基于树到串对齐模板翻译模型的n-best解码算法

黄赟 吕雅娟 刘洋 刘群
中国科学院计算技术研究所智能信息处理重点实验室
E-mail: {huangyun,lvyajuan,yliu,liuqun}@ict.ac.cn

摘 要: 基于树到串对齐模板的统计机器翻译模型是一种新颖的翻译模型，本文提出了基于树到串对齐模板翻译模型的n-best解码算法。实验结果表明，本文提出的解码算法不但提高了开发集上的oracle BLEU值，而且使得最小错误率训练程序能够训练得到更好的模型参数，从而改进了整个翻译系统的翻译质量。

关键词: 机器翻译，树到串对齐模板，n-best算法

The N-Best Decoding Algorithm for TAT-based Translation Model

Yun Huang , Yajuan Lü , Yang Liu , Qun Liu
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS
E-mail: {huangyun,lvyajuan,yliu,liuqun}@ict.ac.cn

Abstract: In this paper, we introduce a search algorithm that provides a simple and efficient way to find n-best results of the decoder of the *tree-to-string alignment template* (TAT) translation model, which describes the alignment between a source parse tree and a target string. Our experiments show that the new decoding algorithm does not only improve the oracle BLEU but also allow minimum error rate training procedure to find better parameters which improve the translation quality.

Keywords: Machine Translation, Tree-to-string Alignment Template, n-best Algorithm

1 引言

自然语言处理中的很多任务要求系统输出n个最好结果（n-best）而不只是输出单个最好结果。Och提出了最小错误率训练方法用来训练模型参数[4]，取得了比最大似然等估计方法更好的结果，最小错误率训练过程需要翻译系统输出n-best翻译结果。重排序（reranking）技术被证明是提高机器翻译质量的一个有效方法[6]，同样要求机器翻译系统输出n-best结果。

在自然语言处理领域，已经有许许多多的n-best搜索算法。Huang和Chiang描述了一个句法分析问题中高效的k-best算法[1]。May和Knight引进了基于加权有穷树自动机的算法，可以输出无重复的n-best树[3]。在人工智能领域，已经有很多成熟的搜索算法，例如广度优先搜索、深度优先搜索、A^*搜索算法等等。大部分n-best算法是基于线性图的，能直接用到树结构的算法很少。本文主要考察基于短语结构树到串对齐模板的统计机器翻译模型[2]，提出了一个高效的n-best解码算法。

2 树到串翻译模型

基于树到串对齐模板的翻译模型（TAT-based translation model）[2]是一种新颖的基于句法的统计机器翻译模型。在这个模型中，训练过程是从训练集中自动抽取树到串对齐模板，并估计概率分布；解码过程分成以下几个步骤：首先对待翻译的源语言句子进行句法分析，然后自底向上搜索推导，最后输出源语言句法树根结点对应推导栈中的翻译。解码器命名为*Lynx*。图1给出了解码的部分过程，子图(a)-(d)四部分给出了解码1、2、4、8结点的细节。

