申请上海交通大学硕士学位论文

面向 ASR 的结构化语言模型的研究

论文作者 吴　越

学　　号 115033910029

导　　师 俞凯教授

专　　业 计算机科学与技术

答辩日期 2017 年 12 月 17 日

Submitted in total fulfillment of the requirements for the degree of Master
in Computer Science and Technology

Research on structured language model for ASR

Yue Wu

Advisor
Prof. Kai Yu

Department of Computer Science and Engineering, School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, P.R.China

Jan. 15th, 2017

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日　期：_____年_____月_____日

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保　密 □，在 _____ 年解密后适用本授权书。

不保密 □。

(请在以上方框内打 ✓)

学位论文作者签名：_____　　指导教师签名：_____

日　期：_____年 _____月 _____日　　日　期：_____年 _____月 _____日

面向 ASR 的结构化语言模型的研究

摘 要

最近，自动语音识别（ASR）和自然语言处理（NLP）的研究推动了语言模型的发展。其中，循环神经网络语言模型（RNNLM）由于其存在特殊的环状结构，使得训练过程中的历史信息可以被记录并作用于当前单词的训练，而比 N-gram 等传统语言模型表现更佳。近几年，Mikolov 等人证明在训练语言模型时加入额外信息同原始数据一起训练能有效提高语言模型的性能，然而引入的信息过于单调，也没有针对中文做研究。因此本文从多视角融合的角度针对中文去考虑。

因为语言本身具有一些约束特性，如单词的识别会依赖于词性的连接规律、话题的约束和场景地点的区分等。本文首先将数据集预处理，分别通过对应的算法得到数据对应的词性、话题和场景等不同的信息，并进行标注得到新的数据集。然后，以词为单位对带标记的数据进行解析，得到其词向量和对应信息的特征向量。最后将这些向量一同作为 RNNLM 的输入层进行训练。通过多组对比实验可以发现，多视角融合的 RNNLM 相比于传统的 RNNLM 在各类数据上均有 8

在这篇论文中，我们将首先介绍语言模型的基本概念和传统的语言模型；然后介绍 RNNLM 的结构和训练流程，以及其关键数学理论基础；接着会详细说明多视角融合的 RNNLM 视角提取量化，以及训练框架和原理；最后展示和讨论实验结果。

关键词： 关键词： 自动语音识别 语言模型 神经网络 多视角语言模型

Research on structured language model for ASR

ABSTRACT

Recently，research in automatic speech recognition （ASR）and natural language processing （NLP）promotes the development of language model. Among them，the RNN （recurrent neural network）language model which has special ring structure can record the history of training and use it in current training，thus performing better than traditional language model，for instance，N-gram language model. Mikolov demonstrates that the performance of language model can be effectively improved by combining the original data with additional information. However the information they used is quite simple，and also don’t aim at Chinese. our work will focus on an advanced method which is combine different views and aimed at Chinese.

The constraints features of the language can help the training process of language model in neural network judge better. For example，word recognition depends on the connection rule of POS（part-of-speech），the constraints of topic，the scene distinction and location. First we preprocess the data set and obtain the information about POS，constraints of topic and scene distinction by using the corresponding algorithm. And we present a new data set by marking the information. Then parsing the marked data measured in word and obtain word vector and feature vector of corresponding information. After that we take these vectors as the input layer of RNN language model for training. By comparing multiple sets of experiments，multi-view RNN language model has 8

In this paper，we introduce the basic concept of language model and traditional language model，explain the structure of the RNN language model，expand the training process and the mathematical theory. Then we illustrate the process of view extraction，quantification and training framework with the principle in multi-view RNN language model. Finally we present and discuss the results of experiment for futher exploration.

KEY WORDS: SJTU, master thesis, XeTeX/LaTeX template

目 录

插图索引

表格索引

算法索引

主要符号对照表

ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率

ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率
ϵ 介电常数
μ 磁导率

第一章 绪论

1.1 语言模型及其研究背景

随着计算机运行速度的提升和机器学习领域的发展，机器学习已经成为当今大的研究趋势。智能语音可以让计算机听懂甚至理解人类的语言，因此成为了智能交互中很重要的一部分，语言模型的作用功不可没。同时，大规模语料库的出现和现代计算机计算能力的提升，为自然语言统计处理方法的实现提供了可能，统计方法的成功使用推动了语言模型的发展。语言模型作为衡量一句话的流畅及合理程度的模型，语言模型在语音识别和自然语言处理任务中都得到了广泛的应用。在自然语言处理任务中，语言模型不仅可以用来衡量语句的符合语言规律及通顺程度，更可以和众多别的自然语言处理任务相辅相成，相互促进。在语音识别任务中，语言模型和声学模型合作完成，通过解码过程将音频信息转化为最为合理的文字。本论文着重介绍语言模型在语音识别任务中的研究和优化。

自古以来，语音都是人类最重要的交流方式之一。自电话发明以来，学者们致力于机器的语音的研究工作。19 世纪晚期，在语音工作的种种问题中，自动语音识别（Automatic Speech Recognition，ASR）成为最具挑战性和吸引力的任务之一，它是通过先记录语音波形并通过一系列算法自动转换为文本。然而，对 ASR 的研究在 20 世纪初却进展缓慢，甚至 1969 年，贝尔实验室的约翰 · 皮尔斯还曾声称自动语音识别在几十年内不会成为现实。

不过，在十九世纪 70 年代，语音识别领域有了巨大的突破。各类方法在语音识别领域崭露头角，其中就包括语言模型。在接下来的几十年里，随着机器学习方法的提出，和深度学习的证实与应用，神经网络被用于语音识别领域。基于神经网络的各种语言模型训练方法也逐步进入研究者的视角。在此基础上，各种结构化神经网络的语言模型相关研究再一次推动了语言模型的性能和应用空间。

语音识别系统的目的是能通过给定的语言波形而产生一个单词序列（或者可能是适用于普通话之类语言的汉字序列）。

ASR 系统的基本结构如图5–2所示[1]。

语音识别的第一阶段是压缩语音信号（逐帧）为声学特征向量的流，称为观测值。

提取的观测向量被设计成包含足够的信息以及足够紧凑来执行有效的语音识别。这一过程被称为前端处理或特征提取。/ 从观测特征向量说开来，ASR 系统可分为三个主要部分：词汇，语言模型和声学模型。

词汇或者称为词典，用于映射出在声学模型中的子字单元到词汇与语言模型中的实际单词。语言模型代表了已说出句子的当地句法和语义信息。它对每个单词序列都分配了个可能性。声学模型，通常是语音研究者们最感兴趣的，映射了语音观测值到子字单元。GMM-HMM 框架一直是 ASR 框架的最先进技术，直到最近深度神经网络（DNN）被引入到 ASR 系统中。计划中的深度神经网络 HMM（DNN-HMM）相比于最先进的 GMM-HMM 来说在两种任务上都取得了显著进步，音素识别[1] 和大词汇连续语音识别（LVCSR）[2] 任务。它使用一个深度神经网络（DNN）来计算聚类后状态并将

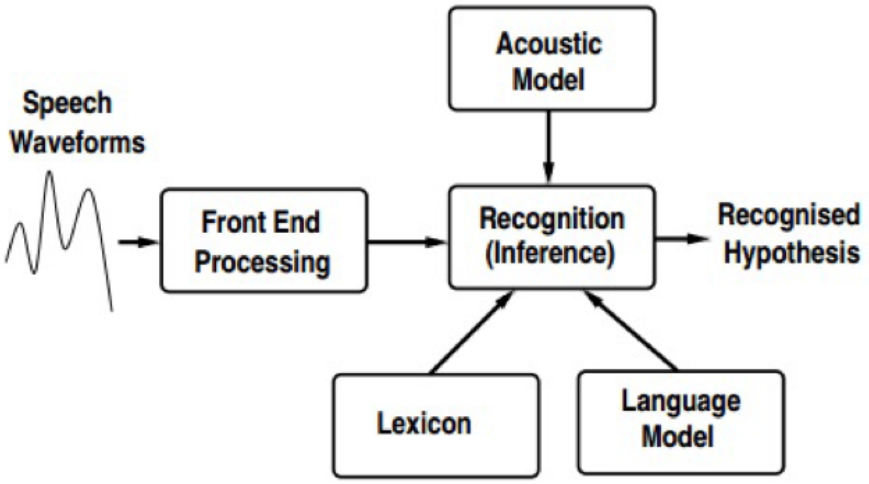


图 1–1 ASR structure
Fig 1–1 自动语音识别系统的结构

其转换为可用于 HMM 的状态和条件似然概率。

深度神经网络（DNN）比起浅层的神经网络有更强大的模拟/建模能力。但是，它也更有可能因为正常初始化策略和反向传播（BP）优化而陷入局部的最优。RBM（受限玻耳兹曼机）的预训练[1]或区别预训练[3] 被提出来用作初始化权重矩阵到一个更好的起点，这使 BP 微调过程更容易，而且收敛更快速。最近，文献[4][5] 介绍了更先进的训练标准，它们已被应用于得到更精确的模型。除了混合框架，DNN 也可以用于获得更完善的瓶颈特征[6]，以训练 GMM-HMM 系统并已经实现了与 DNN 相匹配的性能。

随着自动化语音识别的不断优化和发展，在识别中扮演重要角色的语言模型的研究同样得到迅速发展。从传统的统计学语言模型 N-gram 到神经网络语言模型，从普通的深度神经网络语言模型到有历史记忆功能的循环神经网络，再到更优秀但是更复杂的长短时间变化神经网络，甚至是各种对模型结构上的优化，一步步使语言模型研究更加深入，从而大幅度提升语言模型的性能。

1.2 语言模型的相关研究

在 20 世纪 90 年代，最初的语言模型 N-gram 语言模型开始被提出。N-gram 语言模型原理简单，仅需要掌握简单的概率论基础知识和极大似然概率就能实现，而且速度很快，对于当时的计算机硬件条件来说适用性很高，很快就被广泛接受和应用开来。这样一来，很多缺点和不足也就暴露出来，比如当训练数据不足的情况下很容易出现很多单词的预测概率为 0，于是就有很多学者提出各种平滑算法来解决语料稀疏的问题。比如基于二项后验分布的后退（back-off）平滑方法[7]，基于普通计数的平滑方法[8] 和 Kneser-Ney 平滑[9] 等。其中 Kneser-Ney 平滑算法是目前比较标准的，而且是非常先进的平滑算法。

除了这些平滑算法，在 N-gram 的训练形式和语料的预处理方面专家们也是下足了功夫，例如基

于缓存的（Cache-based N-gram 模型用缓存记录历史信息[10]，基于类别的（Class-based）N-gram 模型将单词进行分类处理[11] 以及基于话题的（Topic-based）N-gram 模型在训练单词的时候为其配备话题附加信息以达到更好的训练等[12]。

随着 21 世纪深度学习方法被证实可用，神经网络算法开始被用于自动语音识别和语言模型的学习当中。开始的 DNN 算法由于无法记录较长的历史信息而不适用于语言模型，因为语言是具有前后文连贯性的。接着随着 RNN 语言模型的提出，发现具有递归结构能够记录历史信息的循环神经网络（Recurent Neural Network,RNN）网络非常适用于语言模型。

RNN 由于递归结构的存在使得语言的历史信息能够被记住，十分符合语言学的规律，使得语言模型的性能从统计模型 N-gram 有了很大的提升。同时 RNN 模型的训练方式也从以前普通 DNN 的反向传播算法（BP）变化成了基于时间的反向传播算法（BPTT），这个算法将会在第二章进行详细介绍。同样的，学者将优化 N-gram 语言模型的思想扩展到语言模型上，提出一系列优化 RNN 语言模型的算法。例如 Class based RNNLM 被提出，它的思想同 N-gram 一样，对训练单词进行分类训练，能够有效的提高训练速度，并且对于丰富多彩的语言来说，那些出现概率偏小的词汇在预测中容易导致概率为 0 的问题得到了平滑。优化深度神经网络（DNN）的各种训练算法同样适用于 RNN 语言模型的训练，从开始的一个词一个词的训练的随机梯度下降，到多个词同步训练的批量梯度下降，以及训练中加入块的机制，都大幅度提升了 RNN 模型的训练速度，同时也一定程度提升了语言模型的性能。从语言模型训练速度的方向考虑，以前的语言模型不能同声学模型一样进行 minibatch 训练（nimibatch 即是把训练数据分批输入到神经网络中进行学习），因为语言模型的识别往往和前后文以及历史信息关系密切，然而去年有学者提出可以换种方式考虑 minibatch，同时读入很多句子，在不同的没有关联的句子中选取不同的单词进行 minibatch 输出，则不会影响学习相关性的情况下还能对速度有所提升。另外多核并行学习，利用 GPU 和 Intel 函数库进行训练等方法逐渐被用于语言模型训练上。

然而 RNN 语言模型依然有一个普遍存在的问题——梯度消失。即当语句过长时，距离现在较长的历史会随着一步步的递归使得历史影响消失，这种情况使得 RNN 虽然理论上能用到的历史信息是全部历史，但是真正产生作用的历史信息依然非常有限。针对这个问题，有人提出加入长短时间记忆（LSTM）的神经网络语言模型来学习那较长的句子。面对这些些有很长一段时间信息滞后并且滞后时间的长短未知的句子，它通过分割记忆块的形式控制误差的传递时机，训练得到语言模型的性能相比传统 RNNLM 具有一定提升。好在无论是训练方法的提升（小批量随机梯度下降），还是计算方法的提升（多线程 GPU 并行）以及硬件计算能力的提升为训练具有相对于 RNN 来说几倍参数量的 LSTM 模型提供了可能性。

在上述技术基础上，针对基础语言模型进行的各种创新结构的研究也在紧锣密鼓的进行。多视角（Multi-view）语言模型是将多种特征向量加入到语言模型中辅助训练更好的语言模型的结构，比如加入前后文相关信息的 RNN 语言模型于近年被 Mikolov 提出，它在训练当前单词的时候加入前后文块，并为词性加入话题信息，结合话题模型一起进行训练，形成一个受话题制约的 RNNLM，可以有效避免数据在不同类型的数据子集上训练话题模型导致的数据分裂。多任务（Multi-task）学习同样受到了很大的关注，其主要思想是在训练语言模型时加入其它的任务混合训练，共用一定数量的隐藏层或参数矩阵，达到相互促进相互制约的作用。联合学习（Joint train）作为多视角和多任务模型的结合，也是最复杂的一个。不仅像多视角模型一样拥有多维视角特征，也像多任务学习一样

多种不同任务共同学习以达到相互促进和泛化的作用。往往多任务的输出会以各种经过尝试的较好方式作为额外视角加入到下个时间点的训练中。

上述提到的各种语言模型的结构化尝试也是本文的工作重点，额外的丰富的信息和合理的结构往往能使语言模型发挥更好的效果。

1.3 本文的研究内容及贡献

1.4 论文结构

本论文剩下的部分是这样组织内容的：第二章主要介绍什么是语言模型, 语言模型的作用，以及各种语言模型。。首先介绍最基本的，出现最早也是应用范围最广的 N-gram 语言模型，它是非神经网络语言模型，本文着重从它的定义和结构，以及数学原理等方面进行分析，然后简单介绍它的优化和变形，在第五章 N-gram 开源工具 srilm 将会作为实验对比工具出现。然后引入当前应用最广的神经网络语言模型，不仅从概念上介绍神经网络的定义，更是从数学原理、模型结构及训练的角度阐述神经网络语言模型。在此基础上由浅入深地介绍从基础的循环神经网络语言模型（RNNLM）到效果更好但更为复杂的长短时间变化（LSTM）RNNLM 的各种细节。毫无疑问，这部分是本论文和本人研究生阶段的研究工作的基石，

第三章主要介绍面向结构化语言模型的 $beta$ 稳定算子自适应算法。先解释了历史研究和研究动机。然后在历史研究的基础上介绍了我们将其加入到 LSTM 语言模型中的方法和原理，并详述其数学原理。最后介绍了加入 L2 范数的优化。

第四章是本文最重要的章节，也是本文工作内容和贡献所在——深入探讨结构化语言模型的研究。前半章是各种结构化语言模型已有工作的调研和实验的复现。主要根据结构的特征分为 Multi-task、Multi-view 和 Joint train 三种不同的模型结构。分别详细介绍这三种结构的特征、表现和各自的优缺点。后半章详细介绍本文的三个主要工作：内容及创新点，以及研究的步骤、技术细节和进展。首先介绍本文在研究中主要用到的三种辅助维度：词性（POS）、命名实体（NER）、语义块标注（Chunk），以及他们对语言模型的训练有何积极作用，然后分别提出各自的特征向量提取方法。紧接着本文提出单向自信息 Multi-view 和 Joint train 模型的结构，包括模型结构的动机，不同的训练方法的尝试以及本模型的创新点及优势。另外本文还介绍了 Teacher-student 的相关研究和细节。

第五章是实验与分析部分, 首先引入实验的基础知识，同时介绍实验的平台、数据集等一系列相关内容。然后介绍实验的设计和流程，本文将从上述三种基础结构和本文的两种创新结构出发，进行全面的实验和对比。最对所有的实验数据进行展示、对比、分析，得出结论并进行总结。

第二章 语言模型

语言模型通常被用来解决这类问题：语音识别、机器翻译、音字转换、自动文摘、问答系统等。在自动语音识别（ASR）领域中，语言模型扮演着十分重要的角色。本文介绍的重点主要围绕语言模型在语音识别中的优化展开。

2.1 语言模型在语音识别（ASR）中的作用

一轮人机交互是从语音到文字的语音识别、从文字到语义的口语语义理解、从语义输入到反馈的对话状态跟踪和管理模块、从语义反馈又到文本的自然语言生成、从文本回到语音的语音合成。

语音识别（Automatic Speech Recognition, ASR）模块是上述一轮对话流程中的第一个组件，而且它的输出被用于口语语义理解的输入信息。语音波形经过前端的特征提取和各种预处理后，通过识别阶段，得到含有识别结果的文本输出，声学模型和语言模型就是合作完成这个识别功能的。语音识别的过程是将声学信号的特征序列（O）映射成词序列（W），如下公式 2–1所示：

$$\hat{W} = \arg\max_{W} P(W|O) \propto P(O|W)P(W) \tag{2–1}$$

公式 2–1最后分成了两部分 $P(O|W)$ 和 $P(W)$：其中 $P(O|W)$ 代表的是声学模型 (Acoustic Model, AM)，将声学信号转成所有可能的词序列；$P(W)$ 代表语言模型 (Language Model, LM)，其作用是在声学模型产生的所有可能的词序列中选择尽可能符合自然语言的词序列。

通俗的说，声学模型将语音的数字信号识别成具体的词和读音，然后输入到语言模型中，形成识别结果。例如有个句子是“我很酷”，当语言模型已经识别出来前两个字是“我”和“很”时，当输入第三读音是 ku 时，博大精深的汉字库并没有办法确定是“酷”还是“苦”或者是其它词，这时语言模型需要给出的是当历史信息是某一确定句子时，后面一个词的出现概率是多少，然后通过某个词在某一特定历史信息下出现的概率得到一段文字（句子）出现的概率，如公式 (2–2) 所示

$$P(s) = P(w_1) * P(w_2|w_1) * P(w_3|w_1w_2) * ... * P(w_m|w_1...w_{m-1}) = \prod_{i=1}^{m} P(w_i|w_1...w_{i-1}) \tag{2–2}$$

w_i 可以是字、词、短语或词类等，称为统计基元，w_i 的概率由 w_1，……，w_{i-1} 决定，由特定的 w_1，...，w_{i-1} 构成的序列成为 w_i 的历史。下文中我们统称 w 为单词。

