

自然语言处理大作业二报告

07152001 班 史桢彬 1120201198

电话: 13152715609 邮箱: ayaishere@163.com

一. 题目选择

【汉语词义自动消歧系统】

词义消歧, 英文名称为 Word Sense Disambiguation, 英语缩写为 WSD, 其任务是根据歧义词语的语义环境确定其在该使用环境中的语义。

很多词汇具有一词多义的特点, 但一个词在特定的上下文语境中其含义却是确定的。很多词语都有几个意思或者语义, 如果把这样的词从上下文中独立出来考虑, 就会产生语义歧义。

通常情况下, 人类可以很容易地解析出词语在特定环境中的语义, 但计算机却无法分辨出不同环境中词语的不同指代对象及不同意思。汉语词义自动消歧系统就可以在无需人脑的条件下自动根据不同上下文判断某一词语的特定含义。

二. 实验环境

ASUS VivoBook + Win10 + Pycharm 2021.2.3 + Python 3.9 + Anaconda 3.7

三. 实验人员及分工情况

由本人独自完成本项目。

四. 项目成果展示及分析

项目成果分为视频与截图两部分, 视频请详见项目文件夹“视频.mp4”文件, 截图详见下方。

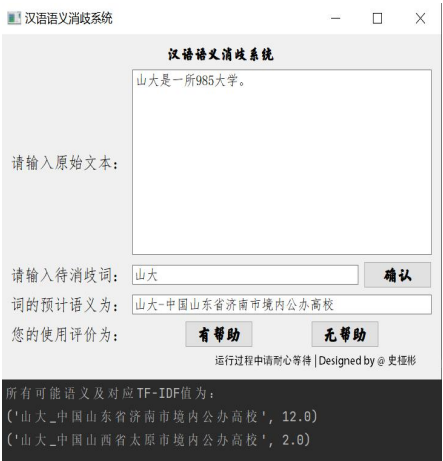
截图中包含有“火箭”、“遇见”、“苹果”、“山大”、“法网”、“林夕”、“奥利奥”、“小米”等歧义词在某一特定语义环境中的自动消歧结果。其中, 原文本、歧义词、预测语义及正确率均记录在 SQLITE3 数据库中。

原文本	歧义词	预测语义	是否正确	目前正确率
火箭队的比赛很精彩!	火箭	火箭-NBA职业篮球队	1	100.0%
苹果手机有着广泛的海外市场。	苹果	苹果-美国高科技公司	1	100.0%
孙燕姿的遇见真的很好听。	遇见	遇见-孙燕姿演唱歌曲	1	100.0%
山大是一所985大学。	山大	山大-中国山东省济南市境内公办高校	1	100.0%
昨天的法网女单决赛你看了吗?	法网	法网-网球大满贯赛事	1	100.0%
林夕的词写得真不错!	林夕	林夕-中国香港词作人、创作总监	1	100.0%
你爱吃奥利奥吗?	奥利奥	奥利奥-美国饼干品牌	1	100.0%
火箭队的比赛很精彩	火箭	火箭-NBA职业篮球队	1	100.0%
小米是雷军一手养大的公司	小米	小米-2010年成立的互联网企业, 世界!1	1	100.0%
你喜欢吃可爱多雪糕吗	可爱多	可爱多-冰淇淋品牌	1	100.0%

实验过程中共测试不同语境十次, 汉语词义自动消歧系统均成功分析出词语的真实语义, 但运行期间, 不同词语的词义分析速度不同, 主要与词语的义项数目及不同义项在百度百科中的与语料数目有关, 但基本均在 1 分钟内完成语义消歧。

可以看出, 本系统具有一定的消歧能力, 但对于语料不丰富的义项或语句信息不充分的文本, 消歧的效果不佳, 会出现“该文本信息量不足, 无法判断语义”的判断结果; 对于无歧义或无明显歧义的词语, 会出现“该词语无歧义”的判断结果。

系统在测试过程中, 主要对名词进行消歧处理, 并未对其余词性词语进行分析处理, 因此该系统后续仍有一定的改进空间。



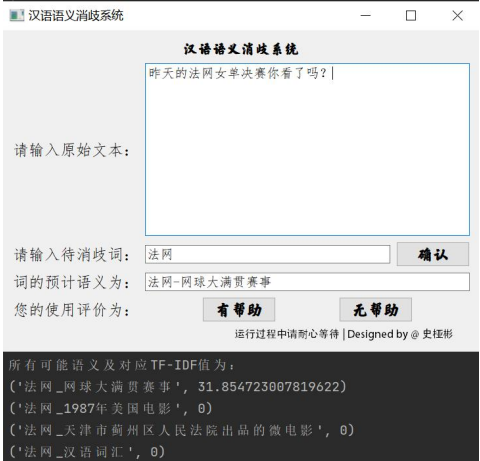
(山大)



