



北京邮电大学

Beijing University of Posts and Telecommunications

信息与知识获取 实验报告

[信息提取]

学院：计算机学院
2020211376 马天成
2020211439 李彬傲
2020211487 孙嘉琦
2023 年 6 月 30 日

目录

1. 实验目的和要求.....	3
1.1 实验目的.....	3
1.2 实验要求.....	3
2. 设计和流程.....	3
2.1 输入&输出.....	3
2.2 文本文件目录.....	4
2.3 LTP 中文大模型	4
3. 架构和实现.....	5
3.1 解析和构造模块.....	5
3.1.1 对外封装 – module	5
3.1.1 英文解析 – en	7
3.1.2 中文解析 – ch.....	8
3.2 窗口模块 – window	9
3.3 主函数入口 – main.....	10
4. 功能测试和让人工评价.....	11
4.1 解析	11
4.2 人工评判.....	11
4.2.1 英文文章词性提取.....	11
4.2.2 中文文章词性提取.....	13
4.3 正则表达式匹配.....	14
5. 实验总结.....	16

1. 实验目的和要求

1.1 实验目的

信息抽取是从文本中自动提取结构化信息的任务,其主要目的是从非结构化或半结构化文本中抽取出具有特定语义意义的实体、关系和事件等信息。该实验旨在展示如何使用自然语言处理技术进行信息抽取,并结合正则表达式进行筛选,以便根据特定的模式从文本中抽取出所需的信息。

1.2 实验要求

基本要求: 自己动手设计实现一个信息抽取实验系统, 中、英文皆可, 可以在作业 2 信息检索系统的基础上实现, 也可以单独实现。特定领域语料根据自己的兴趣选定, 规模不低于 100 篇文档, 进行本地存储。对自己感兴趣的特定信息点进行抽取, 并将结果展示出来。其中, 特定信息点的个数不低于 5 个。可以调用开源的中英文自然语言处理基本模块, 如分句、分词、命名实体识别、句法分析。信息抽取算法可以根据自己的兴趣选择, 至少实现正则表达式匹配算法的特定信息点抽取。最好能对抽取结果的准确率进行人工评价。界面不作强制要求, 可以是命令行, 也可以是可操作的界面。提交作业报告和源代码。鼓励有兴趣和有能力的同学积极尝试优化各模块算法, 也可关注各类相关竞赛。

扩展要求: 鼓励有兴趣和有能力的同学积极尝试多媒体信息抽取以及优化各模块算法, 也可关注各类相关竞赛。自主开展相关文献调研与分析, 完成算法评估、优化、论证创新点的过程。

2. 设计和流程

2.1 输入&输出


首先非常抱歉没有时间尝试多媒体的信息提取了。

系统需要的输入是一个文本文件, 其格式是 txt, 解码方式为 utf-8。我们首先使用了实验二的英文文本文件, 在此基础上, 我们还增加了中文识别功能。此外还引入了不同性质的文本。


所以我们的输入主要分英文文本输入和中文文本输入:

- 英文文本

1. 102 个 txt

-  1984-George Orwell.txt

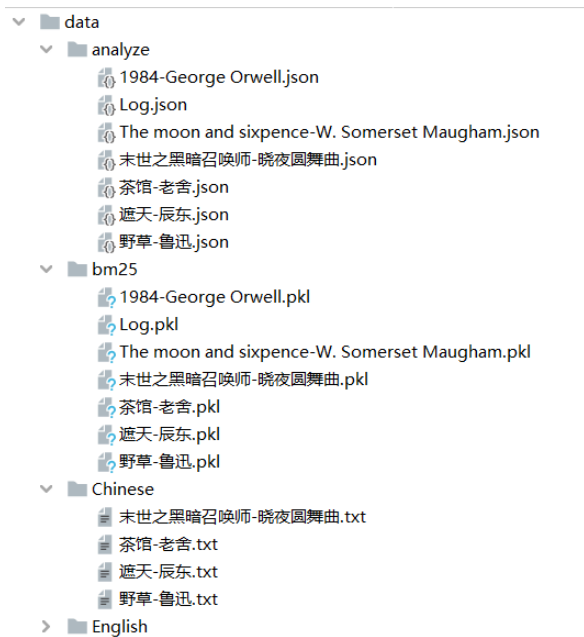
-  Log.txt

2.  The moon and sixpence-W. Somerset Maugham.txt

● 中文文本

末世之黑暗召唤师-晓夜圆舞曲.txt
茶馆-老舍.txt
遮天-辰东.txt
野草-鲁迅.txt

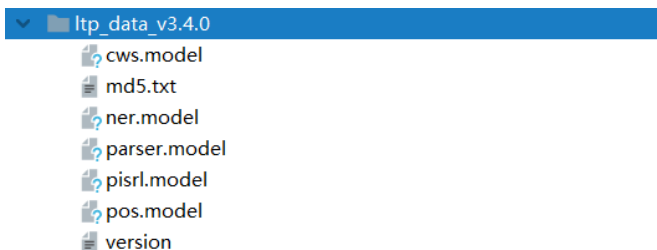
2.2 文本文件目录



分四个文件夹：

1. Chinese：中文文本
2. English：英文文本
3. analyze：语言、分句、分词、词性 tag、命名实体空间
4. bm25：bm25 模型

2.3 LTP 中文大模型



其中主要用到的是这三个文件：

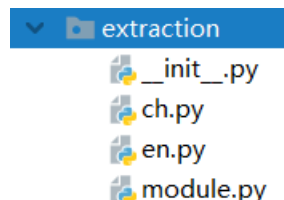
```
class Chinese:
    def __init__(self):
        self.segmentor = Segmentor(os.path.abspath('.')+"\\ltp_data_v3.4.0\\cws.model")
        self.postagger = Postagger(os.path.abspath('.')+"\\ltp_data_v3.4.0\\pos.model")
        self.recognizer = NamedEntityRecognizer(os.path.abspath('.')+"\\ltp_data_v3.4.0\\ner.model")
```

分别对应分词，标 tag，命名实体识别三个功能。

3. 架构和实现

3.1 解析和构造模块

该 package 实现核心的解析文件和信息提取功能。



开放给外界模块是 module，她负责提供解析，检索和提取的接口。

3.1.1 对外封装 – module



对外封装的函数有以上，其注释也写了相应的功能。

这个 package 的对外只有一个调用它的 window 模块（ui 界面），所以集成了所有的功能。

- `__init__(self)`

初始化类的实例，设置文件夹路径和相关属性。初始化时会读取所有已经解析的数据，将 **analyze** 目录下文件的存储内容（语言、分句、分词、词性 tag、命名实体空间）存储到类的 **analyze** 变量，将 **bm25** 目录下文件的存储内容（bm25 模型）存储到类的 **bm25** 变量。

- `get_tag(self, language)`

根据给定的语言，获取相应的词性标签。

参数：**language**：语言类型，可以是"Chinese"或"English"。

返回：如果语言类型有效，则返回相应语言的词性标签列表；否则返回 False。

- **get_re(self)**

获取常用的正则表达式字典。

返回：包含常用正则表达式的字典。

- **generate(self, name, language)**

执行解析，从文本中提取信息，并进行存储。如果文件已被解析，则不执行解析直接返回；若文件未被解析，则根据不同的语言使用不同的解析模块执行解析，并存储解析数据（语言、分句、分词、词性 tag、命名实体空间、bm25 模型）到本地变量。

参数：**name**：文件名；**language**：语言类型，可以是"Chinese"或"English"。

返回：如果解析结果在本地存储中存在，则返回"Analyze from software storage."；否则返回"Analyze from local files."。

