文章编号:2095-6134(2019)01-0109-06

# WFST 解码器词图生成算法中的非活跃节点检测与内存优化*

丁佳伟[1],刘 加[1†],张卫强[1],冯运波[2],刘利军[2],于 乐[2]

(1 清华大学电子工程系,北京 100084;2 中国移动通信信息安全管理与运行中心,北京 100053)

(2017 年 12 月 22 日收稿;2018 年 3 月 2 日收修改稿)

**Ding J W, Liu J, Zhang W Q, et al. Inactive-node detection and memory optimization in WFST decoder lattice generation algorithm[J]. Journal of University of Chinese Academy of Sciences, 2019,36(1):109-114.**

**摘 要** 解码器引擎是语音识别系统的核心模块,而基于加权有限状态机(WFST)的解码器则是解码器的一种典型形式。分析静态 WFST 解码器在实际应用中的资源占用问题,提出一种在解码和词图生成过程中通过检测非活跃节点动态回收系统资源的策略。最后,在 OpenKWS 15 数据集上进行实验,证明该策略使解码器的内存消耗比不回收系统资源的解码器降低 75% 左右。

**关键词** 语音识别解码器;加权有限状态机;工程应用;内存回收

**中图分类号**:TN912 **文献标志码**:A **doi**:10.7523/j.issn.2095-6134.2019.01.015

# Inactive-node detection and memory optimization in WFST decoder lattice generation algorithm

DING Jiawei[1], LIU Jia[1], ZHANG Weiqiang[1], FENG Yunbo[2], LIU Lijun[2], YU Le[2]

(1 *Department of Electronic Engineering, Tsinghua University, Beijing* 100084, *China*;
2 *China Mobile Information Security Center, Beijing* 100053, *China*)

**Abstract** Decoder is the core module of speech recognition system, and the decoder based on the weighted finite-state transducers (WFST) is a typical form of decoder. We analyze the resource occupation of WFST-based static decoder in practice, and propose a strategy for dynamical recovery of system resources by detecting inactive nodes during decoding and lattice generation. Finally, we carry out experiments on the OpenKWS 15 dataset to show that the decoder with this strategy consumes about 75% less memory than decoders that do not reclaim system resources.

**Keywords** speech recognition decoder; WFST; engineering application; memory recycling

大词汇量连续语音识别(large vocabulary continuous speech recognition, LVCSR)技术是目前互联网诸多语音识别系统的主流技术,在语音检索、语音识别输入法、关键词检测以及人机交互领域均有广泛的应用。语音识别解码器是语音识别系统的核心模块,在语音识别中起到举足轻重

* 国家自然科学基金(U1836219)资助

† 通信作者,E-mail:liuj@tsinghua.edu.cn

的作用[1]。它以由预处理的语音信号提取出的特征作为输入,通过特定的解码算法对输入特征进行解码,获得解码结果。

语音识别中的解码器根据解码网络的构造方式可以分为两大类,即动态网络解码器(简称动态解码器)与静态网络解码器(简称静态解码器)。在动态解码器中,解码的搜索网络是在解码过程中通过预测算法动态生成的,一般根据动态生成的声学和语言模型实现解码搜索。动态解码器的一个典型代表是使用词法前缀树(lexical prefix tree)的方法[2]。动态解码器的特点是内存消耗较少,而且可以几乎不受限制地使用庞大的高阶语言模型。德国亚琛工业大学开发的 RWTH ASR 系统是具有代表性的动态解码器[3]。

在静态解码器中,系统提前将声学模型与语言模型静态结合,构造一个完整的解码网络,解码器所有的过程均在该静态网络中进行,并生成完整的搜索空间。其特点是解码速度相较动态解码器大大提升,但由于其生成的解码网络相当庞大,搜索过程中所需的搜索空间也随之增大,因此在实际应用中会占用大量的机器资源。静态解码器的代表是约翰霍普金斯大学的 Kaldi 系统[4]和瑞士 IDIAP 的 Juicer 系统[5]。静态解码器的一个典型模式是基于加权有限状态机(weighted finite state transducers, WFST)的解码器[6-7]。本文从工程应用的角度出发,对基于 WFST 的语音识别解码器进行资源优化,明显改善了其资源占用的情况。

