粘贴模型在依存语法统计机器翻译中的应用*

张 育，李良友，贡正仙，周国栋
苏州大学 计算机科学与技术学院，苏州 215006
苏州大学 自然语言处理实验室，苏州 215006
E-mail: {20084227065099, 20104227013, zhxgong, gdzhou}@suda.edu.cn

摘 要：基于依存语法的统计机器翻译中，由于依存文法的扁平化，随着节点数目的增多，解码过程中很难匹配到完整的规则，会有大量未覆盖节点需要进行粘贴操作，传统的粘贴操作采用邻近保序策略，但会在一定程度上造成译文顺序的混乱。本文构建了一个基于最大熵的粘贴模型，利用丰富的上下文信息指导译文粘贴。在 NIST2005 汉英测试集上的实验结果表明，使用粘贴模型能够有效的减少粘贴操作带来的错误影响，相对于基准系统提高了 1.13 个 BLEU 值。

关键词：统计机器翻译；依存翻译模型；粘贴模型

Attachment Model for Dependency-based Statistical Machine Translation

Zhang Yu, Li Liangyou, Gong Zhengxian, Zhou Guodong
Department of Computer Science and Technology, Soochow University, Suzhou 215006
Natural Language Processing Lab, Soochow University Suzhou 215006
E-mail: {20084227065099, 20104227013, zhxgong, gdzhou}@suda.edu.cn

Abstract: In dependency-based statistical machine translation, with the source words increased, it is difficult to match complete templates for treelet nodes during decoding process. And lots of uncovered nodes in treelets need to be translated using a traditional attached operation, called as neighborhood preserving strategy. However, this method cannot order the target string sequence well and thus reduce the readability of output translation. In order to address this, the paper constructs a ME-based statistical attached model, which uses context information to guide decoder to keep reasonable order for the target side sequence. On NIST 2005 evaluation set, the baseline system applying attached model can get an improvement by 1.13 BLEU score.

Keywords: statistical machine translation; dependency-based model; attachment model

1 前言

近年来，基于句法的统计机器翻译受到了越来越多的关注，并在近几年的 NIST 评测中取得了不错的成绩。根据 Chiang（Chiang，2005）的分类方法，基于句法的翻译模型可以分成形式化基于句法的模型和语言学基于句法的模型两类。其中形式化基于句法的翻译模型（Wu，1997；Xiong et al.，2006；Chiang，2005；Shen et al.，2008；Su et al.，2010）仅仅借助了形式化语法结构，没有包含任何语言学知识；语言学基于句法的翻译模型不仅采用了形式化的句法体系，本身也包含丰富的语言学知识，该类模型通常需要对源语言端或目标语言端进行句法分析，根据使用的结构树不同，该类模型可以进一步分为基于短语结构树的模型（Yamada et al.，2001，2002；Galley et al.，2006；Marcu et al.，2006；Liu et al.，2006；Huang，2006；Mi et al.，2008；Zhang et al.，2009；Liu et al.，2009）和基于依存树的模型（Lin,2004；Quirk et al.，2005；Xiong et al.，2007）。

依存文法相对于短语结构树而言，一定程度上能够体现出更深层次的语义信息，具有天然词汇化特征，以中心词驱动（张育等, 2010），对于待翻译的节点如果能够搜索到可用的规则，在指导重排序方面会有很大的优势。但是依存树结构趋向于“扁平化”，随着节点数目的增多，很难匹

* 本文承国家自然科学基金重大研究计划培育项目（批准号：90920004）的资助。

配到完整的规则，加上句法分析的错误影响，导致更多的依存结构需要划分成细小的结构来搜索可用的规则，因此对于译文粘贴组合顺序的控制就显得极为重要。

本文提出了一个基于最大熵的粘贴模型用来预测译文之间的组合次序，首先从真实语料中抽取粘贴实例，然后抽取不同的特征用于最大熵分类器建模。在NIST2005汉英测试集上的实验表明，使用粘贴模型提高了1.13个BLEU值。

