一种基于状态转移的N元语言模型快速查询方法

付晓寅　魏玮　徐波

中国科学院自动化所　数字内容技术与系统研究中心　北京　100190

E-mail：{xyfu, weiwei, xubo}@hitic.ia.ac.cn

摘　要：融合基于海量语料的语言模型是提高统计机器翻译系统性能的有效手段。随着模型规模的不断增加，如何实现语言模型的快速查询成为影响系统性能的关键。本文根据N元语言模型查询具有的上下文相关特性，提出一种基于状态转移的语言模型快速查询方法。该方法通过构造Trie树索引保存查询状态，有效避免了语言模型的重复查询。实验结果表明，该方法在严格控制语言模型规模的基础上，能够显著提高统计机器翻译系统的解码效率。

关键字：N元语言模型、状态转移、快速查询、数据结构

A Fast Query Method based on State Transition for N-Gram Language Model

Xiaoyin Fu, Wei Wei and Bo Xu

Digital Media Content Technology and System Research Center, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

E-mail：{xyfu, weiwei, xubo}@ hitic.ia.ac.cn

Abstract: *Integrating N-Gram language model (LM) on massive corpus is one of the effective methods to improve the performance of the statistical machine translation (SMT). As the size of language model growing, how to efficiently use the language model has became the key point to affect the system performance. This paper proposes a fast query method based on State Transition for N-Gram language model, according to the context features in model queries. The method stores the query states by constructing the trie index, which can and effectively avoid the repetition in model queries. The experiment shows that our method can strictly control the size of LM and improve the decoding speed on SMT significantly.*

Keywords: *N-Gram Language Model, State Transition, Fast Query, Data Structure;*

1 引言

N元语言模型是统计机器翻译系统的重要组成部分，对生成符合语法规范和习惯用法的译文起着关键作用。Google的研究结果显示，随着N元语言模型元数的增加和训练语料规模的扩大，系统的翻译性能会逐步提高[Brants, 2007]。由于在解码过程中，翻译一段译文往往需要对语言模型进行数百万次的查询[Paul, 2011]，因此随着语料规模的不断增加，语言模型的查询效率问题逐渐凸显并成为影响统计机器翻译系统性能的主要瓶颈。

近年来，不少学者结合N元语言模型的特点，通过在查询过程中使用优化的查询策略或引入缓存机制等方法，提高模型的查询效率。

[骆卫华, 2010]针对采用布隆过滤器[Talbot, 2007]过滤语言模型造成查询速度下降的问题，提出了一种基于预录信息（book-keeping）的优化加速方法。该方法通过预先记录查询

过程的启发式信息，在压缩模型规模的同时，明显提升了语言模型的查询速度。但是该方法建立在数据压缩过滤的基础上，是准确率和存储空间的折中，无法改变信息缺失的事实，不适用于对系统准确率要求较高的场合。

[Li, 2008]把语言模型的查询等效为左右状态（Equivalent Left/Right LM State）的查询。通过对左右查询状态的缓存，避免查询回退时出现重复查询的情况，从而提高查询速度。由于在查询前需要对状态递归检查，造成系统查询效率的降低。[Pauls, 2011]从数据压缩、查询缓存的角度，提出一种语言模型快速查询方法。该方法对语言模型的词和上下文（word-and-context encoding）进行编码，既可以实现[Li，2008]状态缓存的目的，又能避免对状态的检查。通过设计哈希（Hash）函数对语言模型进行索引，极大提高语言模型查询效率，但由于需要对哈希函数预留一定的空闲空间，造成系统空间的浪费。

本文根据 N 元语言模型查询上下文相关的特点，提出一种基于状态转移的语言模型快速查询方法。该方法采用 Trie 树作为语言模型的基本结构，通过对 Trie 树之间的节点进行索引，将模型的查询转化为在 Trie 树节点上的状态转移，实现查询状态在 Trie 树之间的快速跳转。这种基于状态转移的方法同样具有上文中提到的状态缓存的能力，能够极大提高语言模型的查询效率。与[Pauls, 2011]的方法相比，采用 Trie 树的结构组织语言模型能够有效避免采用哈希函数带来的存储空间浪费问题，具有更为广泛的适应能力。

