基于微引擎流水线的机器翻译系统结构

刘 群
(中国科学院计算技术研究所 北京 100080)
(北京大学计算语言学研究所 北京 100871)

摘 要 该文比较了现有各种多引擎机器翻译方法的优缺点,提出了基于微引擎流水线的机器翻译系统结构,详细介绍了有关的数据结构和算法.这种结构的优点在于在部件层次上实现多重算法的并存,通过对微引擎的增删和流水线结构的调整可以方便地尝试各种机器翻译方法的组合,而不需要修改系统的整体算法.文章最后介绍了这种机器翻译系统结构在面向新闻领域的汉英机器翻译系统中的具体实现,给出了实验数据,并进行了总结.

关键词 多引擎机器翻译;微引擎流水线;机器翻译
中图法分类号 TP391

Machine Translation Architecture Based on Micro-Engine Pipeline

LIU Qun
(*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080)
(*Institute of Computational Linguistics, Peking University, Beijing* 100871)

Abstract This paper surveys current approaches of multi-engine machine translation and proposes a micro-engine pipeline machine translation architecture, gives the data structure and algorithm in detail. In such architecture several components with different algorithms are used in each phases of the system. The new approach can be tested under the architecture by adding a new micro-engine or adjusting the pipe structure, without changing the overall algorithm of the system. An implementation of this architecture in a news-oriented Chinese-English machine translation system and the experiment results are given, followed by a conclusion.

Keywords multi-engine machine translation; micro-engine pipeline; machine translation

1 引 言

由于各种不同的机器翻译方法各有特长,也各有缺点,没有哪一种单一的机器翻译方法能够达到理想的效果,因此采用多引擎的方法,希望各种方法能够互补,以达到总体效果的最优,就成为了一种自然的选择.目前多引擎的机器翻译已经被广泛采用,而实践证明这种方法也确实有效.

目前常用的多引擎机器翻译系统主要有三种结构形式,即并行的结构、串行的结构和混合的结构.

在并行结构的多引擎机器翻译系统中,各个翻译引擎各自独立地对输入的文本进行翻译,并将翻译的结果放到一个统一的数据结构中,最后由一个译文选择模块选择出最好的译文组合.

Frederking[1] 提出了一种典型的并行多引擎机器翻译的方法.该方法基本思想描述如下:

1. 多个翻译引擎同时对输入的句子进行翻译,不仅仅对整个句子进行翻译,而且对句子的任何一个片断也可以给出相应的译文,同时对这些译文片断给出一个评分.

2. 各个翻译引擎共享一个类似线图的数据结构,根据其源文片断所处的位置,将这些译文片断放在这个公共的线

收稿日期:2003-02-24;修改稿收到日期:2004-01-12.本课题得到国家"九七三"重点基础研究发展规划项目基金(G1998030507-4,G1998030510)资助.刘 群,男,1966 年生,博士研究生,副研究员,研究领域包括自然语言处理、中文信息处理、机器翻译、信息提取、语言资源与评测技术等.. E-mail:liuqun@ict.ac.cn.

图结构之中.

3. 对各个引擎给出的片断的评分进行一致化处理, 使之具有可比较性.

4. 采用一个动态规划算法(称为 Chart Walk 算法)选择一组刚好能覆盖整个源文输入句子, 同时又具有最高总分的译文片断, 作为最后输出的译文.

系统结构如图 1 所示.

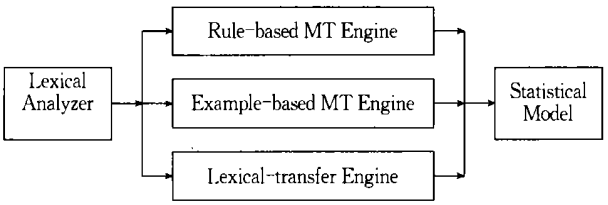

图 1 并行多引擎机器翻译系统结构

Hogan[2] 通过一个简单的实验, 证明这种方法确实可以得到比任何一种单一的方法都更高的准确率.