## 1 WFST 语音识别解码器

加权有限状态机是自动机理论的一个扩展[8],在语音识别解码网络的构建上得到了较好的应用。一个典型的语音识别网络构成如下式所示:

$$\mathrm{fact}(\pi_{\varepsilon}(\min(\det(H \circ C \circ \det(L \circ G))))). \tag{1}$$

式中:"∘"表示两个 WFST 的合成;det、min、$\pi_{\varepsilon}$、fact 分别为确定化、最小化、空边去除以及简化运算,其目的是优化最终的网络结构,使其尽可能化简,且不改变原始网络的功能。式子中 $G$ 表示语言模型,定义词到句子的映射;$L$ 表示发音词典,定义音素到词的映射;$C$ 定义上下文相关音素到单音子的映射;$H$ 定义 HMM 状态到上下文相关音素的映射。式子中的 $H$、$C$、$L$、$G$ 均代表一个 WFST,通过式(1)中一系列的操作将其级联合并,最终就可以获得 HMM 状态到句子的映射,以语音信号对应的 HMM 状态序列作为输入时,就可以直接得到对应的句子(词序列),也就是解码器的功能。

生成解码网络后,就需要针对输入特征在网络中进行时间同步的搜索,寻找与输入信息最匹配的状态序列。这是一种广度优先的策略,通常使用维特比束搜索算法(Viterbi beam search)和令牌传递算法(token passing)进行搜索。在搜索结束后,得到的全局最优路径对应的词序列就是解码器的解码结果,称之为"1-best"结果。由于解码器解码过程中会同时生成一个包含全部中间节点的网络(称之为令牌图),因此除 1-best 结果以外,还可以通过对搜索生成的令牌图进行后处理,获得包含更多信息的词格[9]或词图[10]等,用于进一步后处理后的其他应用[11-12],如关键词检测等。

WFST 解码器中的词图(lattice)即是基于令牌图生成的。词图的表现也为 WFST 模式,边上的输入和输出为词标签,并附有对应的权重。文献[10]提出一种根据 WFST 解码器搜索结果直接生成词级别词图的算法,首先在令牌图中根据广度优先搜索确定词边界和对应的时间信息,并使用由词对假设[13]引申而来的词对过滤算法去除词图中的大量冗余,再利用确定化算法就可以得到确定化的,带有准确时间标签的词图。从上文可以看出,词图的质量直接由令牌图的质量决定。

另外,根据后文需要,本文在此简要介绍解码过程中的实际搜索方式。

语音识别解码器的搜索过程基于事先生成的 WFST 解码网络,该网络以 HMM 状态序列作为输入,以词标签作为输出。网络中同时存在以 HMM 状态作为输入的非空边,和无任何输入的空边。实际的搜索过程分为非空边搜索与空边搜索两种。非空边搜索是指,根据某一个语音帧的特征输入,对当前所有待搜索节点进行非空边的对应传递,一般只会向后传递至多一条边的距离;空边搜索则是指,对所有搜索节点进行空边的自然扩张和传递,直到无法继续基于空边扩张为止。在实际的搜索过程中,首先应该对解码网络进行初始化,即进行一次从初始节点开始的空边搜索。之后根据输入语音对应的帧特征序列,交替进行空边和非空边的搜索,完成整个解码的搜索过程。在整个解码的搜索空间生成后,便可通过对权重

最优的终止节点进行回溯操作,获得最优路径的解码结果。

上文中提到,静态解码器由于其网络庞大,其需求的内存资源也相对较多。但在实际的应用过程中发现,搜索过程中生成的中间节点所占用的资源要远远超出静态解码网络的占用。其原因是中间节点中包含大量搜索的中间信息,例如前向累计权重和后向累计权重等等,这些信息的加入便造成节点空间相较静态网络会占用更多的空间。本文的内存优化策略即针对节点空间进行,最终将解码过程中的节点空间占用缩减至未优化系统的 25% 左右。

## 2　解码器的内存优化策略

本文提到的语音识别解码器中的内存优化操作,实际上是指解码搜索过程中的内存回收策略。由于解码器的解码过程是基于维特比束搜索的令牌传递算法,故而其在搜索过程中会不可避免地产生大量非活跃的节点(令牌)。为节省资源,在搜索过程中,应该实时地检测这些非活跃节点的出现,并把其占用的空间及时地释放。但直接进行非活跃节点的释放会造成内存占用的碎片化,起不到节省机器资源的作用,因此还需要将当前的活跃节点重新排列,以满足应用中对资源的实际需求。

