很感谢老师订阅了Shahrur的研究文章并给我分享了博客里其他人的研究方法！

上下游关系确认方法（来自《Industry structure and horizontal takeovers: Analysis of wealth effects on rivals, suppliers, and corporate customers》）：

文章中，作者使用**美国经济的基准投入产出账目**确定客户和供应商行业。

> For any pair of supplier and customer industries, the Use table reports **estimates of the dollar value of the supplier industry's output that is used as input in the production of the customer industry's output**.

> 对于任何一对供应商和客户行业，使用表报告了**供应商行业产出**的美元价值估计值，该价值被用作**客户行业产出生产的投入。**

也就是说作者将 **供应商行业产出 = 客户行业产出生产的投入** 作为判断值

从而确定供应商和客户从而确定上下游关系

所以用到这种方法确定上下游需要用到**美国经济的基准投入产出账目表**，似乎是不适用于我们当前已知信息来确定上下游匹配的(对不起老师qwq

---

公司名称确认方法（来自《Fee_Thomas_2004_Sources of gains in horizontal me》）：

当公司名称与缩写明显相对应时：

1. 将缩写中的**字母数量和顺序**与 the Center for Research in Security Prices (CRSP)上列出的**公司名称中的字母**进行比较
2. 我们从CRSP中确定了四个最有可能与缩写对应的公司名称
3. 在看过并检查后确定几乎确定的明显匹配的情况下，我们**将缩写与CRSP名称和永久标识号联系起来。**

当可能有多个公司于缩写相对应时：

1. 在一些情况下，多个公司名称似乎与缩写对应，在这些情况下，我们使用**Compustat行业细分信息来识别客户。**

对于使用上述方法仍找不到对应公司的缩写：

运用the Directory of Corporate Affiliations（公司从属机构目录），是否无法找到与特定缩写匹配的原因是，该缩写对应于上市公司的子公司。

虽然我们将一些公司编码为可能匹配的公司，但如果无法通过将缩写与前几年的客户描述进行比较或使用上述参考来确认该公司是否匹配，我们会选择将这些公司排除在分析之外。也就是说**无法通过以上方法，判断缩写和公司，只能单独另外用永久标识号区分**

文章中**使用结果数据库来识别样本中合并公司的客户，然后反转数据库来识别供应商**（在我们这里要用这种方法的话，可能需要对所有拥有标识号的公司一一进行数据库匹配客户，然后反转匹配供应商）

---

python进行数据模糊匹配

> 参考链接：在Python|中进行名称匹配的令人惊讶的有效方法作者：马拉深|迈向数据科学 (towardsdatascience.com)

一点学习笔记

- N-grams
  用N-grams进行模糊匹配：
  n元模型中，如果 x = 在句子k中单词的数量，那么句子k中n-grams的数量为 x - (n-1)
  （n = 2或n = 3在句子中比较常用）
- TF-IDF
  术语频率（TF）:一个单词在文档中出现的次数除以该文档中的总单词数，即衡量一个术语在文档中的出现频率。
  反向文档频率（IDF），计算为语料库中文档数的对数除以出现特定术语的文档数，即衡量术语的重要性。
- SciKit Learns algorithm模型：将每一个单词转换成SciKit Learns algorithm可以理解的矢量，一次性计算IDF、TF-IDF和字数

difflib--计算差异的辅助工具

> 参考链接：https://docs.python.org/zh-cn/3/library/difflib.html

difflib. get _close_matches(word, possibilities, n=3, cutoff=0.6)
返回由**最佳"近似"匹配构成的列表。**

- **word 为**一个指定目标近似匹配的序列（通常为**字符串**）
- **possibilities** 为一个由用于匹配 word 的序列构成的列表（通常为**字符串列表**）。
  *可选参数 n (默认为 3) 指定**最多返回多少个近似匹配**； n 必须大于 0.
  *可选参数 cutoff (默认为 0.6) 是一个 [0, 1] 范围内的浮点数。 与 word 相似度得分未达到该值的候选匹配将被忽略。
  *候选匹配中**（不超过 n 个）的最佳匹配将以列表形式返回****，按相似度得分排序，最相似的排在最前面**

```
>>> get_close_matches('appel', ['ape', 'apple', 'peach', 'puppy'])
['apple', 'ape']
>>> import keyword
>>> get_close_matches('wheel', keyword.kwlist)
['while']
>>> get_close_matches('pineapple', keyword.kwlist)
[]
>>> get_close_matches('accept', keyword.kwlist)
['except']
```

我认为用difflib真的能很好的实现公司名称的比对，于是开始尝试用它写代码进行公司名称辨识的工作，明天的工作日志中应该就能出现可用的代码了。