第六届全国人机语音通讯学术会议，203-206 页，2001 年 11 月 20-22 日，深圳

一种基于 HTK 的词图搜索算法

罗春华，张继勇，郑方，徐明星

清华大学计算机系人工智能与系统国家重点实验室
语音技术中心
springofcn@yahoo.com.cn

摘要

在连续语音识别中，为了能够在搜索的过程中实现更有效的剪枝策略，必须充分应用语言模型提供的信息。对于在一遍搜索过程中同时使用声学模型和语言模型的搜索算法而言，虽然能够获得比较高的识别率，但是耗时比较多。为此，本文实现了一种能够在后续处理过程中有效地利用 Trigram 语言模型和更复杂语言模型信息的词图搜索算法。它是基于 HTK 平台的。文中对词图的数据结构和词图的生成算法给出了非常详细的论述，基于给定词图的搜索算法也在文中给出。实验表明，词图搜索算法能够充分地利用语言模型提供的信息指导搜索而且速度很快。

1. 引言

在语音识别的过程中，如果我们能够利用一些知识来指导搜索过程，一方面可以有效的提高搜索速度，另一方面可以提高识别率。在 HTK 中搜索使用的是 Token Passing[1]算法。该算法由一个识别网络进行控制，识别网络的生成充分利用了词典和 HMM 模型提供的信息。在 Token passing 算法中，Token 表示网络中从时刻 0 扩展到时刻 t 的部分路径。时刻 0，在每个可能的开始结点上都有一个 Token，随着时间的推移，它们沿着识别网络的弧进行传播，直到到达一个 HMM 的退出状态。如果一个结点有几个出口，则该 Token 会被拷贝以保证多条路径能够并行的得到扩展。当 Token 沿着弧和结点进行传播时，它的对数似然度分数会加上相应的转移概率和输出概率。当 Token 沿着网络进行传播时，它必须保留它的历史路由的记录。历史记录保留的详细程度取决于识别输出的需要，通常保留词边界的信息就能够满足大多数应用的需要。

由于 HTK 是一个实验平台，因此它的算法没有考虑到实际应用的需要。为了能够保证该算法的通用性，它的数据结构设计得很复杂。一个典型的例子就是它的识别网络。对于汉语连续语音识别，通常词表的大小为 5－6 万词，按照它的算法构造的网络非常庞大，因此要想在它的上面使用 Trigram 语言模型几乎是不可能的。为了能够建立一个实用的识别器，我们在 HTK 的基础上，借鉴了它的一些好的思路，实现了自己的识别器。通过 Trigram 的加入，取得了比较好的效果。我们算法的基本思路是首先在一遍搜索的过程中使用声学模型和 Bigram 语言模型来产生一些可能的词候选，利用这些词候选来产生词图，然后在后续的过程中利用 Trigram 对词图里的可能路径进行分数重估，最后输出具有最大似然度的路径作为识别候选。

本文将按照如下方式组织：在论文的第二部分，将介绍利用复杂语言模型的词图搜索算法的框架；基于该框架结构的词图重估算法将在第三部分给出；论文的第四部分为采用这种词图搜索算法的实验结果；相关的结论将在论文的第五部分给出；论文的最后给出相关的参考文献。

2. 词图搜索算法的框架

采用词图搜索算法的语音识别系统的框架结构如下图所示：

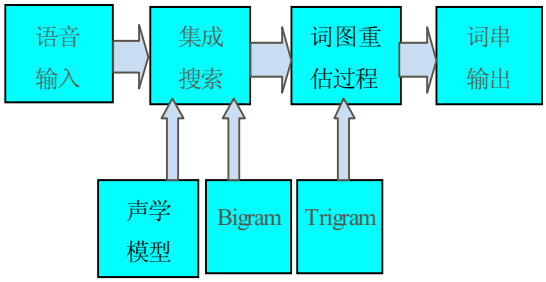

图 1. 采用词图搜索算法的连续语音识别系统的框架

语音输入后，经过特征提取，首先经过集成搜索（Integrated search）模块，这个过程利用声学模型来产生声学候选，同时利用 Bigram 语言模型来剪枝，该模块产生的词图（Word graph）被词图重估模块（Word graph rescoring）用来进行后处理，在这个过程中可以充分地利用 Trigram 语言模型提供的信息，最后我们可以得到识别出来的中文词序列。