本文将在第二节介绍依存结构的翻译模型，第三节介绍粘贴模型的构建，第四节给出所有的实验结果，最后第五节对本文进行总结，并展望下一步研究工作。

2 依存结构翻译模型

本文介绍的依存结构翻译模型，同Xiong（Xiong et al.，2007）的类似，其规则可以视为一个三元组$\langle \tilde{D}, \tilde{S}, \tilde{A} \rangle$，其中$\tilde{D}$为源语言端的依存treelet，$\tilde{S}$为目标语言段相对应的翻译串，$\langle \tilde{D}, \tilde{S} \rangle$为翻译对，$\tilde{A}$为翻译对之间的对齐关系。依存treelet定义为依存树中任意连通的子图（Quirk, 2005）。规则两端允许变量，但不允许间隔。例如：

<（公布|0（中国|-2）（今天|-1）），today China announced，1:3 2:2 3:1>

<（成|0（兑换|-1（#_#|-1））（#_#|1）），exchange #_#1 into #_#2，1:3 2:1 1:2 4:4>

图1给出了上述两个规则的图形化表示：

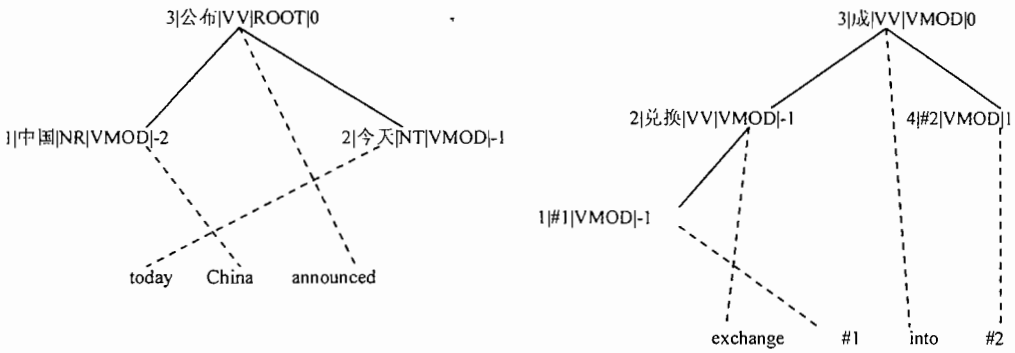

图1 对齐模板的图形化表示

采用规则抽取算法从训练集上抽取所有的规则形成规则库，计算相关的概率，解码时，自底向上搜索匹配所有可用的规则，通过规则的合并，导出最后译文（Xiong et al., 2007）。

这里的翻译概率表示为：

$$\tilde{e}_1^J = \arg\max_{e_1^J, z_1^K} \left\{ \lambda_{lm} h_{lm}(e_1^J) + \lambda_d h_d(z_1^K) + \sum_{n=1}^{N} \lambda_n h_n(e_1^J, f_1^I, z_1^K) \right\} \quad (1)$$

其中包括语言模型$h_{lm}(e_1^J)$，规则总数$h_d(z_1^K)$，以及每个规则所使用的特征函数$h_n(e_1^J, f_1^I, z_1^K)$。

3 粘贴模型构建

在基准系统解码过程中，对于规则的非终结符即变量需要进行替换操作，对于规则的未覆盖部分，进行译文的粘贴。本节先介绍粘贴操作，然后分成粘贴实例的抽取，粘贴特征使用以及模型构建三个部分介绍粘贴模型的构建。

3.1 粘贴操作

粘贴操作是指将规则中未覆盖的节点产生的译文片段按照某种策略粘贴到现有的译文片段中。基准系统采取了邻近粘贴策略，首先查找未覆盖节点 n 的父亲节点和兄弟节点中距离节点 n

最近的节点 m，然后按照源语言中节点 n 和 m 出现的先后次序进行粘贴，即如果源语言端节点 n 出现在 m 之前，那么在目标语言端节点 n 对应的译文也插入到节点 m 对应的译文之前。例如翻译如图 2 所示的一棵源语言依存树。