很久以来，许多最新水平的语音识别系统都采用隐马尔科夫模型（Hidden Markov Models, HMMs）来给语音的声学信息建模[13]。而在最近几年，随着深度学习技术的发展，判别式神经网络被用进来估计 HMM 状态概率[14, 15]，甚至包括使用循环神经网络（Recurrent Neural Networks, RNNs）直接进行端到端的语音识别建模[16]。另外关于语言模型的建模技术，也得益于深度学习的发展，从传统的离散的 n-gram 语言模型[11] 转向连续空间的神经网络语言模型[17]。

因为一些原因限制，新的神经网络语言模型技术往往不能直接应用到语音识别解码（First-pass decoding）中去，因此神经网络语音模型往往会做 Rescore，即在解码器先使用 N-gram 模型给出一些中间结果，再让神经网络语言模型去做重估计（rescore）。

实际上语音识别的输出并不是简单的文本句子，因为在给定输入句子语音的情况下，语音识别模块会将其后验概率分配给相应识别出来的词。一种典型的输出形式就是 N 个最佳假设列表（N-best list）以及它们相应的概率，其中 N 是一个整数值，根据需求可以取 1 或者 10 等。这样的输出形式使用 N 个最可能的句子来近似在所有可能句子上的完整分布，这是一种有限的近似。通常情况下，这些最佳假设之间仅仅只有少量的词不同，而且很多都是短功能的词（比如语气词、冠词、其他一些停用词等，像“吗”、“么”、“的”等）。这样就会使得这些最佳假设句子的语义其实是差不多的。此外，一些概率较低的词往往会被这个 N 最佳假设列表忽略掉。一个 N-best list 的例子如图 2–1所示。

N-best list

Rank	Hypothesis	**AM** log probability	**LM** log probability
1	it's an area that's naturally sort of mysterious	-7193.53	-20.25
2	that's an area that's naturally sort of mysterious	-7192.28	-21.11
3	it's an area that's not really sort of mysterious	-7221.68	-18.91
4	that scenario that's naturally sort of mysterious	-7189.19	-22.08
5	there's an area that's naturally sort of mysterious	-7198.35	-21.34
6	that's an area that's not really sort of mysterious	-7220.44	-19.77
7	the scenario that's naturally sort of mysterious	-7205.42	-21.50
8	so it's an area that's naturally sort of mysterious	-7198.35	-21.34
9	that scenario that's not really sort of mysterious	-7217.34	-20.70
10	there's an area that's not really sort of mysterious	-7226.51	-20.01

语音识别的解码器可以输出词格或者词混淆网络作为中间结果。词格（word lattice）和词混淆网络（word confusion network）就提供了信息量更大的词识别的后验分布，而不像 N 最佳假设那样裁剪了得分较低的词。词格是一种可以高效地编码多种可能的词序列的有向图结构[19]，如图 2–1所示。所有可能的词序列就是每一条从开始结点到终结点的路径。另外，图中没有显示的信息还包括每条边的权值以及每个词的开始时刻和结束时刻信息。类似词格，词混淆网络也是一个可以枚举所有可能路径的有序图，不同的是它的结构有更多的限制。如图2–1所示，词混淆网络由一个结点序列组成，每个结点表示的是词边界，且连续的两个结点之间由一些互斥的词连接。需要注意的是从词格转换成词混淆网络时会有一些额外的路径产生，比如图 2–1中“*the scenario area*”在词混淆网络有但不在词格中出现。然而，词格中的路径都会同时存在于词混淆网络中。此外，词混淆网络中还引入了空边 *NULL*，用于表示与具体词无关的结点转移。

因为本工作研究的主要是循环神经网络自身的优化，所以一般来说我们都是先用一个基线的 N-gram 模型让语音识别解码器生成一个 N-best 列表，然后用循环神经网络对其进行重估计，用新模型的分数替换原本的分数，重排序，然后再重新对得分最高的假设句进行 WER 计算。

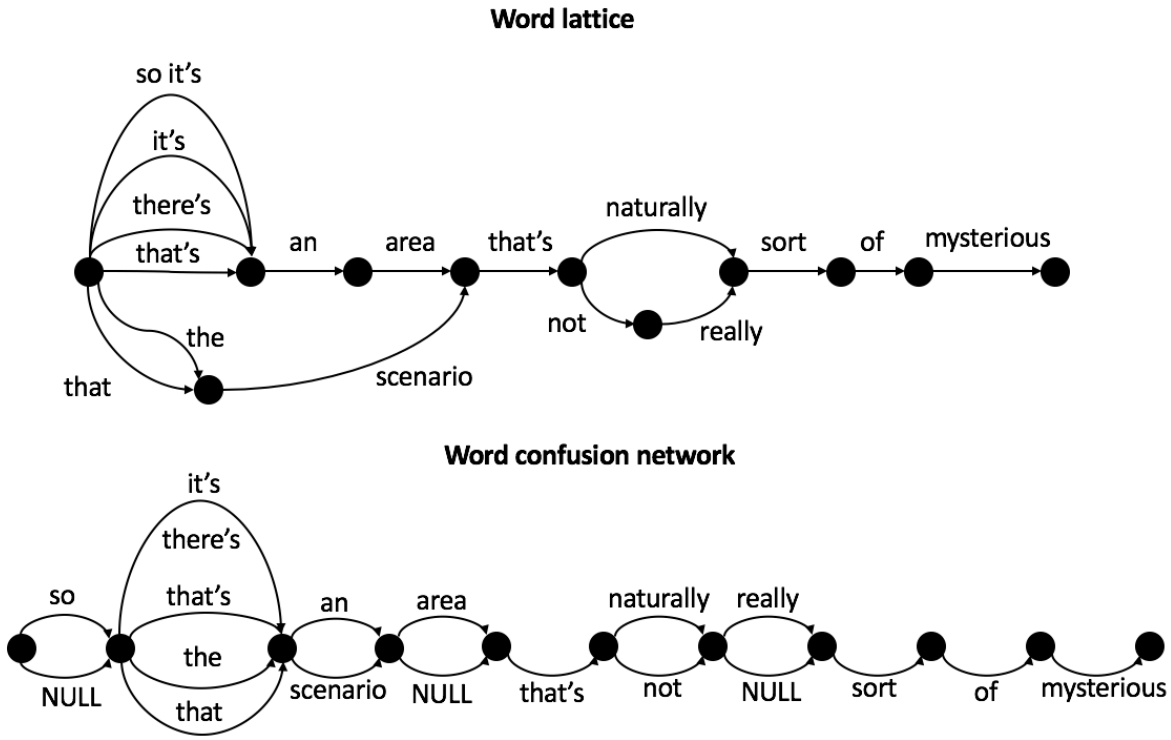


图 2–1 N-best 列表、词格、词混淆网络是常见的总结了各种备选句子及其后验概率的结构。这些例子引用自[18]。

2.2 N-gram 语言模型

2.2.1 N-gram 的概念

N-gram 是一种基于概率的语言模型，广泛应用于概率、通信理论、计算语言学。根据语言模型的公式可知，当前单词出现的概率依赖于前面的历史，一直到第一个单词。然而直观理解一下，很早的单词因为距离当前单词距离足够远，以至于影响近乎为零；并且如果一直考虑到第一个单词，会导致参数空间过大，时空复杂度过大，对计算资源造成严重浪费。因此我们考虑在计算一个单词出现概率的时候只考虑距离最近的 N 个历史，如图2–2所示，

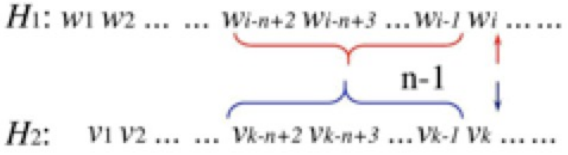


图 2–2 N-gram structure

Fig 2–2 N-gram 结构

我们将历史映射到等价类 S 中，则得到当前单词出现的概率公式**??**，

$$\begin{aligned} & S(w_1, w_2, ..., w_i) = S(v_1, v_2, ..., v_k) \\ iff \quad & H_1 : (w_{i-n+2}, ..., w_i) = H_2(w_{k-n+2}, ..., w_k) \\ & P(w_i|w_1, w_2, ..., w_{i-1}) = P(w_i|S(w_1, w_2, ..., w_{i-1})) \end{aligned} \tag{2–3}$$

考虑到整个句子，其 N-gram 语言模型概率计算如公式2–2,

$$P(S) = \prod_{i=1}^{N} P(w_i|w_{i-n+1}^{i-1}) \tag{2–4}$$

当 $n = 2$ 和 $n = 3$ 时分别称为 bigram 和 trigram。

2.2.2 数据平滑技术

当数据匮乏或稀疏时，会导致很多词的概率预测为零，大规模数据统计方法与有限的训练语料之间必然产生数据稀疏问题，对于这种零概率问题，我们用到数据平滑技术去解决它们。其基本思想是：在保证概率和为 1 的情况下，降低已出现 n-gram 条件概率分布，以使未出现的 n-gram 条件概率分布非零。

目前数据平滑的技术有很多，它是构造高鲁棒性语言模型的重要手段。且数据平滑的效果与训练语料库的规模有关。训练语料库规模越小，数据平滑的效果越显著；训练语料库规模越大，数据平滑的效果越不显著，甚至可以忽略不计。

2.2.3 N-gram 的扩展

N-gram 语言模型衍生出了一些变种，简单介绍一下：1) Class-based N-gram Model[11] 是按照词的分类建立语言模型，可以有效缓解数据稀疏问题。它在语言模型训练的时候将单词以某种规则分成不同的类，可以让部分出现很少的单词不至于无法识别以致概率接近零，另一方面且可以方便融合部分语法信息，同时还能解决 OOV（out of vocabulary）导致 PPL 偏高的问题。

2) Topic-based N-gram Model 将训练集按主题划分成多个子集，并对每个子集分别建立 N-gram 语言模型，例如采用狄利克雷混合作为参数的多项式分布估计的方法[12]，以解决语言模型的主题自适应问题。按照划分主题，可以有效降低语言模型的 PPL。

3) Cache-based N-gram Model 是一种新颖的语言模型，该模用一种缓存组件模式反映短期单词使用的情况[10]，即利用 cache 缓存前一时刻的信息，以用于计算当前时刻概率，为语言模型加入历史信息，以解决语言模型动态自适应问题。

4) Skipping N-gram Model 和 Trigger-based N-gram Mode 从名字可以发现，一改 N-gram 忽略远距离相关性的弊端，因为可能当前单词的预测会与较远时刻曾经出现过的那次发射概率有关。两者核心思想都是刻画远距离约束关系。

2.2.4 N-gram 的优劣

N-gram 模型的最主要的优势就是编码简单和训练快速。因为抽象来说只需要将训练数据过一遍，然后将信息和概率都存起来，就完成了。这方便不过多赘述。

但是缺点也很明显。首先，N 元组对长距离的上下文关系的建模就很有问题，如“我吃饭之后不喜欢 **洗碗**”，这里的“洗碗”这个词如果考虑到上下文里“吃饭”的信息就很容易作预测，但假设我们用的是三元组模型的话，模型只会考虑“不喜欢”这个上下文，“洗碗”就无从谈起了。

另外，在 N-gram 语言模型中，词与词之间是“离散”的，举个例子，“周三”和“周四”两个词我们都知道是很相似的词，然而，但在 N-gram 语言模型中，两个词却毫无联系，N-gram 模型只是统计了元组在数据中出现的次数。因此，可能“语文课“定在周三”得到的概率比较高（假设这句话数据中出现了多次），但“语文课定在周四”得到的概率就很低（如果这句话出现次数非常少的话）。

除此之外，N-gram 还有着巨大的参数量，且随着 N-gram 中 N 的增加，参数量成指数级增加，而当 N 比较小时语言模型的性能又收到影响。b n

2.3 循环神经网络语言模型 RNNLM

2.3.1 人工神经网络

人工神经网络是一种模仿生物神经网络的激励传播的结构和功能的数学模型或者说计算模型，它由大量的人工神经元以某种方式进行连接和计算，在学术界也常直接简称为神经网络（neural network，NN）。

神经网络由大量的节点相互连接构成，这些节点被称为“神经元”，或“单元”。每个节点实际上是一个运算单元，代表着一种特定的函数，称为激励函数（activation function）每个连接传递到节点的信息经过这些神经元就相当于进行了一次神经元所代表的函数的运算。

每两个节点间的连接都代表一个对于通过该连接信号的加权值，称之为权重（weight）或参数矩阵，这些参数矩阵代表着相当于人的神经网络的记忆功能，训练一个神经网络实质上是对这些权重不断修改不断完善的过程。网络的输出则根据网络的连接方式、权重值和激励函数的不同而不同，从而组成了不同类型不同功能的神经网络。每个神经网络自身通常都是对自然界某种算法或者函数的逼近，也可以说是对一种逻辑策略的表达。每个神经元的结构示意如图2–3所示，x_1 到 x_n 为输入向量，b 为偏置，f 为激励函数，t 为神经元输出。神经网络也分为单层神经网络和多层神经网络，

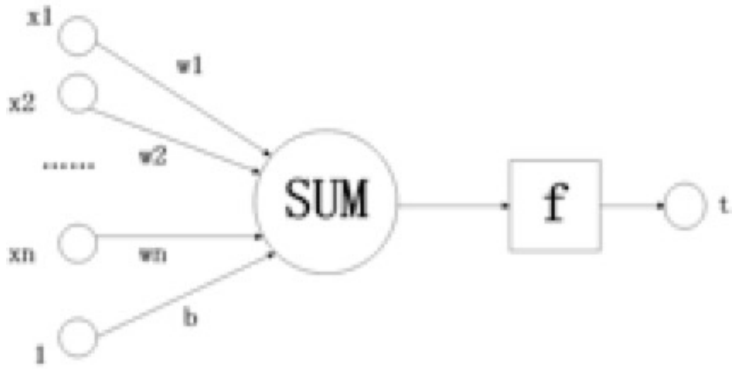


图 2–3 neure structure

Fig 2–3 神经元结构

后面我们研究涉及到的都是多层神经网络。神经网络的种类有很多：前馈神经网络、循环神经网络(RNN)、卷积神经网络等。

多层神经网络也称深度神经网络（DNN），是一种前馈神经网络[6]，是多层神经网络。它除了输入层和输出层，还有两个或两个以上的隐层，“深度”这个名字就体现在它有很多层上，其结构示意图如图2–4所示，神经网络语言模型是当今最成功的统计类语言模型，他们可以被广泛地应用于各种

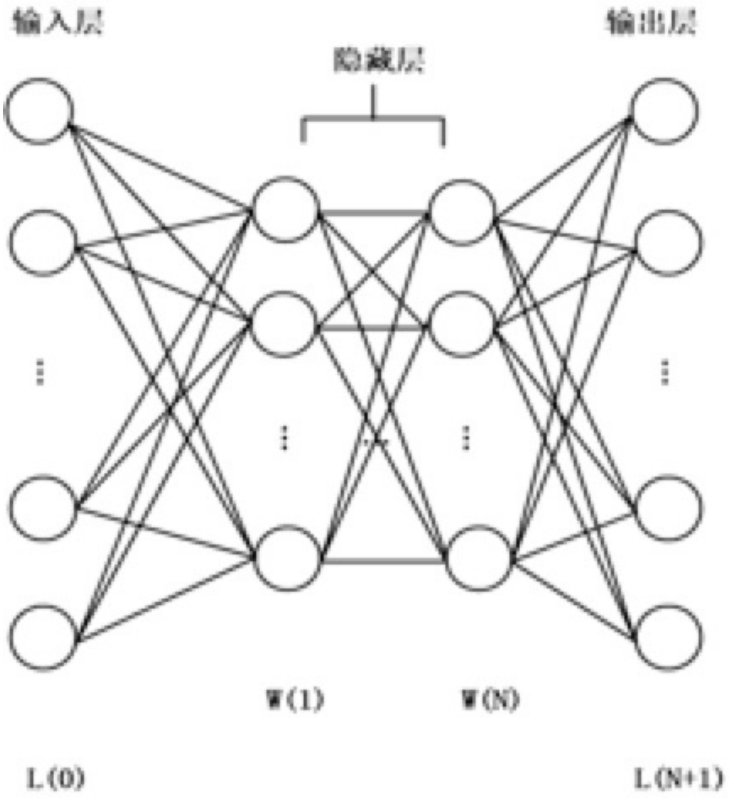


图 2–4 DNN structure
Fig 2–4 DNN 结构

机器学习任务。后文中本文主要使用的 RNN 和 LSTM 网络也和 DNN 类似，如果是深层的结构，则是基于此模型的。

2.3.2 RNNLM 的结构与原理

不同于上文所介绍的前馈神经网络语言模型，本章所要研究的循环神经网络语言模型 RNNLM 和前馈神经网络语言模型之间最主要的区别就是历史的表现：前馈神经网络语言模型中对当前单词有影响的历史仅仅是当前句子它前面的几个单词，而 RNNLM 中对当前单词有效的历史是整个训练过程中出现过的所有单词。原因是 RNNLM 中的隐藏层代表着所有前面出现过的历史，而不仅仅是当前句子，因此这个模型可以从理论上代描绘足够长的历史信息，从而使对单词及句子的预测准确度提升。

下面介绍 RNNLM 的网络结构，假设本文所涉及的 RNNLM 都是只有一个隐藏层的 RNNLM，如图2–5所示：w、s 和 y 都是向量，U、W 和 V 都是矩阵。w 为当前时刻 t 的输入层向量，代表当

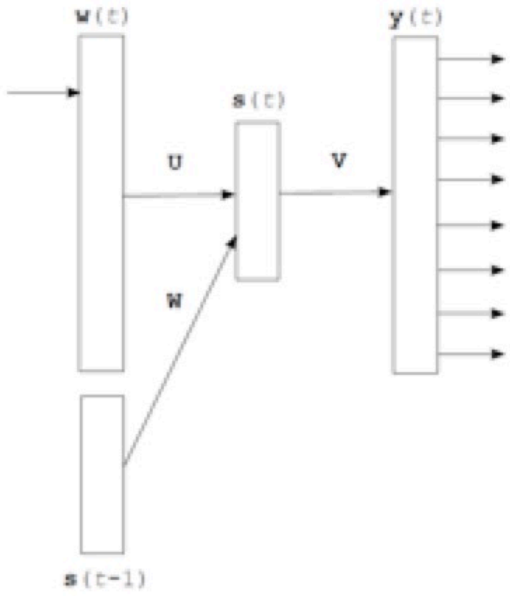


图 2–5 RNN structure
Fig 2–5 RNN 结构

前单词的编码，编码规则是：若当前单词在词典中的标号为 x，则 w 向量中第 x 维为 1，其他都是 0，x 的大小等于词表大小；$s(t-1)$ 是 $t-1$ 时刻的神经网络的隐藏层的输出；$s(t)$ 为当前的隐藏层，其计算方法如公式2–5所示；$y(t)$ 是当前 t 时刻的输出层，代表 t+1 时刻每个单词在当前历史下的概率，即 $P(w_t+1|w_t, s(t-1))$，计算公式如公式2–6所示；矩阵 U 和 W 是在输入层到隐藏层之间，V 是从隐藏层到输出层之间的矩阵。

$$s_j(t) = f\left(\sum_i w_i(t)\, u_{ji} + \sum_l s_l(t-1) w_{jl}\right) \tag{2–5}$$

$$y_k(t) = g\left(\sum_j s_j(t)\, v_{kj}\right) \tag{2–6}$$

上面的公式还能更简单地写成矩阵运算的形式：

$$s(t) = f(Uw(t) + Ws(t-1)) \tag{2–7}$$

$$y(t) = g(Vs(t)) \tag{2–8}$$

其中 $f(x)$ 和 $g(x)$ 分别为 sigmoid 激活函数和 softmax 激活函数。下面分别介绍一下这两个公式的意义、求导及作用：a) sigmoid 函数的公式与求导如公式2–9所示：

$$f(x) = \frac{1}{1+e^{-x}}$$
$$f'(x) = f(x)(1-f(x)) \tag{2–9}$$

2–5所示:

