

外挂式英汉网络翻译浏览器的研究与实现

未末 郑杰 茅于杭

清华大学自动化系语言研究中心, 北京, 100084

weimo@mail.au.tsinghua.edu.cn

zhj@mail.au.tsinghua.edu.cn

myh-dau@tsinghua.edu.cn

摘要: 为克服现行各种网络翻译浏览软件的缺陷, 本文讨论了网络翻译系统设计的一些难点问题, 提出了一些解决 Web 页面获取与分析、分栏式站点处理以及图片处理等问题的方法, 并在这些方法的基础上实现了一个可保留寄主浏览器所有功能的外挂式网络翻译系统。

关键词: 机器翻译, 网络, 浏览器, 外挂式

The Design and Implementation of Attached English-Chinese Translation Browser

Wei Mo Zheng Jie Mao Yuhang

Speech and Language Processing Research Center, Tsinghua Univ., Beijing, 100084

weimo@mail.au.tsinghua.edu.cn

zhj@mail.au.tsinghua.edu.cn

myh-dau@tsinghua.edu.cn

Abstract: In order to solve some problems in many kinds of current Internet translation browsers, this paper discusses several questions about English-Chinese Internet translation browser. Based on some special techniques, several problems, such as getting and analyzing HTML source files, dealing with websites of multi-frame pages, displaying images after translation, have been solved.

Key words: machine translation, Internet, browser, attached

1. 引言

当前，由于 Internet 上的信息大部分是以英文的形式出现的，广大非英语人群迫切需要实用的网络翻译软件来克服语言障碍。为了实现这一目标，我们对网络翻译这一课题进行了较为深入的研究，设计并实现了一个通用的外挂式英汉网络翻译系统。

2. 对当前流行的网络翻译浏览器的特性的分析

当前市面上流行的网络翻译浏览器很多都存在一些缺陷，这在很大程度上是与它们的设计策略分不开的。一般说来，一个网络翻译浏览器的设计与实现不外乎以下几个主要的过程（参见图 2.1）：（1）获取 Web 页面内容；（2）通过版面分析确定需翻译的内容；（3）将需要翻译的内容送入翻译器翻译；（4）将译后内容按一定方式组合起来重新显示。但根据具体实现方式的不同，又可将它们主要分为以下几类：

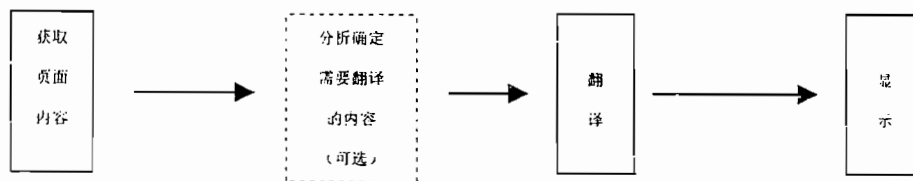


图 2.1 翻译浏览器的共同工作流程（虚线框表示有的翻译浏览器无需经过此过程）

2.1 自建独立的网络翻译浏览器

这种方法实质上是利用 ActiveX 等 Windows 技术，在某个包含有最基本的几项浏览功能的 ActiveX 控件（如 Visual C++ 中的 WebBrowser 控件）上实现自己独立的翻译浏览器。通过对此控件中的各种 Method 的调用，程序可以在内存中对将要显示的 HTML 页面源码进行分解、翻译，然后重新组合起来加以显示。显然，这种方法不是外挂式的。用这种方法实现的系统不必附着在别的浏览器上就能够获取完整的 HTML 信息，当然也能够做到准确的页面分析和完全的译后版面还原。但是这种系统也有较为明显的缺陷：不能充分利用已有的各种专用网络浏览器（如 Internet Explorer、Netscape 等等）的功能。众所周知，市面上的各种专用网络浏览器是 Microsoft、Netscape 等大公司为实现某种商业目标而专门设计的，具备相当完备的浏览功能以及各种辅助功能（如为数众多的菜单项、丰富的 option 设定功能以及强大的 Plug-in 扩充能力），可以给用户以愉悦得多的感受；而独立开发的网译系统则很难做到如此完善。如果仅仅为了浏览一下外文页面而舍弃掉这些功能去使用一个相对来说粗糙得多的浏览器的话，未免会让许多使用者心有不甘。

