型 (4-11) 被用于对字母序列进行建模，模型的定义与训练算法配置如下：

表 4.5　RNN-TRF 中势函数 RNN 的配置。

| 词向量维度 | 200 |
|---|---|
| 前向 LSTM 隐层节点 | 200 |
| 前向 LSTM 隐层数目 | 1 层 |
| 后向 LSTM 隐层节点 | 200 |
| 后向 LSTM 隐层数目 | 1 层 |

- 模型 (4-11) 中的先验分布 $q(x^j)$ 设为均匀分布;
- 势函数 $\phi(x^j,;\theta)$ 使用一个双向 LSTM（BLSTM）来定义，这里的定义与第4.2.2节中提到的 RNN-TRF 有些许不同，首先每个字母被映射成 16 维的连续空间向量，然后被直接送入 BLSTM 中，每个方向的 LSTM 包含一个隐层和 16 个隐层节点，BLSTM 的输出直接求和得到势函数 $\phi(x^j,;\theta)$;
- 模型分布 (4-11) 和噪声分布 (4-15) 中的先验长度概率 $\pi_j$ 均使用经验的长度概率;
- 参数 $\theta$ 基于-0.1 到 0.1 之间的均匀分布进行初始化，归一化常数初始化为 $\zeta_l = l \times \log|V|$ 其中 $|V| = 28$ 表示词表大小;
- 实验使用 Adam 来更新参数 $\theta$ 和归一化系数 $\zeta$，训练集 mini-batch 大小为 $|D| = 10$，$\theta$ 和 $\zeta$ 的学习率分别设为 0.001 和 0.01 且训练过程保持不变。

实验探究了不同的采样数目 $\nu$ 和噪声分布 $p_n(x^l)$ 的影响，实验结果如图4.5，结论如下:

- 首先，NCE 算法可以有效的训练 neural-TRF 模型，模型 NLL 和归一化常数都可以在几个训练周期内获得很好的收敛;
- 其次，采样数目 $\nu$ 从 1 增加到 10 可以使 neural-TRF 收敛到一个更低的 NLL（如图4.5(a)），同时估计的归一化系数 $\zeta$ 更加接近真实的值 $\zeta^*$（如图4.5(b)）;
- 最后，对于一个较小的采样次数，即 $\nu = 1$，选取一个更接近数据分布的噪声分布 $p_n(x^j)$（从 unigram 修改为 bigram）可以显著提高 NCE 的收敛性，而如果采样次数足够大，即 $\nu = 10$，则改变噪声分布对 NCE 的收敛性几乎没有影响。这个结论与已有文献中的结论一致[21]。

### 4.4.4　PTB 数据集上语言模型实验

本节使用 JSA+NCE 来训练 RNN-TRF 语言模型实验（模型定义见第4.2.2节），训练数据和测试数据以及基线语言模型与第4.3.3节相同，相比于 CNN-TRF，将势函数表示成 RNN 可以有效的使用双向的序列化特征，相比于 LSTM 语言模型，使用 TRF 的框架可以避免繁琐的局部归一化，从而有效的提高模型的推理效率。

表 4.6　不同语言模型的实验结果①。“PPL”表示 PTB 测试集上的混淆度，“WER”表示 WSJ’92 测试集上的词错误率，“参数”表示模型的参数数目，单位是百万 (M)，“+”表示对数线性插值（见第3.7.1节），插值系数为 0.5，“训练时间”表示模型的训练时间，“推理时间”表示模型计算每句话的 1000 个候选的时间。

| Model | PPL | WER(%) | 参数 | 训练时间 | 推理时间 |
|---|---|---|---|---|---|
| KN5 | 141.2 | 8.78 | 2.3 | 22s (1 CPU) | 0.06s (1 CPU) |
| LSTM-2x200 | 113.9 | 7.96 | 4.6 | 1.7h (1 GPU) | 6.36s (1 GPU) |
| LSTM-2x650 | 84.1 | 7.66 | 19.8 | 7.5h (1 GPU) | 6.36s (1 GPU) |
| LSTM-2x1500 | 78.7 | 7.36 | 66.0 | 1 天 (1 GPU) | 9.09s (1 GPU) |
| discrete TRF[24] | ≥130 | 7.92 | 6.4 | 1 天 (8 CPUs) | 0.16s (1 CPU) |
| CNN-TRF (JSA+AugSA) | ≥37.4 | 7.60 | 4.0 | 3 天 (1 GPU) | 0.40s (1 GPU) |
| RNN-TRF (JSA+NCE) | ~66 | **7.40** | **2.6** | 1 天 (1 GPU) | **0.08s(1 GPU)** |

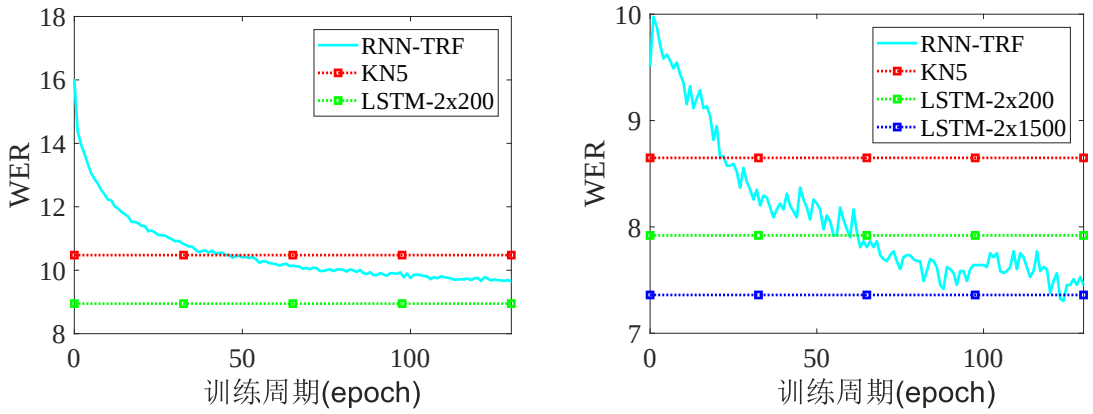
RNN-TRF 中 RNN 势函数的详细参数见表表4.5，第4.4.2节中介绍的 JSA+NCE 算法被用于训练 RNN-TRF，参数配置如下：