美国卡内基·梅隆大学等单位研制的一个著名多引擎的西班牙-英语的机器翻译系统 PANGLOSS 就是采用的这种结构[3]. 该系统总共包括三个翻译引擎: 一个基于转换的翻译引擎、一个基于知识(中间语言)的翻译引擎和一个基于实例的翻译引擎. 其系统结构如图 2 所示.

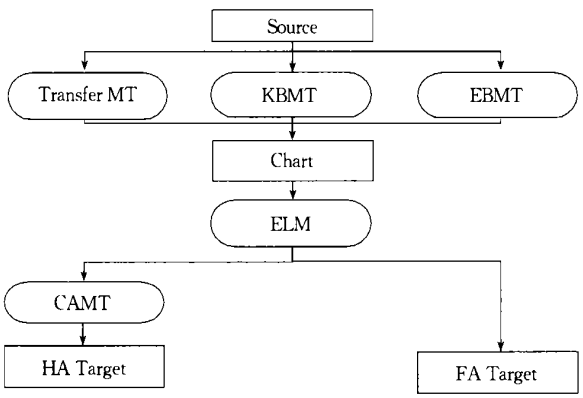

图 2 Pangloss 多引擎机器翻译系统的结构

在很多多引擎的机器翻译系统中, 并不是采用完全独立的多个翻译引擎对源文进行翻译, 而是在机器翻译的不同阶段采用不同的算法, 例如, 在句法阶段采用基于规则的方法, 在转换阶段采用基于实例的方法, 而在生成阶段采用基于统计的方法. 我们把这一种结构称为串行的多引擎机器翻译结构. 在这种情况下, 每个引擎实际上是翻译系统的一个部件, 并不独立完成翻译任务[4,5].

还有很多系统采用的是一种混合的结构, 并行中有串行(并行的多个翻译引擎之一又采用串行的多引擎结构), 串行中有并行(串行的多个翻译部件之一又采用多个组件并行), 形成一种复杂的体系结构.

并行的机器翻译结构中各个翻译引擎的颗粒度非常大, 引擎之间的结合非常松散, 一个翻译引擎无法引用另一个翻译引擎的中间结果, 这严重限制了整个系统性能的提高. 因此, 采用这种方法的系统实际上比较少见, 大多数多引擎的机器翻译系统实际上都是采用后两种结构.

不过, 并行的多引擎机器翻译方法有一个突出的优点也是另外两种方法所不具备的, 就是其易扩充性. 在这种结构下, 各个翻译引擎的程序接口完全相同, 添加和删除新的翻译引擎变得非常简单, 这使得程序的扩充变得非常容易.

而在串行和混合的多引擎机器翻译结构中, 各个翻译引擎(部件)由于实现的功能不尽相同, 各个翻译引擎之间存在复杂的通信关系, 翻译引擎无法采用统一的程序接口, 这使得程序的扩充变得非常困难.

由德国教育与研究部(BMBF)资助开发的 Verbmobil 语音机器翻译系统就是一个典型的混合结构的多引擎机器翻译系统[6]. 该系统规模非常庞大, 整个系统的研制为期 8 年(1993 ~ 2000), 涉及三种语言(德语、英语、日语)的双向翻译. 世界三大洲的 31 个研究机构、369 名科学家和 919 名学生(硕士生、博士生和博士后)参与了这个项目的研究. 系统采用的技术也非常庞杂, 语音处理领域和自然语言处理领域中常见的各种技术都在这个系统中有所反映. 整个系统由 69 个互相交互的模块构成. 其中用到的自然语言处理技术包括组块分析、概率 LR 分析、HPSG 分析、对话行为(dialog act)分析、基于统计的翻译、基于子串(substring)的翻译、基于模板的翻译、基于模板的转换、语义分析、上下文相关歧义的消解、基于规划的话语生成等等.

为了解决翻译引擎之间通信的问题, Verbmobil 系统采用一种多黑板结构用于模块之间的数据交互, 模块之间不能直接通信. 黑板结构还有利于各个模块之间的并行执行. 总共采用了 198 个黑板结构用于 69 个不同模块之间的通信. 一种叫做 VIT(Verbmobil Interface Terms)的数据结构在中心黑板的深层处理中用作深层的语义表示形式. 该系统的整体结构如图 3 所示.