(火箭)



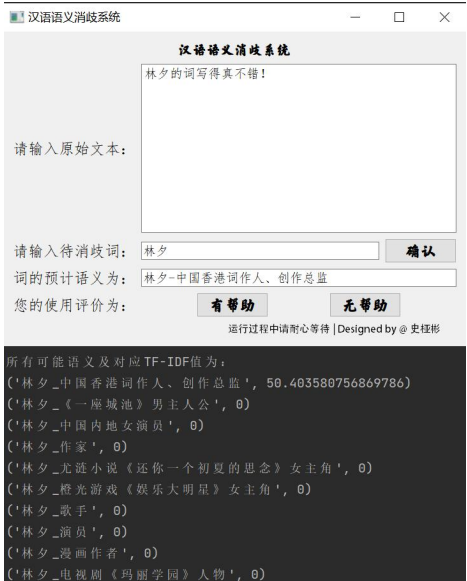
(苹果)



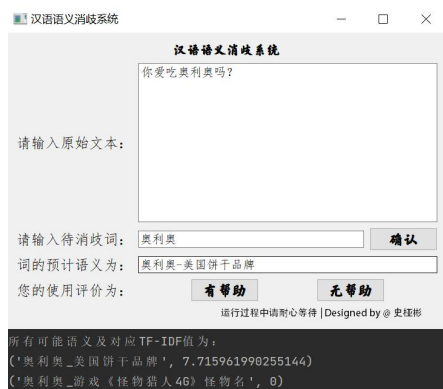
(法网)



(遇见)



(林夕)



(奥利奥)



(小米)

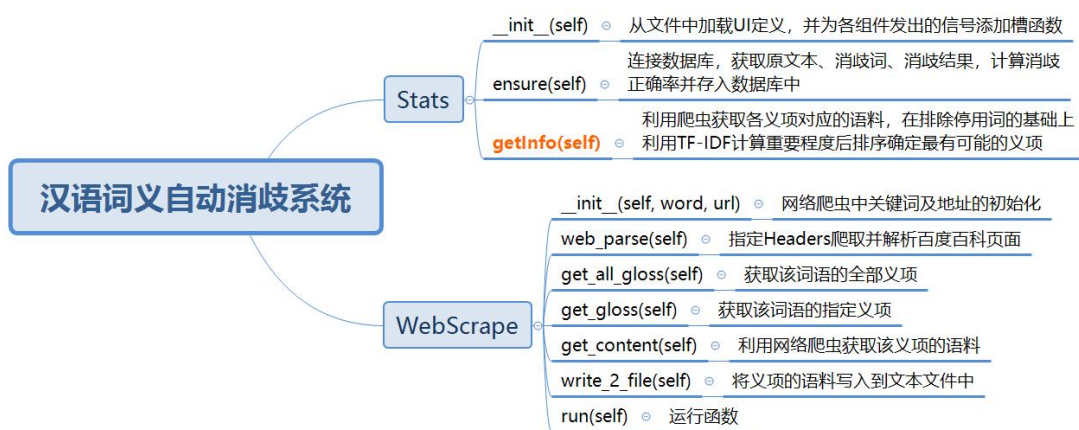
五. 核心思想和算法描述

本实验采用的算法为基于网络爬虫的 Lesk + TF-IDF 算法。核心思想是利用网络爬虫爬取歧义词在百度百科网站中的全部可能义项及相关语料，基于各义项的语料为每一个义项构建一个词典，然后将含有歧义词的文本的分词结果与各词典进行对比分析，判断相似性并计算各语义对应的 TF-IDF 值，将 TF-IDF 值最大的义项作为该词语在文本中的最可能语义。

其中，Lesk 算法是迈克·莱斯克于 1986 年提出的词义消歧算法。该算法基于词语会与上下文有相同的主题这个假设，认为一个词在词典中的词义解释与该词所在的句子具有相似性，将有歧义的词语在字典中的定义与上下文进行比较，分析确定词语最有可能的义项。