- **search_bm25(self, name, query)**

使用 BM25 算法在文本中进行查询，并根据相似度得分排序结果。

参数：**name**：文件名；**query**：查询字符串。

返回：如果文件存在，则返回排序后的文本列表和得分；否则返回 False。

- **search_byTag(self, name, query_tag)**

根据词性标签在文本中进行查询。

参数：**name**：文件名；**query_tag**：查询的词性标签。

返回：包含查询结果的列表。

- **search_reByName(self, name, pattern)**

在指定文件中根据正则表达式模式进行提取。

参数：**name**：文件名；**pattern**：正则表达式模式。

返回：匹配到的结果列表。

- **search_reByData(self, data, pattern)**

在给定的数据中根据正则表达式模式进行提取。

参数：**data**：待提取的数据列表；**pattern**：正则表达式模式。

返回：匹配到的结果列表。

- **filter_named_entities(self, name, data)**

在指定文件的命名实体中过滤给定的数据。

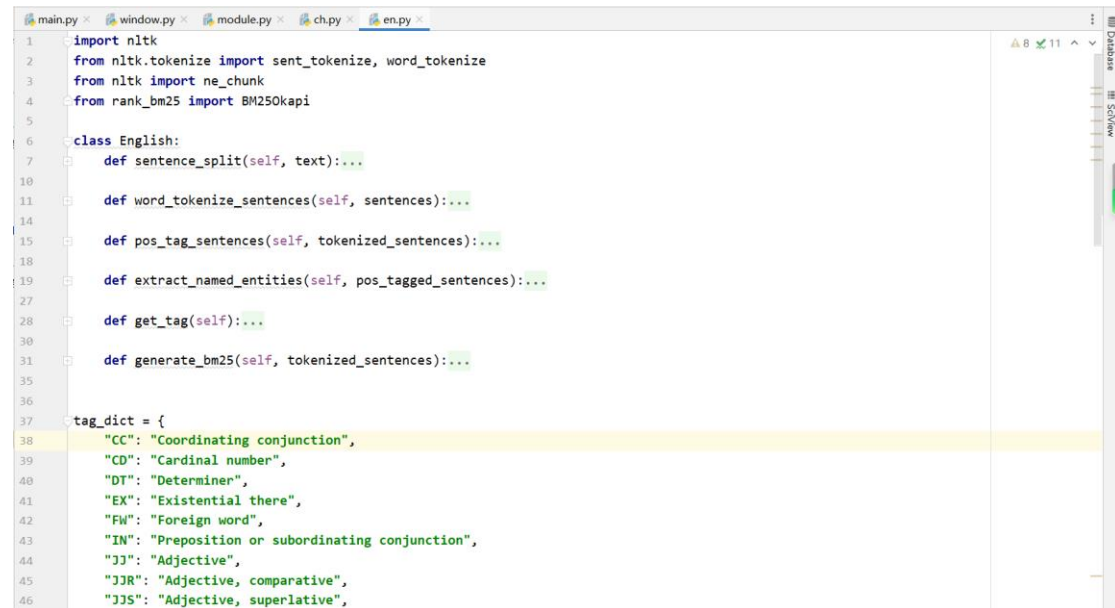
参数：**name**：文件名；**data**：待过滤的数据列表。

返回：过滤后的结果列表。

- **re_dict**

常用的正则表达式字典，包含不同类型的正则表达式模式。

3.1.1 英文解析 – en



```
1 import nltk
2 from nltk.tokenize import sent_tokenize, word_tokenize
3 from nltk import ne_chunk
4 from rank_bm25 import BM25Okapi
5
6 class English:
7     def sentence_split(self, text):...
10
11     def word_tokenize_sentences(self, sentences):...
14
15     def pos_tag_sentences(self, tokenized_sentences):...
18
19     def extract_named_entities(self, pos_tagged_sentences):...
27
28     def get_tag(self):...
30
31     def generate_bm25(self, tokenized_sentences):...
35
36
37 tag_dict = {
38     "CC": "Coordinating conjunction",
39     "CD": "Cardinal number",
40     "DT": "Determiner",
41     "EX": "Existential there",
42     "FW": "Foreign word",
43     "IN": "Preposition or subordinating conjunction",
44     "JJ": "Adjective",
45     "JJR": "Adjective, comparative",
46     "JJS": "Adjective, superlative",
```

