

基于立方剪枝的短语机器解码算法改进

朱海^{1,2}, 李淼¹, 乌达巴拉¹, 张健¹

(1 中国科学院 合肥智能机械研究所, 安徽 合肥 230031;

2 中国科学技术大学 信息科学技术学院, 安徽 合肥 230027)

摘要: 柱搜索算法是短语统计机器翻译广泛使用的解码算法。文中在立方剪枝算法的基础上提出了另一种对柱搜索算法的改进算法—基于立方剪枝的逆向递归算法。柱搜索算法对栈中所有的假进行扩展, 立方剪枝算法有选择地扩展栈中的前 k 个最好假设, 而基于立方剪枝的逆向递归算法只扩展栈中评分高的假设。实验结果表明获得相同的翻译质量, 立方剪枝算法比柱搜索算法快 10 倍, 立方剪枝的改进算法在大规模语料中比立方剪枝算法更节省时间。

关键词: 基于短语统计机器翻译系统; 柱搜索算法; 立方剪枝; 逆向递归

中图分类号: TP391.1

文献标识码: A

文章编号: 1000-7180(2011)02-0153-04

Improvement to the Decoding Algorithm of Phrase-Based Statistical Machine Translation Based on Cube Pruning

ZHU Hai^{1,2}, LI Miao¹, Wudabala¹, ZHANG Jian¹

(1 Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, China; 2 School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: Beam search is the popular decoding algorithm for phrase-based statistical machine translation systems. This paper describes another improved algorithm to beam search algorithm based on cube pruning algorithm—reverse recursive algorithm based on cube pruning. Beam search expands all the possible hypothesis on the stack. Cube pruning only expands the top k best hypothesis and reverse recursive algorithm based on cube pruning just expands the hypothesis scoring high. The results show that cube pruning algorithm use less time to achieve the same translation quality and reverse recursive algorithm based on cube pruning is quicker than cube pruning without affecting the translation quality much.

Key words: phrase-based statistical machine translation; beam search; cube pruning; reverse recursive

1 引言

解码问题一直是统计机器翻译的一大难题。即使对于简单的翻译模型, 解码问题依然是一个 NP 难题。目前, 基于短语的统计机器翻译^[1]系统主要采用柱搜索算法进行解码^[2]。这种算法在解码时枚举所有可能的翻译假设。虽然算法最终能够搜索到一个较好的翻译结果, 但花费时间长。

针对柱搜索算法时间性能不佳的问题, 一种高

效的改进方案是: 解码时先不考虑语言模型特征, 计算出前 k 个最佳翻译假设, 然后再对这 k 个假设进行语言模型重评分。该方案速度较快, 同时也产生了一定的解码错误。因为最佳翻译假设可能在这 k 个最佳假设之外。但通过调节 k 值可以降低解码错误。Chiang 在其层次短语机器翻译系统中详细描述了一种改进算法, 立方剪枝 (cube pruning) 算法^[3]。与柱搜索算法相比, 立方剪枝算法能在不降低翻译质量的条件下非常显著地提高翻译速度。

收稿日期: 2010-01-03; 修回日期: 2010-03-21

基金项目: 中国科学院知识创新工程重要方向项目 (KGCX2-SW-511)

立方剪枝算法每次都要计算固定的 k 个最佳假设,浪费了一些不必要的时间.基于立方剪枝的逆向递归算法,作为立方剪枝算法的改进,通过设置阈值,只扩展评分高的假设,不需要对每个节点都扩展固定的 k 个假设,因而可以比立方剪枝算法更节省时间.

文中详细介绍了立方剪枝算法和基于立方剪枝的逆向递归算法.通过构建基于短语的机器翻译系统,在大规模的 ACL-WMT08 法英语料上对三种算法进行了测试,并采用 BLEU 评分来评价翻译质量.

2 立方剪枝算法

一般柱搜索算法采用对数线性模型^[4]将语言模型、翻译模型等所有的特征函数融合,组成评分.一个翻译假设的总评分为:

$$\tilde{t} = \arg \max_i \{pr(t | s)\} = \arg \max_i \left\{ \sum_{m=1}^M \lambda_m h_m(t, s) \right\} \quad (1)$$

解码时扩展每一个假设都计算其所有特征的评分.但大部分的假设,扩展后都将因评分过低被剪枝掉.如果能不扩展评分过低的假设,则可以节省大量的时间而不会降低翻译质量.立方剪枝算法采用了这种策略,对每个栈有选择地扩展假设.