先前版本的Lynx实现的算法不能输出n个最好结果。作为替代，我们输出根推导栈中的

推导，称为*伪n-best*输出。这样做的主要问题是导致最小错误率训练程序不能找到好的模型参数。

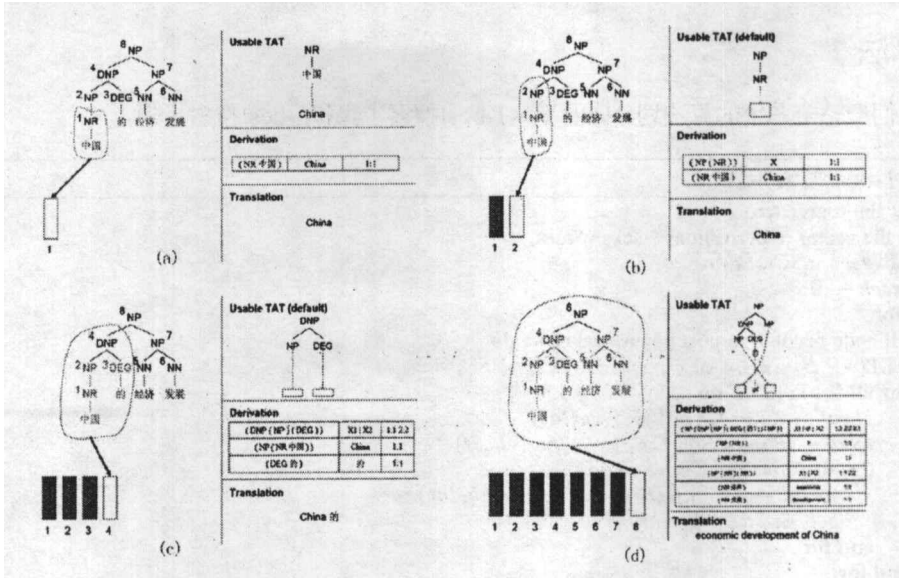

图 1: 1-best解码过程

3 术语定义

我们采用自底向上的推导栈作为解码器的数学抽象。

定义 1 *树到串对齐模板（tree-to-string alignment template，TAT）*： 树到串对齐模板是一个三元组$<\tilde{T},\tilde{S},\tilde{A}>$。这个三元组描述了源语言句法分析树$\tilde{T}=T(F_1^{J'})$和目标语言字符串$\tilde{S}=E_1^{I'}$之间的对齐关系$\tilde{A}$。图1中每个子图的右上角部分是一些TAT的示例。

定义 2 *推导（derivation）*： 推导定义为TAT的一个序列。此外，推导还包含累积分数向量、子推导、源语言词向量、目标语言词向量、源语言词和目标语言词的对齐关系等信息。图1中每个子图的右侧中部是一些推导的示例，图中只列出了推导的TAT序列。

定义 3 *重合并（recombination）*： 为了在不丢失信息的同时降低搜索空间复杂度，我们合并且只合并同一推导栈中具有相同译文语言模型分值的两个推导。假设n是语言模型元数[1]，有两个或多个推导的译文的$n-1$个词和末尾$n-1$个词完全相同，那么我们合并这几个推导，并称概率小的推导被概率大的推导合并掉。我们把重合并的推导实现成一个链接表。

定义 4 *推导栈（derivation stack）*： 推导栈定义成包含多个推导的向量。每个源语言树结点对应一个推导栈。在自底向上的1-best beam search过程中，我们只把那些在重合并过程中没有被合并掉的推导加入到推导栈中。换句话说，被合并掉的推导是不会出现在推导栈中的。图1中每个子图左下角的方框表示推导栈，其中存放的是该结点对应的所有推导。

定义 5 *n-best项（n-best item）*： n-best项定义为推导的序列。n-best项是n-best搜索步骤中所采用A^*算法的状态。我们很容易用一个简单的广度有限搜索由根推导栈中的推导和它的子推导信息构造n-best项。例如在图1中，根推导栈中最终的推导8[2]是由结点2中的推导2和结

