Part1 Knowledge Graph Mining

该部分是对经处理过后的知识图谱文件进行频繁模式挖掘，目标为挖掘出知识图谱上的有效元知识。

该知识图谱文件共有4590547条json记录，每条记录组成为qid字段，types字段，head\_triples字段，tail\_triples字段，types字段中含有零或多个type类型，head\_triples和tail\_triples字段包含零或多个三元组。

types字段中对于一个实体而言一般不会拥有超过10个type，大多数拥有0-4个type。head\_triples和tail\_triples字段在某些实体中可能会出现上万个乃至接近10w个三元组。本次实验的目的是挖掘知识图谱中的有效元知识，重点便在处理types与关系三元组之间的关联，因而types字段中不含type的实体，以及关系三元组为空的实体在实验中并不需要去关注，在数据处理过程中跳过即可，即便存在着大量的空关系与空type实体，剩余的信息量依旧十分庞大。

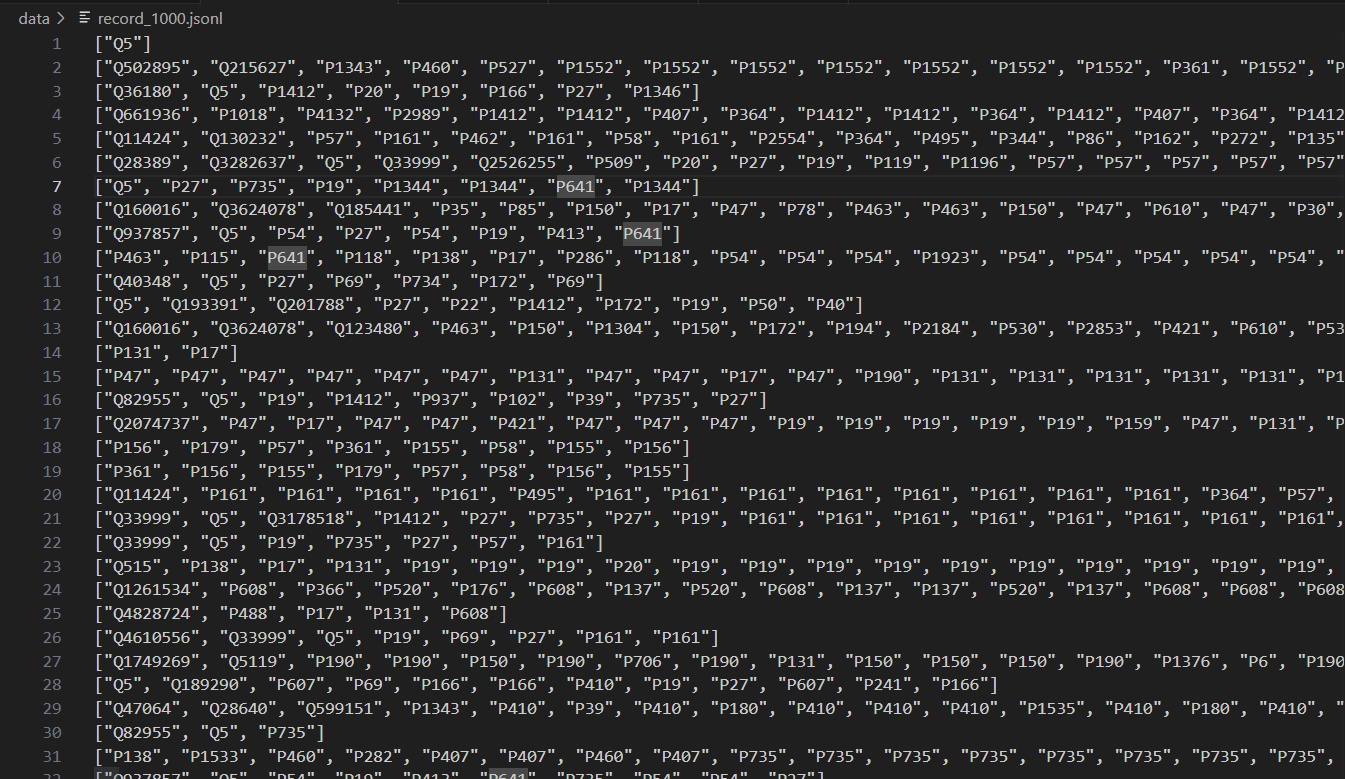
实验过程中选用构造FP-tree，采用FPgrowth算法进行频繁模式挖掘，构造FP-tree只需要遍历两次数据集即可完成构造。记对FP-tree进行频繁项集挖掘所需时间开销为T1，频繁1项集数量为N1，FP-tree平均深度为D1，生成条件模式树开销约为N1\*D1，对于每个深度为D2的条件模式树，生成其所有的频繁项集所需时间约为。

T1 ≈ N1 \* D1 + N1 \* (粗略计算)

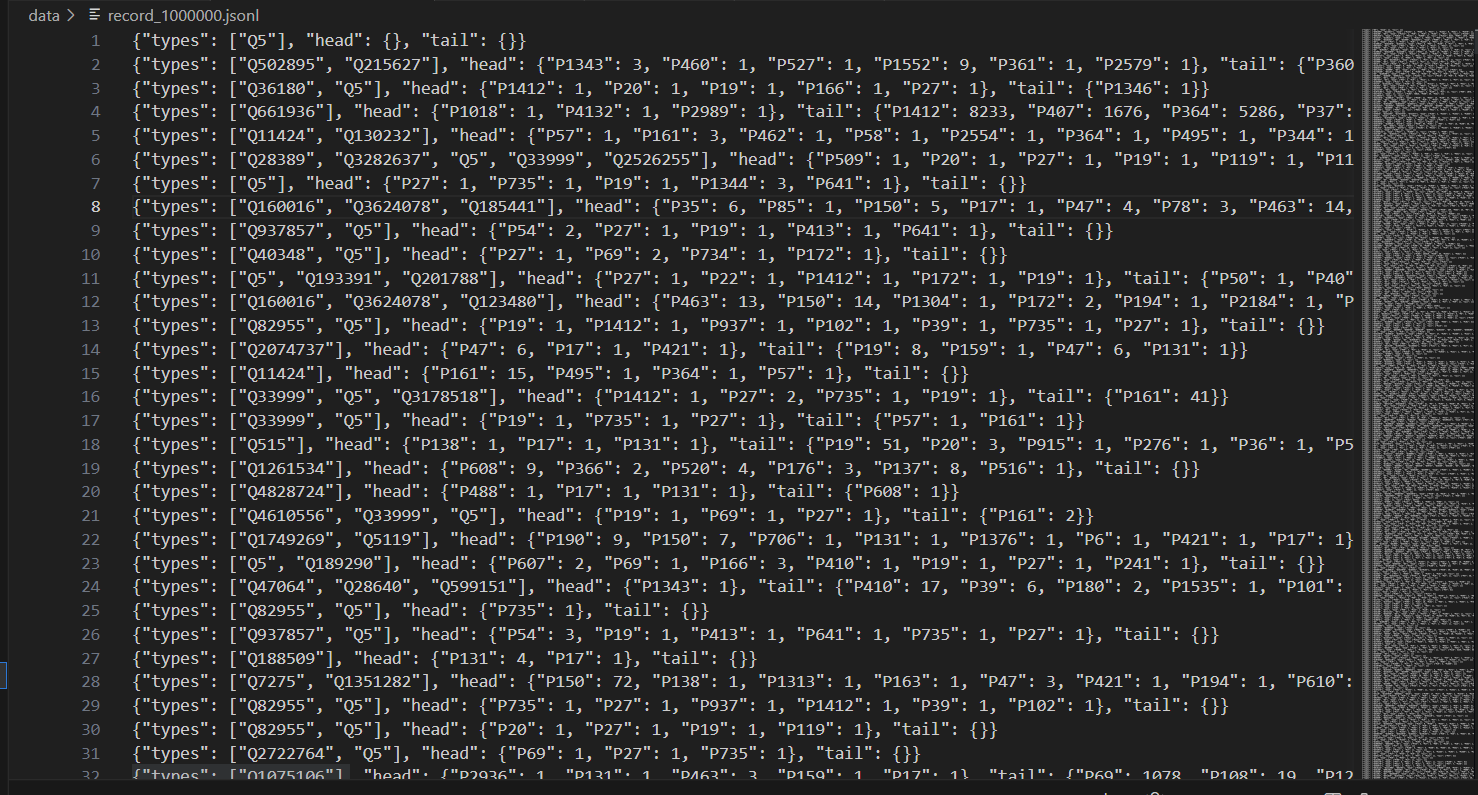
记生成的频繁项集数量为N3，平均频繁项集长度为L，则对频繁项集进行挖掘生成推理规则时，需要对于每一个频繁项集递归剔除项并计算置信度直到频繁项集被剔除至长度为1，则所需时间开销T2 ≈ N3 \* L！。