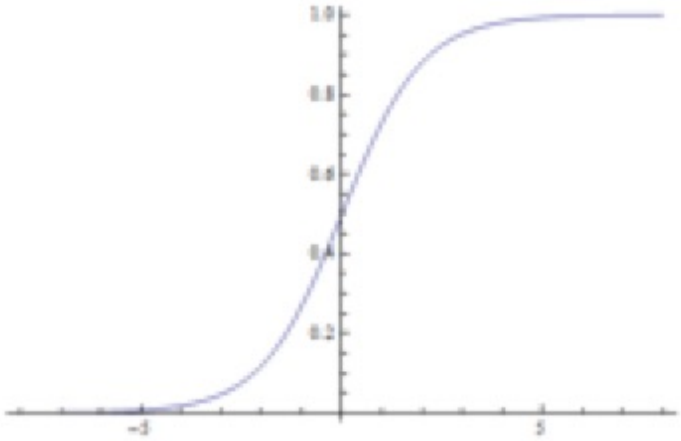


图 2–6 sigmoid structure
Fig 2–6 sigmoid 结构

函数图像如图2–6所示，结合图2–3的神经元结构，其 f 函数即为求隐藏层 s 的 sigmoid 函数，我们将输入层神经元的输入表示为 $x_1 - x_n$，输出层为 y，b 为偏置，则每一个神经元的计算过程可用公式2–25表示。

$$y = sigmoid\left(\sum_i x_i + b\right) \tag{2–10}$$

Sigmoid 函数相加可以模拟任何非线性函数，来表示语言模型的计算。

b) softmax 激活函数的公式如公式2–11所示，隐层的输出和矩阵 V 进行矩阵相乘后通过 softmax 函数得到输出层的发射概率。其中假设 x 为 softmax 函数的输入，输出为 g。

$$g(x_m) = \frac{e^{x_m}}{\sum_k e^{x_k}} \tag{2–11}$$

上文提到过，输出层的发射概率向量指的是从 0 时刻一直到 t 时刻以当前为历史的下一个单词的概率，如 y_i 指的是下一个单词是此表中的标号为 i 的单词出现的概率。而直观上理解 softmax 激活函数的意义是将输出层的输出形式转换成一个有效的概率分布，即输出向量的所有元素都大于 0，并且和为 1。

2.3.3 RNNLM 的练流程和步骤

结合上面的 RNNLM 的结构图，需要强调 RNNLM 需要学习的是输入层和隐藏层之间的矩阵 U、W 以及隐藏层到输出层之间的矩阵 V，而并不是那些神经元，神经元只是完成计算功能的单位。知

道了这一点之后，可以归纳出 RNNLM 的训练流程：如图2–7所示。

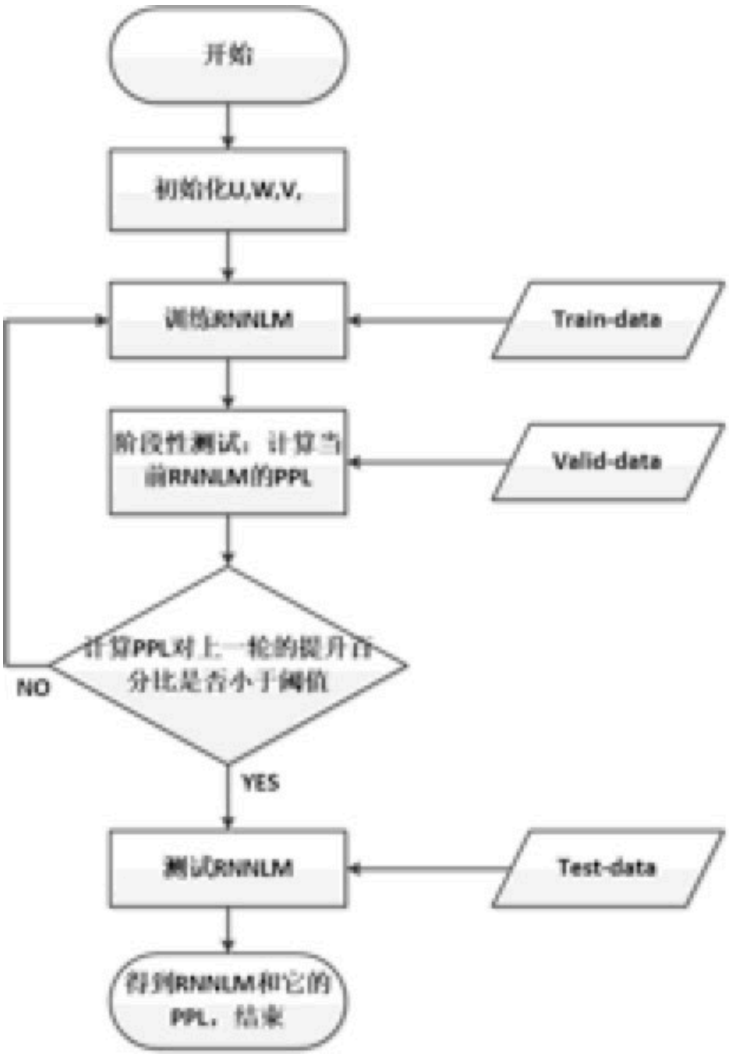


图 2–7 RNNLM training process

Fig 2–7 RNNLM 训练流程

首先对 RNNLM 进行初始化，即 U、W 和 V 的初始化，我们采取的方式是随机初始化，对矩阵中的每个元素随机一个 0 至 0.1 之间的实数；接着我们将训练集输入到神经网络中训练这三个矩阵，得到第一轮的训练结果；这时我们用 valid 集合去计算当前 RNNLM 的混淆度（PPL）和上一轮的结果比较（至少训练两轮，第一轮不比较），看 PPL 提升的程度是多少，若小于一定的数值则说明当前 RNNLM 已经趋于稳定，继续训练下去基本上对它没有改变，这时退出训练过程，得到最终的 RMMLM；最后我们参考前面用 valid 集测试语言模型的方法，用 test 集对其进行测试，测出来的 PPL 即为最后的 PPL。对于第二步训练 RNNLM 有多轮训练，其某一轮训练的具体的训练步骤如图2–8所示。

（1）设置一个时间计数器 t，初始 $t=0$，初始化隐藏层神经元的状态，令其为 1。

（2）把时间计数器 t 的值增加 1。

（3）更具当前的单词 wt 处理出当前输出层的向量 $w(t)$，具体方法是：初始化一个大小为的每一维都为 0 的向量，是词表的大小；求出当前单词 w_t 在词表中的序号 p，将 $w(t)$ 的第 p 维置为 1。

（4）将上一时刻，即 $t-1$ 时刻的隐藏层向量 $s(t-1)$ 复制储存，作为这时刻的输入。

（5）执行神经网络的前向传播得到隐藏层 $s(t)$ 和输出层 $y(t)$，前向传播的具体方法是：首先输入层 $w(t)[1*v_s]$ 与矩阵 $U[v_s*h_s]$ 进行矩阵相乘的结果中间向量 $s_1[1*h_s]$ 与上一轮隐藏层 $s(t-1)[1*h_s]$ 与矩阵 $W[h_s*h_s]$ 进行矩阵相乘得到中间向量 $s2[1*h_s]$；然后将中间向量 s_1 与中间向量 s_2 相加得到当前 t 时刻的隐藏层 $s(t)$；最后将 $s(t)[1*h_s]$ 与矩阵 $V(h_s*v_s)$ 进行矩阵相乘得到输出层向量 $y(t)[1*v_s]$。

（6）计算输出层的梯度误差 $e(t)$。

（7）反向传播输出层的梯度误差 $e(t)$ 依次到输出层与隐藏层之间的矩阵 V、隐藏层 $s(t)$、输入层到隐藏层之间的矩阵 U 和 W，并根据梯度误差和学习率修改它们。

（8）判断是否所有的训练数据都被处理过了，如果是则跳出循环，否则转到步骤（2）。

2.3.4 RNNLM 的训练数学基础及 BPTT

2.3.4.1 普通反向传播 (BP)

要把循环神经网络中的基于时间的反向传播算法讲清楚，我们首先要介绍 DNN 中的普通的反向传播。见 DNN 的结构图2–4，其具有多个隐层，反向传播的思想就是先求出输出层的梯度误差，然后每一个隐藏层的梯度误差都可以根据其后面一层的隐藏层的梯度误差得到；得到梯度误差后，我们可以根据微分的方法求出待求参数的梯度，根据梯度误差、梯度、和学习率对当前参数进行更新。首先我们需要求输出的损失函数，在分类器中最长见的损失函数是负似然对数函数（negative log-likelihood），如图2–5在输出层我们会对输出向量进行 softmax 操作，因此我们可以将损失函数写成公式2–12的形式：

$$L(\theta) = -\sum_{i=1}^{T} \log\left(P\left(s=t_i|o_i\right)\right) \tag{2–12}$$

损失函数也被称为交叉熵我们可以推断出损失函数对输出层的偏导数公式如公式2–13：

$$\begin{aligned} e_o(t) = \frac{\partial L}{\partial {y_i}^{L+1}} &= -\partial \log\left(\frac{e^{{y_t}^{L+1}}}{\sum\limits_i e^{{y_i}^{L+1}}}\right) \Big/ \partial {y_t}^{L+1} \\ &= \frac{{e_i}^{L+1}}{\sum\limits_i e^{{y_i}^{L+1}}} - \frac{\partial {y_t}^{L+1}}{\partial {y_i}^{L+1}} \\ &= P\left(s=i|o\right) - [t=i] \end{aligned} \tag{2–13}$$

我们令 $d(t)$ 表示代表第 $t+1$ 个单词（待预测单词）的目标向量，即 $w(t+1)$ 在词表中的标号的那一维为 1，其它维数都为 0；$y(t)$ 为输出层的输出，可以得到公式2–14：

$$e_o(t) = d(t) - y(t) \tag{2–14}$$

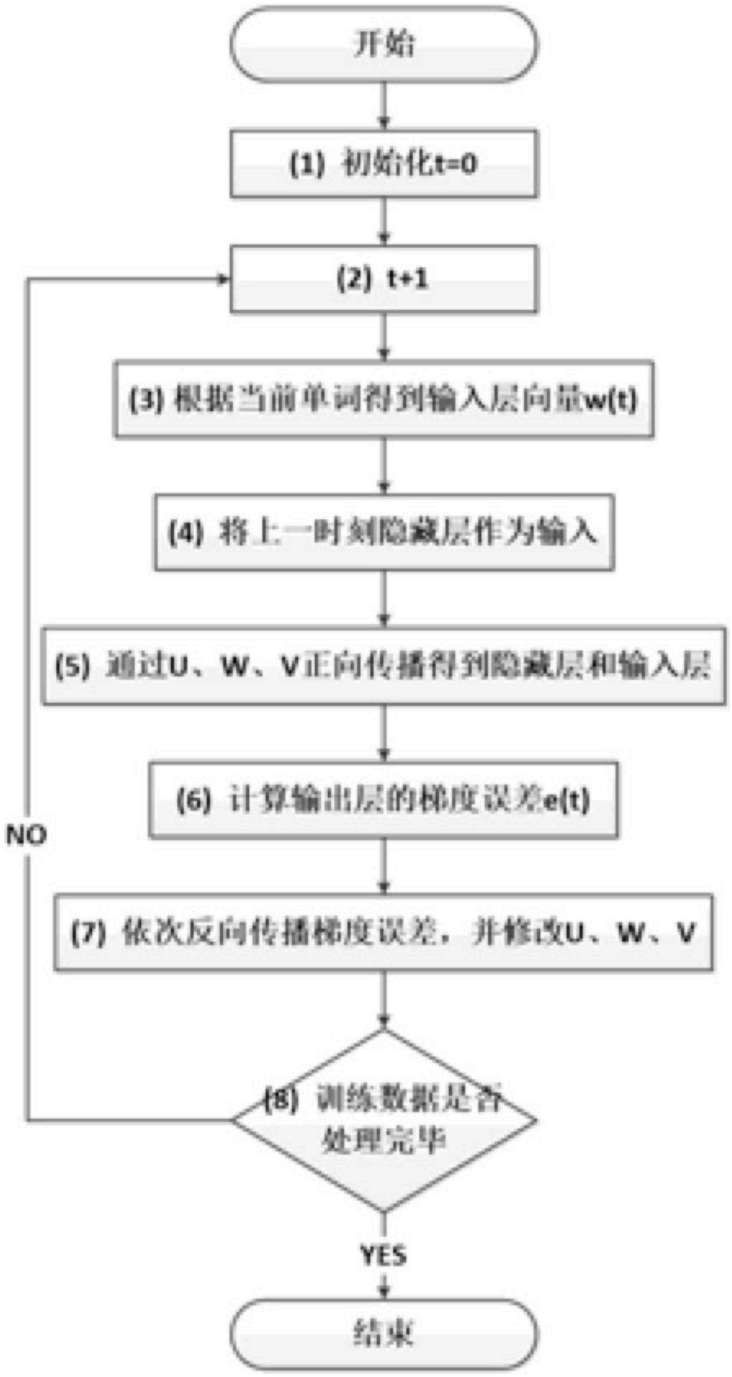


图 2–8 RNNLM training steps

Fig 2–8 RNNLM 训练步骤

因为非常碰巧的损失函数对输出层的偏导数的表现形式为期待的正确向量与概率预测向量之间的差，所以也被称为梯度误差或 error。

接着分析输出层的梯度误差是如何传递的。如图2–9所示：

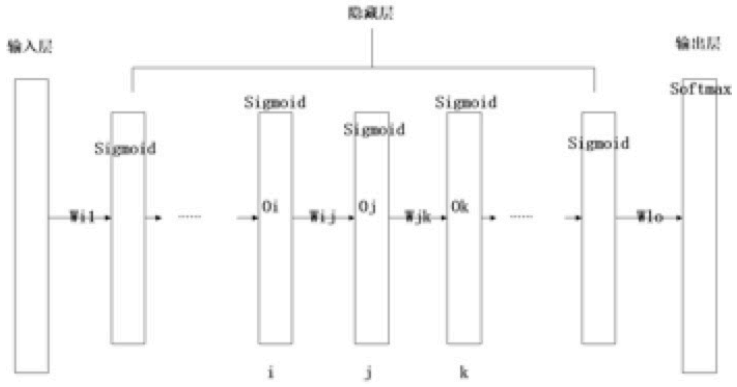


图 2–9 DNN back propagation

Fig 2–9 DNN 反向传播

在 DNN 中假设相邻的三个隐藏层的标号分别为 i、j、k（比 $l-1$，l，$l+1$ 表示更易于区分），o_i、o_j、o_k 分别是隐藏层 i、j、k 的输出（output）向量，w_ij 是 i 层到 j 层之间的矩阵，w_jk 是 j 层到 k 层之间的矩阵。正在推导公式之前先要介绍一下多元复合函数求导的链式法则，首先有个函数 $f(x_1,...,x_n)$ 并且对于每个 x_i 有 $x_i=g_i(u_1,...,u_m)$，并且所有函数都是可导的，则有公式**??**。

$$\frac{\partial f}{\partial u_i}=\sum_j \frac{\partial f}{\partial x_j}\cdot\frac{\partial g_j}{\partial u_i} \tag{2–15}$$

首先，我们求损失函数关于 w_ij 的偏导数，采用上述的链式法则，如公式**??**。

$$\begin{aligned}\frac{\partial L}{\partial w_{ij}}4&=\frac{\partial L}{\partial o_j}\cdot\frac{\partial o_j}{\partial w_{ij}}\\&=\frac{\partial L}{\partial o_j}\cdot o_j\cdot(1-o_j)\cdot o_i\\\because\quad o_j&=sigmoid(o_i\cdot w_ij+bias)\\\therefore\quad o_j{}'&=o_j\cdot(1-o_j)\end{aligned} \tag{2–16}$$

我们定义前面一部分为第 j 层的梯度误差，如公式**??**

$$err_j=\frac{\partial L}{\partial o_j}\cdot o_j\cdot(1-o_j) \tag{2–17}$$

然后可以求得 w_ij 的梯度为 err_j*o_i。

正向传播时我们是按箭头的方向，将和 o_i 和 w_ij 进行矩阵相乘后经过神经元的 sigmoid 函数，o_i 为作为第 j 层的输出，反向传播我们则反过来，现在求出 err_j 后，我们来看 err_j 和第 k 层的 err_k

有什么关系，继续用复合函数的链式法则，得出公式2–18。

$$\begin{aligned} err_j &= \frac{\partial L}{\partial o_j} \cdot o_j \cdot (1-o_j) \\ &= \sum_{k=1}^{hiddensize} \left(\frac{\partial L}{\partial o_k} \cdot \frac{\partial o_k}{\partial o_j} \right) \cdot o_j \cdot (1-o_j) \\ &= \sum_{k=1}^{hiddensize} \left(\frac{\partial L}{\partial o_k} \cdot o_k \cdot (1-o_k) \cdot w_{jk} \right) \cdot o_j \cdot (1-o_j) \\ &= \sum_{k=1}^{hiddensize} (err_k \cdot w_{jk}) \cdot o_j \cdot (1-o_j) \\ \because \quad r_k &= \frac{\partial L}{\partial o_k} \cdot o_k \cdot (1-o_k) \end{aligned} \tag{2–18}$$

结合 RNN 的结构图2–5，只有一个隐藏层，我们根据上述公式去更新矩阵 V、U、W。我们可以根据输出层的梯度误差得到隐藏层和输出层之间的矩阵 V 的更新方式，如公式2–19通过输出层的梯度误差对其进行更新。

$$v_{jk}(t+1) = v_{jk}(t) + s_j(t)\, e_{ok}(t)\, \alpha \tag{2–19}$$

其中 α 是学习率，学习率会随着训练的轮次一轮一轮递减。公式中 j 从 1 循环到隐藏层的大小，k 从 1 循环到输出层大小。$s_j(t)$ 是当前 t 时刻的隐藏层的第 j 个神经元的输出，$e_o k(t)$ 是当前时刻输出层第 k 个神经元的梯度误差。接下来我们用 L2 归一化方法得到公式2–20。

$$v_{jk}(t+1) = v_{jk}(t) + s_j(t)\, e_{ok}(t)\, \alpha - v_{jk}(t)\, \beta \tag{2–20}$$