立方剪枝算法将柱搜索算法中的栈看作一个节点,将解码过程当作推导过程:

$$h_i^j + t_{j+1}^k \rightarrow h_i^k \quad (2)$$

式中, h_i^j 表示翻译了句子中位置 i 到 j 的翻译假设, t_{j+1}^k 表示在位置 $j+1$ 到 k 处的翻译选项, h_i^k 表示 h_i^j 用翻译选项 t_{j+1}^k 扩展后的翻译假设.则,评分存在如下关系:

$$\text{score}(h_i^k) = \text{score}(h_i^j) + \text{score}(t_{j+1}^k) + \text{scoreLM}(\text{merge}(h_i^j + t_{j+1}^k)) \quad (3)$$

其中, score 计算所有的评分, scoreLM 计算翻译假设用翻译选项扩展时的语言模型评分.

立方剪枝算法对解码过程中的每个节点,先计算出不考虑语言模型(scoreLM)的前 k 个评分最高的扩展假设,再对这 k 个假设进行语言模型重评分.以汉语到英语的翻译为例,设翻译的汉语句子为:

我₀真₁不₂知道_{3,4}

当前节点翻译了句子的 0 到 1 位置(用 h_0^1 表示),在位置 2 到 4 处有翻译选项(用 t_2^4 表示)可扩展当前节点,则:

$$h_0^1 + t_2^4 \rightarrow h_0^4$$

$$\text{score}(h_0^4) = \text{score}(h_0^1) + \text{score}(t_2^4) + \text{scoreLM}(\text{merge}(h_0^1 + t_2^4))$$

立方剪枝算法先计算不考虑 scoreLM 项的前 k 个评分最高的扩展节点,再对这 k 个节点进行语言模型重评分.实际上, $[0 \sim 1]$ 间的短语翻译有: I really 等共 20 个, $[2 \sim 4]$ 间的翻译选项有: don't know. 等共 4 个.将 $[0 \sim 1]$ 间的短语与 $[2 \sim 4]$ 间的翻译选项按其评高低排列,组成立方剪枝中的一个格,如图 1 所示.图中方框内的评分即为 scoreLM , 也就是组合损失(仅列出部分).组合损失有时为正,是因为翻译选项的评分存在误差.

	(don't know)-6.6	(didn't know)-7.1	(know)-7.2	(known)-12
(I really)-3.4	2.3	-1.0		
(I true)-5.1	4.2			
(I real)-6.8	-0.1			
(Really)-7.5				
...				

图 1 立方剪枝的格

在图 1 中,假设 $k=3$,用 $D(v)$ 表示节点 v (将栈看作一个节点)的前 k 个最佳翻译假设.分别计算出了 v 的两个前驱结点 u_1 和 u_2 的 $D(u_1)$ 和 $D(u_2)$, $D(u_1)$ 为翻译了 $[0 \sim 1]$ 的假设, $D(u_2)$ 为 $[2 \sim 4]$ 间的翻译选项.那么对后继结点 v ,只需要从 $(D_i(u_1), D_j(u_2)) (i \in [1, 20], j \in [1, 4])$ 的 80 个组合中找出评分最高的前 3 个假设.因为前驱节点中的元素已经排序,如果组合损失 scoreLM 可以忽略,那么后继节点的评分就只是前驱节点评分的加权和.

于是评分最高的假设 $D_1(v) = (D_1(u_1), D_1(u_2))$, 为格的上左角落,评分第二高的假设, $D_2(v) = \max((D_1(u_1), D_2(u_2)), (D_2(u_1), D_1(u_2)))$ 为上左角落的两个邻居.使用类似的方法,可以以评分的高低依次计算出假设.这种情形类似于 k -best 分析,所以可以采用 Hunag 的 k -best 改进算法^[5]以评分高低的优先次序从图 1 的格中计算出扩展假设.

实际上组合损失不可忽略.考虑到组合损失,上面使用的方法不再能保证以评分的高低计算出格中的假设.在图 1 中,依前面的方法,组合损失为 2.3 的假设(总评分为: $-3.4 + -6.6 + 2.3 = -7.7$)首先被枚举出来,而组合损失为 4.2 的后继结点(总评分为: -7.5)的总评分却比这一假设高.所以这种算法与将语言模型完全融合到解码中的柱搜索算法相比,将产生一定的搜索错误.但通过调节 k 值,这种

解码错误可以变得很小。

通常可以从不同位置的翻译短语用翻译选项扩展得到一个句子的翻译,如图2所示。若栈4的假设为最终翻译结果,则可以由栈0、1、2、3扩展而来。定义超边为连接当前翻译的短语和翻译选项,扩展得到新的节点。所有的超边称为一个超边集合。这些不同的超边组成了图1中格的第三维,形成了一个立方(cube)。

