1. **选题：不能直接用5G承载网背景，需要更换背景，可参考《基于人工智能的网络告警智能诊断系统》，弱化承载网背景**
2. **告警数据集：**

表名：t\_alarmloghist\_xxxxx，后缀无关紧要，几个表可以随机抽出数据分成训练集和测试集

表中一行数据代表一条告警，不同列代表不同属性，列字段说明见下面表格：（目前我觉得重要的字段已经加粗）

每个设备有自己的编号（对应**cneid**），一个设备上有多个线路端口（对应**clineport**）；告警定位信息（**clineport**）可以定位到设备的具体端口，不过是用文字描述的，从文字中可以提取出设备的空间信息；设备告警时会记录告警时间（**coccurutctime**）、设备告警类型（**calarmcode**）、设备告警等级（**calarmlevel**）。



图1 设备的定位信息（clocationinfo）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 描述 | 数据  类型  （old） | 数据类型  （new） | NULL | PK | 备注 |
| clogid | 日志ID | int |  | N | Y |  |
| clstid | 告警列表ID | int |  | N |  |  |
| cprelogid | 本条日志的前一条告警日志 | int | int | N |  |  |
| crelatedlogid | 相关告警ID | int | int | N |  |  |
| cbussinessid | 相关业务ID | int | int |  |  |  |
| cobjectid | 告警对象ID | int | int | N |  |  |
| cobjectline | 告警盘线路号 | int | int | N |  |  |
| **calarmcode** | 告警代码 | int | int | N |  |  |
| calarminfohash | 告警信息Hash值 | int | int | N |  |  |
| **cneid** | 影响网元ID | int | int | N |  |  |
| cnetype | 网元类型 | int | int | N |  |  |
| cobjectidtype | 告警对象ID类型 | int | int | N |  |  |
| cdevicetype | 设备类型 | int | int |  |  |  |
| calarmcatalog | 告警类别 | int | int |  |  |  |
| calarmkind | 告警类型 | int | int |  |  |  |
| **calarmlevel** | 告警级别（3紧急，2重要，1次要，0提示） | int | int |  |  |  |
| cconfirmtype | 确认类型 | int | int |  |  |  |
| ccleartype | 清除类型 | int | int |  |  |  |
| **coccurutctime** | 设备报告告警发生标准时间 | datetime | int |  |  |  |
| crecvoccurutctime | 告警发生网管接收标准时间 | datetime | int |  |  |  |
| cclearutctime | 设备报告告警清除标准时间 | datetime | std::string |  |  |  |
| crecvclearutctime | 告警清除网管接收标准时间 | datetime | std::string |  |  |  |
| cconfirmutctime | 告警确认网管标准时间 | datetime | std::string |  |  |  |
| cconfirmuser | 确认用户 | char(20) | std::string |  |  |  |
| cclearuser | 清除用户 | char(20) | std::string |  |  |  |
| **clocationinfo** | 定位信息 | varchar(255) | std::string |  |  |  |
| **clineport** | 线路端口字符串 | varchar(255) | std::string |  |  |  |
| calarminfo | 设备报告的告警信息（小于255B） | varchar(255) | std::string |  |  |  |
| cremark | 用户备注 | varchar(1000) | std::string |  |  |  |
| calarmexinfo | 设备报告的动态附加信息 | varchar(2000) | std::string |  |  |  |
| ccustominfo | 用户信息 | varchar(255) | std::string |  |  |  |
| corialarmcode | 原始告警码 | int | std::string |  |  |  |
| cboardtype | 单盘类型 | int | std::string |  |  |  |
| creservedint | 保留字段 | int | int |  |  |  |
| creserved | 保留字段 | char(1) | int |  |  |  |
| contmac | ONT MAC地址 | char(20) | std::string |  |  |  |
| contpwd | ONT密码 | varchar(128) | std::string |  |  |  |
| clogicalsn | 逻辑SN号 | char(24) | std::string |  |  |  |
| clogicalpwd | 逻辑密码 | varchar(128) | std::string |  |  |  |