- 模型 (4-11) 中的先验分布 $q(x^j)$ 设为均匀分布；
- 所有神经网络的参数均被初始化为-0.1 到 0.1 之间的均匀分布；
- 噪声分布 $p_n(j, x^j; \mu)$ 定义为 LSTM，包含 1 个隐层，200 个隐层节点；
- 每次迭代，算法从训练集随机选择 $K_D = 100$ 个句子，用来更新噪声分布 $p_n(j, x^j; \mu)$，同时从噪声分布 $p_n$ 中产生样本数目为 $K_D = 100$ 和 $2\nu K_D = 200$ 的两个样本集（即 $\nu = 1$），用来更新模型参数 $\theta$ 和 $\zeta$；
- 学习速率设为 $\gamma_{\mu,t} = 0.1$、$\gamma_{\theta,t} = 10^{-3}$ 和 $\gamma_{\zeta,t} = 0.01$ 且在训练过程中保持不变；
- 实验使用 Adam 算法来更新参数 $\theta$ 和 $\zeta$，而使用普通的 SGD 来更新参数 $\mu$。

为了确定算法的终止条件，实验对 PTB 开发集中的每句话随机插入、删除或替换一些词，构造了一个虚假的多候选开发集，然后对这个虚假的开发集使用语言模型进行打分，选择语言模型得分最低（即似然最高）的那句话作为这个数据集的“识别”结果。迭代过程中观测 RNN-TRF 在这个虚假多候选数据集中的错误率，如果错误率没有明显变化则停止迭代。WSJ’92 测试集上的 WER 和虚假多候选数据集上的 WER 随迭代次数的收敛见图4.6，实验结果展示在表4.6中，实验结论如下。

- 首先，RNN-TRF 获得 7.40% 的 WER，不仅比使用 JSA+AugSA 训练的 CNN-TRF 错误率更低，而且训练的时间减小为 1/3（CNN-TRF 使用了 3 天，而 RNN-TRF 使用了 1 天）；
- 其次，相比于经典的 $n$-gram 语言模型，RNN-TRF 获得约 16% 的相对错误率

① 表4.6和表3.5中结果的训练集都是 PTB 的训练集，WER 都是在 WSJ’92 测试集上计算的。而计算 PPL 的数据不同，表4.6是在 PTB 测试机上计算的 PPL，而表3.5是在 WSJ’92 测试集上计算的 PPL 和 NLL。




(a)　(b)

图 4.6　(a) 虚假多候选数据集上的 WER 和 (b)WSJ 测试集上的 WER 随训练周期的变化曲线。

下降，相比于神经网络语言模型，RNN-TRF 仅仅使用了 4% 的参数就获得了和“LSTM-2x1500”相近的性能；

- 最后，由于 RNN-TRF 避免了局部归一化操作，这使其推理效率显著提高，实验中 RNN-TRF 计算每个句子的 1000 个候选的平均时间为 0.08 秒，效率比“LSTM-2x200”和“LSTM-2x650”提高了约 80 倍，比“LSTM-2x1500”提高了约 114 倍。

### 4.4.5　HKUST 中文语音识别实验

本节在一个中文数据集上进行语音识别实验，进一步验证 neural-TRF 语言模型和 JSA+NCE 算法的性能及其语言无关性。实验数据为 HKUST 普通话电话录音数据[①]，由于数据语言非常的口语化，且数据没有分词标注，因此实验构建了基于汉字的语言模型。语言模型的训练语料包含大约 2.4M 字，字典大约包含 4000 个汉字，不同的语言模型对预先生成的 100-best 多候选识别结果进行重新打分，来计算字错误率（CER），多候选结果是使用 Kaldi 脚本基于 LF-MMI 的声学模型（又名‘chain’ models）生成的。实验对多候选识别结果进行切分，选取其中 1082 句（占 20%）当做开发集，剩余的 4331 句话当做测试集，所有的超参数均在开发集上调到最优。

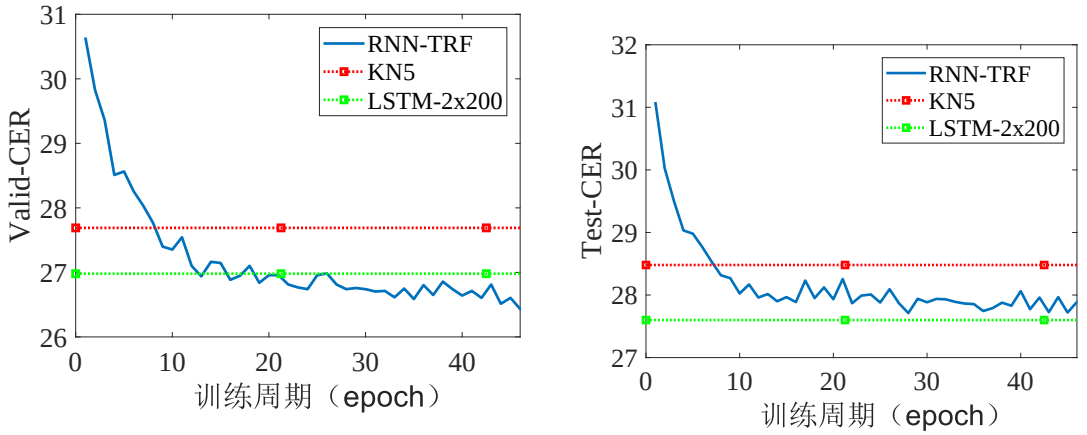
实验使用 JSA+NCE 算法训练了一个 RNN-TRF 语言模型，具体细节如下：