源语言句子：旅游 的 目的 是 什么 ？

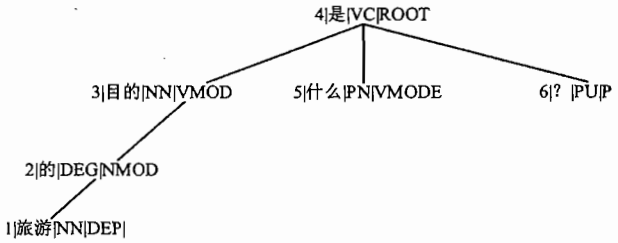

图 2　解码器输入的一棵源语言依存树

翻译时可用的规则为：

（是|0（#_#|-1）（什么|1）（？|2））||| what is #_#？

（目的|-1）||| the purpose

（的|-1（旅游|-1）||| of tour

因为在源语言端“旅游 的”出现在“目的”之前，所以“旅游 的”对应的译文“of tour”也被粘贴在“目的”对应的译文“purpose”前面。输出的译文为“what is of tour the purpose？”，造成译文顺序不正确。

本文针对这种粘贴操作问题提出了一个基于最大熵的粘贴模型。由于我们抽取的对齐模板时要求目标语言端连续（变量也当作一种连续），那么粘贴操作实际上是一个重排序的过程，粘贴模型的核心问题可以看作如何预测两个相邻的块 b1 和 b2 顺序的问题。最简单的方法是直接从训练语料中统计相关信息，但是随着合并和粘贴操作的使用，会造成一定的数据稀疏。因此我们利用最大熵模型来捕捉粘贴信息。

3.2　粘贴实例的获取

在实例抽取之前，首先说明块（block）的定义。由于训练和解码都是以源语言端依存树为基础，这里的block是由源语言端treelet和目标语言端连续的单词串组成的二元组：$block=\{D_{ni}^{nj},S_{mi}^{mj}\}$，其中，$D_{ni}^{nj}$ 是源语言端 treelet 中包含的 $n_i,...,n_j$ 的节点，S_{mi}^{mj} 是目标语言端从 $m_i,...,m_j$ 连续的单词串，由于源语言端是treelet，因此节点并不要求连续，只要求可以构成treelet即可。块block需要满足如下约束：

$$\forall(n,m)\in M,n\in\{n_i,...,n_j\}\leftrightarrow m_i\leqslant m\leqslant m_j \tag{2}$$

一个粘贴实例 I 是一个三元组<$b1$，$b2$，O>，其中块 $b1$ 和块 $b2$ 的目标语言端串是相邻的，$b1$ 的treelet的根节点和 $b2$ 的treelet的根节点是父子关系或者兄弟关系，O 是 $b1$ 和 $b2$ 的顺序。如果 $b1$ 和 $b2$ 的源语言和目标语言顺序保持一致，则 O 为 1，表示保序，否则 O 为 0，表示逆序。

我们在抽取对齐模板的基础上进行粘贴实例的抽取，首先定义了与算法相关的变量和函数。

1）treespan：记录 treelet 的节点集合和对应的目标语言序列，treespan.tb 表示目标语言串开始位置，treespan.te 表示目标语言串结束位置，treespan.idset 表示源语言节点集合；

2）treespanVector：存放所有的 treespan；

3）span：记录连续的目标语言序列和对应的源语言端 id 集合，span.tb 表示目标语言串开始位置，span.te 表示目标语言串结束位置，span.idset 表示源语言节点集合；

4）spanVector：存放所有的 span；

5）straightVector：存放目标语言端保序粘贴实例；

6）invertedVector：存放目标语言端逆序粘贴实例；

7）targetspan[i,j]：记录目标语言从 i 到 j 的连续单词序列；

8）GetSourceID(treelet)：获取 treelet 的所有节点 id；

9）GetTargetSpan(idset)：获取 idset 对应的目标语言序列；

10）GetSourceSpan(targetspan[i,j])：获取目标语言 i 到 j 对应的源语言端 id 集合；

11）GetTreeletfromVector(targetspan[i,j])：获取目标语言 i 到 j 对应的 treelet；

12）GetBlock(treelet)：获取源语言端为 treelet 的 block；

13）CheckConsistent(span)：判断 span 是否满足符合对齐一致性即满足在 span.tb 至 span.te 之间的目标语言单词对齐都在 span.idset 中，反之亦然；

