计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(9):268−272 [doi: 10.15888/j.cnki.csa.006563]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于规则的多语种音译软件设计①

梁建飞, 王军元, 刘　敏

(61243 部队, 新疆 830006)
通讯作者: 梁建飞, E-mail: 284893203@qq.com

摘　要: 为了快速获得音译结果, 借鉴人工处理方式, 开发了基于规则的多语种音译软件. 该软件采用算法和规则独立设计的思想, 能满足多种语言的音译需求. 完整的音译过程包括单词预处理、字母识别与切分、字母重组与定位、规则表查询四个步骤. 在字母重组阶段提出一种确定最佳音节划分的方法, 有效解决了音节划分错误较多的难题, 从而保证了最终音译结果的质量. 通过对英语、罗马语和俄语进行分组实验, 经人工检查, 音译正确率达到95% 以上.

关键词: 规则表; 多语种; 音译; 音节划分; 双语地图

引用格式: 梁建飞,王军元,刘敏.基于规则的多语种音译软件设计.计算机系统应用,2018,27(9):268–272. http://www.c-s-a.org.cn/1003-3254/6563.html

Design of Multilingual Transliteration Software Based on Rules

LIANG Jian-Fei, WANG Jun-Yuan, LIU Min

(61243 Troops, Xinjiang 830006, China)

Abstract: In order to obtain the transliteration results quickly, a multi-lingual transliteration software based on rules is designed on the basis of manual processing in this study. The software can meet the transliteration needs of various lingual, because algorithm and rule are designed separately. The complete transliteration process includes four steps of word pre-treating, letter recognition and segmentation, letter recombination and localization, and rule table searching. In the letter recombination, this study proposes a method of determining the best syllable division, which can reduce the error rate of syllable division effectively and improve the quality of final transliteration results. The results of the experiments for English, Roman, and Russian in this study show that the transliteration accuracy can reach to 95% or more.

Key words: rule table; multilingual; transliteration; syllable division; bilingual map

音译通常用于命名实体的翻译, 如地名、人名等, 是指利用源语言及目标语言发音规则的异同将源语言形式翻译成目标语言形式[1]. 由于音译从读音角度处理翻译问题, 在处理未登录词翻译问题上有着良好的效果[2], 因此在很多跨语言任务如机器翻译[3]、双语地图[4]中有着广泛的应用, 其中双语地图对音译的要求非常高, 必须遵循严格的音译规则才能得到可供使用的双语注记[4].

目前有大量音译规则可供使用, 如联合国地名标准化会议要求: 世界各国的地名, 要实行单一罗马化拼写, 即以罗马语为官方语言的国家, 应提供标准的地名拼写形式; 使用非罗马语的国家, 要制定出非罗马字母转写为罗马字母的转写规则, 经联合国标准化会议通过在国际上推广使用. 在此基础上, 我国国家质量技术

① 收稿时间: 2018-01-31; 修改时间: 2018-02-27, 2018-03-13; 采用时间: 2018-03-21; csa 在线出版时间: 2018-08-16

监督局最新发布了《外国地名汉字译写导则》, 包括英语、法语、德语、俄语、西班牙语、阿拉伯语、葡萄牙语、蒙古语共八种. 除此之外, 中国地名委员会制定了 50 个语种的音译规则表, 这些规则表依据原语种字母罗马化后的不同发音所对应的国际音标, 分为元音字母和辅音字母, 由元音字母和辅音字母单独发音或组合发音进行译写.

针对音译的研究一般分为音译等价对挖掘的研究和音译模型构建的研究两大类, 前者是指从平行或可比较语料库中挖掘双语音译等价对, 构建一个更大更新的音译词典, 后者是指使用平行双语语料库进行训练, 根据其本身信息及上下文信息自动建立一个音译模型.

根据音译的方式不同, 音译方法可分为基于发音的方法和基于字形的方法[5]. 前者如文献[6], 利用源语言发音规则将源语言转换为发音中间体, 然后根据目标语言的发音规则将中间体转换为目标语言; 后者如文献[7], 直接由源语言不经过任何中间体转换为目标语言, 这类方法信息丢失最少, 音译效果最优. 文献[8]综合上述两种方法的优点, 提出将音节和字形特征相融合的方法.