[1]在我们的实验中，$n=3$。

[2]简化起见，本例中用推导所在的结点号作为推导的标号。

点7中的推导7构成，而推导2由推导1构成，另外假设推导7是由推导5和推导6构成，那么从根推导8构造的n-best项是序列：$(8,2,7,1,5,6)$。

4 算法

我们把整个系统的算法划分成两步：1-best搜索步骤和n-best搜索步骤。

算法 1 1-best搜索步骤

Require: the source tree T.
Ensure: the vector of derivations stacks *vStack*.
1: **for all** *stack* in *vStack* **do**
2: *stack* $\leftarrow \emptyset$
3: **end for**
4: **for all** node pn of T in post transversal order **do**
5: *vTAT* $\leftarrow SEARCHAVAILABLETAT(pn)$
6: **for all** tat in *vTAT* **do**
7: *vLeaf* $\leftarrow GETLEAFVECTOR(tat)$
8: *vSub* $\leftarrow ENUMSUBDERIVATION(vLeaf)$
9: **for all** derivation d in *vSub* **do**
10: $newd \leftarrow BUILDDERIVATION(vSub, tat)$
11: add $newd$ to *vStack*[pn]
12: **end for**
13: **end for**
14: recombine derivations in *vStack*[pn]
15: prune derivations in *vStack*[pn]
16: **end for**

4.1 1-best搜索步骤

我们算法的第一部分称为1-best搜索步骤，伪代码[3]见算法1，图1给出了解码示例。

4.2 n-best搜索过程

当我们完成1-best搜索步骤时，存在三种推导：被合并掉的推导，被剪枝掉的推导，既没有被合并也没有被剪枝掉的推导。留在推导栈中的那些没有被合并和剪枝掉的推导在1-best搜索过程中已经被计算过了，而那些被剪枝掉的推导显然已经无法再被找到，所以我们进行n-best搜索步骤的目的是找到那些被合并掉的推导。

这一步骤类似A^*搜索，其中n-best项是A^*搜索算法中的状态。由于n-best项记录了构造根推导过程中的所有使用到的推导，因此树结构转换成线性图结构，使得A^*算法能够适用。在搜索的每一次迭代中，算法展开当前n-best项对应的推导序列中每一个可能展开的地方，并向根回溯，沿途构建新推导，最终由新构建的根推导构造n-best项，加入open表。n-best搜索过程伪代码见算法2。

5 完备性证明

针对机器翻译解码问题，定义算法的完备性如下：给定正整数n，如果存在多于n个翻译结果而且算法可以输出n个分数最高的翻译结果，或者少于n个翻译结果而且算法可以输出所有翻译结果，那么我们称这个算法是完备的。我们的解码算法包含两个步骤，显然1-best搜索步骤不可避免地会剪枝掉好的候选推导，所以我们只证明基于A^*算法的n-best搜索步骤是完备的。

[3]在算法描述时，斜体$words$表示变量；斜体加黑体***words***表示向量；大写$WORDS$表示子函数。

算法 2 n-best搜索步骤

Require: the source tree T, the derivation stack vector ***vStack***.
Ensure: the vector ***vBest*** containing top n derivations.
1: initialize open list ***vOpen*** $\leftarrow \emptyset$, $found \leftarrow 0$
2: **for all** non-recombined derivation d in the root stack **do**
3: $tempitem \leftarrow DERIVATIONTOITEM(d)$
4: add $tempitem$ to ***vOpen***
5: **end for**
6: **loop**
7: **if** $found = n$ or ***vOpen*** is empty **then**
8: **break** // we found n results, or no more items
9: **end if**
10: $curritem \leftarrow$ the first item of ***vOpen***
11: remove the first item of ***vOpen***
12: **if** $curritem$'s same translation flag is false **then**
13: $found \leftarrow found + 1$
14: add $curritem$'s root derivation to ***vBest***
15: **end if**
16: ***currseq*** $\leftarrow curritem$'s derivation sequence
17: **for all** derivation d in ***currseq*** **do**
18: **if** d has no next recombined derivation **then**
19: **continue** // no next recombined derivation
20: **end if**
21: $nextd \leftarrow d$'s next recombined derivation
22: **while** $nextd$ is not in the root stack **do** // trace back to root
23: $pd \leftarrow nextd$'s parent derivation
24: ***vSub*** $\leftarrow pd$'s sub derivation vector
25: replace d with $nextd$ in ***vSub***
26: $tat \leftarrow$ the TAT which pd used
27: $nextd \leftarrow BUILDDERIVATION(vSub, tat)$
28: **end while**
29: $tempitem \leftarrow DERIVATIONTOITEM(nextd)$
30: $SETSAMEFLAG(tempitem, vOpen, vBest)$
31: add $tempitem$ to ***vOpen***
32: **end for**
33: sort ***vOpen*** by the derivation score (high to low)
34: **if** ***vOpen*** has more than n items **then**
35: $PRUNEDIFFTRANS(vOpen, n - found)$
36: **end if**
37: **end loop**

