基于“MMa爬虫”实现的word文档自动粤语翻译&术语注释系统

黄政菘　中山大学计算机学院　**21307149**

一、引言

在广东生活也有一年多了，我却还不太懂说几句粤语；步入大二中期，科研训练也逐渐提上了日程，我却还苦恼于看不懂学术论文中那些生僻的英文专业术语……诶，我可是计算机专业的学生，这点小问题可难不倒我！既然Mathematica这么强大，要不就拿它来试一试？

有注意到Mathematica虽然功能十分强大，但却没有将文本进行粤语翻译的功能（内置的TextTranslation文本翻译语句是基于国外Google翻译和Microsoft翻译实现的，因而语言范围有限），也没有释义（将专业术语都有什么意思逐一列举）的功能，又想到要实现这些功能就需要外部接口，这和网络爬虫爬取数据的操作有异曲同工之处，于是我决定亲手搓一个，并用其实现word文档的自动翻译或注释工作。

由于本人是计算机科学与技术专业的学生，以前也鼓捣过一些自动化的小程序，因此这次选择了做这么一个独辟蹊径的项目，运用了大量编程语句，结构框架也和C++面向对象编程的结构相似，还希望老师能多多包涵。

另外，这个项目也符合“软件工程”的思想，面向客户（其实就是我自己）的需求，并且实现了模块化和前后端分离。

二、项目实现

1. 初始化

```
QuantityTimes[mult_?NumericQ, q_Quantity] := Quantity[mult q[[1]], q[[2]]];
                                数值量判定  数量          数量
(*定义倍乘函数，方便后续计算*)

SetDirectory[NotebookDirectory[]];
设置目录        当前笔记本的目录
(*将保存目录设置为当前文件夹*)

If[PacletFind["WebTools"] === {},
…  查找小数据包
  PacletInstall[NotebookDirectory[] <> "WebTools-0.1.1.paclet"]];
   安装数据包        当前笔记本的目录
(*第一次运行需先安装WebTools小数据包*)
```

2. 翻译

(1) 启动网页会话

```
InitBaiduTranslate[from_ : "en", to_ : "yue"] := (
  $Session = StartWebSession[Visible → False];
             启动网页会话          可见否      假
  (*在后台打开一个ChromeDriver程序，模拟正常的浏览器运作，内存占用很小，无需担心
    若将Visible设置为True可以看到整个翻译流程的具体运作过程*)
  Pause@1;
  暂停
  (*设置1秒的延时，等待ChromeDriver初始化加载*)
  WebExecute["OpenPage" → ("https://fanyi.baidu.com/#" <> from <> "/" <> to)]
  网页执行
  (*加载完成就可以打开百度翻译的网页“https://fanyi.baidu.com”了，
   所加的后缀即为原语言和目标语言的英文简写*)
 )
```

(2) 获取翻译结果

```
GetBaiduTranslateResult[] := Quiet@StringDelete[
                               不…    删除匹配的字符
   Check[
   校验
    WebExecute["ElementText" → ("CSSSelector" →
    网页执行
          "[Class=\"ordinary-output target-output clearfix\"]")][[1]] <> "", ""
   ], "\n"
  ]
(*网页模拟爬取翻译结果*)
```

```
ClearTranslateContent[] :=
  WebExecute["ClickElement" → ("CSSSelector" → "[title=\"清空\"]")];
  网页执行
(*清空前一次翻译的内容，方便进行后一次的输入*)
```

(3) 百度翻译

```
BaiduTranslateRaw[text_] := Block[{s},
                            块
  If[StringMatchQ[text, WhitespaceCharacter ...], Return[""]];
  … 字符串匹配判定        单个空白字符          返回
  (*若该文本为一个及以上的空白字符，则提前结束返回*)
  ClearTranslateContent[];
  WebExecute[
  网页执行
   "TypeElement" → {"CSSSelector" → "[id=\"baidu_translate_input\"]", text}];
  (*模拟选择输入框，并所要翻译将文本输入进去*)
  Pause@.5;
  暂停
  (*暂停半秒等待百度翻译服务端响应，输出翻译结果*)
  While[GetBaiduTranslateResult[] === "", Pause@.1];
  While循环                              暂停
  Pause@.5;
  暂停
  GetBaiduTranslateResult[]
 ]
```

```
BaiduTranslateRaw[text_] /; StringLength[text] > 4500 :=
                            字符串长度
 StringJoin[BaiduTranslateRaw /@ StringPartition[text, UpTo[4000]]]
 连接字符串                     按长度分割字符串          多达
(*若句子长度大于4500则对其进行划分，再输入进网页进行翻译，防止句子过长放不进去*)
```

