基于统计的机器翻译在嵌入式系统上的实现①

Implementation of Statistic-Based Machine Translation on Embedded System

曾微维[1]　郑善贤[1]　成　钢[2]　(1.湖南大学 电气与信息工程学院 湖南 长沙 410082;
2.康明电子有限公司 上海研发部 上海 200020)

摘　要： 由于技术上的限制，基于统计的机器翻译系统没有在嵌入式平台上得到广泛应用。通过分析机器翻译解码的原理，结合相关实验，找到基于统计的机器翻译系统在嵌入式平台上实现的瓶颈，提出优化方法，使得基于短语统计的机器翻译能在嵌入式平台上有着满意的效果。

关键词： 机器翻译 嵌入式系统 基于统计

1　引言

应用一直是研发的最终目的。将机器翻译普遍应用正是机器翻译的研究意义所在。在当前全球化的趋势下，国际间的交往和各个领域间的合作日益频繁和深入，语言的差异成了严重的障碍，机器翻译的重要性逐渐被人们认识，同时，也被称为要在 21 世纪解决的科技难题之一。

目前，机器翻译已经在 PC 上广泛应用，很多软件和网站都提供了机器翻译的功能，虽然目前还不能完全替代人工翻译，但是已经极大的减小了人们阅读外文的障碍。随着嵌入式设备的普及，各种复杂的应用不断从 PC 平台转向嵌入式平台，同样，机器翻译也已经出现在嵌入式设备上，目前嵌入式设备上机器翻译系统主要是基于语言规则的，这种方式不要庞大的库文件，其解码算法所需要的时间也相对少，但是这种翻译系统本身存在一些难以解决的缺陷；其次这种方法没有统一的模型，很难支持多国语言互译[1]。

相比之下，基于短语统计的机器翻译能够较好的支持多国语言互译，而且翻译质量也比基于语言规则的机器翻译好，但是翻译的过程需要的时间比较长，本文将从理论入手，通过反复实验，调整解码过程的部分参数以及对主要算法进行优化，在保证翻译质量的情况下，提高翻译速度，在嵌入式设备上实现基于短语统计的机器翻译系统。

2　基于短语统计的机器翻译系统概述

2.1 系统结构

在 1945 年 W. Weaver 提出机器翻译的信源信道模型思想中(公式 1)[2]，一个完整的统计机器翻译系统主要由语言模型、翻译模型、解码器和一些预处理以及翻译后处理模块组成。如图 1 所示。

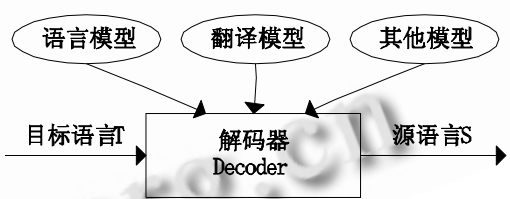

图 1　统计型机器翻译系统结构

在主要的 3 大模块中，语言模型的主要作用是利用单语语料库训练语言模型参数表，在解码翻译时完成语言模型(令语言模型为 Pr(S),S 为待翻译的语句)的计算问题；翻译模型的作用是利用双语单词对齐的语料库，训练翻译模型的参数表，重新生成一个短语语料库，在解码时完成翻译模型(令翻译模型为 Pr(T|S),即输入语句 S 被翻译成目标语句 T 的条件概率) 的计算问题，在该短语语料库中，每个数据项的内容包括(以中英文为例)中文短语，对应英文短语，以及他们之间的翻译概率等参数，我们将该库中的每个数据项称之为翻译选项；解码器是实际完成翻译过程

