基于扩展 N 元文法模型的快速语言模型预测算法

单煜翔 [1] 陈 谐 [1] 史永哲 [1] 刘 加 [1]

摘 要 针对基于动态解码网络的大词汇量连续语音识别器, 本文提出了一种采用扩展 N 元文法模型进行快速语言模型 (Language model, LM) 预测的方法. 扩展 N 元文法模型统一了语言模型和语言模型预测树的表示与分数计算方法, 从而大大简化了解码器的实现, 极大地提升了语言模型预测的速度, 使得高阶语言模型预测成为可能. 扩展 N 元文法模型在解码之前离线生成, 生成过程利用了 N 元文法的稀疏性加速计算过程, 并采用了词尾节点前推和分数量化的方法压缩模型存储空间大小. 实验表明, 相比于采用动态规划在解码过程中实时计算语言模型预测分数的传统方法, 本文提出的方法在相同的字错误率下使得整个识别系统识别速率提升了 5∼9 倍, 并且采用高阶语言模型预测可获得比低阶预测更优的解码速度与精度.

关键词 语音识别, 语言模型预测, N 元文法模型, 解码

引用格式 单煜翔, 陈谐, 史永哲, 刘加. 基于扩展 N 元文法模型的快速语言模型预测算法. 自动化学报, 2012, **38**(10): 1618−1626

DOI 10.3724/SP.J.1004.2012.01618

Fast Language Model Look-ahead Algorithm Using Extended N-gram Model

SHAN Yu-Xiang[1] CHEN Xie[1] SHI Yong-Zhe[1] LIU Jia[1]

Abstract For a dynamic network based large vocabulary continuous speech recognizer, this paper proposes a fast language model (LM) look-ahead method using extended N-gram model. The extended N-gram model unifies the representations and score computations of the LM and the LM look-ahead tree, and thus greatly simplifies the decoder implementation and improves the LM look-ahead speed significantly, which makes higher-order LM look-ahead possible. The extended N-gram model is generated off-line before decoding starts. The generation procedure makes use of sparseness of backing-off N-gram models for efficient look-ahead score computation, and uses word-end node pushing and score quantitation to compact the model′s storage space. Experiments showed that with the same character error rate, the proposed method speeded up the overall recognition speed by a factor of 5∼9 than the traditional dynamic programming method which computes LM look-ahead scores on-line during the decoding process, and that using higher-order LM look-ahead algorithm can achieve a faster decoding speed and better accuracy than using the lower-order look-ahead ones.

Key words Speech recognition, language model look-ahead, N-gram, decoding

Citation Shan Yu-Xiang, Chen Xie, Shi Yong-Zhe, Liu Jia. Fast language model look-ahead algorithm using extended N-gram model. *Acta Automatica Sinica*, 2012, **38**(10): 1618−1626

语言模型 (Language model, LM) 预测是基于词树的大词汇量连续语音识别器中常见的搜索加速手段. 其原理是在解码尚未到达词尾, 即词的最终身份尚未确定时, 尽可能早地将语言模型分数加入, 从而使剪枝更为精准、有效[1]. 相关研究表明, 采用高阶的语言模型进行预测, 可以获得更加精确、约束性更强的语言模型分数, 进一步提升剪枝效率, 改善搜索精度. 但是由于语言模型预测往往采用动态规划的方法从词的语言模型分数自底向上地计算出预测分数, 算法本身具有很高的时间复杂度, 所以在采用高阶语言模型进行预测时, 计算开销往往掩盖甚至超越了其所带来的解码速度提升[2−3], 故而在实际应用中一般采用二元文法模型实现语言模型预测.

针对高阶语言模型的预测问题, 前人的研究主要集中在两个方面: 1) 从语言模型的存储结构和缓存策略上减小语言模型预测的开销, 例如: 采用保序的最小完美哈希函数加速语言模型的访问速度[4−5]、采用节点相关的缓存策略[4−6] 避免预测分数的重复计算等等.这类方法对语言模型预测分数的计算过程本身没有改进, 因此难以从本质上提升语言模型预测的效率. 2) 近年来在利用 N 元文法的稀疏性加速语言模型预测分数计算的研究上取得了长足的进展[7−8]. 这种方法从算法根源着手, 成倍地提高了语言模型预测的效率. 但是这种方法毕竟还是需要

收稿日期 2012-01-01 录用日期 2012-03-22
Manuscript received January 1, 2012; accepted March 22, 2012
国家高技术研究发展计划 (863 计划) (2008AA040201) 国家自然科学基金 (90920302), 国家科技支撑计划 (2009BAH41B01), 国家自然科学基金委员会与香港研究资助局联合科研基金 (60931160443) 资助
Supported by National High Technology Research and Development Program of China (863 Program) (2008AA040201), National Natural Science Foundation of China (90920302), National Science and Technology Pillar Program of China (2009BAH41B01), and National Natural Science Foundation of China and Research Grants Council of Hong Kong (60931160443)
本文责任编委 刘成林
Recommended by Associate Editor LIU Cheng-Lin
1. 清华大学电子工程系 清华信息科学与技术国家实验室 北京 100084
1. Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084

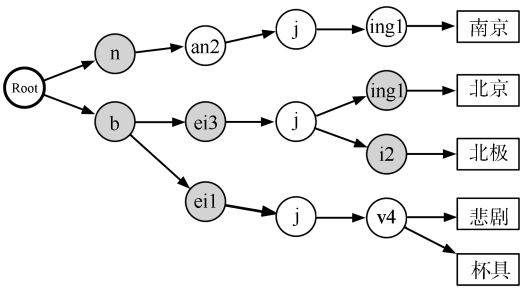
在搜索过程中实时地计算语言模型预测分数, 因此会引入一定的额外计算开销. 当语言模型阶数升高时, 这一额外计算开销也会随之增长, 变得无法忽略[3, 8].