In[754]:=

```
Module[{memory = <||>, MaxCacheMemory = 1.*^8, tmp},
模块
 ClearMemory[] := (memory = <||>); (*清除句子缓存*)

 BaiduTranslate[text1_] :=
  Quiet@With[{text = ToString[text1, CharacterEncoding → "Unicode"]},
  不…   With循环   转换为字符串   字符编码
    If[KeyExistsQ[memory, text], AppendTo[memory, text → memory[text]];
    …  键存在判定                附加
     memory[text], (*若之前已经翻译过，则直接取，加速翻译过程*)
     If[StringLength@text ≥ 60,
     …  字符串长度
      BaiduTranslateRaw@text,    (*若所要翻译的文本长度不少于60，则直接翻译*)
      tmp = BaiduTranslateRaw[text];
      (*否则可能会是单词/短语，顺便将其放入缓存可加速翻译*)
      AppendTo[memory, text → tmp];
      附加
      While[ByteCount@memory > MaxCacheMemory, memory = Rest@memory];
      Whil…  字节数                                  去掉第一个
      (*及时清理句子缓存，确保缓存在设定的范围内，避免给计算机带来负担*)
      tmp (*返回翻译好的句子*)
     ]
    ]
   ]
]
```

3. 术语注释

In[755]:=

```
memoryt = {};
Store = {};
letters = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l",
   "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"};
(*首字母列表以及存储空间清空初始化*)
```

(1) 建立术语中英对照表

In[758]:=

```
CreateTermsTable[num_ : 26] := Block[{len, i, tmp},
                               块
  letters = Take[letters, num]; (*截取首字母表前num个字母*)
            选取
  InitBaiduTranslate["en", "zh"]; (*将翻译模式设置成英文翻译成中文*)
  AppendTo[memoryt, StringCases[URLRead[
  附加              字符串匹配     读取URL响应
      "https://www.tldevtech.com/computer-terms-that-start-with-the-letter-" <> #,
      "Body", FollowRedirects → True], "<div class=\"wp-block-columns my20\">" ~~
              遵循重定向                真
```

```
       Shortest[a__] ~~ "<b>" ~~ Shortest[c__] ~~ "</b>" :> c]] & /@ letters;
       最短                                  最短
memoryt = Catenate[memoryt];
          序连
(*爬虫，根据首字母a-z依次爬取国外英文网站tldevtech的计算机科学术语表，
 并存入memory列表中（数量比较少，共388个，相当于只是一个demo）*)

len = Length[memoryt];
      长度
Store = Table["", {i, len}, {j, 2}];
        表格
(*制表，方便后续存储和导出成xls*)

Monitor[
监控
  (ts = Now;
        此刻
   For[i = 1, i ≤ len, i++,
   For循环
    Store[[i, 1]] = memoryt[[i]];
    tmp = StringCases[URLRead["https://dict.youdao.com/search?q=" <>
          字符串匹配      读取URL响应
        StringReplace[memoryt[[i]], " " → "%20"], "Body", FollowRedirects → True],
        替换字符串                                    遵循重定向              真
      "<div class=\"trans-container\">" ~~ Shortest[a__] ~~
                                           最短
        "<li>" ~~ Shortest[c__] ~~ "</ul>" :> c];
                  最短
    (*预处理：爬虫爬取相应单词/短语在有道词典里的页面内容，初步提取相关释义*)
    If[tmp === {}, Store[[i, 2]] = BaiduTranslate[memoryt[[i]]],
    如果
     Store[[i, 2]] = StringReplace[
                     替换字符串
       StringDelete[StringDelete[StringDelete[
       删除匹配的字符   删除匹配的字符   删除匹配的字符
          tmp[[1]],
          "</li>"], " "], "\n"], "<li>" → " "];
     If[StringContainsQ[tmp[[1]], "current"],
     … 字符串包含判定
      Store[[i, 2]] = BaiduTranslate[memoryt[[i]]], If[StringContainsQ[
                                                    … 字符串包含判定
        tmp[[1]], "abstract"], Store[[i, 2]] = BaiduTranslate[memoryt[[i]]]];
    ];
    (*若在有道词典里找不到对应的词条，
     什么也爬取不到（爬取内容列表为空），则直接将原单词/短语进行百度翻译替代；
     否则正常进行，去除网页分隔符，提取该词条，
     若提取到了奇怪的东西（其中current或abstract标识是他们的共性），
     则直接将原单词/短语进行百度翻译替代*)
    If[StringContainsQ[Store[[i, 2]], ">" ~~ __ ~~ "<"],
    … 字符串包含判定
     Store[[i, 2]] = StringDelete[StringDelete[StringCases[
                     删除匹配的字符   删除匹配的字符   字符串匹配
```

```
            URLRead["https://dict.youdao.com/search?q=" <> StringReplace[
            读取URL响应                                  替换字符串
                memoryt[[i]], " " → "%20"], "Body", FollowRedirects → True],
                                                    遵循重定向          真
            RegularExpression["<b>(.*?)</b>"]], "<b>"], "</b>"][[1]];];
            正则表达式
        (*若在有道词典里找不到对应的词条，但爬取到了相应的网络释义，则提取第一个网络释义*)
        If[StringMatchQ[Store[[i, 2]], RegularExpression["[a-zA-Z0-9\\s]*"]],
        … 字符串匹配判定                  正则表达式
          Store[[i, 2]] = BaiduTranslate[memoryt[[i]]];(*若在有道词典里找不到对应的词条，
          但爬取到了奇怪的英文，则直接将原单词/短语进行百度翻译替代*)
        ];),
      Refresh[ (*进度显示模块*)
      刷新
        Framed[(*设置边框*)
        加边框
          Column[{
          列
            "英文专业术语已爬取成功，正在逐一爬取对应的释义词条...请耐心等候",
            Row[{"进度: ", i, "/", len}],
            行
            Row[{"当前已耗时: ",
            行
              UnitConvert[Now - ts, MixedUnit[{"Hours", "Minutes", "Seconds"}]]}],
              单位转换          此刻          混合单位
            Row[{"估计剩余时间: ", UnitConvert[QuantityTimes[(len - i) / i, Now - ts],
            行                    单位转换                                  此刻
              MixedUnit[{"Hours", "Minutes", "Seconds"}]]}],
              混合单位
            Row[
            行
             {"预计完成时间: ", TimeObject[Now + QuantityTimes[(len - i) / i, Now - ts]]}],
                                 时间对象          此刻                                  此刻
            ProgressIndicator[i / len](*添加进度条*)
            进度指示器
          }]
        , RoundingRadius → 8, FrameStyle → Blue]
          圆角的圆半径          边框样式          蓝色
      , UpdateInterval → 1, TrackedSymbols :→ {}]] ×
        更新间隔                被跟踪的符号
    Export["TermsTable.xls", Store]]
    导出
(*导出成xls文件*)
```