TF-IDF 是一种用于信息检索与数据挖掘的常用加权技术，用以评估字词对一个语料库中的其中一份文件的重要程度。TF-IDF 的主要思想是如果某个词或短语在一篇文章中的出现频率 TF 很高，并且在其他文章中的出现频率 IDF 很低，则认为此字词具有很好的类别区分能力，适合用来分类。

六. 系统主要模块流程



- `__init__(self):`

```
def __init__(self):
    # 从文件中加载UI定义
    self.ui = uic.loadUi("myWindow.ui")
    # 按钮禁用
    self.ui.pushButton_2.setEnabled(False)
    self.ui.pushButton_3.setEnabled(False)
    # 定义不同按钮的槽函数
    self.ui.pushButton.clicked.connect(self.getInfo)
    self.ui.pushButton_2.clicked.connect(self.ensure)
    self.ui.pushButton_3.clicked.connect(self.ensure)
```

- `ensure(self):`

```
def ensure(self):
    # 连接数据库
    conn = sqlite3.connect('myLink.db')
    cur = conn.cursor()
    with conn:
        # 获取用户自行判断的预测结果
        if self.ui.sender() == self.ui.pushButton_2:
            right = 1
        elif self.ui.sender() == self.ui.pushButton_3:
            right = 0
        # 获取原文本和消歧词
        text = self.ui.plainTextEdit.toPlainText()
        word = self.ui.lineEdit.text()
        predict = self.ui.lineEdit_2.text()

        cur.execute("select * from myCounter WHERE 是否正确=1")
        res_right = len(cur.fetchall()) + right
        cur.execute("select * from myCounter")
        res_total = len(cur.fetchall()) + 1
        # 计算目前正确率
        percent = str(round(res_right / res_total * 100, 1)) + "%"

        # 将结果存入数据库
        cur.execute("""INSERT INTO myCounter VALUES (?, ?, ?, ?, ?)""", (text, word, predict, right, percent))
```

- `getInfo(self):`

```
def getInfo(self):
    # 获取待消歧文本sent和歧义词wsd_word
    sent = self.ui.plainTextEdit.toPlainText()
    wsd_word = self.ui.lineEdit.text()

    myList = []

    # 利用爬虫获取基于百度百科的语义基址starturl
    base_url = "https://baike.baidu.com"
    quote = urllib.parse.quote(wsd_word, encoding="utf-8")
    starturl = base_url + "/item/" + quote
    myList.append(starturl)

    # 利用爬虫获取其余语义变址的列表ans，并对各网址进行遍历
    ans = str(WebScrape(wsd_word, starturl).get_all_gloss()).split()
    for i in ans:
        if ('href=' in i):
            myUrl = base_url + i.strip("href=")
            if ('>' not in myUrl):
                myList.append(myUrl)
```

```

# 对歧义词所有语义在百度百科网站中的网址进行爬取，获得不同义项的语料
for i in mylist:
    url = i
    WebScrape(wsd_word, url).run()

# 向分词词典添加歧义词便于分词，并选用精确模式进行分词
jieba.add_word(wsd_word)
sent_words = list(jieba.cut(sent, cut_all=False))

# 加入“哈工大停用词库”、“四川大学机器学习智能实验室停用词库”、百度停用词表”等各种停用词表的综合停用词表
with open('stopword.txt', 'r', encoding='utf-8') as f:
    stopwords = [_.strip() for _ in f.readlines()]
stopwords.append(wsd_word)

# 将分词结果中的停用词剔除
sent_cut = []
for word in sent_words:
    if word not in stopwords:
        sent_cut.append(word)
print(sent_cut)
print()

```

```

# 计算词的TF-IDF以及频数
wsd_dict = {}
for file in os.listdir('./' + wsd_word):
    if wsd_word in file:
        wsd_dict[file.replace('.txt', '')] = read_file(wsd_word + '/' + file)

# 统计每个词语在语料中出现的次数
tf_dict = {}
for meaning, sents in wsd_dict.items():
    tf_dict[meaning] = []
    for word in sent_cut:
        word_count = 0
        for sent in sents:
            example = list(jieba.cut(sent, cut_all=False))
            word_count += example.count(word)

        if word_count:
            tf_dict[meaning].append((word, word_count))

idf_dict = {}
for word in sent_cut:
    document_count = 0
    for meaning, sents in wsd_dict.items():
        for sent in sents:

```

```

            if word in sent:
                document_count += 1

    idf_dict[word] = document_count

# 输出值
total_document = 0
for meaning, sents in wsd_dict.items():
    total_document += len(sents)

# 计算TF-IDF值
mean_tf_idf = []
for k, v in tf_dict.items():
    tf_idf_sum = 0
    for item in v:
        word = item[0]
        tf = item[1]
        tf_idf = item[1] * log2(total_document / (1 + idf_dict[word]))
        tf_idf_sum += tf_idf

    mean_tf_idf.append((k, tf_idf_sum))

```



```
# 对不同词义的可能概率排序
sort_array = sorted(mean_tf_idf, key=lambda x: x[1], reverse=True)

# 根据词频及TF-IDF值确定是否有歧义及最大概率的语义
if (len(sort_array) == 0):
    # 按钮启用
    self.ui.lineEdit_2.setText("该词语无歧义。")
else:
    if sort_array[0][1] == 0:
        self.ui.lineEdit_2.setText("该文本信息量不足，无法判断语义。")
    else:
        # 按钮启用
        self.ui.pushButton_2.setEnabled(True)
        self.ui.pushButton_3.setEnabled(True)
        print("所有可能语义及对应TF-IDF值为：")
        for i in sort_array:
            print(i)
        true_meaning = sort_array[0][0].split('_')[1]
        self.ui.lineEdit_2.setText(wsd_word + "-" + true_meaning)
```

- `__init__(self, word, url):`

```
def __init__(self, word, url):
    self.url = url
    self.word = word
```

- `web_parse(self):`

```
# 爬取百度百科页面
def web_parse(self):
    # 指定headers
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36'}
    req = requests.get(url=self.url, headers=headers)

    # 解析网页，定位到main-content部分
    if req.status_code == 200:
        soup = BeautifulSoup(req.text.encode(req.encoding), 'lxml')
        return soup
    return None
```

- `get_all_gloss(self):`

```
# 获取该词语的全部义项
def get_all_gloss(self):
    soup = self.web_parse()
    if soup:
        lis = soup.find('ul', class_="polysemanList-wrapper cmn-clearfix")
        return lis
```

- `get_gloss(self):`

```
# 获取指定网站的义项
def get_gloss(self):
    soup = self.web_parse()
    if soup:
        lis = soup.find('ul', class_="polysemanList-wrapper cmn-clearfix")
        if lis:
            for li in lis('li'):
                if '<a' not in str(li):
                    gloss = li.text.replace('=', ' ')
                    return gloss
    return None
```

- `get_content(self):`

```
# 获取该义项的语料，以句子为单位
def get_content(self):
    # 发送HTTP请求
    result = []
    soup = self.web_parse()
    if soup:
        paras = soup.find('div', class_='main-content').text.split('\n')
        for para in paras:
            if self.word in para:
                sents = list(SentenceSplitter.split(para))
                for sent in sents:
                    if self.word in sent:
                        sent = sent.replace('\xa0', '').replace('\u3000', '')
                        result.append(sent)
    result = list(set(result))
    return result
```

- `write_2_file(self):`

```
# 将该义项的语料写入到txt
def write_2_file(self):
    gloss = self.get_gloss()
    result = self.get_content()
    path = r'./'+ self.word
    if (os.path.exists(path) == False):
        os.mkdir(path)
    if result and gloss:
        with open('./' + self.word + '/%s_%s.txt' % (self.word, gloss), 'w', encoding='utf-8') as f:
            f.writelines(['_ + '\n' for _ in result])
```

- `run(self):`

```
# 运行
def run(self):
    self.write_2_file()
```

七. 实验亮点

本实验采用“网络爬虫 + 百度百科语料 + 综合停用词库 + Leck-TF-IDF 算法 + PyQt5 界面可视化 + SQLITE3 数据库支持”的模式。

- **网络爬虫.** 本次实验的语料获取途径为网络爬虫，能够实现对义项、语料的及时同步，避免了语料库不更新影响消歧效果的可能。
- **百度百科语料.** 本次实验选用的语料库为词语在百度百科中的不同义项对应页面的文本语料，数量庞大，定位明确，兼顾丰富性与正确性。
- **综合停用词库.** 本次实验选用的是“哈工大停用词词库”、“四川大学机器学习智能实验室停用词库”、“百度停用词表”等各种停用词表的综合停用词表，含有 1500+ 条停用词。
- **Leck-TF-IDF 算法.** 本次实验选用基于词典的经典算法 Leck，并利用 TF-IDF 对义项重

要性进行评价分析，具有良好的合理性和研究空间。

- **PyQt5 界面可视化**. 本次实验选用 PyQt5 模块对代码进行可视化，实验界面简约大方，具有良好的可视性与操作性。
- **SQLITE3 数据库支持**. 本次实验允许用户通过“有帮助”、“无帮助”按钮记录操作内容及结果，便于分析实验结果、计算消歧正确率。