为进一步减小语言模型预测的额外计算开销, 在前人研究的基础上, 本文提出了一种新型的快速语言模型预测方法, 该方法在预测分数的获取上利用 N 元文法模型的稀疏性加速计算; 在存储结构上, 将语言模型预测树统一在 N 元文法语言模型的表示框架之内, 生成扩展的 N 元文法模型, 从而简化了解码器的实现, 提升了语言模型预测的速度. 扩展的 N 元文法模型的生成是作为解码前的预处理步骤离线完成, 因此对解码过程中的实时率没有影响. 为了进一步减少扩展的 N 元文法模型的存储空间, 本文采用了词尾节点前推和语言模型量化的方法, 从网络构架和模型表示上对语言模型进行了压缩.

本文组织结构如下: 第 1 节简述动态解码网络和语言模型预测的基本概念, 第 2 节给出扩展 N 元文法模型的定义并描述相应的预测算法, 第 3 节详细阐述扩展 N 元文法模型的生成及其压缩方法, 最后是实验和结论.

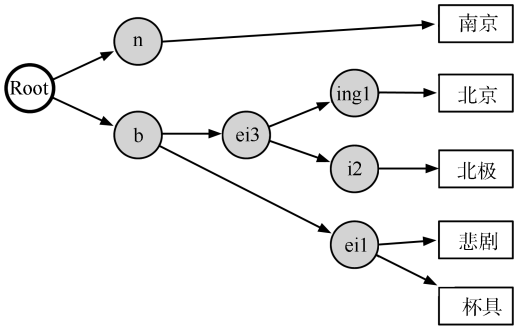
1　动态解码网络与语言模型预测树

与基于静态解码网络的大词汇量连续语音识别器 (如: 基于加权有限状态机 (Weighted finite-state transducers, WFST) 的解码器[9]) 不同, 在基于动态解码网络的语音识别器中, 解码网络仅由声学模型和发音字典构成, 语言模型的信息则在解码过程中动态加入[10]. 通常情况下, 动态解码网络依照字典中词的发音, 构成前缀树的形式. 图 1 (a) 给出了一个仅包含 5 个词的前缀树的示意图, 为清楚起见, 图中采用单音子模型. 树的根节点 (Root) 表示解码网络的起始状态, 树的叶子节点又称为词尾节点, 图中以方框表示, 与字典中的一个词相对应, 表示该词发音的结束和词的身份, 树的中间层节点则对应于声学模型中的一个音素模型 (若采用状态网络则对应于一个声学模型状态), 图中以圆圈表示. 为表达方便, 本文称这些中间层节点为声学节点.



(a) 音素前缀树示例

(a) An example of pronunciation prefix tree



(b) 语言模型预测树示例

(b) An example of language model look-ahead tree

图 1　前缀树解码网络与语言模型预测树示例

Fig. 1　Examples of prefix tree decoding network and language model look-ahead tree

语音识别的任务是在解码网络上找出一条最可能的状态路径. 通常采用令牌传递算法[11] 解决这一问题. 该算法在网络的起始状态上放置一个令牌, 然后将令牌按广度优先的方式向词尾节点传递. 每个令牌对应一条搜索中的候选路径, 记录着该路径的声学分数、语言模型分数、语言模型状态以及词识别假设. 候选路径的总分数一般表示为对数声学分数和对数语言模型分数的加权和. 每当令牌通过一个声学节点, 其对应的声学分数被累加. 当令牌通过词尾节点时, 该词的语言模型分数也被加入到路径分数中去. 为了缩小搜索空间, 通常采用 “束搜索”, 即: 在搜索过程中仅保留路径分数高于一定门限的令牌[10, 12]. 在以上的解码过程中, 由于语言模型的约束在词尾节点才被加入, 所以一定程度上影响了剪枝效率.

语言模型预测的目的是使语言模型分数尽可能早地参与到剪枝过程中去, 使剪枝更加精准、有效, 从而提高解码的速度. 一般的做法是, 当令牌传递到某一网络节点时, 选取从该节点可达的词尾节点中语言模型分数的最大值, 即从当前路径可达的词中概率最大的, 作为语言模型预测分数加入到路径总分数中去. 令 n 为解码网络中的一个节点, h 表示当前令牌经过的历史词序列, 则语言模型预测分数可表示为

$$\pi(n,h)=\pi'(W(n),h)=\max_{w\in W(n)}p(w|h) \quad (1)$$

式中, $W(n)$ 表示从节点 n 可达的词尾节点集合, $p(w|h)$ 表示词 w 在历史路径 h 下的语言模型分数. 为方便下文描述, 这里引入一个辅助函数 $\pi'(W(n), h)$, 它与 $\pi(n,h)$ 表示相同的含义, 区别在于辅助函数的第一个参数为从节点 n 可达的词尾节点集合. 通过语言模型预测, 当令牌在根节点时便可以将语言模型的约束加入. 随着令牌在解码网络中传递, 预测分数也随之修正, 越来越精确, 直到达到词尾节

点, 将准确的语言模型分数加入.

由于需要在每个网络节点计算预测分数, 以上语言模型预测过程的计算复杂度与解码网络的节点个数直接相关. 但事实上, 由于集合 $W(n)$ 仅在解码网络发生分叉时才会改变, 所以在每个节点都做语言模型预测是没有必要的. 为了降低语言模型预测的计算量, 可以将解码网络压缩成为语言模型预测树, 仅在网络的分叉点 (图中阴影节点) 进行语言模型预测, 如图 1 (b) 所示. 预测树的根节点对应于解码网络的根节点, 叶子节点对应于解码网络中的词尾节点, 中间层节点对应于解码网络中的路径分叉点. 在本文中称这些分叉点为预测节点. 若令 $Children(n)$ 表示预测树节点 n 的子节点, 则预测分数可采用如下的动态规划方法计算[1]:

$$\pi(n,h)=\max_{n'\in Children(n)}\pi(n',h) \tag{2}$$

式中, 若 n' 为词尾节点, 则 $\pi(n',h)=p(w[n']|h)$, $w[n']$ 表示 n' 对应的词.