根据音节划分的粒度, 音译方法可分为以音素为粒度的方法和以字母为粒度的方法. 如 Knight 和 Graeh[6]在日英人名音译中, 提出以英文音素为粒度, 通过发音相似性寻求转换的方法, 而 Knight[9]和 Sherif[10]提出以字母为单位, 不考虑发音过程, 直接进行翻译.

除上述方法外, 文献[11]使用音节相似度模型进行人名翻译; 文献[12]通过人工定义规则的方法进行翻译, 将英文字母划分为元音字母和辅音字母, 切分时则遵循元音字母和辅音字母配对的原则; 文献[13]将音节切分问题转换为序列标注问题, 把机器学习和统计机器翻译模型用于音译.

现有的研究音译效果还不够理想, 主要原因在于音节划分规则不够完善, 音节划分错误较多, 因此无法满足双语地图对注记的要求. 本文提出一种音节划分方法, 能够有效避免音节划分错误, 从而获得比较理想的音译结果; 设计了基于规则的多语种音译软件, 利用现有的规则表资源, 获得了大量高质量的音译成果.

1 音译流程设计

音译的第一步是选择合适的音译规则表 (不同的音译规则表对应不同的通用名词库、专有名词库和特殊规则表), 然后查询通用名词库和专有名词库, 如果输入的单词出现在这两个库中, 直接输出该单词对应的翻译结果, 否则进入单词音译模块. 由于规则表无法覆盖所有音译规则, 因此需要根据特殊规则表对输入的单词进行预处理, 然后经过单词拆分、拆分结果优化、单词重组和规则表查询等环节就可得到音译结果. 详细流程如图 1 所示.

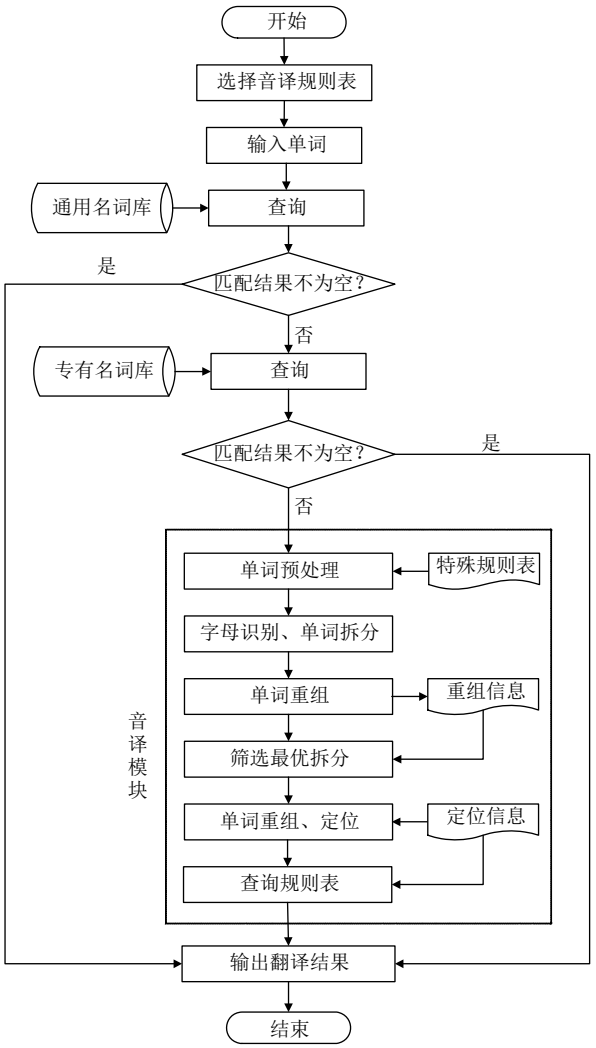

图 1 音译流程图

图 1 中通用名词库、专有名词库、规则表和特殊规则表应按软件要求的格式对数据进行组织. 就某个具体领域而言, 通用名词的数量是非常有限的, 如地图制图领域, 涉及到的通用名词如“山”、“河”、“村”、“湖”等. 专有名词库的数据源可以是相关语言的双语资源, 如人名资源、地名资源等, 也可以是经过审核认可的音译成果. 规则表和特殊规则表的设计应以权威

部门发布的成果为依据, 在使用过程中可适当进行修改和完善. 基于上述的设计, 利用软件能快速获得高质量的音译成果, 避免了人工查找规则费时费力、容易出错的可能性.