经过分析可知，该数据挖掘算法的运行效率极大依赖于record的数量以及record长度，因而在设计record时需要适度缩减数量或者缩减长度才能提高挖掘效率。

在设计record的过程中，如果将types + head\_triples中关系（即”PXXX”） + tail\_triples中关系作为一条record生成则有：

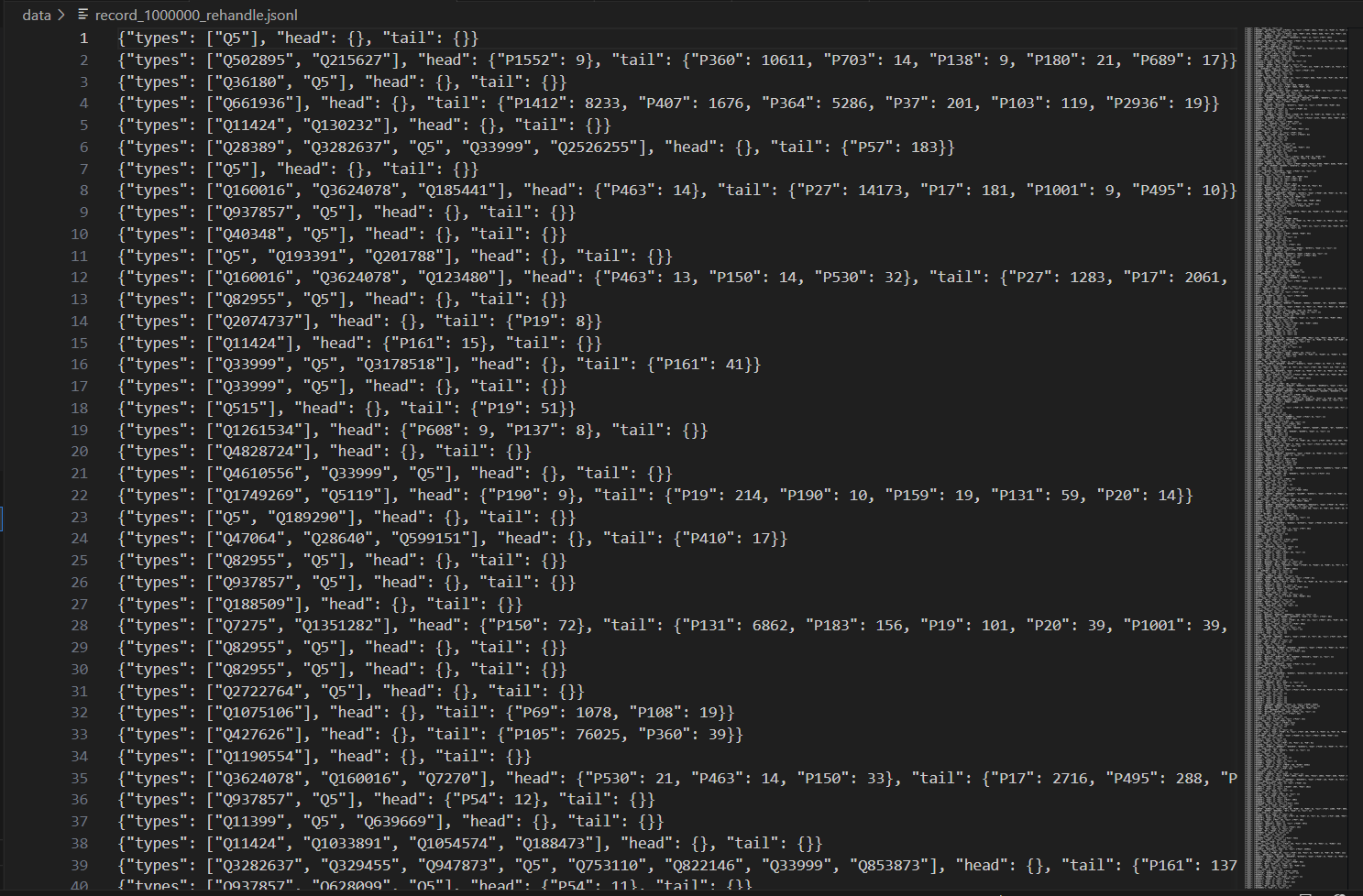


可以观察到关系项是存在大量重复情况的，而record中不应存在重复项，因此应对其做去重处理，但在去重之前先大致统计每个关系项出现的次数，得到结果如下：