2 基于扩展 N 元文法模型的语言模型预测

2.1 N 元文法模型

在语音识别中, 普遍采用 N 元文法模型来对语言层信息进行建模. 该模型利用 $N-1$ 个历史词对当前词出现的概率进行预测, 是一个马尔科夫过程. 给定一个 N 阶历史词序列 (亦称为语言模型历史状态) $h^N=w_{i-1},\cdots,w_{i-N+1}$, 当前词为 w_i 的概率 (即: 语言模型分数) 为

$$p\left(w_i|h^N\right)=\begin{cases}f(w_i,h^N), & 若C(w_i,h^N)>0\\ p(w_i|h^{N-1})\cdot B(h^N), & 其他\end{cases} \tag{3}$$

式中, $f(w_i,h^N)$ 表示平滑后的 N 元文法概率[13], $C(w_i,h^N)$ 表示在训练语料中 N 元文法 (w_i,h^N) 出现的次数, $B(h^N)$ 表示历史状态 h^N 的回退 (Back-off) 平滑系数.

式 (3) 指出, 若 N 元文法 (w_i,h^N) 曾在语言模型训练语料中出现, 则可以通过统计其出现次数获得其概率的经验估计; 若 N 元文法 (w_i,h^N) 不曾出现于训练语料中, 则可以通过回退到低阶语言模型的方法来估计相应的概率值. 在实际应用中, 当识别器字典包含的词数 V 足够大并且采用高阶的 N 元文法模型 ($N\geq 3$) 时, 即便是相当庞大的训练语料库也只能覆盖到全部 N 元文法的很小一部分. 换言之, 给定历史状态 h^N, 绝大多数 N 元文法的概率都是通过回退的方式估计得到. 这一现象称为 N 元文法模型的稀疏性.

由于稀疏性的存在, 语言模型通常采用类似于稀疏矩阵的方式压缩存储. 为了减小解码时动态查找语言模型分数的时间开销, 存储结构通常采用最小完美哈希表实现, 并辅以基于节点的缓存策略[4]以减小重复查询同一个 N 元文法分数的耗时. 本文的基线系统实现了这两种数据结构用于语言模型分数的快速访问. 在这样的基线系统中, 语言模型分数计算和预测算法可以概括如下:

算法 1. 语言模型分数计算

当具有语言模型历史状态 h^k 的令牌传递到词尾节点 n 上时:

1) 从语言模型分数缓存中查找是否存在对应的语言模型分数 $p\left(w[n]|h^k\right)$.

2) 若缓存中存在该分数, 则将分数返回; 否则到存储 N 元文法模型的最小完美哈希表中查找.

3) 若在哈希表中找到该分数, 则将分数返回; 否则采用式 (3) 做回退处理.

算法 2. 语言模型预测分数计算

当具有语言模型历史状态 h^k 的令牌传递到预测节点 n 上时:

1) 从语言模型预测分数缓存中查找是否存在对应的预测分数 $\pi\left(n,h^k\right)$.

2) 若缓存中存在该分数, 则将分数返回; 否则采用式 (2) 计算预测分数.

3) 计算式 (2) 时按算法 1 获得语言模型分数.

2.2 扩展 N 元文法模型

扩展 N 元文法模型的出发点是将预测节点视为一种特殊的 "词尾节点", 从而将语言模型预测树的表示、预测分数的计算与传统的 N 元文法模型统一起来, 简化解码过程, 提高解码效率. 为了实现这一目标, 首先对预测树中的所有预测节点和词尾节点统一编号. 通过编号为每个节点赋予一个全局唯一的标识. 编号的具体规则可以任意选择, 对后续算法没有影响. 在本文中, 我们从词尾节点开始, 向根节点逐层编号. 在下文中, 用 $I[n]$ 表示预测树节点 n 所对应的编号. 假设在识别系统中采用 N 元文法模型和 M 阶语言模型预测 ($1\leq M\leq N$), 那么定义扩展的 N 元文法分数函数如下:

$$f'\left(I[n]|h^k\right)=\begin{cases}p\left(w[n]|h^k\right), & n\ 为词尾节点\\ \pi\left(n,h^{\min(k,M)}\right), & n\ 为预测节点\end{cases} \tag{4}$$

式中, $1\leq k\leq N$, $w[n]$ 表示词尾节点 n 对应的词. 令 $W_D\left(n,h^k\right)=\left\{w|w\in W(n)\ 且 C\left(w,h^k\right)>0\right\}$, 表示节点 n 可达且在语言模型状态 h^k 下不需要回退计算语言模型分数的词尾节点集合, 可定义扩展 N 元文法模型如下:

$$p'\left(I[n]|h^N\right)=$$

$$\begin{cases} f'\left(I\left[n\right],h^N\right), & 若W_D\left(n,h^k\right)\neq\emptyset \\ p'\left(I\left[n\right]|h^{N-1}\right)\cdot B\left(h^N\right), & 其他 \end{cases} \tag{5}$$

由式 (5) 定义的扩展 N 元文法模型同样继承了传统 N 元文法模型的稀疏性, 因此可以采用与后者相同的存储结构和缓存策略, 以实现快速访问.

2.3　基于扩展 N 元文法模型的语言模型预测

在第 2.1 节所述的基于传统 N 元文法模型的语言模型预测算法中, 语言模型预测过程不仅包含了大量语言模型分数的缓存和查找, 还需要维护自己的缓存结构以及动态规划过程, 因此实现起来非常繁复, 难于优化. 当采用扩展的 N 元文法模型后, 可以将语言模型预测过程和语言模型分数计算过程统一简化如下:

算法 3. 基于扩展 N 元文法模型的语言模型预测

当具有 k 阶语言模型历史状态 h^k 的令牌传递到某预测节点或词尾节点 n 时:

1) 从节点 n 的缓存中查找是否存在扩展语言模型分数 $p'\left(I\left[n\right]|h^k\right)$.