14）CheckOrder(block1,block2)：判断 block1 和 block2 是否满足粘贴实例，返回 1 表示保序，返回 0 表示逆序；

图 3 给出了粘贴实例抽取的具体算法。

```
[1] Input：DepTree tree, String target, Alignment M
[2] init treespanVector, spanVector, straightVector, invertedVector;
[3] foreach (template ∈ Ω ) do
[4]   treespan.idset=GetSourceId(template.D);
[5]   targetspan[tb,te]=GetTargetSpan(treespan.idset);
[6]   treespan.tb=tb; treespan.te=te;
[7]   treespanVector.add(treespan);
[8] end for
[9] foreach(targetspan[i,j] ∈ target ) do
[10]  span.idset = GetSourcSpan(targetspan[i,j]);
[11]  span.tb=i;span.te=j;
[13]  spanVector.add(span);
[14]end for
[15]foreach(span in spanVector) do
[16]  CheckConsistent(span);
[17]end for
[18]foreach(span in spanVector)
[19]  foreach(span.tb<=mid<span.te)
[20]      treelet t1=GetTreeletfromVector(targetspan[span.tb.mid]);
[21]      treelet t2=GetTreeletfromVector(targetspan[mid+1,span.te]);
[22]      block b1=GetBlock(t1);
[23]      block b2=GetBlock(t2);
[24]    if(CheckOrder(b1,b2)==1)
[25]        straightVector.add((b1,b2));
[26]    if(CheckOrder(b1,b2)==0)
[27]        invertedVector.add((b1,b2));
[28]  end for
[29]end for
```

图 3　粘贴实例抽取算法

算法的第 3 至 8 行首先要获取所有完全词汇化的规则即该规则只含有终结符，获取该类规则的目的在于得到源语言端所有 treelet 对应的 targetspan[tb,te]以及 treespan，以便进行后续 treelet 的查找。第 9 行至第 13 行用来获得任意连续目标语言 targetspan[i,j]对应的 span。14 至 16 行过滤不合法的 span。17 至 28 行即根据 span 的范围获取所有可能的块，并判断粘贴实例所属类别。

3.3　特征获取

从粘贴实例中抽取特征，以进行最大熵训练。最大熵模型的关键在于如何选取合适而有效的

特征集合，我们在选取特征时考虑了两方面因素：第一，怎样利用较少的特征来表达更加充分的上下文信息；第二，特征在测试集中也能够使用。在粘贴时，处在短语边界的相关信息能够为粘贴位置的预测提供强有力的支持，因此本文定义了如表 1 所示的特征。

对于一个块 $b=<D,S>$，用 *D.r* 表示源语言端 treelet 的根节点中心词，*D.rp* 表示源语言端 treelet 根节点中心词词性，*S.h* 表示目标语言短语首词，*S.t* 表示目标语言短语尾词。

表 1　特征定义

特征类型	特征	特征说明
首词特征	*f1：b1.D.r*	块 1 源语言端根节点首词
	f2：b2.D.r	块 2 源语言端根节点首词
	f3：b1.S.h	块 1 目标语言端首词
	f4：b2.S.h	块 2 目标语言端首词
词性特征	*f5：b1.D.rp*	块 1 源语言端根节点首词词性
	f6：b2.D.rp	块 2 源语言端根节点首词词性
尾词特征	*f7：b1.S.t*	块 1 目标语言端尾词
	f8：b2.S.t	块 2 目标语言端尾词
组合特征	*f9：b1.D.r&b2.D.r*	块 1 和块 2 的源语言端根节点首词组合
	f10：b1.S.h&b2.S.h	块 1 和块 2 目标语言端首词组合
	f11：b1.S.t&b2.S.t	块 1 和块 2 目标语言端尾词组合
	f12：b1.S.h&b2.S.t	块 1 的目标语言端首词和块 2 的目标语言端尾词组合
	f13：b1.S.t&b2.S.h	块 1 的目标语言端尾词和块 2 的目标语言端首词组合
	f14：b1.D.r&b1.S.h &b2.D.r&b2.S.h	块 1 的源语言端根节点和目标语言端首词与块 2 的源语言端根节点和目标语言端首词组合