- 模型 (4-11) 中的先验分布 $q(x^j)$ 设为均匀分布；
- 势函数定义见第4.2.2节，词向量的维度为 200，每个方向的 LSTM 均包含 1

① https://github.com/kaldi-asr/kaldi/tree/master/egs/hkust

表 4.7　HKUST 中文语音识别实验。“Valid” 表示开发集上的字错误率（CER），“Test” 表示测试集上的字错误率（CER），“参数” 表示模型的参数数目，单位是百万，“+” 表示对数线性插值（见第3.7.1节），使用相等的插值系数。“训练时间” 表示模型训练所需要的总时间，“推理时间” 表示模型对每句话的 100 个多候选重新打分的平均时间。

| Model | Valid(%) | Test(%) | 参数 | 训练时间 | 推理时间 |
|---|---|---|---|---|---|
| KN5 | 27.69 | 28.48 | 3.5 | 8s (1 CPU) | 0.004s (1 CPU) |
| LSTM-2x200 | 26.98 | 27.60 | 2.2 | 0.5h (1 GPU) | 0.048s (1 GPU) |
| RNN-TRF | 26.32 | **27.72** | **1.4** | 1 天 (1 GPU) | **0.009s (1 GPU)** |
| LSTM + KN5 | 26.53 | 27.36 | | | |
| KN5 + RNN-TRF | 26.32 | 27.30 | | | |
| LSTM + RNN-TRF | 25.89 | **26.91** | | | |
| LSTM + KN5 + RNN-TRF | 25.96 | **26.87** | | | |




(a)　(b)

图 4.7　(a) 开发集上的 CER 和 (b) 测试集上的 CER 随训练周期的变化曲线。

个隐层以及 200 个隐层节点；

- 所有神经网络的参数均被初始化为-0.1 到 0.1 之间的均匀分布；
- 噪声分布 $p_n(j, x^j; \mu)$ 定义为 LSTM，包含 1 个隐层，200 个隐层节点；
- 每次迭代，算法从训练集随机选择 $K_D = 100$ 个句子，用来更新噪声分布 $p_n(j, x^j; \mu)$，同时从噪声分布 $p_n$ 中产生样本数目为 $K_D = 100$ 和 $2\nu K_D = 800$ 的两个样本集（即 $\nu = 4$），用来更新模型参数 $\theta$ 和 $\zeta$；
- 学习速率设为 $\gamma_{\mu,t} = 0.1$、$\gamma_{\theta,t} = 10^{-3}$ 和 $\gamma_{\zeta,t} = 0.01$ 且固定不变；
- 实验使用 Adam 算法来更新参数 $\theta$ 和 $\zeta$，而使用普通的 SGD 来更新参数 $\mu$。

实验监视 RNN-TRF 在开发集上的 CER，经过 46 个训练周期模型在开发集上的 CER 达到最小值，从而停止迭代（如图4.7），训练总耗时约为 1 天。为了

使 RNN-TRF 的结果更加的稳定，实验选取了最后三个训练周期的 RNN-TRF 模型分别对多候选进行打分，然后将获得的三份语言模型得分进行平均当做最终的语言模型得分，用来计算 CER，因此表中列出的 RNN-TRF 模型的推理时间其实是三个 RNN-TRF 重新打分的总时间。实验对比了 RNN-TRF 与 5gram 语言模型（记为“KN5”）和 LSTM 语言模型（包含 2 个隐层，每层包含 200 个节点，记为“LSTM-2x200”），实验结果显示的在表4.7中，结论总结如下：

- RNN-TRF 显著超越 KN5，相对 CER 下降约 2.7%；
- RNN-TRF 与 LSTM 语言模型获得相近的 CER，但是 RNN-TRF 的参数仅仅是 LSTM 的 63%；
- 在此实验中，由于使用的是基于字的语言模型，且模型仅仅包含约 4000 个汉字，因此对于 LSTM 而言 Softmax 的计算代价不高，尽管如此，RNN-TRF 的推理效率依然比 LSTM 提高了约 5 倍；
- RNN-TRF 展现出与 LSTM 语言模型良好的互补性，LSTM 与 RNN-TRF 的插值模型可以获得最低的错误率，超越 LSTM 与 KN5 的插值模型，相对 CER 下降约 1.6%。

本实验的结论与 PTB 数据集上实验结论一致，从而验证了 LSTM-TRF 语言模型和 JSA+NCE 训练算法的语言无关性。

### 4.4.6　CHiME-4 语音识别实验

本节在一个中等规模的数据集——CHiME-4 数据集上验证 neural-TRF 和 NCE 算法的有效性。语言模型的训练语料包含 37M 词，是 PTB 数据集的 40 倍，词表大小为 5K，包含一个辅助符号“⟨UNK⟩”表示集外词。不同语言模型通过对预先生成的 100 个多候选进行重新打分，来计算 WER，多候选结果是使用一个基于 CHiME-4 官方规则的多通道语音识别系统（更多细节见文献[76]）。所有的超参数，如学习速率、训练周期、语言模型得分系数都是在开发集上调到最优，在 CHiME-4 任务中，不同的识别系统是通过比较 real test data（真实测试数据，即真实噪声环境下录取的测试音频）上的效果来进行评估度的。

本次实验中定义了一个 Mix-TRF，其实势函数的结构见第4.2.3节，具体细节见表4.8，训练过程如下：

- 首先，实验使用 SGD 算法训练了一个包含 2 个隐层 512 个隐层节点的 LSTM 语言模型，作为先验分布 $q(x^j)$（见公式 (4-11)），
- 随后，第4.4.1节中介绍的 NCE 算法被用来训练 Mix-TRF 模型（这里并没有使用第4.4.2节中提出的 JSA+NCE 算法），噪声分布使用一个固定的 bigram