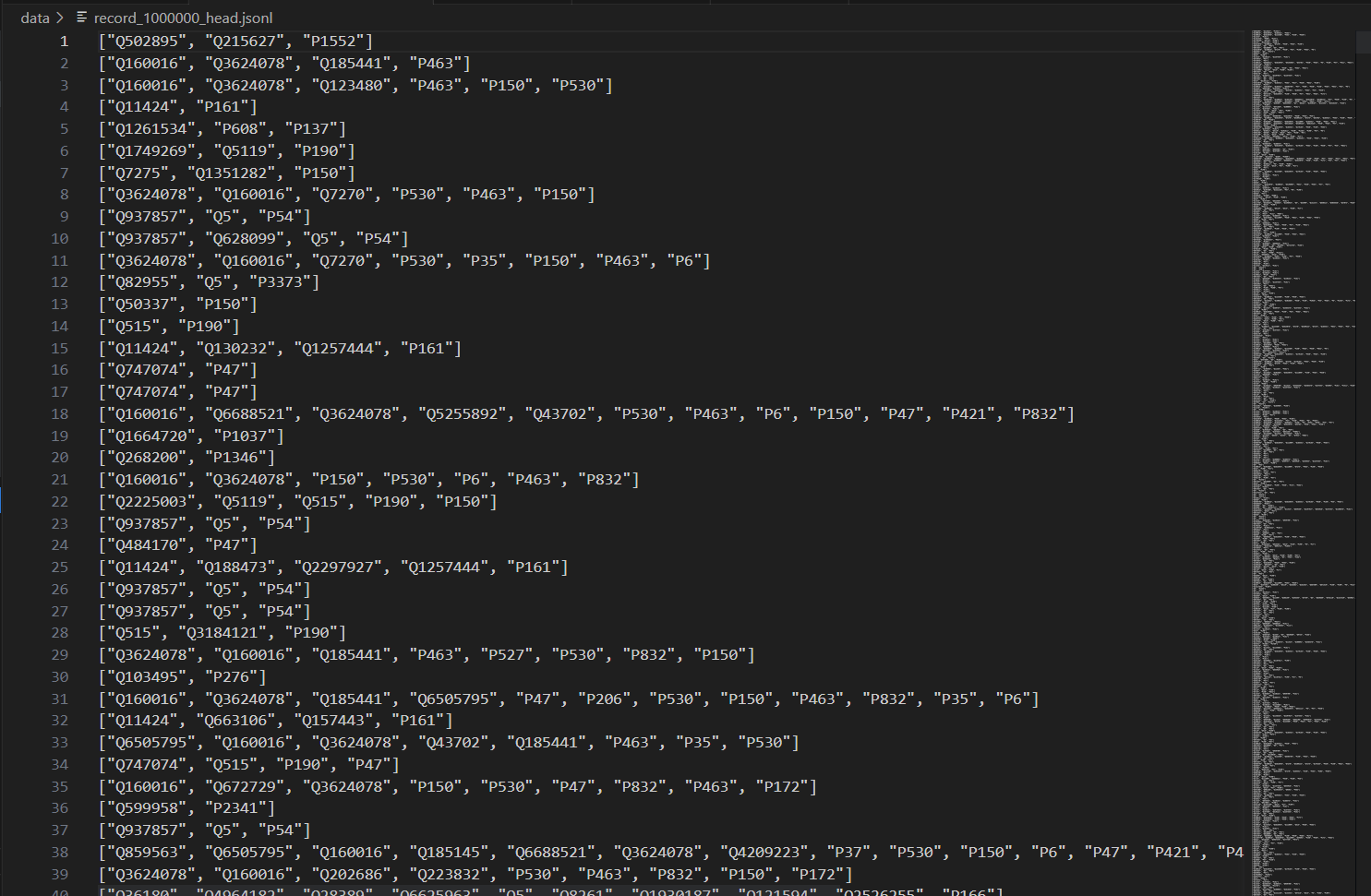


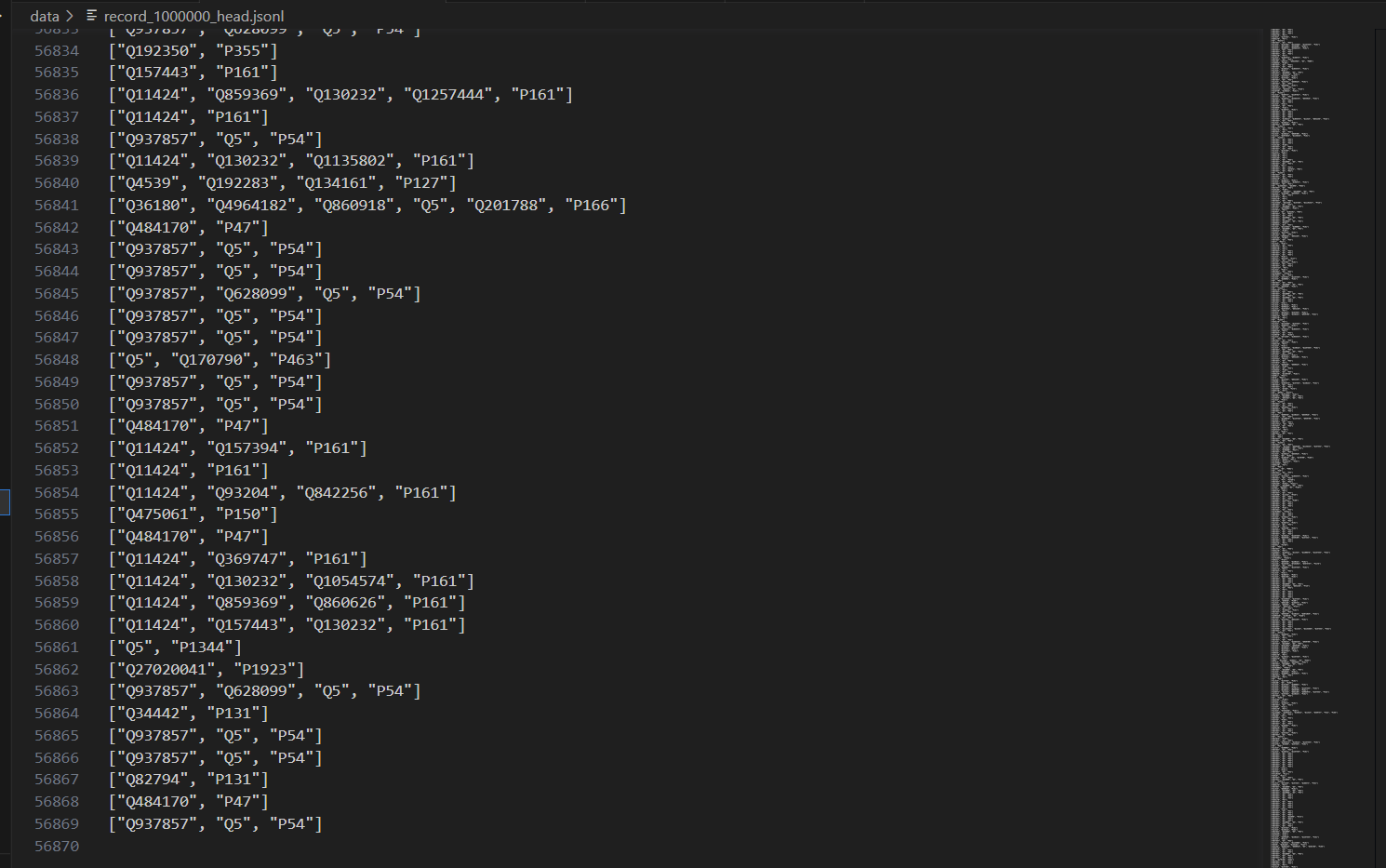
可以观察到知识图谱中一个实体对应的三元组中存在着大量的仅出现一次的关系项，对数据进行去重处理，得到的record取前10000条，其中record最长长度为93，对该数据子集使用上述FPgrowth算法(min\_sup = 10, min\_con = 0.8)，在规则生成部分耗时2h仍未得到挖掘结果，观察其所生成的频繁项集超过20W项，对每个频繁项集挖掘规则时平均耗时2s，同时也存在某些大频繁项集耗时1-2min，故该record设计方法基本可以舍弃。

