文件描述

`asoif.2001030836.ttl`：清洗后的三元组文件，这里面的三元组可以认为都是对的

`candidate_entity_replacement_list_v6.jsonl`：三元组以及识别出来替换尾实体的实体，识别不一定正确

`asoif.2001042058.ttl`：用`candidate_entity_replacement_list_v6.jsonl`中的替换实体替换三元组，因此这里面的三元组不一定正确

`candidate_entity_replacement_list_v5.jsonl`：三元组以及识别出来替换尾实体的实体，替换实体记录了多个候选实体，例如对于三元组`e:乔佛里·拜拉席恩， r:继承人， 托曼一世`，记录了可能替换`托曼一世`的多个实体：

`['e:乔佛里·瓦列利安', 'e:托曼·兰尼斯特一世', 'e:托曼·徒利', 'e:托曼·拜拉席恩', 'e:乔斯琳·拜拉席恩']`。进行匹配时可以匹配这些实体来代替`托曼一世`。之所以要这样主要还是没找到好的方法筛选出正确的实体。每个三元组候选实体的数目不一，最多有14个。

三元组的抓取

从网站冰与火之歌中文维基上抓取需要的三元组信息。

先从三个列表网页上抓取人物列表、城堡列表、家族列表

- Category:人物：抓取人物列表
- Category:贵族家族：抓取家族列表
- Category:城堡：抓取城堡列表

通过上述的列表，可以得到每个人物、家族或城堡对应的页面的url。以人物、家族和城堡组成初级的实体，在他们的页面上抓取相关的关系。主要是从页面上的信息栏提取关系数据。以人物`提利昂·兰尼斯特`为例，网页上的信息栏如下：

我们从这个信息栏中可以提取出以下三元组：

```
e:提利昂·兰尼斯特   r:头衔      "前御前首相".
e:提利昂·兰尼斯特   r:头衔      "前财政大臣".
e:提利昂·兰尼斯特   r:势力      e:兰尼斯特家族.
e:提利昂·兰尼斯特   r:势力      "次子团".
e:提利昂·兰尼斯特   r:宗教      "七神信仰".
e:提利昂·兰尼斯特   r:别名      "小恶魔".
e:提利昂·兰尼斯特   r:别名      "半人、半人提利昂".
e:提利昂·兰尼斯特   r:别名      "Boyman".
e:提利昂·兰尼斯特   r:别名      "耶罗".
e:提利昂·兰尼斯特   r:别名      "胡戈·希山".
```

```
e:提利昂·兰尼斯特   r:出生     "273AC，出生于凯岩城".
e:提利昂·兰尼斯特   r:配偶     "第一任，泰莎".
e:提利昂·兰尼斯特   r:配偶     "第二任，珊莎·史塔克".
e:提利昂·兰尼斯特   r:好友     e:波隆.
e:提利昂·兰尼斯特   r:不和     e:瑟曦·兰尼斯特.
e:提利昂·兰尼斯特   r:恋情     e:雪伊.
```

同时，为了方便之后在文本中的匹配，我们为每个实体添加了`name`、`名`、`姓`、`type`四个关系。`type`的尾实体可以取`character`、`castle`或`house`分别对应人物、城堡、家族。例如上述的实体，我们额外添加了四个三元组表示这四个关系：

```
e:提利昂·兰尼斯特   r:name   "提利昂·兰尼斯特".
e:提利昂·兰尼斯特   r:type   "character".
e:提利昂·兰尼斯特   r:名 "提利昂".
e:提利昂·兰尼斯特   r:姓 "兰尼斯特".
```

三元组的清洗

- 将繁体字统一为简体
- 替换三个关系：

  ```
  `r:王后` -> `r:配偶`
  `r:继承者` -> `r:继承人`
  `r:丈夫` -> `r:配偶`
  ```

- 更正抓取的错误：

  ```
  e:乔佛里·拜拉席恩   r:全称     "拜拉席恩家族和兰尼斯特家族的乔佛里一世".
  e:兰尼斯特家族      r:封号     "凯岩王".
  e:兰尼斯特家族      r:封号     "凯岩城公爵".
  e:兰尼斯特家族      r:封号     "兰尼斯港之盾".
  e:兰尼斯特家族      r:封号     "西境守护".
  e:兰尼斯特家族      r:封号     "西境统领".
  e:兰尼斯特家族      r:创建     "机灵的兰恩创建于英雄纪元".
  ```

- 去掉原著書目\原著书目、第*季、出场集数、出场季数、提及集数。
- 去掉纯英语的三元组
- 形成三元组文件：`asoif.2001030836.ttl`

统计信息

抓取的这些初级实体共有2,870个，其中人物实体有2,260个，家族有439个，城堡有171个。三元组共有19,159 个。7,488个尾实体无法匹配实体，以字符串的形式记录。

literal实体的消减

抓取、清洗后的三元组中存在一些没能与现有实体匹配的literal实体。为了使得三元组能更好地匹配到句子，考虑将literal的实体替换成现有的实体。例如`e:托曼·拜拉席恩 r:父亲 "詹姆·兰尼斯特（事实）".`中的尾实体`"詹姆·兰尼斯特（事实）"`替换为现有的人物实体`e:詹姆·兰尼斯特`。这样在寻找三元组对应的句子时就不用直接匹配`"詹姆·兰尼斯特（事实）"`，