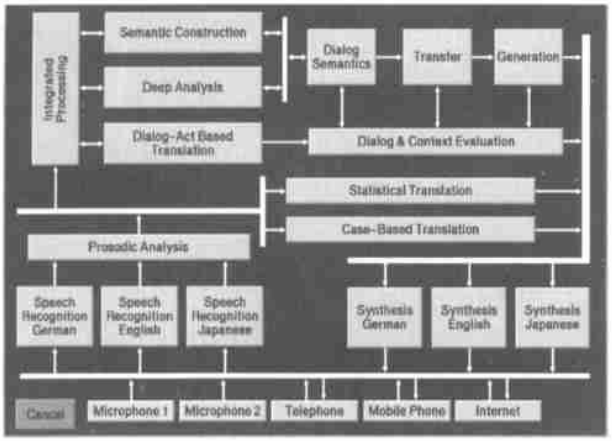

图 3　Verbmobil 语音机器翻译系统结构

可以看到, 由于黑板的设定是比较随意的, 整个系统的复杂程度依然很高, 模块的划分还不是非常清晰, 系统的可扩充性也不是很好.

2　微引擎流水线的机器翻译系统结构

在“面向新闻领域的汉英机器翻译系统”[7,8]中, 我们设计了一种“基于微引擎流水线的机器翻译系统结构”. 这种结构本质上也是一种混合的多引擎机器翻译系统结构. 与一般的混合多引擎结构不同之处在于, 我们为每一个机器翻译引擎(我们称为微引擎) 定义了统一的几类接口, 并给出了清晰的引擎调度算法, 即总体翻译算法. 微引擎的增加、减少和修改都变得非常简单, 并且一个微引擎的调整不会对其他微引擎的算法和总体翻译算法造成干扰, 这样, 系统的扩充变得非常容易.

2.1　微引擎流水线的程序模块结构

一个微引擎流水线的程序模块结构由以下一个七元组构成:

$$\{R, S, T, G, \Sigma, \Omega, \Psi\},$$

其中, R 是一个“识别器(Recognizer)”的集合;

S 是一个“选择器(Selector)”的集合;

T 是一个“转换器(Transferror)”的集合;

G 是一个“生成器(Generator)”的集合;

Σ 是一个向量: $\sigma_1 \sigma_2 \cdots \sigma_n$, 其中 $\sigma_i \in R \cup S$, $1 \leq i \leq n$;

Ω 是一个向量: $\omega_1 \omega_2 \cdots \omega_m$; 其中 $\omega_j \in G$, $1 \leq j \leq m$;

Ψ 是一个 $R \rightarrow T$ 的映射, 即对任意 $r \in R$, 存在唯一的 $t \in T$, 使得 $t = \Psi(r)$;

以上识别器(Recognizer)、选择器(Selector)、转换器(Transferror)、生成器(Generator) 统称为微引擎(Micro-Engine).

Σ 是一个由识别器和选择器组成的流水线, 称为分析流水线; Ω 是一个由生成器组成的流水线, 称为生成流水线.

整个微引擎流水线结构如图 4 所示.