重合并技术有一个称作严格单调性的特性：假设推导a被推导b合并掉，那么任何由a和其他推导构造的新推导的分数，一定比那些由b和相同其他推导构造的新推导的分数要低。在算法n-best搜索步骤的每一次迭代中，我们从open表中删除$curritem$项，在构造新推导的时候利用被$curritem$推导序列中某一个推导所合并掉的推导，序列中的其他推导保持不变。根据重合并的绝对单调性质，当算法执行完向根推导栈的回溯过程时，我们得到根结点处的n-best项，它对应根推导栈中的推导的分数一定比原先$curritem$对应的分数要小。这意味着，在扩展$curritem$时，我们只会构造那些分数比$curritem$小的n-best项。换句话说，在算法执行的任何时候，当前扩展的$curritem$所始终是当前分数最高的n-best项。更重要的是，因为我们把所有新构造出的n-best项都加回到open表，所以我们不会漏掉任何可能潜在更好的n-best项。另外，当对open表进行剪枝的时候，我们只会裁剪掉那些分数比最好n个推导分数都要小的推导，所以我们不会裁剪掉前n个最好推导。总而言之，在算法n-best搜索步骤的每一阶段，open表中始终保存着最好的n个候选推导，这意味着如果推导d是搜索空间中最好的n个推导之一，那么算法一定会输出推导d。

System	n	oracle BLEU	dev BLEU	tst BLEU
pseudo	10	0.2551	0.2548 ± 0.0291	0.2391 ± 0.0248
true	1	0.2216	0.2378 ± 0.0155	0.2273 ± 0.0122
	10	0.2663	0.2479 ± 0.0452	0.2312 ± 0.0395
	100	0.3031	0.2623 ± 0.0067	0.2441 ± 0.0053
	200	0.3083	0.2604 ± 0.0091	0.2418 ± 0.0074
	500	0.3157	0.2647 ± 0.0027	0.2437 ± 0.0031

表 1: n-best算法对统计机器翻译的影响

6 时间复杂度分析

这一章我们将使用下面的符号分析算法的渐进时间复杂度：n表示输出n-best 结果的个数；t表示句法分析树结点数，我们认为其他独立于句法树的参数是在估计复杂度时是常数。

算法的1-best搜索步骤是一个beam search搜索算法。给定一个包含t个结点的源语言端句法分析树，构造t个推导栈的时间复杂度是$O(t)$。由于算法1中搜索匹配模板、枚举子推导栈、构造新推导、重合并、剪枝等过程都与句法树无关，所以时间复杂度是常数。整个1-best搜索步骤的时间复杂度是$O(t)$。

算法的n-best步骤是一个类似A^*的搜索过程，它的时间复杂度比1-best搜索步骤更难估计。我们假设算法2中的循环块（第6-37行）执行m次。在这个循环块中，推导队列的长度显然一定小于分析树的叶结点数t，因此循环复杂度不超过$O(t)$。向根推导栈的回溯过程时间复杂度是$O(t)$。函数$S_{ET}S_{AME}F_{LAG}$在最坏情况下时间复杂度是$O(m)$。排序整个open表时间复杂度是$O(m*log(m))$。循环块最后的剪枝过程时间复杂度是$O(m)$。从上面的分析可以看出，n-best搜索步骤的时间复杂度是$O(m*t^2))$。总体上来说，循环实际的次数m是很难精确估计的，因为它依赖于具体的推导。为了估计m的上限，我们在实验中记录实际执行的次数。我们发现平均来说m渐进增长比n^2慢，所以我们作最坏估计：整个循环块的时间复杂度是$O(n^2)$。从而整个n-best搜索步骤的时间复杂度是$O(n^2*t^2))$。