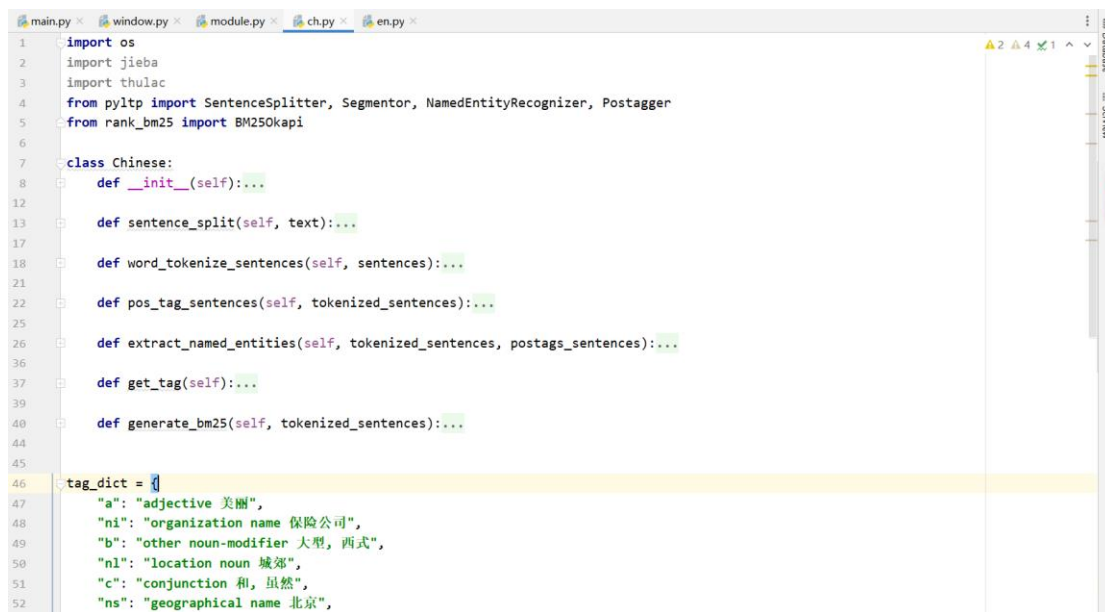
主要是六个函数和一个 tag 字典。tag 字典对应了该语言模型标注的 tag。

:

- **sentence_split(self, text)**
将文本拆分为句子。
参数: **text**: 待拆分的文本。
返回: 句子列表。
- **word_tokenize_sentences(self, sentences)**
对句子进行分词。
参数: **sentences**: 待分词的句子列表。
返回: 分词后的句子列表, 其中每个句子由单词组成。
- **pos_tag_sentences(self, tokenized_sentences)**
对分词后的句子进行词性标注。
参数: **tokenized_sentences**: 已分词的句子列表。
返回: 词性标注后的句子列表, 其中每个句子由词性标注的单词组成。
- **extract_named_entities(self, pos_tagged_sentences)**
从词性标注的句子中提取命名实体。
参数: **pos_tagged_sentences**: 词性标注后的句子列表。
返回: 提取到的命名实体列表。
- **get_tag(self)**
获取英文的词性标签字典。
返回: 包含英文词性标签的字典。

- **generate_bm25(self, tokenized_sentences)**
基于分词后的句子生成 BM25 模型。
参数: **tokenized_sentences**: 已分词的句子列表。
返回: 训练好的 BM25 模型。

3.1.2 中文解析 – ch



```
1 import os
2 import jieba
3 import thulac
4 from pyltp import SentenceSplitter, Segmentor, NamedEntityRecognizer, Postagger
5 from rank_bm25 import BM25Okapi
6
7 class Chinese:
8     def __init__(self):...
12
13     def sentence_split(self, text):...
17
18     def word_tokenize_sentences(self, sentences):...
21
22     def pos_tag_sentences(self, tokenized_sentences):...
25
26     def extract_named_entities(self, tokenized_sentences, postags_sentences):...
36
37     def get_tag(self):...
39
40     def generate_bm25(self, tokenized_sentences):...
44
45
46 tag_dict = {
47     "a": "adjective 美丽",
48     "ni": "organization name 保险公司",
49     "b": "other noun-modifier 大型, 西式",
50     "nl": "location noun 城郊",
51     "c": "conjunction 和, 虽然",
52     "ns": "geographical name 北京",
```