表 4.8　Mix-TRF 配置参数。“cnn-$k$-$n$”表示一个 1 维 CNN，宽度为 $k$，输出维度为 $n$。“$A \to B$”表示网络 $A$ 的输出被送入到网络 $B$。

| | | |
|---|---|---|
| $\pi_j$ | 先验长度分布 | |
| $q(x^j)$ | LSTM 语言模型，包含 2 个隐层，每层包含 512 个节点[40] | |
| $\phi(x^j;\theta)$ | 词向量维度 | 200 |
| | Conv1D bank | cnn-$k$-128-ReLU, 其中 $k$ 从 1 到 10 |
| | Conv1D layers | cnn-3-128-ReLU → cnn-3-128-ReLU → cnn-3-128-ReLU |
| | BLSTM | 1 个隐层，每层 128 个节点 |

表 4.9　CHiME-4 数据集上的语音识别 WER。“Dev”表示开发集上，“Test”表示测试集。“real”表示真实环境噪声情况下的音频，“simu”表示仿真噪声情况下的音频。“+”表示使用相等权重的对数线性插值。

| model | Dev | | Test | |
|---|---|---|---|---|
| | real | simu | real | simu |
| KN5 | 5.03 | 4.79 | 7.38 | 5.78 |
| LSTM2x512 | 3.63 | 3.24 | 5.70 | 4.53 |
| Mix-TRF | 3.53 | 3.20 | 5.68 | 4.36 |
| KN5 + LSTM2x512 | 3.56 | 3.29 | **5.71** | 4.18 |
| KN5 + Mix-TRF | 3.53 | 3.22 | 5.54 | 4.20 |
| KN5 + LSTM2x512 + Mix-TRF | 3.42 | 3.10 | **5.44** | 4.13 |

语言模型；

- NCE 的采样系数 $\nu = 20$，训练集 mini-batch 大小 $|D| = 100$；
- 训练过程中使用 SGD 算法来更新参数 $\theta$ 和归一化系数 $\zeta$，而先验分布 $q(x^j)$ 固定不变，$\theta$ 和 $\zeta$ 的学习速率均初始化为 0.01 且每个训练周期将其减半；
- 实验中参数 $\theta$ 被初始化为-0.1 到 0.1 之间的均匀分布，而归一化系数 $\zeta$ 被初始化为 $\zeta_j = j$ for $j = 1, \cdots, m$；

两个训练周期后 Mix-TRF 在开发集上的错误率达到最低，从而停止迭代。总共训练时间为 1 天。

不同语言模型的识别结果见表4.9，包括一个使用 Modified Kneser-Ney 平滑的 5gram 语言模型（记为“KN5”）和模型 $q(x^j)$（记为“LSTM2x512”）。首先 LSTM 语言模型显著超越 KN5 语言模型，real test data 上的相对错误率下降高达 22.8%，LSTM 与 KN5 的插值模型“KN5+LSTM2x512”可以获得目前最好的有向图结果。在 LSTM 语言模型的基础上，仅仅经过 2 个训练周期的训练，Mix-TRF 可以进一步降低 WER。同时，将 KN5、LSTM 和 Mix-TRF 组合起来可以获得最低的 WER 5.44%，比“KN5+LSTM2x512”相对下降 4.7%，这个结果展示了 Mix-TRF 模型和有向图模型的互补性。

## 4.5　本章小结

本章的主要内容总结如下：

- 将跨维随机场（TRF）的基本理论与神经网络相结合，提出了神经跨维随机场（neural-TRF）模型。neural-TRF 兼具 TRF 模型与神经网络的优点，一方面作为一种无向图模型，可以有效避免有向图模型中存在的标签偏置问题，同时因避免局部归一化可以保证模型拥有很高的推理效率；另一方面由于嵌入了神经网络特征，又可以发挥神经网络连续空间建模以及 LSTM 长跨度建模的优势。
- 将联合随机近似（JSA）的思想引入到增强随机近似（AugSA）训练算法中，提出了 JSA+AugSA 训练算法。算法定义了一个辅助分布来加速采样，提高了采样效率，成功将 AugSA 训练算法应用于 neural-TRF 语言模型中。实验中使用 JSA+AugSA 训练的 CNN-TRF 语言模型超越了使用离散特征的 TRF，进一步降低了识别错误率。
- 研究了噪声对比估计（NCE）算法在 neural-TRF 训练中的性能。NCE 通过对训练数据与噪声数据进行鉴别来训练模型，可以同时估计参数和归一化系数，是一种适用于未归一化模型训练的算法。本章对 NCE 算法进行了一些改进，提出了 JSA+NCE 算法，首先使用一个动态调整的噪声分布来产生样本，其次对噪声分布和噪声与数据的插值分布做鉴别，这些改动提高了算法的收敛性和稳定性，避免了模型过拟合在经验样本上。
- 相比于使用离散特征的 TRF，neural-TRF 可以进一步提高模型性能，降低语音识别错误率，同时因为连续特征的引入，neural-TRF 可以显著减小参数数目。PTB 英文数据集上的实验表明，相比于 LSTM 语言模型，neural-TRF 仅仅使用 4% 的参数既可以获得与 LSTM 语言模型相近的结果，且 neural-TRF 推理效率比 LSTM 语言模型提升约 114 倍（见第4.4.4节）。
- 本章验证了 neural-TRF 模型和训练算法的语言无关性。在 HKUST 数据集上，neural-TRF 依然可以超越经典的 $n$-gram 语言模型，获得与 LSTM 语言模型相近的性能。同时实验表明，即使是在 Softmax 的计算代价不大的字模型中，相比于 LSTM 语言模型，neural-TRF 推理效率依然可以获得 5 倍的提升（见第4.4.5节）。