2 N 元语言模型

2.1 基本概念

N元语言模型是一种词序列概率的表示形式，可以根据已有词语序列预测下一个词的出现概率。设w_1^n表示词序列$w_1 w_2 \ldots w_n$，w_i ($1 \leqslant i \leqslant n$)为词序列中的单词。语言模型用 LM 表示。该词序列的语言模型计算公式为

$$P(w_1^n) = \prod_{i=1}^{n} P(w_i \mid w_{i-N+1}^{i-1}) \tag{1}$$

如果查询的单词片段在语言模型中不存在，则需要通过查询回退系数得到语言模型的概率。对于一个查询序列$P(w_i|w_{i-N+1}^{i-1})$而言，如果词序列$w_{i-N+1}^{i} \notin LM$，则语言模型计算公式为

$$\begin{cases} P(w_i \mid w_{i-N+1}^{i-1}) = P(w_i \mid w_{i-N+2}^{i-1}) \times \lambda(w_{i-1} \mid w_{i-N+1}^{i-2}) \\ \lambda(w_{i-1} \mid w_{i-N+1}^{i-2}) = 1 \qquad if(w_{i-N+1}^{i-1} \notin LM) \end{cases} \tag{2}$$

其中，λ 表示语言模型的回退系数。

2.2 查询优化

根据公式（1），当待查询的 N 元条目在语言模型中不存在时，每个片段均需要回退重新查询；当前 N 元条目查询结束转入下一个查询时，同样需要重新查询，如图 1 左所示。从图中可知，这种方法通常需要经过多次查询才能得到最终的结果。在统计机器翻译系统中，面对上百万次的模型查询，频繁地进行概率的重新查询无疑会造成系统性能的下降。

通过 2.1 节分析，对于一个特定的词序列，语言模型的查询过程不是随机的查询，而是一个动态连续的查询。在 N 元语言模型的查询过程中，充分利用词序列的上下文信息能够有效避免回退过程的重复查询，如图 1 右所示。通过在查询过程中引入回溯节点的位置信息，查询路径能够直接在语言模型节点之间快速跳转，有效减少重复查询的情况，从而提高查询效率。根据 N 元语言模型查询的这一特点，我们提出一种基于状态转移的语言模型快速查询方法。该方法能够快速获得语言模型查询的上下文信息，实现查询状态在模型上的连续动态转移，从根本上提高语言模型的查询效率。

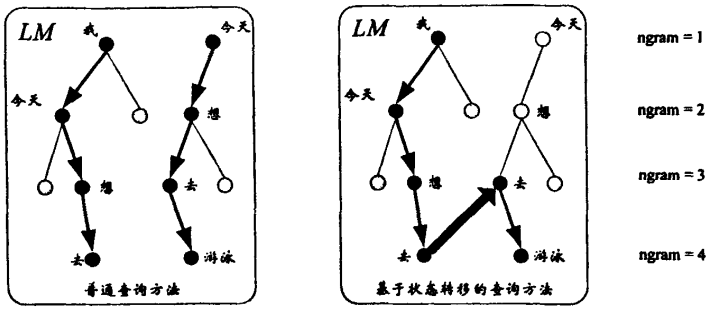

图 1. “我今天想去游泳”4 元语言模型查询方法对比

3 基于状态转移的 N 元语言模型

3.1 存储结构

目前，常用语言模型开源工具生成的语言模型结构通常为Trie树，如SRILM[1]以及IRSTLM[2]等。Trie树的每个节点表示语言模型的一个单词，从根节点出发的每条路径表示模型的一个n元条目。[Whittaker, 2001]将Trie树中相同深度的节点用数组表示，相邻深度的节点之间采用偏移量（offset）索引，由此可将每个节点的存储空间减少一半。[Hsu, 2008]为了提高回退（Back-off）系数的查询效率，将语言模型按照反向的方式存储。然而[Pauls, 2011]指出，采用正向方式存储语言模型比反向存储的查询效率更高。