中文就比较复杂了。虽然看起来架构和英文一样，但是需要套大模型，因为中文从分词开始就不好解析。我们尝试了 jieba, thulac, lpt, 最终使用了 ltp 作为我们的语言模型。

- **__init__(self)**
类的构造函数，初始化分词器（Segmentor）、词性标注器（Postagger）和命名实体识别器（NamedEntityRecognizer）。
- **sentence_split(self, text)**
将文本拆分为句子。
参数: **text**: 待拆分的文本。
返回: 句子列表。
- **word_tokenize_sentences(self, sentences)**
对句子进行分词。
参数: **sentences**: 待分词的句子列表。
返回: 分词后的句子列表，其中每个句子由分词结果组成。
- **pos_tag_sentences(self, tokenized_sentences)**
对分词后的句子进行词性标注。
参数: **tokenized_sentences**: 已分词的句子列表。
返回: 词性标注后的句子列表，其中每个句子由词性标注的单词组成。

- **extract_named_entities(self, tokenized_sentences, postags_sentences)**
从分词和词性标注的句子中提取命名实体。
参数: tokenized_sentences: 已分词的句子列表; postags_sentences: 词性标注后的句子列表。
返回: 提取到的命名实体列表。
- **get_tag(self)**
获取中文的词性标签字典。
返回: 包含中文词性标签的字典。
- **generate_bm25(self, tokenized_sentences)**
基于分词后的句子生成 BM25 模型。
参数: tokenized_sentences: 已分词的句子列表。
返回: 训练好的 BM25 模型。

3.2 窗口模块 – window

该模块就是一个很简单的封装操作，封装了一个窗口，实现效果如下：



三个按钮对应了三个功能。这里边函数就不细讲了，代码量还是很大的，主要是做一些控件的位置、功能，事件互斥，数据检查，提醒，解析计时，调用检索提取模块和展示结果集这些功能。主要功能会在后边测试的时候展示，这里就不进行阐述了。

(代码量还是很大的，因为要保证健壮性，中英文选项的互斥，点击事件对数据的修改，各种选择的叠加逻辑，以及数据检验，所以还是花费了将近一半的开发时间)

__init__(self, folder_path, module): 初始化 UIApp 类的实例。它接受一个文件夹路径和一个信息提取模块作为参数，并创建一个 Tkinter 窗口。

create_widgets(self): 创建应用程序的各个部件，包括标签、下拉选项框、输入框、按钮和滚动文本框等。

on_closing(self): 处理窗口关闭事件的函数。当用户关闭窗口时，它会销毁窗口并退出应用程序。

english_combobox_selected(self, event): 处理选择英文书目下拉选项框的事件。在选择英文书目后，它会更新相关的下拉选项框。

chinese_combobox_selected(self, event): 处理选择中文书目下拉选项框的事件。在选择中文书目后，它会更新相关的下拉选项框。

english_tag_combobox_selected(self, event): 处理选择英文 tag 下拉选项框的事件。

chinese_tag_combobox_selected(self, event): 处理选择中文 tag 下拉选项框的事件。

checkbox_click(self): 处理复选框点击事件。根据复选框的状态，设置是否在命名实体空间中提取。

analyze_button_click(self): 解析按钮的点击事件处理函数。调用信息提取模块的 generate 方法解析选中的书目，并将解析结果显示给用户。