2.1. 集成搜索模块

该模块的主要功能是利用声学模型和 Bigram 语言模型提供的信息来产生词候选，便于词图的生成。为了能够进行上下文相关建模，我们选取了声母和韵母作为语音识别基元，在此基础上，利用 HTK 提供的训练工具 HINIT，HREST 和 HEREST 来训练 TriIF 模型（请注意，这里的 IF 指的是声母或者韵母）。由于汉语的常用词大约有 50000 多个，因此我们采用树型结构来组织这些词，它可以有效的合并具有相同前缀的词。每个词的模型通过连接对应的 TriIF 模型来获

得。如词“饱餐”由“b+ao”，“b-ao+c”，“ao-c+an”，“c-an”组成，当然如果考虑上下文相关（即 Cross-word）[2]，则词的模型要考虑到它的前驱词的具体内容才能最终确定。

在搜索的过程中，充分利用了目前比较常用的一些剪枝技术，如声学剪枝、语言模型剪枝和直方图剪枝，同时还使用了两种 Look-ahead 技术，分别是语言模型 Look-ahead[3]和 IF Look-ahead。语言模型 Look-ahead 技术的采用是为了能够比较早地使用语言模型提供的信息来指导搜索。为了确保算法能够比较高效地运行，我们利用的是 Bigram。至于 IF Look-ahead 技术，它的引入是为了更好的进行声学层的剪枝，从而提高识别的速度。

2.2. 词图的生成

在集成搜索过程中我们会保留一些可能的候选，这些候选被用来生成词图。在我们的搜索算法中，词图采用六元组 (a,e,w,s_w,t_a,t_e) 来表示，其中 a,e 是词 w 的逻辑开始和结束结点，t_a,t_e 为逻辑开始和结束结点对应的时间帧序号，s_w 为词 w 在区间 $[t_a,t_e]$ 上的 HMM 分数的对数值。在图 2 中我们给出一个词图的例子。

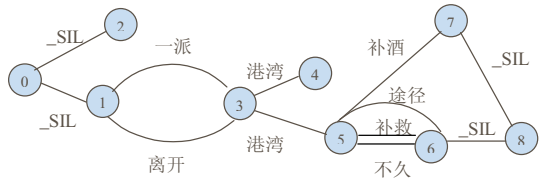

图 2. 词图的例子

我们可以看出词图具有以下特点：

- 在同样的时间区间内，能够保留多个候选，如 1－3 之间的“一派”和“离开”。
- 同一个词可能有不同的结束时刻，如 3－4 之间的“港湾”和 3－5 之间的“港湾”。
- 同一个词可能有不同的开始时刻，如 7－8 之间的“_SIL”和 6－8 之间的“_SIL”。

在对词图做具体描述时，我们采用了如下的数据结构：

词图的数据结构为：

- Graphnode node[MAXGRAPHNODE] 记录结点的信息
- Grapharc arc[MAXGRAPHARC] 记录弧的信息
- int link[MAXGRAPHARC] 记录每个结点发出去的弧
- int offset[MAXGRAPHNODE+1] 记录每个结点发出去的弧的偏移量
- int nodeNum 记录词图中总的结点数目
- int arcNum 记录词图中总的弧数目

其中 MAXGRAPHARC 和 MAXGRAPHNODE 为两个预先定义好的常量，分别表示词图所能拥有的弧的最大数目和结点的最大数目。

Graphnode 为记录词图中结点信息的数据结构，其具体的表示形式为：

- int nodeIdx 结点的索引
- int timeId 结点的时间戳，和时间帧相对应
- float maxScore 记录通过一个结点的所有路径的最大分数

Grapharc 为记录词图中弧的信息的数据结构，其表示形式为：

- float likeScore 记录词的似然度得分
- int wordIdx 词的索引，该索引是从词典中获得的词的编号
- char wordString[20] 记录表示词的中文字符串的数组