2) 若缓存中存在该分数, 则将分数返回; 否则到存储扩展 N 元文法模型的最小完美哈希表中查找.

3) 若在哈希表中找到该分数, 则将分数返回; 否则采用式 (5) 做回退处理.

3　扩展 N 元文法模型的快速生成和压缩

以扩展 N 元文法模型的方式处理语言模型预测的问题好处为: 1) 所有预测分数在解码前预先计算好, 省去了解码过程中动态计算的时间; 2) 可以采用与语言模型相同的最小完美哈希表作为存储结构, 时、空效率俱佳; 3) 扩展的 N 元文法模型统一了语言模型预测树副本和语言模型本身的表示方式以及分数计算过程, 从而简化了解码器实现. 由于在高阶 N 元文法模型中, 不同的语言模型历史状态的数量非常巨大, 所以扩展 N 元文法模型方法是否可行的核心问题是如何以合理的空间换取时间. 具体而言, 是扩展 N 元文法模型的生成与压缩存储问题.

为描述方便, 本文称在给定一个语言模型历史状态下, 计算得到预测树中所有预测节点分数的过程为生成一个预测树副本. 原理上讲, 预测树副本生成的算法并不唯一. 然而, 由于扩展 N 元文法模型的生成需要为所有在 N 元文法模型中出现的语言模型历史状态生成预测树副本, 所以计算量非常巨大. 如果直接采用原始式 (1) 计算, 常常无法在一个可合理的时间内完成生成操作. 针对这一问题, 本文提出了采用 N 元文法模型的稀疏性加速扩展 N 元文法生成的算法. 该方法将扩展 N 元文法模型的生成时间缩短了上百倍 (见第 4 节).

在压缩存储问题上, 如上文所述, 我们已利用了扩展 N 元文法模型的稀疏性, 将其以稀疏矩阵的方式存储, 并采用最小完美哈希表最小化额外的存储开销和加速查找速度. 本文进一步从三个方面减小扩展 N 元文法模型的存储空间占用: 1) 在生成扩展 N 文法的过程中, 当低阶语言模型预测分数和高阶语言模型预测分数足够接近时, 用低阶预测分数替代高阶预测分数. 相应地, 高阶语言模型预测分数也不再存储. 2) 从解码网络结构上, 通过词尾节点前推, 减小预测节点个数. 3) 通过对模型分数进行量化, 减小存储空间. 通过这三方面的措施, 扩展 N 元文模型的大小可减小约 50 % (见第 4 节).

当然, 因为扩展 N 元文法模型的方法是以空间换时间, 所以当存储空间受限时, 存在由于使用的字典和语言模型过于庞大, 以至于无法一次性全部存储于内存之中的可能性. 在这种情况下, 可以采用 "按需载入" 的方式, 在解码时根据需要动态地将模型载入. 这一方法在若干年前已经提出并应用于卡内基梅隆大学开源的 Sphinx 解码器中[14], 因此为简化问题描述, 在本文中暂不讨论.

3.1　扩展 N 元文法模型的快速生成算法

给定语言模型预测树的一个节点 n 和 k 阶语言模型历史状态 h^k $(1\leq k\leq N)$, 式 (2) 可进一步分解为

$$\begin{aligned} &\pi\left(n,h^k\right)=\max\left\{\pi_D\left(n,h^k\right),\pi_I\left(n,h^k\right)\right\} \\ &\pi_D\left(n,h^k\right)=\pi'\left(W_D\left(n,h^k\right),h^k\right) \\ &\pi_I\left(n,h^k\right)=\pi'\left(W_I\left(n,h^k\right),h^{k-1}\right)\cdot B\left(h^k\right) \end{aligned} \tag{6}$$

式中,

$$W_I\left(n,h^k\right)=W\left(n\right)-W_D\left(n,h^k\right)= \left\{w|w\in W\left(n\right)\text{且}C\left(w,h^k\right)=0\right\} \tag{7}$$

为需要回退计算语言模型分数的词的集合.

在低阶的语言模型预测树副本已经存在的情况下, $\pi_I\left(n,h^k\right)$ 部分可以直接从低阶预测树上获得, 仅需要计算 $\pi_D\left(n,h^k\right)$ 部分. 基于这样的分析, 采用语言模型稀疏性的语言模型预测树副本快速生成算法概括如下:

算法 4. 语言模型预测树副本快速生成

1) 获得低阶语言模型状态 h^{k-1} 的预测树副本 $T\left(h^{k-1}\right)$;

2) 采用 $T\left(h^{k-1}\right)$ 初始化 h^k 的预测树副本 $T\left(h^k\right)$, 初始化预测分数为

$$\pi\left(n,h^k\right)=\pi\left(n,h^{k-1}\right)\cdot B\left(h^k\right)$$

3) 对于任意词尾节点 $n_{we}\in T\left(h^k\right)$, 若 n_{we} 对应的词 $w\left[n_{we}\right]\in W_D\left(Root,h^k\right)$, 则更新 n_{we} 的预测分数为

$$\pi\left(n_{we}, h^k\right) = p\left(w\left[n_{we}\right]|h^k\right)$$

4) 采用动态规划算法, 自词尾节点开始向根节点更新所有 $W_D\left(n, h^k\right)$ 非空的节点的语言模型预测分数.

图 2 给出了算法 4 的示意图. 该算法采用增量方式从低阶语言模型预测树副本计算得到高阶语言模型预测树副本. 考虑到语言模型的稀疏性, 当采用高阶语言模型预测时, 有 $\left|W_D\left(Root, h^k\right)\right| \ll \left|W_I\left(Root, h^k\right)\right|$, 绝大多数 N 元文法概率都需要通过回退的方式计算, 因此计算复杂度大大降低.

