基于反向转录语法的机器翻译混合解码策略*

张 浩，肖 桐，朱靖波

东北大学 自然语言处理实验室，辽宁 沈阳 110004

E-mail: zhanghao1216@gmail.com; {xiaotong, zhujingbo}@mail.neu.edu.cn

摘 要：Cocke-Younger-Kasami（CYK）解码算法和移进-归约解码算法是基于反向转录语法（ITGs）的统计机器翻译系统通常采用的两种解码算法。它们在翻译准确率和解码速度方面各有不同的优劣势。例如，基于 CYK 解码算法实现的解码器具有较高的翻译准确率，但解码速度较慢；而基于移进-归约解码算法实现的解码器解码速度很快，但翻译准确率较低。然而，实验显示后者相比于前者在处理短句子时具有相近的翻译准确率。根据这个发现，本文为基于 ITG 的短语机器翻译系统设计了一种混合解码策略。该策略结合了 CYK 解码算法和移进-归约解码算法各自的优势。实验结果表明，本文设计的混合解码策略有效地平衡了翻译准确率和解码速度这两方面因素的考虑。此外，对于需要高速解码速度的应用环境（个人机上超过 40 个句子每秒），该策略显示了较强的可用性。

关键词：反向转录语法；统计机器翻译；解码算法；混合解码策略

A Hybrid Decoding Strategy for ITG-based Machine Translation

Zhang Hao, Xiao Tong, Zhu Jingbo

NLP Laboratory at Northeastern University, Shenyang, 110004

E-mail: zhanghao1216@gmail.com; {xiaotong, zhujingbo}@mail.neu.edu.cn

Abstract: The CYK and the shift-reduce parsing algorithms are two popular decoding algorithms for ITG-based Machine Translation Systems. However, these two algorithms have different strengths and weaknesses in terms of translation accuracy and decoding speed. Specifically, a CYK-style decoder generally achieves high translation accuracy, but has a relatively low decoding speed. By contrast, a shift-reduce-style decoder runs much faster, but degrades in translation accuracy. However, from experiments we observe that the shift-reduce-style decoder achieves comparable translation accuracy with its CYK-style counterpart on short sentences. Motivated by this observation, we design a hybrid decoding strategy to combine the advantages of both algorithms for a phrase-based SMT system with ITG constraints. Experimental results show that our hybrid decoder finds a good trade-off between translation accuracy and decoding speed. Moreover, it shows a very promising applicability for high-speed decoding circumstances (e.g., over 40 sentences per second on PCs).

Keywords: ITGs; statistical machine translation; decoding algorithm; hybrid decoding strategy

1 引言

反向转录语法（Inversion Transduction Grammars）只允许两种调序方向，即正向调序和反向调序。这种调序约束不仅极大得减少了调序模型的复杂度，还保持了刻画真实语料中不同语言之间调序所需的灵活性[1]。反向转录语法的这些性质使得一些具有多项式时间复杂度的解码算法成为可能[2][3]。例如，CYK 解码算法和移进-归约解码算法已被应用于基于反向转录语法的短语统计机器翻译系统中[4-5]。

CYK 解码算法是一种动态规划算法，它对源语言句子的每一个跨度（span）搜索所有可能的解码路径以产生该跨度下的候选翻译选项（hypothesis）。所以，基于 CYK 解码算法实现的解码器往往可以达到较高的翻译准确率；但它的时间复杂度较高，是 $O(n^3)$。另一方面，移进-归约解码

* 基金项目：国家自然科学基金资助项目（60873091，61073140）；中央高校基本科研业务费专项资金、高等学校博士学科点专项科研基金资助（20100042110031）

算法在启发信息的指引下，对每一个跨度贪婪地选择少数系统认为较好的解码路径以产生该跨度下的候选翻译选项。理论上，这种确定性移进-归约算法的时间复杂度是线性的[6]。因此，基于移进-归约解码算法实现的解码器具有非常快的解码速度；但是，作为代价，这种搜索策略也导致更多搜索错误的引入，从而降低了翻译准确率。