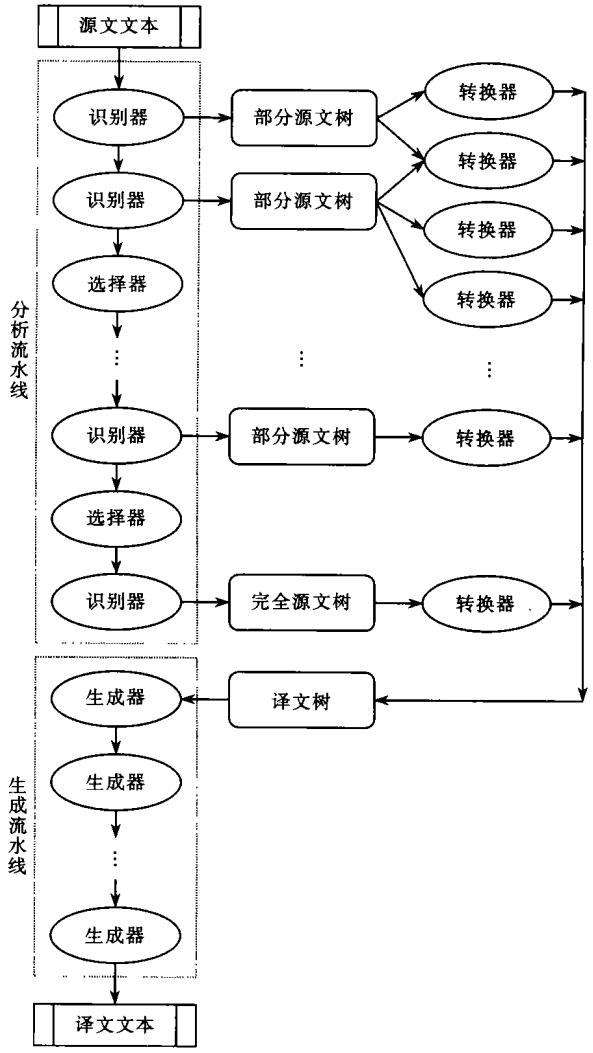

图 4　微引擎流水线机器翻译系统结构

由图 4 中可以看出, 识别流水线结构较为复杂, 由一系列识别器和选择器构成, 其中每个识别器又对应于一个转换器(分析器可以共享转换器). 生成流水线结构较为简单, 单纯由一系列生成器构成.

2.2　微引擎流水线的公共数据结构

微引擎流水线的公共数据结构总共包括两类: 一类是线图结构, 一类是句法树结构. 其中句法树结构又分为源文句法树和译文句法树.

线图结构如图 5 所示.

线图中涉及的数据结构定义见图 6.