2 基于规则的音译算法

2.1 单词预处理

单词预处理是依据特殊规则表对源语言单词进行处理, 以便新单词利用规则表中的规则获得可信的音译结果. 通过对大量数据进行分析总结, 制定特殊规则表 (以罗马语-汉语为例) 格式如表 1 所示.

表 1 特殊规则表

类型	原字符串	新字符串	说明
词尾	kot	-果德	kot 在词尾翻译为果德
	h	null	h 在词尾不发音
	abad	-abad	人名后的 abad 单独翻译
	ua	uwa	ua 结尾时相当于 uwa
词首	a	ā	a 在词首按ā翻译
替换	mb	nb	m 在 b 前按 n 发音
	aun	au-n	au 后的 n 单独发音

表 1 中的例子包含罗马化的普什图语和印地语的部分特殊规则, 从中可以看出按替换位置划分, 特殊规则可以分为词首、词尾和替换三类, 分别表示用新字符串替换单词中词首、词尾和任意位置的原字符串, 其中“-”为单词分隔符, “null”表示空字符串. 单词预处理的过程比较简单, 输入的单词与特殊规则表逐行进行匹配, 如果相应的位置上出现原字符串, 则用新字符串替换并记录对应的说明信息, 用于在最终成果中输出.

2.2 单词拆分

单词拆分的第一步是字母识别, 即单词与所有元音字母和辅音字母进行匹配, 每个字母的匹配结果用四个参数表示: 字母、位置、长度、类型. 以罗马语单词“taisar”为例, 字母识别的结果为: t 0 1 f; a 1 1 y; i 2 1 y; ai 1 2 y; s 3 1 f; a 4 1 y; r 5 1 f, 其中每行的第一列表示单词中的字母, 第二列表示该字母在单词中的索引位置, 第三列表示字母的长度, 第四列表示字母的类型, “f”为辅音字母, “y”为元音字母. 根据字母识别结果对单词进行拆分, 就是从位置为 0 的字母开始, 搜索位置等于前一个字母位置和长度之和的字母, 从而得到一个连续的、不重叠的字母序列, 详细算法用伪码表示如算法 1.

算法 1. 单词拆分算法

```
1. Hashtable chash; //存储单词拆分结果的哈希表
2. Hashtable thash ← (“”,0); //存储过程数据的哈希表
3. begin while(End(thash,wlen)!=true) //判断是否结束循环
4. chash.clear(); //清空哈希表
5. begin foreach(object obj in thash) //遍历哈希表
6. int s ← obj.value; //搜索位置赋值
7. begin for(int r ← 0;r<rows;r++) //rows 为二维数组行数
8. begin if(arrs[r][1]=s)
9. begin if(arrs[r][3]=”f”)
10. chash.add(obj.key+arrs[r][0]+”|”,s+arrs[r][2]); //”|”为辅音字母识别码
11. end if
12. begin else
13. chash.add(obj.key+arrs[r][0]+”!”,s+arrs[r][2]); //”!”为元音字母识别码
14. end else
15. end if
16. end for
17. end foreach
18. thash ← chash;
19. end while
```

上述算法中, 函数 End(thash,wlen) 用于判断循环是否结束, 其中参数 wlen 表示待翻译单词的字符串长度, 循环结束的条件是哈希表 thash 中各项的 value 值均等于 wlen. 以单词“taisar”为例, 单词拆分的结果包含两项: t|a!i!s|a!r|和 t|ai!s|a!r|. 接下来对这两个拆分结果分别进行字母编组并筛选出最优拆分方案.

2.3 单词重组

单词重组的目的是实现字母编组, 方法是用字符“!”对单词拆分结果进行分割, 对不同类型的分割结果分别进行处理, 原则是尽可能多的出现“辅音字母+元音字母”组合, 尽可能少的出现单个元音字母或单个辅音字母组合, 详细算法用伪码表示如算法 2.