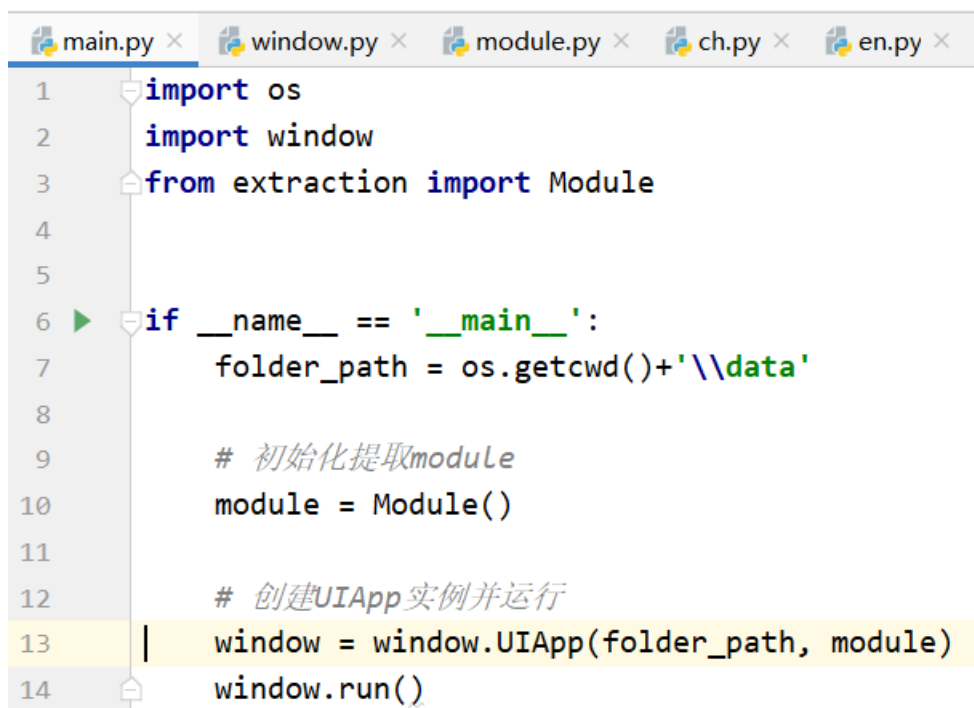
retrieval_button_click(self): 检索按钮的点击事件处理函数。根据输入的信息，在选中的书目中进行检索，并将检索结果显示给用户。

extract_button_click(self): 提取按钮的点击事件处理函数。根据输入的信息，在选中的书目中进行信息提取，并将提取结果显示给用户。

run(self): 启动应用程序的主循环，使用户界面保持运行状态。

3.3 主函数入口 – main

这就没啥好讲的了。



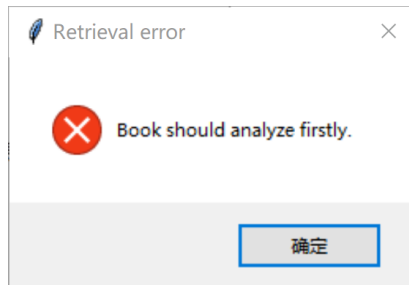
```
1 import os
2 import window
3 from extraction import Module
4
5
6 if __name__ == '__main__':
7     folder_path = os.getcwd()+'\\data'
8
9     # 初始化提取module
10    module = Module()
11
12    # 创建UIApp实例并运行
13    window = window.UIApp(folder_path, module)
14    window.run()
```