(2) 提取注释

```
ExtractComments[text2_] := Block[{temp = "注释: ", l, ll, cache, i, j, k, cc = 1},
                            块
  cache = StringSplit[text2]; (*将英文句子划分为一个个单词*)
          按模式匹配分割字符串
  If[Store === {},
  如果
   memoryt = {};
   Store = Import["TermsTable.xls"][[1]];
           导入
   (*若之前已经建立好对照表，则直接导入使用*)
   n = Length@Store;
       长度
   For[k = 1, k ≤ n, k++, AppendTo[memoryt, Store[[k, 1]];];];
   For循环                  附加



  l = Length@cache;
      长度
  ll = Length@memoryt;
       长度

  For[i = 1, i ≤ l, i++, For[j = 1, j ≤ ll, j++,
  For循环                  For循环
    If[ToLowerCase[cache[[i]]] === ToLowerCase[Store[[j, 1]]],
    … 转换为小写                 转换为小写
     (*两层循环遍历，若匹配到专业术语，则添加注释*)
     temp =
      temp <> "[" <> ToString[cc] <> "]. " <> Store[[j, 1]] <> " : " <> Store[[j, 2]] <> "  ";
                    转换为字符串
     cc = cc + 1]
   ]
  ];
  If[cc === 1, temp = ""];    (*若没匹配到，则返回空*)
  如果
  temp]
```

4. 解析文件

(1) 解压与压缩

In[760]:=

```
Module[{$TempDir = "temp_" <> ToString[RandomInteger[10^20 - 1]]},
 ZipDir[dir_] := DirectoryName[dir] <> $TempDir;
]
(*确定缓存文件夹，为后续对Word文档的加工处理做准备*)
```