算法 2. 单词重组算法

```
1. string itext; //单词重组结果
2. int gps ← 0; //组合数
3. int fys ← 0; //辅音元音组合数
4. string[] strs ← split(ctext,”!”); //用字符“!”分割 ctext
5. begin foreach(string str in strs)
6. int c ← count(str); //计算 str 中字符”|”的个数
7. begin switch(c)
8. begin case 0:
9. itext ←itext+”|”+str+space; //space 表示空格
10. gps ← gps+1;
11. break;
```

```
12. end case
13. begin case 1:
14. itext ←itext+str+space;
15. gps ← gps+1;
16. begin if(!EndWith(str,"|")) //判断 str 是否以字符"|"结尾
17. fys ← fys+1;
18. end if
19. break;
20. end case
21. begin default:
22. begin if(EndWith(str,"|"))
23. string[] ss ← split(str,"|");
24. begin foreach(string s in ss)
25. itext ← itext+s+"|"+space;
26. gps ← gps+1;
27. end foreach
28. end if
29. begin else
30. string[] ss ← split(str,"|");
31. begin for(int i←0;i<length(ss)-2;i++) // length 返回数组中元素个数
32. itext ← itext+ss[i]+"|"+space;
33. gps ←gps+1;
34. end for
35. itext ← itext+ss[length(ss)-2]+"|"+ ss[length(ss)-1]+space;
36. gps ← gps+1;
37. fys ← fys+1;
38. end else
39. break;
40. end default
41. end switch
42. end foreach
```

上述算法中, 针对不同类型的拆分结果分别进行了处理, 结果包含重组字符串以及对应的组合数和辅音元音组合数两个参数, 用于筛选单词的最佳音节划分. 对字符串 t|a!i!s|a!r|和 t|ai!s|a!r|进行单词重组, 得到的重组字符串分别是"t|a |i s|a r|"和"t|ai s|a r|", 对应的组合数分别是 4 和 3, 辅音元音组合数均为 2.

排除极少数有歧义的情况, 对于多个拆分结果, 只有一个是正确的, 即最佳音节划分. 通过对大量音译成果进行分析发现, 最佳音节划分具备两个特点: a.组合数最小; b.辅音元音组合数最大, 这也符合自然语言的发音规则. 因此, 单词 taisar 的最优拆分方案是 t|ai!s|a!r|, 对应的重组结果即音节划分结果是"t|ai s|a r|". 采用这种策略不仅能获得正确的结果, 还能保证结果的唯一性.

2.4 规则表定位

根据重组结果确定音译结果, 其核心是确定每个组合中辅音字母和元音字母在规则表中的列号和行号, 从而定位目标语言项, 详细算法用伪码表示如算法 3.

算法 3. 规则表定位算法

```
1. int[][] parrs; //规则表定位结果
2. int pf,py; //辅音、元音字母索引位置
3. string[] strs ← split(itext,space); //用空格符分割 itext
4. begin foreach(string str in strs)
5. begin if(EndWith(str,"|")||StartWith(str,"|"))
6. begin if(EndWith(str,"|")
7. int len ← length(str);
8. string subf ← str.substring(0,len–1); //去掉 str 中最后一个字符"|"
9. pf ← pos(subf,flist); //在辅音字母列表 flist 中定位字母 subf
10. parrs ← (pf,1); //将定位信息追加到数组中
11. end if
12. begin else
13. string suby← str.substring(1); //去掉 str 中第一个字符"|"
14. py ← pos(suby,ylist); //在元音字母列表 ylist 中定位字母 suby
15. parrs ← (1,py);
16. end else
17. end if
18. begin else
19. int idx ← index(str,"|"); //返回字符"|"在 str 中的索引位置
20. string subf ← str.substring(0,idx);
21. string suby ← str.substring(idx+1);
22. pf ← pos(subf,flist);
23. py ← pos(suby,ylist);
24. parrs ← (pf,py);
25. end else
26. end foreach
```

上述算法中, 辅音字母列表和元音字母列表中前两位存储空字符串, 辅音字母或元音字母从第三位开始存储, 单独辅音字母组合或单独元音字母组合对应规则表的行号或列号为 1. 根据二维数组 parrs 中存储的值, 可以很方便地在规则表中查找到对应的音译结果. 如字符串"t|ai s|a r|"中各个组合对应的定位值分别为 (4,7)、(5,2) 和 (13,1), 对应的音译结果分别为"泰"、"萨"和"尔", 因此罗马语单词 taisar 的音译结果为"泰萨尔".

3 实验结果与评价

3.1 算法实现

本文提出的算法是在 Win7 操作系统上用 C#语言编程实现的, 软件使用到的通用名词库、专有名词库、特殊规则表和音译规则表均独立于软件本身, 使用者可根据实际情况单独进行编辑, 以满足不同语种音译的需要.