本章的研究进一步验证了 TRF 模型对复杂特征乃至句子级别特征的灵活支持，双向 LSTM 由于模型定义问题无法被用于条件概率模型中，但却可以很容易被嵌入到 TRF 的模型框架中。本章的研究对于推动无向图理论与神经网络的结合有着重要的意义，一方面神经网络的连续空间特征和非线性建模能力已经得到越

来越多的认可，另一方面有向图建模方法在序列建模问题中依然存在着先天的局限性，而 neural-TRF 为突破有向图的局限，同时兼容神经网络的优点，提供了一个很好的思路。

# 第 5 章　总结和展望

## 5.1　总结

为了突破条件概率模型的限制，本文提出了跨维随机场（TRF）模型，首次将随机场的基本方法从定维情形拓展到跨维情形，并成功应用于语言模型这一复杂的产生式建模任务。TRF 模型相比于条件概率模型有如下优势：

- TRF 避免了标签偏置问题；
- TRF 可以灵活的支持各种局部特征和句子级别的特征，支持离散特征和神经网络特征；
- TRF 因避免了局部归一化，使得模型推理效率较神经网络模型有很大提高。

为了有效的训练 TRF 语言模型，本文提出了一系列行之有效的训练方法，算法的优劣性总结如下：

- **增强随机近似（AugSA）算法**。AugSA 依托于随机近似的基本理论，通过基于 MCMC 采样产生满足模型分布的样本来计算模型分布下的期望，用来更新参数和归一化系数。AugSA 方法的优点在于算法的思路简单，稳定性强，随机近似的理论保证了模型可以收敛到最大似然解，并且算法可以很方便的进行并行。算法的缺点在于采样效率慢，由于语言模型的词表巨大，如果不进行任何改动，对自然语言句子直接进行 Gibbs 采样几乎是不可行的，因此文中通过对词表进行聚类来加速采样，但这种方法需要在模型中引入类别特征，对模型的定义提出了要求。
- **JSA+AugSA 算法**。JSA+AugSA 算法就是为了解决 AugSA 算法采样效率慢的问题。算法定义了一个辅助的分布来作为采样的提议分布，同时优化辅助分布和模型分布的 KL 距离来确保辅助分布接近模型分布，从而提高了采样算法接收率。JSA+AugSA 算法的优点是采样效率比 AugSA 算法有了显著的提升，并且对 TRF 模型势函数的定义没有约束，因此文中将势函数定义为一个深层的 CNN。JSA+AugSA 算法的缺点是收敛较慢，这主要有两方面的原因，一方面对于 neural-TRF 而言，由于势函数是参数的非线性函数，模型优化不再是一个凸优化；另一方面，算法的采样是基于 MH 采样，由于辅助分布很难足够接近模型分布，因此 MH 采样的接受率往往不高（大约 20% 左右），这使得样本质量不高。
- **噪声对比估计（NCE）算法**。算法通过对数据样本和噪声样本做鉴别来训练模型，其优点在于可以同时估计模型参数和归一化系数，并且由于噪声样本

是从一个固定的噪声分布中产生的，因此采样效率和训练效率比 AugSA 和 JSA+AugSA 高。NCE 方法的缺点包括两点，首先由于算法选择了一个简单的噪声分布（如 bigram），为了能够获得较好的训练效果，噪声样本的数目往往是训练集的 10 倍甚至 100 倍；其次，由于训练数据有限，模型容易过拟合在训练集上。

- **JSA+NCE 算法**。算法在 NCE 的基础上进行了一些改进，首先使用了一个动态调整的噪声分布来产生样本，由于噪声分布渐进收敛到数据分布，因此每次迭代不需要很多的噪声样本就能可以获得很好的训练效果；其次对噪声分布和噪声与数据分布的插值分布做鉴别，避免了因数据样本的不充分性而造成的过拟合。JSA+NCE 算法很好的解决了 NCE 算法存在的两个问题，是本文中表现最好的训练算法，但 JSA+NCE 算法依然存在不足，这是因为算法需要从噪声分布产生样本，而对于离散序列，尤其是自然语言的词序列，即使使用条件概率模型，产生离散的词序列依然效率很低，这是 JSA+NCE 算法的计算瓶颈所在。

本文成功将 TRF 模型应用于语言建模这一变长序列建模任务，通过模型混淆度和语音识别错误率两个指标衡量了 TRF 语言模型的性能，结论总结如下：

- TRF 语言模型可以灵活的支持各种丰富的特征，包括复杂的离散特征和各种神经网络特征，且使用神经网络特征的 TRF 可以获得更好的模型性能；
- 使用离散特征的 TRF 语言模型（discrete-TRF）可以显著超越经典的 $n$-gram 语言模型，相对错误率下降约 10%，并且在 PTB 训练集上达到和 LSTM 语言模型相近的性能；
- 使用神经网络特征的 TRF 语言模型（neural-TRF）相比于 discrete-TRF，模型性能进一步提高，在 PTB 实验中，RNN-TRF 的识别错误率比 $n$-gram 语言模型相对下 16%，相比于神经网络语言模型，RNN-TRF 仅仅使用 4% 的参数即可获得与 LSTM 语言模型相近的性能；
- discrete-TRF 语言模型和 neural-TRF 语言模型的推理效率都显著超越 LSTM 语言模型，本文的实验中，discrete-TRF 的推理效率是 LSTM 的 63 倍，neural-TRF 的推理效率是 LSTM 的 114 倍；
- 本文在多个英文数据集上和 HKUST 普通话数据集上分别进行了实验，验证了 neural-TRF 和 JSA+NCE 训练算法的语言无关性。
- TRF 语言模型与条件概率模型有良好的互补性，TRF 与 LSTM 语言模型的对数线性插值可以获得最好的识别效果。