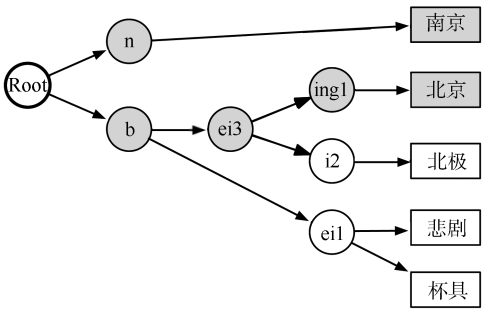


图 2 快速语言模型预测树副本生成算法 (阴影部分表示高阶语言模型历史状态特有的语言模型分数和预测分数, 即增量计算部分)

Fig. 2 Fast LM look-ahead tree copy generation algorithm (Shaded nodes represent higer-order LM sepcific LM scores and LM look-ahead scores, i.e., the increamental computing part.)

基于以上的语言模型预测树副本快速生成算法, 扩展 N 元文法模型可以采用如下的算法生成.

算法 5. 扩展 N 元文法模型快速生成

令 $k = 1, \cdots, N$, 依次遍历语言模型中所有的历史状态 h^k:

1) 对于预测树中的所有词尾节点 n_{we}, 得到:

$$p'\left(I\left[n_{we}\right]|h^k\right) = p\left(w\left[n_{we}\right]|h^k\right)$$

2) 若 $k \leq M$, 则按照算法 4 生成 h^k 的语言模型预测树副本 $T\left(h^k\right)$;

3) 对于 $T\left(h^k\right)$ 中的所有预测节点 n, 若 $\pi\left(n, h^k\right) \neq \pi\left(n, h^{k-1}\right)$, 则得到:

$$p'\left(I\left[n\right]|h^k\right) = \pi\left(n, h^k\right)$$

算法 5 从生成一元文法语言模型的预测树副本开始, 采用增量的方式计算得到二元、三元乃至更高阶的语言模型预测树副本. 由于一元文法模型不考虑语言模型的历史状态, 所以仅有一个预测树副本. 二元文法模型的预测树副本个数与语言模型字典里不同的词的个数相同. 更高阶的语言模型预测树副本的总数与语言模型中不同历史状态的个数相同. 尽管预测树副本个数众多, 但是由于只计算并保存了高阶语言模型特有的预测分数, 所以时间和空间效率很高. 事实上, 判断 $\pi\left(n, h^k\right) \neq \pi\left(n, h^{k-1}\right)$ 进一步地扩展了稀疏性的概念, 即: 当低阶语言模型预测分数和高阶语言模型预测分数足够接近时, 用低阶预测分数替代高阶预测分数. 二者的相近程度可以通过设定相应的门限来判断. 在本文的实现中, 由于对语言模型分数进行了量化表示 (见第 4.2 节), 所以这一门限设定为一个量化误差.

3.2 扩展 N 元文法模型的压缩存储

图 3 给出了一种依据稀疏性存储扩展 N 元文法模型的数据结构. 给定语言模型历史状态 h^k, 每个预测节点或词尾节点需要用一个二元组 $\left(I\left[n\right], p'\left(I\left[n\right]|h^k\right)\right)$ 表示. 二元组中, $I\left[n\right]$ 表示该节点的标识, 通常用一个 32 位整数表示; $p'\left(I\left[n\right]|h^k\right)$ 表示扩展的 N 元文法分数, 一般采用 32 位的浮点数表示. 因此, 为了存储预测树副本中的一个节点, 需要 8 个字节.

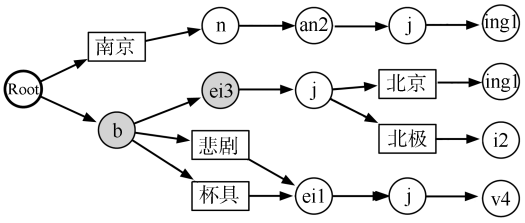
$h1$	$I[n1], p'(I[n1])\|h1$	$I[n2], p'(I[n2])\|h1$	...
$h2$	$I[n1], p'(I[n1])\|h2$	$I[n3], p'(I[n3])\|h2$	...
$h3$	$I[n4], p'(I[n4])\|h3$	$I[n5], p'(I[n5])\|h3$	...
...			

图 3 一种依据稀疏性存储扩展 N 元文法模型的数据结构

Fig. 3 A data structure for extended N-gram storage which makes use of sparseness property

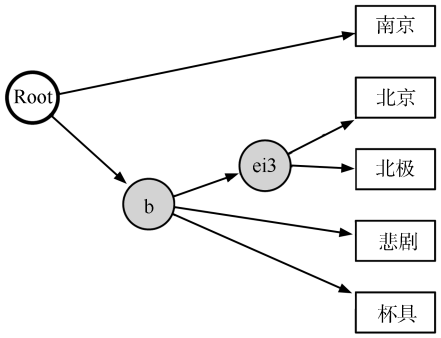
为了减小这一存储大小, 本文首先将分数量化到 8 个比特. 具体做法是统计扩展 N 元文法模型分数的动态范围, 然后在此范围中做 256 级均匀量化. 节点标识的动态范围由预测树的节点数目决定, 节点数目则由发音字典和声学模型决定. 假设发音字典中包含 V 个词, 那么预测节点的总数一般不会超过 $2V$ 个[1]. 考虑到常用的发音字典大小, 24 比特足以表示节点标识. 这样仅需要 4 个字节便可以表示一个预测节点, 空间占用减小了一半.

在预测树结构上, 观察图 1 (b) 可以发现节点 "n"、"ing1"、"i2" 和 "ei1" 的后继只有词尾节点. 将这些节点从预测树中移除可以大大减小预测树规模, 而对解码精度不会造成任何影响, 因为准确的语言模型分数会立即在词尾节点加入. 这一 "移除" 操作对应于原始解码网络之中, 相当于将所有的词尾节点前推, 直至最近的拥有多个子节点的祖先节点. 图 4 给出了这一过程的示意图. 对于同音字 (词) 的情况, 如图中的 "悲剧" 和 "杯具", 可将这两个并联的词尾节点作为一个整体前推.