实验显示，基于移进-归约解码算法实现的解码器同基于 CYK 解码算法实现的解码器在处理短句子时具有相近的翻译准确率。这个结果启发我们可以利用这两种解码算法在处理长短句子上的不同能力结合它们，以充分利用它们各自在翻译准确率和解码速度上的优势。本文正是根据这一发现为基于反向转录语法的短语统计机器翻译系统设计了一种简单有效的混合解码策略。该策略的基本思想是：先使用移进-归约解码算法解码用标点符号隔开的子句，再使用 CYK 解码算法组合子句的候选翻译选项以完成对整个句子的解码过程。本文将这一策略应用于一套基于反向转录语法的短语统计机器翻译系统中。实验结果表明，该策略有效地平衡了翻译准确率和解码速度这两方面的考虑。相比于 CYK 解码算法，该混合解码策略以较小 BLEU 值上的损失作为代价，使解码速度提升了 25 倍多。此外，对于需要高速解码速度的应用环境（例如，在个人计算机上解码速度超过 40 个句子每秒），该混合解码策略比 CYK 解码算法和移进-归约解码算法具有更高的翻译准确率，显示了较强的可用性。

2 基准解码算法

本节主要描述了基准解码算法，即应用于基于反向转录语法的短语统计机器翻译系统中的 Cocke-Younger-Kasami（CYK）解码算法和移进-归约（shift-reduce）解码算法。

假设源语言句子 $\boldsymbol{f}=f_1f_2...f_J$，其中 J 表示源语言句子的长度。那么一个跨度 $span=[m,n]$ 被定义为一个从第 m 个词到第 n 个词的连续词串，其中 m，n 满足关系 $1\leqslant m\leqslant n\leqslant J$。CYK 解码算法和移进-归约解码算法的输入都是一个源语言句子 $\boldsymbol{f}$，输出是 k 个翻译系统评分较高的该源语言句子的翻译结果（k-best）。两种算法都采用一张二维表 $\boldsymbol{C}$ 来存储源语言句子的每一个可能跨度下的候选翻译选项。二维表 $\boldsymbol{C}$ 根据短语表（phrase-table）能覆盖的源语言跨度进行初始化。

2.1 CYK 解码算法

CYK 解码算法可用如下伪代码表示。

CYK 解码算法

```
1: 初始化二维表 C
2: foreach l = 1 to J − 1
3:     foreach beg = 1 to J − l
4:         end = beg + l
5:         foreach mid = beg to end − 1
6:             COMPOSE(beg, mid, end)
```

图 1　CYK 解码算法伪代码

其中，**COMPOSE** 函数对跨度 $\boldsymbol{C}$ [*beg*，*mid*]和跨度 $\boldsymbol{C}$ [*mid*+1，*end*]下各自所有的候选翻译选项进行正向和反向相互组合，将组合生成的所有翻译片段作为跨度 $\boldsymbol{C}$ [*beg*，*end*]下的候选翻译选项。为了加速解码过程，**COMPOSE** 函数使用了 beam 搜索算法，采用矩方图剪枝策略（即对每一跨度只保留系统评分前 20 的候选翻译选项）。图 2 给出了 **COMPOSE** 函数的基本过程。由于本文采用了对数线性模型，在组合候选翻译选项时，除了语言模型这一特征需要重新进行计算外，其他特征均具有可加性，只需要线性相加。一旦源语言句子 $\boldsymbol{f}$ 下的所有跨度（n - $m\geqslant 1$）都已通过 **COMPOSE** 函数进行扩展，最后生成的 k-best 翻译结果可从跨度 $\boldsymbol{C}$ [1，*J*]中获得。

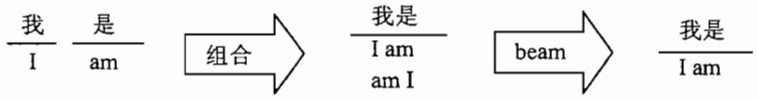

图2 **COMPOSE** 函数组合翻译候选示例

2.2 移进-归约解码算法

移进-归约解码算法需要两种数据结构以表示解码过程中的一个中间状态 $s=<\boldsymbol{S}, \boldsymbol{Q}>$，其中，

- $\boldsymbol{S}$ 是一个用来存储当前状态已经覆盖的源语言句子跨度的堆栈，
- $\boldsymbol{Q}$ 是一个用来存储当前状态尚未覆盖的源语言句子跨度的队列。