3.2 实验结果与分析

对英语地名单词、罗马化的普什图语地名单词、罗马化的印地语地名单词和俄语地名单词进行实验, 分别随机抽取 3000 个单词, 分为 5 组, 每组 600 个单词, 人工检查音译结果并统计正确率, 实验结果如表 2 所示.

表 2 正确率统计结果 (单位: %)

音译类别	英语-汉语	罗马语<普什图语>-汉语	罗马语<印地语>-汉语	俄语-汉语
组 1	93	93	92	92
组 2	93	92	93	93
组 3	92	92	93	92
组 4	92	91	92	93
组 5	93	92	92	91

从实验结果可以看出, 音译正确率可达 91% 以上, 通过对音译结果错误的单词进行分析发现, 导致错误的原因是: 1) 个别特殊规则未加入到特殊规则表中; 2) 未设置好特殊规则的优先级; 3) 少量单词的最佳音节划分结果不唯一. 在接下来的实验中, 本文将“连续两个辅音字母按一个辅音字母发音”这一规则加入到特殊规则表中并在单词预处理阶段优先使用, 实验结果如表 3 所示.

表 3 本文方法的正确率统计结果 (单位: %)

音译类别	英语-汉语	罗马语<普什图语>-汉语	罗马语<印地语>-汉语	俄语-汉语
组 1	96	96	95	97
组 2	96	95	96	95
组 3	97	96	96	96
组 4	95	96	97	96
组 5	96	97	95	95

从实验结果可以看出, 完善规则表可有效提高音译正确率. 现有的音译规则研究成果是音译的重要基础, 随着任务的不断拓展和深入, 需要不断完善和充实音译规则表, 确保音译成果正确可靠. 最佳音节划分结果不唯一的单词数量很少, 但需人工干预以确定正确的音译结果.

4 结论与展望

实验结果表明, 本文提出的算法能有效解决多语种音译问题, 在实际使用中只要设计好通用名词库、专有名词库、规则表和特殊规则表就能获得良好的音译效果. 由于本文提出的算法较好地解决了音节划分问题, 有效避免了音节划分错误, 因此进一步提高了音译正确率, 可以满足双语地图、机器翻译对双语资源的需要.

参考文献

1 王丹丹. 英汉人名音译的研究[硕士学位论文]. 大连: 大连理工大学, 2014.

2 于恒, 涂兆鹏, 刘群, 等. 基于多粒度的英汉人名音译. 中文信息学报, 2013, 27(4): 16–21. [doi: 10.3969/j.issn.1003-0077.2013.04.003]

3 李业刚, 黄河燕, 史树敏, 等. 多策略机器翻译研究综述. 中文信息学报, 2015, 29(2): 1–9. [doi: 10.3969/j.issn.1003-0077.2015.02.001]

4 童杉姗, 庞小平, 张璐璐. 双语地图中地图注记的设计. 地理空间信息, 2010, 8(2): 154–156. [doi: 10.3969/j.issn.1672-4623.2010.02.051]

5 Karimi S, Scholer F, Turpin A. Machine transliteration survey. ACM Computing Surveys(CSUR), 2011, 43(3): 17–46.

6 Knight K, Graehl J. Machine transliteration. Computational Linguistics, 1998, 24(4): 599–612.

7 Haizhou L, Min Z, Jian S. A joint source-channel model for machine transliteration. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. 2004. 159–166. [doi: 10.3115/1218955.1218976]

8 Oh JH, Choi KS. An ensemble of transliteration models for information retrieval. Information Processing & Management, 2006, 42(4): 980–1002.

9 Yaser AO, Kevin K. Translating named entities using monolingual and bilingual resources. Proceedings of the Association for Computational Linguistics. Association for Computational Linguistics. 2002. 400–408.

10 Tarek S, Grzegorz K. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. Proceedings of the Association for Computational Linguistics. Association for Computational Linguistics. 2007. 864–871.

11 Lin WH, Chen HH. Backward machine transliteration by learning phonetic similarity. Conference on Natural Language Learning. Association for Computational Linguistics. 2002. 1–7. [doi: 10.3115/1118853.1118870]

12 蒋龙, 周明, 简立峰. 利用音译和网络挖掘翻译命名实体. 中文信息学报, 2007, (1): 23–29. [doi: 10.3969/j.issn.1003-0077.2007.01.004]

13 邹波, 赵军. 英汉人名音译方法研究. 第四届全国学生计算语言学研讨会论文集. 2008. 336–343.