其中 β 是归一化参数。归一化是为了防止过拟合而采用的，根据实验结果和前人论文经验，后面实验数据所采用的 β 都是 10^{-6}，公式2–20还能写成矩阵相乘的形式如公式2–21。

$$V(t+1) = V(t) + s(t)\, e_o(t)^T \alpha - V(t)\, \beta \tag{2–21}$$

隐藏层的梯度误差可以写成公式2–22的形式，其中表示矩阵乘法

$$\begin{aligned} e_h(t) &= d_h\left(e_o(t)^T V, t \right) \\ d_{hj}(x,t) &= x s_j(t)\,(1-s(j)) \end{aligned} \tag{2–22}$$

同样的根据上文推导出来的求梯度公式，我们得到了隐藏层的梯度误差 $e_h(t)$ 可以得出更新输入层和隐藏层之间的矩阵 U 的公式2–23,

$$u_{ij}(t+1) = u_{ij}(t) + w_i(t)\, e_{hj}(t)\, \alpha - u_{ij}(t)\, \beta \tag{2–23}$$

写成矩阵相乘的形式:

$$U(t+1) = U(t) + w(t)\, e_h(t)^T \alpha - U(t)\, \beta \tag{2–24}$$

需要注意的是，在当前 t 时刻，只有一个神经元是活跃的，当输入向量为 $w(t)$ 的时候，$w(t)$ 中为 0 的元素所连接的 w 都是不会改变的，因此在程序中实现更新的时候只用更新 $w(t)$ 为 1 的那个元素对应的一列 U 矩阵的权值，这样可以达到提高速度的目的。当前的 W 矩阵修改为2–24:

$$w_{lj}(t+1) = w_{lj}(t) + s_l(t-1)\, e_{hj}(t)\, \alpha - w_{lj}(t)\, \beta \tag{2–25}$$

写成矩阵相乘的形式为2–26

$$W(t+1) = W(t) + s_l(t-1)\, e_{hj}(t)^T \alpha - W(t)\, \beta \tag{2–26}$$

2.3.4.2 基于时间的反向传播算法 (BPTT)

上文所介绍的训练算法是普通的反向传播算法，一般用于 DNN 训练中，虽然 RNN 的训练和普通的只有一层隐藏层的前馈网络（包括 DNN）是一样的，但是唯一的不同是 RNN 结构中输入层不仅是当前单词的表示向量，还包括上一步或者说上一个时刻训练得到的隐藏层。

然而，可以发现这样的训练方法并不是最优的，因为神经网络试图根据当前的视神经网络状态和当前的单词去尽量正确的预测下一个单词，但是当前隐藏层储存的实际上有用信息太少了，因为只有一个隐藏层，就算有多个隐藏层，它也是有限的。所以如果这个神经网络可以记忆更多的前后文信息，并将它们储存在隐藏层中，预测的准确率就会高很多。

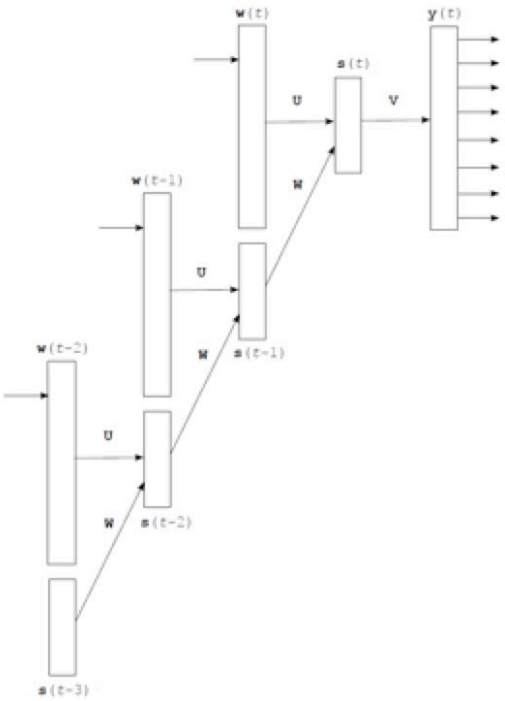


图 2–10 RNNLM BPTT structure

Fig 2–10 BPTT 结构图

因此我们保持前向传播的算法不变，在反向训练通过梯度误差对网络参数进行修改的时候，对其结构进行一些调整：我们将只有一个隐藏层的 RNN 神经网络经过 N 个时间、完成 N 各步骤的过程看成是一个有 N 个隐藏层的神经网络，因为每一个时刻的隐藏层的维度相同，并且连接他们之间的 W 矩阵也是完全一样的，参见上文 RNN 结构图，上一时刻的隐藏层会作为而这一时刻的输出，再往前推更多的时刻，看起来就是一个 DNN 的结构。

如图2–10中的神经网络的训练方式可以按照常规的梯度下降的方法，梯度误差会从当前隐藏层 $s(t)$ 依次传向 $s(t-1)$、$s(t-2)$、...、$s(1)$，并依次更新当前误差所在层和前一次之间的权值矩阵（记作 W），梯度误差的向前传递是一个递归的过程，写成如公式2–27的形式

$$e_h\left(t-\tau-1\right)=d_h\left(e_h(t-\tau)^T W, t-\tau-1\right) \tag{2–27}$$

其中函数在公式2–19中定义了。

但是这样的展开是无穷无尽的，前面出现过多少训练数据当前的 BP 就能向前展开多少层，但是梯度误差会随着传播的过程变得越来越小直到趋近于 0，但是展开的层数越多所耗费的时间复杂度和空间复杂度就越大，因为耗费巨大时空代价去对 W 进行接近于 0 的修改是没有必要的，因此我们采取一种截断的 BPTT，即向前展开有限层的 BPTT，实验发现向前展开 5 层左右是比较好的。这就类似 N-gram 的语言模型算法，理论上要往前推到头，但是映射等价类的做法就好比这里的截断操作，我们在 N-gram 算法中常用到 4-gram。结合图2–10，U 矩阵在 BPTT 中的学习公式为2–28：

$$u_{ij}\left(t+1\right)=u_{ij}\left(t\right)+\sum_{z=0}^{T} w_i\left(t-z\right)e_{hj}\left(t-z\right)\alpha-u_{ij}\left(t\right)\beta \tag{2–28}$$

其中 T 是这个神经网络 BPTT 时向前展开的总层数，同样的，它也可以写成矩阵相乘的形式2–29

$$U\left(t+1\right)=U\left(t\right)+\sum_{z=0}^{T} w\left(t-z\right)e_h(t-z)^T\alpha-U\left(t\right)\beta \tag{2–29}$$

十分需要注意的是，矩阵 U 的更新是一次性完成的，并不随着梯度误差的反向传播而递增，那样会导致训练的不稳定性。同样的，矩阵 W 的更新方式和 U 类似

$$w_{lj}\left(t+1\right)=w_{lj}\left(t\right)+\sum_{z=0}^{T} s_l\left(t-z-1\right)e_{hj}\left(t-z\right)\alpha-u_{lj}\left(t\right)\beta \tag{2–30}$$

写成矩阵相乘为2–31

$$W\left(t+1\right)=W\left(t\right)+\sum_{z=0}^{T} s\left(t-z-1\right)e_h(t-z)^T\alpha-W\left(t\right)\beta \tag{2–31}$$

通过上述公式，阐述了 RNNLM 在进行 BPTT 的时候的理论算法。

2.4 长短时间记忆神经网络语言模型 LSTMLM

LSTM 是循环神经网络的一种，它于 1997 年被 Sepp Hochreiter and Jürgen Schmidhuber 提出 citehochreiter1997long，和大部分的 RNN 一样，一个 LSTM 神经网络在给定了足够的网络单元时可以

完成一台常见计算机可以完成的任何计算。但是不同于传统的 RNN，LSTM 更适合用于学习那些有很长一段时间滞后并且滞后时间的长短未知的重要事件，这也是为什么 LSMT 在有的方面比 RNN，N-gram 等语言模型算法更优秀的原因。例如，LSTM 在不分段的连写识别中取得了最好的成绩[20]，另外，LSTM 神经网络也在自动语音识别中取得很好的成绩[21]。

训练一个 LSTM 神经网络同样用的是迭代梯度下降算法和基于时间的反向传播算法，然而在 1991 年突然意识到随着重要事件之间的间隔时间，误差梯度指数会迅速消失[22][23]，然而，对于每一个 LSTM 块，当输出层的误差被计算算出来时，这个误差会被困在当前的记忆块中，这个记忆块会不断地传回错误直到切断这个值，这个 BPTT 的方法对于训练一个 LSTM 块去记住一个持续时间的值非常有效。

因为句子所具有的时序性，循环神经网络语言模型（RNNLM）作为一种有隐层自环的时序性神经网络模型，能够保存有历史信息并对其加以利用，打破了传统统计学语言模型的统治地位，并在短时间内发展迅速。其中，更有长短时间记忆（LSTM）单元替代传统的 RNNLM 中的隐层单元，以达到对更长历史信息的记忆和利用。其结构中有很多的门来控制对历史信息的记忆，公式如下：

在训练语言模型时，求损失函数对每个参数的偏导数即可得到梯度，接着利用梯度和步长（学习率）对模型参数进行更新。

2.4.1 LSTM 单元及其数学原理

2.4.2 LSTM 的优劣

第三章 面向结构化语言模型的学习率自适应研究

3.1 研究动机和研究意义

前一章介绍了循环神经网络（RNN）和加入长短时间记忆单元（LSTM）语言模型。

在神经网络的训练过程中，学习率是一个重要并且敏感的参数，需要用到先验知识和进行大量实验去找到符合特定模型的最优学习率才能得到最优解。在过去的研究中，在[24] 中提出了在 DNN 模型中采用学习率的自适应算法，能够降低模型对初始学习率的敏感度和提升收敛速度。在此基础上，这个算法被应用到 LSTM 循环神经网络声学模型中[25]，虽然可以一定程度上降低对初始学习率的敏感度和收敛速度，却会牺牲掉一部分性能，效果并不理想。

在语言模型中引入稳定算子 beta 进行初始学习率的自适应能够很大幅度忽略初始学习率对结果的影响；同时，稳定算子能够较大幅度提升语言模型的收敛速度，有效减少不必要的训练时间；在多种 LSTM 语言模型引入稳定算子的算法中，每个隐层的参数分别拥有自己独立的稳定算子会使结果更优。

尤其是在本文的主要工作——结构化语言模型的研究中，不同的结构的特点和任务不一样，会有主次之分，也会有部分经过预训练的情况。用相同的学习率往往会使结果达不到预期。因此动态的调整不同结构中的学习率至关重要。本人也对传统 DNN 上的 β 稳定算子方法做了相应的优化，将其在 LSTM 语言模型中进行实现、测试。最后证实有效并应用于结构化语言模型的研究。

在此部分的工作中，我们将学习率的自适应算法应用在语言模型中，后文统一称为引入稳定算子 beta，在训练中动态调整 beta 以达到自适应学习率的目的。我们在不同的语言模型网络结构中尝试加入稳定算子，探究其对训练结果、收敛速度、初始学习率的敏感程度上的影响。同时，我们参考各机器学习工具的 RNN 实现中，往往会通过对参数加入 L2 范数优化以达到避免过拟合、提升模型性能的目的，对稳定算子加入 L2 范数，研究优化后的模型性能。

3.2 随机梯度下降法和学习率

神经网络模型的训练开始用的是批梯度下降这种最优化求解方法。因为训练速度过慢而采用随机梯度下降法（SGD)[26]。随机梯度下降（SGD）是一种常用的最优化问题求解方法，然而该方法依然存在迭代次数过多以及搜索盲目的问题。而小批量随机梯度下降（Mini-batch SGD，MBGD）能够折衷训练时间和训练效果，更是广泛地被用于各类神经网络的训练中[27]。

下面我们介绍一下小批量随机梯度下降的数学原理和理论过程。训练目标是更新神经网络中的参数 θ，首先根据损失函数 L 计算出每个参数 θ_i 对应的梯度为公式3–1，

$$\Delta_i = \frac{\partial L}{\partial \theta_i} \tag{3–1}$$

根据梯度和学习率对该参数更具公式3–2进行更新

$$\theta_i = \theta_i - \eta \Delta_i \tag{3–2}$$

其中 η 是学习率，即通俗意义上的步长。

学习率 α 在训练过程中需要根据某个训练策略进行调整。在我们的实验中，一般都会使用一个开发集数据 (development set) 来观察训练结果。接着，一开始我们会使用较大的学习率，然后如果在开发集上经过一轮训练后性能不再提升，我们就会开始进行改变学习率。一个需要调整的配置是 mini-batch 的大小，mini-batch 大小大的话，因为并行度的提高，训练速度会提高，但是更新次数少了的话可能导致训练收敛减慢。另外，如果把 mini-batch 设小了，因为并行度的减小，训练速度变慢，这对我们用户来说也是不太可以接受的，总的来说，需要选择一个适中的值。一般来说，一个大小为 10 到 50 的 mini-batch 的大小都会在实践中使用。

实际训练中，学习率会随着训练的进行变化，调整学习率的方法有好几种，目前用的比较多的两种是提前停止法[28] 和减半法[29]。提前停止法是当某一轮监督集上的结果变差了即停止训练。减半法是当某一轮的监督集上的结果变差时将学习率减半，继续进行训练。

但无论是哪一种调整学习率的方法，都对初始学习率的值非常敏感，学习率太大会导致无法收敛，学习太小同样会导致无法收敛或者收敛速度过慢。并且不同的初始学习率可能会对总的学习轮数影响较大，不同任务的合理的初始学习率也各不相同，为了得到合理的初始学习率，需要做大量的试验。因此为了减少甚至消除初始学习率对试验结果的影响，弱化掉选取初始学习率这一过程，关于在训练算法中加入稳定算子来进行学习率的自适应的算法应运而生。本文着重研究在 LSTM 语言模型中加入稳定算子的影响，并对其进行 L2 范数以达到优化目的。

3.3 稳定算子 Beta 及自适应

在微软的研究中，提出引入稳定算子进行模型学习率的自适应，使得神经网络对于学习率的敏感度降低，将研究人员从调整学习率的繁琐重复工作中解放出来。具体方法是为 DNN 神经网络中每一层配备一个稳定算子 beta。普通的 DNN 隐层的矩阵计算公式为3–3

$$y = Wx + b \tag{3–3}$$

其中 x 是输入向量，y 是输出向量，W 是现行变化参数矩阵，b 是偏移系数向量。加入 beta 稳定算子后的矩阵公式为公式**??**，

$$y = e^{\beta} Wx + b \tag{3–4}$$

e 是自然对数，是稳定算子。加入 β 后的模型训练过程中，前向传播可以直接按照上述公式进行，但是在反向传播训练的过程中，需要计算 x，W，b 和 β 的梯度。其中 x 和 W 的梯度进行了小幅度修改，b 的梯度没有改变，如公式3–5所示

$$\begin{aligned} \frac{\partial L}{\partial x} &= e^{\beta} W^{\mathrm{T}} \frac{\partial L}{\partial y}, \\ \frac{\partial L}{\partial W} &= e^{\beta} \frac{\partial L}{\partial y} x^{\mathrm{T}}, \\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial y} \end{aligned} \tag{3–5}$$

剩下的问题是如何更新 β 呢，根据求导的链式法则有公式3–6

$$\frac{\partial L}{\partial x}=\frac{\partial L}{\partial y}\frac{\partial y}{\partial \beta}=e^{\beta}\frac{\partial L^{\mathrm{T}}}{\partial y}Wx \tag{3–6}$$

根据上面求 x 的梯度公式，可以得到公式3–7

$$\frac{\partial L}{\partial \beta}=\frac{\partial L^{\mathrm{T}}}{\partial x}x \tag{3–7}$$

故有公式3–8

$$\beta=\beta-\eta\frac{\partial L^{\mathrm{T}}}{\partial x}x \tag{3–8}$$

3.4 LSTM 语言模型引入稳定算子

LSTM 是一种特殊的 RNN 网络，将原始的 RNN 网络的隐层单元替换为 LSTM 记忆单元，它不仅保留着 RNN 的记忆历史的功能，还能在记忆门的作用下动态调整记忆长短，如公式3–9所示。

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t+W_{hi}h_{t-1}+W_{ci}c_{t-1}+b_i)\\
f_t &= \sigma(W_{xf}x_t+W_{hf}h_{t-1}+W_{cf}c_{t-1}+b_f)\\
c_t &= f_t c_{t-1}+i_t\tanh(W_{xc}x_t+W_{hc}h_{t-1}+b_c)\\
o_t &= \sigma(W_{xo}x_t+W_{ho}h_{t-1}+W_{co}c_t+b_o)\\
h_t &= o_t\tanh(c_t)
\end{aligned} \tag{3–9}$$

由微软提出的 DNN 网络中的学习率自适应方法适用于普通深度神经网络中的现行矩阵变换，然而无法直接应用于在 LSTM 网络中，因为在每个 LSTM 单元中，分别有三个门和一个仿射变换操作。因此在刘奇的研究中，主要比较了三种不同的方法：每层共享一个 β，相同种类的门共享一个 β，以及每个门的 beta 都各不相同。其中第三种方式中，每个 β 能够单独地分别调整每个参数矩阵，因而表现最好。该方法公式的修改如公式3–10：

$$\begin{aligned}
\mathrm{i}_t &= \sigma(\mathrm{e}^{\beta_{xi}}\mathrm{W}_{xi}\mathrm{x}_t+\mathrm{e}^{\beta_{hi}}\mathrm{W}_{hi}\mathrm{h}_{t-1}+\mathrm{e}^{\beta_{ci}}\mathrm{W}_{ci}\mathrm{c}_{t-1}+\mathrm{b}_i)\\
\mathrm{f}_t &= \sigma(\mathrm{e}^{\beta_{xf}}\mathrm{W}_{xf}\mathrm{x}_t+\mathrm{e}^{\beta_{hf}}\mathrm{W}_{hf}\mathrm{h}_{t-1}+\mathrm{e}^{\beta_{cf}}\mathrm{W}_{cf}\mathrm{c}_{t-1}+\mathrm{b}_f)\\
\mathrm{c}_t &= \mathrm{f}_t\cdot\mathrm{c}_{t-1}+\mathrm{i}_t\cdot\tanh(\mathrm{e}^{\beta_{xc}}\mathrm{W}_{xc}\mathrm{x}_t+\mathrm{e}^{\beta_{hc}}\mathrm{W}_{hc}\mathrm{h}_{t-1}+\mathrm{b}_c)\\
\mathrm{o}_t &= \sigma(\mathrm{e}^{\beta_{xo}}\mathrm{W}_{xo}\mathrm{x}_t+\mathrm{e}^{\beta_{ho}}\mathrm{W}_{ho}\mathrm{h}_{t-1}+\mathrm{e}^{\beta_{co}}\mathrm{W}_{co}\mathrm{c}_t+\mathrm{b}_o)\\
\mathrm{h}_t &= \mathrm{o}_t\cdot\tanh(\mathrm{c}_t)
\end{aligned} \tag{3–10}$$

3.5 稳定算子 beta 的 L2norm 范数

在语言模型中，原始的损失函数为公式3–11，

$$L(\theta)=-\sum_{i=1}^{T}\log\left(P\left(s=t_i|o_i\right)\right) \tag{3–11}$$

目前在各机器学习框架的实现过程中，会对所有参数矩阵加入 L2 正则项优化，防止过拟合，使训练结果更好，加入 L2 优化后的损失函数为公式3–12，

$$L(\theta) = -\sum_{i=1}^{T} \log\left(P\left(s = t_i | o_i\right)\right) + \frac{\lambda}{2}||W||_2 \tag{3–12}$$