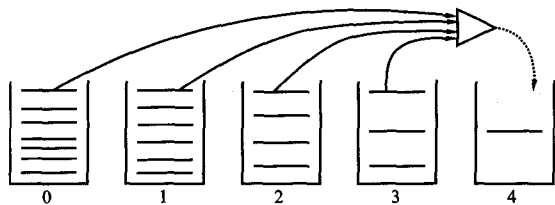


图2 立方剪枝算法栈中假设的扩展

图2中,栈4之前的栈中的假设扩展后通过超边束(实线箭头)存入一个优选队列(三角形表示),然后再放入栈4中。

考虑到源于 v 存在的多条超边,只需要所有超边中评分最高的假设。由Hunag的 k -best算法,用每条超边的最佳假设初始化一个优先队列,然后依一条超边的处理方式找到最佳假设。立方剪枝的具体算法可参见文献[3]。

3 基于立方剪枝的逆向递归算法

尽管立方剪枝算法比柱搜索算法速度快,却仍需计算每个节点固定数量(k 个)的假设。这些假设中的大部分对获取根节点(保存最终翻译结果的那个栈)的最佳假设没有用处。如果能按需求扩展节点,只扩展那些最有可能组成根节点最佳假设的节点将会更加地提高算法效率。

基于立方剪枝的逆向递归算法,以深度优先的次序递归地采用立方剪枝的方式扩展每个节点。对每个节点,通过设置阈值函数,该算法只扩展最有可能组成根节点最佳假设的节点。以一条翻译句子为例:假设翻译的句子长度为4,则生成5个栈。栈 i 中为翻译了句子中 i 个词的假设,栈4(根节点)中则为将句子全部翻译完了的假设。基于立方剪枝的逆向递归算法首先扩展根节点栈4,计算最佳翻译结果。开始栈4中没有假设,而栈4可以由栈0、1、2、3扩展而来,于是递归调用算法扩展栈0、1、2、3找到当前栈中最佳假设。类似地再对各个子节点,栈0、1、2、3进行递归扩展。基于立方剪枝的逆向递归算法递归扩展节点过程如图3所示。

扩展每个子节点时,不采用立方剪枝算法中对

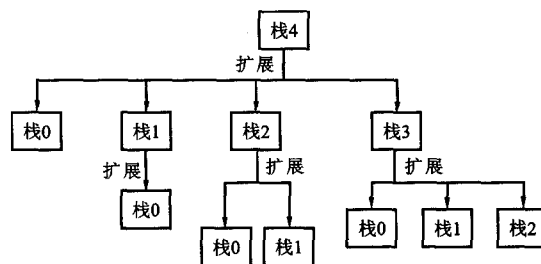


图3 基于立方剪枝的逆向递归算法节点扩展过程

每个节点固定地扩展 k 个假设的策略,而是设置一个阈值函数 $h(\alpha)$ 。当扩展假设的评分与待扩展假设中的最高评分的差小于阈值时,就以为已找到评分最高的假设并停止扩展,即

$$h(x) - h_{\max}(x') < h(\alpha) \quad (4)$$

其中, $h(x)$ 表示扩展假设评分, $h_{\max}(x')$ 表示待扩展假设中的最高评分, $h(\alpha)$ 表示阈值函数。阈值函数的设定将由实验来决定。

基于立方剪枝的逆向递归扩展算法可简述如下:

Step1:对长度为 n 句子生成编号为0到 n 的栈,编号 i 表示翻译完 i 个词的假设。

Step2:首先扩展栈 n ,若栈中没有假设转Step3,若存在转Step4。

Step3:栈 n 由前边编号为 $0 \cdots n-1$ 的栈扩展而来。对 $0 \cdots n-1$ 的每个栈:

① 设当前栈编号为 i ,可以由 $0 \cdots i-1$ 栈扩展而来。依次检查0到 $i-1$ 栈,若其中的 j 栈($0 \leq j \leq i-1$)没有假设则转①;否则转②。

② 对 j 栈,用立方剪枝算法计算出从 j 栈扩展到 i 栈的最佳扩展假设,当待扩展假设的评分与已扩展假设的评分差超过阈值时结束;若不存扩展假设则结束,否则将扩展假设存入 i 栈中。

Step4:翻译完句子,在编号为 n 栈中找到最佳翻译。

4 系统实验

为了对柱搜索算法、立方剪枝算法和基于立方剪枝的逆向递归算法在基于短语的统计机器翻译系统中的解码效率进行比较,设计了如下实验:使用大规模的ACL-WMT08法英语料,从翻译速度和翻译质量上比较了这三种算法。翻译质量的评价采用BLEU(4元)评分。计算机的CPU为英特尔酷睿四核Q6700(频率2.66GHz),内存4GB;操作系统为ubuntu8.10。

实验使用WMT08机器翻译专题讨论会上的

法语—英语语料. 训练集为 948 484 条法英句对, 语言模型的训练使用的是 1 412 546 句英语单语语料, 测试集的规模为 2 000 句. 使用 GIZA++^[6] 对训练语料进行词对齐训练, 采用三元的语言模型.