对上述提取的数据先进行一次筛选，因为已经可以观察到有大量仅出现一次的关系项，因而在筛选时可以设定一个阈值，仅保留大于等于阈值的关系项，如设定阈值为8时得到的数据如下：



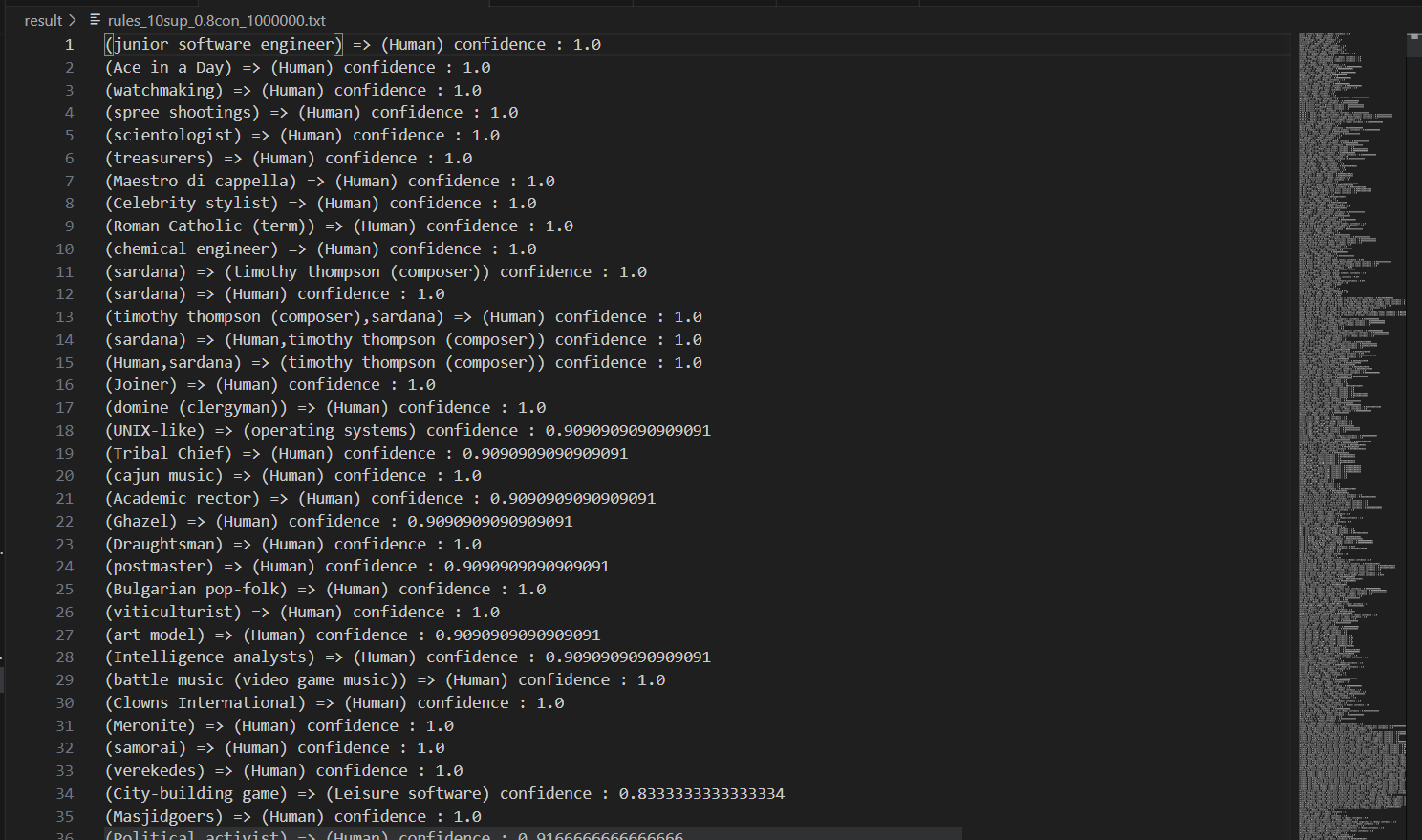
可以观察到大部分的关系项已经被剔除，保留下的关系项出现频率非常高，有些甚至达到上万次，对此数据分割组成新record，按types + head\_关系组成record，再按types + tail\_关系组成另一个record集，得到结果如下：