(a) 前推词尾节点后的音素前缀树

(a) Pronunciation prefix tree with word-end nodes pushed forward



(b) 前推词尾节点后的语言模型预测树

(b) Language model look-ahead tree with word-end nodes pushed forward

图 4　前推词尾节点后的音素前缀树和语言模型预测树示例

Fig. 4　Examples of prefix tree decoding network and language model look-ahead tree with word-end nodes pushed forward

4　实验与讨论

4.1　实验设置

本节采用一个典型的基于动态解码网络的大词汇量连续语音识别器作为基线系统验证本文提出的算法. 此基线系统采用 12 维 Mel 频标倒谱系数 (Mel frequency cepstral coefficients, MFCC) 与 1 维能量作为特征. 声学模型采用 4 000 状态的跨词三音子模型, 每状态有 32 个高斯混合分量. 声学模型训练数据为从 TDT2, TDT3, Broadcast 97, Intel 和微软库中选取的约 220 小时语音数据. 发音字典包含 68 000 个词. 语言模型采用三元文法模型, 采用人民日报和 Gigaword 中文数据库训练得到. 语言模型经过剪枝, 在不影响识别性能的前提下, 保留了 63 079 个一元文法、2 665 603 个二元文法和 6 005 594 个三元文法. 测试数据采用所在实验室采集的约 2 小时 CCTV 新闻广播语音数据.

基线系统中, 解码器采用令牌传递算法[11] 实现, 搜索时采用全局束剪枝和词尾剪枝[12]. 图 5 给出了基线系统搜索算法的流程图. 解码网络中的每一个节点与声学模型中的一个状态相对应, 依据发音字典和声学模型构成前缀树的形式. 语言模型存储采用最小完美哈希表[5]. 语言模型预测采用与式 (2) 相同的动态规划算法. 为了加快存取速度, 基线系统分别为语言模型分数和语言模型预测分数设置了基于节点的缓存[4] (见算法 1 和算法 2). 在下面的实验中, 我们用本文提出的算法替换基线系统中的语言模型分数计算和语言模型预测部分, 并和原始的基线系统做比较.

生成前缀树解码网络.
在网络起始节点放置初始令牌.
按照时间顺序依次处理每一帧特征:
词内令牌传递
对于所有非词尾节点上的令牌, 若路径分数高于束剪枝门限, 则将它传递到后续节点, 并与后续节点上已存在的令牌合并. 如果后续节点是: • 声学节点: 计算并更新路径的声学分数; • 预测节点: 计算并更新路径的语言模型预测分数. 若该路径的语言模型状态是第一次计算预测分数, 则为之生成预测树副本.
词间令牌子传递
对于所有词尾节点上的令牌, 若路径分数高于词尾剪枝门限, 则: • 更新路径的语言模型分数和语言模型状态; • 记录新的词识别假设和词边界信息; • 合并语言模型状态相同的路径(令牌); • 将合并后的令牌反馈到解码网络的根节点, 以便开始下一个词的搜索.
更新剪枝门限
回溯已记录的词识别假设信息, 得到最优识别结果.

图 5　基线系统搜索算法流程图

Fig. 5　Workflow of the baseline system′s search algorithm

在实验中, 本文采用实时率作为语音识别速度的评价指标. 实时率定义为语音识别耗时与被识别语音的总时长的比值, 其物理意义是平均每单位时长的语音 (如: 一分钟) 需要多长时间进行处理. 识别精度指标则采用字错误率 (Character error rate, CER) 以及替代 (Sub)、删除 (Del)、插入 (Ins) 三种错误率和字正确率 (Corr).

4.2　语言模型分数量化的效果

语言模型分数量化主要用于压缩扩展 N 元文法模型的存储空间, 但是由于量化后精度降低, 所以会对识别精度带来一定的影响. 为了便于在后续实验中采用量化后的语言模型并对比识别精度, 我们首先检验语言模型量化对识别精度的影响. 实验采用第 3.2 节中的方法 (暂不做词尾节点前推), 在原始基线系统上进行, 表 1 给出了在语言模型分数

量化前后的识别性能对比. 可以看出, 分数量化后, 识别错误率仅仅上升了 0.06 %, 对识别性能的影响微乎其微. 原始的语言模型文件大小为 266 MB, 未做量化时基线系统用于存储语言模型的内存大小为 165 MB, 量化后仅需要 71 MB, 空间占用减少了约 57 %. 这一实验表明, 尽管语言模型规模很大, 但数值动态范围相对较小, 因此量化对识别精度影响不大, 故而在下面的实验中均采用量化后的语言模型, 扩展 N 元文法模型亦采用量化方式存储.

表 1 语言模型分数量化前后的识别精度比较 (%)

Table 1 Recognition accuracy comparison between decoders with and without language quantitation (%)

语言模型	CER	Sub	Del	Ins	Corr
量化前	14.15	11.43	2.12	0.60	86.44
量化后	14.21	11.48	2.11	0.62	86.41

4.3 词尾节点前推的效果

此实验在第 4.2 节实验的基础上, 通过第 3.2 节中的词尾节点前推方法压缩语言模型预测树的大小. 表 2 给出了词尾节点前推前后语言模型预测节点个数 (不含词尾节点) 对比, 以及相应的扩展 N 元文法模型的大小. 可见, 通过词尾节点前推, 预测节点数减少了约 52 %, 相应地, 扩展 N 元文法模型的大小也减小了约 6 %. 模型大小的减少量远远小于预测节点数的减少量是因为扩展 N 元文法模型本身已采用了增量存储的方式, 即: 给定一个语言模型历史状态 h^k, 只存储无法通过式 (5) 回退计算的部分. 这一增量存储方式本质上源于 N 元文法模型的稀疏性. 由词尾节点前推而移除的预测节点的后继只有词尾节点. 当语言模型非常稀疏时, 绝大多数词尾节点的语言模型分数都要通过式 (3) 回退计算, 因此作为它们直接前驱的预测节点自然也需要通过回退计算其预测分数. 由于这些节点必然不会存储于扩展 N 元文法模型中, 所以模型大小的减少量也较小. 扩展 N 元文法模型的存储中包含了传统 N 元文法模型的部分, 在本实验中占 71 MB. 其余为语言模型预测树部分, 在本实验中共存储了 1 325 291 个预测树副本, 做与不做词尾节点前推的情况下分别占用 460 MB 和 426 MB. 词尾节点前推带来的另一个益处是通过减少预测节点数目, 大大减少解码时语言模型预测的次数, 从而减少语言模型预测在解码过程中的时间开销.