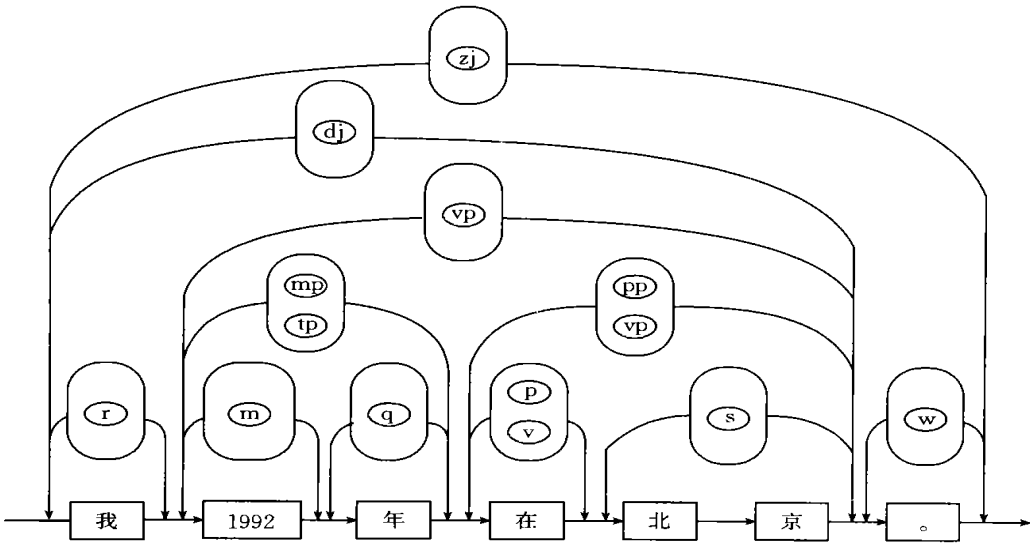

图 5 微引擎流水线的公共数据结构——线图

SrcSection	线图结构中相邻两个词语节点之间的一段文本, 是句子中可能构成词的最小单位, 一般是一个汉字, 或一串数字, 或一个外文单词.
SrcSectionTable	是一个由 SrcSection 组成的数组.
SrcNode	线图中的弧, 或句法树中的结点, 可以是一个词, 也可以是一个短语或句子; 每一个 SrcNode 由首尾两个 SrcSection 确定其位置.
SrcPackage	位置相同的所有 SrcNode 构成一个结点包, 存放在一个 SrcPackage 的数据结构中.
SrcPackageTable	是一个由 SrcPackage 组成的数组, 通过函数: getPackage(int first, int last) 可以存取任何指定位置的 SrcPackage, 参数 first 和 last 用于指定首尾 SrcSection 序号.
SrcChart	由一个 SrcSectionTable 和一个 SrcPackageTable 组成.

图 6 线图结构的数据说明

同时, 各 SrcNode 通过子结点关系构成源文句法树. 源文句法树经过转换生成就得到了译文句法树. 其形式与源文句法树类似.

2.3 各种微引擎的程序接口和功能说明

(1) 识别器(Recognizer)

识别器需要实现两个函数:

函数: 初始化(Initialize)

输入: 线图结构(SrcChart)

输出: 无

说明: 为识别操作准备初始数据, 每个识别器只需执行一次.

函数: 识别(Recognize)

输入: 线图结构(SrcChart)

输出: 一个源文结点(SrcNode)

说明: 从线图结构中识别出一个结点. 此操作被反复执行.

(2)选择器(Selector)

选择器需要实现两个函数:

函数: 初始化(Initialize)

输入: 线图结构(SrcChart)

输出: 无

说明: 为选择操作准备初始数据, 每个选择器只需执行一次;

函数: 选择(Select)

输入: 线图结构(SrcChart)

输出: 一个源文结点表(list〈SrcNodes〉)

说明: 从线图结构中选择一些结点放入输出的源文结点表中, 凡是不在该表中的结点将在后续的操作中不再有效(被剪枝). 要注意的是, 选择器并不要求输出唯一的结果, 暂时无法解决的歧义结点完全可以都保留下来, 留给以后处理;

(3)转换器(Transferror)

转换器需要实现两个函数:

函数: 初始化(Initialize)

输入: 一个源文结点(SrcNode)

输出: 无

说明: 为转换操作准备数据. 由于转换所需的与结点有关的数据都存放在 SrcNode(或其派生类) 中, 因此此操作需对每个 SrcNode 执行一次.

函数: 转换(Transfer)

输入: 一个源文结点(SrcNode)

输出: 一个译文结点(TgtNode)

说明: 对以输入的源文结点为根结点的源文子树进行转换, 得到一个译文子树, 并输出译文子树的根结点. 此函数可通过递归调用, 实现对其子孙结点的转换.

(4) 生成器(Generator)

生成器需要实现两个函数:

函数: 初始化(Initialize)

输入: 一个译文结点(TgtNode)

输出: 无

说明: 为生成操作准备数据. 此操作需对每个 TgtNode 执行一次.

函数: 生成(Generate)

输入: 一个译文结点(TgtNode)

输出: 另一个译文结点(TgtNode)

说明: 对以输入的译文结点为根结点的译文子树进行某种特定类型生成操作, 并输出所得的新译文子树的根结点.

2.4　总体翻译算法

对于微引擎流水线结构的机器翻译系统来说, 总体翻译算法是固定的, 不需要修改. 对于翻译系统的调整主要体现在微引擎的实现算法和流水线的安排上.

整个翻译算法分为分析、转换、生成三个步骤.

算法 **1**.　分析算法.

```
BEGIN
  REPEAT 依次从分析流水线中取一个微引擎
  WHILE 该微引擎不为空
    IF 该微引擎是识别器
    THEN
      调用该识别器的初始化函数
      REPEAT 调用该识别器的识别函数
        IF 识别出的结点覆盖整个输入文本
        THEN 返回成功, 将该结点置为源文根结点
        ELSE 将识别出的结点加入到线图结构中
        ENDIF
      ENDREPEAT
    ELSE
      调用该选择器的初始化函数
      调用该选择器的选择函数
      根据返回的结点表重新构造线图结构, 删除不在表中的结点
    ENDIF
  ENDREPEAT
  返回失败
END
```