请参考表内容查看字段含义。关于每一种告警类型分别是什么告警，后续在制作演示系统时需要提供。

1. **我们要干什么**

根据这些网络告警数据，找出其中潜在的规则与模式，利用这些模式对真正重要的告警数据进行筛查，帮助运维人员更方便、准确的处理告警，最终制作一个演示系统。

例如，根据历史数据找出了一种告警的模式：a很大概率会导致b的发生。于是当a发生时，系统会对告警b进行预警。

再例如，在上面的基础上，又从历史数据中找出了一种告警模式：b和c大概率会同时发生，于是当a发生时，不仅要对b进行预警，还要对c进行预警

中间可能需要使用到的知识：关联分析、序列预测

1. **相关知识**

**4.1关联分析**

**4.1.1 频繁项集挖掘、频繁序列挖掘、关联规则挖掘、序列规则挖掘**

关联分析是在大规模数据集中寻找关联关系的任务。这些关系可以有两种形式，它们是2种递进的抽象形式，并且前者是后者的抽象基础。

代表共现关系的**频繁项集**：频繁项集（frequent item sets）是经常出现在一块儿的物品的集合，它暗示了某些事物之间总是结伴或成对出现。本质上来说，不管是因果关系还是相关关系，都是共现关系，所以从这点上来讲，频繁项集是覆盖量（coverage）这个指标的一种度量关系。

例如，{a,b,c}是一个频繁项集，a发生了b和c大概率也会同时发生。

代表因果/相关关系的**关联规则**：关联规则（association rules）暗示两种物品之间可能存在很强的关系，它更关注的是事物之间的互相依赖和条件先验关系。它暗示了组内某些属性间不仅共现，而且还存在明显的相关和因果关系，关联关系一种更强的共现关系。所以从这点上来将，关联规则是准确率（accuracy）这个指标的一种度量关系。

例如，{a,b} -> {c}置信度=70%，支持度=30；这样的一条规则说明a和b已经发生的情况下，c发生的概率是70%，这样的情况总共出现了30次（有关置信度和支持度具体概念请看下一小节的链接）

