**Steam游戏评价爬取工具技术文档**

**一、项目概述**

本项目针对Steam平台用户评价“滚动加载、无显式分页”的特性，通过接口分析、反爬突破与结构化解析技术，实现全量评论的自动化爬取。工具可将非结构化HTML数据转化为JSON/Excel格式，支持大规模数据获取（实测连续爬取10000条评论无反爬拦截），为游戏评价分析、用户反馈挖掘提供技术支撑。

**二、核心技术实现细节**

**1. 评论接口定位与分页游标机制破解**

Steam评论数据通过异步接口动态加载，其分页逻辑依赖“游标参数”实现，具体技术解析如下：

**（1）接口定位原理**

打开目标游戏评论页（如https://store.steampowered.com/app/{appid}/reviews/）后，浏览器通过“Fetch/XHR”请求从后端获取评论数据。通过开发者工具捕获的首个接口URL（形如https://store.steampowered.com/appreviews/{appid}?p=1&userreviewsoffset=0&...）包含初始分页参数，返回内容为嵌套评论数据的HTML片段——这是爬取的起点。

**（2）分页游标提取逻辑**

游标是实现连续爬取的核心标识，其提取过程为：

* 解析当前页HTML内容，通过字符串定位找到包含游标值的标签：在HTML源码中搜索name="userreviewscursor" value="，该字符串后紧跟的内容即为游标值，直至下一个双引号（"）结束。例如，若HTML中存在name="userreviewscursor" value="abc123"，则提取的游标值为abc123。
* 游标值具有唯一性和时效性，每一页的游标仅对应下一页接口，无法提前预测，必须从当前页HTML中实时提取。

**（3）下一页URL构造规则**

基于当前页URL和提取的游标，通过参数迭代生成下一页URL，核心逻辑为：

* 从当前URL中解析出基础参数（如appid、l（语言）、browsefilter（排序方式）等）；
* 更新分页参数：页码（p）递增1，偏移量（userreviewsoffset）递增10（因每页固定返回10条评论），游标（userreviewscursor）替换为新提取的值；
* 保持其他固定参数（如numperpage=10、forceanon=1）不变，重新拼接URL。

通过上述逻辑，工具可从首个接口出发，循环提取游标→更新参数→生成新URL，实现无人工干预的连续爬取。

**2. 反爬机制突破策略与技术实现**

Steam通过检测异常请求特征（如固定User-Agent、高频访问、非浏览器行为）识别爬虫，工具通过多层级技术手段规避拦截，具体实现如下：

**（1）动态User-Agent模拟**

使用fake\_useragent库生成随机浏览器标识，每次请求前更新请求头：

* 库内置主流浏览器（Chrome、Firefox、Safari等）的User-Agent模板，通过ua.random方法随机调用，模拟不同设备和浏览器的访问特征；
* 避免因固定User-Agent被Steam后端标记为“异常请求源”。

**（2）请求频率控制与延迟优化**

* 基础延迟：每次请求前设置2-5秒随机延迟（通过random.uniform(2,5)实现），模拟人类手动滚动页面的间隔；
* 动态调整：若检测到潜在反爬信号（如响应速度骤降、HTML内容异常简短），自动将延迟延长至10-20秒，降低触发频率限制的风险。

**（3）反爬页面识别与重试机制**

* 反爬特征检测：通过关键词匹配识别反爬页面——在返回的HTML中搜索“access denied”“robot check”“captcha”“cloudflare”等特征字符串，若命中则判定为反爬拦截；
* 阶梯式重试：触发反爬后，首次重试延迟10-20秒，二次重试延迟30-60秒，最多重试3次，避免短期内重复触发拦截规则；
* 状态码适配：针对429（请求过多）、403（禁止访问）等特定状态码，单独设置延长等待时间（429对应30-60秒，403对应20-40秒），提升重试成功率。