算法 **2**.　转换算法.

```
BEGIN
  取源文根结点的识别器
  取该识别器对应的转换器
  调用该转换器的初始化函数, 对源文结构树根结点进行转换初始化
  调用该转换器的转换函数, 对源文结构树进行转换
  返回得到的译文结点, 作为译文结构树根结点
END
```

算法 **3**.　生成算法.

```
BEGIN
  REPEAT 依次从生成流水线中取一个生成器
  WHILE 该生成器不为空
    调用该生成器的初始化函数, 对译文结构树根结点进行生成初始化
    调用该生成器的生成函数, 对译文结构树进行生成
    将返回的译文结点作为新的译文结构树根结点
  ENDREPEAT
  返回译文根结点
END
```

可以看到, 在分析过程中, 各个识别器依次对输入文本进行处理, 由于各个识别器采用的算法不同, 各种算法取长补短, 尽可能得到一个较好的分析结果. 而选择器用于对过多的识别结果进行剪枝排歧, 以减少搜索的空间.

在转换算法中, 采用的方法是自顶向下的一遍扫描. 不同的转换器用于对不同类型的识别器产生的结点进行转换.

在生成算法中, 采用的方法是自顶向下的多遍扫描. 每一个生成器都要对整个译文结构树进行一次扫描.

3　具体实现方案

目前在我们现有的"面向新闻领域的机器翻译系统"中, 已经实现了这种基于微引擎流水线的机器翻译系统结构. 其实现方案如下:

分析流水线依次由以下识别器构成:

词法分析识别器: 一个融汉语词语切分、未定义词识别、词性标注为一体的词法分析模块;

扩充词典识别器: 我们综合了几十部词典, 采用自动整理加少量人工校对的办法, 合成了一部大规模的汉英词典[10]. 对于词法分析识别器没有识别出的词或短语, 可以利用这部词典作为补充;

短语库识别器:利用一个短语库识别句子中的短语;

基于规则的识别器:利用一套句法分析规则和线图分析算法识别句子中的短语;

软失败识别器:在无法将输入句子分析得到一个完整的结点时,使用软失败识别器将已有的结点尽可能组合成一个覆盖整个句子的结点.

转换器包括以下几种:

核心词典转换器:根据核心词典进行转换;

扩充词典转换器:根据扩充词典进行转换;

中国人名转换器:中国人名的翻译;

中国地名转换器:中国地名的翻译;

译名转换器:译名的翻译;

数词转换器:数词的翻译;

短语库转换器:根据短语库进行转换;

基于规则的转换器:根据转换规则进行转换;

软失败转换器:对软失败识别器产生的结点进行转换.

生成流水线依次由以下生成器构成:

基于规则的结构生成器:根据原有基于规则的机器翻译系统中的结构生成模块改造而来,采用一种基于合一的语法形式进行英语句法结构的调整,主要进行局部词序的调整和根据时态和语态添加助动词;

基于规则的词语生成器:生成英语词语的变形,包括动词的分词形式和动名词形式、名词复数形式、形容词比较级和最高级等等.

可以看到,目前的微引擎流水线结构还是比较简单的,主要是在已有的基于规则的机器翻译系统基础上扩充整理而成.也还没有用到选择器.

4 实验结果及分析

这个系统的基础是我们原先开发的一个基于合一语法的机器翻译系统.在现在这个基于微引擎流水线的机器翻译系统中,原先的机器翻译系统被拆分成若干个微引擎,并和新增加的微引擎合成了一个整体.

我们利用了一个美国国家标准与技术研究所(NIST)开发的机器翻译自动评测程序来对我们的系统在不同的微引擎组合情况下的翻译结果进行了测试.NIST 的自动评测系统源于 IBM 在其机器翻译自动评测程序 BLEU 中提出的基于 N 元语法的机器翻译自动评测的思想[9].其基本设想是首先请多位翻译人员(一般至少 4 位)对被测试的语料进行翻译,然后将机器翻译的译文跟这些人工翻译得到的参考译文进行比较,计算机器译文中的单词、2 元单词组、3 元单词组、…、N 元单词组在参考译文中出现的比例,从而得到对机器译文的评价.NIST 对 BLEU 的改进主要是对每个 N 元语法根据其所包含的信息量进行了加权,从而使得测试系统对不同质量的译文比 NIST 具有更好的区分能力①.