2.2 利用 DLL 来直接获取屏幕显示内容而后翻译

通过加载动态链接库 (DLL), 可以监控 Windows 本身的各种屏幕显示输出函数 (如 TextOutA, DrawTextA 等), 从而获取屏幕上的文本内容来进行翻译。这种方法是很多商业化的中文平台 (如 RichWin) 和在线字典 (如金山词霸) 所采用的基本方法, 可以方便地实现外挂功能; 同时, 由于直接从屏幕上取得的内容刚好就是需要翻译的内容, 这种方法可以跳过图 2.1 中虚线框所示的第二步。但是, 由于这种方法无法获取 Javascript、VBScript 等网页脚本代码和 Web 页面中的各种 HTML 标签, 译后显示时必然会导致原有网页风格的扭曲, 无法做到完全的版面还原。同时, 这种方法的稳定性也不尽如人意。

2.3 通过与浏览器通信来设计外挂式网络浏览器

这种方法的基本思路是通过浏览器自身的接口或菜单项来获取页面的 HTML 源文件, 接着分析此源文件确定出需要翻译的部分, 翻译后再重新加以显示。这种方法是外挂式的, 能够保留各种专用浏览器的功能; 同时它也能够获取完整的 HTML 标记信息, 因而能够完整地复现原页面的风格。然而, 当前许多采用此思路的网络翻译软件也有一些缺陷: 它们实质上是通过操纵浏览器的菜单项来获取页面的 HTML 源文件的, 这对于普通的 Web 页面来说确实有效; 但是, 对于近来越来越多的站点所采用的分栏式页面 (或称“多帧页面”, 比较典型的有三栏式的华南木棉站: <http://bbs.gznet.edu.cn/cgi-bin/group?g=4>), 这种获取方式则无能为力。不妨就以三栏式站点为例: 浏览者实际看到的内容均位于三个分栏区域内, 而用这种方法只能获取浏览器用以确定这三个分栏各自显示区域的另一个 HTML 文件 (在这里我们称之为“主源文件”或者“主文件”), 取不到对应各个分栏的 HTML 文件; 然而却正是这三个分栏文件包含着需要翻译的内容。因此, 这些软件是无法翻译分栏式页面的。

另外, 许多采用第一类或第二类方法的系统对 HTML 源码的分析不够完善, 导致译后页面中的图片和超链接很多时候无法正常显现。如果页面中含有较多 Script 代码的话, 这些系统的分析就可能发生紊乱, 严重时甚至会导致整个翻译过程的中断。

3. 本系统的策略和实现方法

为了解决这些问题, 本系统采取了如下的策略: 仍然采用第三类浏览器的一部分指导思想, 但是通过另外的方法来获取分栏式站点的各个分栏区域所对应的 HTML 源文件; 同时, 在更深层次上对 HTML 源码进行分析, 力求从根本上解决图片、链接以及 Script 代码所造成的各种问题。系统总的逻辑过程图如下所示:

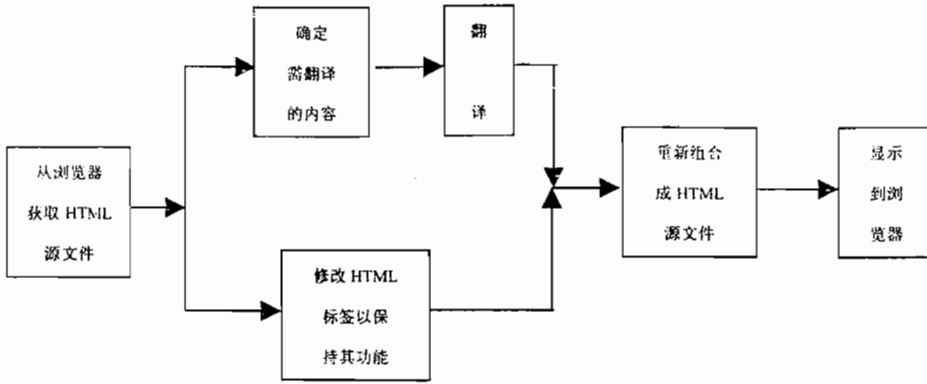


图 3.1 本系统的逻辑过程图

3.1 完整地获取页面的 HTML 源文件

准确而完整地获取页面的 HTML 源文件是整个网络翻译系统得以实现的先决条件。现在 Netscape 公司已经公布了其浏览器的源代码，人们可以通过修改其源代码的方法来另建一个浏览器，当然也就可获得所需的 HTML 源文件。然而 Microsoft 公司并未公布 Internet Explorer 的源代码，所以这种方法不具有通用性。因此，在本系统的设计中，我们决定利用 Windows 系统自身的特性，首先通过 C++ 语言中的 FindWindow() 等函数来获取浏览器窗口的句柄，然后分析 Windows 系统的 Z-order 分布状况以确定正在显示的页面是否是分栏式的（浏览器在显示分栏式页面时会在 Z-order 中表现得类似于多了几个子窗口）。接着，对应各个分栏（如果有的话）发送消息以操纵浏览器本身的菜单项，从而打开对应各分栏（如果有的话）的 HTML 源文件编辑窗口。最后，再通过向这些窗口发送 WM-GETTEXT 之类的消息来获取这些源文件的内容。由于近几年发布的各个 Windows 版本，包括 Windows95，Windows97，Windows98 以及 WindowsNT4 等均自带有缺省的纯文本文件编辑器，故这种方法的适用范围较广。

3.2 HTML 页面分析

相对于一般的文本翻译来说，网络翻译的一个难点就在于网络翻译所面对的文本是夹杂有大量 HTML 标签 (TAG) 的 HTML 源码，而不是能直接用来翻译的正文。分析 HTML 页面的主要目的是识别并抽取出需要翻译的正文部分以送入翻译器进行翻译，同时处理各种 HTML 标签以确保译文经版面恢复后再显示时能够复现原页面的风格。

可以确认，所有的 HTML 标签都如同最典型的标签 <HTML> 一样是由两个字符 “<” 和 “>” 括起来的，因此程序可以通过判定 “<” 和 “>” 来确定标签部分和非标签部分的起始与结束。随后，通过分析就可以确定出非标签部分是否需要翻译的部分（因为有时也可能是 Script 代码），接着就可将需翻译的部分送入翻译器翻译，而标签部分和不应翻译的非标签部分则原则上应当予以保留。

然而，现代 HTML 语言中的有些标签仅靠原封不动地硬插回译后文件中并不能完成其应有功能。因此，还须进一步分析各种标签的内部属性。对这些标签的特殊处理大致可分为以下几类：