In[761]:=

```
UnZipDocx[dir_] := (ExtractArchive[dir, ZipDir[dir]]; ZipDir[dir]);
(*预处理：解压Word文档，Word文档可以被看作是一个压缩包，
 其内容主要是用XML写成的，将其解压提取XML是对其进行处理的第一步*)
ZipDocx[dir_, newdir_] :=
  With[{$TempFile = FileNameJoin[Append[Most@FileNameSplit[newdir],
        "temp_" <> ToString[RandomInteger[10^20 - 1]] <> ".zip"]]},
   CreateArchive[FileNames[ZipDir[dir] <> "\\*"], $TempFile];
   RenameFile[$TempFile, newdir, OverwriteTarget → True];
   newdir
  ];
(*打包压缩Word文档缓存文件，生成新的Word文档*)

ZipCleanup[dir_] := DeleteDirectory[ZipDir[dir], DeleteContents → True];
(*清空处理完残留的缓存*)
```

(2) XML操作

In[764]:=

```
OperateXML[xml_, operation_, crit_, displaytext_] :=
 ReplacePart[xml,
  Block[{ts = Now, tot, count = 0, (*初始化局部变量*)
    cont = GatherBy[With[
        {p = Position[xml, XMLElement[{_, "t"}, {___}, {_}], Infinity]},
        (*在相应的位置上提取句子*)
        Thread[{p, Extract[xml, p][[ ;; , 3, 1]]}]
```

```
    ], #[[1, ;; crit]] &](*cont为提取的将要翻译或注释的句子列表*)
  }, tot = Length@cont;
          长度

  Monitor[
  监控

   (count++;
      (#[[-1, 1]] ~Join~ {3, 1}) → ToString[StringJoin[#[[-1, 2]], "    ",
                  连接              转换为…   连接字符串
         operation@StringJoin@#[[ ;; , 2]]], CharacterEncoding → "Unicode"]
                    连接字符串                  字符编码
    ) & /@ cont,
   (*每个句子被映射为替代#符号,
    对每个句子分别进行翻译或注释的操作，处理完后以4个空格为间隔连接到原句的末尾*)

   Refresh[ (*进度显示模块*)
   刷新
    Framed[(*设置边框*)
    加边框
     Column[{
     列
       displaytext,
       Row[{"进度: ", count, "/", tot}],
       行
       Row[{"当前已耗时: ",
       行
         UnitConvert[Now - ts, MixedUnit[{"Hours", "Minutes", "Seconds"}]]}],
         单位转换      此刻      混合单位
       Row[{"估计剩余时间: ", UnitConvert[QuantityTimes[(tot - count) / count,
       行                    单位转换
           Now - ts], MixedUnit[{"Hours", "Minutes", "Seconds"}]]}],
           此刻        混合单位
       Row[{"预计完成时间: ",
       行
         TimeObject[Now + QuantityTimes[(tot - count) / count, Now - ts]]}],
         时间对象      此刻                                    此刻
       ProgressIndicator[count / tot](*添加进度条*)
       进度指示器
      }]
     , RoundingRadius → 8, FrameStyle → Blue]
       圆角的圆半径          边框样式       蓝色
    , UpdateInterval → 1, TrackedSymbols :→ {}]
      更新间隔              被跟踪的符号
  ]
 ]
]
```

```
OperateNotes[notes_, operation_ : BaiduTranslate] :=
 OperateXML[notes, operation, 3, "正在为您翻译/注释脚注和尾注...马上就要完成啦！"]
OperateDocs[docs_, operation_ : BaiduTranslate] :=
 OperateXML[docs, operation, -5, "正在快马加鞭地为您翻译/注释正文..."]
(*分别为Word正文和注释定义接口，其中缺省操作为进行百度翻译*)
```