例如，算法的初始状态定义为 $s_I=<\emptyset, [1, 1][2, 2]...[J, J]>$，终止状态定义为 $s_T=<[1, J], \emptyset>$。在解码过程中，算法自左至右扫描源语言句子，根据当前状态不断地调用 **SHIFT** 或 **REDUCE** 函数以生成新的状态。**SHIFT** 函数从队列 $\boldsymbol{Q}$ 向堆栈 $\boldsymbol{S}$ 移进一个跨度以改变当前状态。**REDUCE** 函数从当前状态的堆栈 $\boldsymbol{S}$ 弹出两个跨度，根据这两个跨度调用 **COMPOSE** 函数以生成对应更大跨度下的候选翻译选项，再将新生成的更大跨度压入堆栈 $\boldsymbol{S}$。移进-归约解码算法的伪代码表示如下。

移进-归约解码算法

```
1: 用初始状态 s_I 初始化优先堆 H
2: while 从 H 中 pop 出一个状态 s_c
3:     if s_c 是终止状态 s_T then
4:         count = count + 1
5:         if count > d then return
6:     else
7:         if s_c 的队列 Q 不为空
8:             s_c' = SHIFT(s_c)
9:             push HEUR(s_c')到 H
10:        if s_c 的堆栈至少有两个元素
11:            s_c' = REDUCE(s_c)
12:            push HEUR(s_c')到 H
```

图3 移进-归约解码算法伪代码

其中，所有状态通过一个全局优先堆 $\boldsymbol{H}$ 进行管理，而一个状态的好坏通过启发函数 **HEUR** 进行评价。该启发函数类似开源平台摩西（Moses）所使用的启发函数[1]。本文使用启发函数，通过评价一个跨度下最好的候选翻译选项来评价一个跨度的好坏，再将堆栈 $\boldsymbol{S}$ 和队列 $\boldsymbol{Q}$ 下所有跨度评分的乘积作为对当前状态的评价。另外，参数 d 表示到达终止状态的解码路径数。理论上，对于确定性移进-归约算法，其时间复杂度是线性的，这时具有最高的解码速度（$d=1$）。但实际中，因为受限于启发函数的设置，算法需要搜索更多的解码路径以缓解翻译准确率的下降。另外，单条解码路径产生的翻译结果容易导致采用最小错误率方法训练的翻译模型参数不稳定[7]。通常可以通过适当增加参数 d 来平衡这几方面的考虑。在本文中，参数 d 设置为 30。

3 混合解码策略

3.1 初步实验结果及分析

本文实现并比较了上述两种基准解码算法。实验所使用的短语表是从包含 2M 句对的语料库

[1] http://www.statmt.org/moses/摩西的启发函数使用模型评分（model score）和未来代价（future cost）来评价一个候选翻译选项。

中抽取出来，该语料库选自 NIST 2008 汉英机器翻译任务受限领域数据集[1]。对于调序模型，本文同时使用了由熊德意等人提出的基于最大熵的词汇化调序模型[4]和由 Galley 和 Manning 提出的层次化调序模型[8]。表 1 显示了两种算法在 NIST MT 2006 测试数据集（1664 个句子）上的翻译性能和解码速度。可以看到，在解码速度上，移进-归约解码算法要比 CYK 解码算法快 86 倍多，但在翻译性能上则下降了 6 点多 BLEU 值。

表 1　CYK 解码算法和移进-归约解码算法的比较

解码算法	IBM BLEU-4（%）	速度（句子数每秒）
CYK	32.90	0.50
移进-归约	26.71	43.19

很明显，翻译准确率上的大幅下滑使得移进-归约解码算法对大多数需要高翻译质量的应用系统是不适用的。为了深入研究这个问题，本文比较了移进-归约解码算法和 CYK 解码算法在不同长度句子上的性能。从图 4 可以看到，两种算法在翻译短句子（≤10 个词）时具有很相近的 BLEU 值，而在翻译长句子时性能差异较大。这种现象是由它们所采取的不同搜索策略引起的。当翻译一个短句子时，其搜索空间[2]要远小于翻译一个长句子时的搜索空间。此时，虽然移进-归约解码算法采用的是贪心搜索策略，但引入的搜索错误不多。相反，在处理长句子时，巨大的搜索空间使得移进-归约解码算法更有可能引入搜索错误，最终导致翻译性能较大程度的下降。