(1) 处理控制分栏的标签

这类标签的典型例子是 `< frame name="list" src="/tmplist.htm" scrolling="auto" marginHeight="3">`。这类标签一般只存在于分栏式页面的主文件中，它可以使一个浏览器窗口显示出多个子窗口，每个子窗口均可以独立显示并与用户交互。这极大地丰富了页面，然而不可否认的是它也给网络翻译系统获取源代码造成了不容忽视的困难。实际上，浏览器在显示一个 n ($n=1$ 时表示普通的非分栏式页面) 栏式页面时会自动地在机器硬盘上的某个缓冲区里创建 1 个 ($n=1$ 时) 或 $n+1$ 个 ($n>1$ 时) 对应的 HTML 源文件，而且当 $n>1$ 时，除一个是统管几个分栏的主文件外，其余 n 个文件各自对应 n 个分栏。在前面取得各分栏内容的同时，程序可以得到这些分栏各自对应的缓冲文件的路径 (path)，通过以此 path 来替换属性 src 的值，使之成为类似于 `src="c:\pwin95\temp\tmplist.htm"` 的形式，即可在显示译文时使之也呈现出与原文一样的分栏式效果。当然还应对应好各个分栏的位置，以免显示时出现分栏之间的相对位置与原页面中相反的情况。这是因为在 Windows 中，各分栏子窗口的显示顺序与它们在 Z-order 中的排列顺序刚好相反。

有时分栏与分栏之间存在多层嵌套，如下所示：

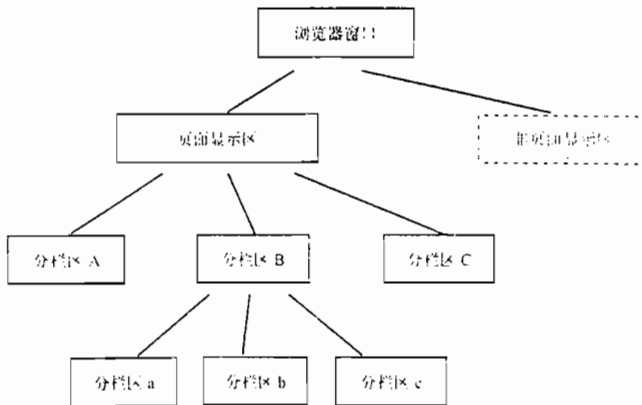


图 3.2 分栏式页面的典型结构图

这样一来形成了一个树形图。但是，树形图的每个分枝处仍然类似于上面所说的这种结构，因而在每个节点处仍可以用同样的方法处理。

(2) 处理与 Script 代码相关的标签

这类标签以 `<script language="JavaScript">` 为典型代表。这类标签与 `</script>` 之间夹的即为 Script 代码，如果将它们也不加区分地送至翻译器翻译的话，则很可能会使代码受到破坏 (例如把其中的某些编程语句给翻译了)；同时，这也不必要地增加了机器的负担。因此，应该在程序中判别出 Script 代码，使其免去翻译处理这一环节。

(3) 处理超链接

`` 是这类标签的典型代表。这类标签的功能是使浏览者能通过鼠标点击其作用的文本或图片来跳转至 href 属性所指示的位置。但是，由于 href 属性的值既可以如上所示以绝对寻址方式来表示，也可以以相对寻址方式来表示，如 ``。采用相对寻址方式的标签如果仅仅被硬插回到译文文件中将不能起到超链接作用。因此，应该把该服务器的 URL 经过处理后加到这些相对链接之前，使之成为绝对链接。

(4) 处理图片、音乐等多媒体标签

此类标签以 `` 为典型代表，其功能是显现图片、音乐等多媒体信息。同样，如果其 src 属性采用了相对寻址方式的话，程序可以用与 (3) 相似的办法来使其功能得以保持。

3.3 文本翻译

文本翻译是本翻译系统的中心部分，它在很大程度上决定着系统的性能。我们认为，由乔姆斯基最先提出的短语结构语法实质上是一种单值标记函数理论，亦即对于句子结构树形图中的每一个节点，这种理论只用一个相应的标记来描述其特征。从实践来看，这种理论的描述能力偏弱，而生成能力过强，容易产生大量的歧义结构。因此，本系统采用了这种理论的一种改进模型——“多又多标记树模型”（Multiple-branched and Multiple-labeled Tree Model, 简称 MMT 模型）——来作为我们语法分析的基础。MMT 模型用多值标记函数（Multiple-labeled Function）来描述树形图中的节点，这样，对应树形图中的一个节点就有多个标记来描述其特征。这显然能提高描述能力，因而能减低歧义结构出现的可能性。