### 2.1　数据结构的调整和优化

一些开源的解码器工具中已经存在相应的内存回收策略,例如 kaldi。但 kaldi 中采用的分层结构在内存回收过程中会将内存碎片化,并且其占用的机器资源也并不固定,无法满足实际的工程需求。图 1 表示 kaldi 系统解码器大致的数据结构。

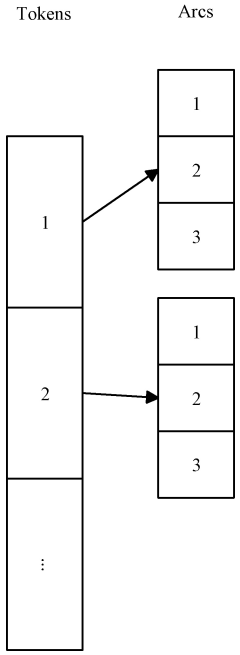


**图 1　Kaldi 中的数据结构简图**

**Fig.1　Data structure diagram in Kaldi**

在工程应用中,一般希望解码器占用固定且尽量小的机器资源,针对这种需求,考虑到 WFST 的节点-边结构,可设计如图 2 所示的数据结构,其中节点空间和边空间均为连续的数据块,而每一个节点则对应边空间中的连续块,具体表现为对每一个节点添加对应边在边空间内的起始位置和终止位置。这样既可保证后续的回收和重排操作仍然在该空间中进行,且经过回收后仍能保持如图所示的连续结构,保证解码器的资源稳定性。

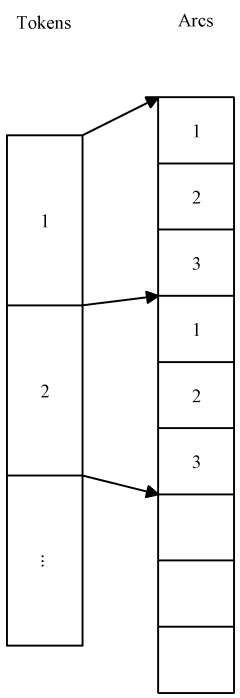


**图 2　本文设计的数据结构简图**

**Fig.2　Our data structure diagram**

### 2.2　数据结构的动态维护

本文提出的数据结构虽然能够保证解码器的资源稳定性,但其要求节点和边空间中的节点和边必须按照起始位置升序排列。搜索过程生成的中间节点按照生成顺序排列,可以直接满足该要求;但由于搜索过程由空边搜索和非空边搜索组成,因此生成的边并不按照起始位置顺序排列,因此需要适时地对边按照起始位置进行排序,保证数据结构不被破坏。为保证搜索过程的正常进

行,这样的排序操作应当在每一个数据帧都进行一次。下面提出一个排序策略:在每一帧的非空边搜索后,对上一帧的空边搜索和该帧的非空边搜索生成的边按照边起始位置进行升序排序,并存放在边空间中的相同位置。基于这种排序策略即可实时地维护整个边空间的数据结构,下面对其进行证明:

该策略的可行性基于两个命题:第一,经过一次非空边搜索时,所有生成的新节点并不产生以这些节点为起始节点的边;第二,经过一次非空边搜索时,所有既存的节点的全部可能边均已生成完毕。根据第 1 节提到的空边与非空边搜索的概念,不难发现这两个命题显然是成立的。如图 3 所示,图 3(a)的节点和边空间经过一次空边搜索和一次非空边搜索后,得到的数据结构分别如图 3(b)和图 3(c)所示。可以看出,在空边搜索结束时,第 $n$ 帧的节点全部生成完毕,且在下一次非空边搜索结束时,第 $n$ 帧的边空间也全部生成完毕。因此在某一帧的非空边搜索后,其之前生成的所有节点和边均已稳定,此时进行数据结构维护在保证整个边空间有序的同时,也避免了重复排序带来的时间浪费。