值得一提的是，TRF 与条件随机场（CRF）同属于随机场模型，但是两者又存

在明显的不同。首先，CRF 建模的是给定观测序列情况下的标注序列的条件概率，是鉴别模型，而 TRF 直接建模观测序列的联合概率，是产生式模型；其次，CRF 的状态空间比 TRF 小的多，这是因为标注的取值（如词性标注）往往要比观测的取值（如自然语言的词）要少，这使得 CRF 在使用低阶特征的情况下，归一化常数和模型分布下的期望可以通过动态规划算法来精确计算，而 TRF 的归一化系数和模型分布下的期望是无法精确计算，因此计算复杂度上的差距是 TRF 与 CRF 的本质区别。假如 CRF 的状态空间也变的很大，或者特征阶数和复杂度也变的很高，这样 CRF 也会面临与 TRF 类似的问题，这时文中提到的训练算法也会为复杂 CRF 的训练提供借鉴。

## 5.2　展望

本文首次将随机场方法应用于大状态空间的变长序列的产生式建模，成功的将随机场方法从定维情形（如图像）拓展到跨维情形（即变长序列），推动了随机场方法在序列建模中的发展，为序列建模在条件式模型之外提供了一条崭新的思路。本文在多个数据集上进行进行了一系列的实验完整的验证了 TRF 语言模型在语音识别中的性能，以及通过英文语音识别和普通话语音识别实验验证了模型和算法的语言无关性。

为了能够将 TRF 模型更好的应用于实际系统中，**未来的工作应该更关注 TRF 模型在超大规模训练数据上的性能**。为了实现这一点，文中提出的模型训练算法可以通过一些工程化处理来应对更大规模的数据集的情况。具体而言，对于 JSA+NCE 训练算法，算法首先从噪声分布中产生样本，然后使用反向传播算法来更新参数，因此实际中可以使用多机多核并行的方式来并行产生更多的样本，由于算法中往往样本数据越多算法收敛越好，因此产生更多的样本同时可以提高算法的收敛性，减少模型迭代周期数。此外，基于反向传播的梯度更新同样可以使用并行的方式来计算，这就可以大大减少模型的训练时间，从而实现在超大规模数据集上训练 TRF 模型的目的。

TRF 的理论方法不仅仅可以用于语言模型任务中，**未来 TRF 的工作着重于将 TRF 方法应用于带标注的序列建模任务中**，TRF 处理带标注序列有如下优势：

- TRF 可以对标注序列和观测序列进行联合产生式建模，获得观测序列和标注序列的联合概率，这使得模型不仅仅可以用于标注任务，还可以用作语言模型；
- 就标注任务而言，相比于 Seq2seq 方法，TRF 作为无向图模型可以从理论上避免标签偏置问题；相比于 CRF 方法，TRF 不仅仅可以使用有标注数据进

行有监督训练，还可以使用无标注数据进行半监督训练；

- 就语言模型任务而言，引入标注信息可以使用更有意义的语言学特征，并且由于 TRF 可以在无标注数据上进行半监督训练，解决了标注数据不足的问题。

**TRF 的研究不仅仅局限于序列建模任务中，也可以用于对非序列化的观测进行建模**，如自然语言处理中的句法树，这也是未来 TRF 研究的一个重要方向。但是对定义在非序列化观测上的 TRF 模型的进行训练和推理还存在很多亟待解决的问题，首先，TRF 模型的训练核心在于采样，是否能高效的获得满足模型分布的样本直接影响了训练效率和模型性能，因此如何对非序列化的观测进行采样是模型训练的主要问题；其次，虽然使用 TRF 直接计算联合概率十分的快捷，但是很多任务需要计算边缘概率或条件概率，如给定词序列最优的句法树，或者计算词序列的边缘概率，这种情况下模型的推理变得十分的困难，往往需要借助蒙特卡洛方法来进行近似推理。

随机场方法有很大的潜力，本文的工作仅仅是冰山一角，虽然目前基于神经网络的方法取得了巨大的成功，但是需要看到，神经网络的优势在于连续空间特征，这使得有向图模型对转移概率的估计更加的平滑，但却无法突破有向图模型的先天局限性。因此如何将无向图的模型框架与神经网络连续空间特征相结合，已然成为了主流的研究方向，典型的案例是 CRF 与神经网络的结合而构建一个鉴别模型，在这种背景下，TRF 的提出便具有重要的意义，一方面 TRF 是定义在变长序列上的无向图模型，另一方面 TRF 可以很好的兼容神经网络特征，虽然面临很多的困难，但 TRF 提出为变长序列的无向图建模打开了一扇崭新的大门。

# 参考文献


[1] Chater N, Manning C D. Probabilistic models of language processing and acquisition[J]. Trends in cognitive sciences, 2006, 10(7): 335–344.

[2] Chen S F, Goodman J. An empirical study of smoothing techniques for language modeling[J]. Computer Speech & Language, 1999, 13: 359–394.

[3] Berger A L, Pietra V J D, Pietra S A D. A maximum entropy approach to natural language processing[J]. Computational Linguistics, 1996, 22: 39–71.

[4] Schwenk H. Continuous space language models[J]. Computer Speech & Language, 2007, 21: 492–518.

[5] Mikolov T, Kombrink S, Burget L, et al. Extensions of recurrent neural network language model [C]//Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2011.

[6] Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling.[C]//Proc. Interspeech. 2012.

[7] Kurata G, Ramabhadran B, Saon G, et al. Language modeling with highway LSTM[J]. arXiv preprint arXiv:1709.06436, 2017.

[8] Chen X, Ragni A, Liu X, et al. Investigating bidirectional recurrent neural network language models for speech recognition[J]. Proc. ICSA INTERSPEECH, 2017.