5. 执行模块

```
Options[OperateAll] = {"TranslateFrom" → "en", "TranslateTo" → "yue"};
选项
(*可以自行选择要翻译的语言，以及要翻译的目标语言，这里分别为英语和粤语*)

OperateAll[filein_, fileout_, operation_ : BaiduTranslate, OptionsPattern[]] :=
                                                                选项模式
 CheckAbort[Block[{doctemp = UnZipDocx[filein], $ZipDir = ZipDir[filein] <> "\\"},
检测中止      块
   (*Initialize*)
     预置
   If[operation === BaiduTranslate,
   如果
    InitBaiduTranslate[OptionValue["TranslateFrom"], OptionValue["TranslateTo"]]];
                          选项值                          选项值


   Pause@1;
   暂停
   If[FindFile[$ZipDir <> "word\\document.xml"] =!= $Failed,
   … 找文件                                      失败
    (*先判断是否有正文，避免报错，下同*)
    Export[$ZipDir <> "word\\document.xml",
    导出
     OperateDocs[Import[$ZipDir <> "word\\document.xml", "XML"], operation],
                 导入
     OverwriteTarget → True]; (*翻译/注释正文*)
     覆盖目标              真


   If[FindFile[$ZipDir <> "word\\footnotes.xml"] =!= $Failed,
   … 找文件                                        失败
    Export[$ZipDir <> "word\\footnotes.xml",
    导出
     OperateNotes[Import[$ZipDir <> "word\\footnotes.xml", "XML"], operation],
                  导入
     OverwriteTarget → True]]; (*翻译/注释脚注*)
     覆盖目标              真


   If[FindFile[$ZipDir <> "word\\endnotes.xml"] =!= $Failed,
   … 找文件                                      失败
    Export[$ZipDir <> "word\\endnotes.xml",
    导出
```

```
    OperateNotes[Import[$ZipDir <> "word\\endnotes.xml", "XML"], operation],
                 导入
    OverwriteTarget → True]]; (*翻译/注释尾注*)
    覆盖目标              真

  ZipDocx[filein, fileout];
  ZipCleanup[filein];

  If[operation === BaiduTranslate,
  如果
   Framed["提示：翻译后的Word文档已生成，路径👉 " <> fileout,
   加边框
    RoundingRadius → 5, FrameStyle → Green],
    圆角的圆半径          边框样式        绿色
   Framed["提示：注释后的Word文档已生成，路径👉 " <> fileout,
   加边框
    RoundingRadius → 5, FrameStyle → Green]]
    圆角的圆半径          边框样式        绿色
 ], ZipCleanup[filein]]
```

三、客户端

使用说明:

1. 确保已经安装了Chrome浏览器或Firefox浏览器（MMa的网页浏览器自动化只支持这两款；推荐Chrome(点击进入官网下载)，安装极速且占用小）

2. 首先点击菜单栏最左边的红色图标旁的菜单，选择“初始化单元”（每次运行前最好都初始化一下，避免出现奇怪的错误）

3. 接着运行以下其中一个区块的代码，选择需要处理的文档，以及处理好的文档的保存路径

4. 第一次运行可能会稍久，请耐心等待

（1）粤语翻译系统

请选择示例文件：TranslateTest1.docx、TranslateTest2.docx、TranslateTest3.docx

In[7]:=

```
Block[{infile = SystemDialogInput["FileOpen",
块                 系统对话输入
    {NotebookDirectory[], {"Word Files" → {"*.docx"}, "All files" → {"*"}}},
     当前笔记本的目录              单词                          全部
    "WindowTitle" → "Browse Target .docx File"], outfile},
     视窗标题                                  文件位置的符号表示
 (*选择将要翻译的文件*)
 outfile = SystemDialogInput["FileSave", StringReplace[infile,
           系统对话输入                    替换字符串
    StartOfString ~~ Longest[dir___] ~~ "\\" ~~ Shortest[file___] ~~ EndOfString :→
    字符串的开始处       最长的                          最短                  字符串末端
     dir <> "\\" <> If[StringFreeQ[file, "."], file <> "_out.docx", StringReplace[file,
                  … 无字符串匹配判定                                  替换字符串
        StartOfString ~~ Longest[pre___] ~~ "." ~~ Shortest[ap___] ~~ EndOfString :→
        字符串的开始处       最长的                          最短                  字符串末端
         pre <> "_out." <> ap]], "WindowTitle" → "Save Created .docx File"];
                                  视窗标题          保存                  文件位置的符号表示
 (*选择翻译后文件的保存路径，也可以更改文件名，
  默认保存在当前所在文件夹，文件名为在原文件后添加“_out”后缀*)
 If[infile =!= $Canceled && outfile =!= $Canceled,
 如果          被取消                  被取消
  OperateAll[infile, outfile, BaiduTranslate]]
 (*若选择成功则开始翻译*)
]
DeleteObject[$Session]
删除对象
(*关闭ChromeDriver，程序结束*)
```