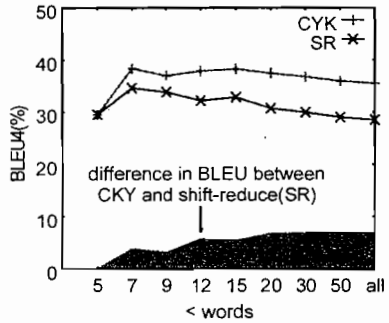

图 4　对不同的句子长度两种算法的性能差异

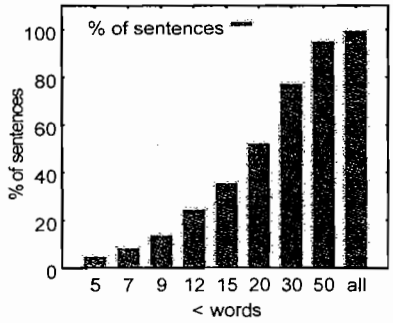

图 5　数据集 MT06 中不同长度句子比例

3.2　提出的解码策略

根据上述实验观察，本文进一步研究了结合 CYK 解码算法和移进-归约解码算法各自在处理长短句子时不同优势的混合解码策略。一种较直接的方式是用移进-归约解码算法解码短句子，并用 CYK 解码算法解码长句子。但是，如同图 5 所示，短句子在整个数据集上的比例非常小。例如，只有大约 25%的句子长度小于 12 个词。这表明，当综合考虑翻译准确率和解码速度时，该方法不能充分利用移进-归约解码算法处理短句子的能力，只有较少的性能提升空间。

本文设计了另一种混合解码策略以更好得利用这两种解码算法的性质。其基本思想非常简单，就是先使用移进-归约解码算法解码标点符号之间的子句，再用 CYK 解码算法组合子句的候选翻译选项以完成对整个句子的解码过程。其设计的动机在于大多数标点符号之间的子句都是短句（例如，在 MT06 测试数据集上，平均句长为 8.89 个词）。这种设计使得移进-归约解码算法在短句子上的处理能力得以更大程度的发挥。下面的伪代码总结了本文设计的混合解码策略。

---

[1] LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 和 LDC2006E92。

[2] 对于机器翻译系统，搜索空间的大小以句子长度指数级增长[9]。

```
混合解码策略
1: SP = 找到所有标点符号隔开的最大跨度
2: foreach span in SP
3:     call 移进-归约算法解码 span
4:     标记 span 及其所有子跨度为已扩展
5: foreach l = 1 to J − 1
6:     foreach beg = 1 to J − l
7:         end = beg + l
8:         if C[beg, end]还没有被标记为已扩展 then
9:         foreach mid = beg to end − 1
10:            COMPOSE(beg, mid, end)
```

图 6 混合解码策略伪代码

其中，第 2-4 行的代码调用移进-归约解码算法解码标点符号之间的子句，并将各个子句对应的跨度及其所有子跨度标示为已扩展，以避免 CYK 解码算法重复计算。第 5-10 行的代码调用 CYK 解码算法组合子句的候选翻译选项，以完成对整个句子的解码过程。

4 实验

4.1 实验设置及数据

实验所使用的翻译系统为东北大学自然语言处理实验室开发的基于反向转录语法的短语机器翻译系统[4][10]。系统实现采用对数线性模型，所使用的特征包括：两个短语翻译概率 $p(e|f)$和 $p(f|e)$，两个词汇化短语翻译概率 $lex(e|f)$和 $lex(f|e)$，一个语言模型概率，一个目标语词长惩罚因子（word penalty），一个目标语短语数目惩罚因子（phrase penalty），六个基于层次化调序模型的词汇化调序特征和一个基于最大熵调序模型的词汇化调序特征，一个记录空翻译次数的特征[1]和一个双语词典特征。其中，双语词典特征记录了源语言短语和目标语短语同时出现在双语词典中的次数，而双语词典是从汉英翻译词典（LDC2002L27）中抽取出来的。