而是可以匹配`e:詹姆·兰尼斯特`的名、别名等。

具体方法

记需要替换尾实体的三元组为`s r o`。现有的所有实体集合记为E

1. `o`与E中每个实体计算相似度

   `o`与某一实体e的相似度为：`o`与`e.name`的莱文斯坦比 + `o`与`e.name`的最长公共子串长度/(len(`o`)+len(`e.name`))

2. `s`与E中每个实体计算相似度

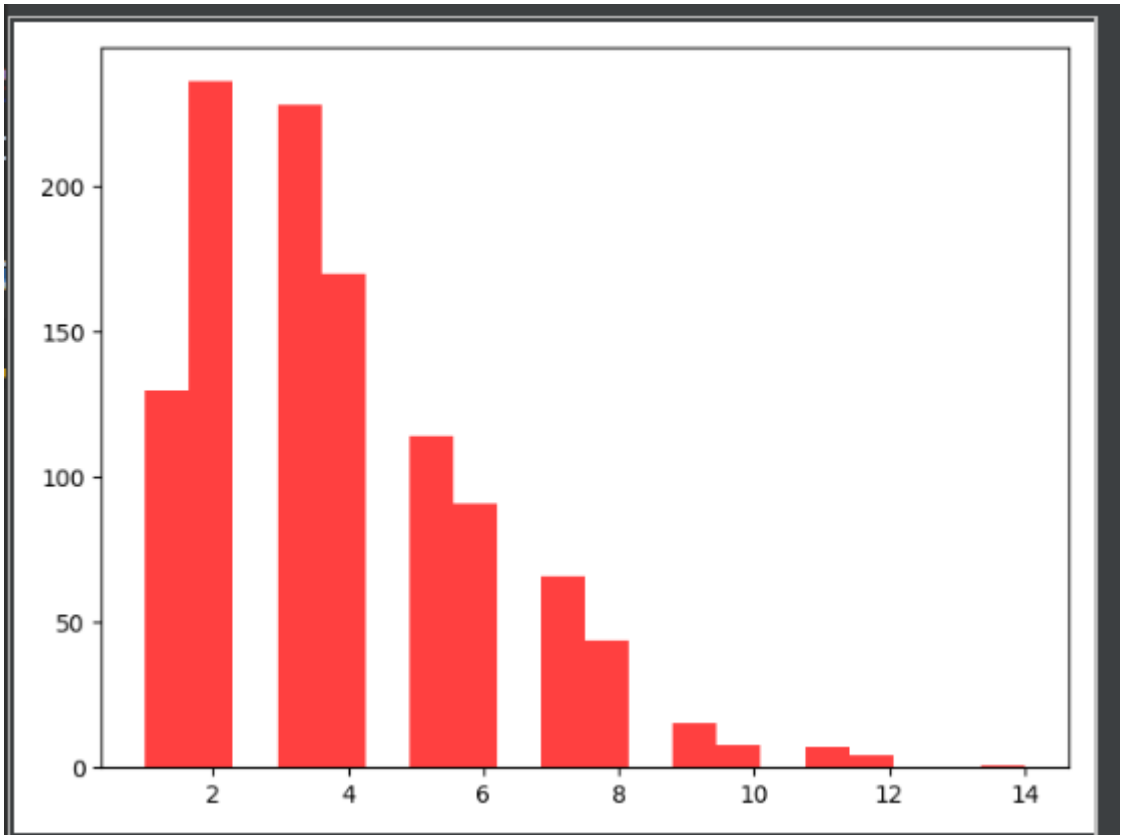
   `s`与某一实体e的相似度为：`s.name`与`e.name`的莱文斯坦比 + `s.name`与$e.name$的最长公共子串长度/(len(`s.name`)+len(`e.name`))

3. 将上述两个相似度都大于0.4的实体都记录下来作为候选实体。假如有三元组没有两个相似度都大于0.4的候选实体，则找出1个两个相似度都大于0.1的实体作为候选实体。再没有的话则该三元组不考虑进行尾实体的替换。

4. 每个候选实体`e`都去文本中计算和头实体`s`的共现次数

5. 根据上述计算，对于一个三元组，可以得到一组候选实体，每个候选实体有三个特征值：与`o`的相似度；与`s`的相似度；与`s`的共现次数

6. 利用每个候选实体的三个特征值，找出候选实体中的skyline点。经过这一步筛选候选实体数量统计如下：

7. 为了在上面的skyline实体中进一步选出一个用于替换的实体，我们考虑寻找skyline实体中的lower bound。

   具体方法是将skyline实体分别按照三个属性值进行排序，并找出在这三个排序中都处于前k个的实体。通过不断降低k的值，直到剩下一个实体即为用于替换的实体

   假如上述做法没能找出唯一一个实体用于替换，则考虑只使用与`o`的相似度；与`s`的共现次数两个属性计算lower bound

   若仍不能找出唯一一个替换实体，则使用与`o`的相似度最大的那个skyline实体进行替换

8. 通过1-7步得到的替换实体记录在文件`candidate_entity_replacement_list_v6.jsonl`中，每一行为一个三元组，用字典表示。key`s`、`r`、`o`分别是原来三元组的主谓宾，`candidate_entity`为选出来替换`o`的实体
9. 用第8步得到的替换实体，实际应用到三元组文件中，得到文件`asoif.2001042058.ttl`

9. 第7步选出的替换实体中仍有一部分是错误的。因此我们也将每个三元组第6步选出的skyline实体记录到文件`candidate_entity_replacement_list_v5.jsonl`，到句子里匹配三元组的时候，可以认为匹配到任意一个skyline实体都是匹配到了`o`