**（4）SSL验证与会话保持**

* 关闭SSL严格验证（session.verify = False）：避免因Steam部分节点证书配置问题导致的请求失败；
* 会话保持（requests.Session()）：复用TCP连接，模拟浏览器持续会话状态，减少因频繁建立连接被识别为爬虫的概率。

上述策略组合实现了对Steam反爬机制的有效规避，实测可支持连续爬取10000条评论（1000页）而无拦截。

**3. HTML解析与结构化数据提取**

接口返回的HTML为非结构化数据，需通过精准解析提取评论字段，核心技术逻辑如下：

**（1）评论容器定位**

使用BeautifulSoup库解析HTML，通过find\_all('div', class\_='apphub\_Card')定位所有评论容器——每个apphub\_Card类的div标签对应一条独立评论，包含该评论的全部信息（用户、内容、评分等）。

**（2）核心字段提取逻辑**

* **推荐状态**：通过评论容器内的“点赞/点踩”图标判断——查找src属性包含“icon\_thumbs”的img标签，若src中含“thumbsUp”则为“推荐”，含“thumbsDown”则为“不推荐”；
* **用户名称**：定位class="apphub\_CardContentAuthorName"的标签，提取其下a标签中链接的文本内容（如<a>Player123</a>→“Player123”）；
* **评论内容**：定位class="apphub\_CardTextContent"的标签，先移除其中class="date\_posted"的子标签（包含发布时间），再提取剩余文本作为纯评论内容；
* **发布日期**：从上述date\_posted子标签中提取文本，去除“发布于：”前缀后保留原始日期格式；
* **游戏时长**：直接提取class="hours"标签的文本（如“120.5小时”）；
* **有价值人数/欢乐人数**：解析class="found\_helpful"标签的文本，通过正则表达式（r'有\s\*(\d+)\s\*人觉得这篇评测有价值'和r'有\s\*(\d+)\s\*人觉得这篇评测很欢乐'）提取数字；
* **奖励数量/评论数量**：对class="review\_award\_aggregated"和class="apphub\_CardCommentCount"标签的文本，通过re.findall(r'[\d,]+')提取数字并去除千位分隔符（如“1,234”→1234）。

通过上述标签定位与文本解析逻辑，工具可将HTML中的非结构化信息转化为包含9个字段的结构化字典，最终聚合为JSON/Excel格式输出。

**三、操作流程与技术参数**

**1. 前置准备**

* 网络加速：启动Steam++加速Steam商店，确保可稳定访问https://store.steampowered.com；
* 环境配置：安装Python 3.6+及依赖库，开发者使用版本为Python3.12.3（pip install requests beautifulsoup4 pandas fake\_useragent openpyxl）。

**2. 爬取执行步骤**

1. **获取起始接口URL**：进入目标游戏评论页→F12打开开发者工具→Network→Fetch/XHR→下拉页面加载评论→复制首个接口的Request URL；
2. **运行爬取脚本**：执行get\_info.py，输入起始URL、爬取页数（1页对应10条）、保存目录，脚本自动按分页逻辑连续爬取并保存HTML文件（命名格式page\_1.html至page\_N.html）；
3. **提取结构化数据**：执行extract\_data.py，输入HTML保存目录，脚本遍历文件解析评论，输出JSON（{目录名}\_dataset/）和Excel（{目录名}\_dataset\_xls/）。

**四、技术细节与扩展说明**

**参数可配置性**：默认爬取“简体中文+好评排序”评论，若需调整，可在extract\_next\_url函数中修改l（语言，如english）和browsefilter（排序，如recent为最近评论）参数；

**五、合规性与免责声明**

* 遵守Steam的robots.txt协议，不爬取Disallow字段限制的路径；
* 数据仅用于技术研究，严禁商业使用或侵犯用户隐私；
* 使用者需自行承担IP封禁等风险，开发者不承担连带责任。

技术交流:3351515598@qq.com