整个翻译器的整体逻辑图如下：

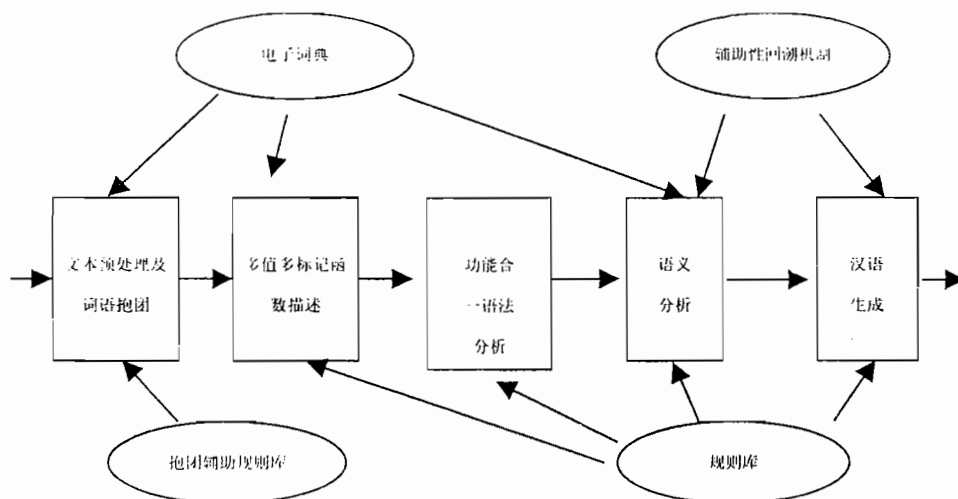


图 3.3 翻译器的整体逻辑图

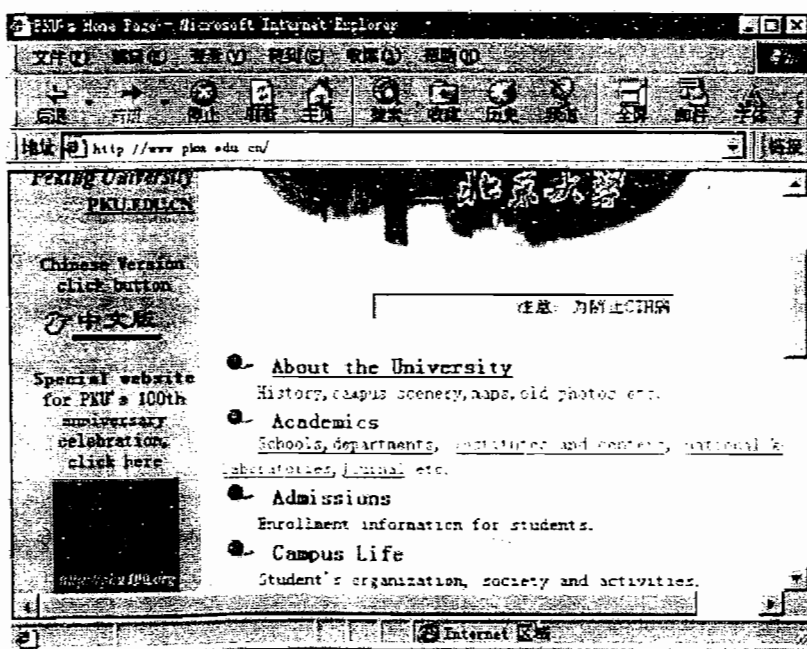
词语抱团的正确率对随后的语法、语义分析有重大影响。可以说，无论何种翻译系统，词语抱团的高正确率都是翻译成功的先决条件。因此，本系统特地建立了一个专用的抱团规则库来加强抱团处理。