基于状态转移的语言模型采用 Trie 树作为基本的存储结构，存储的方式按照词序列的实际顺序正向存储。Trie 树的每个节点除了保存与 SRILM 开源工具相同的单词词号，孩子节点偏移量，语言模型概率、回退概率信息外，还保存了回溯节点的位置（position）信息以及深度（depth）信息。如图 2 所示，图中的虚箭头表示回溯节点的指针。

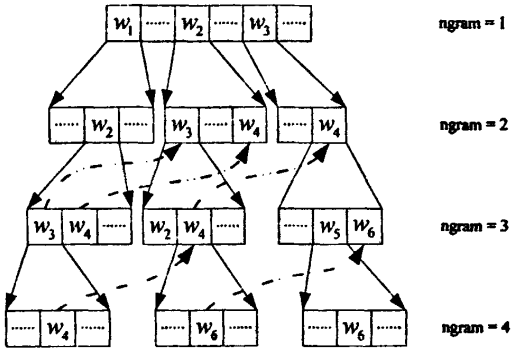

图 2. 基于状态转移的 4 元语言模型存储结构

由图 2 可知，基于状态转移的语言模型存储结构与普通 Trie 树的不同之处在于，该方法在 Trie 树节点保留了状态回溯时的跳转信息，即在 $w_1^n(n\geq 3)$ 的节点中同时存储 w_2^n 的位置信息。该结构可以实现语言模型查询状态在 Trie 树之间的快速跳转，具有与[Li，2008]相同的状态缓存功能，并且能够有效避免重复查询，提高模型的查询效率。

3.2 查询算法

对于特定的词序列 w_1^n，其语言模型的查询过程本质上是一个状态转移的过程。w_i ($1\leq i \leq n$) 表示一个输入变量，可以触发查询状态在Trie树上的一次转移。每个状态分别对应树中的一个节点。图 3 表示语言模型查询状态的转移结构。

[1] http://www.speech.sri.com/projects/srilm/
[2] http://hlt.fbk.eu/en/irstlm

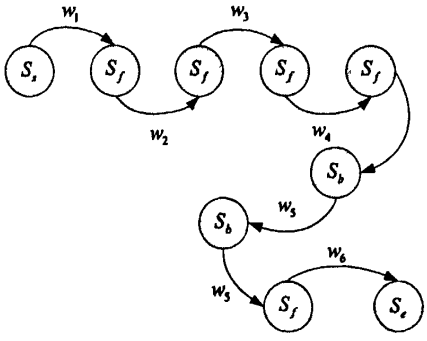

图 3. N-gram 语言模型查询状态转移

查询过程中共有 4 种状态，分别为起始状态S_s、前跳状态S_f、回跳状态S_b以及结束状态S_e。如图 3 所示。设经过序列$w_i^j(1 \leqslant i < n)$后，当前查询状态为S_i。当输入单词为w_{i+1}时，如果w_{i+1}在Trie树当前节点的子节点中存在，则前向跳转进入下一个状态S_{i+1}，否则进入回跳状态S_b。另外，当状态转移至Trie树的末端节点时，自发进入回跳状态S_b，然后根据输入单词继续进行状态的转移。直到输入最后一个单词w_n，状态转移至结束状态S_e，同时返回语言模型概率的查询结果。具体查询算法如下：