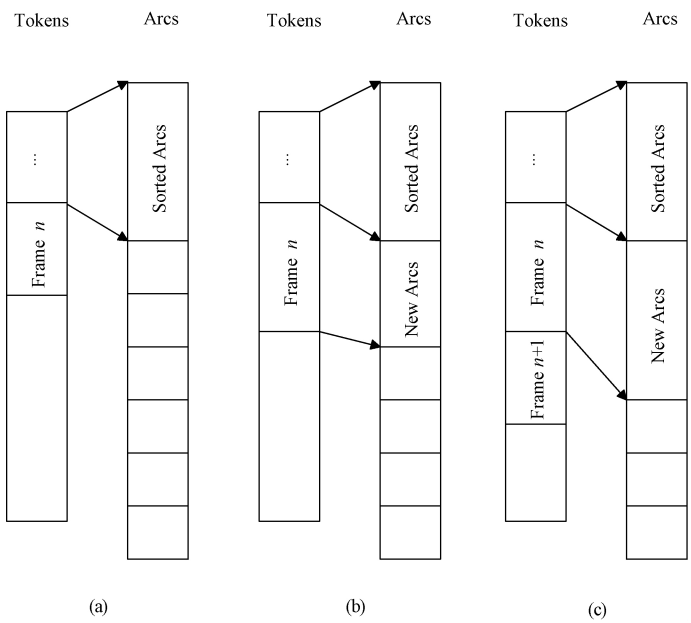


**图 3 第 $n$ 帧的解码搜索过程示意图**

**Fig.3 Diagram of the $n$-th frame decoding search process**

## 2.3 解码过程中的内存回收策略

如上文所提到的,内存的回收操作应该在解码搜索的过程中实时进行,因此本文提出如下的回收策略:以一个给定的时间长度为间隔,对每一个间隔对应的时间段内所生成的所有节点和边进行活跃性检测和回收,取出该段时间内产生的非活跃节点与边并将当前的所有活跃节点与边在数据结构内重新调整,以满足后续搜索过程的需求。

解码过程中的非活跃节点,确切来说应该称作“失活节点”。它一般在解码搜索中被作为活跃的新节点生成,但在后续的搜索中发现其无法继续向下搜索,如果不对其进行处理,其将继续占用内存资源直到解码完毕为止。在 WFST 解码器框架下,这些非活跃节点会一直保留在令牌图中,使令牌图结构肥大化。由于前文提到,词图生成算法的基础是令牌图,而非活跃节点的检测与回收即是对令牌图进行的结构精简操作,因此其对与词图的生成算法也具有正面的作用。

### 2.3.1 非活跃节点的检测

在回收过程中,对于非活跃边的检测相对容易,因为只要其起始或终止节点中至少含有一个非活跃节点,那么其就应被判为非活跃边。而非活跃节点的判定相对复杂,由于段内最后一帧的后续状态未知,因此只能假设其均为活跃节点。对于其之前的段内节点,本文在此引入两种动态检测非活跃节点的策略。

一种是连通性检测。这种策略的判据是,无法通过任何路径到达当前最终节点的节点即被标注为非活跃节点。具体的操作方式为:在生成的解码网络中,对某一个间隔中的时间段,以该时间

段内最后一帧的所有活跃节点作为起始点进行回溯,无法被回溯到的节点即是应该被回收的非活跃节点。这种方法简单直观,但因其未考虑语音识别解码器中维特比束搜索算法的特性,因此并不能尽可能充分地对非活跃节点进行标注。下面介绍一种更加适用于语音识别解码过程的回收策略。

在这种策略中,对每一个中间节点添加一个“额外权重”,其计算方式如下:

$$C_{t_i} = \min(\{W(\{L \mid t_i \text{ in } L\})\}) - \min(\{W(\{L\})\}), \quad (2)$$

式中:$C_{t_i}$ 是第 $i$ 个节点 $t_i$ 的额外权重,$W(\{L\})$ 表示对路径集合 $L$ 求权重获得的权重集合。简单说,一个节点的额外权重,是指其所在的所有完整路径的最小权重和当前全部路径的最小权重的差值。图 4 给出一个节点空间的路径示例,其中最优的路径是 0-1-3-6-9,其权重为 1.3,则这条路径上的所有中间点,即 1、3、6 的额外权重即为 0;而假设要计算节点 4 的额外权重,则需要找到包含该节点的最优路径,即 0-1-4-7-6-9,其权重为 1.4,故节点 4 的额外权重即为 1.4-1.3=0.1。这一权重描述一个节点在整个节点空间中的可信程度。如果该权重越大,则表示该节点的可信度越低,此时即可人为定义一个阈值,将超过该值的节点也视为非活跃节点。由于无法到达最终节点的节点该额外权重为无穷大,因此这种策略最终包含连通性检测的全部非活跃节点,并拥有更好的标记性能。本文应用的即是这种标记方法。一些开源工具中也包含相近的额外权重计算方法,例如 kaldi 等。