柱搜索算法的栈大小设置为 200, 柱剪枝阈值设置为 0.000 01. 三种算法在实验上的 BLEU-4 评分及消耗的时间如表 1 所示(表 1 中的基于立方剪枝的逆向递归算法使用了阈值函数).

表 1 解码效率表

语料	解码算法	BLEU	时间/s
法英	柱搜索算法	30.31	3021
	立方剪枝算法	30.31	294.6
	基于立方剪枝的逆向递归算法	30.31	217.11

由表 1 可以看出, 达到同样的翻译效果 (BLEU), 两种改进算法花费的时间仅为柱搜索算法的 1/10 左右. 基于立方剪枝的逆向递归算法在实验中的解码性能如图 4 所示. 在法英语料上, 算法稳定并达到与柱搜索算法相同的 BLEU (30.31) 时所花费的时间为 217.11s, 比立方剪枝算法 (294.6s) 要快(比较 BLEU 稳定后所花费的时间). 改进算法比立方剪枝算法更节省时间且算法稳定性更好, 不会像立方剪枝算法一样随着 k 值的不断变大而使花费的时间变长.

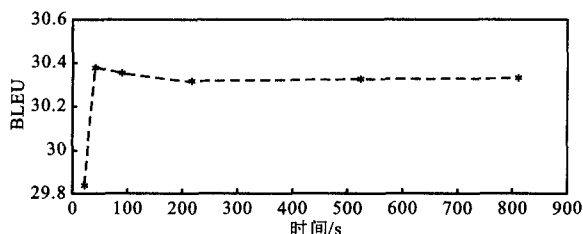


图 4 基于立方剪枝的逆向递归算法解码效率

实验的结果证明: 立方剪枝算法和基于立方剪枝的逆向递归算法能在不降低翻译质量的条件下比柱搜索算法快 4~10 倍; 基于立方剪枝的逆向递归算法在大规模语料中比立方剪枝算法更节省时间, 且算法稳定性好.

5 结束语

针对短语机器翻译系统中柱搜索算法速度较慢

的问题, 文中描述了两种改进算法: 立方剪枝算法和基于立方剪枝的逆向递归算法. 为了对这三种算法的解码效率进行了比较, 采用标准的训练集和测试集构建了基于短语的统计器翻译系统进行实验. 结果表明, 两种改进算法获得同样的翻译质量比柱解码算法快 10 倍; 基于立方剪枝的逆向递归算法在大规模语料中比立方剪枝算法解码速度更快, 且稳定性更好. 在对翻译速度有要求的系统中, 这两种改进算法有一定实用价值.

但是对基于立方剪枝的逆向递归算法, 如何设计阈值函数是一关键问题. 如能更好有效地估设计启发函数, 基于立方剪枝的逆向递归算法将会有更高的解码效率.

参考文献:

- [1] 袁里驰, 钟义信. 一种新颖的词性标注模型[J]. 微电子学与计算机, 2005, 22(9): 1-3.
- [2] 强静, 张建, 李森. 多民族语言农业处理平台的构建[J]. 微电子学与计算机, 2008, 25(8): 168-171.
- [3] Liang Huang, David Chiang. Forest rescoring: faster decoding with integrated language models[C]// Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07). Spain: Barcelona, 2007: 144-151.
- [4] Franz Josef Och. Statistical machine translation: from single-word models to alignment templates[D]. Germany, RWTH Aachen, 2002.
- [5] Liang Huang, David Chiang. Better k -best parsing [C]// Proceeding of the 9th International Workshop on Parsing Technologies. Vancouver, BC, 2005.
- [6] Franz Joseph Och, Hermann Ney. The alignment template approach to statistical machine translation [J]. Computational Linguistics, 2004(30): 417-449.

作者简介:

朱海男, (1984-), 硕士研究生. 研究方向为统计自然语言处理与机器翻译.

李森女, (1955-), 研究员. 研究方向为人工智能与农业知识工程, 博士生导师.

乌达巴拉女, (1982-), 实习研究员. 研究方向为统计自然语言处理与机器翻译.