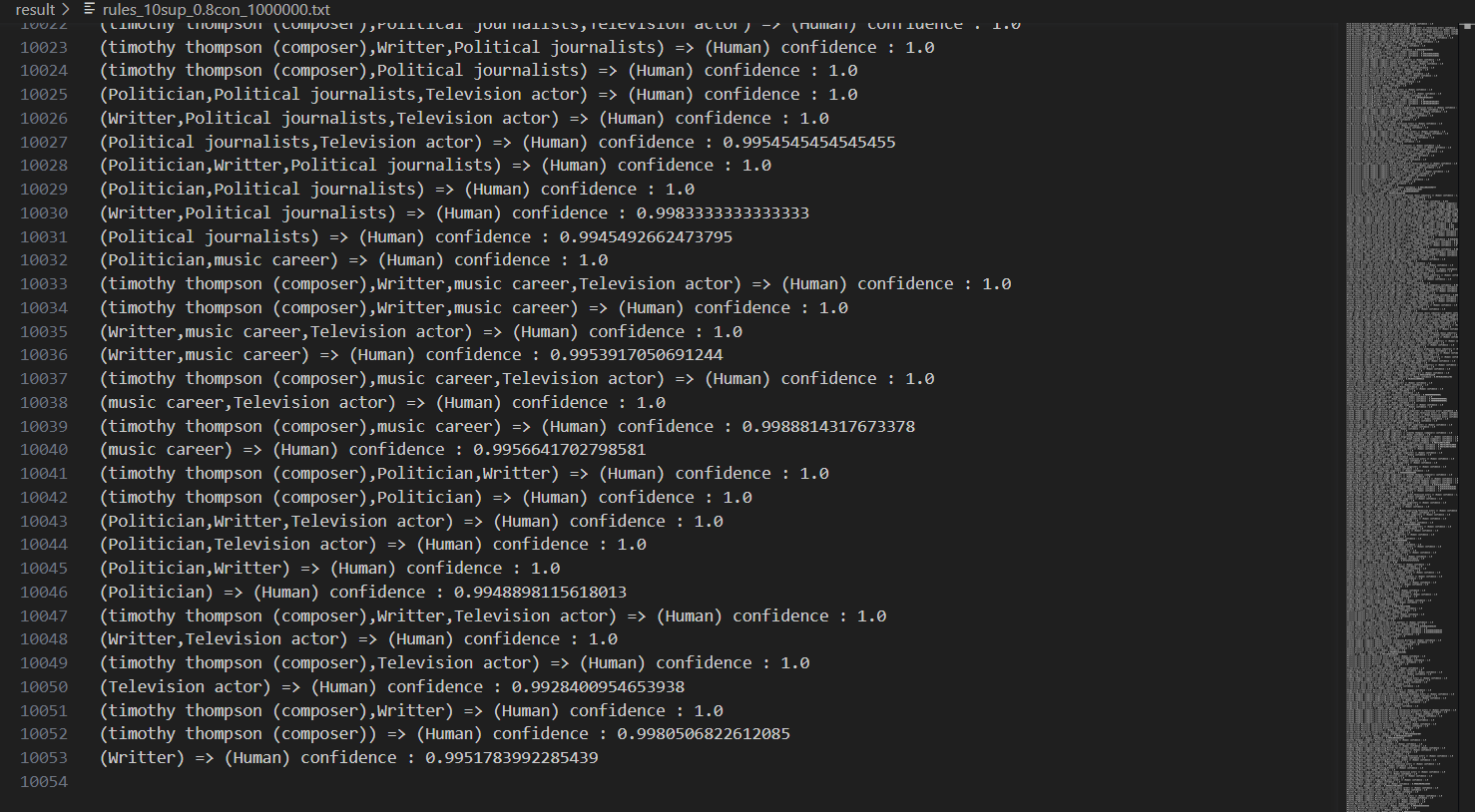




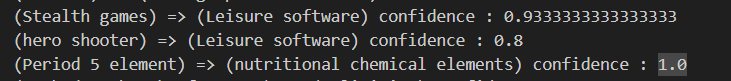
对前100W条数据进行该处理后得到了平均长度为5，最大长度为18的types + head\_关系record集，共有56869项record，对该数据使用FP-growth算法进行挖掘，生成的频繁项集仍非常可观，生成了上百万个频繁项集，对其进行规则挖掘仍是开销巨大的，因而从生成频繁项集数量的角度对其减小开销。对于这样的record的构成我们可以分为两种形式进行挖掘，第一种是仅对types项组成record集，因为types集普遍较短，不限制频繁项集最大长度的条件下，生成的频繁项集也不会过多，因此在types项上就可以对概念的从属或共现关系进行单独的挖掘，第二种是对types项和head\_关系或tail\_关系组成record，在使用FPgrowth算法进行挖掘时限定最长频繁项集为2，仅挖掘type -> h\_关系的规则。