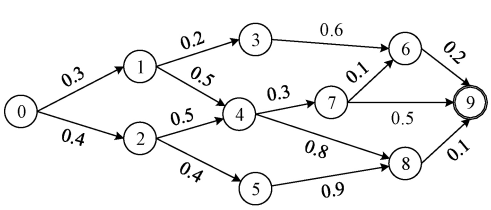


**图 4　解码器搜索空间示例**

**Fig.4　Decoder search space example**

### 2.3.2　段时间长度的选择

本节给出回收长度段的选择方案。回收长度段是指,对某段时间内生成的节点和边进行回收操作时该段时间的长度,这里记作 $T_n$。$T_n$ 的长度选取一般根据实际需要进行调节,但是对固定的解码系统而言,其一般拥有一个最优值。下面对其原因进行说明。

首先考虑计算速度。较大的 $T_n$ 段内的节点和边数较多,每次回收的计算量较大但总回收次数少;反之较小的 $T_n$ 每次回收的计算量较小但总回收次数多,因此在计算量方面一般难以产生明显的差别;如果考虑回收数量,较小的 $T_n$ 由于可操作的段内节点数和边数均较少,因此其回收的效率并不高;较大的 $T_n$ 可以获得很高的回收效率,但较大的时间容许量很可能导致节点和边空间在段内时间内严重膨胀,造成短时的大量资源占用。因此一定存在一个 $T_n$ 的取值,既能防止短时的资源膨胀,又能获得较高的回收效率。而在实际应用中,可以进行调节最终获得 $T_n$ 的最优取值。

## 3　实验和结果

本文的实验基于美国国家标准与技术研究所(National Institute of Standards and Technology, NIST)在 2015 年的语音关键词检索(OpenKWS)评测任务。在实验中,以 NIST OpenKWS 2015 斯瓦西里语关键词检索的评测数据集作为实验数据。其中,供训练的语料为受限低资源数据集(very limited language pack, VLLP),仅包含原始带标注语料 3 h 和无标注语料 40 h,开发集语料 10 h。实验首先采用该数据集训练的 DNN 声学模型,并使用 fbank 特征基础上扩展 pitch 特征作为声学特征;之后对声学模型、发音词典以及一个三音子语言模型进行联合建模获得静态 WFST 解码网络,并使用上文提到的搜索和词图生成算法对测试数据进行解码并生成词图。其中测试数据包含 10 813 条语音,分别来自 60 个不同的说话人。评测的评价指标来自关键词检测中的最大查询词加权值(maximum term-weighted value, MTWV),该值越大说明系统的性能越好。本实验只对解码器的资源占用情况和运行效率进行评估,MTWV 仅作为验证算法准确性的依据。

实验首先按照上文所示的解码器系统结构构建基线系统,并在其基础上添加内存回收策略。实验的评价指标为内存占用量和实时率(RTF)两项,其中内存占用量使用解码过程中的实时最大节点数(MaxTokens)和边数(MaxArcs)表示,其值越小说明解码器占用内存越低;实时率为解码器运行耗时与输入语音长度的比值,其值越小说明解码效率越高。对非活跃节点的标记策略采用

2.3.1 中添加额外权重的方法,判定阈值固定为 8.0,调整不同的 $T_n$ 值后的实验结果如表 1 所示。

**表 1 实验结果**

**Table 1 Experimental results**

| $T_n$ /frames | MaxTokens | MaxArcs | RTF | MTWV |
|---|---|---|---|---|
| BaseLine(Inf) | 26 865 293 | 44 935 189 | 0.389 | 0.463 3 |
| 100 | 8 365 678 | 15 535 941 | 0.444 | 0.463 3 |
| 200 | 7 433 409 | 13 649 701 | 0.455 | 0.463 3 |
| 300 | 6 776 321 | 12 281 351 | 0.457 | 0.463 3 |
| 400 | 7 222 604 | 13 950 476 | 0.450 | 0.463 3 |