频繁项集与频繁序列：频繁项集是经常同时发生的项目集合（没有时间先后关系），频繁序列则是经常发生的一个项目的序列（有时间先后关系）。

关联规则与序列规则：关联规则与频繁项集分别对应序列规则与频繁序列。

**4.1.2项与项集、支持度、置信度、提升度等小概念**

参考https://zhuanlan.zhihu.com/p/118439868，相关资料多自行搜索、

**4.1.3 相关算法**

Apriori、FPGrowth（频繁项挖掘）

GSP、Prefixspan（频繁序列挖掘）

其他数据挖掘算法：

https://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php

**4.2 序列预测**

顾名思义就是对序列数据进行预测，例如对已经训练的模型输入a,b,c，它输出预测结果d。

相关算法：机器学习算法如强化学习、有向图或者深度学习LSTM等算法。

这一部分我们还没开始，我个人想试试CPT算法。

1. **演示模型的参考**

（1）按告警的时间顺序，模拟实时告警。

后端利用告警数据训练序列预测模型。

网页端点击一个按钮，就产生一条告警，后端接受到告警数据后进行预测，同时返回前端预测结果，提醒用户需要对预测结果中的告警进行预防。

1. 同样也是按告警的时间顺序，模拟实时告警。

网页端选择产生告警的速度

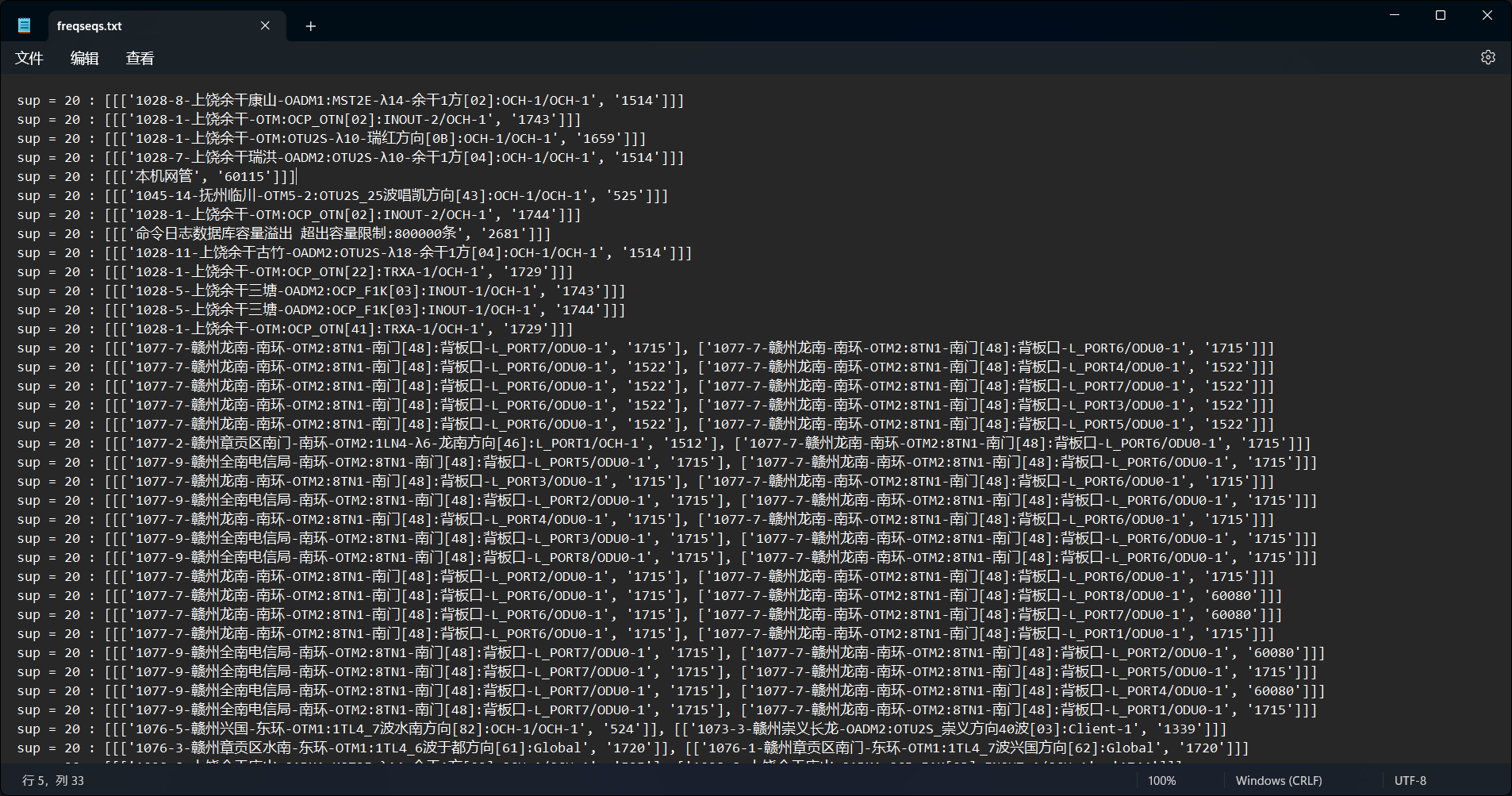
后端检测一段时间内告警的阈值，如果超过了阈值考虑是产生了“告警风暴”现象。大量告警数据对运维人员很不友好，最好只先显示有差异性的告警。于是根据频繁项集对告警数据进行分类，每一个频繁项集只挑出一个返回给网页端（当然，如果能挑出其中最重要的、导致其他告警发生的最根本告警更好）。

网页端最终显示的是具有差异性、根因不相同的告警。

1. 如果上面两个演示效果不好，也可以加一点统计相关内容，例如统计每个设备发生了多少次告警，显示发生的最多的几个；显示出现次数最多的告警类型，并且给出相应的处理方法........

总之要把演示模型做好看一点

1. **频繁序列展示**



在挖掘算法方面我是直接用的spmf的库，你们可能需要自己实现一下。

数据处理流程可以参考我的代码（因为一直在我本地电脑上写，没有进行版本管理，加上很多修改，非常丑陋）和ppt