Matching name

1. 改进对*公司全名和公司缩写的数据预处理*代码
    1. 新消除后缀'USA'的影响

```
#判断剩下字符中是否含有以下字符串，如果有则删去
    clean_suffix0 = ['A/S','N.V.','SA/AG','S.A.','A.S.','USA']
    for cs in clean_suffix0:
        if cs in string:
            if string[:string.rfind(cs[0])] == '':
                return string
            else:
                string = string[:string.rfind(cs[0])]
```

2. 改善对数据预处理后结果变为空的情况

```
if string[:string.rfind(cs[0])] == '':
                return string
            else:
                string = string[:string.rfind(cs[0])]
```

改善结果1：有效提高了数据匹配精准度，之前因为'USA'受影响的公司得到了有效匹配的结果，原来没有结果的公司也有了匹配结果

---

2. 改进*公司名称匹配*代码
    1. 原代码中有部分公司在去重时被删去了，导致有的名称没有经过匹配，改进后不会有此bug：

```
        if(similarity > 0.65):

matchcp.append([clname,similarity,full_name[cleaned_name.index(clname)]])
    # 对匹配到的matchcp列表进行去重 不同公司的clname可能相同！
    matchcp = [list(t) for t in set(tuple(_) for _ in matchcp)]
```

2. 增加了比较字符串数量：先根据删去无意义部分后的字符串进行相似度比较，得出相似度大于0.65的字符串（原来是大于0.5），然后排序找出前10个字符串（原来只找出前3个），再根据公司类型、缩写子字符串个数对这10个匹配公司重新排序(原来对公司类型和缩写子字符串的比较代码没有太清晰的思路，这里完成了）：

```
 # 当第一个匹配度为1时，不用比较后面的公司类型和子字符串个数直接写进表中
        if matchcp[0][1] == 1:

            for i,j in zip(range(0,5,2),range(3)):
                Matchcpname.write(indexnow+1, i+2,matchcp[j][2])
                Matchcpname.write(indexnow+1, i+3,matchcp[j][1])
        else:
    # 如果第一个匹配度不为1，按照公司类型和所含子字符串个数排序
        # 1. 比较公司类型
            match = [matchcp[i].append(0)  for i in range(0,10)] #在match的前10个公
司最后加一个0
            matchcp_new = [matchcp[i] for i in range(0,10)] # 把前10个列表提取出来，
组成一个新的列表，代表前10个公司
            for cp_sy_n in matchcp_new: #cp_sy_n(匹配到处理后的公司全名，相似度，处理前
的全名，0)
                if uk_abbr[3] == cptype[full_name.index(cp_sy_n[2])]:
                    # uk_abbr是[3]是缩写公司类型
                    # cptype[cleaned_name.index(cp_sy_n[0])]是找到这个匹配公司的类型
                    # 如果公司类型相同，就给最后那个0+1
                                        cp_sy_n[3] += 1

        # 2.比较匹配到的公司名是否包含有缩写字段
            abbr_sp = uk_abbr[2].split(' ') # 按空格分割公司缩写清理后的字符串
            for ab in abbr_sp: # 遍历切割后的缩写字符串ab
                i = 0 # 从第0个开始遍历
                while(i < 10):# 比较0，1，2，...10个清理后的公司名称
                    if ab in matchcp_new[i][0]: #当这个切割后的子字符串在全名里面存在
时，给他的n值+1
                        matchcp[i][3] += 1
                    i += 1 # 遍历下一个
        # 对matchcp_new按n值从小到大排序
            matchcp_new = sorted(matchcp_new,
                                 key=lambda mn:mn[3]
                                     ,reverse =True)
        # 将最后排序好的matchcp_new写前三个到Match Company name中
            for i,j in zip(range(0,5,2),range(3)):
                Matchcpname.write(indexnow+1, i+2,matchcp_new[j][2])
                Matchcpname.write(indexnow+1, i+3,matchcp_new[j][1])
```

改善结果2:增加了公司类型和缩写子字符串的匹配之后，数据可信度更高了，比如下面这张图：

改进前：

Adapt Pharma Limited	MAAT PHARMA SA	0.88080808

改进后：

	缩写	第一匹配数据
改进前的结果	Adapt Pharma Limited	MAAT PHARMA SA
改进后	Adapt Pharma Limited	ADVENT PHARMA LTD

相比于改进前的结果，改进后很明显改进后考虑了公司类型后，数据的可信度获要高了很多

> **遇到的问题**：
> 有些公司（如缩写为MuleHide Products, Inc的公司，全名为 Mule-Hide Products Company Inc）它的缩写其实就是公司原本的名称，按道理是很好匹配的，但匹配结果却是没有合适的公司（方翔的匹配结果也不合适），关于这类公司的匹配结果有两个可能的猜想：
>
> 1. 原本的总数据中就不含有这个客户公司的数据
> 2. 原本数据中只含该公司的总公司数据，但我们只能人工辨别，而人工辨别对于匹配效率是很低的
>
> **可能采取的措施**：
> 该类数据视为无效数据，不参与供应链模型的设立

Supply Chain Modle

收集并阅读与供应链模型建立相关的文章以拓展思路：

- An object-oriented approach for building a flexible supply chain model
- Application of expected value and chance constraint on uncertain supply chain model with cost, risk and visibility for COVID-19 pandemic