如前文所述，在前人工作中，稳定算子 beta 作为参数被加入到 LSTM 语言模型的训练中，并不改变损失函数。只是在更新 beta 的时候，把 beta 当做和其它参数一样求偏导数得到梯度，在学习率的控制下进行更新。本文在损失函数中为 beta 加上 L2 范数项，以达到优化的目的。加入稳定算子 beta 的 L2 正则项后的损失函数为3–13。

$$L_2(\theta) = L(\theta) + \frac{\lambda_1}{2}||W||_2 + \frac{\lambda_2}{2}||e^{\beta}||_2 \tag{3–13}$$

这也是本文采用的方法。在加入稳定算子进行自适应的 LSTM 语言模型的训练过程中，别的参数的梯度求解方法和前面类似，包括 beta 在内的所有参数的更新方法，都是按照新的公式求解梯度，结合稳定算子和学习率进行更新。

关于此任务的所有实验和结果分析将会在第 4 章实验部分给出。最后得到的结论是稳定算子的引入对于普通语言模型和多任务语言模型来看在语言模型的性能上没有明显区别，并不像实验前预期的那样或许能智能调整不同任务的学习速度从而提升最后的训练结果，根据分析发现，这是因为多任务模型中为不同的任务分配权重已经可以达到不同任务学习速度不同的要求。另一方面，在参数量合理、没有超出训练数据的收敛极限的情况下，越深的网络，引入稳定算子的结果越好。引入稳定算子的同时，在训练过程中对其加入 L2 正则优化会略微提升语言模型的性能，提升幅度大概 1

第四章　面向 ASR 的结构化单向辅助信息 Multi-view 语言模型

4.1　现有结构化语言模型研究及现状

4–1 展示了三种基础的在 LSTM 语言模型上的扩展结构，这三种结构正在被广泛使用，也会作为本文工作的基础工作和实验基线。可以初步看出 Multi-task、Multi-view 和 Joint train 模型的结构特征和对比的区别。Multi-task 是有多个输出和训练准则，同样有多个训练标注。Multi-view 则是有多个输入，直观理解是训练一个目标模型，有多个辅助特征的输入。而 Joitn train 则是前两者进行一定程度的结合，偶尔会有一些时序上的错位，比如上一个时刻的第二个任务的的输出作为当前时刻的输入等等。

模型细节和优劣将在后文分别进行描述，实验结果和比较，以及分析将会在第五章呈现。如上文介绍所说，为了进一步提升语言模型的性能，研究者们在语言模型的结构化研究中做了很多工作。其中就有在训练模型的过程中加入额外信息的多视角学习，通过加入词层面的辅助信息和当前词一起训练以提升语言模型的性能[30]. 额外的辅助信息包括词性标注（part of speech,POS），命名实体识别（named entity recognition,NER），语法块（chunking)[31]，句子的环境信息，语法解析信息等。这些辅助信息被合并到 Multi-view 模型中，甚至可以作为联合模型逐帧一起训练[32]. 但是虽然这些工作在混淆度的指标上有非常大的提升，但是在语音识别方向的重打分任务的词错误率（WER）和句子错误率（SER）中却依然没有提升，甚至有的还有所下降[30][32]。

4.1.1　multi-task 语言模型

随着 LSTM RNN 模型被证实是一效果很好的模型，很多的研究工作便在此模型的基础上展开，语言模型也不例外。为了获得更进一步的提升，更多有效的复杂的网络结构也应运而生，比如说本章要讨论的结构化语言模型的研究。

4.1.2　Multi-task 语言模型

正如图4–1(b) 所展示的, 在 Multi-task 模型结构中, 语言模型被设计成和其它的模型一起训练。它们共享输入和一部分的隐层[33],

However most researches on multi-task structure show that usually performance gain is achieved in the cooperating task, instead of LM task itself.

4.1.3　multi-task 模型结构

4.1.4　multi-task 语言模型的表现

4.1.5　Multi-view 语言模型

Considering that some extra linguistic features might contribute to language modeling, word and sentence level features were introduced in LM[30]. This model have multi inputs, which contains different views

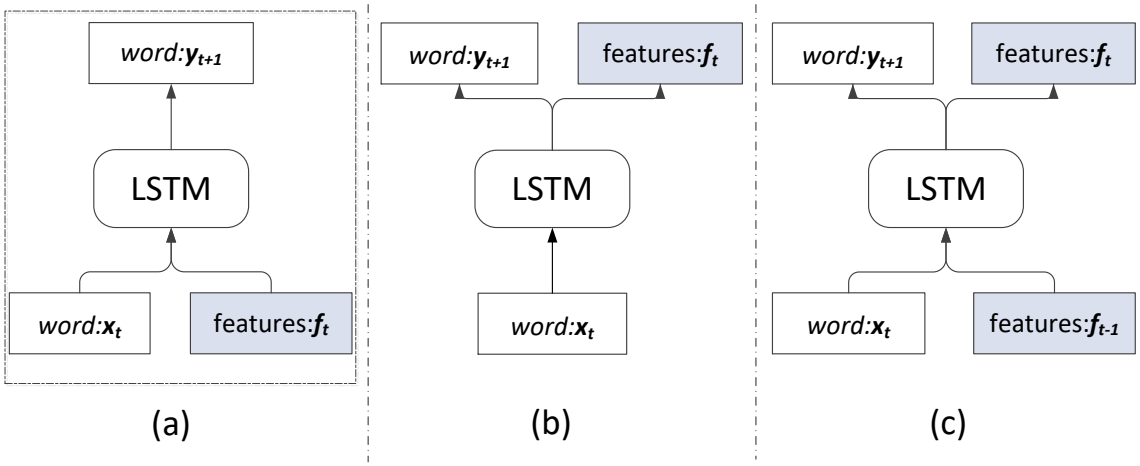


图 4–1 *(a)Multi-view LSTM* 语言模型. *(b)Multi-task LSTM* 语言模型. *(c)Joint train LSTM* 语言模型.

of information. So it is called multi-view model (see 4–1(a)).

As argued in Section 1, research on this kind of model shows improvements on perplexity and word prediction accuracy (WPA), but integrating this model with ASR did not lead to commensurate improvements[32]. That is to say, the straightforward combination of words and features as the inputs of a language model do not contribute to speech recognition.

Our proposed model is based on the multi-view structure, but is specially tailored for ASR task by using word-synchronized auxiliary feature.

4.1.5.1 multi-view 语言模型的结构

4.1.5.2 multi-view 的多视角特征信息介绍

4.1.5.3 multi-view 语言模的表现

4.1.6 Joint-train 语言模型

Other works combined the multi-task structure with multi-view structure, which is shown in Figure 4–1(c). Not only multiple tasks were trained together, but also the inputs of this model were multi-view. LM was jointed with other spoken language understanding (SLU) or natural language process (NLP) task, some models of improved version are researched. Better than multi-task models, these works show slight improvement in PPL and ASR-rescoring, but more promotion is gained in the cooperating task[Liu2016Joint].

4.1.6.1 Joint-train 语言模型结构

4.1.7 研究动机和思路

我们猜测造成上述现象的主要原因是因为那些额外的辅助信息用的都是标准的标注算法，比如最大熵算法和双向 LSTM 模型，这些模型虽然表现优秀，但是在标注过程中用到了全局信息。也就

是不仅用到了前文的信息，更用到了后文的信息。这就意味着在训练语言模型的时候，给当前词的辅助信息中包含后文信息，然后语言模型再用后文信息去预测后面的词。有点类似于信息作弊的感觉。这也是为什么在 PPL 任务上表现优秀的原因。

那为什么又在重打分任务中没有提升呢，这可能是因为重打分任务中的 n-best 集合（在第四章实验部分详细介绍）中的各种句子是本身就不合理的（语言模型得分很低），用不合理的后文信息得到的标注也会存在问题。也就是意味着用不合理的后文信息预测后面的词，自然会出现偏差。

为了验证和解决这个问题，为了使得结构化语言模型不仅提升了语言模型的某个指标（PPL），更是要使得它在实际应用中（ASR 任务）中有所提升，我们提出仅仅使用单向的辅助信息去训练结构化的语言模型。

基于这个思想，我们进行了一系列研究工作。在我们的工作中，我们使用一个单向 LSTM 标注模型去进行双向信息的单向化。保证从词模型中出来的标注信息仅仅会包含历史信息，而不包含未来信息。紧接着，标注模型的输入被作为 Multi-view 语言模型的一个输入接入到一个 LSTM 语言模型结构中。在这种模型结构下，存在不同的训练方法，我们总共尝试了五种不同的训练方法。最后我们将我们提出的模型和前人的相似结构模型进行多方位的在 PPL 和 ASR 任务上的比较。

4.2　单向辅助信息模型的辅助信息选取

4.2.1　词性标注

词性是用来表示一个词的性质的，例如名词、形容词、动词等，每个词都有词性，有的甚至不止一种词性，根据在句子中所扮演的成分不同而有区别。

词性标注（Part-Of-Speech Tagger，POS Tagger）为以前无限的语言模型提供了有限的句法信息，例如限定词往往紧随其后的是名词。此外，对语言模型添加词性标记可以被看作是一种平滑，对于那种在训练数据中出现概率很小的单词，加入词性的语言模型相当于给他们分配了更一般的抽象类，从而增大他们的概率使其不再接近于 0，从而可以更好的处理它们 [20]。词性标注是自然语言处理（NLP，Natural Language Processing）的范畴，

词性标注任务研究如何判别一个句子中的每个词的词性类别，如名词、动词、形容词等。词性是最基础也最常用的一类特征。在大量涉及自然语言的应用，如语音合成、句法分析、机器翻译中，词性都是必不可少的输入特征，其识别结果的正确性对这些系统的最终表现也有着显著的影响。词性标注是一个典型的标注任务，它的输入为一串词序列，每个词所对应的词性即为输出，如图**??**所示，

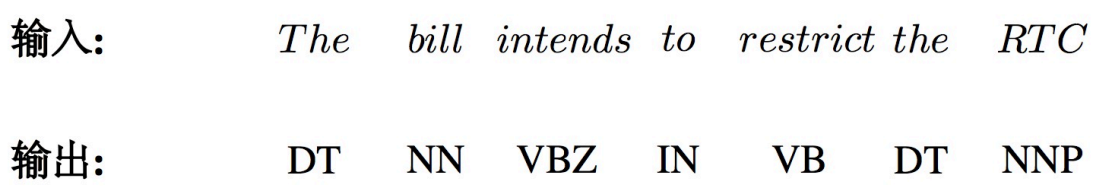


图 4–2 词性标注任务输入输出示例。输入为词序列（即句子），输出为该句子的词性标注序列。

本文采用的词性标注工具是斯坦福大学的开源词性标注工具，具有各种语言之分。输入到词性标注工具中的文本是已经经过分词处理的中文文本，若是英文文本可以直接进行词性标记，没有分词这个步骤。每个词的词性并不是固定的，例如“游泳”既可以是动词，也可以是名词，具体标记的算法由词性标记工具实现。

获得词性标注之后，我们用处理词向量的方式来处理词性标注。同样我们建立一个词性的词表，对于每一个词性，将其映射到词表上，对应的位置为 1，其它为 0，最后得到的就是一个一维的长度为词性的个数，对应位置为 1 其它为 0 的向量，如公式

$$pos_t(i) = \begin{cases} 1(wp_t = vp_i) \\ 0(wp_t \neq vp_i) \end{cases} \tag{4–1}$$

其中 pos_t 为当前 t 时刻的单词对应的词性所表示的向量，i 表示向量的第 i 维，wp_t 表示当前的单词所对应的词性，vp_i 表示词性的词表中第 i 个词性。它即是词性的特征向量。

4.2.2 命名实体识别

命名实体识别（Named Entity Recognition，简称 NER），又称作“专名识别”，是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。命名实体识别也是信息抽取（Information Extraction）相关任务的核心技术，因此有着很高的研究价值。通常包括两部分：（1）实体边界识别；（2）确定实体类别（人名、地名、机构名或其他）。英语中的命名实体具有比较明显的形式标志（即实体中的每个词的第一个字母要大写），所以实体边界识别相对容易，任务的重点是确定实体的类别。和英语相比，汉语命名实体识别任务更加复杂，而且相对于实体类别标注子任务，实体边界的识别更加困难。

与语块切分类似，命名实体识别也是一个切分任务，同样可以使用 IOBES 的标注策略转化为标准标注任务，输入输出样例如图4–3所示。

Japan	*began*	*the*	*defence*	*of*	*their*	*Asian*	*Cup*
S-LOC	O	O	O	O	O	B-MISC	E-MISC

图 4–3 命名实体识别任务输入输出示例。输入为词序列，输出为该句子的命名实体识别标签序列（使用了 IOBES 标注策略）。

4.2.3 语法块标注

语块切分任务也称为浅层句法分析（shallow parsing），它研究如何把句子切分成不同的句法模块，如名词词组（Noun Phrase，NP）、动词词组（Verb Phrase，VP）、介词词组（Preposition Phrase，PP）等。与词性标注类似，语块切分同样也是一种常用且重要的特征，对于句法分析、语义理解任

Scheme	Begin	Inside	End	Single	Other
IOB	B-X	I-X	I-X	B-X	O
IOE	I-X	I-X	E-X	E-X	O
IOBES	B-X	I-X	E-X	S-X	O

表 4–1 **IOB**，**IOE**，**IOBES** 标注策略比较。

务尤其重要。与词性标注不同，语块切分是一个分割（segmentation）任务。分割任务需要明确指定当前词所属类别的边界，边界内的词都属于同一类别，简单套用本章开头定义的标注模型不能确保上述限制条件。不过通过使用特定的标注策略如 IOBES、IOB、IOE 等，语块切分任务也可以转成一个标准的标注任务。

以 IOBES 标注策略为例，该标注策略将每个原有的类别 X 扩展成四个子类别 B-X，I-X，E-X 与 S-X，每个子类分别表示当前词属于 X 类且处于分割的开始位置（**B**egin），中间位置（**I**ntermediate），结尾（**E**nd）与单词分割（**S**ingle）。此外，该策略还额外引入标签 O 表示该标签不属于任何分割（**O**ther）。使用该标注策略的语块切分系统，首先预测每个词的 IOBES 扩展后的子类别，然后以 B，E，S,O 为边界将输出切分成段。使用 IOBES 策略的输出与切分结果示例如图4–4所示。

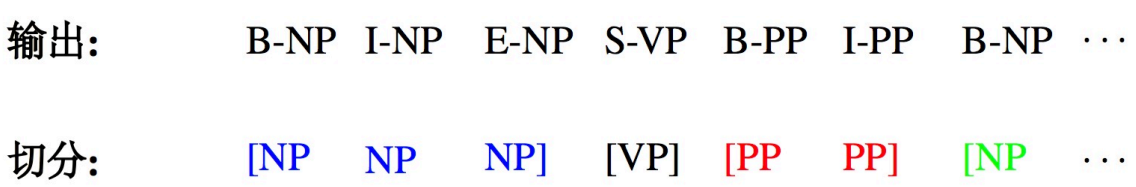


图 4–4 **IOBES** 标注策略输出与切分示例。

IOB，IOE 与 IOBES 类似，只是更加简化只有三个子类。表4–1展示比较了上述三类标注策略。在实际应用中，这三种标注方式都有工作使用，关于他们的优劣目前也无定论。本文选择使用 IOBES 策略。关于语块切分的输入输出样例如图4–5所示。

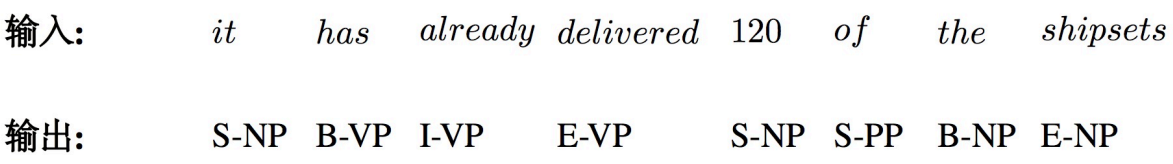


图 4–5 语块切分任务输入输出示例。输入为词序列，输出为该句子的语块切分标签序列（使用了 IOBES 标注策略）。

4.3 单向辅助信息的 Multi-view 语言模型结构

我们的模型总共由两个部分组成: 第一部分是用来生成单向辅助信息特征的标注模型; 另一个是一个和标注模型链接在一起的 Multi-view 语言模型。对于这两个模型, 我们都是用的单层单向 LSTM 模型，并且将这两个模型以此层面的粒度链接在一起。关于模型的详细介绍和数学基础将在下文给出。

4.3.1 单向 LSTM 标注模型部分

标注模型是一种分类模型，一个标注模型是通过训练得到为每个输入序列找到它所属类别并输出。神经网络现在被广泛用于各种标注模型中[34]，因为它能大幅图提升传统统计学标注模型的性能。其中双向神经网络更为普遍[35]，由于前后文信息的紧密结合，更是使得标注模型进一步提升。神经网络的标注模型是根据输入输出它所属的类别的概率分布。

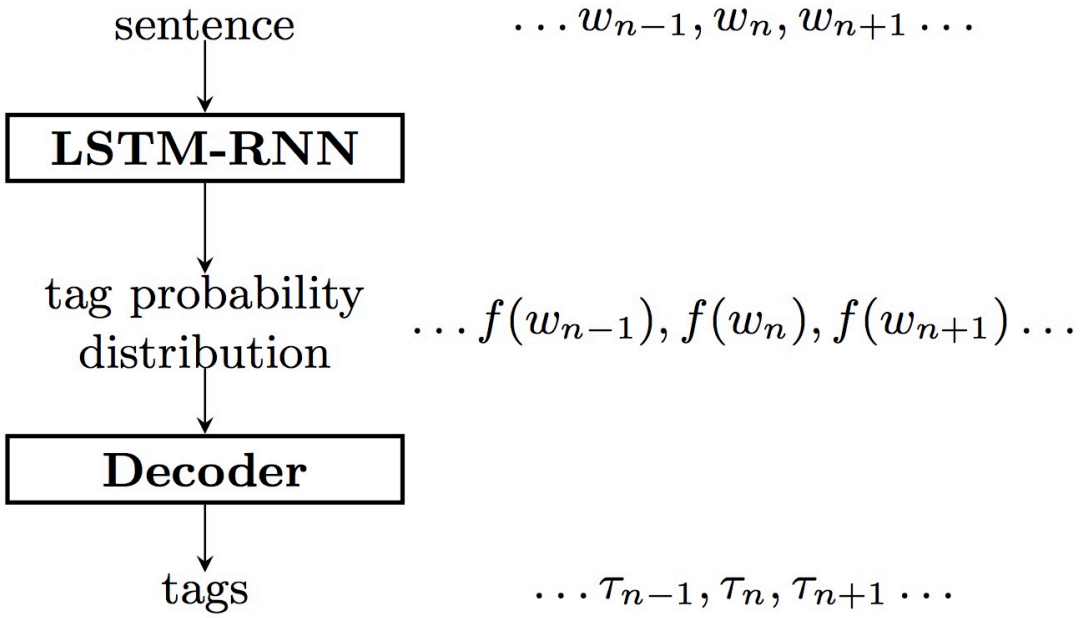


图 4–6 *uni-directional LSTM tagging model*

不过如前文所说，本文这里必须使用单向神经网络。另外在我们的验证实验和我们的三种标注任务中, 双向 LSTM 模型相比于相同参数的单向 LSTM 模型仅仅有非常微小的性能提升。如图4–6所示，这是一个单向模型的模型结构示意图，一个 LSTM 网络结构就和 RNN 一样有隐层的递归的自环，其中每个隐层单元被替换成了具有长短时记忆功能的 LSTM 单元。在后文中，LSTM 记忆单元被统一表示成 $\mathcal{L}$。另外为了避免混淆，标注模型和语言模型的 LSTM 记忆单元分别被表示成 $\mathcal{L}_{\text{tag}}$ 和 $\mathcal{L}_{\text{LM}}$。

向量 $\boldsymbol{w}_t$ 使用 one hot 编码来表示当前时刻 t 的当前词，这个 one hot 向量也同时是 $\mathcal{L}_{\texttt{tag}}$ 和 $\mathcal{L}_{\texttt{LM}}$.