在对types类型进行单独挖掘时得到的结果如下：

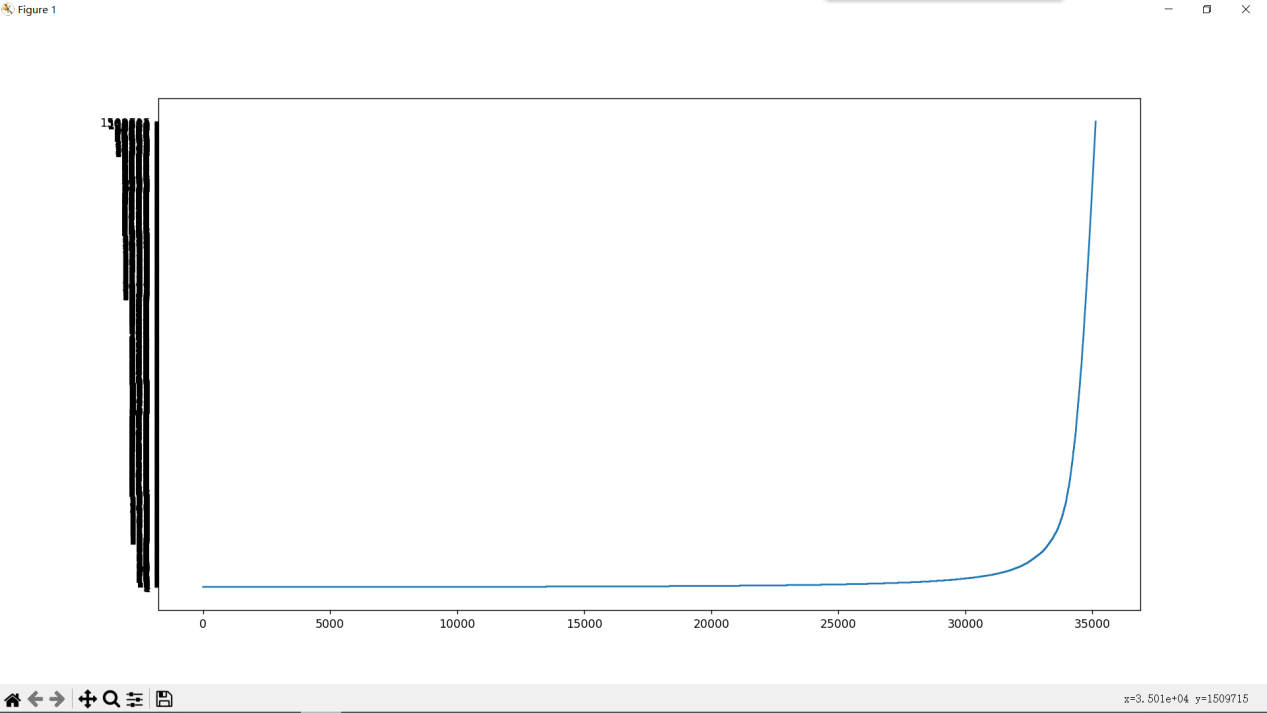




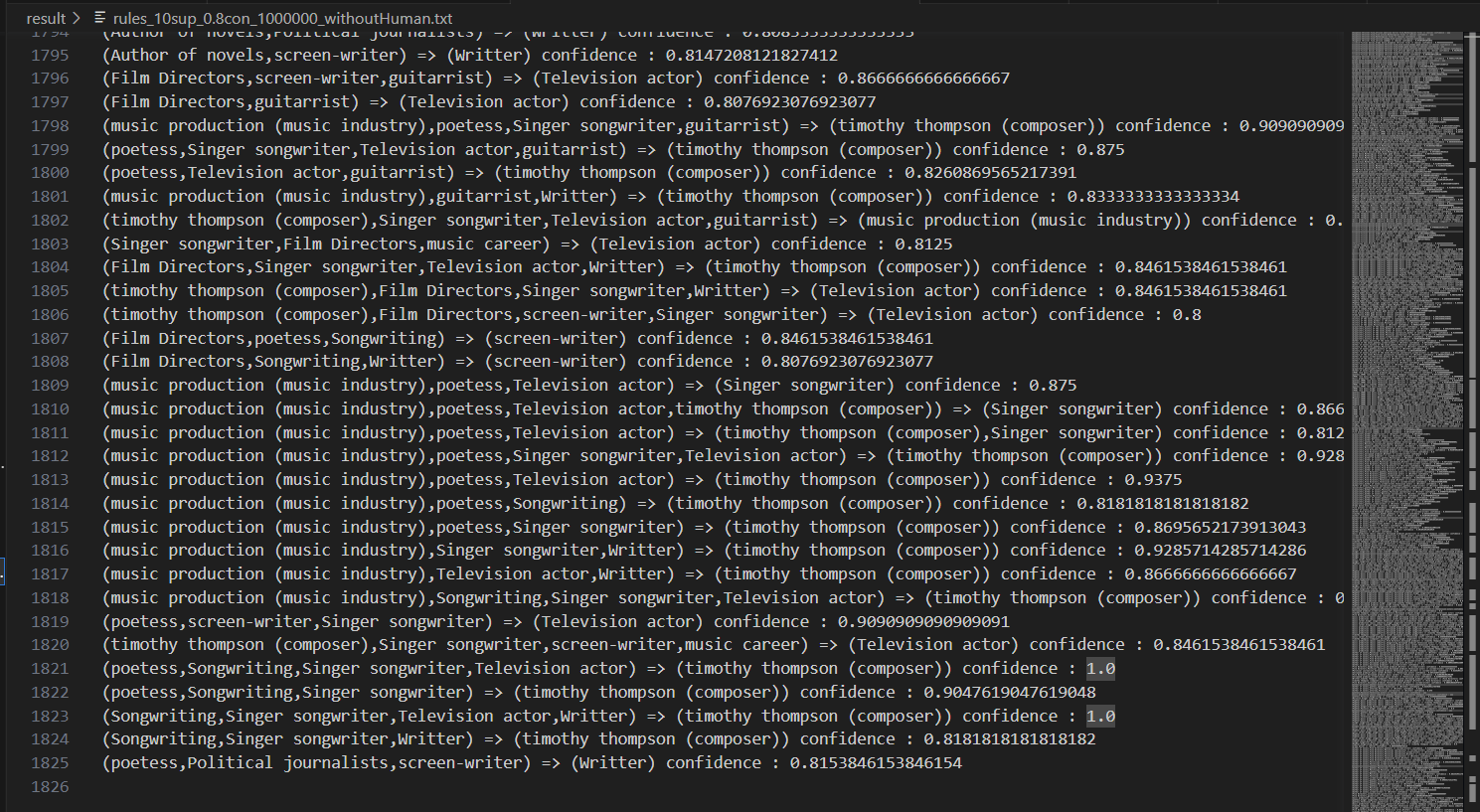
前100w数据对应types的record，使用min\_sup = 10,min\_con = 0.8的FPgrowth算法得到了如上图结果，共10053条规则，可以观察到如(Unix-like) => (operating system)这样的从属关系，以及大量的(职业)=>(human)的从属关系，还有类似下图中



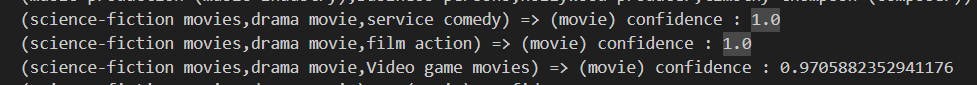
休闲软件的从属关系，化学元素的从属关系等，但是在上述结果中存在着大量的推出”Human”的规则，原因是在types中”Human”的出现频率非常之高，共计出现了1509715次，而大部分(2w+的type)的types仅出现了不到10次，总共也仅有35000+种type。



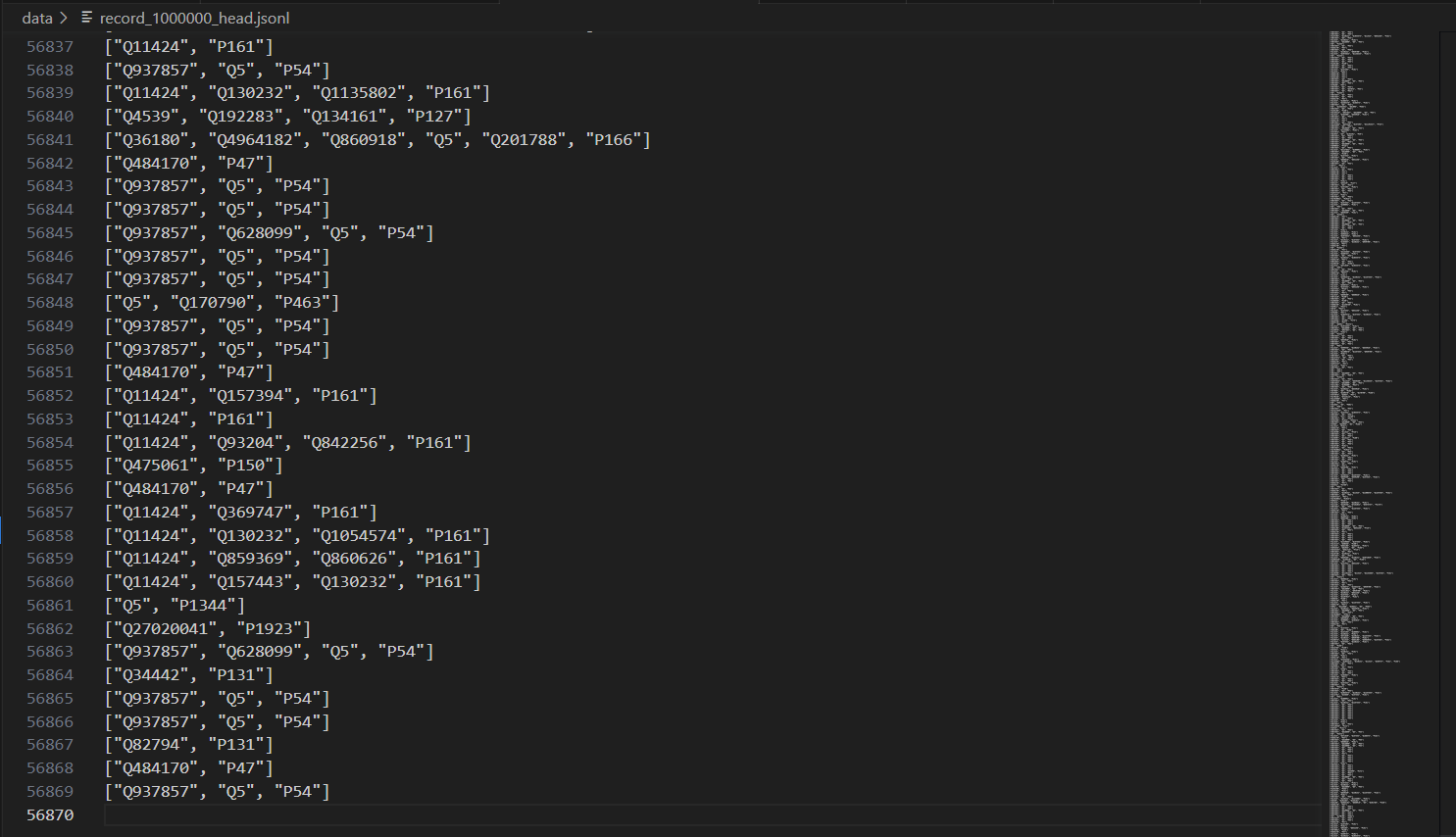
因此在对type类型进行频繁模式挖掘时出现了大量的与”Human”相关的规则，而这样的规则在概念上又显得比较宽泛，并不能提供太多的元知识内容，因此在去除”Human”这一type后再次对前100W条数据进行了一次挖掘。