（2）术语注释系统

请选择示例文件：CommentTest.docx

```
SetDirectory[NotebookDirectory[]]; Flag = 0;
Block[{infile = SystemDialogInput["FileOpen",
    {NotebookDirectory[], {"Word Files" → {"*.docx"}, "All files" → {"*"}}},
    "WindowTitle" → "Browse Target .docx File"], outfile},
 (*选择将要注释的文件*)
 outfile = SystemDialogInput["FileSave", StringReplace[infile,
    StartOfString ~~ Longest[dir___] ~~ "\\" ~~ Shortest[file___] ~~ EndOfString :>
     dir <> "\\" <> If[StringFreeQ[file, "."], file <> "_out.docx", StringReplace[file,
        StartOfString ~~ Longest[pre___] ~~ "." ~~ Shortest[ap___] ~~ EndOfString :>
         pre <> "_comment." <> ap]]], "WindowTitle" → "Save Created .docx File"];
 (*选择注释后文件的保存路径，也可以更改文件名，
  默认保存在当前所在文件夹，文件名为在原文件后添加“_comment”后缀*)
 If[infile =!= $Canceled && outfile =!= $Canceled,
  If[FindFile["TermsTable.xls"] === $Failed,
   CreateTermsTable[]; Flag = 1;];
  (*第一次执行需爬虫爬取数据，并在当前目录生成计算机科学术语中英对照表，
   接受一个1-26的整型参数，表示生成首字母a-z前多少个的术语，默认为26
    大约耗时7min，快慢主要取决于网速*)
  OperateAll[infile, outfile, ExtractComments]]
 (*若选择成功则开始注释*)
]
If[Flag === 1, DeleteObject[$Session]]
```

四、总结

1. 存在的问题

（1）没为爬取英文专业术语模块（约耗时1min）添加进度条，因为是基于首字母映射进行的爬取，难以抓取一个衡量标准

（2）术语注释的部分，若碰到短语（中间有空格分开）或名词的其他形式（如复数）就无法识别，改进的方法是有的，但那样就太耗费时间和精力了

（3）没实现Compile编译 / Parallel并行编程，部分过程有些耗时（如建立术语中英对照表），若能实现编译或并行也许能大大加快运行速度，但时间不太允许了（还有其他作业要赶），只好将就一下，完成率能有90%就满足了

2. 结语

这个项目总共耗时将近5天，因为我原本就有一点爬虫的基础，所以Mathematica的爬虫只是花了大概半天时间来适应，耗时主要是在复习基础知识以及爬虫试错上。学习方面我先是边写代码边学习了老师在课上没有讲的各种文件操作（包括解压、创建文件夹等）和编程式写法（Block/With/Module……），还学习了一些进阶的字符串操作（如匹配、计算等），途中还掌握了一些语句的简洁写法（如“@”、“<>”和“&/@”等），这让我越写越顺手；爬虫的部分我先是用我熟悉的Python爬一下，看看能否成功，若成功了再将其迁移至Mathematica中，但在一堆Html语句中去提取想要部分的过程真的很痛苦……

这个项目的idea也主要是围绕着老师的鼓励方向“爬虫”进行的。

翻译网站都是动态的，当输入要翻译的句子进去才会返回结果，而不是静静地摆在那里，这给爬虫带来了巨大的困难。因此粤语翻译的部分本来是打算用URLRead、HTTPRequest等传统“爬虫”语句来实现的，但奈何它们的能力实在有限，实现“动态爬虫”十分困难，而且Mathematica内置的浏览器内核太低级，访问不了百度翻译（只有它有粤语翻译），于是只好用万能的浏览器自动化功能来实现“伪爬虫”。

而幸运的是，我偶然间发现有道词典是静态实现的，只要在地址栏中给出参数即可爬取相对应的结果，因此想到可以基于此实现文本注释的功能，于是为了弥补粤语翻译“伪爬虫”的难堪，术语注释系统便实现了“双重真爬虫”——先是爬英文专业术语，再去爬对应的词条释义。

建模的部分，我是基于本专业计算机科学与技术所学的知识，用标志性的文件操作来进行，将翻译或注释好的文本写入Word文档中，进一步实现自动化，开发两个小工具，解决实际生活中遇到的困难。

另外不得不说，Mathematica的本地帮助文档真的很强大，不仅有简单的例子，还有不同的进阶用法，我遇到困难都会翻它，能解决90%以上的问题！

通过这次期末项目大作业的实践，我深刻感受到了Mathematica功能之强大，简直无所不包，各种Function拿来就用。这次项目作业也让我的Mathematica编程水平有了极大的提升，能额外掌握一门编程语言真的很令人开心！

最后的最后，希望老师能够原谅我迟交这么久，若要酌情扣分还望手下留情啊。。。