接下来，这个词的此嵌套 $\boldsymbol{x}_t$ 可以这样得到:

$$\boldsymbol{x}_t = E_{tag}\boldsymbol{w}_t \tag{4–2}$$

这里 E_{tag} 是指标注模型的词嵌套矩阵。

单向 LSTM 标注模型的输出层 $\boldsymbol{h}_t$ 由下述公式计算得到:

$$\boldsymbol{h}_t = \mathcal{L}_{\texttt{tag}}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{4–3}$$

其中 LSTM 计算单元 $\mathcal{L}$ 内部的详细的函数表达式如公式 refeq:memoryb 所展示的这样:

其中 σ 是逻辑函数 (sigmoid)，以及 i, f, o and c 分别表示 *input gate*、*forget gate*、*output gate* and *cell* 激励向量.

$\boldsymbol{f}_t$ 是标注模型的输出，这是个总和为 1 的向量, 表示输出的标注分类的概率分布，同样的它可以根据模型的隐层 LSTM 记忆单元求得，公式如下:

$$\boldsymbol{f}_t = softmax(W_{ho}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{4–4}$$

其中 softmax 是归一化函数，目的是让概率分布总和为 1.

根据截止到目前所描述的 LSTM-RNN 语言模型，观测到的每个时刻的模型的输出概率分布和其它时刻是相互独立的，仅仅根据数据训练得出。然而，在有些任务中，比如说 NER 和 Chunking 任务中，标注之间具有一些隐含的规则，它们之间和前后的标注具有强关联性。其中一部分种类仅仅能存在在部分特定的种类之后，而一部分不能存在于某些之后。比如说 NN-B（名字块开头）后面不可能跟 VV-E（动词块）结尾，加了类似于这些规则之后标注模型的准确率会有一定的提升。

为了将上述的标注之间的约束关系利用起来，我们在每一步中引入转化矩阵的概念，如果两个标注类别之间可以连接，则矩阵为 1，否则为 0。矩阵是单向的，比如 B 能在 A 后面出现，则 $matrix[A][B] = 0$，而 B 不一定能在 A 之前出现，若不能则 $matrix[B][A] = 0$。这个矩阵和标注模型的概率分布一起进行解码[35]，便能得到概率最大的标注序列。这个解码过程可以用经典的 Viterbi 算法**Andrew1967Viterbi** 完成。

在本文中，解码过程被表示成 $\mathcal{D}(\cdot)$，解码过程的输出 τ_t 是一系列最终预测得到的标注序列，它们同样用 one hot 向量表示如下:

$$\boldsymbol{\tau}_t = \mathcal{D}(\boldsymbol{f}_t) \tag{4–5}$$

到此我们可以发现，标注模型的输出传到语言模型中可以有两种表示：第一种是概率分布的序列，第二种是经过 Viterbi 解码后得到的确切的 one hot 序列。

4.3.2 Multi-view 语言模型部分

图4–7 展现了我们的仅用前文辅助信息的 Multi-view LSTM 语言模型架构。

我们提出的模型是一个和单向标注模型连接起来的 Multi-view 语言模型。

语言模型的第一个输入 $\boldsymbol{w}_t$ 是一个代表当前词的 one hot 向量, 这个向量同时也是标注模型的输入。第二个输入是标注模型部分的输出，可以是经过 Viterbi 解码之后的 ont 序列，也可以是直接的

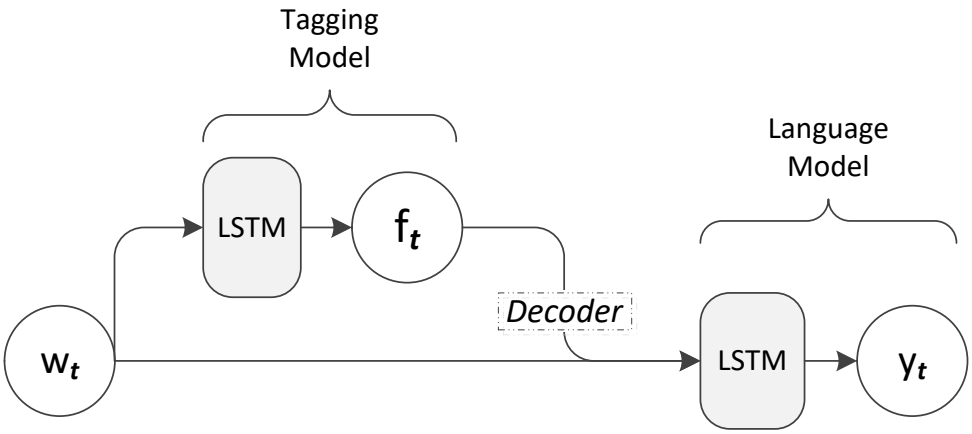


图 4–7 *multi-view LSTM language model with word-synchronized auxiliary feature*

概率分布。无论这个解码过程有没有进行，我都对于两者都在第四章的实验部分进行了实验与比较。它们的不同点也体现在输入部分的公式上，如果使用了 Viterbi 解码，语言模型的第二个输入将会是：

$$\boldsymbol{\zeta}_t = W_{tag}\boldsymbol{\tau}_t + E_{word}\boldsymbol{w}_t \tag{4–6}$$

其中 $\boldsymbol{\tau}_t$ 是解码部分输出的 one hot 向量。否则，语言模型的输入是:

$$\boldsymbol{\zeta}_t = W_{tag}\boldsymbol{f}_t + E_{word}\boldsymbol{w}_t \tag{4–7}$$

其中 $\boldsymbol{f}_t$ 是标注模型的概率分布形式的输出。E_{word} is 是词嵌套矩阵，W_{tag} 是标注模型到语言模型的隐层之间的参数矩阵。它们将会在每个时序点被加起来作为语言模型最终的输入。

LSTM 语言模型的隐层的输出为 $\boldsymbol{h}_t$，它的计算公式是：

$$\boldsymbol{h}_t = \mathcal{L}_{\text{LM}}(\boldsymbol{\zeta}_t, \boldsymbol{h}_{t-1}) \tag{4–8}$$

LSTM 记忆快的具体函数 $\mathcal{L}$ 的公式已经在上一小节介绍过了。$\boldsymbol{y}_t$ 是语言模型的输出，表示预测的下一个词的概率分布 $P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1{:}\boldsymbol{x}_t)$, 它同样根据 LSTM 隐层单元得到，公式如下：

$$\boldsymbol{y}_t = softmax(W_{ho}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{4–9}$$

上述就我我们单向自信息 Multi-view 语言模型的结构描述。

4.4 模型的训练方法

我们提出的模型由两部分组成：标注模型和语言模型。语言模型训练过程遵循标准约定，先计算每个单词的交叉熵，然后进行反向传播。我们采用基于小批量的随机梯度下降法 (SGD) 作为优化

方法。但是由于语言模型和标注模型的连接方式的不同，我们尝试了五种不同的训练方法，并进行实验和比较。五种训练方式如下：

1) 将 LSTM 标记模型作为一个独立的模型进行训练，在训练多视角语言模型时将标注模型固定，并不继续进行训练更新。五种方式中只有这种方法才能利用解码过程，因为后面的方法需要训练标注模型，但解码过程出来的结果是确定的值，无论是解码还是采样过后都不支持将误差从语言模型传递到标注模型，从而不支持继续训练标注模型。其优势在于，解码器有利于提升标注模型，但它的缺点是在语言模型训练时标注模型不能更新。

2) 提前对 LSTM 标注模型进行训练，不同于第一种方法，这种方法中标注模型也会在训练语言模型的时候进行训练和更新，此时标注模型的输出是以概率分布的形式输入到语言模型中。标注模型的学习率下降速度和语言模型保持一致。这个方法有个致命缺点是：训练好的标注模型会因为不恰当的学习率在训练语言模型的时候训毁。

3) 并不事先训练好标注模型，而是对其进行随机初始化，然后整个系统——标注模型和语言模型部分一起记性训练，使用相同的学习率。

4) 第三种和第四种方式都是实用的相同学习率，本方法在第二种训练方法的基础上，采用学习率稳定算子 β 自适应算法[36][37] 进行模型的更新，目的是为了合理的调整学习旅使得在不同的模型部分使用适合于它自己的学习率。比如第二种方法中标注模型是事先训练好的，调整幅度就应该非常小。

5) 与第四种方式相似，这个方法是在第三种方法的基础上加入稳定算子 β 自适应算法。

由于我们所提出的模型的最佳训练方法是未知的，所以这五种方法都在实验中进行了测试和评估。结果如第 4 章实验部分所示。

第五章 实验与分析

5.1 实验设计

5.1.1 实验目的

在进行我们主要的语言模型 Multi-view 架构测试与对比的实验之前，我们先将辅助的实验进行测试，选取，比较。

首先，我们想探究什么样的辅助信息可以被应用提升我们的模型。因为我们的架构的两部分的模型：标注模型和语言模型是以此层面链接在一起的，语音识别的过程中的语言模型也是以词为单位的，因为只考虑词层面的辅助特征，因为我们仅仅考虑词性标注（POS）、命名实体识别（NER）和语法块标注（CHUNK）等三种信息。在此基础上，我们需要完成并实现辅助信息标注的提取工具并进行测试和评估。

第二，根据前文所说的，我们架构中包含一个单向标注模型，而这个标注模型相比于当前应用比较多的双向标注模型之间的差距我们得测试清楚，需要确定采用单向标注是否会损失太多模型信息。为了测试这个模型，我们的训练数据采用斯坦福大学的标注工具作为真实标注，比较它们之间的差距。其中斯坦福的工具为 Core-NLP tools[1] 后面也将以这个标注作为对比进行我们模型和其他模型的比较。

第三，由于实验架构中使用到了本人的另一个研究内容：前文详细描述过的 β 稳定算子自适应算法在 LSTM 语言模型中的应用。因此我们将 β 稳定算子自适应算法在 LSTM 语言模型中的影响和效果，相比于普通的 LSTM 语言模型进行比较并得出最好的使用方法的结论。

第四，根据前文提到的五种对于本文模型架构的不同训练方法，由于我们没有那么多机器时间对所有的数据集都在五种方法上进行尝试和比较，因此我们仅在最小的中文短信数据集上对五种方式进行实验比较，最终得出效果最好的一种训练方式，并将这种方式用于所有其它数据集上的实验。

最后就是首先对本文提出的第一个单向辅助信息 Multi-view 语言模型和其它 LSTM 和 Mulit-view 语言模型以及 Multi-task 语言模型之间的比较。比较实验主要在 PPL 和 ASR 重打分两个维度上展开。

结构在本文提出的第二个结构化语言模型——Teacher-student 模型结构上的实验和对比。

实验部分在所有实验，无论是辅助实验还是主要结构化比较实验中，都一边给出实验参数、展示实验数据，一边进行分析和给出结论。

在所有的多视角语言模型实验和 Teacher-student 实验中，我们采取的都是单层 LSTM 语言模型，在学习率自适应的实验中我们会采取多层的模型进行验证。其中所有的隐层都是大小为 300 的 LSTM 结构，并且所有的模型都采用 dropout 比率为 0.5 的训练方法。如果有极少数的意外情况会在实验结果说明中提及。

[1] http://nlp.stanford.edu/software/

5.1.2 测试标准

混淆度是一种常见的衡量自然语言处理领域（NLP）中的语言模型的好坏的指标，通俗的讲就是对于语言模型所估计的一句话出现的概率，用句子长度进行归一化。混淆度是由语言模型和测试句子之间的交叉熵算出来的，不同的测试句子测得的混淆度可能会有轻微区别，但是随着测试句子的增多会收敛。在后面的测试中，我们在比较不同的语言模型或者不同的参数对语言模型的影响的时候会控制训练数据和测试数据相同。

字错误率和女字错误率是 ASR 重打分任务重的一个评判标准。由于 PPL 任务往往应用于自然语言处理中的语言模型的评判，然而语言模型的 PPL 性能和它在语音识别中的好坏并不完全成正相关，因此作为想评判语言模型在语音识别中的性能还需要重打分这个任务。其中最重要的两个指标就是字错误率和句子错误率。

下面将分开介绍这三个指标的计算方式。（1）混淆度（PPL）我们常常用计算一个语言模型的混淆度（Perplexity，PPL）来估计语言模型之间的优劣差别。对于一个句子 w（长度为 K）的 PPL 计算方式如公式5–1。

$$PPL(w) = \sqrt[K]{\prod_{i=1}^{K} \frac{1}{P\left(w_i|w_{1....i-1}\right)}} = 2^{-\frac{1}{K}\sum_{i=1}^{K}\log_2 P(w_i|w_{1....i-1})} \tag{5–1}$$

（2）字错误率（Character Error Rate，CER）假设待识别句子 S 长度为 N（由 N 个单词组成），识别得到的句子 L 相对原句子来说会有所不同，可能会出现有的单字没有识别出来，有的字识别错误，有的识别出来的单词在原句子中没有出现。我们假设没有识别出来的单字个数为 D，在 L 中识别出来但是在 S 中并没有出现的单字个数为 I，在 S 和 L 中都有单识别错误的单字为 E，则可以得到字错误率的公式如公式5–2。

$$CER = (D + I + E)/n \cdot 100\% \tag{5–2}$$

通过字错误率可以判断识别结果的好坏，从而判断语言模型的好坏 CER 越低说明语言模型越优秀。

（3）句错误率（Sentence Error Rate，SER）同字错误率一样，若识别得到的句子 L 相对原句子来说有不同，根据句子不同的百分比算得句错误率，计算方法为识别错误的句子数量除以句子总数。SER 越低说明越优秀。

5.1.3 实验参数

在所有的多视角语言模型实验和 Teacher-student 实验中，我们采取的都是单层 LSTM 语言模型，在学习率自适应的实验中我们会采取多层的模型进行验证。其中所有的隐层都是大小为 300 的 LSTM 结构。

其中，所有的模型都采用 dropout 比率为 0.5 的训练方法。如果有极少数的意外情况会在实验结果说明中提及。dropout 参数是指在语言模型的训练过程中，随机地扔掉一些训练数据，扔掉的比率即为 dropout 设计的比率。这样做的原因是为了防止过拟合。

另外，mini-batch 参数为 15，这个参数的表示在进行小批量梯度下降的时候的这个“小批量”是多少。chunksize 的大小为 20，这个参数往往是训练过程中会采用分块训练的方式，一个快的特征向量一起经过神经网络进行计算。其余参数均为模型默认参数。

5.2 实验准备

5.2.1 数据集

第一个数据集是 PTB（Penn Treebank Corpus）[Taylor2003The], 它是一个在语言模型中比较通用的英文数据集，这个数据集比较小，因此一轮时间很快。并且由于十分通用，所以很多进行语言模型研究工作的研究人员都会在这个数据集上进行实验，因此有很多可以用来比较的实验基线。这个数据集有大概 4 万句训练句子，监督集有大概 3 千句，测试集合有大概 4 千句英文句子。词表大小非常固定为准确的 1 万。

这个测试集合中的训练语料句子之间是连续的，也就是说是前后文相关的，因此每个句子之间是否截断有比较大不同。是否截断是指在训练一个新的句子时，是否清空前面句子的历史。由于 PTB 句子之间的强相关性，如果不截断相会比截断的训出来的语言模型更好、PPL 更小。目前这两种训练方法中本文采用截断式，即每次训练一句新的句子会清空前面的历史，无论是实验基线还是本文提出的模型，会保持一致具有可比性。

另外 PTB 数据集无法进行 ASR 重打分实验，因此这个数据集由于比较小，更多被我们用来测试一些准备性质的实验。

Swb-fisher 数据集也是一个比较通用英文数据集，它的训练集有一千万个词，测试集部分分两种：测试阶段，使用了 NIST 2000 Hub5e 数据集的交换机子集 (称为 Hub5e，1831 个句子)。对于相应的 ASR 重打分任务，在每一层中，用 7 层的 CD-DNN-HMM 和 2048 神经元对声学模型进行训练。基于傅立叶变换的具有 40 个系数的对数滤波器组被用作特征。在 Swb-fisher 数据集上训练的一个英语插值的 trigram 语言模型用于 1-pass 解码，生成 ASR 重打分任务的 n-best 列表。

SMS 是一个中文数据集，这个数据集从短信服务中收集了两百万单词。在解码阶段，我们采用了一个中文自动对话测试集合 (大约 25 小时)。大概 5000 个小时的中文对话被用来训练解码中用到的 DNN-HMM 声学模型。最后，用 40000 个单词 SMS 数据集一个训练的 tri-gram 语言模型被用来进行 1-pass 解码。

5.2.2 实验硬件

本文所有试验均在同一服务器上运行，其配置如下：本实验硬盘大小：3T；CPU 型号：X5560 @ 2.80GHz；物理 CPU 个数：2 个；CPU 核心（core）总数：16 个；内：64G；网卡：2*1Gbps；

5.2.3 实验平台及工具

（1）SRILM Srilm 是一个用于构建和应用统计语言模型的工具包。它编写于 1995 年斯里兰卡语音技术和研究实验室，并于 1995 至 1997 年经过 Johns Hopkins University 夏季研讨会的修改变得更加出色。它编写自 C++ 语言，相当于一个 C++ 函数库的集成，可以供大家进行基于 N-gram 语言模型的训练和实验，它是开源且免费的，因为优秀的速度正被大家广泛使用[38]。

SRILM 并不是因机器翻译而诞生的，它主要是为语音识别所开发的，全称为 Stanford Research Institute Language Modeling Toolkit。SRILM 用来构建和应用统计语言模型，主要用于语音识别，统计标注和切分，以及机器翻译，可运行在 UNIX 及 Windows 平台上。它主要包含以下几个部分：• 一组实现的语言模型、支持这些模型的数据结构和各种有用的函数的 C++ 类库；• 一组建立在这些类

库基础上的用于执行标准任务的可执行程序，如训练语言模型，在数据集上对这些语言模型进行测试，对文本进行标注或切分等任务。• 一组使相关任务变得容易的各种脚本。

SRILM 的主要目标是支持语言模型的估计和评测。估计是从训练数据（训练集）中得到一个模型，包括最大似然估计及相应的平滑算法；而评测则是从测试集中计算其困惑度（MIT 自然语言处理概率语言模型有相关介绍)。其最基础和最核心的模块是 n-gram 模块，这也是最早实现的模块，包括两个工具：ngram-count 和 ngram，相应的被用来估计语言模型和计算语言模型的困惑度。

（2）Nerv Nerv 是一个实验室自主开发底层是 c 语言，上层是 lua 语言开发的深度学习工具，可以支持神经网络的 DIY ，进行完毕神经网络搭建后能自动进行模型的训练。可以支持单卡 GPU 运算，比较轻量级。