表 1 是我们利用 NIST 的测试程序,针对不同的微引擎组合情况,对机器翻译系统产生译文测试的结果.

表 1 各种微引擎组合情况下的 NIST 测试结果

对词法分析识别器 使用扩充词典转换器	使用短词库识别器和 短词库转换器	使用扩充词典识别器 和扩充词典转换器	使用规则识别器 和规则转换器	翻译 时间(ms)	NIST 评分
*	*	*	*	3293172	5.8697
*	*	*		103609	5.4669
*	*		*	962594	5.7706
*	*			97250	5.2793
*		*	*	3386672	5.8351
*		*		43110	5.4073
*			*	666516	5.6695
*				27515	5.1449
	*	*	*	3397828	5.4332
	*	*		96797	5.0443
	*		*	906187	5.3243
	*			97937	4.8343
		*	*	3224968	5.3866
		*		52516	4.9869
			*	658859	5.2121
				36172	4.6993

表 1 中,第 1 列如果没有选中(对应单元格为空),表示只将切分标注产生的词送到核心词典转换器进行转换,而不送到扩充词典转换器,如果选中

① 关于 NIST 评测系统的详情请访问网站:http://www.nist.gov/speech/tests/mt,上面有论文:Automatic Evaluation of Machine Translation Quality: Using n-gram Co-occurence Statistics, Research Report for NIST MT Evaluation 和相关的测试软件(用 Perl 语言编写).

(对应单元格为 *), 表示将切分标注产生的词, 在通过核心词典转换器进行转换失败时, 送到扩充词典转换器进行转换; 第 2 列表示是否使用短语库识别引擎和短语库转换引擎; 第 3 列表示是否使用扩充词典识别引擎和扩充词典转换引擎; 第 4 列表示是否使用规则识别引擎和规则转换引擎. 最后两列分别是翻译时间和 NIST 评分.

从上面的结果可以看到, 仅使用一部核心词典和汉语切分标注程序, 结果评分即可达到 4.6993. 仅加入规则引擎, 结果评分可达到 5.2121, 仅加入扩充词典和短语库, 不使用规则引擎, 结果评分可达到 5.4669, 加入全部微引擎, 结果评分可达到5.8697. 这个结果告诉我们, 通过扩充词典导致的翻译效果改善甚至会大于加入翻译规则导致的翻译效果改善, 而综合使用所有引擎, 翻译效果有很大提高.

通过这个实验可以看到, 在微引擎流水线的机器翻译结构下, 我们可以方便地实现翻译微引擎的自由组合, 通过对不同组合情况下的结果进行评分, 我们很容易了解各个微引擎在系统中所起到的作用, 这特别有利于我们对各种翻译算法进行取舍和调整.

5 总　结

在本文中, 我们提出并实现了一种微引擎流水线的机器翻译体系结构, 其优点如下:

(1) 在机器翻译的每个阶段, 都可以采用多个算法不同的微引擎, 以达到取长补短的效果;

(2) 句法分析阶段, 将分析器和选择器的功能分开; 分析器只关注可能性, 无需考虑与其他结点的冲突; 选择器专门处理结点间的冲突, 这种分工有助于在系统设计中对排歧问题进行更全面考虑, 更有益于歧义冲突的解决;

(3) 每个微引擎的编程接口非常清晰, 微引擎设计者在了解自己的任务分配的情况下, 可以着重关注于算法本身, 而无需考虑与其他模块的交互, 这样就把复杂的机器翻译问题分解成了一系列可以独立处理的小问题, 化繁为简, 有助于对机器翻译系统进行团队式开发, 有利于探索机器翻译中的新算法和新思路;

(4) 采用面向对象的编程方法, 开发一个新的微引擎只需在已有的微引擎基类的基础上派生出新的子类, 可靠性高, 易于实现;

(5) 整个的机器翻译算法是固定的, 任何时候都无需改变; 通过设计新的微引擎和调整微引擎流水线的结构, 可以实现对翻译系统功能的任意裁减, 以产生不同的输出结果(如产生切分标注结果的输出、句法分析结果的输出等).