实验所使用的训练数据如 3.1 小节所述，来源于 LDC，共包括大约 2M 句对。首先使用 GIZA++ 工具获得源语和目标语方向的词对齐，并使用 inter-sect-diag-grow 方法获得最终的词对齐结果。接着抽取短语翻译模型和调序模型。在抽取短语时，本文限制源语言短语最大词数为 3，目标语短语最大词数为 5。抽取出来的翻译模型和调序模型同 3.1 小节。另外，系统使用了一个 5 元语言模型，其语料来源于 Gigaword Xinhua 和 AFP 部分，再加上双语语料中的英语部分。系统将 NIST 2006 数据集作为开发集（tuning set），使用最小错误率方法[11]训练对数线性模型中所有特征的权重。系统使用 NIST 2003，NIST 2004，NIST 2005 和 NIST 2008 数据集作为测试集（test set）。在解码过程中，系统限制最大调序长度（distortion distance）为 10，使用额外的翻译模块对数字（number）、日期（date）、时间（time）和 byline 进行翻译，并整合到最终的翻译结果中。实验结果使用 IBM 版本的大小写无关的 BLEU-4（%）和解码速度（句子数每秒）进行评价。

4.2 实验结果

表 2 显示了本文提出的混合解码策略和基准解码算法比较的结果。可以看到，混合解码策略（第四行）在翻译准确率上比移进-归约算法（第三行）至少高出 3.8 点 BLEU 值；在解码速度上比 CYK 算法（第一行）快 25 倍多。所以，该混合解码策略有效地平衡了翻译准确率和解码速度这

[1] 这个特征在英文中被称为：evil feature。它在 NIST 评测数据集（特别是 MT03~05 数据集）上非常有效。

两方面的考虑。为了对比，解码算法 CYK_PU（第二行）是对原始 CYK 算法的改进。它也采取了对源语言句子按标点切分后进行解码的策略。不难看出，这种改进也加快了解码速度，但本文的混合解码策略在解码速度上依然是改进后的 CYK 算法的 3 倍多。

表 2　混合解码策略和基准解码算法的比较

解码算法		MT06 (开发集)		MT03		MT04		MT05		MT08		全部数据	
		BLEU	速度	BLEU	速度	BLEU	速度	BLEU	速度	BLEU	速度	BLEU	速度
1	CYK	32.90	0.50	37.06	0.50	36.17	0.39	35.17	0.38	26.17	0.46	33.77	0.44
2	CYK_PU[1]	32.79	3.45	36.92	2.91	36.12	3.03	35.19	2.90	26.25	4.02	33.74	3.29
3	移进-归约	26.71	43.19	29.88	41.84	28.99	38.74	29.07	40.40	21.69	42.08	27.39	41.17
4	混合策略	30.11	12.98	34.18	12.20	33.10	9.89	33.42	10.31	24.53	10.55	31.14	11.16
句子数目		1664		919		1788		1082		1357		6810	

本文进一步研究了该混合解码策略适合于什么样的应用环境。图 7 显示了这三种解码方法通过调整 beam 大小，在翻译准确率和解码速度上的对照。可以看到，在解码速度较低时（<20 个句子每秒），CYK 算法的翻译性能要好于混合解码策略。但是，在解码速度较高时（> 40 个句子每秒），本文的混合解码策略是一个更好的选择。例如，当解码速度需要达到 50 句每秒时，混合解码策略的 BLEU 值为 29.01%，这比 CYK 算法的翻译准确率要高得多。这说明，该混合解码策略更适合实际应用，例如实时机器翻译系统。