构造词图所需要的所有信息都在集成搜索模块搜索过程中保存的 Path 数据结构里。在 HTK 中，Path 数据结构的定义如下：

- path *prev 记录前一个词的记录
- LogDouble like 记录累计的似然度得分，注意这里不仅包括声学模型得分，也包括语言模型分数
- LogFloat lm 记录累计的词的语言模型得分
- DWORD nodeidx 记录结点在词法树中对应的结点编号
- int frame 记录词结束的时刻（用帧数表示）
- Boolean used 是否被引用的标志位
- int usage 被下一条路径引用的次数
- path *link 链表中的下一条路径
- path *knil 链表中的前一条路径

由该数据结构可以看出，路径信息组成了一个双向链表，这样做的主要目的是为了方便地对链表进行插入、删除和查找等操作。如果我们从那些能够到达最后时间帧的路径开始回溯，则一般能够找出大约 10 条左右的路径。在一遍搜索过程中，通常是从这些路径中挑选出具有最大似然度概率的路径作为最终的识别结果。而在采用词图数据结构的搜索算法中，搜索的路径范围大大扩大了。我们仍然以图 2 为例，在结点 3，这个时候产生的词为“港湾”，它的前驱词可能为“一派”和“离开”。在结点 3 向后扩展之前，将首先执行剪枝操作。这个时候“一派”和“离开”中的某一个词很有可能就被剪掉，失去了继续向后扩展的机会。我们假设“一派”被剪掉，这样将导致“一派”所在的路径无法到达语音输入的结尾帧，从而不可能被识别器输出。而如果采用词图数据结构来保留候选，我们可以看到，“一派”和“离开”都有可能是“港湾”的前驱，这时的可能路径条数就由原来的一条变成了两条。词图中可能路径的变多，使得我们能够充分的使用 Trigram 来评测各种可能的候选路径。候选路径的多少从某种程度上反映了词图的稀

疏程度。词图过密，保留的候选必然很多，这样在词图重估的时候比较费时，但是它能够提高识别率。词图过疏，重估过程必然加快，但是它可能导致一些期望的候选没有在词图中出现。这个时候无论采用什么重估方法，都将无法获得期望的词串输出，此时出现的错误是无法恢复的。因此需要在识别率和计算复杂度之间作权衡考虑。

3. 词图重估算法

词图重估算法是词图搜索算法中比较重要的一个模块，其主要目的是对词图中保存的可能的候选路径进行分数重估，在重估的过程中使用 Trigram 语言模型或更复杂语言模型提供的信息来指导搜索。这个过程一般来说耗时比较少。

词图重估算法可以采用以下的伪码语言来描述：

1. 对于链表中的每个实例 inst 执行如下操作
   {
   步骤 1：根据结点 inst->nodeidx 发出去的弧来扩展 inst，同时把 inst 的 token src，dwHistoryId，nodeidx，arcidx 传递给新产生的实例。
   - 在实例链表中寻找历史词为 dwHistoryId，弧索引为 arcidx 的实例 inst1
   - 如果 inst1 为空，则创建实例 inst1，计算 Trigram 语言模型概率，并且赋给 inst1->nodelm，同时把它加到实例链表的末尾*
   - 更新 src 的似然度：src->like += inst1->nodelm + wordgraph.arc[arcidx].likescore
   - 如果 src->like > inst1->state->like，则把 token src 传递给 inst1 的状态 0**
   - 如果 inst1->like 低于 wordgraph.node[inst1->nodeidx]的剪枝阈值，则把它从实例链表中移走，防止后续的扩展
   - 否则如果*为假而**为真，则扩展 inst1

   步骤 2：利用路径记录目前正被处理的实例的信息和它的前趋词的记录
   }
2. 对路径数组进行回溯以获得最佳的中文词序列串。

请注意，这个时候路径数组里所保存的路径的似然得分是各个词的纯声学得分以及 Trigram 语言得分的总和，因此，即使是和一遍搜索里产生的路径完全一样（这里的一样指的是词序列一样，而且对应词的时间分点也一样），由于一遍搜索里使用的是 Bigram 分数，它们的似然得分也是不一样的。

4. 实验

实验采用的模型为 TriIF 模型，由 863 数据库中 70 个人的男声数据训练而得，测试集选取了 863 数据库中的 2 个人的男声数据，另外为了做对比，也选取了训练集中的 2 个人来做测试。特征参数为 16 维的 MFCC 和对数能量，以及它们的 ARCEP 参数，共 34 维。每个测试人的数据包括 521 个句子。