参　考　文　献

1 Frederking R., Nirenburg S.. Three heads are better than one. In: Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP-94), Stuttgart, Germany, 1994, 95~100

2 Hogan C., Frederking R.. An evaluation of multi-engine MT architecture. In: Proceedings of the 3rd Conference of the Association for Machine Translation in Americas (AMTA'98), Langhorne, PA, 1998, 113~123

3 Brown R., Frederking R.. Applying statistical English language modeling to symbolic machine translation. In: Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95), Leuven, Belgium, 1995, 221~239

4 Zhang Min, Choi Key-Sun. Multi-engine machine translation: Accomplishment of MATES/CK system. In: Proceedings of TMI99, Chester, 1999, 228~238

5 Hatzivassiloglou V., Knight K.. Unification-based glossing. In: Proceedings of the 14th International Joint Conference Artificial Intelligence, Montreal, Quebec, 1995, 1382~1389

6 Wahlster W.. Mobile speech-to-speech translation of spontaneous dialogs: An overview of the final verbmobil system. In: Wahlster W. eds.. Verbmobil: Foundations of Speech-to-Speech Translation, ISBN 3-540-67783-6, Berlin: Springer, 2000, 3~21

7 Liu Qun, Chang Bao-Bao, Zhan Wei-Dong, Zhou Qiang. A news-oriented Chinese-English machine translation system. In: Proceedings of International Conference on Chinese Computing (ICCC2001), Singapore, 2001, 386~389

8 Liu Qun. A Chinese-English machine translation system based on micro-engine architecture. In: Chan Sin-Wai Eds.. Translation and Information Technology. Hong Kong: The Chinese University Press, 2002, 23~30

9 Papineni K., Roukos S., Ward T., Zhu Wei-Jing. Bleu: A method for automatic evaluation of machine translation. IBM Research, RC22176 (W0109-022), 2001

10 Liu Qun, Zhang Tong. Construction of Chinese-English expanded dictionary for machine translation system. In: Huang He-Yan Eds.. Development on Machine Translation Research (Proceedings of National Symposium on Machine Translation). Beijing: Publishing House of Electronics Industry, 2002, 25~33(in Chinese)
(刘　群, 张　彤. 汉英机器翻译系统扩充词典的建造. 见: 黄河燕编. 机器翻译研究进展(全国机器翻译研讨会论文集). 北京: 电子工业出版社, 2002, 25~33)

LIU Qun, born in 1966, associate professor of Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He got his B. S. in Computer Science in University of Science and Technology of China in 1989, and his M. S. in Computer Science in ICT, CAS in 1992. He is studying for Ph. D. in Peking University as an on-job student from 1999. His research interests include natural language processing, Chinese language processing, machine translation, information extraction, language resources and evaluation technology.

Background

Although varied technologies are developed in machine translation, none of them reach an optimal output on free text. So multi-engine MT approaches have been proposed to integrate several MT engines in one system. Currently, there are three kinds of multi-engine MT (MEMT) approaches, which we named as parallel MEMT, serial MEMT, and hybrid MEMT. For parallel MEMT, the modularity of the system is good, but the engines are coupled loosely and cannot benefit from each other. On the contrary, in a serial MEMT or hybrid MEMT, the engines are coupled tightly, but the system is hard to be defined in a good modularity. This paper proposes a micro-engine pipeline approach for machine translation. In such an architecture, four types of normalized MT components which are called micro-engines are defined. The micro-engines are arranged as a pipeline, and several micro-engines of same kind can be used in each phase of the system. The system can be reconstructed by simply adding or deleting the micro-engines, or reordering the micro-engines in the pipeline, without changing the main algorithm of the system.