初始化解析模块，然后运行窗口（并把初始化后的模块——已经存储了所有已经解析过的数据）传给窗口。

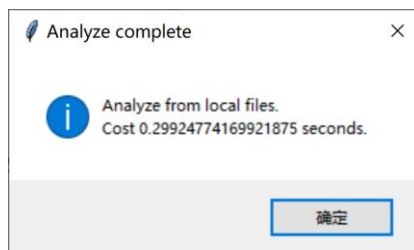
4. 功能测试和让人工评价

4.1 解析

未解析时尝试检索/提取



解析：



4.2 人工评判

4.2.1 英文文章词性提取

我们自己对短文：A Country Music Song Changed Her Life Forever 进行阅读，并找出其中表示数字的词，结果如下

A Country Music Song Changed Her Life Forever⁴²
When Sarah was a teenager, she used to fight over almost everything with her family. But **five** years ago, while she was studying abroad in England, she heard a song full of feelings about returning home on the radio. It made Sarah think about her family and friends back in the US. She came to realize how much she actually missed all of them. Ever since then, she has been a fan of American country music.⁴³
Country is a traditional kind of music from the southern states of America. Nashville, Tennessee is the home of country music. Many songs these days are just about modern life in the US, such as the importance of money and success, but not about belonging to a group.⁴⁴ However, country music brings us back to the "good old days" when people were kind to each other and trusted **one** another. It reminds us that the best things in life are free laughter, friends, family, and the beauty of nature and the countryside.⁴⁵
Sarah hasn't been to Nashville yet, but it is her dream to go there **one** day. She has already read a lot about the place and done some research on it. She knows that there is a Country Music Hall of Fame Museum in Nashville.⁴⁶
There are also always a lot of great country music concerts with famous musicians and singers, like Garth Brooks. Sarah has already listened to most of his songs. "Garth is **one** of the most successful musicians in American history. He's sold more than **120 million** records. I hope to see him sing live **one** day!"⁴⁷

使用抽取系统得到结果如下：

信息提取

英文书目: A Country Music Song Ch. 中文书目: 解析

英文tag: CD-Cardinal number 中文tag:

常用正则表达式: ☐ 是否在命名实体空间中提取

输入信息: 检索 提取

输出结果: Find 7 tokens.
five, one, one, one, 120, million, one.

三个功能: 1.解析选中文本; 2.针对选中文本检索; 3.针对选中文本提取词性&正则式匹配

因此我们认为结果是准确的.

我们自己查询单词: Nashville, 发现三处结果

导航

Nashville

3 个结果

标题 页面 **结果**

music from the southern states of America.
Nashville, Tennessee is the home of country music.

Sarah hasn't been to **Nashville** yet, but it is her dream to go there one day. She has already read

is a Country Music Hall of Fame Museum in **Nashville**.

使用检索系统得到结果如下:

当然我们可以提取很多其他的词汇，比如**加上命名实体空间筛选**后的方位：

信息提取

英文书目:	<input type="text"/>	中文书目:	<input type="text" value="遮天-辰东"/>	<button>解析</button>
英文tag:	<input type="text"/>	中文tag:	<input type="text" value="nl-location noun 城郊"/>	
常用正则表达式:	<input type="text"/>	<input checked="" type="checkbox"/> 是否在命名实体空间中提取		
输入信息:	<input type="text"/>			<input type="button" value="检索"/> <input type="button" value="提取"/>

输出结果:

```
Find 198 tokens.  
东方, 东方, 临头, 临头, 临头, 东方, 东荒, 东荒, 鼎中, 鼎  
中, 鼎中, 鼎中, 鼎中, 鼎中, 石中, 临头, 鼎中, 鼎  
中, 鼎中, 东方, 西皇, 鼎中, 鼎中, 鼎中, 鼎中, 鼎中, 鼎中,  
鼎中, 鼎中, 鼎中, 东荒, 东方, 鼎中, 石中, 墟中, 鼎中, 石中, 东  
方, 墟中, 虚空, 岳中, 东荒, 墟中, 东荒, 鼎中, 石中, 东方  
东方, 东方, 东方, 东方, 东方, 东方, 东方, 东方,  
东方, 东方, 墟中, 墟中, 东荒, 鼎中, 东方, 东方, 墟中, 鼎  
中, 鼎中, 鼎中, 九幽, 然中, 鼎中, 西漠, 东方, 东方, 东方, 东方,  
东方, 东方, 东方, 东方, 东方, 东方, 空中, 东方, 东方, 墟中,
```

三个功能：1.解析选中文本；2.针对选中文本检索；3.针对选中文本提取属性&正则匹配

以及一些专属名词：

[illegible]

4.3 正则表达式匹配

我们提供了日期和书名两个正则表达式。当然你可以在文件里 dict 变量加上更多的值，让他可以提取更多的表达式。这里我们只以实现功能为目标。

对英文文章《Log》提取日期

[illegible]

对中文文章《茶馆》提取书名

信息提取

英文书目:

中文书目:

茶馆-老舍

解析

英文tag:

中文tag:

常用正则表达式:

书籍-《.》

☒ 是否在命名实体空间中提取

输入信息:

检索

提取

输出结果:

Find 5 tokens.
《茶馆》, 《武家坡》, 《纺棉花》, 《辕门斩子》, 《虫八蜡庙》,

三个功能: 1.解析选中文本; 2.针对选中文本检索; 3.针对选中文本提取词性&正则式匹配

5. 实验总结

本实验中，我们创建了一个基于 Tkinter 的用户界面应用程序，用于信息提取，我们所设计的应用程序包含了以下功能：

1. 选择书目：我们设计了英文书目和中文书目的下拉选项框，用户可以从中选择需要处理的书目文件。
2. 解析功能：用户可以点击“解析”按钮来解析选中的书目文件。解析过程将使用预先初始化的模块来提取书目文件中的信息，并将解析结果展示在提示框中。
3. 检索功能：用户可以在输入框中输入检索信息，并点击“检索”按钮来进行文本检索。检索过程将使用 BM25 算法在选中的书目文件中查找与输入信息相似的文本，并将结果按照匹配度排序展示在滚动文本框中。
4. 提取功能：用户可以选择要提取的标签和正则表达式，并点击“提取”按钮来从选中的书目文件中提取符合条件的词汇。提取过程将根据用户选择的标签和正则表达式在书目文件中进行匹配，并将结果展示在滚动文本框中。
5. 命名实体过滤：用户可以选择是否在命名实体空间中进行提取。如果勾选了“是否在命名实体空间中提取”复选框，则提取功能将根据预先识别的命名实体进行过滤，只提取在命名实体空间中的词汇。

用户可以方便地进行书目信息的解析、检索和提取等操作。界面清晰简洁，操作简便直观。我们设计的这个应用程序提供了一个可视化的工具，能帮助用户更好地利用信息提取模块完成相关任务。

● 解析功能

通过解析功能，用户可以选择书目文件并进行解析，提取其中的信息。在实验中，我使用了预先初始化的模块来处理书目文件，提取出标题、作者、出版社等相关信息。通过点击“解析”按钮，应用程序将执行解析过程，并将解析结果展示在提示框中。实验结果表明，解析功能能够准确提取书目文件中的信息，并以清晰的方式呈现给用户。

● 检索功能

实验中，我使用了 BM25 算法来实现检索功能，用户可以在输入框中输入关键字，并点击“检索”按钮来在选中的书目文件中进行文本检索。检索过程将使用 BM25 算法计算输入关键字与书目文件中各文本之间的匹配度，并按照匹配度排序展示结果。实验结果表明，检索功能能够准确找到与输入关键字相似的文本，并以匹配度高低的顺序展示给用户，帮助用户快速找到所需信息。

● 提取功能

在提取功能中，用户可以选择要提取的标签和正则表达式，并点击“提取”按钮来从选中的书目文件中提取符合条件的词汇。实验中，我根据用户选择的标签和正则表达式在书目文件中进行匹配，并将匹配的结果展示在滚动文本框中。实验结果表明，提取功能能够准确提取符合条件的词汇，并将结果以清晰的方式展示给用户。

通过这个实验，我们成功实现了解析、检索和提取功能的用户界面应用程序。这个应用程序提供了一个直观简洁的界面，方便用户进行信息提取相关的操作。实验结果表明，这些功能能够准确地提取、检索和过滤书目文件中的信息，帮助用户更好地利用信息提取模块完

成相关任务。

- 算法优化思考

这里的难点主要是中文处理。我询问了我在做 nlp 方向的同学，首先他给我梳理了流程和核心点，每个步骤的输入和输出，让我对这个工作有了更深入的了解。我们首先使用 jieba，分词非常抽象，才了解到他不是用来做命名实体识别的。之后用了 luc，ltp 最终使用了 ltp 作为我们的选择，因为它的 tag 更加优秀。

- 完成度

除了要求的多媒体没有完成，其他所有工作都完成了。