另外，从图中可以看出，本系统采用了“多值标记函数——功能合一语法”这一思路来进行分析，这使得需要另外加以辨别的歧义结构出现的概率明显减少。同时，系统在语义分析时采用了格关系合一算法来进一步解决歧义。实际结果也表明，这样获得的译文质量相对于单纯用短语结构语法所获得的译文质量来说有所提高。

3.4 译文的显示

显示译文是整个系统工作流程的最后一个环节，一般情况下只需将译文文件的 path 送入浏览器的地址框，然后向浏览器发送确认消息（如 WM-KEYDOWN）即可。对于分栏式页面，由于在“HTML 页面分析”阶段中已经处理了“译后页面中各分栏区域之间的相对排列顺序可能发生颠倒”这一问题，所以仍只须将其最上层的主文件的 path 送入地址框，加以确认后浏览器即可正确地复现出整个页面的各个分栏。

经本系统翻译后的一个网页如下所示（对应的英文页面是北京大学的英文版主页：<http://www.pku.edu.cn/>）：



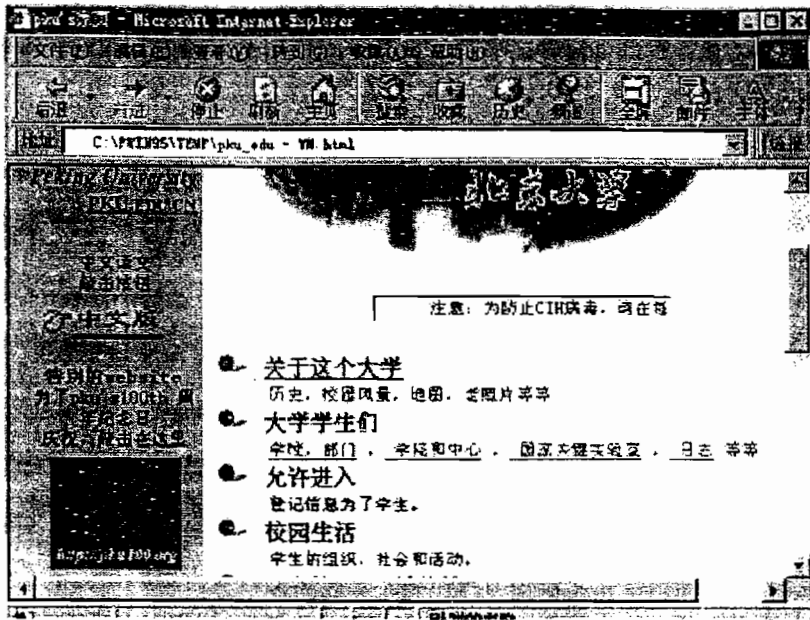


图 3.4 本系统的一个网络翻译的例子

4. 结束语

本外挂式英汉网络翻译浏览器可外挂于现有的多种浏览器如 IE4.0 和 IE5.0 上, 在保留浏览器功能的同时能够使翻译后的页面完整地复现原文页面的版面风格和各种特性。另外, 系统的译前处理过程与翻译的语种完全无关, 可方便地加载在其它翻译器之前。然而, 我们看到, 当今社会上浏览器和网页设计方法都在不停地更新换代和推陈出新, 同时, 人们对翻译质量的要求也在不断地提高。这一切都要求我们不断地改进和完善这一系统, 以更好地适应使用者的要求。

参考文献

- [1]. 自然语言机器翻译新论, 冯志伟, 语文出版社
- [2]. 语言工程, 陈立为, 袁琦, 清华大学出版社, 1997
- [3]. 语言信息处理专论, 黄昌宁, 夏莹, 清华大学出版社, 广西科学技术出版社, 1996
- [4]. 中文信息处理技术及基础, 上海交通大学出版社, 1990 年, 191-193
- [5]. Brian Farrar. Special Edition Using ActiveX .Que Corporation. 1996
- [6]. 中文信息 (BYTE Chinese) 《中文信息》杂志社, 1998.3. 第 15 卷, 第 2/3 期