在本文中，主要用来进行单向辅助信息多视角语言模型相关的一系列实验。对比实验中的实验基线和创新模型皆来自于此模型。由于对这个工具相当熟悉，因此修改起来比较容易。

（3）PyTorch PyTorch 是使用 GPU 和 CPU 优化的深度学习张量库。类似于 tensorflow , Caffe , MXnet 一样，非常底层的框架，它的前身是 torch，主要的语言接口是 Lua，在如今 github 上前事的机器学习项目有九个都是 python 的时代，一直没有太多的人使用，比较小众。而 pytorch 如今重新归来，用 python 重写了整个框架，又重新回到了我的视线。

在本论文中，Pytorch 主要用来完成 Teacher-student 的一系列对比实验。

（4）Stanford-NLP 词性标注工具和命名实体标注工具斯坦福大学自然语言处理组是世界知名的 NLP 研究小组，他们提供了一系列开源的 Java 文本分析工具，包括分词器 (Word Segmenter)，词性标注工具（Part-Of-Speech Tagger)，命名实体识别工具（Named Entity Recognizer)，句法分析器（Parser）等。他们还为这些工具训练了相应的中文模型，支持中文文本处理。在使用 NLTK 的过程中，发现当前版本的 NLTK 已经提供了相应的斯坦福文本处理工具接口，包括词性标注，命名实体识别和句法分析器的接口，不过可惜的是。

Stanford-NLP 词性标注工具是 Stanford 大学编写的自然语言处理（NLP）工具集合中的一个，主要用于词性的标记，同样支持英文、中文等多国语言。输入数据为已经经过分词处理的文本，输出数据为在每个单词后面加入词性信息的文本，单词和词性之间由 # 连接，每组之间同样由空格分隔。

命名实体标注工具同上，同样支持多国语言，输出的文本为加入命名实体标注的文本。

（5）Chunking 语法块生成工具语法块标注比较复杂。由于网上现有的语法块标注工具都是仅支持英文，斯坦福 NLP 工具中没有这个标注工具。但是有论文工作主要完成的这个功能。因此我们自己写了个语法块标注工具。我们首先用斯坦福 NLP 工具中的语法树对所有文本进行预处理，然后根据[31] 中提供的语法树转语法块的算法对数据进行标注，最后得到我们想要的标注。

5.3 实验结果与分析：面向结构化语言模型的学习率自适应算法

在第一部分试验中，我们在单层语言模型网络中进行试验，控制网络结构和除学习率以外的所有参数相同，通过变化学习率比较最后训练结果，实验结果如表5–1，通过实验结果发现加入稳定算子确实可以使语言模型的训练免于对初始学习率不优的干扰。

第二部分实验结果如5–2，我们主要是通过实验探究稳定算子加入到语言模型中是否会提升语言模型的性能；以何种方式加入稳定算子会取的更好的效果；稳定算子在不同结构、不同参数以及不同深度的语言模型中的表现各是怎样的。实验结果如表 2 所示，其网络类型中的 LM 为普通语言

表 5–1 稳定算子对不同的初始学习率的影响

Table 5–1 The influence of the β on the different initial learning rates

初始学习率	是否加入 β	PPL
0.015	否	138.8
0.08	否	124.0
0.15	否	103.3
0.8	否	117.1
1.5	否	129.4
0.015	是	108.4
0.08	是	108.0
0.15	是	107.3
0.8	是	109.4
1.5	是	110.1

模型，Multitask 为多任务模型，在本文中是用的语言模型任务加词性标注任务；隐层分为单层网络和多层网络，单层网络皆为 LSTM 网络，多层的分为全为 LSTM 网络和半 LSTM 半 DNN 网络，其中在 Multitask 任务中，前面部分为两个任务公用隐层，后面的部分为每个任务各自独立的隐层。本文所有实验均在 nerv 上实现，采用的隐层大小为 300，训练总轮数固定 25，batch 大小为 20，chunk 大小为 15，不加分类训练及 dropout。β 类型为空即为不加入稳定算子。

表 5–2 稳定算子在不同模型中的 PPL

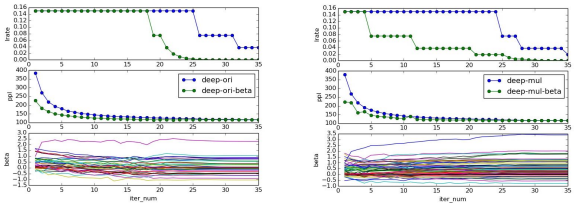
Table 5–2 PPL of different models with β

实验编号	网络类型	β 类型	隐层参数	PPL
1	LM	-	1 lstm	103.3
2	LM	Same	1 lstm	107.3
3	LM	Diff	1 lstm	106.9
4	Multitask	-	1 lstm	102.2
5	Multitask	Same	1 lstm	106.1
6	Multitask	Diff	1 lstm	105.6
7	LM	-	1 lstm+2 dnn	111.8
8	LM	Same	1 lstm+2 dnn	111.2
9	LM	Diff	1 lstm+2 dnn	111.0
10	LM	Diff	3 lstm	139.0
11	Multitask	-	1 lstm+2 dnn	110.6
12	Multitask	Same	1 lstm+2 dnn	110.1
13	Multitask	Diff	1 lstm+2 dnn	109.8
14	Multitask	Diff	1 lstm+2lstm	131.9

[1]

通过对比实验 2,3 和对比实验 5,6 发现第二种稳定算子引入算法稍微优于第一种，即每个隐层的每个 LSTM 单元的每个门分别拥有各自独立的稳定算子 β。对比实验 1,2 和 4,5 发现普通的稳定算子 β 的引入算法对于单层 LSTM 语言模型和 Multitask 语言模型的提升没有帮助，反而有阻碍作用；然而对比实验 7,9 和实验 11,13 发现当网络结构加深的情况下，稳定算子开始起了作用，而本实验无法继续加深网络，因为 PTB 训练数据小，使得参数量大的网络无法训练至收敛，这也是 3 层的网络结构比单层网络结构差的原因。比较实验 10 和 9 以及实验 13 和 14 可以发现，因为 LSTM 参数量远大于 DNN 网络，使得 PTB 无法有效训练深层的 LSTM 语言模型。

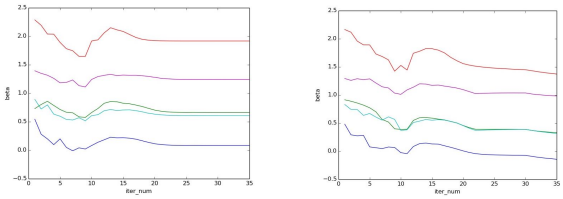
为了进行更深入的分析，观察为何加入稳定算子的 PPL 没有进一步减小，以及针对性的进行后面的工作，我们将实验 7,9 和实验 11,13 这两组分别进行对比，比较他们的学习率（lrate）、混淆度（ppl）和每个门的 beta 数值在 35 轮中的变化情况，分别如图**??**和图**??**所示。



(a) house (b) house

图 5–1 学习率、ppl 和 β 随着轮数的变化曲线图

Fig 5–1 The change curve of learning rate, PPL and β



(a) house (b) house

图 5–2 beta 变化曲线图

Fig 5–2 The change curve of β

通过这两组对比实验，我们可以更直观的发现，加入稳定算子 beta 的模型，ppl 会更快速地收敛，以至于学习率下降得更快，到了一定范围内学习率几乎接近于 0 以至于后面的训练几乎不起作用，从而被不加稳定算子的模型接近甚至超越。另外，通过横向对比我们发现，多任务学习的 PPL 和学习率的收敛速度比单任务语言模型快，并且收敛后的 β 的绝对值要更大。为了探究加入稳定算子的 L2 范数是否能够提升语言模型，我们设计第三组实验，同样是在 PTB 数据上，我们控制对照组的模型结构和参数相同，一组加入原始的 β，另一组在第一组的基础上加上 β 的 L2 范数。实验结果如表5–3。

如表 3 所示，在代码实现过程中加入 β 的 L2 范数优化后，语言模型的 PPL 结果有微小提升，大概 1

5.4 实验结果与分析：面向 ASR 的单向辅助信息 Multi-view 语言模型

5.4.1 标注模型的测试与评估

POS, NER and CHUNK 是本模型主要用到的三种不同的标注.

表5–4 展示了单向 LSTM 标注模型在所有数据集上对三种标注的正确率。我们对于这些标注模型的训练数据的标注并不是真实标注，这些标注来源于斯坦福大学开发的 NLPCore 工具进行的标注，但是在这里我们将这些标注当成是真实标注。因此也可以说这些正确率是相比于斯坦福 NLP 工具的标注的相似度，甚至说相似度更为贴切。

对于这三种标注来说，可以发现正确率都不低，这也意味着使用单向模型并不会丢失太多的信息。这是在句子都是合理的情况下，当重打分任务重句子不合理的情况，相似度没有这么高，但是单向模型反而会比双向模型的标注更为合理，因为它没有使用后文不合理的信息。

5.4.2 Evaluation of training methods

表5–5 展示了五种不同的训练方式去训练模型的 PPL 的区别，这五种方式在第三章提到过。对比这五种方式我们用的是 sms 短信数据集。

实验的结果如表5–5，表明第一种方式是对于单向辅助信息 Multi-view 语言模型来说最优的方式。第二种和第三种方法明显比第一个方法差，因为语言模型和标记模型相对独立，是两个不同的

表 5–3 加入 $\beta - L2norm$ 的比较

Table 5–3 Comparison with β or not

网络类型	隐层参数	是否加入 β 和 L2norm	PPL
LM	1lstm	None	103.346
LM	1lstm	Only β	107.316
LM	1lstm	β+L2norm	104.755
Multitask	1lstm	None	102.207
Multitask	1lstm	Only β	106.101
Multitask	1lstm	β+L2norm	103.115
LM	1 lstm+2 dnn	None	111.8
LM	1 lstm+2 dnn	Only β	111.2
LM	1 lstm+2 dnn	β+L2norm	110.1
Multitask	1 lstm+2 dnn	None	110.6
Multitask	1 lstm+2 dnn	Only β	110.1
Multitask	1 lstm+2 dnn	β+L2norm	109.0

[1]

表 5–4 *Accuracy of word-level features on all data-sets we used*

Data	Accuracy(decoding/no decoding)		
	POS	NER	CHUNK
ptb	96.36/96.32	80.32/82.06	84.23/81.77
fisher	94.24/94.24	79.93/81.97	85.72/83.47
sms	94.87/94.84	80.44/82.12	85.55/82.98

表 5–5 *Perplexity comparison of different methods for training proposed model*

Model	Training Method	PPL
LSTM LM	-	102.11
Multi-view LM	1	98.02
	2	135.72
	3	148.65
	4	105.47
	5	107.21

任务。将两个模型串联在一起，并对它们进行相同的学习速进行训练，将无法对两者进行适当的学习，往往会使标注模型的学习率不合适，因此不会产生双赢的结果。这一结论得到了第四和第五种方法的实验结果的支持。在使用能调节不同 LSTM 中学习速率的稳定算子时，PPL 将远远小于第二和第三种方法。

然而，最后两种方法并没有如预期的那样超过第一个方法，因为 β 稳定算子只能弥补不完善的学习速率的不足，不会比用自己适当的学习速度训练它们更好。此外，第一种方法中还进行了 Viterbi 解码，这对于标注的合理性来说具有相当大的约束，比相比于斯坦福模型的一致性来说更重要。因为在 ASR 重打分任务的 n-best 句子往往不是那么合理，因此使用斯坦福的标注工具得到的标注会有很多是不准确的（即使我们将其当作标注模型的真实标注），而单向标注模型的信息抽象能力，结合 Viterbi 解码的约束，使得单向标注模型和斯坦福标注的准确性（一致性）不高，却经常出现斯坦福标注不合理而单向标注模型的输出更合理的情况。

这个方法看似简单，却并不是没有意义的，因为不进行这个比较实验就无法得到最好的训练方式，也不利于之后的实验分析。

因此，采用第一种方法将会被用来对所有的单向辅助信息的 Multi-view 语言模型进行训练。在后续的所有实验过程中，将对标注模型进行独立训练，通过 Viterbi 解码，然后与 LSTM 语言模型连接在一起。

5.4.3 Multi-view 语言模型的测试

在本节中，我们使用上述数据集来评估我们的多视角语言模型的有效性。首先，我们在 PTB 数据集上尝试了各种各样的模型，并将它们各自的 PPL 进行了比较，但是需要注意到 PTB 集合不支持 ASR 重打分任务。

如图5–6所示，对于多视角（Multi-view）语言模型来说，无论是使用斯坦福工具标注的，或者

是使用我们的模型，对于 PPL 来说都提升非常多。其中以词性标注的提升最为巨大，大概有 10%。

表 5–6 *Perplexity comparison of different LMs on PTB data set*

Model	Tagging	PPL
4-gram	-	141.46
LSTM LM	-	98.73
Multi-task+POS	-	100.88
Multi-view+Multi-task+POS	-	93.47
Multi-view+POS	Standford	91.69
	LSTM	**93.82**
Multi-view+NER	Standford	97.63
	LSTM	**97.92**
Multi-view+CHUNK	Standford	94.34
	LSTM	**95.63**

表5–7 展示了混淆度，词错误率，和句子错误率在 SMS 短信数据集加入不同的标注上的表现。结果表示在加入词性标注的情况下，语言模型会有非常大的提升，命名实体和语法块的加入也使语言模型有所提升，但是提升不是很大。主要原因可能是命名实体标注是很稀疏的标注，导致了绝大部分词的标注都是空，添加的信息不够丰富。而对于语法块来说，标注工具的准确性并不高。基于以上的观点，为了后续实验更加容易，我们后面仅仅在加入词性标注的额外信息基础上进行，在比较大的 Swb-fisher 上进行测试。

表5–8展示了混淆度，词错误率，和句子错误率在 Swb-fisher 数据集上加入不同的标注上的表现。这组数据的观察结果与中文短信的数据集的结果相似。

表 5–7 *Perplexity, WER (%) and SER (%) comparison on Chinese SMS task*

Model	Tagging	PPL	WER	SER
4-gram	-	124.23	13.41	42.16
LSTM	-	102.11	11.30	41.59
Multi-task+POS	-	103.42	11.32	41.63
Multi-task+Multi-view+POS	-	98.24	10.91	40.89
Multi-view+POS	Standford	94.41	11.19	41.92
	LSTM	**98.02**	**10.83**	**40.71**
Multi-view+NER	Standford	101.88	11.28	41.59
	LSTM	**102.08**	**11.32**	**41.72**
Multi-view+CHUNK	Standford	96.71	11.25	41.63
	LSTM	**100.30**	**11.02**	**41.23**

我们将本文提出的单向辅助信息 LSTM 语言模型和实验基线（普通传统的 LSTM 语言模型）相比，混淆度、词错误率和句子错误率分别减少了 4.0%, 4.0%, 2.0%，无论是在中文数据集还是英文数

表 5–8 *Perplexity, WER (%) and SER (%) comparison on English Fisher task*

Model	Tagging	PPL	WER	SER
4-gram	-	79.12	16.3	53.75
LSTM	-	65.42	15.62	53.42
Multi-task+POS	-	65.93	15.60	53.65
Multi-task+Multi-view+POS	-	92.77	15.20	52.23
Multi-view+POS	Standford	60.24	15.73	53.40
	LSTM	**62.71**	**15.01**	**52.19**

据集上。另外，我们的模型相比于普通的 Multi-view 多视角语言模型来说，在加入词性标注辅助信息时，词错误率和女字错误率上分别也减少了 4.4%, 2.2% and 3.2%, 2.8% .

在这里需要强调的是，虽然我们的模型在混淆度这个维度上，在加入词性标注的情况下没有比普通的多视角语言模型好，但这是正常的，在面第一章和第三章的研究动机反复提到过的因为普通的多视角语言模型会引入不应该被加入的额外过多的后文信息导致混淆度很好但是在语音识别上没有提升。

对于所有的数据集，Stanford 和 LSTM 标注模型的多视图 LSTM 语言模型，与 LSTM RNN 语言模型相比，在困惑方面有了显著的改进。与斯坦福词性标注的普通多视角 LSTM 语言模型在混淆度上的结果更好，这在之前的工作中得到了广泛的验证，但在 ASR 重打分任务中没有显示出相应提升[32]。

斯坦福大学的 CoreNLP POS 标记工具使用了最大熵算法模型[39]，利用未来的信息，未来的信息只对 PPL 任务有贡献，而不是 ASR 重打分任务。最大熵模型使用上下文未来信息来计算文本的后验概率。也就是说，在语言模型训练中添加这个词性标注信息，就等于将未来的信息添加到现在的单词中。这导致了一种对于混淆度的理所当然的优化，但在重新分析任务中没有 WER 和 SER 的减少。

另外，我们想确认词性标注部分是否对 PPL 有贡献，因此添加了另一个对比实验来验证我们的结论。为了比较所提出的和传统模型之间的标注信息的影响，我们通过对模型的输入进行删除和添加标记特性来进行实验。

The comparison is shown in Table 5–9. ”Without-tag” means that we deliberately remove the feature feed from the trained tagging part of proposed model. The results indicate that tests with tag feature are generally better than those without. And the latter even does not outperform an ordinary LSTM LM, which means tag feature plays an important role in the LM performance. Furthermore, the result confirms the contribution of our POS tagging model. Moreover, our model performs better than the original multi-view LM without tags, which confirms the contribution of our tagging model in the proposed multi-view structure. 该比较显示在表5–9中。“without-tag”指的是我们故意从被训练的标签部分中删除该特性。结果表明，带有标记特征的测试通常比无标记的测试更好。后者甚至没有超过普通的 LSTM LM，这意味着标记特性在 LM 性能中起着重要的作用。此外，结果证实了我们的 POS 标记模型的贡献。此外，我们的模型比原始的多视图 LM 模型表现得更好，这也证实了我们的标签模型在多视图结构中的贡献。

The proposed LSTM LM with word-synchronized auxiliary feature, performs better not only on ASR-rescore task but also PPL task. For this result, we consider the following two aspects. On one hand, the tagging

表 5–9 *Perplexity on different test-data*

Data-set	Tagging	PPL	
		without-tag	with-tag
PTB	Standford	98.77	91.69
	LSTM	98.64	93.82
Fisher	Standford	66.25	60.24
	LSTM	65.79	62.71
SMS	Standford	102.20	94.41
	LSTM	102.13	98.02

model change the contextual information to word-synchronized auxiliary information, which is rather useful in ASR-rescore task. On the other hand, the proposed LM with unidirectional tagging process permits the decoding of LVCSR to proceed in real time. 该 LSTM LM 系统具有文字同步辅助功能，不仅在 asr - rescore 任务上表现更好，而且在 PPL 任务中表现更好。为此，我们考虑以下两个方面。一方面，标记模型将上下文信息更改为单词同步辅助信息，这在 asr - rescore 任务中非常有用。另一方面，所提出的具有单向标记过程的 LM 系统允许对 LVCSR 进行实时解码。

In this paper, we propose a multi-view LSTM LM with unidirectional tagging model, which produces word-synchronized auxiliary feature that only incorporate previous contextual information. This auxiliary feature is combined with the word sequence to train a multi-view LSTM LM. Five different training methods for this model are tested, and the best one is used in the large-scale ASR task. In the comparison experiments between our model and related works (N-gram LM, multi-task LM, multi-view LM with Stanford tagging and multi-task combined with multi-view LM), the used data sets are PTB, English Fisher and Chinese SMS and PPL, WER and SER are used as the evaluation criteria. Our proposed model shows significant improvements for all word-level features including POS, NER and chunking on WER and SER. Especially for the POS feature on English Fisher, our proposed model not only gives gian (4.0%) on PPL, but also shows significant WER and SER reduction (relative 4.0% and 2.0%) in ASR task, compared with the baseline (LSTM LM). Most importantly, comparing to the multi-view LM with POS feature produced by traditional model (Stanford tool), our model shows better result of WER and SER (4.4% and 2.2%) in ASR-rescore task. For more related models like multi-task and multi-task combined with multi-view models, our model all shows achievement in varying degrees.