输入：待查询语言模型条目；输出：语言模型条目概率。

1. 初始化查询状态S_s
2. for i = 2 to N
3. 根据当前状态，计算输入第 i 个单词后查询进入的状态。当 i = N 时，转入步骤 6。
4. 如果$w_i^i \notin LM$，查询进入状态S_b，转入步骤 5。否则进入状态S_f，转入步骤 2。
5. 如果$w_2^i \notin LM$，继续进入状态S_b，执行步骤 5。直到状态转入状态S_f，转入步骤 2。
6. 查询状态进入结束状态S_e，返回语言模型查询概率。

4 实验结果与分析

4.1 实验设置

本文以[Chiang, 2007]的层次短语（Hierarchical Phrase Based, HPB）翻译系统作为基准系统，在口语和新闻两个领域进行中文—英文的翻译实验，并深入比较本文方法和传统方法在语言模型查询效率方面的性能。翻译实验所用的语料规模如表 1 所示。语言模型训练工具为 SRILM 开源工具。两个领域的测试集分别为 IWSLT-07 和 NIST 2006，分别包含 489 条和 1664 条测试语句。实验硬件环境为 CPU 2.00GHz (Inter® Xeon® 5130)，内存为 16G，操作系统为 Ubuntu 7.10。

表 1. 不同领域翻译模型和语言模型训练语料统计

	模型	句对数	中文词数	英文词数
口语领域	翻译模型[3]	382K	3.0M	3.1M
	语言模型[4]	1.3M	——	15.2M
新闻领域	翻译模型[5]	3.4M	64M	70M
	语言模型[6]	14.3M	——	377M

[3] BTEC (Basic Traveling Expression Corpus)和CJK (China-Japan-Korea corpus)双语语料。

[4] BTEC+JCK+CWMT2008 语料的英文部分。

[5] NIST08 提供的LDC语料。包括LDC2002E18, LDC2002T01, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004T07, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006T04, LDC2007T09。

[6] LDC2007T07 语料的英文部分。

4.2 实验结果

4.2.1 存储空间

在[Pauls，2011]采用的哈希方法中，为了保证模型的查询效率，需要增加一定数量的空闲节点，这就造成语言模型存储空间的浪费。根据Pauls的实验结果，即使经过压缩存储的语言模型大小仍然比普通语言模型增加 1 倍左右。通过 3.1 节的分析可知，采用状态转移方法的语言模型没有增加Trie树的节点数目，只是在树上的每个节点[7]增加了 8 个Byte的回溯信息，包括一个int类型的位置信息和一个int的深度信息，因此语言模型的大小正比于Trie树的节点数目。表 2 比较了原始语言模型（HPB）与基于状态转移的语言模型（HPB+ST）的存储空间。

表 2. 基于状态转移语言模型规模与原始规模的比较（单位：Mb）

	语言模型	HPB	HPB+ST	增加比例
口语领域	4-gram	65.7	105.0	59.8%
	5-gram	89.8	140.5	56.4%
新闻领域	4-gram	860.3	1424.4	65.6%
	5-gram	998.5	1549.7	55.2%

从表中可以看出，与 SRILM 工具训练得到的语言模型相比，新方法在两种领域的模型大小均增加了 60%左右，表明采用状态转移的查询方法能够有效控制语言模型的规模，避免了存储空间的过度膨胀。

4.2.2 查询效率

为了比较采用状态转移方法与传统方法的 N 元语言模型的查询效率，我们分别统计两个领域在不同元数语言模型情况下，翻译系统解码的查询时间。此外，为了进一步提高语言模型的查询效率，我们对每个测试语句根据翻译模型得到的目标语言单词对语言模型进行过滤，在过滤后的模型上统计模型的查询时间。实验结果如表 3 所示。其中，FLT 表示语言模型过滤方法，*为模型过滤时间。

表 3. 基于状态转移语言模型查询时间比较（单位：秒）

语言模型	查询方法	口语领域	新闻领域
4-gram	HPB	163	15433
	HPB+ST	**76**	**6886**
	HPB+ST+FLT	49 (23*)	4607 (2031*)
5-gram	HPB	261	25172
	HPB+ST	**94**	**8590**
	HPB+ST+FLT	61 (35*)	5828 (2290*)