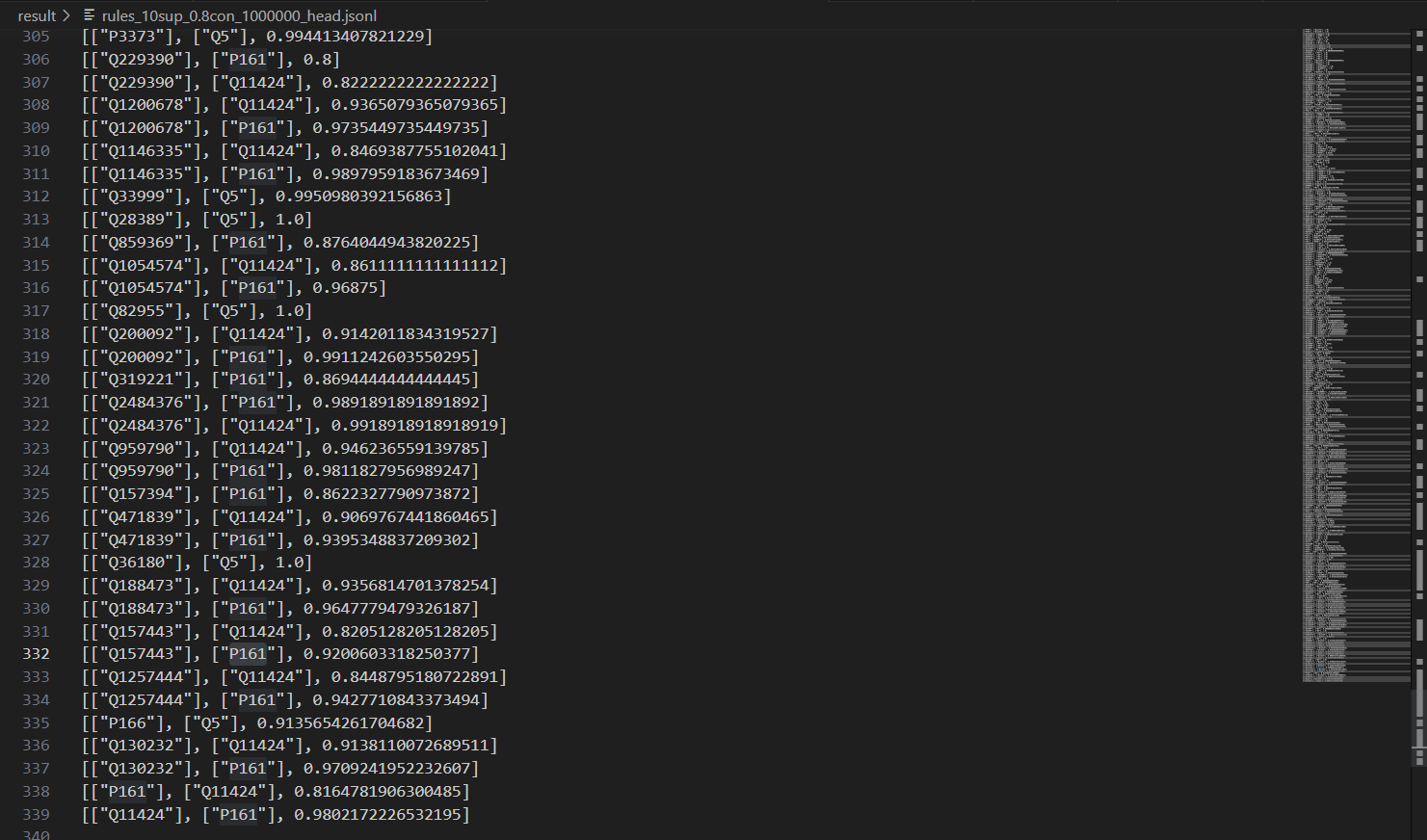
这次仅剩1825条规则，但从结果而言，其提供的元知识内容从总体上看质量更高。

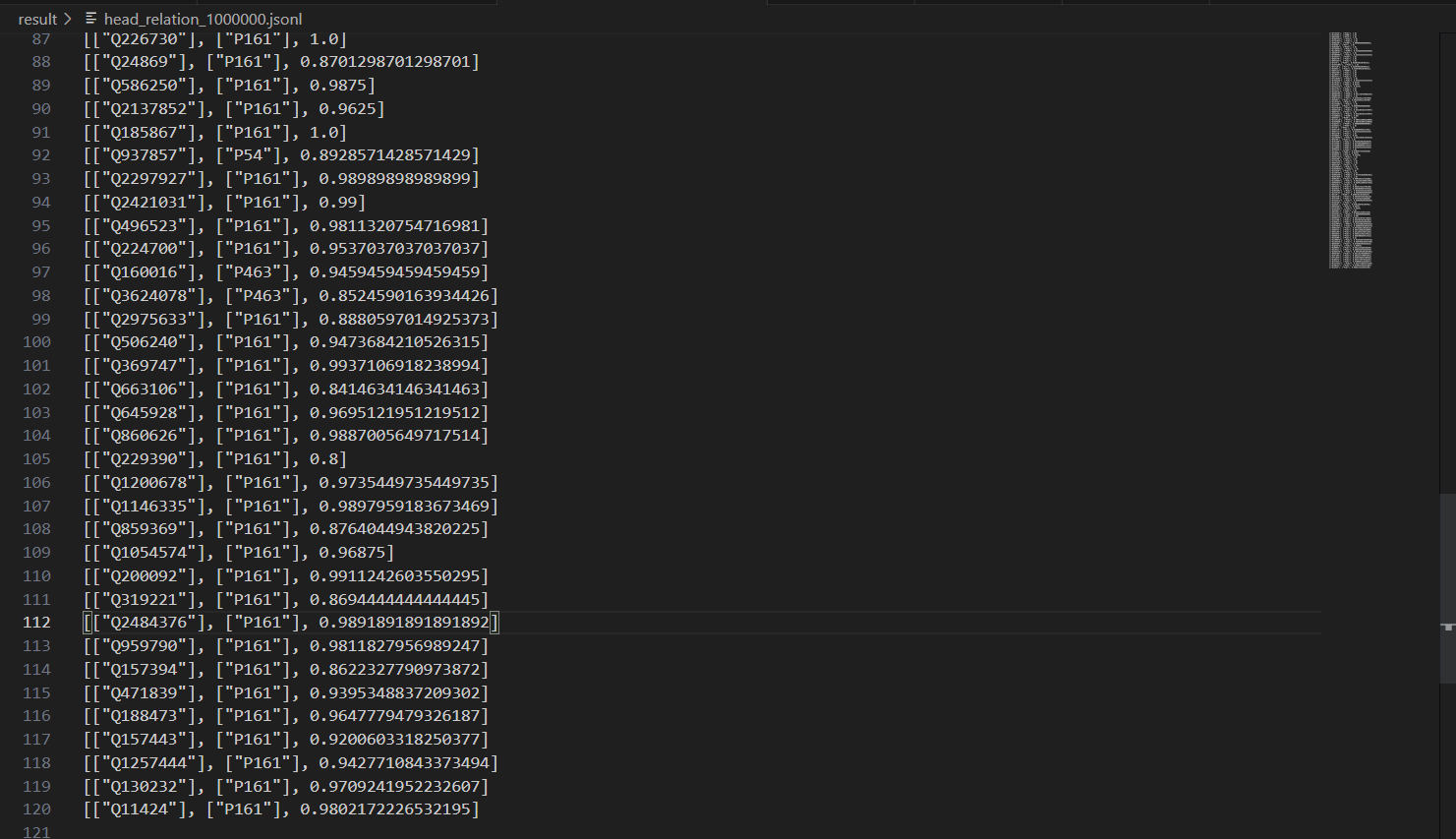


接下来对types + h\_关系组成的record进行频繁模式挖掘，前100W条记录生成了56869条record。

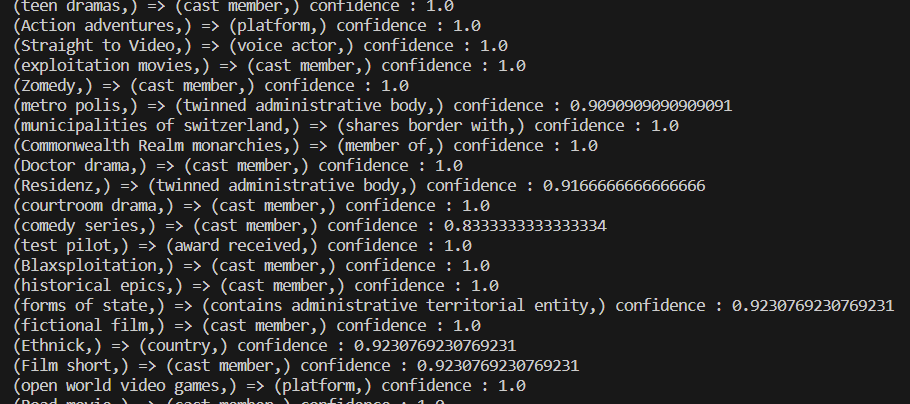


在min\_sup = 10, min\_con = 0.8, max\_freq\_len = 2的参数下产生了300条规则，再细化提取(QXXXX) => (PXXX)的规则时仅剩下120条规则。





转换为对应实体后结果如下：

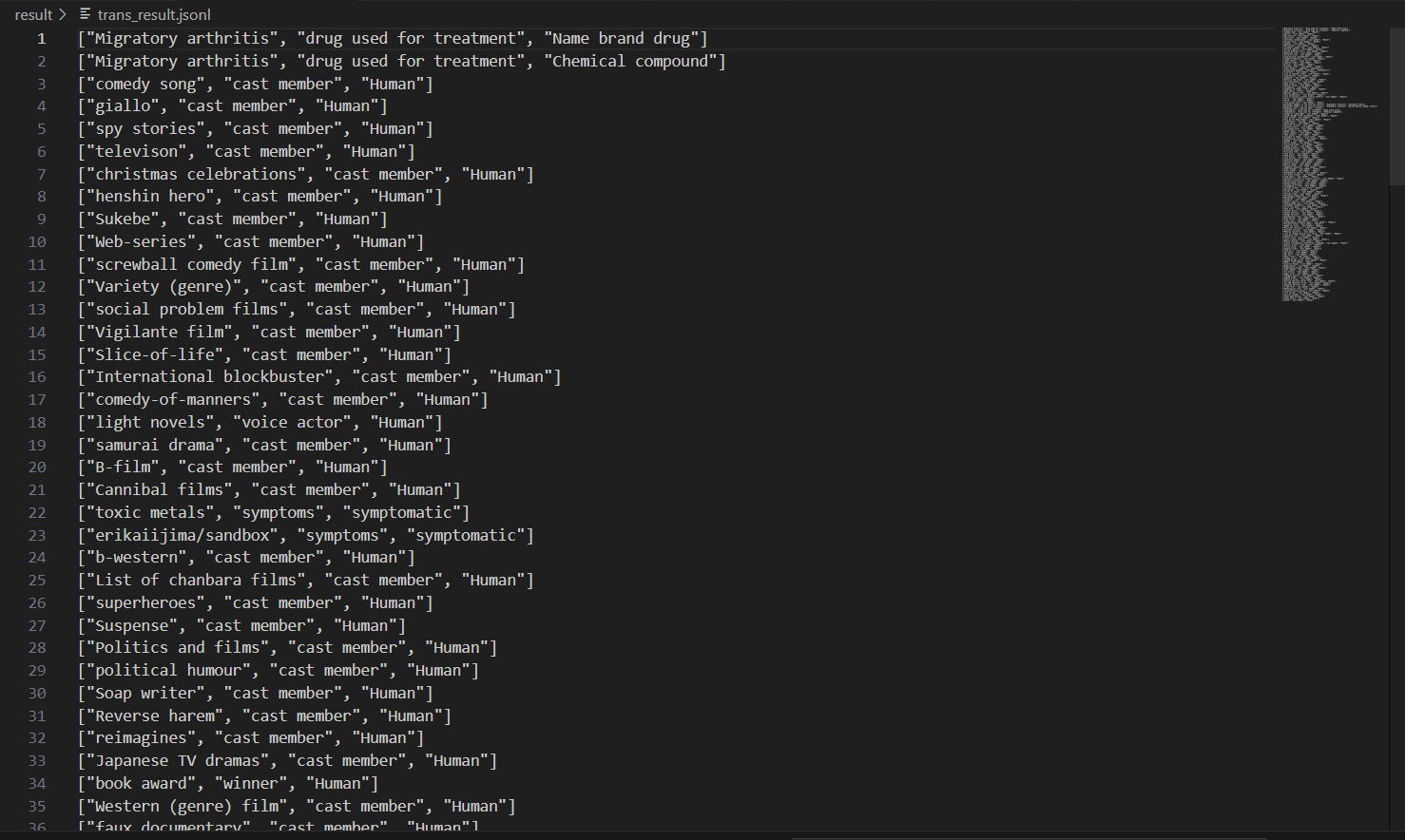


挖掘的效果并不理想，仅有少量有价值的关系，如



但是由于关系项太过稀少，在对types + tail\_关系record同样进行如此挖掘后，并不能连接上多少完整的三元组。

因此将min\_sup降至3再进行挖掘，此次得到规则数约是之前的2倍，对该规则进行连接后得到结果如下：



得到了一些具有一定意义的元路径：

 [“移行性关节炎”、“治疗药物”、“名牌药物”]

[“迁移性关节炎”、“用于治疗的药物”、“化合物”]



[“安全理事会常任理事国”、“外交关系”、“主权国家”]

[“安全理事会常任理事国”、“外交关系”、“联合国会员国”]

等元路径，但是本次挖掘得到的大部分元路径都是关系项为cast member, tail为Human的较为宽泛的概念，挖掘到的元知识内容并不多，需要对参数（如第一遍筛选时关系项的阈值，以及FPgrowth算法的参数）再进行调整挖掘知识元路径。