[9] Lafferty J, McCallum A, Pereira F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[J]. Proc. International Conference on Machine Learning (ICML), 2001: 282–289.

[10] Andor D, Alberti C, Weiss D, et al. Globally normalized transition-based neural networks[J]. arXiv preprint arXiv:1603.06042, 2016.

[11] Wiseman S, Rush A M. Sequence-to-sequence learning as beam-search optimization[J]. arXiv preprint arXiv:1606.02960, 2016.

[12] Ostendorf M. Continuous-space language processing: Beyond word embeddings[C]// International Conference on Statistical Language and Speech Processing. : Springer, 2016.

[13] Goodman J T. A bit of progress in language modeling extended version[J]. Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72, 2001.

[14] Chelba C. Structured language modeling[J]. Academic Press Ltd, 2000.

[15] Khudanpur S, Wu J. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling[J]. Computer Speech & Language, 2000, 14: 355–372.

[16] Dyer C, Kuncoro A, Ballesteros M, et al. Recurrent neural network grammars[J]. arXiv preprint arXiv:1602.07776, 2016.

[17] Li X, Qin T, Yang J, et al. LightRNN: Memory and computation-efficient recurrent neural networks[C]//Advances in Neural Information Processing Systems. 2016: 4385–4393.

[18] Shim K, Lee M, Choi I, et al. SVD-Softmax: Fast softmax approximation on large vocabulary neural networks[C]//Advances in Neural Information Processing Systems. 2017: 5469–5479.

[19] Mikolov T. Statistical language models based on neural networks[J]. Ph.D. thesis, Brno University of Technology, 2012.

[20] Grave E, Joulin A, Cissé M, et al. Efficient softmax approximation for GPUs[J]. arXiv preprint arXiv:1609.04309, 2016.

[21] Gutmann M, Hyvärinen A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models[C]//Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 2010.

[22] Kumar S, Nirschl M, Holtmann-Rice D, et al. Lattice rescoring strategies for long short term memory language models in speech recognition[J]. arXiv preprint arXiv:1711.05448, 2017.

[23] Wang B, Ou Z, Tan Z. Trans-dimensional random fields for language modeling[C]//Proc. Annu. Meeting of the Association for Computational Linguistics (ACL). 2015.

[24] Wang B, Ou Z, Tan Z. Learning trans-dimensional random fields with applications to language modeling[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2017.

[25] Xu H, Ou Z. Joint stochastic approximation learning of helmholtz machines[J]. International Conference on Learning Representations (ICLR) Workshop Track, 2016.

[26] Allauzen C, Mohri M, Roark B. Generalized algorithms for constructing statistical language models[C]//Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. 2003.

[27] Martin S, Liermann J, Ney H. Algorithms for bigram and trigram word clustering[J]. Speech Communication, 1998, 24: 19–37.

[28] Brown P F, Desouza P V, Mercer R L, et al. Class-based n-gram models of natural language[J]. Computational linguistics, 1992.

[29] Chen S F. Shrinking exponential language models[C]//Proc. Human Language Technologies: Ann. Conference of the North American Chapter of the Association for Computational Linguistics. 2009.

[30] Wang B, Ou Z, Li J, et al. Joint-character-poc n-gram language modeling for chinese speech recognition[C]//Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on. : IEEE, 2014: 24–28.

[31] Rosenfeld R. Adaptive statistical language modeling: a maximum entropy approach[J]. Ph.D thesis, 1994.

[32] Pietra S D, Pietra V D, Lafferty J. Inducing features of random fields[J]. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 1997, 19: 380–393.

[33] Mnih A, Hinton G. Three new graphical models for statistical language modelling[C]// Proceedings of the 24th international conference on Machine learning. : ACM, 2007.

[34] Kingma D, Ba J. Adam: A method for stochastic optimization[J]. arXiv:1412.6980 [cs.LG], 2014.

[35] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011.

[36] Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural Networks for Machine Learning, 2012.

[37] Martens J. Deep learning via hessian-free optimization[C]//Proc. International Conference on Machine Learning (ICML). 2010.

[38] Bordes A, Bottou L, Gallinari P. Sgd-qn: Careful quasi-newton stochastic gradient descent[J]. The Journal of Machine Learning Research, 2009, 10: 1737–1754.

[39] Byrd R H, Hansen S, Nocedal J, et al. A stochastic quasi-newton method for large-scale optimization[J]. arXiv preprint arXiv:1401.7020, 2014.

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

[41] Zoph B, Vaswani A, May J, et al. Simple, fast noise-contrastive estimation for large rnn vocabularies.[C]//HLT-NAACL. 2016.

[42] Chen X, Liu X, Gales M J, et al. Recurrent neural network language model training with noise contrastive estimation for speech recognition[C]//Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. 2015.

[43] Oualil Y, Klakow D. A batch noise contrastive estimation approach for training large vocabulary language models[J]. arXiv preprint arXiv:1708.05997, 2017.

[44] Kim Y, Jernite Y, Sontag D, et al. Character-aware neural language models[C]//Thirtieth AAAI Conference on Artificial Intelligence. 2016.

[45] Jozefowicz R, Vinyals O, Schuster M, et al. Exploring the limits of language modeling[J]. arXiv preprint arXiv:1602.02410, 2016.

[46] Chen X, Liu X, Ragni A, et al. Future word contexts in neural network language models[J]. ASRU, 2017.

[47] Dauphin Y N, Fan A, Auli M, et al. Language modeling with gated convolutional networks[J]. arXiv preprint arXiv:1612.08083, 2016.

[48] Pham N Q, Kruszewski G, Boleda G. Convolutional neural network language models[C]//Proc. of EMNLP. 2016.

[49] Rosenfeld R, Chen S F, Zhu X. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration[J]. Computer Speech & Language, 2001, 15: 55–73.