3.4　粘贴模型构建

我们利用最大熵分类器构建粘贴模型。目前最大熵模型比较成熟。主要思想是，在已知部分知识的前提下，关于未知分布的最合理推断即为符合已知知识最不确定或者最随机的推断。本文所使用的最大熵工具包原型为 maxent-2.4.1，在训练过程中，迭代次数设为 100，高斯先验设为 1，其他为缺省值。

译文粘贴 $h_a(e_1^J)$ 的得分表示为：

$$H_a(b1.S,b2.S)=\frac{\exp(\sum_i \theta_i f_i(b1,b2,O))}{\sum_O \exp(\sum_i \theta_i f_i(b1,b2,O))} \tag{3}$$

4　实验和分析

4.1　特征选择实验

我们首先使用了小规模的数据集进行特征的选择实验。使用了哈尔滨工业大学提供的双语语料，其中训练语料共 20000 句对，语言模型英语单语语料为 20000 句，开发集 100 句，测试集 141 句。

从粘贴实例中抽取了 5 万条记录作为开放测试集，其他作为训练样本，将首词特征 f1-f4 作为基本特征，表 2 给出了实验结果，左边部分为特征单独加入的实验结果，右边部分为各特征依次累加使用的实验结果。

表 2　粘贴模型特征选择实验

各特征单独使用			各特征依次累加使用		
实验方案	特征类型	测试精度	实验方案	特征类型	测试精度
1:f1-f4	首词特征	0.9577	*1:f1-f4*	首词特征	0.9577
2:f5-f6	首词词性	0.9362	*2:f1-f6*	+首词词性	0.9613
3:f7-f8	尾词特征	0.9291	*3:f1-f8*	+尾词特征	0.9622
4:f9-f14	组合特征	0.9432	*4:f1-f14*	+组合特征	0.9660

从表 2 可以看出，首词特征的贡献是最大的，因为首词通常标志着 treelet 和短语块的开始，对顺序控制有指导作用。其次是组合特征，组合特征为首词或尾词组合，也表达了短语块的边界信息。表 3 给出使用首词特征和所有特征的粘贴模型对系统性能的影响。

表 3　粘贴模型对系统的性能的影响

	Baseline
不使用粘贴模型	0.3852
加入首词特征	0.3895
加入全部特征	0.3924

从表 3 可以看出，加入全部特征，Baseline 的 BLEU 值可以提高 0.72 个点。说明粘贴模型能够有效的指导译文的粘贴，减少其操作错误对系统结果的影响。

4.2　粘贴模型实验

使用 FBIS 作为训练集，NIST MT 2002 作为开发集，NIST MT 2005 作为测试集，语言模型训练语料使用 LDC 发布的 GigaWord 新华社部分加上 FBIS 英文单语语料，使用 SRILM 训练 3 元模型，大小为 868M，对比实验系统采用经典的基于短语的系统 Moses。实验结果如表 4 所示：

表 4　粘贴模型在 NIST2005 测试集上的结果

模型	规则	BLEU
Moses	BP	0.2326
Dep-Sys	Baseline	0.2017
	Baseline+phrase	0.2156
	Baseline+AM	0.2130
	Baseline+AM+phrase	0.2263

Baseline 为基准系统，BP 为短语规则，AM 为基于最大熵的粘贴模型。

表 4 可以看出，基准系统集成双语短语之后 BLEU 值有了明显的提高，说明双语短语在基于句法的翻译中的重要作用，将短语的局部重排和善于翻译习惯表达的优势同句法模型全局重排的优势结合起来也是符合人们的思维的，在本文的翻译模型中，融合短语并不需要改变翻译模型的训练过程，可以较容易的集成双语短语，提高翻译结果。