表中的结果表明,在实时率方面,选取不同的 $T_n$ 值对系统实时率的影响并不大,由于数据结构维护和内存回收策略的计算耗时引入,因此实时率相对基线系统( $T_n$ 等效地为无穷大)均有损失,均在 15% 左右。而在内存占用量上, $T_n$ 的选取影响显著:随着 $T_n$ 值的逐渐增大,系统的内存占用量(定量映射为实时最大节点数和边数)呈现明显先降后升的趋势;另外从表中看出,当 $T_n$ = 300 时系统的内存占用量最低,为基线系统的 25.2%,系统占用内存得到极大缩减。而且根据其在表中的变化规律,也能说明在 $T_n$ = 300 帧(即 3 s)左右的范围内应存在一个令该系统内存占用量最低的值。而 MTWV 的结果没有发生变化,证明系统是稳定且正确的。

另外,本文提出的回收策略仅应用在解码器的搜索阶段,其只作用于 WFST 结构的令牌图上,对解码器的具体识别任务(例如语种,语言模型建模等)没有实质的约束。因此对于其他的 LVCSR WFST 静态解码器,该方法也能起到标记并回收非活跃节点的作用,这也体现了该方法的泛用性。

## 4 结论

本文从应用角度出发,针对基于 WFST 的语音识别解码器提出在解码搜索过程中实时地检测并回收非活跃节点和边的策略,相比无回收的解码系统可以获得显著的机器资源节约。最后在 OpenKWS 15 评测上的实验中,表明该回收策略相较不进行回收策略的基线系统获得约 75% 的内存节省,且在实时率方面仅有 15% 的损失。添加内存回收策略的解码器系统能够更加满足工程以及实际应用的需要,同时适合单个 PC 运行以及服务器端的多线程应用。

## 参考文献


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

[2] Rybach D, Ney H, Schluter R. Lexical prefix tree and WFST: a comparison of two dynamic search concepts for LVCSR[J]. Audio, Speech, and Language Processing, IEEE Transactions on, 2013, 21(6): 1 295-1 307.

[3] Rybach D, Gollan C, Heigold G, et al. The RWTH Aachen University open source speech recognition system[C]//Tenth Annual Conference of the International Speech Communication Association. ISCA, 2009: 2 111-2 114.

[4] Povey D, Ghoshal A, Boulianne G, et al. The Kaldi speech recognition toolkit[C]//IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, 2011.

[5] Moore D, Dines J, Doss M M, et al. Juicer: a weighted finite-state transducer speech decoder[C]//International Workshop on Machine Learning for Multimodal Interaction. Springer Berlin Heidelberg, 2006: 285-296.

[6] Hori T, Nakamura A. Speech recognition algorithms using weighted finite-state transducers[J]. Synthesis Lectures on Speech and Audio Processing, 2013, 9(1): 1-162.

[7] Mohri M, Pereira F, Riley M. Weighted finite-state transducers in speech recognition[J]. Computer Speech & Language, 2002, 16(1): 69-88.

[8] Hopcroft J E, Motwani R, Ullman J D. Introduction to automata theory, languages, and computation[J]. ACM SIGACT News, 2001, 32(1): 60-65.

[9] 李伟, 吴及, 王智国. 一种快速的语音识别词图生成算法[J]. 清华大学学报(自然科学版), 2009 (S1): 1 254-1 257.

[10] Pan G, Lu C, Liu J. An exact word lattice generation method in the WFST framework[C]//Information Science and Technology (ICIST), 2016 Sixth International Conference on. IEEE, 2016: 394-398.

[11] Ljolje A, Pereira F, Riley M. Efficient general lattice generation and rescoring[C]//EUROSPEECH. ISCA, 1999: 1 251-1 254.

[12] Liu X, Chen X, Wang Y, et al. Two efficient lattice rescoring methods using recurrent neural network language models[J]. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2016, 24(8): 1 438-1 449.

[13] Ortmanns S, Ney H, Aubert X. A word graph algorithm for large vocabulary continuous speech recognition[J]. Computer Speech & Language, 1997, 11(1): 43-72.