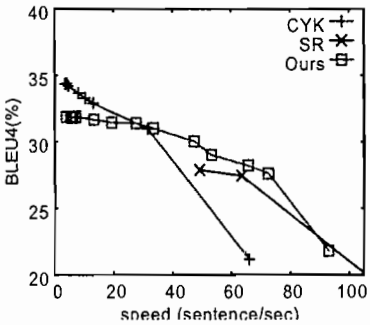

图 7　各解码算法在翻译性能和解码速度上的趋势

5　相关工作

许多研究已经将反向转录语法应用于基于短语的统计机器翻译系统之中，以提高调序和解码效率。例如，熊德意等人[4]提出了基于反向转录语法的最大熵调序模型，并采用 CYK 解码算法完成解码过程。冯洋[5]等人设计了一种倾向于归约的移进-归约解码算法，使得目标语言受到反向转录语法的约束。他们将它应用在摩西开源平台上。实验结果显示他们的方法相比于摩西略微提高了解码速度。本文分别采用 CYK 解码算法和移进-归约解码算法实现了基于反向转录语法的解码器。对于 CYK 解码算法，本文使用标点切分的技术改进了原 CYK 解码算法的解码速度。而对于移进-归约解码算法，本文将反向转录语法的约束应用在源语言端，采用启发函数决定执行移进或者归约动作的决策。

事实上，CYK 算法和移进-归约算法早已在相关领域被研究了很多年[6][12]。然而，据我们所知，

[1] 该 CYK 算法也使用了用标点切分源语言句子的技术。

还很少有相关工作，在基于短语的统计机器翻译背景下比较这两种算法。本文在此背景下，于大规模数据上系统地比较了这两种解码算法。并且本文提出了一种混合解码策略，有效地平衡了 CYK 解码算法和移进-归约解码算法各自的优势，在翻译准确率和解码速度上找到了一种折中。相比于改进后的 CYK 解码算法，该混合策略在解码速度上依然有较大幅度的提升。

6 结论

本文在满足反向转录语法的基于短语的统计机器翻译背景下，首先经验性地比较了 CYK 解码算法和移进-归约解码算法各自的优劣势，接着提出了一种结合两种算法优势的混合解码策略。实验结果显示，该混合解码策略很好得平衡了翻译准确率和解码速度这两方面的考虑。对于需要高速解码速度的应用环境（例如在个人计算机上超过 40 个句子每秒），该混合解码策略比 CYK 解码算法性能更好，具有更大的提升空间。

参 考 文 献

[1] Zens, Richard, Hermann Ney, Taro Watanable, and Eiichiro Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proc. of COLING*, pages 205-211.

[2] Wu Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL*, pages 152-158.

[3] Wu Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23: 377-403.

[4] Deyi Xiong, Qun Liu and Shouxun Lin. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proc. of ACL*, Sydney, pages 521-528.

[5] Feng Yang, Haitao Mi, Yang Liu and Qun Liu. 2010. An Efficient Shift-Reduce Decoding Algorithm for Phrased-Based Machine Translation. In *Proc. of COLING*, pages 285-293, Beijing, August 2010.

[6] Sagae Kenji and Alon Lavie. 2005. A Classifier-Based Parser with Linear Run-Time Complexity. In *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 125-132, Vancouver, Canada.

[7] Franz Josef Och, Daniel Gildea, and Sanjeev Khudanpur et al. 2004. Final report of johns hopkins 2003 summer workshop on syntax for statistical machine translation (revised version). Technical report, February 25.

[8] Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of EMNLP* 2008, pages 848-856.

[9] Kevin Knight. 1999. Decoding complexity in word replacement translation models. *Computational Linguistics*, 25(4): 607-615, MIT Press, USA.

[10] Tong Xiao, Rushan Chen, Tianning Li, Muhua Zhu, Jingbo Zhu, Huizhen Wang and Feiliang Ren. NEUTrans: a Phrase-Based SMT System for CWMT2009. In Proc. of 5th China workshop on Machine Translation (CWMT), Nanjing, China, 2009: 40-46.

[11] Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160-167, July 2003.

[12] Stuart M. Shieber. 1983. Sentence Disambiguation by a Shift-Reduce Parsing Technique. In *Proc. of ACL* 1983, pages 113-118, Stroudsburg, PA, USA.