八. 实验心得

通过这次实验，我第一次学习实践了比较有名的 TF-IDF，还是非常有成就感的。从简单的 Pycharm 里的控制台运行到最后的 Qt 可视化界面，从即时运行到数据库存储运行结果，每一点完善带来的视觉冲击和自豪感让我不断地完善整个程序，最后呈现出这样一个还算有模有样的结果，我非常开心。

运行过程中，我也不断努力地试图改进消歧结果，从语料优化，到停用词广覆盖，再到算法改进，整个过程坎坷异常但也颇有趣味。

最后，有点遗憾的是整个系统主要还是面向名词进行消歧，其他词性的词语的消歧结果并没有很好地进行验证，这点还是有很大的测试改进空间，今后得闲我还会去改进代码逻辑。

非常感谢老师布置这次作业，让我有机会以完成任务的实践形式深化自己的知识，真正做出点什么，我真的收获颇丰，感激！

附录

```
import os
import jieba
import sqlite3
import requests
import urllib.parse
from PyQt5 import uic
from math import log2
from bs4 import BeautifulSoup
from pyltp import SentenceSplitter
from PyQt5.QtWidgets import QApplication

class Stats:
    def __init__(self):
        # 从文件中加载 UI 定义
        self.ui = uic.loadUi("myWindow.ui")
        # 按钮禁用
        self.ui.pushButton_2.setEnabled(False)
        self.ui.pushButton_3.setEnabled(False)
        # 定义不同按钮的槽函数
        self.ui.pushButton.clicked.connect(self.getInfo)
        self.ui.pushButton_2.clicked.connect(self.ensure)
        self.ui.pushButton_3.clicked.connect(self.ensure)
```



```

def ensure(self):
    # 连接数据库
    conn = sqlite3.connect('myLink.db')
    cur = conn.cursor()
    with conn:
        #获取据用户自行判断的预测结果
        if self.ui.sender() == self.ui.pushButton_2:
            right = 1
        elif self.ui.sender() == self.ui.pushButton_3:
            right = 0
        # 获取原文本和消歧词
        text = self.ui.plainTextEdit.toPlainText()
        word = self.ui.lineEdit.text()
        predict = self.ui.lineEdit_2.text()

        cur.execute("select * from myCounter WHERE 是否正确=1")
        res_right = len(cur.fetchall()) + right

        cur.execute("select * from myCounter")
        res_total = len(cur.fetchall()) + 1

        # 计算目前正确率
        percent = str(round(res_right / res_total * 100, 1)) + "%"

        # 将结果存入数据库
        cur.execute("""INSERT INTO myCounter VALUES (?, ?, ?, ?, ?)""", (text,
word, predict, right, percent))

def getInfo(self):

    # 获取待消歧文本 sent 和歧义词 wsd_word
    sent = self.ui.plainTextEdit.toPlainText()
    wsd_word = self.ui.lineEdit.text()

    myList = []

    # 利用爬虫获取基于百度百科的语义基址 starturl
    base_url = "https://baike.baidu.com"
    quote = urllib.parse.quote(wsd_word, encoding="utf-8")
    starturl = base_url + "/item/" + quote
    myList.append(starturl)

    # 利用爬虫获取其余语义变址的列表 ans，并对各网址进行遍历
    ans = str(WebScrape(wsd_word, starturl).get_all_gloss()).split()

```

```

for i in ans:
    if ('href=' in i):
        myUrl = base_url + i.strip("href=\"")
        if ('>' not in myUrl):
            myList.append(myUrl)

# 对歧义词所有语义在百度百科网站中的网址进行爬取，获得不同义项的语料
for i in myList:
    url = i
    WebScrape(wsd_word, url).run()

# 向分词词典添加歧义词便于分词，并选用精确模式进行分词
jieba.add_word(wsd_word)
sent_words = list(jieba.cut(sent, cut_all=False))

# 加入“哈工大停用词词库”、“四川大学机器学习智能实验室停用词库”、百度停用词表“等各种停用词表的综合停用词表
with open('stopwprd.txt', 'r', encoding='utf-8') as f:
    stopwords = [_.strip() for _ in f.readlines()]
stopwords.append(wsd_word)

# 将分词结果中的停用词剔除
sent_cut = []
for word in sent_words:
    if word not in stopwords:
        sent_cut.append(word)
print(sent_cut)
print()

# 计算词的 TF-IDF 以及频数
wsd_dict = {}
for file in os.listdir('./' + wsd_word):
    if wsd_word in file:
        wsd_dict[file.replace('.txt', '')] = read_file(wsd_word + '/'
+ file)

# 统计每个词语在语料中出现的次数
tf_dict = {}
for meaning, sents in wsd_dict.items():
    tf_dict[meaning] = []
    for word in sent_cut:
        word_count = 0
        for sent in sents:
            example = list(jieba.cut(sent, cut_all=False))