在 Baseline 的基础上加入粘贴模型，可以取得 1.13 个 BLEU 值的提高。相比特征选择实验，在较大规模的语料中能够抽取较多的粘贴实例，可以利用更多的统计信息，并且对于长句而言，会产生非常多的未覆盖节点，相应的粘贴次数也大大增加，因此加入粘贴模型来控制未覆盖节点译文粘贴顺序对系统性能会有很大的提高。

Baseline 加入粘贴模型并集成双语短语之后，BLEU 值可以达到 0.2263，但是相较于 Moses 的结果而言，低了约 0.8 个百分点，原因在于 Moses 是目前成熟的短语翻译系统，我们的系统开发时间较短，技术细节方面还不成熟。

5 结论与未来工作

在基于依存语法的统计机器翻译中，由于依存文法的扁平化，解码过程中会有大量节点需要进行粘贴操作，在一定程度上造成译文顺序的混乱。本文构建了一个基于最大熵的粘贴模型，利用丰富的上下文信息指导粘贴操作。在NIST2005汉英测试集上的结果表明，使用粘贴模型能够有效的减少粘贴操作带来的错误影响，BLEU值相对于基准系统提高了1.13个BLEU值。

在未来的工作中，我们将继续测试粘贴模型在依存语法翻译模型中的效果，相比于基于短语结构树的模型，基于依存树的翻译模型研究相对较少，仍然有很大的潜力可以挖掘，我们希望通过完善模型和解码技术来提高翻译的质量。

参 考 文 献

[1] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora[J]. Computational Linguistics. 1997: 377-403.

[2] David Chiang. A Hierarchical Phrase-Based Model for Statistical Machine Translation[C]. In Proceedings of the ACL. 2005: 263-270.

[3] Libin Shen, Jinxi Xu, Ralph Weischedel. A New String-to-Dependency Machine Transaltion Algorithm with a Target Dependency Language Model[C]. In Proceedings of the ACL. 2008: 577-585.

[4] Jinsong Su, Yang Liu, Haitao Mi, et al. Dependency-Based Bracketing Transduction Grammar for Statistical Machine Translation[C]. In Proceedings of COLING 2010: 1185-1193.

[5] Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-String Alignment Template for Statistical Machine Translation[C]. In proceedings of the ACL. 2006.

[6] Yang Liu, Yajuan Lv, Qun Liu. Improving Tree-to-Tree Translation with Packed Forests[C]. In Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP. 2009: 558-566.

[7] Haitao Mi, Liang Huang, Qun Liu. Forest-Based Translation[C]. In Proceedings of ACL. 2008: 192-199.

[8] Deyi Xiong, Qun Liu, and Shouxun Lin. A Dependency Treelet String Correspondence Model for Statistical Machine Translation[C]. In proceedings of the Second Workshop on Statistical Machine Translation. 2007: 40-47.

[9] Deyi Xiong, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation[C]. In Proceedings of COLING/ACL. 2006: 521-528.

[10] Michel Galley, Mark Hopkins, Kevin Knight, et al. What's in a translation rule?[C]. In Proceedings of HLT/NAACL. 2004: 273-280.

[11] Daniel Marcu, Wei Wang, Abdessamad Echihabi, et al. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases[C]. In Proceedings of EMNLP. 2006.

[12] Chris Quirk, Arul Menezes and Colin Cherry. Dependency Treelet Translation:Syntactically Informed Phrasal SMT[C]. In proceedings of the ACL. 2005.

[13] Liang Huang, Kevin Knight, Aravind Joshi. Statistical Syntax-Directed Translation with Extended Domain of Locality[C]. In Proceedings of AMTA. 2006: 66-73.

[14] Dekang Lin. A Path-based Transfer Model for Machine Translation[C]. In proceedings of COLING 2004.

[15] Min Zhang, Hongfei Jiang, Aiti Aw, et al. A Tree Sequence Alignment-based Tree-to-Tree Translation Model[C]. In Proceedings of the ACL. 2008: 559-567.

[16] 张育, 王红玲, 周国栋. 基于两种句法分析的语义角色标注比较研究[J]. 计算机应用与软件, 2010, 27(8): 13-16.