① 收稿时间:2008-12-25

的软件程序，解码器根据语言模型 Pr(S) 和翻译模型 Pr(T|S) 计算目标语言文本T 到源语言文本S 的翻译概率，并利用某种策略找出翻译概率最大的源语言文本 S[3]。

$$\hat{S} = \arg\max \Pr(S)\Pr(T \mid S) \quad (1)$$

一个输入语句可能存在很多种翻译结果，公式(1)就是指在众多的可能的翻译结果中，综合考虑语言、翻译等模型对翻译结果影响，找出翻译概率最大的一种翻译方式，是机器翻译的本质过程和目的。

2.2 翻译概率

在翻译概率计算公式上，我们引入了最大熵模型思想[4]。对于语言模型和翻译模型，在最大熵模型中，我们都只是作为一个普通的因素进行考虑。在这里，除了语言模型 $P_{LM}(e)$ (P:概率，LM:Language Mode,e 为进行语言模型概率计算的输入语句)和翻译模型 $P_T(f \mid e)$ (即上一节提到的 Pr(T|S))，我们还考虑位置变化模型 $P_D(e,f)$ (P:概率，D:Distance,e:翻译后的译文，f:待翻译的原文，后文公式中出现的 e,f 均表示此意)、长度惩罚模型 $\omega^{length(e)}$ (长度惩罚模型对根据译文的单词长度进行惩罚，影响翻译结果，该模型的公式请参阅其他资料)等影响翻译概率的因素。

综合这些因素，我们给出解码器中用到的翻译概率计算公式如下：

$$P(e \mid f) = P_T(f \mid e)^{\lambda_T} \times P_{LM}(e)^{\lambda_{LM}} \times P_D(e,f)^{\lambda_D} \times \omega^{length(e)\lambda_{W(e)}} \quad (2)$$

通过 GIZA++(GIZA++是一个完成 IBM 翻译模型参数训练的软件程序)训练后生成对齐文件，由此对齐文件生成的短语语料库中的每个短语都有 4 个参数，分别是 $\phi(\bar{e} \mid \bar{f})$ (正向短语翻译概率)、$P_w(\bar{e} \mid \bar{f})$ (反向短语翻译概率)、$\phi(\bar{f} \mid \bar{e})$ (正向翻译单词概率)、$P_w(\bar{f} \mid \bar{e})$ (反向翻译单词概率)。在解码器中，进行翻译模型概率计算时，这 4 个参数值我们都会运用到，而且我们也可以对它们进行不同的权值分配。这样，实际上 $P_T(f \mid e)$ 用下面的式子进行计算：

$$P_T(f \mid e) = \phi(e \mid f)^{\lambda_1} \times P_w(e \mid f)^{\lambda_2} \times \phi(f \mid e)^{\lambda_3} \times P_w(f \mid e)^{\lambda_4} \quad (3)$$

其中，$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$。

2.3 解码过程

解码过程分为以下四大部分：

a) 将输入的句子进行分词。这里主要是针对单词间没有分隔符的语言(如中文)进行单词的划分。而英文等单词间已经有分隔符的语言，则分词的处理要容易很多。

b) 生成翻译选项表。找寻经过分词后的句子有可能组成的全部短语的翻译选项。

c) 利用 beam search[5]算法，来寻找最优的翻译路径。该过程用伪算法描述如下：

```
initialize hypStack[0..n]//申请堆内存
create initial hypothesis hyp_init;//初始化
add hyp_init to stack hypStack[0];//从栈 0 开始循环遍历找出最佳翻译路径
for i=0 to n-1:
for each hyp in hypStack[i]:
for each new_hyp that can be derived from hyp:
nf[new_hyp] = number of Chinese words covered by new_hyp;
add new_hyp to hypStack[nf[new_hyp];
prune hypStack[nf[new_hyp]];
find best hypothesis best_hyp in hypStack [n];
output best path that leads to best_hyp; //输出翻译语句
```

d) 根据最优翻译路径，反向生成目标语言的句子。

3 分析与优化

基于短语统计的机器翻译系统在嵌入式设备上实现的瓶颈在哪里？我们的方法是先将 PC 上已有的算法不加任何原理上的改变，让其在嵌入式平台上运行，并针对翻译结果和翻译过程的时空间耗费来找到存在的问题以及分析问题可能的原因。