表 2 词尾节点前推前后的预测节点数与扩展 N 元文法模型大小

Table 2 The number of look-ahead nodes and the size of the extended N-gram model before and after word-end node pushing

词尾节点	预测节点数	扩展 N 元文法模型大小 (MB)
未前推	91 395	531
前推后	43 561	497

4.4 扩展 N 元文法模型的生成速度

在第 4.3 节实验的基础上, 本实验对比了采用不同的算法生成扩展 N 元文法模型时的时间效率. 表 3 给出了分别采用定义计算 (式 (1))、采用动态规划方法 (式 (2)) 以及采用本文提出的快速生成算法 (算法 5) 生成扩展 N 元文法模型时的耗时统计. 该数据在一台 Intel Xeon 5504 处理器, 8 GB 内存的计算机上采用单线程计算测得. 从实验结果可以看出, 本文提出的快速生成算法可以极大地提高扩展 N 元文法模型的生成效率, 较之定义计算速度提升了约 133 倍, 较之普遍采用的动态规则方法亦提升了近 14 倍.

表 3 采用不同算法时的扩展 N 元文法模型生成时间

Table 3 Extended N-gram model generation time with different algorithms

算法	生成时间
定义计算	约 24 小时
动态规划	2.5 小时
快速算法	651 秒

4.5 基于扩展 N 元文法模型的识别系统性能

利用第 4.4 节中生成的扩展 N 元文法模型, 本实验检验本文提出的基于扩展 N 元文法模型的语言模型预测算法 (算法 3) 的性能. 图 6 给出了原始基线系统 (分别采用算法 1 和算法 2 进行语言模型分数计算和语言模型预测) 和采用扩展 N 元文法模型的系统 (统一采用算法 3 进行语言模型分数计算和语言模型预测) 的 "实时率 – 字错误率" 曲线. 为公平起见, 这两个系统都采用量化后的语言模型, 并且词尾节点都进行前推处理. 实时率数据在一台 Intel Xeon 5504 处理器, 8 GB 内存的计算机上采用单线程计算测得. 图中一阶语言模型预测仅绘出了一条曲线, 这是因为一阶语言模型预测的语言模型历史状态为空, 因此在解码时只涉及一个预测树副本. 在所对比的两系统实现中, 这一预测树副本都是在解码前生成, 在解码过程中读取. 由于二者的实现完全一样, 所以曲线也是重合的.

从曲线图中可以看出, 不论是二阶还三阶语言模型预测, 采用扩展 N 元文法模型的系统都具有更优的性能表现: 在相同的字错误率下, 实时率较之基线系统可获得 5 ∼ 9 倍的提升; 在实时率较低的情况下, 采用扩展 N 元文法模型的系统可获得更低的字错误率. 当剪枝门限足够宽的情况下, 两系统的字错

误率相同, 这是因为算法 3 只改变了语言模型分数和语言模型预测分数的获取方式, 而并不影响分数计算的精度.

图 7 给出了扩展 N 元文法模型方法和文献 [8] 提出快速语言模型预测算法的 "实时率 – 字错误率" 曲线. 后者是典型的利用 N 元文法稀疏性实时计算语言模型预测分数的方法. 可以看出, 即便是采用了效率极高的方法实时计算预测分数, 在相同的字错误率下, 本文方法仍有约 30 % 的速度优势.

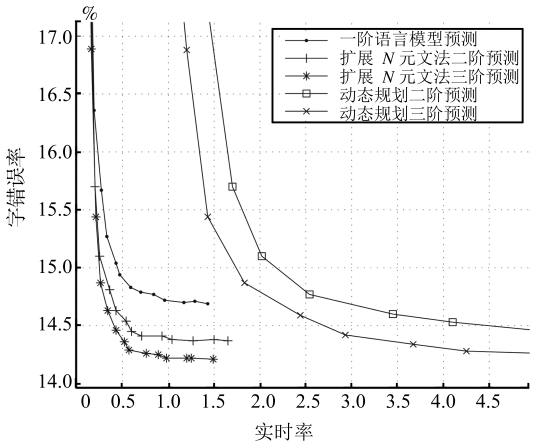


图 6　动态规划方法与扩展 N 元文法模型方法的字错误率 – 实时率曲线

Fig. 6　CER-RTF plots of the dynamic programming method and the extended N-gram model method

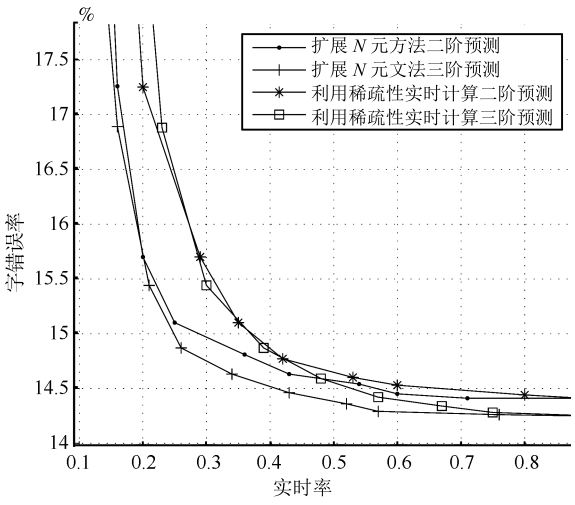


图 7　扩展 N 元文法模型方法和利用稀疏性实时计算方法的字错误率 – 实时率曲线

Fig. 7　CER-RTF plots of the extended N-gram model method and the real-time computing method using the sparseness property

通过对比不同阶语言模型预测的性能, 可以发现采用更高阶的语言模型预测可以获得更优的识别精度. 在识别速度上, 图 6 所示的基线系统由于三阶预测分数计算复杂, 在相同字错误率下的识别速度反而远远不如一阶预测. 而在图 7 所示的实时计算方法中, 当字错误率大于 16 % 时, 相同字错误率下的三阶预测识别速度反而慢于二阶预测. 然而, 在采用扩展 N 元文法模型进行语言模型预测后, 相对于一阶和二阶预测, 三阶预测不仅具有较优的识别精度, 更具有明显的速度优势. 这说明本文提出的算法有效降低了语言模型预测中的计算开销, 修正了传统算法中的缺陷.