从表 3 可以看出，采用状态转移的语言模型查询方法（HPB+ST）在 4 元语言模型上两个领域的查询速度分别提高了 53.4%和 55.4%，5 元语言模型上分别提高了 64.0%和 65.9%。上述实验结果表明，该方法可以显著提高系统的查询速度，对高元语言模型性能的提高更为明显。此外，采用语言模型过滤的方法（HPB+ST+FLT）能够进一步提高模型的查询效率。根据表 3 的实验结果，结合模型过滤和状态转移的方法，可将系统查询效率在 4 元语言模型上分别提高 69.9%和 70.1%，在 5 元模型上提高 76.6%和 76.8%。针对模型过滤需要一定时间的问题可以通过并行算法解决，从而有效减少模型过滤的时间开销，避免由于模型过滤造成系统查询效率的降低。

为了进一步分析采用状态转移方法对系统查询性能的影响，我们对两个领域语言模型发生回退查询的概率进行统计。统计结果如表 4 所示。

[7] 仅在二元及以上Trie树节点添加回溯信息

表 4. 语言模型回退查询发生概率比较

语言模型	口语领域	新闻领域
4-gram	60.5%	60.3%
5-gram	68.0%	67.5%

从表 4 可以看出，语言模型的回退现象在整个查询过程中普遍存在。在 4 元和 5 元语言模型中，两个领域回退查询的发生概率基本相同，这一现象与前面提到的采用状态转移方法提高的查询效率相一致。表明该方法能够有效避免语言模型回退时的重复查询，从而显著提高模型的查询效率。此外，注意到该方法在新闻领域查询效率的提高略高于口语领域，反映出基于状态转移的方法对翻译长句具有一定的适应能力，并通过模型片段间的快速跳转，从根本上提高语言模型查询效率。

5 结论

本文提出一种基于状态转移的 N 元语言模型快速查询方法。该方法从 N 元语言模型的查询特点出发，将语言模型的查询过程归结为状态转移的过程，从而保存了查询过程的历史信息，避免了语言模型频繁重查的情况。状态转移的查询方法在严格控制语言模型规模的基础上，显著提高了模型的查询效率。通过对语言模型进行过滤，能够进一步提高语言模型的查询效率。该方法不仅适用于统计机器翻译中语言模型的快速查询，同时也可用于语音识别、信息检索等广泛使用语言模型的自然语言处理领域。

在目前工作的基础上，我们将进一步研究面向海量语料语言模型的分布式存储和快速查询方法。

参考文献

骆卫华. 统计机器翻译中大规模数据处理若干问题的研究[博士学位论文]. 2010.

Brants, Thorsten, Ashok C.Popat, Peng Xu, Franz J.Och, Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Precessing and Computational Natural Language Learning*. Page 858-867.

Chiang, David. 2007. Hierarchical phrase-based translation. In *Computational Linguistics*. Page 201-208.

Hsu, Bo-June, James Glass. 2008. Iterative language model estimation: Efficient data structure and algorithms. In *Proceedings of Interspeech*.

Li, Zhifei, Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second ACL Workshop on Syntax and Structure in Statistical Translation*. Pages 10-18.

Pauls, Adam, Dan Klein. 2011. Faster and Smaller N-Gram Language Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*. Pages 258-267.

Shen, Libin, Jinxi Xu, Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*. Pages 577-585.

Talbot, David, Osborne Miles. 2007. Randomized Language Modeling for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics(ACL)*. Pages 512–519.

Talbot, David, Thorsten Brants. 2008. Randomized Language Models via Perfect Hash Functions. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics(ACL)*. Pages 505-513

Whittaker, E.W.D., B. Raj. 2001. Quantization based language model compression. In *Proceedings of Eurospeech*.