3.1 时间测量

Koehn 曾通过实验得出的结论，在最大短语长度大于 3 时，翻译质量几乎不受该参数影响[5]。因此我们的最大短语长度定义为 3。测试语句为 50 句。这 50 个句子分词后平均单词个数 n=15(14.6 取整)。并

设定中间状态栈深度 m=100，每个短语所对应的最大翻译选项的个数 p=50。

作为对比，我们同时测量 PC 下，各个阶段的时间消耗。

表 1　测试平台信息

平台	主芯片型号和工作频率(Hz)	存储介质读取速度(MB/秒)	
PC	XP3200+，2.0G	（硬盘）	43.7
嵌入式	ARM 9　266M	（Nand Flash）	8.5

表 2　解码各个时段时间消耗

翻译阶段	PC平台用时 (ms)	嵌入式平台下(ms)
分词	5	81
生成翻译选项表	1112	5040
寻找最优翻译路径	202	2905
反向生成翻译结果	1	13

由以上数据可知，无论是在 PC 平台还是在嵌入式平台，库的读取都占据了大部分时间，但是在嵌入式平台上，寻找最优翻译路径所需的时间比例有所增大。因此，针对于嵌入式平台，应该从生成翻译选项表和寻找最优路径两个方面一起入手来减少翻译时间。

3.2 解码过程的空间复杂度

嵌入式平台上的 RAM(Random Access Memory) 大小远远小于 PC 平台，因此，有必要解码过程的空间复杂度进行一个估算。

设一个句子分词后单词个数为 n，定义最大短语长度为 3，在最坏的情况下，一共有 3*(n-1)个短语，同时，每个短语又对应 p 个翻译选项，那么一共有 3p*(n-1)个翻译选项需要保存。

对于 beam search 的过程，所需要 RAM 空间大小为 BS=n*m*s，(s 为一个中间状态的对象所占的空间，一般不会超过 50)。搜索算法对中间状态的操作次数和栈的深度 m 以及句子长度 n 的平方成正比，当 m=100，n=50 时，BS=50*100*50=250KB，同时，p 取 50，并设，那么，保存所有的翻译选项所需要的最大内存空间为 3p*(n-1) 乘以每个翻译选项占字节数，设为 100)=3 *50*(50-1)*100 约为 750KB。加上中间状态栈所需的空间，约为 1MB，可见内存空间并不是机器翻译在嵌入式系统上实现的主要瓶颈。

3.3 参数调整

根据 3.2 对 beam search 的分析，寻找最优翻译路径的时间复杂度和其空间复杂度一样，是和中间状态栈的深度 m 成正比的。适当的降低 m 的取值可以加快整个 beam search 的过程，但是根据 beam search 的原理，当某个中间状态栈满时，会合并或者淘汰栈中的一些中间状态，被淘汰的中间状态在这一阶段的翻译统计概率比较低，但是由这些中间状态生成的后续状态则可能有着较高的翻译统计概率，过小的 m 取值会使得搜索过程很容易陷入一个局部的最优解。

经实验证明，当 m 从 100 降为 50 时，每个短语对应的最大翻译选项个数 p 从 50 降为 20，不会对翻译质量有着明显的影响，测试用到的 50 个句子的翻译结果，只有个别句子的翻译结果有很小的变化。比如，在原来的参数下，翻译 “你的意见，我是绝对不会接受的。”时，结果为：“Your advice , I will not accept.”，当改变参数后，翻译结果为：“Your opinion,I will not accept .”