5　结论

针对动态网络解码器中的高阶语言模型预测问题, 本文提出了一种采用扩展 N 元文法模型进行快速语言模型预测的算法, 并且给出了一套时间、空间效率可行的实现方案. 实验表明, 采用扩展 N 元文法进行语言模型预测的系统较之于采用动态规划方法进行语言模型预测的传统系统具有更优的识别精度和识别速度. 通过提升解码中语言模型预测的速度, 本文提出的快速语言模型预测算法在一定程度上弥补了动态网络解码器相对于静态网络解码器的劣势, 使得处理静态网络解码器无法承受的超大规模语言和声学模型成为可能.

References

1 Ortmanns S, Ney H, Eiden A. Language-model look-ahead for large vocabulary speech recognition. In: Proceedings of the 1996 International Conference on Spoken Language Processing. Philadelphia, PA, USA: IEEE, 1996. 2095−2098

2 Ortmanns S, Eiden A, Ney H. Improved lexical tree search for large vocabulary speech recognition. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing. Seattle, WA, USA: IEEE, 1998. 817−820

3 Soltau H, Saon G. Dynamic network decoding revisited. In: Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition and Understanding. Merano, Italy: IEEE, 2009. 276−281

4 Cardenal-López A, Diéguez-Tirado P, Garcia-Mateo C. Fast LM look-ahead for large vocabulary continuous speech recognition using perfect hashing. In: Proceedings of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing. Orlando, FL, USA: IEEE, 2002. 705−708

5 Li X L, Zhao Y X. A fast and memory-efficient N-gram language model lookup method for large vocabulary continuous speech recognition. *Computer Speech and Language*, 2007, **21**(1): 1−25

6 Huijbregts M, Ordelman R, de Jong F. Fast N-gram language model look-ahead for decoders with static pronunciation prefix trees. In: Proceedings of the 9th Annual Con-

ference of the International Speech Communication Association. Brisbane Australia: ISCA, 2008. 1582−1585

7 Chen L Z, Chin K K. Efficient language model look-ahead probabilities generation using lower order LM look-ahead information. In: Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing. Las Vegas, Nevada, USA: IEEE, 2008. 4925−4928

8 Nolden D, Ney H, Schluter R. Exploiting sparseness of backing-off language models for efficient look-ahead in LVCSR. In: Proceedings of the 2011 IEEE International Con-ference on Acoustics, Speech and Signal Processing. Prague, Czech: IEEE, 2011. 4684−4687

9 Mohri M, Pereira F, Riley M. Speech recognition with weighted finite-state transducers. *Handbook on Speech Processing and Speech Communication, Part E: Speech Recognition*. Heidelberg, Germany: Springer-Verlag, 2008. 559−584

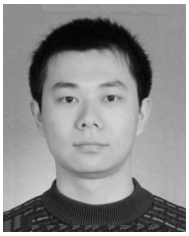
10 Young S J. A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine*, 1996, **13**(5): 45−57

11 Young S J, Russell N H, Thornton J H S. Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems. Technical Report CUED/F-INFENG/TR38, Engineering Department, Cambridge University, USA, 1989

12 Pylkkönen J. New pruning criteria for efficient decoding. In: Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech 2005). Lisboa, Portugal: IEEE, 2005. 581−584

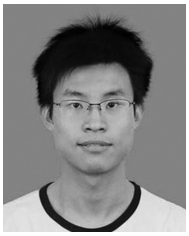
13 Chen S F, Goodman J. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, USA, 1998

14 Ravishankar M K. Efficient Algorithms for Speech Recognition [Ph. D. dissertation], Carnegie Mellon University, USA, 1996



单煜翔 清华大学电子工程系博士研究生. 主要研究方向为语音识别, 关键词检测, 说话人识别. 本文通信作者.
E-mail: syx06@mails.tsinghua.edu.cn
(**SHAN Yu-Xiang** Ph. D. candidate in the Department of Electronic Engineering, Tsinghua University. His research interest covers speech recognition, keyword spotting, and speaker recognition. Corresponding author of this paper.)

陈 谐 清华大学电子工程系硕士研究生. 主要研究方向为语音识别, 关键词检测.
E-mail: chenxie09@mails.thu.edu.cn
(**CHEN Xie** Master student in the Department of Electronic Engineering, Tsinghua University. His research interest covers speech recognition and keyword spotting.)



史永哲 清华大学电子工程系博士研究生. 主要研究方向为语音识别, 语言模型, 音频索引和检索.
E-mail: yongzhe87@163.com
(**SHI Yong-Zhe** Ph. D. candidate in the Department of Electronic Engineering, Tsinghua University. His research interest covers speech recognition, language modelling, and audio indexing and searching.)

刘 加 清华大学电子工程系教授. 主要研究方向为信号处理, 语音识别, 语音合成, 语音编码和多媒体通信.
E-mail: liuj@tsinghua.edu.cn
(**LIU Jia** Professor in the Department of Electronic Engineering, Tsinghua University. His research interest covers signal processing, speech recognition, speech synthesis, speech coding, and multimedia communication.)