```

```

        word_count += example.count(word)

    if word_count:
        tf_dict[meaning].append((word, word_count))

idf_dict = {}
for word in sent_cut:
    document_count = 0
    for meaning, sents in wsd_dict.items():
        for sent in sents:
            if word in sent:
                document_count += 1

    idf_dict[word] = document_count

# 输出值
total_document = 0
for meaning, sents in wsd_dict.items():
    total_document += len(sents)

# 计算 TF-IDF 值
mean_tf_idf = []
for k, v in tf_dict.items():
    tf_idf_sum = 0
    for item in v:
        word = item[0]
        tf = item[1]
        tf_idf = item[1] * log2(total_document / (1 + idf_dict[word]))
        tf_idf_sum += tf_idf

    mean_tf_idf.append((k, tf_idf_sum))

# 对不同词义的可能概率排序
sort_array = sorted(mean_tf_idf, key=lambda x: x[1], reverse=True)

# 根据词频及 TF-IDF 值确定是否有歧义及最大概率的语义
if (len(sort_array) == 0):
    # 按钮启用
    self.ui.lineEdit_2.setText("该词语无歧义。")
else:
    if sort_array[0][1] == 0:
        self.ui.lineEdit_2.setText("该文本信息量不足，无法判断语义。")
    else:
        # 按钮启用

```

```

        self.ui.pushButton_2.setEnabled(True)
        self.ui.pushButton_3.setEnabled(True)
        print("所有可能语义及对应 TF-IDF 值为：")
        for i in sort_array:
            print(i)
        true_meaning = sort_array[0][0].split('_')[1]
        self.ui.lineEdit_2.setText(wsd_word + "-" + true_meaning)

class WebScrape(object):
    def __init__(self, word, url):
        self.url = url
        self.word = word

# 爬取百度百科页面
def web_parse(self):
    # 指定 headers
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36'}
    req = requests.get(url=self.url, headers=headers)

    # 解析网页，定位到 main-content 部分
    if req.status_code == 200:
        soup = BeautifulSoup(req.text.encode(req.encoding), 'lxml')
        return soup
    return None

# 获取该词语的全部义项
def get_all_gloss(self):
    soup = self.web_parse()
    if soup:
        lis = soup.find('ul', class_="polysemantList-wrapper cmn-clearfix")
        return lis

# 获取指定网站的义项
def get_gloss(self):
    soup = self.web_parse()
    if soup:
        lis = soup.find('ul', class_="polysemantList-wrapper cmn-clearfix")
        if lis:
            for li in lis('li'):
                if '<a' not in str(li):
                    gloss = li.text.replace('▪', '')
                    return gloss
    return None
```

```

# 获取该义项的语料，以句子为单位
def get_content(self):
    # 发送 HTTP 请求
    result = []
    soup = self.web_parse()
    if soup:
        paras = soup.find('div', class_='main-content').text.split('\n')
        for para in paras:
            if self.word in para:
                sents = list(SentenceSplitter.split(para))
                for sent in sents:
                    if self.word in sent:
                        sent = sent.replace('\xa0', '').replace('\u3000',
''))

                        result.append(sent)

    result = list(set(result))
    return result

# 将该义项的语料写入到 txt
def write_2_file(self):
    gloss = self.get_gloss()
    result = self.get_content()
    path = r'./' + self.word
    if os.path.exists(path) == False:
        os.mkdir(path)
    if result and gloss:
        with open('./' + self.word + '/%s_%s.txt' % (self.word, gloss), 'w',
encoding='utf-8') as f:
            f.writelines([_ + '\n' for _ in result])

# 运行
def run(self):
    self.write_2_file()

# 读取每个义项的语料
def read_file(path):
    with open(path, 'r', encoding='utf-8') as f:
        lines = [_ .strip() for _ in f.readlines()]
    return lines

app = QApplication([])
stats = Stats()
stats.ui.show()
app.exec_()

```