[50] Amaya F, Benedí J M. Improvement of a whole sentence maximum entropy language model using grammatical features[C]//Proc. Ann. Meeting of the Association for Computational Linguistics (ACL). 2001.

[51] Ruokolainen T, Alumäe T, Dobrinkat M. Using dependency grammar features in whole sentence maximum entropy language model for speech recognition.[C]//Baltic HLT. 2010.

[52] Wu J, Khudanpur S. Combining nonlocal, syntactic and n-gram dependencies in language modeling.[C]//Proc. EUROSPEECH. 1999.

[53] Buys J, Blunsom P. A bayesian model for generative transition-based dependency parsing[J]. arXiv preprint arXiv:1506.04334, 2015.

[54] Henderson J. Inducing history representations for broad coverage statistical parsing[C]// Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics. 2003.

[55] Vinyals O, Kaiser Ł, Koo T, et al. Grammar as a foreign language[C]//Advances in Neural Information Processing Systems. 2015: 2773–2781.

[56] Koller D, Friedman N. Probabilistic graphical models: principles and techniques[M]. : MIT press, 2009

[57] Tan Z. Optimally adjusted mixture sampling and locally weighted histogram analysis[J]. Journal of Computational and Graphical Statistics, 2017, 26: 54–65.

[58] Robbins H, Monro S. A stochastic approximation method[J]. The Annals of Mathematical Statistics, 1951: 400–407.

[59] Benveniste A, Métivier M, Priouret P. Adaptive algorithms and stochastic approximations[M]. : New York: Springer, 1990

[60] Chen H. Stochastic approximation and its applications[M]. : Springer Science & Business Media, 2002

[61] Gu M G, Zhu H T. Maximum likelihood estimation for spatial models by markov chain monte carlo stochastic approximation[J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2001, 63: 339–355.

[62] Polyak B T, Juditsky A B. Acceleration of stochastic approximation by averaging[J]. SIAM Journal on Control and Optimization, 1992, 30: 838–855.

[63] Green P J. Reversible jump markov chain monte carlo computation and bayesian model determination[J]. Biometrika, 1995, 82: 711–732.

[64] Rosenfeld R. A whole sentence maximum entropy language model[C]//Proc. Automatic Speech Recognition and Understanding (ASRU). 1997.

[65] Wiesler S, Ney H. A convergence analysis of log-linear training[M]//Advances in Neural Information Processing Systems. 2011

[66] Goodman J. Classes for fast maximum entropy training[C]//Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2001.

[67] Chelba C, Mikolov T, Schuster M, et al. One billion word benchmark for measuring progress in statistical language modeling[J]. arXiv preprint arXiv:1312.3005, 2013.

[68] Hinton G. A practical guide to training restricted boltzmann machines[J]. Momentum, 2010, 9: 926.

[69] Shazeer N, Pelemans J, Chelba C. Sparse non-negative matrix language modeling for skip-grams [C]//Proc. INTERSPEECH. 2015.

[70] Neal R M. Annealed importance sampling[J]. Statistics and Computing, 2001, 11(2): 125–139.

[71] Wang B, Ou Z. Language modeling with neural trans-dimensional random fields[C]//IEEE Automatic Speech Recognition and Understanding Workshop. 2017.

[72] Wang B, Ou Z. Learning neural trans-dimensional random field language models with noise-contrastive estimation[J]. IEEE International Conference on Acoustics, Speech and Signal Processing, 2018.

[73] Van Den Oord A, Dieleman S, Zen H, et al. Wavenet: A generative model for raw audio[J]. CoRR abs/1609.03499, 2016.

[74] Liu J S. Monte carlo strategies in scientific computing[M]. : Springer Science & Business Media, 2008

[75] Vaswani A, Zhao Y, Fossum V, et al. Decoding with large-scale neural language models improves translation.[C]//EMNLP. 2013.

[76] Xiang H, Wang B, Ou Z. The THU-SPMI CHiME-4 system: Lightweight design with advanced multi-channel processing, feature enhancement, and language modeling[C]//CHiME-4 Workshop. 2016.

# 致　谢

衷心感谢导师欧智坚副研究员对本人的悉心指导。他的言传身教将使我终生受益。

在美国罗格斯大学统计系进行六个月的合作研究期间，承蒙 Zhiqiang Tan 教授热心指导与帮助，不胜感激。

本课题承蒙国家自然科学基金资助，特此致谢。

# 声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

# 个人简历、在学期间发表的学术论文与研究成果

## 个人简历

1990 年 02 月 08 日出生于河北省阜城县。

2008 年 9 月考入清华大学电子工程系电子信息科学与技术专业，2012 年 7 月本科毕业并获得工学学士学位。

2012 年 9 月免试进入清华大学电子工程系攻读信息与通信工程博士学位至今。

## 发表的学术论文

[1] Bin Wang, Zhijian Ou, Jian Li, Akinori Kawamura. "Joint-Character-POC N-Gram Language Modeling For Chinese Speech Recognition". International Symposium on Chinese Spoken Language Processing (ISCSLP), 2014. (B 类会议，EI 收录，检索号：20144900274060)

[2] Bin Wang, Zhijian Ou, Zhiqiang Tan. "Trans-dimensional Random Fields for Language Modeling". Annual meeting of the Association for Computational Linguistics (ACL), 2015. (A 类会议，EI 收录，检索号：20154201386950)

[3] 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), 2017, 40(4): 876-890. (SCI 收录, 影响因子:8.329, 论文检索号:FY2ZU)

[4] Bin Wang, Zhijian Ou. "Language Modeling with Neural Trans-dimensional Random Fields". IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2017. (B 类会议，已录用)

[5] Bin Wang, Zhijian Ou. "Learning Neural Trans-dimensional Random Field Language Models with Noise-contrastive Estimation". IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018. (顶尖级国际会议，已录用)