3.4 对生成翻译选项表的优化

翻译选项表不仅包括在 beam search 中所需要的各个短语对应的翻译选项，同时还包括在搜索过程中，用于合并和淘汰中间状态算法所需要的各种参数，例如源句子中已经翻译的部分短语的翻译耗费，以及未翻译部分的翻译耗费估计值。经过对“生成翻译选表”模块中各个子模块的跟踪，发现主要的时间耗费是在翻译选项库文件的读取上，前面提到，当短语所包含最大单词长度定为 3 时，一个长度为 n 的句子，最多有 3n-3 个短语，我们现有的预料库中有 280 多万个短语，根据二分法的效率可知，每找到一个短语对应的翻译选项要比较的平均次数在 20 次以上，找到全部短语要比较的次数为 60*(n-1)，当句子长度 n 取 21，一共要进行 1200 次的二分法超找，如果仅仅是用二分法查找内存中

的数组，1200 次二分法的操作的时间是非常短的，几乎是可以忽略的，那么在嵌入式平台上生成翻译选项表所用的时间与在 pc 上生成翻译选项表所需时间的比例，应该是和两个平台上 beam search 所需要的时间比例是相同的，但是在 pc 上，库文件存储在磁盘介质上，而嵌入式平台上，是存储在 Nand flash 上。

由表 2 可以看出，这个部分的时间比例，约等于两个测试平台上存储介质的读取速率。如何减少搜索过程中读取短语语料库的次数成为该阶段优化的主要目标。我们采用建立一个二级索引表的方式，将 280 万条短语分成一万个等份，在初始化翻译系统时将这个索引表读入内存。相当于将库的大小减少 1 万倍，这样一次二分法从原来的 1200 次操作减少到不到 500 次，读短语语料库的时间不到原来的一半。

3.5 对库文件的优化

综合考虑 公式(2)和公式(3)以及约束条件，我们事先调整好约束条件中的 4 个权值，计算出公式(3)的结果，直接将该结果保存于库文件中，减小了库文件的存储空间，也减少了解码过程中的运算次数。

这么做的缺点是在不改变库文件的情况下，没有办法根据需要动态调整公式(2)中翻译模型的取值，但是平均翻译质量是最高的。

3.6 优化后的测试结果

通过以上优化，最后在嵌入式平台上测试的结果如下

表 3　优化后测试结果

翻译阶段	用时
分词	80　(ms)
生成翻译选项表	2200(ms)
寻找最优翻译路径	1550(ms)
反向生成目标语句	13　(ms)

表 4　部分测试语句翻译结果

中文句子及其对应翻译结果
请你再说一遍，好吗？
Would you repeat that, please?
当然，大家开始大叫起来很大声。
Of course, everyone began to shout very loudly.
答案是他们没有叫我吉姆。
The answer is they didn't call me Jim.
有时候人们问我谈我的名字。
Sometimes people ask me about my name.

4　结语

在测试嵌入式平台的平均每句翻译时间不到 4 秒，并且有着较高的翻译质量，这样的结果在嵌入式平台上还是可以接受的，通过不同的库文件，基于短语统计的机器翻译系统能够在很方便的在嵌入式系统上实现多国语言互译，在掌上电脑，电子辞典等嵌入式终端设备上有着实用价值。

同时，搜索最佳翻译路径的部分仍然存在较大的优化空间，采取更好的中间状态的合并和淘汰算法，可以让栈深度 m 变的更小;如果在进行 beam search 前就淘汰掉一部分翻译选项作为起始点，也会使得整个搜索过程更加迅速。

参考文献

1 赵铁军.机器翻译原理.哈尔滨:哈尔滨工业大学出版社, 2000.

2 Weaver W. Translation(1949).Machine translation of language 1995. MIT Press, Cambridge, MA.

3 Brown, Peter E, Cocke, etc. A statistical approach to machine translation. Computational Linguistics, 1990, 16(2):79－85.

4 Franz Josef OCH, etc. Discriminative training and maximum entropy models for statistical machine translation. Proceedings of ACL2002.

5 Koehn P. Pharaoh:A beam search decoder for phrase-based statistical machine translation models. Proceedings of the Association of Machine Translation in the Americas (AMTA 2004), October 2004.