1)一遍搜索和词图搜索的比较

表.1: 一遍搜索和词图搜索的性能的比较

测试人 / 算法	M00	M08	M11	M16
一遍搜索	68.4%	80.2%	66.0%	50.0%
词图搜索	69.0%	80.7%	67.6%	52.6%

这里 M00 和 M08 来自于训练集的数据，M11 和 M16 来自于测试集的数据。

2)时间复杂度的比较

选取的测试数据为 M16 的 100 句话，从 m16c1041.mfc 一直到 m16c1140.mfc，采用的方法分别为：直接使用 Trigram 语言模型的一遍搜索（也称集成搜索），在利用 Bigram 的一遍搜索产生的词图基础上的词图搜索。

表 2: 时间复杂度的比较

评测量 / 算法	词图搜索	一遍搜索 I	一遍搜索 II
时间	41 分钟	62 分钟	108 分钟
识别率	44.9%	42.3%	43.8%

在使用 Trigram 语言模型的一遍搜索中，为了能够达到比较理想的识别率，剪枝阈值不能设置得太小，否则由于没有保留足够多的历史信息，使用 Trigram 并不一定能够提高识别率。表 2 中一遍搜索 I 和一遍搜索 II 的区别就在于一遍搜索 II 中的阈值设置得比较大，在搜索的过程中保留了比较多的候选（这样正确的候选就更有可能被保留下来），因此通过使用 Trigram 语言模型分数，正确的候选才更有可能被挑选出来，导致识别率有了提高，但是要想超过词图搜索的识别率，剪枝阈值应该设置得更宽，此时耗费的时间必然更多，这对于一个实时系统而言是无法容忍的，因此在一遍搜索中直接使用 Trigram 是不大可能的。

5. 结论

1. 在词图搜索中使用 Trigram 语言模型是可行的，而且耗费的时间很少。通过实验结果表 2 可以看出，在没有很好的剪枝策略的前提下，是没有办法在集成搜索中直接使用 Trigram 的，所以此时使用词图搜索算法就成为了必然。
2. 词图搜索算法使得复杂语言模型的利用成为可能（比如 Long-span 语言模型）。词图搜索算法相比集成搜索算法的优越之处不仅在于它能够很轻松的使用 Trigram 语言模型，更重要的是，我们可以在词图的

基础上尝试使用各种复杂的语言模型，比如 Long-range 的 Trigram 等。可以这么说，词图搜索算法能够作为我们测试语言模型性能的一种方法。

3． 词图搜索算法在测试集中的性能表现要优于训练集，这主要得益于它从相似候选中选取出正确候选的能力。从表 1 可以看出，在训练集（M00、M16）的测试中，M00 和 M08 的识别率没有明显的提高，而在测试集（M11、M16）中，M11 和 M16 的识别率有了一定程度的提高。这种现象出现的主要原因在于：对于训练集，由于声学模型就是用这些数据训练出来的，因此相互之间匹配得比较好，这样在识别的时候，它产生的候选质量比较高，更为重要的是，我们期望的候选（正确的声学候选）所在的路径的分数比较高，这样只通过使用 Bigram，这些正确的候选就能够被挑选出来，因此在训练集中的测试看不出明显的效果。Trigram 挑选候选的能力在测试集中体现得比较明显。这充分的说明词与词之间的搭配关系如果在识别的过程中得到正确的应用，是能够提高识别率的，而且不会带来明显的计算负担。

6. 参考文献

[1] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V. & Woodland, P., “The HTK Book”, Microsoft Corporation, Dec, 1995bbu

[2] Beulen, K., Ortmanns, S. & Elting, C., “Dynamic Programming Search Techniques for a Cross-Word Modeling in Speech Recognition”, IEEE Internal Conference on Acoustic, Speech and Signal Processing, vol.2, Phoenix, AZ, Mar. 1999

[3] Ortmanns, S., Eiden, A., Ney, H. & Coenen, N., “Look-ahead Techniques for Fast Beam Search”, Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol.3, Munich, Germany, Apr. 1997, pp.1783-1786