综上所述，算法的1-best搜索步骤的复杂度是$O(t)$，n-best搜索步骤的复杂度是$O(n^2*t^2))$，其中n是输出n-best结果的个数，t是句法分析树的结点数。

7 实验

我们在汉英机器翻译任务上测试n-best解码算法，训练集包含31,149个句对（包含843,256个汉语词和949,583个英语词）。在31,149英语句子的单语语料上用SRI的语言模型工具箱[4]训练了一个三元语言模型。从2002年NIST汉英机器翻译测试集中选取了571个较短的句子作为开发集，全部的2005年NIST汉英机器翻译测试集作为我们的测试集。用BLEU值[5]作为评价机器翻译质量的标准，在计算BLEU评价n元组时大小写敏感。

为了评估新算法的n-best输出，我们设计实验计算给定不同的n时，开发集上的oracle BLEU值。另外，我们设计实验评估n-best算法对树到串统计机器翻译的影响。为了验证我们在第6章估计的算法复杂度，我们记录并比较给定不同n时n-best搜索步骤所耗费的时间。因为最小错误率训练过程存在随机因素，所以每个实验都重复5次，然后对得到的数据进行统计分析。

[4]http://www.speech.sri.com/projects/srilm/

7.1 Oracle BLEU

为了评估解码算法用于重排序的潜力，我们计算了开发集上的oracle BLEU值。在表1中，*pseudo*表示未采用新的n-best解码算法的老解码器的伪n-best 输出，采用新算法的解码器用*true*表示）。列n和列*oracle BLEU*分别表示输出结果的个数和开发集上对应的oracle BLEU值。我们对每一组给定的n在开发集上训练5次参数，然后记录下在开发集上取得最高BLEU值的那组参数来计算开发集上的oracle BLEU值。

实验结果反映了越大的n能获得更高的oracle BLEU值。我们注意到，当n相同时（实验中都设置成10），输出真正n-best的新解码器比原来输出伪n-best的解码器能获得更高的oracle BLEU值，这意味着新解码算法在重排序后处理过程中有获得更好结果的潜力。

7.2 对统计机器翻译的影响

我们使用Lynx解码器，使用最小错误率训练程序[4]训练模型参数。实验结果列在表1中。其中列*dev BLEU*和列*tst BLEU*分别表示开发集上的BLEU值和测试集上的BLEU值。

在表1中，我们注意到当n大于10时，开发集和测试集上的BLEU值都比n小于等于10时有显著的提高。这个结果说明了新算法输出真正的n-best结果可以让最小错误率训练程序训练得到更好的参数。但我们同时注意到，当n等于100，200和500时，BLEU值对于不同的n在统计上没有显著的差别。出现这个结果可能是因为在1-best搜索步骤中，一些较好的推导已经被裁减掉了，所以在n-best搜索步骤中不可能再找到那些裁减掉的推导。我们表1中的各项BLEU值的标准差可以看出当n较大时，最小错误率训练程序能较稳定地训练得到好的模型参数。

8 结论

本文提出了一个高效的针对基于树到串模板翻译模型的n-best解码算法。这个新解码算法不但提高了解码器在开发集上的oracle BLEU值，而且使得最小错误率程序能够找到更好的模型参数，从而提高系统的翻译质量。

参 考 文 献

[1] Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9st International Workshop on Parsing Technology*, pages 53–64.

[2] Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July.

[3] Jonathan May and Kevin Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 351–358, New York City, USA, June.

[4] Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics*, pages 160–167.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics*, pages 311–318, Philadephia, USA, July.

[6] Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology Conference and the 5th Meeting of the North American Association for Computational Linguistics*, Boston, USA.