全文总结

这里是全文总结内容。

2015 年 2 月 28 日，中央在北京召开全国精神文明建设工作表彰暨学雷锋志愿服务大会，公布全国文明城市（区）、文明村镇、文明单位名单。上海交通大学荣获全国文明单位称号。

全国文明单位这一荣誉是对交大人始终高度重视文明文化工作的肯定，是对交大长期以来文明创建工作成绩的褒奖。在学校党委、文明委的领导下，交大坚持将文明创建工作纳入学校建设世界一流大学的工作中，全体师生医护员工群策群力、积极开拓，落实国家和上海市有关文明创建的各项要求，以改革创新、科学发展为主线，以质量提升为目标，聚焦文明创建工作出现的重点和难点，优化文明创建工作机制，传播学校良好形象，提升社会美誉度，显著增强学校软实力。2007 至 2012 年间，上海交大连续三届荣获“上海市文明单位”称号，成为创建全国文明单位的新起点。

上海交大自启动争创全国文明单位工作以来，凝魂聚气、改革创新，积极培育和践行社会主义核心价值观。坚持统筹兼顾、多措并举，将争创全国文明单位与学校各项中心工作紧密结合，着力构建学校文明创建新格局，不断提升师生医护员工文明素养，以“冲击世界一流大学汇聚强大精神动力”为指导思想，以“聚焦改革、多元推进、以评促建、丰富内涵、彰显特色”为工作原则，并由全体校领导群策领衔“党的建设深化、思想教育深入、办学成绩显著、大学文化丰富、校园环境优化、社会责任担当”六大板块共 28 项重点突破工作，全面展现近年来交大文明创建工作的全貌和成就。

进入新阶段，学校将继续开拓文明创建工作新格局，不断深化工作理念和工作实践，创新工作载体、丰富活动内涵、凸显创建成效，积极服务于学校各项中心工作和改革发展的大局面，在上级党委、文明委的关心下，在学校党委的直接领导下，与时俱进、开拓创新，为深化内涵建设、加快建成世界一流大学、推动国家进步和社会发展而努力奋斗！

上海交通大学医学院附属仁济医院也获得全国文明单位称号。

附录 A 搭建模板编译环境

A.1 安装 TeX 发行版

A.1.1 Mac OS X

Mac 用户可以从 MacTeX 主页[1]下载 MacTeX 2015。也可以通过 brew 包管理器[2]安装 MacTeX 2015。

```
brew cask install mactex
```

A.1.2 Linux

建议 Linux 用户使用 TeXLive 主页[3]的脚本来安装 TeXLive 2015。以下命令将把 TeXLive 发行版安装到当前用户的家目录下。若计划安装一个供系统上所有用户使用的 TeXLive，请使用 root 账户操作。

```
wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
tar xzvpf install-tl-unx.tar.gz
cd install-tl-20150411/
./install-tl
```

A.2 安装中文字体

A.2.1 Mac OS X、Deepin

Mac 和 Deepin 用户双击字体文件即可安装字体。

A.2.2 RedHat/CentOS 用户

RedHat/CentOS 用户请先将字体文件复制到字体目录下，调用 fc-cache 刷新缓存后即可在 TeXLive 中使用新字体。

```
mkdir ~/.fonts
cp *.ttf ~/.fonts                # 当前用户可用新字体
cp *.ttf /usr/share/fonts/local/    # 所有用户可以使用新字体
fc-cache -f
```

[1] https://tug.org/mactex/

[2] http://caskroom.io

[3] https://www.tug.org/texlive/

附录 B Maxwell Equations

选择二维情况，有如下的偏振矢量：

$$\mathbf{E} = E_z(r,\theta)\hat{\mathbf{z}} \tag{B–1a}$$
$$\mathbf{H} = H_r(r,\theta))\hat{\mathbf{r}} + H_\theta(r,\theta)\hat{\boldsymbol{\theta}} \tag{B–1b}$$

对上式求旋度：

$$\nabla\times\mathbf{E} = \frac{1}{r}\frac{\partial E_z}{\partial\theta}\hat{\mathbf{r}} - \frac{\partial E_z}{\partial r}\hat{\boldsymbol{\theta}} \tag{B–2a}$$
$$\nabla\times\mathbf{H} = \left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta}\right]\hat{\mathbf{z}} \tag{B–2b}$$

因为在柱坐标系下，$\overline{\mu}$ 是对角的，所以 Maxwell 方程组中电场 $\mathbf{E}$ 的旋度：

$$\nabla\times\mathbf{E} = \mathbf{i}\omega\mathbf{B} \tag{B–3a}$$
$$\frac{1}{r}\frac{\partial E_z}{\partial\theta}\hat{\mathbf{r}} - \frac{\partial E_z}{\partial r}\hat{\boldsymbol{\theta}} = \mathbf{i}\omega\mu_r H_r\hat{\mathbf{r}} + \mathbf{i}\omega\mu_\theta H_\theta\hat{\boldsymbol{\theta}} \tag{B–3b}$$

所以 $\mathbf{H}$ 的各个分量可以写为：

$$H_r = \frac{1}{\mathbf{i}\omega\mu_r}\frac{1}{r}\frac{\partial E_z}{\partial\theta} \tag{B–4a}$$
$$H_\theta = -\frac{1}{\mathbf{i}\omega\mu_\theta}\frac{\partial E_z}{\partial r} \tag{B–4b}$$

同样地，在柱坐标系下，$\overline{\epsilon}$ 是对角的，所以 Maxwell 方程组中磁场 $\mathbf{H}$ 的旋度：

$$\nabla\times\mathbf{H} = -\mathbf{i}\omega\mathbf{D} \tag{B–5a}$$
$$\left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta}\right]\hat{\mathbf{z}} = -\mathbf{i}\omega\overline{\epsilon}\mathbf{E} = -\mathbf{i}\omega\epsilon_z E_z\hat{\mathbf{z}} \tag{B–5b}$$
$$\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial\theta} = -\mathbf{i}\omega\epsilon_z E_z \tag{B–5c}$$

由此我们可以得到关于 E_z 的波函数方程：

$$\frac{1}{\mu_\theta\epsilon_z}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial E_z}{\partial r}\right) + \frac{1}{\mu_r\epsilon_z}\frac{1}{r^2}\frac{\partial^2 E_z}{\partial\theta^2} + \omega^2 E_z = 0 \tag{B–6}$$

附录 C 从 CJK–LaTeX 转向 XeLaTeX

我习惯把 v0.2a 使用 dvipdfmx 编译的硕士学位论文模板称为“CJK–LaTeX 模板”，而这个使用 XeLaTeX 引擎 (xelatex 程序) 处理的模板则被称为“XeLaTeX/LaTeX 模板”。从 CJK–LaTeX 模板迁移到 XeLaTeXLaTeX 模板的好处有下：

☺ 搭建 XeLaTeX 环境比搭建 CJK–LaTeX 环境更容易；

☺ 更简单的字体控制；

☺ 完美支持 PDF/EPS/PNG/JPG 图片，不需要“bound box(.bb)”文件；

☺ 支持 OpenType 字体的复杂字型变化功能；

当然，这也是有代价的。由于 XeLaTeX 比较新，在我看来，使用 XeLaTeX 模板所必须付出的代价是：

☹ 必须把你“古老的”TeX 系统更新为较新的版本。TeXLive 2012 和 CTeX 2.9.2 能够编译这份模板，而更早的版本则无能为力。

☹ 需要花一些时间把你在老模板上的工作迁移到新模板上。

第一条就看你如何取舍了，新系统通常意味着更好的兼容性，值得升级。而转换模板也不是什么特别困难的事情，可以这样完成：

1. 备份你要转换的源文件，以防你的工作成果丢失；
2. 将你原来的 tex 以及 bib 文件另存为 UTF-8 编码的文件。iconv、vim、emacs、UEdit 等等工具都可以完成。WinEdt 对文件编码识别功能很差 (到了 v6.0 还是如此)，不推荐作为字符编码转换工具；
3. 将 diss.tex 导言区中的内容替换为 XeTeX 模板 diss.tex 导言区的内容；
4. 将你对原先导言区的修改，小心翼翼地合并到新的导言区中；
5. 使用 XeTeX 模板中的 GBT7714-2005NLang.bst 替换原有的 bst 文件，新的 bst 文件只是将字符编码转换为 UTF-8；
6. 删除 bouding box 文件；
7. 使用本文**??**介绍的方法，重新编译文档；

附录 D 模板更新记录

2016 年 12 月 v0.9.5 发布，改用 GB7714-2015 参考文献风格。

2016 年 11 月 v0.9.4 发布，增加算法和流程图。

2015 年 6 月 19 日 v0.9 发布，适配 ctex 2.x 宏包，需要使用 TeXLive 2015 编译。

2015 年 3 月 15 日 v0.8 发布，使用 biber/biblatex 组合替代 BibTeX，带来更强大稳定的参考文献处理能力；添加 enumitem 宏包增强列表环境控制能力；完善宏包文字描述。

2015 年 2 月 15 日 v0.7 发布，增加盲审选项，调用外部工具插入扫描件。

2015 年 2 月 14 日 v0.6.5 发布，修正一些小问题，缩减 git 仓库体积，仓库由 sjtu-thesis-template-latex 更名为 SJTUThesis。

2014 年 12 月 17 日 v0.6 发布，学士、硕士、博士学位论文模板合并在了一起。

2013 年 5 月 26 日 v0.5.3 发布，更正 subsubsection 格式错误，这个错误导致如”1.1 小结” 这样的标题没有被正确加粗。

2012 年 12 月 27 日 v0.5.2 发布，更正拼写错误。在 diss.tex 加入 ack.tex。

2012 年 12 月 21 日 v0.5.1 发布，在 LaTeX 命令和中文字符之间留了空格，在 Makefile 中增加 release 功能。

2012 年 12 月 5 日 v0.5 发布，修改说明文件的措辞，更正 Makefile 文件，使用 metalog 宏包替换 xltxtra 宏包，使用 mathtools 宏包替换 amsmath 宏包，移除了所有 CJKtilde(~) 符号。

2012 年 5 月 30 日 v0.4 发布，包含交大学士、硕士、博士学位论文模板。模板在github上管理和更新。

2010 年 12 月 5 日 v0.3a 发布，移植到 XeTeX/LaTeX 上。

2009 年 12 月 25 日 v0.2a 发布，模板由 CASthesis 改名为 sjtumaster。在 diss.tex 中可以方便地改变正文字号、切换但双面打印。增加了不编号的一章“全文总结”。添加了可伸缩符号 (等号、箭头) 的例子，增加了长标题换行的例子。

2009 年 11 月 20 日 v0.1c 发布，增加了 Linux 下使用 ctex 宏包的注意事项、.bib 条目的规范要求，修正了 ctexbook 与 listings 共同使用时的断页错误。

2009 年 11 月 13 日 v0.1b 发布，完善了模板使用说明，增加了定理环境、并列子图、三线表格的例子。

2009 年 11 月 12 日上海交通大学硕士学位论文 LaTeX 模板发布，版本 0.1a。

参考文献

[1] MOHAMED A.-R, DAHL G E, HINTON G. Acoustic modeling using deep belief networks[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2012, 20(1): 14–22.

[2] DAHL G E, YU D, DENG L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition[J]. IEEE Transactions on audio, speech, and language processing, 2012, 20(1): 30–42.

[3] BENGIO Y, LAMBLIN P, POPOVICI D, et al. Greedy layer-wise training of deep networks[C]// Advances in neural information processing systems. [S.l.]: [s.n.], 2007: 153–160.

[4] MOHAMED A.-R, YU D, DENG L. Investigation of full-sequence training of deep belief networks for speech recognition[C]// Eleventh Annual Conference of the International Speech Communication Association. [S.l.]: [s.n.], 2010.

[5] VESELỲ K, GHOSHAL A, BURGET L, et al. Sequence-discriminative training of deep neural networks.[C]// Interspeech. [S.l.]: [s.n.], 2013: 2345–2349.

[6] YU D, SELTZER M L. Improved bottleneck features using pretrained deep neural networks[C]// Twelfth Annual Conference of the International Speech Communication Association. [S.l.]: [s.n.], 2011.

[7] KAWABATA T, TAMOTO M. Back-off method for N-gram smoothing based on binomial posteriori distribution[C]// Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on. Vol. 1. IEEE. [S.l.]: [s.n.], 1996: 192–195.

[8] MOORE R C, QUIRK C. Improved smoothing for N-gram language models based on ordinary counts[C]// Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. Association for Computational Linguistics. [S.l.]: [s.n.], 2009: 349–352.

[9] PICKHARDT R, GOTTRON T, KÖRNER M, et al. A generalized language model as the combination of skipped n-grams and modified kneser-ney smoothing[J]. ArXiv preprint arXiv:1404.3377, 2014.

[10] KUHN R, DE MORI R. A cache-based natural language model for speech recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 1990, 12(6): 570–583.

[11] BROWN P F, DESOUZA P V, MERCER R L, et al. Class-based n-gram models of natural language[J]. Computational linguistics, 1992, 18(4): 467–479.

[12] GILDEA D, HOFMANN T. Topic-based language models using EM[C]// Sixth European Conference on Speech Communication and Technology. [S.l.]: [s.n.], 1999.

[13] GALES M, YOUNG S. The application of hidden Markov models in speech recognition[J]. Foundations and trends in signal processing, 2008, 1(3): 195–304.

[14] DENG L, LI J, HUANG J.-T, et al. Recent advances in deep learning for speech research at Microsoft[C]// IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013. IEEE. [S.l.]: [s.n.], 2013: 8604–8608.

[15] HINTON G, DENG L, YU D, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups[J]. Signal Processing Magazine, IEEE, 2012, 29(6): 82–97.

[16] GRAVES A, JAITLY N. Towards end-to-end speech recognition with recurrent neural networks[C]// Proceedings of the 31st International Conference on Machine Learning (ICML-14). [S.l.]: [s.n.], 2014: 1764–1772.

[17] MIKOLOV T, KARAFIÁT M, BURGET L, et al. Recurrent neural network based language model[C]// INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010. [S.l.]: [s.n.], 2010: 1045–1048.

[18] JURAFSKY D, MARTIN J H. Speech and language processing[M]. [S.l.]: . Pearson, 2014.

[19] MURVEIT H, BUTZBERGER J, DIGALAKIS V, et al. Large-vocabulary dictation using SRI's DECIPHER speech recognition system: Progressive search techniques[C]// IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993. ICASSP-93, 1993. Vol. 2. IEEE. [S.l.]: [s.n.], 1993: 319–322.

[20] GRAVES A, LIWICKI M, FERNÁNDEZ S, et al. A novel connectionist system for unconstrained handwriting recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2009, 31(5): 855–868.

[21] GRAVES A, MOHAMED A.-R, HINTON G. Speech recognition with deep recurrent neural networks[C]// Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE. [S.l.]: [s.n.], 2013: 6645–6649.

[22] HOCHREITER S. Untersuchungen zu dynamischen neuronalen Netzen[J]. Diploma, Technische Universität München, 1991, 91.

[23] HOCHREITER S, BENGIO Y, FRASCONI P, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[24] GHAHREMANI P, DROPPO J. Self-stabilized deep neural network[C]// Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE. [S.l.]: [s.n.], 2016: 5450–5454.

[25] LIU Q, TAN T, YU K. An investigation on deep learning with beta stabilizer[C]// Signal Processing (ICSP), 2016 IEEE 13th International Conference on. IEEE. [S.l.]: [s.n.], 2016: 557–561.

[26] LECUN Y, BOTTOU L, ORR G B, et al. Efficient backprop[G]// Neural networks: Tricks of the trade. [S.l.]: Springer, 1998: 9–50.

[27] KONEČNỲ J, LIU J, RICHTÁRIK P, et al. Mini-batch semi-stochastic gradient descent in the proximal setting[J]. IEEE Journal of Selected Topics in Signal Processing, 2016, 10(2): 242–255.

[28] YAO Y, ROSASCO L, CAPONNETTO A. On early stopping in gradient descent learning[J]. Constructive Approximation, 2007, 26(2): 289–315.

[29] GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning[M]. [S.l.]: MIT press, 2016.

[30] SHI Y, WIGGERS P, JONKER C M. Towards Recurrent Neural Networks Language Models with Linguistic and Contextual Features.[C]// INTERSPEECH. [S.l.]: [s.n.], 2012: 1664–1667.

[31] TJONG E F Kim Sang, BUCHHOLZ S. Introduction to the CoNLL-2000 shared task: chunking[C]// The Workshop on Learning Language in Logic and the Conference on Computational Natural Language Learning. [S.l.]: [s.n.], 2000: 127–132.

[32] SHI Y, LARSON M, PELEMANS J, et al. Integrating meta-information into recurrent neural network language models[J]. Speech Communication, 2015, 73: 64–80.

[33] COLLOBERT R, WESTON J. A unified architecture for natural language processing: deep neural networks with multitask learning[C]// International Conference. [S.l.]: [s.n.], 2008: 160–167.

[34] SCHMID H. Part-of-speech tagging with neural networks[C]// Proceedings of the 15th conference on Computational linguistics-Volume 1. Association for Computational Linguistics. [S.l.]: [s.n.], 1994: 172–176.

[35] WANG P, QIAN Y, SOONG F K, et al. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding[J]. Computer Science, 2015.

[36] GHAHREMANI P, DROPPO J. Self-stabilized deep neural network[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. [S.l.]: [s.n.], 2016: 5450–5454.

[37] QI L, TIAN T, KAI Y. An Investigation on Deep Learning with Beta Stabilizer[C]// IEEE International Conference on Signal Processing. [S.l.]: [s.n.], 2016: 557–561.

[38] STOLCKE A, et al. SRILM-an extensible language modeling toolkit.[C]// Interspeech. Vol. 2002. [S.l.]: [s.n.], 2002: 2002.

[39] TOUTANOVA K, MANNING C D. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger[C]// Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the Meeting of the Association for Computational Linguistics. [S.l.]: [s.n.], 2000: 63–70.

致 谢

感谢所有测试和使用交大学位论文 LaTeX 模板的同学!
感谢那位最先制作出博士学位论文 LaTeX 模板的交大物理系同学!
感谢 William Wang 同学对模板移植做出的巨大贡献!

攻读学位期间发表的学术论文

[1] CHEN H, CHAN C T. Acoustic cloaking in three dimensions using acoustic metamaterials[J]. Applied Physics Letters, 2007, 91:183518.

[2] CHEN H, WU B I, ZHANG B, et al. Electromagnetic Wave Interactions with a Metamaterial Cloak[J]. Physical Review Letters, 2007, 99(6):63903.

攻读学位期间参与的项目

[1] 973 项目“XXX”

[2] 自然基金项目“XXX”

[3] 国防项目“XXX”